

Konzept und prototypische Implementierung eines erweiterbaren und modularen Simulations-Frameworks für eine Spiel-Plattform zur Ausbildungsbegleitung in der Bauphysik

DISSERTATION

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

an der Fakultät Bauingenieurwesen

der

Bauhaus-Universität Weimar

vorgelegt von

Dipl.-Wirtsch.-Ing. (FH) Heinrich Söbke M. Comp. Sc.

geb. am 15.07.1968 in Gronau/Westf.

Weimar, 28. März 2013

Meiner Schwester Maria gewidmet.

Danksagung

Dass der langwierige Prozess, der zur vorliegenden Arbeit geführt hat, zum guten Schluss doch abgeschlossen wurde und nicht gescheitert ist, ist keine Selbstverständlichkeit. Es ist das Ergebnis vieler günstiger Umstände und der Begegnung mit einer Vielzahl von Menschen, die alle auf ihre Weise zum Gelingen der Arbeit beigetragen haben. Diese Arbeit ist der fassbare Teil einer für mich bemerkenswerten persönlichen Weiterentwicklung sowohl in fachlicher als auch in menschlicher Hinsicht.

Mein besonderer Dank geht an Prof. Dr. Oliver Kornadt, der mir diese Arbeit erst ermöglicht hat durch seine Entscheidung, mich in das Projektteam aufzunehmen. Für die wissenschaftliche Betreuung bin ich ihm ebenfalls zu größtem Dank verpflichtet: Sein Blick für das Wesentliche trug in zahlreichen fachlichen Diskussionen in höchstem Maße zu einer stetigen Fortentwicklung der Arbeit bei. Mit seiner Erfahrung hat er mir sehr geholfen, die richtigen Dinge zu tun.

Mein Forschungsaufenthalt an der University of Wisconsin - Madison ist ebenfalls auf die Initiative von Prof. Kornadt zurückzuführen. Wenn ich vielleicht noch nicht restlos überzeugt war vom Potential des spielerischen Lernens, dann wurde ich dort endgültig mit der Leidenschaft für dieses Thema infiziert: Bedanken möchte ich mich bei Prof. Kurt Squire, Ph.D., Prof. Constance Steinkuehler, Ph.D. und Prof. Rich Halverson, Ph.D. sowie bei Susan Millar, Ph.D., die mich bereitwillig aufnahmen, mir viele wertvolle Anregungen gegeben haben und geben. Sie haben das unvergessliche Erlebnis ermöglicht, ein Teil des mit modernsten Technologien und Konzepten arbeitenden, brandneuen Morgridge Institute for Research zu sein. Genauso möchte ich den weiteren Mitgliedern des ERCA-Teams - namentlich genannt seien hier Nathan Patterson, Ph.D., Kevin Harris, Michael Beal, Shannon Harris, Sarah Chu, Matt Gaydos, und Javier Corredor, Ph.D. sowie Ben Shapiro, Ph.D. - für ihre Hilfsbereitschaft und ihre große Geduld bei der spielerischen Heranführung an das Thema danken.

Zurück in Thüringen haben auch Prof. Dr. Helmut Niegemann und Prof. Dr. Klaus Peter Jantke durch ihre Bereitschaft zu Diskussionen und mit ihren Hinweisen Anteil am Gelingen der Arbeit gehabt. Prof. Dr. Jörg Londong konnte zusätzliche Perspektiven für die Anwendung des errungenen und erspielten Wissens aufweisen. Dafür sei ihnen allen recht herzlich gedankt.

Allen Kollegen am Lehrstuhl Bauphysik der Bauhaus-Universität Weimar – namentlich Jörg Arnold, Karin Gorges, Markus Hofmann, Thomas Lichtenheld, Aner Martinez, Dr. Richard Rudolph, Jens Schmidt, Venanda Sulistya, Albert Vogel und Dr. Conrad Völker - möchte ich ebenso herzlich für das angenehme Arbeitsklima sowie die unzähligen, geduldigen Hilfen danken. Ein großer Dank gilt auch meinen beiden außergewöhnlichen Projektkollegen Christiane Hadlich und Thomas Bröker – mit ihnen fühlte sich Arbeit nicht als solche an.

Ohne die tatkräftige Unterstützung meiner hilfswissenschaftlichen Mitarbeiter wäre diese Arbeit nicht möglich gewesen. Daher möchte ich mich bei Janina Held, Anna Beyer, Adriana Cabrera, Daniel Schwarz und Philipp Blaschke für die ideenreiche Hilfe bei der Entwicklung der Spielplattform bedanken. Ebenso konnte ich sehr von Abschlussarbeiten am Lehrstuhl Bauphysik profitieren, deren Erstellungsprozess ich betreuen durfte. Dafür danke ich Janina Held, Christin Kühnert, Linda Luthardt, Philipp Schürenberg, Naira Müller, Mario Aubel, Maria Borowski und Carolin Franke. Auch die vielen Teilnehmer studentischer Projekte haben mit ihrem Arbeitsprozess und

ihren Ergebnissen zum Gelingen dieser Arbeit beigetragen – namentlich genannt sei das „Residential Lab“-Team bestehend aus Christian Baar, Marcel Döpel und Oliver Herterich, die verantwortlich sind für die im Titelbild dieser Arbeit gezeigte Spielszene.

Die Arbeit wurde erst ermöglicht durch das Förderprogramm „InnoProfile“ des Bundesministeriums für Bildung und Forschung. Ich danke den Verantwortlichen für die Gelegenheit, das Thema ohne direkte wirtschaftliche Zwäng aus mehr als den üblichen Perspektiven betrachten zu dürfen und bin zuversichtlich, dass es sich in nicht allzu ferner Zukunft gelohnt haben wird.

Herzlich bedanken möchte ich mich auch bei meiner Familie und meinen Freuden, die ausnahmslos fehlende Zeit mit viel Geduld und großer Rücksicht ertragen haben. Ohne ihre andauernden und entschiedenen Ermutigungen wäre diese Arbeit nicht zustande gekommen. Allen weiteren, die direkt oder indirekt zur vorliegenden Arbeit beigetragen haben und nicht namentlich an dieser Stelle erwähnt werden (u.a. weil ich trotz aller theoretischen Kenntnisse zum Thema Lernen doch auch vergesse), sei ebenso mein Dank ausgesprochen. Zuletzt möchte ich meinen Dank für die stetige Aufmunterung und Unterstützung und die noch größere Geduld an Andrea senden: Ohne Dich gäbe es für mich und die Welt – trotz Computerspielen ;-) – viel weniger Sonne.

Ich bin dankbar und empfinde die Möglichkeit zu dieser Arbeit als Teil eines Glückspilzdaseins.

Inhaltsverzeichnis

Prolog	5
1 Einführung	7
1.1 Problemstellung	7
1.2 Idee und Methodik	8
1.3 Aufbau der Arbeit	10
2 Stand des Wissens	13
2.1 Spielbasiertes Lernen	13
2.1.1 Das Medium „Computerspiel“	13
2.1.2 Motivation als Entwurfsziel	18
2.1.3 Social Network Games	19
2.1.4 Gamification	21
2.2 Simulation in der Bauphysik	23
2.2.1 Softwarepakete	23
2.2.2 Stationäre und instationäre Prozesse	24
2.3 Instructional Design	24
2.3.1 Lernziele und Lernzieltaxonomien	25
2.3.2 Lernzieltaxonomie nach Bloom	25
2.3.3 Component Display Theory (CDT)	27
2.3.4 Four-Component Instructional Design Model (4C/ID)	27
3 Spielplattform	28
3.1 Funktion	28
3.2 Spielkonzept	28
3.3 Systemaufbau	31
3.3.1 Architektur	31
3.3.2 Software	35
4 Simulationskern	40
4.1 Design-Grundsätze	40
4.1.1 Modularität	40
4.1.2 Erweiterbarkeit	41
4.2 Ergebnisorientierte Implementierung des Prototyps	45
4.2.1 Anforderungen an den Szenario-Editor	45
4.2.2 Spezifikationsorientierte Implementierung	45
4.3 Simulation: Objektmodell	47
4.3.1 Entity: SimuElement	48
4.3.2 Property: Parameter	48
4.3.3 Action	49
4.3.4 Event	50
4.3.5 Commands	51
4.4 Konzept weiterer Frameworkelemente	51
4.4.1 User Manager (UM)	51
4.4.2 Skill- & Assessment Component (SAC)	52
4.4.3 Scenario Manager (SM)	53
4.4.4 Monitoring Component (MC)	53

4.4.5 Simulation Engine (SE)	54
5 Szenarien ausgewählter bauphysikalischer Probleme	59
5.1 Basis des Szenariodesigns	59
5.2 Szenariobeschreibung	59
5.2.1 Definition der Lernziele	61
5.2.2 Fähigkeiten	61
5.2.3 Basiskonfigurationen sowie Basis-, Ergänzungs- und Kombinationsszenarien	61
5.2.4 Beschreibungsschema einer Basiskonfiguration	62
5.2.5 Beschreibungsschema eines Kombinationsszenarios	63
5.2.6 Identifikatoren	64
5.2.7 Tags	64
5.2.8 Systemnotation	65
5.3 Übersicht der beschriebenen Szenarien	69
5.3.1 Basiskonfigurationen	69
5.3.2 Szenarien	69
5.3.3 Lernziele	69
5.3.4 Missionen	71
5.3.5 Fachliche Präzision	71
5.4 Basiskonfigurationen	72
5.4.1 Basiskonfiguration: Einraumhaus [BK001]	72
5.4.2 Basiskonfiguration: Vierwohnungsbungalow [BK002]	74
5.5 Szenarien	76
5.5.1 Basisszenario: Heizwärmebedarf in Abhängigkeit von der Fensterqualität [B001]	76
5.5.2 Ergänzungsszenario: Entwicklung der Transmissionswärmeverluste [E001]	81
5.5.3 Ergänzungsszenario: Solare Wärmegewinne [E002]	87
5.5.4 Basisszenario: Schallpegel in Abhängigkeit von der Fensterqualität [B002]	92
5.5.5 Ergänzungsszenario: Schalldämmung einer Außenwand [E003]	96
5.5.6 Ergänzungsszenario: Schalldämmung von Zwischenwänden [E004]	100
5.5.7 Kombinationsszenario: Schalldämmung und Wärmedämmung [K001]	105
5.5.8 Weitere Ansätze für Szenarien	106
6 Zusammenfassung und Diskussion	108
6.1 Ergebnisse	108
6.2 Bewertung und Thesen	109
Epilog	115
Tabellenverzeichnis	116
Abbildungsverzeichnis	117
Formelverzeichnis	118
Verzeichnis der Quellcodeauszüge	119
Verzeichnis der Pseudocodedarstellungen	120
Abkürzungen	121
Glossar	123
Übersetzungen	127
Ehrenwörtliche Erklärung	128
Referenzen	129

Anhang A: Übersicht ähnlicher Softwarepakete

Anhang B: Konstruktive Maßnahmen zur Verminderung des Software Aging

Anhang C: Grundlagen der Simulation

Anhang D: Fallstudie: Designanforderungen an bauphysikalische Simulationssoftware

Anhang E: Computerspiele als Lernmedium

Anhang F: Easy Gold - ein Minispiel in der Gemeinschaft

Anhang G: Plattformbeschreibung

Prolog

Ab und an ist man dermaßen überrascht, dass man den Umstand, der zu der Überraschung geführt hat, nicht mehr vergisst. Bei derartigen Erlebnissen lernt man in der Regel - sie werden daher auch Lernerfahrungen genannt. Eine solche hatte ich, als ein Bekannter erklärte, weshalb er ziemlich gut Englisch spricht: Er hat es sich beim Spielen angeeignet. In einem Mehrspieler-Online-Spiel gehörte er einer Gruppe an, in der die Kommunikation wegen internationaler Zusammensetzung der Spieler ausschließlich auf Englisch erfolgte. Durch die regelmäßige Notwendigkeit, sich im Spiel mit den anderen auszutauschen, konnte er seine Sprachkenntnisse verbessern. Er war entsprechend motiviert, zu allen möglichen anderen Gelegenheiten - natürlich auch in der Schule - die Chancen zum Lernen der englischen Sprache zu nutzen.

Bei einem Aufenthalt in den USA 2010 wurde ich anlässlich der Verabschiedungsfeier eines Kollegen in ein Gespräch über die europäischen Fußballligen verwickelt. Mein Gesprächspartner aus dem Land des American Football zeigte sich als profunder Kenner des europäischen Soccer: Er kannte die Top-Clubs der europäischen Ligen, sogar einige in der letzten Zeit nicht so stark im Rampenlicht stehende Vereine - aus Deutschland u.a. auch den 1. FC Kaiserslautern und Borussia Mönchengladbach. Er wusste auch Bescheid über die Schuldenproblematik der spanischen Primera División und der englischen Premier League. Er sah die Fußball-Bundesliga als eine der wirtschaftlich gesündesten Ligen in Europa. Er schwärmte von den Erfolgen des Dortmunder Trainers Jürgen Klopp. Erstaunt über das geballte Wissen, das mir genauso unerwartet wie fast unbegrenzt entgegenschlug, fragte ich ihn, weshalb er sich so gut auskenne. Seine Antwort war der Verweis auf ein Fußball-Manager-Spiel - eines der Computerspiele, die zu seinen beliebtesten Freizeitbeschäftigungen zählen.

TED (Technology, Entertainment, Design) ist eine Konferenzserie, bei der sich Fachleute unterschiedlicher Domänen austauschen. Einige sehr inspirierende Vorträge sind auch per Internet auf einer Video-Plattform zugänglich. Das erste von mir gesehene TED-Video war der Vortrag „Gaming can make a better World“ von Jane McGonigal¹, einer amerikanischen Game-Designerin. Sie nennt dort auch die Größe 10.000 Stunden - und zwar in drei Zusammenhängen: 10.000 Stunden sind die Zeitdauer, die ein Amerikaner bis zum 21. Lebensjahr mit Computerspielen verbracht hat. Die beiden anderen Zahlen geben dieser Zeitspanne eine für den Zuhörer fassbare Bedeutung: 10.080 Stunden ist die Zeit, die ein amerikanischer Schüler von der 5. bis zur 12. Klasse plangemäß im und mit dem Schulunterricht verbringt. McGonigal interpretiert, dass die Spiele einen kompletten zweiten Bildungsweg neben der Schule darstellen. Dann zitiert sie Malcolm Gladwell, der in seinem Buch „Überflieger“² die These vertritt, dass zur perfekten Beherrschung einer Fähigkeit („mastery“) diese 10.000 Stunden vor dem 21. Lebensjahr in das Erlernen jener Fähigkeit investiert sein müssen³. McGonigal wertet diese Aussage als Hinweis,

¹ http://www.ted.com/talks/jane_mcgonigal_gaming_can_make_a_better_world.html, letzter Zugriff am 10.09.2012

² Malcolm Gladwell (2009): Überflieger: Warum manche Menschen erfolgreich sind - und andere nicht

³ Diese von Ericsson, Krampe & Tesch-Römer (1993) beeinflusste These, die sie aus einer Studie mit Violinisten ableiten, steht stark in der Diskussion - auch aufgrund des Argumentes, dass es sich um eine Untersuchung aus der Rückschau handelt, die keine Aussage dazu macht, welche Maßnahmen in welchem Umfang notwendig sind, um eine bestimmte Fähigkeit zu erreichen.

dass mit dem betriebenen Aufwand durch Spiele und Spielen Bedeutsames geschaffen werden kann: Computerspiele haben einen großen Einfluss - er sollte genutzt werden.

Das sind drei Beispiele, in denen sich die Verbindung von Computerspielen und Lernen widerspiegelt, die auf den ersten Blick so nicht erkannt wird und in einer leistungsorientierten Gesellschaft traditionellen Ansätzen und Einschätzungen des Lernens entgegensteht. Das Versprechen des gefühlt mühelosen Lernens macht Computerspiele zu einem faszinierenden Medium, das - trotz aller zurzeit noch vorhandenen Unfassbarkeit und Unsicherheit - in der Lehre eingesetzt wird und werden kann. Mit der vorliegenden Arbeit möchte ich einen kleinen Beitrag leisten, die Nutzung von Computerspielen in der Ausbildung zielgerichteter, handhabbarer und planbarer zu machen.

*An interest is a terrible thing to waste.
(Roger C. Schank⁴)*

1 Einführung

1.1 Problemstellung

Die Bauphysik ist eine ingenieurwissenschaftliche Disziplin, die sich mit der Wirkung von physikalischen Phänomenen in und an Gebäuden beschäftigt. Als Ingenieurwissenschaft betrachtet und optimiert die Bauphysik Systeme. Beispielsweise müssen bei der Auswahl des Materials und der Stärke einer Wand unter anderem die Anforderungen des Wärmeschutzes und die des Schallschutzes mit den Behaglichkeitsanforderungen der Benutzer und den Wünschen nach geringen Betriebskosten sowie der Notwendigkeit von niedrigen Baukosten abgeglichen werden. Wie dieses Beispiel zeigt, sind die zu lösenden Probleme komplex. So widersprechen die Anforderungen an eine Lösung einander oft und es gibt in vielen Fällen keine eindeutige Lösung (Bröker & Kornadt, 2009). Dörner (1981, 2003) stellt fest, dass es dem Menschen im Allgemeinen schwerfällt, sich in komplexe Systeme einzudenken. Ein zeitlicher Versatz von Ursache und Wirkung ist eines der Merkmale, die ein System schwer handhabbar werden lassen. Dörner führt weiter aus, dass der Umgang mit Systemen durch die Nutzung von Simulationen trainiert werden kann. Durch die Abstraktion eines Systems in einer virtuellen Welt können Probleme modelliert und dem Lernenden visuell und akustisch dargestellt werden. Er ist nicht allein auf theoretische Überlegungen angewiesen, sondern kann das System miterleben. Mit Hilfe von Simulationen lassen sich „echte“ Erfahrungen gewinnen, jedoch ohne möglicherweise schädliche Auswirkungen im realen Leben hinnehmen zu müssen⁵.

Schell (2008, S. 48) bezeichnet Computerspiele als das derzeit umfassendste Medium: Sie können alle bisherigen Medien wie Texte, Animationen und Tondokumente aufnehmen und bieten zusätzlich noch die Möglichkeit der Interaktion. Der Spieler kann das Geschehen beeinflussen. Von Will Wright, dem Designer zahlreicher kommerziell erfolgreicher Computerspiele, wie beispielsweise den Sims und SimCity (2007), stammt der Vergleich mit einem Spielzeug: Ein Computerspiel ist ein Spielzeug, das ein System repräsentiert. Durch den Umgang mit dem Spiel kann der Spieler das System kennenlernen: Er wird mit den Auswirkungen seiner Aktionen im Spiel als geschützter Umgebung konfrontiert. Dadurch wird er trainiert, das Verhalten des Systems einzuschätzen.

Spiele als Lernmedium - auch *Serious Games* genannt - fördern das Lernen durch verschiedene Prinzipien. Zum einen veranlassen sie den Spieler zum aktiven Handeln. Seine Handlungen finden in einem Kontext statt, der für den Spieler bedeutungsvoll ist. Das hilft ihm, das notwendige Wissen selbst zu konstruieren. Auch bieten die komplexen Aufgaben einiger Spiele immer wieder Gelegenheit, mit anderen Spielern über mögliche Lösungen zu kommunizieren. Damit bekommt das Handeln auch einen sozialen Kontext. Lave & Wenger (1991) beschreiben das in ihrer Theorie des situierten Lernens auch mit Hilfe von sogenannten *Communities of Practice*. Ein weiteres Merkmal, das Lernen mit Hilfe von Spielen fördert, ist das Erstellen eigener Inhalte. Seymour Papert (1980) prägte diese Theorie unter dem Namen *Konstruktionismus*.

⁴ Vgl. Schank et al. (1994)

⁵ Vgl. Psychosocial Moratorium Principle (Gee, 2003)

Kommerzielle Spiele werden als Vorbild gesehen für die Heranführung des Spielers an komplexeste Systeme (Gee, 2003, 2005): Sie erwecken erstens ausreichend intrinsische Motivation im Spieler, um zweitens auch schwierige, langwierige Probleme als lohnenswerte, erfreuliche Ziele anzunehmen. Beide Prinzipien sollen genutzt werden, um mit Hilfe von Computerspielen auch Fachwissen vermitteln zu können - durch sogenannten Serious Games. Das erste genannte Prinzip eines Computerspiels - die Erzeugung intrinsischer Motivation - ist notwendig, um von einem Spiel sprechen zu können: Ohne Spielspaß verschwindet der Unterschied zu herkömmlichen Lernmethoden - es bleibt eine computergestützte Simulation. Das zweite Prinzip - Problemszenarios in authentischen Kontexten - ist Gegenstand des *Instructional Designs*: Wie müssen Lernmaterialien aufgebaut werden, damit der Lerner sich nicht überfordert fühlt, gleichzeitig aber auch nicht unterfordert und damit gelangweilt ist? Werden die gewünschten Lernziele mit den Lernmaterialien erreicht? Diese Fragen müssen für Serious Games auf den Entwurf von Spielsituationen übertragen werden.

Eine eindrucksvolle Entwicklung der Computerspiele - sowohl bezüglich der Verbreitung als auch der Qualität - wird nicht zuletzt durch Fortschritte in der Technik gefördert. Dennoch ist die Erstellung eines Computerspiels immer noch eine Herausforderung: Zum einen als techniklastiges Projekt, zum anderen auch als Design-Aufgabe, in der sowohl die fachlichen Lernziele berücksichtigt werden müssen als auch der Spielspaß nicht vernachlässigt werden darf. Die Erstellung eines Computerspiels ist damit selbst ein höchst komplexes Problem. Die Anforderung, es zu Lernzwecken nutzen zu wollen, steigert die Komplexität weiter. Gute Serious Games gelten als herausragende Werkzeuge zur Wissensvermittlung, sie gelten aber ebenso als Mangelware. Oft entstehen misslungene Kombinationen von Lerninhalten und Spielmechaniken (Egenfeldt-Nielsen, 2007; Kirriemuir, 2002). Brenda Laurel (2001) hat für solche Fehlschläge den Begriff „chocolate covered broccoli“ geprägt. Niegemann (2009) weist ebenfalls auf methodische Probleme bei der Erstellung und Anwendung von Spielen hin und kommt zu dem Fazit, dass „die Brauchbarkeit von Computerlernspielen für formal eingebundenes, systematisches, ergebnisorientiertes Lernen [...] eingeschränkt“ ist und der Einsatzbereich von Spielen eher in informellen Lernkontexten liegt. Simulationen (im Sinne von Simulationsspielen) bescheinigt er „realitätsnahe Handlungen bzw. Handlungsentscheidungen und informative Rückmeldungen“. „Goal Based Scenarios“ haben den Nachteil, dass die Realisierung sehr aufwändig ist. Sie sind jedoch als „theoretisch gut fundiert“ und „nachweislich effektiv“ angesehen.

1.2 Idee und Methodik

Today's students are no longer the people our educational system was designed to teach.
(Marc Prensky, 2001)

Der Ansatz, ein Serious Game für den Einsatz in der akademischen bauphysikalischen Ausbildung zu konzipieren, wird im Rahmen dieser Arbeit insbesondere von zwei Faktoren getrieben. Der erste Faktor ist gegeben durch das Eclipse RCP-Framework. Dieses Framework wird in der kommerziellen Softwareentwicklung eingesetzt und wird als Open Source Software betreut von der *Eclipse Foundation*, die von namhaften Industrieunternehmen gefördert wird. Die Software hat sich als sehr leistungsfähiges Werkzeug erwiesen, u.a. durch die starke Unterstützung der Prinzipien Modularität und Erweiterbarkeit. Mit diesen wird die Erstellung von wartbarer Software sehr vereinfacht. Eclipse wird von verschiedenen großen Software-Unternehmen als Basis für

ihre Produkte eingesetzt, so zum Beispiel von IBM, SAP und Google. Im Rahmen dieser Arbeit soll untersucht werden, ob ein Einsatz in der Spielentwicklung möglich ist.

Der zweite Faktor ist gegeben durch das Aufkommen von Social Network Games (SNGs). Diese Spiele basieren auf Social Network Services (SNS) und können dadurch auf einen schon gegebenen sozialen Kontext aufbauen. Durch die leichte Zugänglichkeit - Spieler müssen lediglich einen Webbrowser bedienen können - und die zumeist kurzen Spielzyklen („casual game play“) kann eine völlig neue Spielerdemographie erschlossen werden (Snow, 2010). Zudem sind die Entwicklungskosten im Vergleich zu herkömmlichen Computerspielen wesentlich geringer (Mahajan, 2010). Die Beobachtung der Spieler ist durch die kontinuierliche Client-Server-Kommunikation ebenfalls sehr einfach und kann zur Weiterentwicklung der Spiele genutzt werden (Nutt, 2011; Söbke, Hadlich et al., 2012). Im Rahmen dieser Arbeit soll daher ein Konzept entwickelt werden, das die Vorteile von SNGs auf das zu entwickelnde Spiel anwendet.

Neben diesen zwei Faktoren gibt es noch einige weitere Rahmenbedingungen, denen das zu erstellende Konzept Rechnung tragen soll. Dazu gehört die Möglichkeit für die Spieler, eigene Beiträge zum Spiel zu leisten. In Form von sogenannten Szenarios - das sind abgeschlossene kleine Welten, die fachliche Aufgaben für die anderen Spieler bereithalten - kann der Spieler sein Wissen dem Spiel zur Verfügung stellen. Das ursprüngliche Spiel wird durch diesen Ansatz zu einer Spielplattform. Dieser Ansatz der Szenarios - die insgesamt durch eine übergreifende Geschichte miteinander verbunden sind - bietet verschiedene Vorteile: Zum einen wird dem Spieler die Gelegenheit gegeben, durch die Erstellung eigener Artefakte zu lernen - wie es im Konstruktivismus postuliert wird (Papert, 1980). Zum anderen erlaubt dieser Ansatz, die Spieler als Ressourcen mit einzubinden: Um die Erstellungsaufwände für die Gesamtsystem in Grenzen zu halten und auf mehrere Partner zu verteilen, kann die Plattform mit einer initialen Ausstattung an Szenarios erstellt werden. Im Laufe der Zeit werden weitere Szenarien durch die Spieler ergänzt (Bröker, Söbke & Kornadt, 2011). Ein weiterer Vorteil ergibt sich als Spielmechanik für bestimmte Spielertypen: Durch die Aufzeichnung von Spieldaten der erstellten Szenarien - beispielsweise die Häufigkeit der Auswahl eines bestimmten Szenarios oder die Häufigkeit des erfolgreichen Lösen einer Aufgabe innerhalb eines Szenarios - lassen sich verschiedene Spielmechaniken benutzen. So können Bestenlisten geführt werden oder Auszeichnungen für das Erreichen definierter Grenzwerte vergeben werden. Zu guter Letzt bietet die Einheit des Szenarios aus methodischer Sicht den geeigneten Rahmen, Vorgehensweisen des Instructional Designs anzuwenden und gegebenenfalls für Spiele weiterzuentwickeln: Jedes Szenario ist in sich abgeschlossen und stellt eine Entwurfseinheit dar.

Wesentliches Merkmal eines Spiels und insbesondere auch eines Spiels mit bauphysikalischen Inhalten ist die Simulation von Systemen. Die Simulation und die zugehörige Modellbildung gehören zu den wichtigen theoretischen Grundlagen einer Spielplattform. Daher wird dieses Thema im Rahmen der Arbeit ausführlich betrachtet.

Die zu entwerfende Spielplattform hat nicht den Anspruch der - überspitzt formuliert - allumfassenden Lernmaschine, die jedwede weitere Lernaktivität überflüssig macht und vorprogrammiert Bauphysik-Experten produziert. Vielmehr wird sie als begleitendes Angebot im Lernprozess gesehen: Das Spielen soll freiwillig bleiben, weil ansonsten Spaß in Arbeit umgewandelt wird.

Dann jedoch verlieren Spiele ein prägendes Alleinstellungsmerkmal⁶. Die Aufgabe der Spielplattform ist es, auf unterhaltsame und spielerische Weise Interessen zu erzeugen und zu kanalisieren. Sie soll die Studenten in einen Dialog über fachliche Probleme verwickeln in der Hoffnung, dass sich eine Kultur um das Spiel entfacht (Squire, 2011). Ein latenter „Infektionsdruck“ mit bauphysikalischen Themen soll unterstützt werden durch die virulenten Methoden, die ein Social Network als Werkzeug bereitstellt. Genauso wie die Plattform das Spiel in der Gemeinschaft unterstützten soll - wie beispielsweise durch die Einbindung von Mitspielern bei der Lösungsfindung und Bewertung von Szenarien - so soll sie ebenfalls den Einzelspieler unterstützen. Ein Spieler soll auch dann Szenarien lösen können und damit die Ziele des Spiels erreichen können, wenn er es zeitweilig oder allgemein bevorzugt, ohne Interaktion mit Mitspielern zu agieren. Die Intention hinter der Plattform ist es, ein Angebot zu machen, aus dem der potentielle Spieler ihm zusagende Elemente annehmen kann⁷. Ein systemimmanenter Vorteil der Plattform ist, dass sie aufgrund der Zugreifbarkeit über das Internet auch im Rahmen von E-Learning-Aktivitäten eingesetzt werden kann. Sie ist damit eine Ergänzung derartiger Lernkonfigurationen, die unter den Prämissen von zeitlicher und räumlicher Unabhängigkeit, technischen Fortschritts sowie lebenslangen Lernens immer größere Verbreitung finden (Gorges, Bröker, & Kornadt, 2007; Kornadt & Gorges, 2004).

Im Rahmen dieser Arbeit wird eine Plattform mit den oben dargestellten Merkmalen konzipiert. Es wird ein Vorgehen bei der Szenarioentwicklung unter Beschreibung der zugehörigen Lernziele entworfen. Mit Hilfe von Prototypen werden dabei kritische, mit größerer Unsicherheit behaftete Bereiche der Software validiert. Nicht zum Umfang der Arbeit gehört eine Evaluierung der Plattform im praktischen Einsatz, da nicht mit einer ausreichenden technischen Belastbarkeit des Prototyps bei begrenzten Ressourcen gerechnet werden kann.

1.3 Aufbau der Arbeit

Anschließend an diese Einleitung wird in Kapitel 2 der Stand der Technik dargelegt. Dazu gehört das Thema *Spielbasiertes Lernen* als bestimmende Grundlage. Darauf folgend wird das Thema *Simulation* dargestellt. Simulation ist ein wesentlicher Bestandteil einer Spielplattform für bauphysikalische Inhalte. Daher wird das Thema in diesem Kapitel systematisch untersucht, um eine stabile Basis für das Konzept liefern zu können. Schließlich werden Aspekte des Entwurfes einer Spielplattform aus lerntheoretischer Sicht behandelt. Als treibendes Element wird hier das Lernziel herausgestellt.

In Kapitel 3 wird die Spielplattform in der Übersicht beschrieben - hier ist sowohl das übergreifende Spielkonzept als auch eine Beschreibung des Systemaufbaus zu finden.

Kapitel 4 befasst sich mit dem *SimuFrame* genannten Simulationskern. Das ist die Serverkomponente, in der das erweiterbare Framework Eclipse RCP zum Einsatz kommt. In diesem Kapitel wird insbesondere auf die Entwurfs-Grundsätze Modularität und Erweiterbarkeit und deren konstruktive Unterstützung durch das Framework eingegangen. Nach grundsätzlichen Überlegungen zur verwendeten Implementierungsstrategie für den Prototyp wird das Konzept für das

⁶ Rockwell & Kee (2011) formulieren hierzu: „Without the culture of play a game is just another assignment.“

⁷ Die typgerechte Adaptibilität wird schon im Bereich des E-Learning als ein erfolgskritischer Faktor für den Lernerfolg angesehen (Gorges & Kornadt, 2011).

verwendete Simulationsobjektmodell dargestellt. Weiterhin erfolgt eine Beschreibung wichtiger Komponenten des Simulationskerns. Als Ergebnis des Entwicklungsprozesses werden wichtige Aspekte des Implementierungskonzepts dargelegt.

Kapitel 5 stellt die Anwendung des konzipierten Softwaresystems auf die Inhalte der Bauphysik dar. Zunächst werden die Grundlagen und Richtlinien des Szenariodesigns zusammen mit einem Beschreibungsmuster entwickelt. Dann werden beispielhaft Szenarien entworfen, um die Anwendbarkeit des Gesamtsystems zu demonstrieren.

In Kapitel 6 erfolgt die Zusammenfassung und kritische Wertung der Arbeitsergebnisse in Form von Thesen.

Zum Abschluss finden sich die Verzeichnisse der benutzten Formeln, Tabellen und Abbildungen.

Der Arbeit beigelegt sind sieben Anhänge:

- **Anhang A „Übersicht ähnlicher Softwarepakete“:** Dieser Anhang ist das Ergebnis einer Recherche, die in einem frühen Stadium der Arbeit erfolgte und zum Ziel hatte, schon existierende Softwarepakete mit den gewünschten Merkmalen zu finden. Durch diese Recherche konnte kein fertiges Softwarepaket gefunden werden. Sie erbrachte aber viele Hinweise für die Entwicklung eines solchen.
- **Anhang B „Konstruktive Maßnahmen zur Verminderung des Software Aging“:** In diesem Anhang wird ebenfalls das Ergebnis einer frühen Arbeit dargestellt: Aus Anlass der Untersuchung bestehender bauphysikalischer Softwarepakete wurden einige schon seit längerer Zeit auf unveränderter technischer Basis existierende Programme (z.B. TRNSYS) gefunden. In diesem Anhang werden diejenigen Faktoren des Software-Engineering untersucht, die zu einem derartig langen Lebenszyklus beitragen können.
- **Anhang C „Grundlagen der Simulation“:** Simulation ist eine der wesentlichen Grundlagen der zu entwerfenden Spielplattform. In diesem Anhang werden die Ergebnisse einer methodischen Aufarbeitung des Gebietes der Simulation dargestellt. Die Ergebnisse dienen als Basis für die im Rahmen der Plattform stattfindende Simulation.
- **Anhang D „Fallstudie: Designanforderungen an bauphysikalische Simulationssoftware“:** Ingenieurmässiges Vorgehen in der Bauphysik wird bereits mit Hilfe von Simulationssoftware unterstützt. In diesem Anhang werden zwei Beispiele von Softwarepaketen dargestellt, die bezüglich jeweils eines Aspekts als Vorbild - und damit als Grundlage - für die Spielplattform angesehen werden können.
- **Anhang E „Computerspiele als Lernmedium“:** Der Einsatz von Computerspielen als Hilfsmittel zum Lernen - sowohl in formellen als auch in informellen Kontexten - stellt ein faszinierendes Gebiet dar, das im Rahmen dieser Arbeit untersucht werden konnte. Mit diesem Anhang werden interessante Grundlagen und Erkenntnisse dokumentiert.
- **Anhang F „Easy Gold - ein Minispiel in der Gemeinschaft“:** Dieses Dokument beschreibt das SNG „Easy Gold“, das prototypisch im Rahmen dieser Arbeit entwickelt wurde und insbesondere dem zweiten treibenden Faktor dieser Arbeit, Computerspiele auf Basis von SNSs (hier: Facebook) beleuchtet.
- **Anhang G „Plattformbeschreibung“:** Dieser Anhang ist die technische Dokumentation der erstellten Artefakte, es werden die verschiedenen Module, Erweiterungsschnittstellen und Benutzeroberflächenelemente des Simulationskerns beschrieben.

Im Sinne einer besseren Lesbarkeit wird bei Personenbezeichnungen und zugehörigen Pronomen durchgängig die männliche Form genutzt. Diese Form schließt die weibliche Form ein, die auch stattdessen an den betreffenden Stellen gewählt werden könnte.

2 Stand des Wissens

Was man lernen muss, um es zu tun, das lernt man, indem man es tut.
Aristoteles

2.1 Spielbasiertes Lernen

2.1.1 Das Medium „Computerspiel“

Computerspiele gehören zur Gruppe der Medien: Jesse Schell⁸ sieht sie gar als eines der umfassendsten Medien: In Computerspielen lassen sich alle anderen Vertreter dieser Gruppe, wie Schriftstücke, Ton- und Videoaufnahmen integrieren. Zudem bieten sie dem Spieler die Möglichkeit der Interaktion - der Spieler kann aktiv Einfluss nehmen auf das Spielgeschehen und den Status des Spieles kontrollieren (Schell, 2008)⁹.

Die inzwischen enorme wirtschaftliche Bedeutung des Mediums Computerspiel sowohl für die Gesamtwirtschaft als auch für Einzelpersonen zeigt sich an den folgenden Zahlen:

- Die Umsatzvorhersage für den weltweiten Computerspiel-Gesamtmarkt belief sich für 2011 auf 65 Milliarden US \$ (Baker, 2011). Zum Vergleich: die Summe der Umsätze an den Kinokassen betrug in 2011 gut die Hälfte, nämlich 32.6 Milliarden US \$ (MPAA, 2012).
- Der Umsatz an virtuellen Gütern in Spielen wurde für die USA in 2012 auf 2,9 Milliarden US \$ geschätzt (Eldon, 2011). Hierzu zählen beispielsweise auch die Einzelkäufe von virtuellen Gegenständen in SNGs wie *FarmVille* auf Facebook. Der weltweite Gesamtmarkt wird auf 14,8 Milliarden US \$ geschätzt (Superdata, 2012).
- Aufgrund der Tatsache, dass virtuelle Gegenstände des Online-Spiels *Everquest* zwischen den Spielern gegen echtes Geld gehandelt wurden, konnte Edward Castronova 2001 errechnen, dass ein durchschnittlicher *Everquest*-Spieler einen Stundenlohn von 3,42 US \$ hat. Ein solcher Stundenlohn führt zu einem Bruttosozialprodukt pro Einwohner, das zur damaligen Zeit zu Rang 77 in der Liste aller Volkswirtschaften reichte.
- Das Handeln von Spielgegenständen auf Märkten außerhalb des Spiels hat zum Geschäftsmodell des „Gold Farming“ geführt (Gillmore, 2010): Menschen aus Regionen mit geringen Arbeitslöhnen spielen berufsmäßig: Sie „produzieren“ dabei Spielgegenstände, die sich gegen echtes Geld an Spieler mit höheren Einkommen verkaufen lassen oder bieten jenen über Plattformen Dienste wie das Hochleveln¹⁰ von Spielfiguren an. Lehdonvirta & Ernkvist (2011) schätzen diesen Markt, der zum großen Teil aus China bedient wird, auf ein Volumen von 3 Milliarden US \$.

⁸ Schell ist Game Designer und Professor der Carnegie Mellon University, Pittsburgh, Pennsylvania.

⁹ Diese Aussage ist zu differenzieren: Auch in Online-Lernumgebungen ist Interaktion für den Lerner möglich, vorgesehen und oft dem Lernerfolg zuträglich - wie von Niegemann (2011) diskutiert wird. Ebenfalls können heutige Online-Lernumgebungen alle anderen Medien einbetten. Die Aussage von Schell kann dennoch als ein Hinweis auf die exponierte Stellung von Computerspielen innerhalb der Medien gewertet werden.

¹⁰ Hochleveln ist das gezielte Entwickeln einer Spielfigur (Avatar), die durch Erreichen höherer Stufen mächtiger wird bzw. angesehener ist.

- Bei sogenannten E-Sports¹¹ Veranstaltungen gibt es beträchtliche Preisgelder zu gewinnen, die mitunter für den Sieger mehr als 100.000 US \$ betragen können. Es haben sich professionelle Ligen und Veranstaltungsserien etabliert (Breslau, 2011).

a) Eigenschaften des Spiels

Grundlage eines jeden Computerspieles ist ein Spiel - das Computerspiel ist die digitalisierte Form eines Spiels. Für den Begriff „Spiel“ gibt es eine Vielzahl von Definitionen mit teils unterschiedlichen Aspekten. Zwei von ihnen werden an dieser Stelle genauer betrachtet und die einzelnen Bestandteile erläutert.

Spiel ist eine regelbasierte Interaktion, die die Spielerinnen und Spieler emotional bindet und innerhalb eines von der objektiven Realität abgegrenzten Raums stattfindet.

- Michael G. Wagner (2008)

A game is a problem-solving activity, approached with a playful attitude.

Jesse Schell (2008)

Die beiden Definitionen enthalten die folgenden Facetten eines Spiels:

Regeln

Ein Spiel findet immer im Rahmen von Regeln statt. Die Regeln definieren den abgegrenzten Raum, in dem eine Lösung für ein Problem zu finden ist.

Problemlösung

Aufgabe eines Spiels ist es, ein gegebenes Problem im Rahmen der Regeln zu lösen. Dieses Problem kann entweder durch Mitspieler erzeugt werden oder durch die Regeln des Spiels vorgegeben sein.

Interaktion/Aktivität

Wesentliches Merkmal aller Spiele ist die Aktivität des Spielers: Der Spieler kann durch Handeln innerhalb der gegebenen Regeln den Ausgang des Spiels beeinflussen.

Emotionale Bindung

Die Motivation, sich in einem Spiel zu engagieren, ist nicht zweckgebunden: Mit der Teilnahme will der Spieler kein außerhalb des Spielrahmens liegendes Ergebnis erreichen. Die Befriedigung (allgemein gesprochen: die positiven Emotionen), die aus dem Spielen resultiert, ist die Motivation für eine Teilnahme an dem Spiel. Es wird auch von intrinsischer (im Gegensatz zu extrinsischer) Motivation gesprochen.

Abseits der Realität

Ein Spiel wird auf den Grundlagen definierter Regeln durchgeführt. Diese Regeln schaffen einen von der Realität abgekoppelten Raum, auch wenn sie in einigen Fällen der Realität nachempfunden sein können.

¹¹ Mit E-Sports wird das wettbewerbsorientierte Spielen von Computerspielen bezeichnet. Die Struktur ist ähnlich der einer herkömmlichen Sportart wie Fußball, nur dass das Sportgerät ein anderes ist: Statt eines Fußballes wird ein Computerspiel wie StarCraft II genutzt. Ereignisse der Profiligen werden - ebenso wie es bei Spielen professioneller Fußballmannschaften üblich ist - von mehreren tausend Zuschauern besucht und im Fernsehen bzw. online übertragen. Ein Zentrum des E-Sports ist Südkorea (Taylor, 2012).

Spielerische Einstellung

Der Spieler nimmt freiwillig an einem Spiel teil, es bereitet ihm Vergnügen, es gibt keine direkten Konsequenzen für das reale Leben - weder in positiver noch in negativer Hinsicht.

b) Computerspiel

Der Computer ist eines von vielen möglichen Hilfsmitteln zur Durchführung von Spielen. Zu anderen Hilfsmitteln zählen auch Spielsteine, Würfel, Bälle und Schläger¹². Von einigen Spielen gibt es Versionen mit dem Hilfsmittel Computer und ohne das Hilfsmittel Computer. Die Computerversion eines Spieles hat die Voraussetzung, dass auch tatsächlich ein oder mehrere Computer für die Durchführung des Spieles zur Verfügung stehen. Ein Vorteil der Computerversion ist die automatisierte Überprüfung der Einhaltung der Spielregeln. Neben weiteren Vorzügen wie der Benutzung von komplexen Simulationsmodellen erlaubt der Computer auch die Integration von akustischen und visuellen Effekten, die die Attraktivität dieses Mediums erhöhen. Zusätzlich ermöglicht der Computer die Ortsunabhängigkeit der Spieler in Mehrspielerspielen. Dadurch ergeben sich gewöhnlich Alleinstellungsmerkmale von Computerspielen gegenüber der nicht digitalisierten Form. Eines der Alleinstellungsmerkmale ist die mögliche Funktion des Computers als Spielgegner. Existierende Ausprägungen reichen vom Schachcomputer, der in fast jeder Spielstärke eine Herausforderung für den Spieler darstellt, bis zum Non-Player Character (NPC), der ebenfalls in Abhängigkeit von zugeordneten Algorithmen eine nahezu beliebig komplexe Aufgabe bietet. Kritisiert wird bei Computerspielen häufig die fehlende Kommunikation mit anderen Spielern, die zu einer Isolation des einzelnen Spielers führe¹³.

Die komplexen Simulationsmodelle, die einem Computerspiel möglicherweise zugrunde liegen, können ein System-Modell der Realität sein. Damit ist es dann möglich, Computerspiele als virtuelle Welt zu nutzen, um Systemzusammenhänge der realen Welt dem Spieler zu vermitteln.

Die Akzeptanz des Mediums Computerspiel ist personengruppenabhängig: Prensky (2001) nennt Personen, die mit neuer Kommunikationstechnologie¹⁴ aufgewachsen sind, „Digital Natives“ - hingegen werden mit „Digital Immigrants“ diejenigen bezeichnet, die sich als Erwachsene den Umgang mit diesen Technologien erst mühevoll aneignen mussten. Wahrscheinlich nicht zuletzt aufgrund dieser unterschiedlichen Voraussetzungen gibt es gesellschaftliche Vorbehalte gegenüber diesem Medium, die sich auch darin äußern, dass das Medium Computerspiel gesellschaftlich nicht in dem Maße akzeptiert ist wie beispielsweise das Medium Buch (S. Anderson, 2012)¹⁵.

c) Computerspiele als Third Place

Der Begriff *Third Place* wurde von Ray Oldenburg (1999) geprägt, damit werden Umgebungen abseits von Wohnen und Arbeiten unter dem Aspekt der gesellschaftlichen Bedeutung zusam-

¹² Der Computer erreicht eine besondere Stellung unter den Hilfsmitteln: Mit seiner Hilfe können komplette Spiele abgebildet werden.

¹³ Dies ist nur ein Kritikpunkt von vielen - Computerspiele werden teilweise sehr kritisch gesehen, erinnert sei an die Diskussion, inwieweit Gewalt durch Computerspiele ausgelöst wird (Dörner, 2010) oder an die Folgen übermäßiger Nutzung von Computerspielen.

¹⁴ Unter dieser Bezeichnung sind sinngemäß auch Computerspiele einzuordnen.

¹⁵ Deutlich wird ein solcher Vorbehalt auch in der gesellschaftskritischen Fernsehserie „Southpark“: In Folge 1008, „Make Love, not Warcraft“ fordert ein Vater seinen computerspielenden Sohn auf, doch etwas mit seinen Freunden zu unternehmen. Daraufhin weist ihn sein Sohn darauf hin, dass er doch gerade mit seinen Freunden spielt und kommuniziert - in World of Warcraft (<http://www.southpark.de/alle-episoden/s10e08-make-love-not-warcraft>, 0:58, letzter Zugriff 11.10.2012).

mengefasst. Putnam (1995) stellt eine Abnahme des gesellschaftlichen Zusammenhalts fest aufgrund der Abnahme der traditionellen Third Places fest. Schließlich weisen Steinkuehler & Williams (2006) die Bedeutung von Online-Spielen als Third Places dieser Zeit nach. Rao (2008) untersucht die Bedeutung von SNGs als Third Place am Beispiel des SNS Facebook.

Als konkretes Beispiel für die Bedeutung von Spielen als Third Place, als Ort des gesellschaftlichen Austausches soll das SNG Fliplife¹⁶ vorgestellt werden: Dieses Spiel zeichnet sich durch - sogar im Vergleich zu anderen SNGs, die allgemein als sehr leicht zu meistern gelten - äußerst einfache Spielmechaniken aus. Dennoch verbringt eine Anzahl von ernsthaften Spielern einen Großteil ihrer Freizeit in und mit dem Spiel. Es kommt dabei teilweise zu sehr privater und bedeutungsvoller Kommunikation abseits des Spielgeschehens. Sieht man davon ab, dass die Kommunikation durch die nicht physische Präsenz der Beteiligten eine andere - nicht notwendigerweise schlechtere - Qualität hat, als die direkte Kommunikation an traditionellen Third Places, so kann Fliplife als ein Third Place dieser Zeit gewertet werden (Naira Müller, Hennig, Aubel, Hesse & Schneider, 2012; Söbke, Hadlich et al., 2012).

In Bezug auf Lernprozesse haben Third Places eine Bedeutung, weil an ihnen eine Gemeinschaft entsteht: Sie bieten einen informellen Lernkontext. Überall dort, wo Menschen zusammenkommen, entstehen zwischen den Menschen Beziehungen, die u.a. auch eine Basis des Lernens bieten können (s.a. Anhang E).

Ein weiteres Beispiel für Onlinespiele als virtueller Treffpunkt - nun in einem formalen Kontext - ist eine Konferenz von Wissenschaftlern, die im Online-Spiel *World of Warcraft* stattgefunden hat (Bohannon, 2008): Statt ein herkömmliches Konferenzwerkzeug zu nutzen, konnten die Wissenschaftler mit Hilfe der Infrastruktur eines Spiels miteinander in Kontakt treten.

d) Games und games - die Kultur um ein Spiel herum

Eine weitere Erscheinung, mit der ein Spiel einhergeht, ist die Kultur, die sich um Spiele aufbaut: Spiele dienen als Kondensationspunkte für kulturelle und soziale Interaktionen, die für das Leben der Menschen eine weitaus größere Bedeutung haben als das bloße Spiel selbst. Ein Beispiel für die europäische Kultur von Spielen ist das Spiel Fußball: Das Spiel selbst ist relativ einfach: Zwei Mannschaften von je 11 Spielern versuchen, ein kugelförmiges Spielgerät einer bestimmten Größe und mit bestimmten physikalischen Eigenschaften unter Nutzung von bestimmten Körperteilen durch rechteckige Ziele einer festgelegten Größe an bestimmten Positionen des Spielfeldes zu bewegen. Die um dieses Spiel mit einfachen, abstrakt klingenden Regeln herum entstandene vielschichtige Kultur zeigt sich an zahllosen Stellen, einige davon sind:

- Inhalte des Spiels werden durch die meisten Medien transportiert, die von Menschen bereitwillig konsumiert und finanziert werden. Das Spiel ist Gegenstand in der Kommunikation der Menschen untereinander.
- Der Besuch solcher Spiele sowohl im Stadion als auch bei öffentlichen Live-Übertragungen („Public Viewing“) ist ein soziales Ereignis, an dem Menschen freiwillig und mit großer Begeisterung teilnehmen. Das Spiel wird zum zeitweiligen Lebensinhalt der Menschen, die damit in Gesellschaft ihre Freizeit verbringen.
- Für viele Menschen hat das Spiel eine große Bedeutung, es ist Teil ihrer Identität: Sie verbinden Erfahrungen mit dem Spiel - entweder selbst als Spieler oder aber als Fans ei-

¹⁶ www.fliplife.com, letzter Zugriff am 16.10.2012

ner bestimmten Mannschaft. Deutlich wird dieses bei sogenannten Lokalderbys, bei denen die Identifikation der Fans mit ihrer Mannschaft besonders stark gelebt wird.

In ähnlicher Weise findet auch Kultur um Computerspiele statt, Beispiele hierfür sind:

- Menschen identifizieren sich selbst als Spieler („Gamer“) (Bogost, 2011a).
- Spiele dienen als Third Place: als Treffpunkt für Menschen mit verwandten Werthaltungen und ähnlichem Denken, Handeln und Fühlen.
- Über Spiele und mit Spielen wird kommuniziert: u.a. in Foren findet ein reger Austausch statt.
- Es wird oft gemeinsame Wissenskonstruktion durchgeführt, beispielsweise mit Hilfe von Wikis (Söbke, Corredor & Kornadt, 2011).
- Spiele dienen als Inspiration für gestalterische Prozesse: Unter dem Namen *Fan Fiction* werden Texte über Inhalte von Medien wie Bücher und Filme sowie über tatsächliche Personen zusammengefasst, die im nicht-professionellem Kontext entstanden sind und von Gleichgesinnten gelesen und rezensiert werden (McCardle, 2003).
- Einige Spiele selbst werden als Kunstgegenstand gesehen (Solarski, 2012; Swaim, 2009).
- Auch lassen sich Spiele als Mittel künstlerischer, gestaltender Tätigkeit einsetzen. Ein Beispiel ist die künstlerische Gestaltung von virtuellen Spielwelten. In Abbildung 1 ist eine aufwändig aus Einzelgegenständen zusammengestellte Farm des SNGs FarmVille zu sehen, auf der übergeordnete Geometrien zu erkennen sind.
- Generell können Spiele als Ergebnis eines gestalterischen Prozesses auch als Kulturgegenstand gesehen werden.



Abbildung 1: Künstlerisch gestaltete FarmVille-Welt (Wei, 2010)

e) Spiele als Vermittlern von Systemen

Dass der Umgang mit Systemen für den Menschen oft schwierig ist, zeigt Dörner (1981, 2003) auf. Er kategorisiert Ursachen der Schwierigkeiten und nennt die folgenden Schwerpunkte:

- **Nicht-Berücksichtigung von zeitlichen Abläufen**
Das Wissen um den reinen Wert einer Zustandsvariablen ist weniger wichtig als die Entwicklung dieser Variablen über den Zeitverlauf: „Nimmt ihr Wert ab?“, „Nimmt ihr Wert zu?“, „Mit welcher Geschwindigkeit?“ sind Fragen, deren Beantwortung für die Einschätzung des Systemverhaltens und bei Entscheidungen über mögliche Maßnahmen weit wichtiger ist.
- **Falsche Einschätzung von exponentiellen Entwicklungen**
Exponentielle Zunahmen werden in ihrer Bedeutung von Menschen gewöhnlich nicht korrekt vorausgesagt. In vielen Systemen der realen Welt sind sie aber anzutreffen.
- **Denken in Kausalketten statt Kausalnetzen**
Systeme sind netzartig aufgebaut. Maßnahmen beeinflussen gewöhnlich nicht nur die eine Systemvariable, sondern auch weitere Systemvariablen in Form von Nebenwirkungen - die aber gewöhnlich in einem ersten, intuitiven Lösungsansatz nicht beachtet werden.

Um die Bedeutung von Systemen und deren Betrachtung für Spiele zu unterstreichen, sei das Beispiel der Mehrspieler-Weltraumsimulation EVE Online genannt: Für dieses sogenannte *Massively Multiplayer Online Game* (MMOG), das seit Juni 2003 existiert und inzwischen mehrere Hunderttausend Spieler hat, wurde ein promovierter Wirtschaftswissenschaftler eingestellt mit der Aufgabe, die ökonomischen Prozesse in der virtuellen Welt zu untersuchen (CCP Games, 2007; Plumer, 2012). Als Beispiel seiner Arbeit sei der Blog-Eintrag „What is going on with mineral prices?“ genannt - eine Abhandlung über die Entwicklung der Preise von Mineralien¹⁷ mit der abschließenden Feststellung, dass sich für den aufmerksamen Spieler aller Wahrscheinlichkeit nach interessante Chancen im Handel ergeben werden (Dr.EyjoG, 2012). An dieser Stelle gibt es keinen Unterschied zu realen Prozessen der Wirtschaft.

Dieses Beispiel zeigt eine generelle Eignung von Computerspielen auf: Sie sind ein Werkzeug, um den Umgang mit Systemen zu erlernen: Will Wright, der Designer der Computerspiele SimCity und Die Sims sowie Spore bezeichnet Computerspiele als Werkzeuge zum Erlernen des Umgangs mit Systemen (Wright, 2007).

2.1.2 Motivation als Entwurfsziel

Das Gehirn ist auch ein Speicherorgan. Aber in erster Linie ist das Gehirn ein Organismus zur Abwehr unwillkommener Lernerfahrungen.
Peter Sloterdijk

Unter Motivation wird der Antrieb verstanden, eine Handlung durchzuführen. Es wird zwischen **intrinsischer** und **extrinsischer** Motivation unterschieden: intrinsisch ist ein Mensch motiviert, wenn sein Antrieb zu einer Aktion allein aus der Aktion selbst entstammt - beispielsweise wenn ein Student liest, weil es ihm Freude bereitet. Extrinsisch motiviert hingegen wäre derselbe Student, wenn es einen außerhalb der Aktion liegenden Anreiz gibt - so zum Beispiel liest er, um sich das nötige Wissen zum Bestehen einer Prüfung anzueignen (R. Ryan & Deci, 2000).

¹⁷ Dies sind in EVE Online Rohstoffe, die weiter zu Produkten veredelt werden.

Die Motivation zu einem Spiel ist gewöhnlich intrinsisch. Der Spieler nimmt freiwillig an dem Spiel teil, weil es ihm Freude im weitesten Sinne bereitet. Dieses ist ein Unterschied gegenüber traditionellen pädagogischen Formaten, wie beispielsweise Frontalunterricht in der Schule. Das Kalkül des Game Based Learning ist es, die intrinsische Motivation des Spielens mit dem Erlernen von Fähigkeiten, die für das reale Leben relevant sind, zu verbinden. Daher ist die Erzeugung intrinsischer Motivation der wesentliche erfolgskritische Faktor des Game Based Learning. Entsprechend ist intrinsische Motivation in Spielen Gegenstand der Forschung. Einige wesentliche Ergebnisse werden im Folgenden dargestellt.

Malone & Lepper (1987) haben eine Taxonomie derjenigen Elemente erstellt, die beim Design einer Lernumgebung¹⁸ berücksichtigt werden sollten, um intrinsische Motivation zu erzeugen. Die Einordnung ist zunächst zweigeteilt: Es wird unterschieden zwischen individuellen („individual motivations“) und zwischenmenschlichen („interpersonal motivations“) Motivationsarten. Die zwischenmenschlichen Motivationsarten umfassen diejenigen, die sich aus dem Zusammenleben der Menschen ergeben. Dazu gehören Zusammenarbeit, Wettbewerb und Anerkennung. Beispielsweise motiviert Wettbewerb: Der Drang, sich mit anderen zu vergleichen und ein besseres Ergebnis zu erreichen ist für viele Menschen ein starker Motivator. Hingegen zählen zu den individuellen Motivationsarten Herausforderung, Neugierde, Kontrolle und Fantasie: So stellen ehrgeizige Ziele, die dennoch erreichbar zu sein scheinen, einen großen Anreiz dar.

Die Arbeit von Malone & Lepper definiert die Arten von Motivation, die durch das Design einer Lernumgebung bzw. eines Spiels erreicht werden sollen. Sie gibt aber keine Auskunft darüber, wie die entsprechenden Anreize erzeugt werden können. Es ist Aufgabe des Designers mit Spielmechaniken derartige Anreize zu generieren. Dafür existieren jedoch keine eindeutigen Handlungsanweisungen und Rezepte. Die Mittel müssen jeweils dem Kontext angepasst sein und sie verändern unter Umständen ihre Wirkung wenn sie zusammen angewendet werden. Daher sollte jede Kombination einem Testprozess unterzogen werden, um die gemeinsame Wirkung der Komponenten im Zusammenspiel zu testen. Gegebenenfalls muss modifiziert werden (Fullerton, 2008). Es entsteht ein iterativer Entwicklungsprozess, bis ein zufriedenstellendes Produkt vorliegt. M. G. Wagner (2009) sieht dieses Vorgehen als essentiell für die Erstellung von Spielen an, sequentielle Vorgehensmodelle führen gewöhnlich nicht zum Erfolg. Spielspaß lässt sich in der Regel nicht algorithmisch entwerfen, sondern ist oftmals inspiriert durch die Erfahrung und das Gespür des Spieldesigners. Daher werden die Erkenntnisse erfahrener Designer in sogenannten Mustern zusammengefasst. Die ergeben eine Art Werkzeugkasten, mit dessen Hilfe die Entwicklung von anregenden Lernumgebungen bzw. Spielen effizienter werden kann. Ein Beispiel ist die Arbeit von Holopainen & Björk (2008), die Muster von motivierenden Spielmechaniken zusammengestellt haben.

2.1.3 Social Network Games

a) Grundlagen

Auf der Basis der seit Mitte des letzten Jahrzehnts schnell wachsenden Social Networking Services (SNS) haben sich Social Network Games (SNG) als neues Format von Computerspielen etab-

¹⁸ Gegenstand der Forschung sind hier Lernumgebungen und nicht Spiele. Wichtig ist jedoch, dass der Lerner intrinsisch motiviert ist zu lernen - und Malone & Lepper geben Hinweise, wie eine solche Motivation erzeugt werden kann.

liert. Diese Spiele nutzen die Infrastruktur von SNS wie Nachrichten, Statusmeldungen und Pinnwandeinträgen als Spielmechaniken und Koordinationsinstrument zwischen den Spielern. Asynchronizität als wichtige Eigenschaft solcher Spiele wurde von Bogost schon 2004 beschrieben: Die Spieler müssen nicht zur selben Zeit online sein, sondern können ihre Spielaktionen sequentiell durchführen - ohne, dass eine rundenbasierte Spielmechanik notwendig ist. Damit werden die Voraussetzungen für ein gemeinsames Spiel gelockert - die Gruppe der potentiellen Spieler wird hierdurch vergrößert.

Järvinen (2009) identifiziert neben der Asynchronizität noch vier weitere, maßgeblich den Entwurf prägende Merkmale von SNGs, die einem Einsatz als Lernmedium zugutekommen:

- **Inhärente Geselligkeit** („inherent sociability“)
Die meisten Spiele besitzen eine gesellige Komponente, diese wird SNGs gezielt genutzt: Sind die Spieler in einen sozialen Kontext eingebunden, so ergeben sich positive Effekte unter anderem auf die Motivation.
- **Symbolische Körperlichkeit** („symbolic physicality“)
Durch die Anreicherung der möglichen Interaktionen um körperliche Aspekte, wie z.B. Anstupsen („Poke“) und Abklatschen („High Five“) soll menschliche Präsenz und Wärme vermittelt werden - der soziale Kontext wird erhöht.
- **Spontanität** („spontaneity“)
Mit einfachen Spielmechaniken, bei der sich oft durch einfachen Knopfdruck Aktionen auslösen lassen, wird die Spontanität unterstützt: Eine hohe Zugänglichkeit des Spiels sowie die Möglichkeit des beiläufigen („casual“) Spiels fördern die Nutzung des Spiels. Es sind weder lange Vorbereitungszeiten noch lange Spielzeiten notwendig, um im Spiel Fortschritt zu erzielen.
- **Narrativität** („narrativity“)
Die Handlungen des Spiels werden als Geschichte kommuniziert, beispielsweise über Statusmeldungen und Pinnwandeinträgen. Das Ziel ist es, die Spielmotivation aktueller Spieler zu erhalten sowie gleichzeitig die Neugier von potentiellen Spielern zu erzeugen.

Insbesondere die hohe Zugänglichkeit ist ein wesentlicher Grund für die rasche Verbreitung von SNGs: Jeder Spieler, der in der Lage ist, einen Webbrowser zu bedienen, hat damit auch gleichzeitig Zugriff auf das Spiel. Dies ist ein Unterschied zu Computerspielen, die eine lokale Installation auf einem PC erfordern bzw. den Besitz von Spielkonsolen oder weiterer spezialisierter Hardware voraussetzen (Söbke, Corredor et al., 2011). Damit ist eine neue Gruppe von Spielern entstanden (Snow, 2010): Die Reichweite des Mediums Computerspiel wurde erheblich erhöht.

b) Eignung für die Ausbildung

Zeitliche Struktur

Die Spielmechaniken der SNGs weisen einige Besonderheiten im Vergleich zu herkömmlichen Computerspielen auf: Prägendes Merkmal des Spiels vieler SNGs ist das sogenannte „casual game play“, d.h. das beiläufige Spiel. Es ist gekennzeichnet von kurzen Spielzyklen ohne große Rüstzeiten. Durch spezielle Spielmechaniken, von denen die prominenteste der Einsatz von Timern ist, soll die Wiederkehr der Spieler sichergestellt werden. Somit ergeben sich im Idealfall regelmäßige, kurze Spielzyklen über einen längeren Zeitraum. Dieses steht im Kontrast zu den Spielzeiten in einem herkömmlichen Computerspiel: Dort herrschen eher längere, konzentrierte-

re Spielsitzungen vor, die aber nach einer bestimmten Zeit eingestellt werden, da dann alle Aufgaben des Spiels gelöst wurden¹⁹.

Das zeitliche Schema des Spiels von SNGs stimmt überein mit der zeitlichen Struktur von formalen Lernszenarien in Schule und Studium: Auch dort herrschen zeitliche Strukturen vor, die regelmäßige Aktivitäten über einen längeren Zeitraum verlangen. Aufgrund dieser Ähnlichkeiten können SNGs ein Ansatz sein, formale Ausbildung zu begleiten und mit Hilfe des Spielspaßes die Motivation zu erhöhen.

Soziale Interaktion und Gemeinschaft

Hicks (2010) postuliert für ein durchzuführendes Projekt, in dem ein SNG zu Lernzwecken erstellt werden soll, Lernvorteile durch die Interaktionen innerhalb des Spiels. Auch soll es zu mehr prägenden Lernerfahrungen kommen durch einheitliche Präsentation des Lernmaterials unter Verwendung der Spielmetapher.

In Fortführung dieser Arbeit berichten E. M. Powell et al. (2010 und 2011) von SNAG, einer Sammlung von gemeinschaftsstiftenden Spielen, die zur Verbesserung der Beziehungen von Gruppen im akademischen Kontext entwickelt wurden. Sie kommen zu der Schlussfolgerung, dass der Einsatz dieser Spiele die Zusammenarbeit der einzelnen Gruppenmitglieder verbessert hat.

Motivation

Lewis et al. (2012) haben verschiedene populäre SNGs aus Sicht der Verhaltenspsychologie und -ökonomie untersucht. Ziel ist es, Muster in den Methoden zu finden, die die Spieler zum wiederholten Spiel veranlasst. Das Wissen um diese Zusammenhänge ist ein wichtiger Baustein zur Entwicklung von effektiven Spielen zur Lernbegleitung.

2.1.4 Gamification

Gamification ist ein Schlagwort, das erst in den letzten Jahren stärker in den Fokus gerückt ist. Darunter wird die Anwendung von Spielprinzipien auf Prozesse der realen Welt zum Zwecke der Motivationssteigerung bei der Durchführung dieser Prozesse verstanden. Spaß und intrinsische Motivation, die sich beim Spielen einstellen, soll auf Anwendungen der realen Welt übertragen werden und diese attraktiver machen. Beispiel ist der ortsbasierte soziale Netzwerkservice *foursquare*²⁰: Mitglieder können sich mit Hilfe eines mobilen Endgerätes (z.B. ein Smartphone) an Orten einchecken, wenn sie diese erreichen. Damit zeigen sie im sozialen Netzwerk an, wo sie sich befinden und - abhängig vom Ort - was sie gerade machen. Zusätzlich lassen sich Belohnungen für die Check-Ins verteilen: Die Person mit den meisten Check-Ins an einem Ort wird zum Bürgermeister dieses Ortes. Kommerzielle Partner können Auszeichnungen und Vergünstigungen gewähren, wenn der Spieler an ihren Orten eincheckt - damit wird *foursquare* zu einem Marketinginstrument. Beispielsweise kann eine Auszeichnung vergeben werden beim wiederholten Besuch eines Fitness-Studios - damit tragen die Gamification-Elemente von *foursquare* zum Gesundheitsstatus bei.

Zu den typischen Gamification-Werkzeugen zählen

¹⁹ Ein in Rezensionen von solchen Spielen häufig genanntes Merkmal ist die Angabe der Stunden, die aufgewendet werden müssen, um das Spiel komplett zu spielen.

²⁰ www.foursquare.com, zuletzt abgerufen am 7.8.2012

- **Auszeichnungen**, die ein Benutzer für bestimmte Aktionen erhält,
- **Level**, den ein Benutzer besitzt,
- **Ranglisten**, in denen sich ein Benutzer wiederfindet,
- **Fortschrittsbalken**, mit denen sich ein Benutzer über den Status einer Aufgabe orientieren kann,
- **Währungs- und Punktesystem** zu Belohnungszwecken, beispielsweise im Sinne von Rabattmarkenheften.

Weitere Beispiele des Einsatzes von Gamification ist die Q&A²¹-Software von *Stackexchange*²² sowie die Crowdsourcing-basierenden Spielanwendungen *FoldIt* (FoldIt, 2010) und *EteRNA* (EteRNA, 2011). Die Stackexchange-Software erlaubt über das Stellen und Beantworten von Fragen den Aufbau eines Status mit wachsenden Rechten als Belohnung. Bei FoldIt und EteRNA werden die nutzbringenden Aktionen des Spielers (das Finden einer räumlichen Molekülstruktur bzw. der Struktur eines Moleküls selbst) mit Hilfe von Statusverbesserungen sowie visuellem und akustischen Rückmeldungen belohnt. Abbildung 2 zeigt eine Situation aus EteRNA, in der nach dem Lösen eines Puzzles die aktuelle Position des Spielers im Vergleich zu den Mitspielern angezeigt wird.



Abbildung 2: Anzeigen der Rangliste nach Lösung eines Puzzles in EteRNA

Gamification ist ein mächtiges Werkzeug, das allerdings auch fachgerecht eingesetzt werden muss: Ian Bogost (2011) beklagt die Trennung von Belohnungsmechaniken und wirklichen Spielmechaniken - Gamification umfasst nur Belohnungsmechaniken, aber keine bedeutungsvollen Spielmechaniken mehr. Er spricht daher von *exploitationware* - einem Hilfsmittel zur Ausbeutung: Gamification wird zur Manipulation der Spieler genutzt.

Ähnlich ist die Kritik von Sebastian Deterding (2011) an den Ausführungen von Zichermann & Cunningham (2011): Das Rezept, über eine Anwendung isoliert eine Schicht von Gamification-Elementen zu legen und damit automatisch eine Attraktion für den Benutzer zu erhalten, wird in

²¹ Q&A: Question and Answer Software, eine Art von Forumssoftware, die anhand von Fragen strukturiert wird.

²² www.stackexchange.com, zuletzt abgerufen am 7.8.2012

den meisten Fällen nicht funktionieren: Genau wie beim Entwurfsprozess eines Spiels muss die Gesamtwirkung des „gamifizierten“ Produkts aufgrund der nicht klar vorhersagbaren Wirkungen der einzelnen Elemente im Zusammenspiel jeweils getestet und gegebenenfalls angepasst werden. Dies ist ein nicht-automatisierbarer Prozess, der menschliche Intuition verlangt.

Gamification-Bausteine werden von Drittanbietern als Service zur Verfügung gestellt. Ein bekannter Anbieter ist die Firma *Badgeville*, die als SaaS²³ eine Plattform²⁴ gleichen Namens bereitstellt. Mit Hilfe dieser Software können Aktionen des Benutzers gruppiert und als Missionen mit Belohnungen verknüpft werden sowie der Status des Benutzers im Sinne von verliehenen Orden, erreichten Punkten und Leveln dargestellt werden.

*OpenBadges*²⁵ der *Mozilla Foundation* ist eine weitere Plattform, die Gamification-Elemente unterstützt. Sie geht davon aus, das Lernen lebenslang stattfindet und nicht nur in formalen Kontexten wie Schule und Studium. Daher bietet sie die Möglichkeit, erreichte Abschlüsse und Qualifikationen in digitaler Form zu verwalten. Sie fungiert als Drittanbieter, der die Zertifikate von herausgebenden Instituten entgegennimmt und sie den Lernenden zuordnet und für diese anzeigt.

Gamification kann in der digitalen Form eine wertvolle Bereicherung von Online-Lernangeboten sein: Die zugrundeliegenden Lern-Management-Systeme sind Software-Systeme, die leicht um Gamification Elemente angereichert werden können. Das Portal *Khan Academy*²⁶ ist ein Beispiel für eine internetbasierte Lernumgebung mit integrierten Gamification-Elementen. Muntean (2011) hat zum Einsatz von Gamification-Elementen in E-Learning-Umgebungen ein Konzept erstellt und kommt zu dem Schluss, dass formale Lernangebote durch Gamification nicht zu einem Spiel werden, dass aber Gamification ergänzend für eine Erhöhung der Lernmotivation sorgen kann - was allerdings noch nachzuweisen ist.

2.2 Simulation in der Bauphysik

2.2.1 Softwarepakete

Eine nicht repräsentative Stichprobe der am Lehrstuhl Bauphysik der Bauhaus-Universität Weimar eingesetzten Simulationssoftware ist in der folgenden Tabelle dargestellt:

Name	Beschreibung	Teilgebiete	Referenz
BASTIAN	BASTIAN ist eine Software-Lösung zur Berechnung der Luft- und Trittschallübertragung innerhalb von Gebäuden, sowie der Schallübertragung von Außengeräuschen.	Akustik	(DataKustik, 2012)
ESP-r	Simulationsprogramm für Gebäude, mit dem die Teilgebiete Wärme, Tageslicht und Akustik untersucht werden können.	Akustik, Tageslicht, Wärme	(J.A. Clarke, ESRU University of Strathclyde, 2012)

²³ *Software as a Service* (SaaS) ist ein Ansatz der Softwarebereitstellung, bei der die Software über ein Kommunikationsnetz bei Bedarf zur Verfügung gestellt wird. Die Verarbeitung erfolgt gewöhnlich auf Rechner des Softwareanbieters.

²⁴ www.badgeville.com, zuletzt abgerufen am 8.8.2012

²⁵ www.openbadges.org/en-US/, zuletzt abgerufen am 8.8.2012

²⁶ www.khanacademy.org, zuletzt abgerufen am 9.8.2012

	Existiert seit den 70er Jahren.		
HEAT3	Berechnung von Wärmestrom und Temperatur in Bauteilkonstruktionen	Wärme	(Blocon, 2012)
STAR-CCM+	Numerische Strömungssimulation	Behaglichkeit	(CD-adpco, 2012)
Therm	Berechnung von Wärmestrom und Temperatur in Bauteilkonstruktionen, kostenlos, einfache Bedienung	Wärme	(LBNL, 2012)
TRNSYS	TRNSYS ist ein Werkzeug zur Simulation von energetischen Prozessen in Anlagen und Gebäuden, programmiert in FORTRAN und erweiterbar. Existiert seit 1975.	Wärme	(S. A. Klein et al., 2010)
WUFI	Software zur Berechnung des gekoppelten Wärme- und Feuchtetransports in Bauteilen	Feuchtigkeit, Schimmel	(IBP, 2012)

Tabelle 1: Auswahl von Softwarepaketen zur Simulation bauphysikalischer Probleme

Eine Betrachtung der Fachgebiete der genannten Simulationsprogramme ergibt die - durch das Themengebiet letztendlich nicht überraschende - Gemeinsamkeit, dass gewöhnlich physikalische Phänomene simuliert werden: Wärmetransportvorgänge, Licht, Strömungen und Feuchtigkeit. Vorherrschend ist daher kontinuierliche Simulation. Für eine zu erstellende bauphysikalische Spielplattform folgt daraus, dass sie kontinuierliche Simulation unterstützen muss.

2.2.2 Stationäre und instationäre Prozesse

Eine häufig anzutreffende Unterscheidung von Berechnungen in der Bauphysik ist die mögliche zeitliche Änderung von Parametern: Berechnungsmodelle, deren Eingabeparameter sich mit der Zeit ändern und damit auch einen zeitlich variablen Systemzustand aufweisen, heißen *instationär*. Berechnungsmodelle mit über den Zeitverlauf konstanten Eingabeparametern hingegen werden *stationär* genannt. Oftmals erfordern instationäre Modelle zur Lösung Differentialgleichungen und deren aufwändige numerische Lösung. Stationäre Modelle besitzen hingegen häufig analytische Lösungen, die mit weniger Aufwand bestimmt werden können.

In Anhang D werden zwei Softwarepakete zur Lösung bauphysikalischer Problemstellungen betrachtet. Das Programm *TRNSYS* dient vorzugsweise der exakten und aufwändigen Simulation instationärer Modelle. Hingegen benutzt *CASAnova* stationäre Modelle. Es setzt keine numerischen Lösungsverfahren ein, sondern setzt die Änderung der Eingabeparameter durch den Nutzer mit Hilfe von einfachen Berechnungsvorschriften augenblicklich um.

2.3 Instructional Design

Unter dem Begriff *Instructional Design* verstehen Niegemann et al. (2008, S. 17) „die systematische und vor allem die differenzierte Anwendung pädagogisch-psychologischer Prinzipien bei der Konzeption von Lerngelegenheiten bzw. Lernumgebungen“. Nach Reiser & Dempsey (2007) wird mit *Instructional Design* das methodische Vorgehen beim Entwurf von Lerneinheiten bezeichnet. Besonderes Ziel ist dabei die Förderung der Lerneffizienz sowie die Erreichung eines tieferen Verständnisses der Lerninhalte. M. Merrill, Drake & Lacy (1966) definieren *Instructional Design* als „a technology for the development of learning experiences and environments which promote the acquisition of specific knowledge and skill by students“. Wenngleich der Begriff des

Instructional Designs eher mit traditionellen pädagogischen Methoden verknüpft ist, so deuten die Begriffe „Lerngelegenheit“, „tieferes Verständnis der Lerninhalte“ und „learning experiences“ doch darauf hin, dass auch Computerspiele per Definition als Objekte des Instructional Designs angesehen werden können. Im Folgenden wird der Begriff des Lernziels erörtert, seine Stellung innerhalb des Instructional Designs durch Beispiele verdeutlicht und eine kritische Einschätzung der isolierten Lernziele durch van Merriënboer's 4C/ID-Modell gegeben.

2.3.1 Lernziele und Lernzieltaxonomien

Die Spielplattform soll gezielt dazu genutzt werden, das Wissen bestimmter Sachgebiete zu vermitteln. Um dieses Ziel zu erreichen, ist es sehr hilfreich, das Wissen zu strukturieren. Die Strukturierung kann mit Hilfe des Begriffes des Lernzieles erfolgen. Ein Lernziel besteht aus einer konkret formulierte Definition von Wissen. Bei Erreichung des Lernzieles ist der Lernende in der Lage, dieses Wissen anzuwenden.

Eine Strukturierung des Wissens in Lernziele hat mehrere Vorteile: Zunächst ist damit das gewünschte Wissen überhaupt dokumentiert und definiert. Ohne Beschreibung bleibt es im Vagen und ist nicht handhabbar. Damit ist der Einsatz von Lernzielen Voraussetzung für die strukturierte Vermittlung von Lerninhalten. Desweiteren lässt sich durch Benutzung von Lernzielen eine Abstraktionsschicht erzeugen, die sich beim Erstellen von Lernmaterialien - in diesem Kontext sind das Spielszenarien - nutzen lässt. Die Struktur von Spielszenarien muss nicht mit konkreten Inhalten konzipiert werden, sondern es kann mit Hilfe von abstrakten Lernzieltypen ein Rahmenwerk konstruiert werden. Dieses kann später zur Darstellung auf eine Vielzahl von konkreten Inhalten angewendet werden.

Ein weiterer Vorteil der Definition von Lernzielen ist, dass sie als Maßstab für die Festlegung von Belohnungen in einem Spiel genutzt werden können. „Belohne, was gelernt werden soll“ ist eine gängige Heuristik, die für den Einsatz von Belohnungen gilt²⁷. Die Festlegung von Lernzielen ist damit auch Grundlage der Definition entsprechender Belohnungen.

Es gibt verschiedene Arbeiten, die eine Einteilung von Lernzielen vornehmen. In der Regel besitzen diese eine kognitive und eine faktische Dimension. In der kognitiven Dimension wird definiert, bis zu welchem Grad eine Fähigkeit gelernt werden kann. Die faktische Dimension unterteilt die Fähigkeiten bezüglich ihrer Inhalte.

2.3.2 Lernzieltaxonomie nach Bloom

Eine sehr häufig benutzte Lernzieltaxonomie wurde von Benjamin S. Bloom entwickelt (1956). Sie besitzt neben der kognitiven auch eine affektive und eine psychomotorische Dimension. Die kognitive Dimension kennt die folgenden - nach ansteigendem Schwierigkeitsgrad geordneten - Ziele:

1. Wissen
2. Verstehen
3. Anwenden
4. Analyse

²⁷ Persönliche E-Mail-Kommunikation (Okt. 2010) mit Dan Norton, Creative Director bei Filament Games, ein auf die Herstellung von Lernspielen spezialisiertes Unternehmen (Madison, WI, USA) (<http://www.filamentgames.com/users/dnorton>, letzter Zugriff 18.07.2012)

5. Synthese
6. Evaluation

Anderson & Krathwohl (2001) haben diese Lernziele in einer Überarbeitung der Bloom'schen Taxonomie um eine faktische Dimension erweitert²⁸. Die faktische Dimension strukturiert Wissen in die folgenden Kategorien:

Kategorie	Beispiele
Faktenwissen	Fachliches Vokabular; Grundlagen, die zum Problemverständnis in einer Disziplin notwendig sind.
Konzeptionelles Wissen	Prinzipien, Klassifikationen, Theorien, Modelle
Prozedurales Wissen	Wissen, das hilft, Aufgaben der Disziplin durchzuführen. Dieses können Methoden und Techniken sein.
Metakognitives Wissen	Bewusstsein der eigenen Wahrnehmung, Wissen über Problemlösungsprozesse

Tabelle 2: Faktische Dimension der überarbeiteten Lernzieltaxonomie nach Bloom

Es ergibt sich die in Tabelle 3 dargestellte Matrix:

		Kognitive Prozesse					
		Erinnern	Verstehen	Anwenden	Analysieren	Bewerten	Erzeugen
Wissen	Faktisch						
	Konzeptionell						
	Prozedural						
	Metakognitiv						

Tabelle 3: Lernzieltaxonomie nach Anderson & Krathwohl (2001)

Baumgartner (2011, S. 41-42) erläutert die Anwendung dieses Modells: Lernziele können als Sätze gebildet werden: Die kognitiven Prozesse entsprechen den Handlungen und sind daher als Verb zu sehen. Dieses Verb wird auf ein Substantiv - eingegliedert in die Kategorien des Wissens - angewendet.

Das Lernziel „Die Studenten können verschiedene Materialien nennen, die zur Wärmedämmung genutzt werden“ kann eingeordnet werden als „Erinnern von faktischen Wissen“: „Nennen“ schließt „Erinnern“ ein und „Wärmedämmmaterialien“ gehört zu den Grundlagen des Fachgebietes der Bauphysik. Das Lernziel „Der Student kann die verschiedenen Arten der Wärmeübertragung darstellen“ hingegen ist der Kategorie des „Erinnerns von konzeptionellen Wissens“ zuzuordnen: Es wird nach einer Systematik gefragt, und eine solche gehört zum konzeptionellen Wissen.

Die obigen Beispiele zeigen die Anwendbarkeit der Taxonomie, jedoch ist es in der Praxis nicht immer leicht, eine Zuordnung vorzunehmen. Daher werden in Anderson & Krathwohl (2001) Beispiele und Schlüsselworte gegeben, die zur Zuordnung genutzt werden können.

Anschauliche Darstellungen dieser Lernzieltaxonomie werden auch von Kharbach (2011) und Herr (2009) gegeben.

²⁸ Neben der Erweiterung um die faktische Dimension schlagen sie weitere Änderungen vor: Unter anderem werden die Lernziele durch Verben ausgedrückt, des Weiteren wurden die beiden obersten Stufen miteinander vertauscht: Dem Erzeugen wird der höchste kognitive Schwierigkeitsgrad zugesprochen.

2.3.3 Component Display Theory (CDT)

Merrill entwickelt die *Component Display Theory* zur Anwendung im Rahmen des Instructional Designs. Im Rahmen dieser Theorie existiert eine ähnliche Lernzieltaxonomie wie die oben beschriebene. In Abbildung 3 ist dargestellt, dass es auch in dieser Taxonomie sowohl eine Dimension zur Aufnahme der Inhaltstypen gibt als auch eine Dimension, in der die unterschiedlichen kognitiven Prozesse dargestellt werden („Performance-Content Matrix“) (M. David Merrill, 1994, S. 112).

Anforderungsniveau	Finden				
	Benutzen				
	Erinnern (generell)				
	Erinnern (Beispiel)				
		Fakt	Konzept	Prozedur	Prinzip
		Inhaltstypen			

Abbildung 3: CDT: Matrix von Inhaltstyp und kognitivem Anforderungsniveau

Auch wenn die Darstellung hier nur gekürzt ist, so lässt sich diese Theorie als weiterer Hinweis werten, dass sich Lernziele in der Substantiv-Verb-Form ausdrücken lassen.

2.3.4 Four-Component Instructional Design Model (4C/ID)

Das 4C/ID-Modell wurde von Merriënboer, Jelsma & Paas entwickelt (1992). Es dient der Vermittlung komplexer kognitiver Fähigkeiten. Dabei benutzt es konkrete, authentische und komplette Szenarien zur Erzeugung von Lernerfahrungen, es wird vom Einfachen zum Komplexen vorgegangen - auf ähnliche Mechanismen wird auch die Wissensvermittlung mit Hilfe von Computerspielen zurückgeführt (Squire, 2011). Merriënboer wendet sich gegen eine kleinteilige Vermittlung von isolierten Lernzielen, er benennt die Probleme Fragmentierung und Abschottung („compartmentalization“) bei der fein strukturierten Vermittlung von Lernzielen. Er führt als Nachteil einer sequentiellen Vermittlung der Fähigkeiten schlechtere Transferergebnisse des Gelernten an („Transfer Paradox“) und befürwortet einen ganzheitlichen Ansatz (Merriënboer & Kirschner, 2007). Das 4C/ID-Modell ist weniger von Lernzielen getrieben als von authentischen Lernsituationen - womit es implizit die Lernmöglichkeiten innerhalb von Computerspielen - sofern sie authentische Erfahrungsmöglichkeiten bereitstellen - bestätigt.

3 Spielplattform

Dieses Kapitel beschreibt ein Rahmenkonzept für die zu entwickelnde Spielplattform. Auf der Grundlage des Rahmens werden in den zwei folgenden Hauptkapiteln (*Kap. 4 Simulationskern* sowie *Kap. 5 Szenarien ausgewählter bauphysikalischer Probleme*) zwei wesentliche Teilkonzepte vertieft dargestellt. Das Rahmenkonzept enthält neben einem groben Spielkonzept eine Beschreibung der Ansätze zur Förderung des gemeinschaftlichen, herausfordernden und konstruktiven Spiels. Ebenfalls werden die technischen Grundlagen in Form der verwendeten Software sowie einer Architektur des Systems dokumentiert.

3.1 Funktion

Die Spielplattform soll als zusätzliches Angebot in der formalen bauphysikalischen Ausbildung eingesetzt werden. Sie ist keinesfalls als allumfassendes Lernmedium zu sehen, dass die bisher üblichen Medien ersetzt. In dieser ergänzenden und begleitenden Funktion ist es die Aufgabe der Plattform, spielerisch den Lerner mit Themen der Bauphysik in Kontakt zu bringen. Möglichst viele der Spieler sollen motiviert werden, sich mit auf der Plattform dargebotene Problemstellungen zu beschäftigen und Lösungen zu finden. Durch die Beschäftigung mit den in Szenarien enthaltenen Problemen lernen sie und sollen zu eigenen Recherchen bezüglich der dargebotenen Sachverhalte animiert werden. Squire (2011) nutzt auch den Begriff des „Interest Capturing“. Als ein Beispiel ist die Mission „EnEV 2009: Transmissionswärmeverluste der Außenhülle“ zu sehen: dem Spieler wird der Begriff *EnEV 2009* präsentiert. Diesen Begriff kennt er möglicherweise bereits. Aber auch wenn er ihn nicht kennt, so kann er sich über seine Mitspieler, durch eine Internetrecherche oder das Anklicken des mitgelieferten Links kundig machen. Dieselbe Methodik hilft ihm, die Anforderungen der Mission zu erfüllen - sofern er sich seinem Ziel nicht durch bloßes Ausprobieren nähert: in einer außerhalb der Plattform zu erstellenden Berechnung - z.B. unter Nutzung einer Tabellenkalkulationssoftware - bestimmt der Spieler die notwendigen Parameter des Simulationsmodells. Die Ergebnisse der Rechnung erlauben es ihm, das Szenario zielgerichtet so zu verändern - u.a. vielleicht durch Austausch der Außenwände oder Anbringen einer Dämmung -, dass es die Zielbedingungen der Mission erfüllt²⁹. Der Spieler hat damit eine Aufgabe mit hohem Bezug zur Realität erfolgreich lösen können. Er konnte spielerisch mit Hilfe einer Simulation Fähigkeiten erwerben bzw. vertiefen, die sich höchstwahrscheinlich auch im realen Leben anwenden lassen.

3.2 Spielkonzept

Die Plattform konfrontiert den Spieler mit in der Bauphysik angesiedelten Problemstellungen (Söbke, Hadlich et al., 2011). Dazu befindet sich der Spieler in der virtuellen Welt einer Stadt - er nimmt die Rolle eines bauphysikalischen Sachverständigen mit einem eigenen Büro ein. Innerhalb der Stadt kann er in verschiedene weitere virtuelle Welten eintauchen: seine Beratungsaufträge. Diese virtuellen Welten werden Szenarios genannt und sind jeweils relativ begrenzt und in sich abgeschlossen. In einem Szenario sind verschiedene Simulationsobjekte so miteinander kombiniert, dass sich daraus Problemstellungen entwickeln lassen. Die Problemstellungen werden dem Spieler in Form von Missionen mitgeteilt.

²⁹ Ein derartiges Vorgehen von Spielern wurde u.a. nachgewiesen von Steinkuehler & Duncan (2008).

Ein typisches Szenario ist das eines Hauses mit mehreren Wohnungen. In diese Wohnungen ziehen Mieter ein, die unterschiedliche Behaglichkeitsbedürfnisse besitzen. Die Mieter sind als NPCs³⁰ modelliert und werden automatisch generiert³¹. Die Aufgabe des Spielers ist es, die Wohnungen in einem Zustand zu halten bzw. zu bringen, der die Mieter zufriedenstellt. Sind die Mieter über längere Zeit unzufrieden, reagieren sie mit Auszug. Unbewohnte Wohnungen sind jedoch für den Spieler schlecht, da ihm dann die Mietzahlungen fehlen, die für die Unterhaltung der Wohnungen benötigt werden. Der Spieler befindet sich in dem Konflikt, die Wohnungen bewohnbar zu halten und gleichzeitig durch die Mieten das notwendige Geld für die Unterhaltung einnehmen zu müssen - ohne zahlende Mieter ist dieses unmöglich.

Die Steuerung der Spieler erfolgt durch einen adaptiven Skillmanager (s. Kap. 4.4.2). Dieser gewährleistet, dass dem Spieler die für seinen Wissensstand geeigneten Szenarios angeboten werden. Zu jedem Szenario sind die Voraussetzungen in Form von notwendigen Fähigkeitspunkten definiert. Damit können diejenigen Szenarios mit zu hohen Voraussetzungen schon aussortiert werden. Gleichzeitig wird dafür Sorge getragen, dass Szenarios mit zu einfachen Voraussetzungen ebenfalls nicht angeboten werden. Dieses Verhalten soll dazu führen, dass der Spieler weder unterfordert noch überfordert ist. Der erste Fall würde mit Langeweile einhergehen, der zweite Fall mit Frustration und Angst. Das Ziel aber ist ein Zustand zwischen Langeweile und Frustration, dem der völligen Konzentration auf die Arbeit und dem Eintauchen in die Problemstellung („Immersion“). Csikszentmihalyi (1990) nennt diesen Zustand „Flow“. Das Zeitgefühl verändert sich und ein Gefühl der Mühelosigkeit stellt sich ein - optimale Bedingungen für die Beschäftigung mit den gebotenen Problemen eines Spiels.

Fachliche Lerninhalte werden vermittelt, indem die Häuser entsprechend präpariert werden (beispielsweise werden Wärmebrücken versteckt oder andere Baumängel eingebaut). Gleichzeitig lassen sich auch die Vorteile einer Simulation nutzen: Es können Systeme simuliert werden, die es in der Realität nicht gibt, die aber dennoch Sachverhalte gut demonstrieren können, da sie überzeichnet werden. Beispielhaft kann ein Haus in die Umgebungsbedingungen der Antarktis eingebettet werden, um das Verhalten verschiedener Dämmungen besser untersuchen zu können.

Wesentliches Merkmal eines Mehrspielerspiels sind die Interaktionen der Spieler. Sie fördern den Aufbau einer von Lave & Wenger (1991) beschriebenen *Community of Practice*. Die zu erstellende Spielplattform bietet verschiedene Möglichkeiten, die Bildung einer solchen Gemeinschaft zu unterstützen und anzuregen:

³⁰ NPC - Non-Player Character - so werden alle Figuren in einem Spiel genannt, die nicht von einem Spieler, sondern automatisiert gesteuert werden.

³¹ Die Generierung von Mietern ist ein Teil der Erzeugung von authentischen Lernkontexten. In der dabei behandelten Fragestellung geht es darum, wie die Eigenschaften von Mietern aussehen müssen, um zum einen ein wirklichkeitsrelevantes Bild zu vermitteln, zum anderen aber auch die Grundlage von Konflikten zu bilden. Die Lösung der entstehenden Konflikte soll zu einem Lernergebnis beim Spieler führen. Die für diesen Prozess maßgeblichen Faktoren untersucht Thomas Bröker in seiner parallel entstehenden Dissertation und den entsprechenden Vorarbeiten (Bröker & Kornadt, 2009; Bröker et al., 2011; Bröker, 2013).

- **Gemeinsamer Problemraum:** Die zu lösenden Probleme sind für alle Spieler gleich. Daher gibt es grundsätzlich einen Druck, durch gemeinsame Aktivitäten, z.B. die Kommunikation über informelle³² Wege, eine Lösung voranzutreiben und zu fördern.
- **Kommunikationswege der Plattform:** Die Plattform bietet durch die Möglichkeit des Chats und dem Senden von E-Mails den Spielern einen einfachen Weg, miteinander in Kontakt zu treten, um Probleme des Spiels zu erörtern. Gleichzeitig kann durch Kommunikation über private Themen auch der persönliche Zusammenhalt zwischen den Spielern erhöht werden.
- **Gruppenbildung:** Die Einführung einer Gruppenstruktur baut Kommunikationshemmnisse zwischen den Spielern ebenfalls ab³³ und führt insgesamt zu einer erhöhten Leistungsfähigkeit der Spieler einer Gruppe.
- **Spielmechaniken:** Mit Hilfe von Spielmechaniken können die Spieler zu gemeinsamen Interaktionen ermuntert werden. So kann beispielsweise die Lösung von Missionen eine höhere Belohnung ergeben, wenn Mitspieler auf Einladung die getroffenen Maßnahmen als sinnvoll bestätigen.
- **Mehrere Rollen eines Spielers:** In erster Linie agiert der Spieler aus der Sicht des Gebäudeverantwortlichen. Es besteht aber gleichzeitig die Möglichkeit, als Mieter aufzutreten: Es kann ein eigenes Profil definiert werden und der Spieler wird - sofern eine Wohnung frei ist³⁴ - Mieter bei einem anderen Spieler. Dieses Vorgehen führt zu benutzerdefinierten Problemen - hier ist die Plattform nicht auf programmierte „Intelligenz“ angewiesen, sondern es treten Spieler gegen Spieler an.

³² Informell ist hier so zu verstehen, dass die Kommunikationswege nicht durch das Spiel bereitgestellt werden. Beispiele sind selbsteingerichtete Foren und Wikis.

³³ Das ist der Eindruck des Autors: Er spielte als Teil der Arbeit das Social Game „Fliplife“. Dort gab es immer einen Chat, in dem jeder Spieler mit allen anderen kommunizieren kann. Eine ernsthafte Kommunikation mit anderen Spielern kam für den Autor jedoch erst dann zustande, als Abteilungen mit einem eigenen Chat eingeführt wurden. In Abteilungen, die typischerweise aus 10 bis 25 Spielern bestehen, kennen die Spieler einander - zumindest vom Namen her - gegenseitig. Das führt zu einem geänderten Chatverhalten. Diese Feststellungen sind jedoch bisher unveröffentlicht und nicht quantitativ nachgewiesen. In seinem Buch „Tipping Point: Wie kleine Dinge Großes bewirken können“ stellt Malcolm Gladwell die Regel der 150 auf: 150 ist die maximale Zahl an Menschen, zu denen eine Person bedeutungsvolle soziale Beziehungen aufrechterhalten kann - Grundlagen für diese These könnten zu dem beobachteten Verhalten beitragen.

³⁴ Diese Spielmechanik geht einher mit der Möglichkeit, dass der Spieler bestimmte gewählte Szenarien als dauerhaft kennzeichnen kann. Normale Szenarien gelten nach der Erfüllung der Missionen als abgeschlossen und haben - abgesehen von den erreichten Belohnungen - keinen weiteren Einfluss auf den Status des Spielers.

3.3 Systemaufbau

3.3.1 Architektur

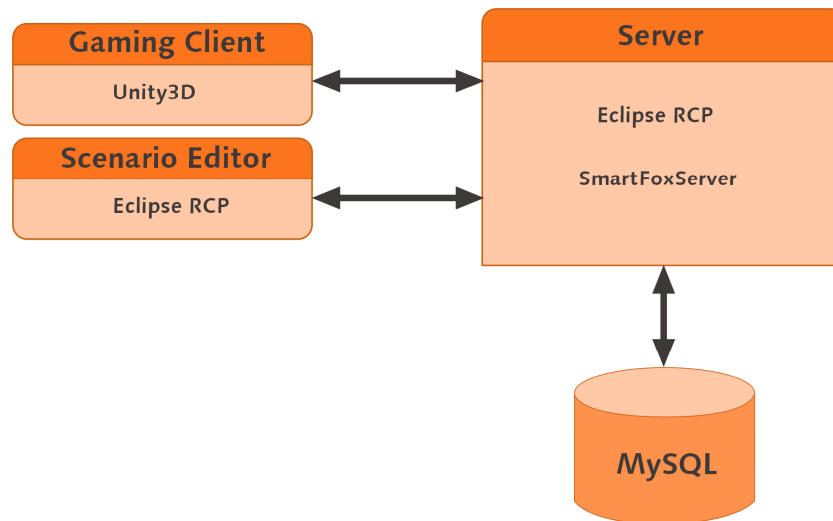


Abbildung 4: Architektur des Systems

Die Architektur des Systems - dargestellt in Abbildung 4 - entspricht einer im MMOG-Bereich üblichen Client-Server-Architektur: Es gibt eine Serverkomponente, die mit einer Datenbank verbunden ist ebenso wie Clients mit verschiedenen Funktionen: Zum einen ist dies der Gaming Client, in dem gespielt werden kann, zum anderen der sogenannte Szenario-Editor, mit dem Spiel-Szenarien durch den Spieler angelegt und verändert werden können.

a) Server

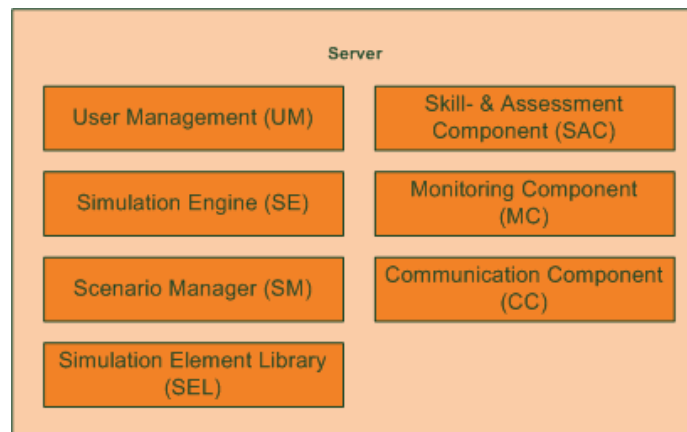


Abbildung 5: Server-Komponenten

Der Server ist die zentrale Komponente der Architektur. Wie Abbildung 5 zeigt, stellt er verschiedene Dienste für die Clients bereit:

- **User Management (UM):** Jeder Spieler wird durch den Server registriert und authentifiziert. Auch werden die Spielerdaten verwaltet: Damit der Spieler von verschiedenen Rechnern aus spielen kann, werden die von ihm erzeugten Daten auf dem Server gespeichert. Das sind beispielsweise Zustandsdaten unterbrochener Simulationen und Errungenschaften („Achievements“).
- **Simulation Engine (SE):** Die Ausführung der Simulationsläufe unter Verwendung der Simulation Engine ist eine wichtige Aufgabe des Servers.

- **Scenario Manager (SM):** Die verschiedenen Szenarios werden durch den Server verwaltet und dem Spieler passend zur Lösung angeboten.
- **Simulation Element Library (SEL):** Neben den Szenarios kann der Spieler auch selbst Simulationselemente erzeugen. Diese werden auf dem Server in der Simulation Element Library vorgehalten und verwaltet.
- **Skill- & Assessment Component (SAC):** Eine wesentliche Aufgabe der Spielplattform ist es, zum Lernfortschritt der Spieler beizutragen. Zur Unterstützung dieses Ziels werden auf dem Server die erreichbaren Fähigkeiten (*Skills*) sowie die jeweils erreichte Punktzahl verwaltet (Held, 2011).
- **Monitoring Component (MC):** Mit Hilfe der Monitoring Component ist es möglich, das Verhalten des Spielers zu beobachten. Es kann eine statistische Auswertung des Spielerverhaltens erfolgen, diese kann als Feedback für das Design weiterer Szenarios benutzt werden.
- **Communication Component (CC):** Der Server dient als zentrale Kommunikationsplattform. Das vermeidet die Notwendigkeit einer direkten Kommunikation der Clients untereinander.

Die **Aufgabenverteilung** von Server und Client festzulegen, ist eine Designentscheidung. Ein Ansatz, die Aufgabenverteilung zu kategorisieren, unterscheidet zwischen *Fat Client* und *Thin Client*. Der Fat Client enthält neben der Präsentation auch die Anwendungslogik, der Server wird gewöhnlich nur zur Speicherung der Daten benutzt. Somit skaliert die Architektur besser, d.h. bei mehr hinzukommenden Clients wird die Verarbeitung durch den Client geleistet und führt nicht zu einer höheren Serverlast. Nachteilig sind erhöhte Erstellungs- und Aktualisierungsaufwände für den Client ("ITWissen," 2012).

Für den im Rahmen dieser Arbeit entwickelten Prototyp ist zunächst nicht mit großen Benutzerzahlen zu rechnen. Um auf der einen Seite die Machbarkeit eines modularen, auf Erweiterbarkeit beruhenden Ansatzes zu demonstrieren, der mit Hilfe der Software Eclipse RCP auf Serverseite realisiert wird und auf der anderen Seite den erhöhten Aufwand zur Erstellung eines sehr komplexen Clients zu sparen, wird die Anwendungslogik soweit wie möglich auf den Server verlagert.

b) Server: Simulationskern

Die bestimmende Serverkomponente ist der Simulationskern („Simulation Engine“). Die Aufgabe des Simulationskerns ist die Durchführung von Simulationsspielszenarien. Ein Framework steuert deren Ablauf und ermöglicht die folgenden Besonderheiten:

- **Hybride Simulation:** Die zu simulierenden physikalischen Prozesse erfordern kontinuierliche Simulationsmechaniken. Auf der anderen Seite gibt es Ereignisse, die zufallsgesteuert oder durch den Spieler ausgelöst werden können und zu einer diskreten Änderung des Simulationsmodells führen. Damit sind sowohl kontinuierliche als auch diskrete Simulationsmechaniken gegeben.
- **Benutzeraktionen:** Grundlage der Simulation sind Spielszenarien. Spielen geht einher mit Aktionen des Spielers. Somit muss das Simulationsframework plötzliche Änderungen des Simulationsmodells unterstützen.
- **Zufallsgenerierte Ereignisse:** Spielszenarien leben häufig davon, dass das System ein nicht-deterministisches Verhalten zeigt, ein Beispiel hierfür ist ein NPC. Daher unterstützt der Simulationskern zufällig auftretende Ereignisse. Dem zu definierenden Ereignis kann eine Wahrscheinlichkeit zugeordnet werden, mit dem das Ereignis auftritt.

- **Zufallsgenerierte Modellstrukturen und Variablenwerte:** Nicht nur zufallsgenerierte Ereignisse sorgen für eine erhöhte Variabilität der Szenarien, sondern auch die durch Zufall gesteuerte Erzeugung von initialen Modellen (beispielsweise die Anzahl der Etagen eines betrachteten Gebäudes) und von initialen Werten für Parametern.
- **Erweiterbare Modelle:** Wesentliches Merkmal des Frameworks ist die Nutzung des Prinzips der Erweiterbarkeit: Ein Modell kann um die für die jeweilige Simulation wichtigen Aspekte erweitert werden. Das kann die Modellstruktur betreffen, indem neue Objekte ergänzt werden, es können aber auch zusätzliche Zustandsvariablen und Attribute in Form von Parametern hinzugefügt werden. Erweiterbarkeit bezieht sich auf alle Elemente eines Szenarios, so sind auch Systemereignisse und Benutzeraktionen hinzufügbare.
- **Modularer Aufbau:** Das bestimmende Strukturierungsmerkmal des Simulationskerns ist das Modul. Ein Modul fasst die Simulationselemente eines technischen oder fachlichen Aspektes zusammen. Zu einem Simulationsszenario gehört immer die Auswahl der Module, aus denen es besteht. Mit den Simulationselementen der ausgewählten Module kann das Simulationsszenario erweitert werden. Zu einem Modul gehört auch die Definition der vorausgesetzten Module, also der Module, auf denen es aufbaut.

Vorteil des Modulansatzes ist die verbesserte Wartbarkeit: Stellen sich die Elemente eines Modules als änderungsbedürftig heraus, so kann innerhalb des Moduls mit beschränkten Auswirkungen geändert werden oder aber das Modul mit begrenztem Aufwand komplett neu entwickelt werden. Unterstützt wird dieses durch spezielle Modul-Tests, die das Verhalten des Modules dokumentieren und fixieren.

Ein weiterer Vorteil des Modulansatzes aus lerntheoretischer Sicht ist die über die Auswahl der Module steuerbare Komplexität des Szenarios: Je nach den Aspekten, die das Szenario demonstrieren soll, können Module mit verschiedenen fachlichen Inhalten hinzugefügt oder entfernt werden.

c) Gaming Client

Der Gaming Client stellt die Umgebung dar, in der sich der Spieler bewegt: Die Spielfläche soll auf der einen Seite ansprechend sein und zum Spielspaß beitragen. Auf der anderen Seite soll sie funktionell und leicht zu bedienen sein. Um dem Charakter eines Spieles Rechnung zu tragen, soll eine Game Engine die Grundlage bieten. Diese stellt Funktionen bereit, die in Spielen häufig benutzt werden und nicht mehr eigenständig programmiert werden müssen.

Der Gaming Client sollte über einen Webbrowser bedienbar sein. Zum einen lässt sich dadurch der Aufwand einer Aktualisierungsinfrastruktur sparen, da der Webserver immer die aktuellste Version ausliefert, zum anderen kann der Client damit in SNSs eingebettet werden. Diese bieten gewöhnlich schon ein Framework für browserbasierte Spiele.

d) Szenario-Editor

Ein wesentliches Merkmal der zu entwickelnden Spielplattform ist die Einbeziehung der Spieler in die Erzeugung der Inhalte, d.h. es werden Fachwissen und Kreativität der Spieler genutzt, um Szenarien zu erstellen. Um den Spieler dabei zu unterstützen und ihn von Programmierarbeiten zu entlasten, gehört zum Konzept der Plattform ein Szenario-Editor. Der Szenario-Editor ist neben dem Gaming-Client eine eigenständige Anwendung. Sie besitzt eine funktionelle Oberfläche und benötigt keine Game Engine zur Realisierung. Der Szenario-Editor hat den Charakter einer Entwicklungsumgebung (IDE) für Spiele und interagiert dabei mit dem Server. Die großen Aufga-

benblöcke des Szenario-Editors sind zum einen die Entwicklung von Szenarien, auf der anderen Seite können auch Module mit all ihren Inhalten zusammengestellt und geändert werden.

Erstellung eines Szenarios: Zu den Aufgaben, die bei der Erstellung eines Szenarios entstehen, gehören:

- **Auswahl der Module eines Szenarios:** Mit den Modulen wird der fachliche Kontext und damit auch der Schwierigkeitsgrad eines Szenarios festgelegt.
- **Entwurf der Struktur des Simulationsmodells:** Die Module bieten Simulationselemente an. Diese müssen nun zu einem Simulationsmodell arrangiert werden.
- **Definition von Voraussetzungen:** Element der Plattform ist die Skill- & Assessment-Component (SAC). Diese bildet die Fähigkeiten des Spielers ab und hilft bei der Entwicklung dieser Fähigkeiten. Damit dem Spieler immer diejenigen Szenarios angeboten werden, die für ihn spannend und fördernd sind, werden für jedes Szenario die Fähigkeiten festgelegt, die der Spieler bereits besitzen muss, um zum Szenario zugelassen zu werden.
- **Festlegen von Missionen des Szenarios und Belohnungen für den Spieler:** Zu einem Spiel gehören Aufgaben, die der Spieler zu erledigen hat. Diese Aufgaben, die Missionen genannt werden, werden belohnt in Form von Spielwährungen, Gegenständen und Auszeichnungen. Die Belohnungen müssen ebenso festgelegt werden wie der Zuwachs an Fähigkeitspunkten durch erfolgreich erfüllte Aufgaben.

Entwicklung von Modulen: Ein Modul bündelt die Simulationselemente eines speziellen Aspektes. Die Entwicklung eines Modules kann von einem Spieler durchgeführt werden, der bezüglich dieses Aspekts Expertise aufgebaut hat. Die Aufgaben der Modulerstellung umfassen unter anderen die folgenden Punkte:

- **Definition von Simulationselementen:** Zu den Simulationselementen gezählt werden Simulationsobjekte wie beispielsweise ein Fenster oder eine Heizung, Ereignisse wie der Auszug eines Mieters und Benutzeraktionen wie der Austausch von Fenstern.
- **Festlegung der vorausgesetzten Module:** Mitunter gibt es Abhängigkeiten unter den Simulationselementen: So macht die Einführung eines Heizkörpers nur Sinn, wenn auch eine Heizung definiert ist. Daher ist es bei der Modulentwicklung notwendig, die Basis zu definieren, auf der das Modul aufsetzt. Das geschieht in Form anderer, schon existierender Module. Der Nutzen, der aus der Definition von vorausgesetzten Modulen gezogen wird, liegt in der Wiederverwendung: Schon definierte Simulationselemente, die bewährt und ausgetestet sind, können wieder benutzt werden.
- **Definition von Attributen:** Unterschiedliche Simulationsobjekte können dasselbe Attribut besetzen. Als Beispiel sei das Attribut U-Wert angeführt: Sowohl das Simulationsobjekt „Fenster“ als auch das Simulationsobjekt „Wand“ besitzen dieses Attribut. Daher macht es Sinn, dieses Attribut separat mit zugehörigen Eigenschaften wie die Einheit zu definieren und erst später die Zuordnung zu den Simulationsobjekten durchzuführen.
- **Definition von Parameter und deren Berechnungsalgorithmen:** Parameter sind ebenfalls Simulationselementen zugeordnet. Im Unterschied zu Attributen ändern sie jedoch ihren Wert. Die Änderung kann zum einen durch Benutzeraktionen oder Systemevents erfolgen, gewöhnlich jedoch wird der Wert durch einen Berechnungsalgorithmus bestimmt, der dem Parameter zugeordnet ist. Dieser Algorithmus ist durch den Modulentwickler zu definieren. Er hängt stark vom Kontext des Parameters ab, d.h. von den Simulationselementen und de-

ren Parametern, in die er eingebettet ist. Für die Berechnungsalgorithmen gilt das Gebot der Einfachheit: Um Rechenaufwand zu sparen und ein performantes Spielszenario schaffen zu können, sollte jeder Berechnungsalgorithmus einen möglichst einfachen Ansatz benutzen. Es sollte beispielsweise nicht versucht werden, Differentialgleichungen numerisch zu lösen.

Testumgebung: Der Szenario-Editor ermöglicht in seiner Funktion als IDE auch den Test von Modulen und Szenarien. Dazu können Simulationsläufe durchgeführt werden, es treten dabei Systemereignisse auf und der Szenarioentwickler kann in seiner Funktion als Spieler auch Aktionen auswählen und durchführen.

3.3.2 Software

a) Eclipse RCP

Eclipse ist ursprünglich eine Java Entwicklungsumgebung. Der Vorgänger *Visual Age for Java* wurde von IBM entwickelt. 2001 wurde der Quellcode freigegeben (Beaton, Helming & Kögel, 2011). 2004 wurde dann die Eclipse Foundation gegründet, eine nicht gewinnorientierte Organisation, die für die Weiterentwicklung von Eclipse sorgt und das dabei entstandene sogenannte Ökosystem - so werden die Erweiterungen, die auf Eclipse aufbauen, in ihrer Gesamtheit genannt - pflegt (Eclipse, 2012). Kennzeichen der Architektur von Eclipse ist die konsequente Ausrichtung auf Erweiterbarkeit. „Everything is a contribution“ lautet die erste Regel für die Architektur der Eclipse Plattform, die in der Programmiersprache Java implementiert ist und zur Laufzeit eine JVM³⁵ benötigt. Das Ziel der Erweiterbarkeit führt dazu, dass Eclipse - eine der am weitesten verbreiteten Java-IDEs (Hs, 2009) - auch als eine Plattform für Entwicklungsumgebungen aufgefasst werden kann. So existieren beispielsweise auch Erweiterungen für die Entwicklung mit der Programmiersprache C++ („Eclipse CDT,” 2012) und der Skriptsprache PHP („PHP Development Tools,” 2012).

Als Beispiele für die Integration weiterer Werkzeuge zur Unterstützung des Softwareentwicklungsprozesses seien genannt:

- **Versionsverwaltung:** Hier werden die meisten Versionsverwaltungen unterstützt. Ein Beispiel ist Subversion, das die Versionsverwaltung Subversion in Eclipse integriert. Mit Hilfe dieser Erweiterung³⁶ ist es beispielsweise möglich, direkt aus der Entwicklungsumgebung heraus eine Datei wieder einzuchecken, d.h. Subversion die Änderungen an einer Datei zu übergeben.
- **Fallbearbeitungssystem**³⁷: *Mylyn* heißt eine sehr bekannte Erweiterung der Eclipse IDE, mit der ein Aufgaben-zentriertes Arbeiten ermöglicht werden soll. Neben der Möglichkeit, den Lebenszyklus eines Falles direkt aus der IDE zu modifizieren (d.h. ihn abzuschließen, weiter-

³⁵ *Java Virtual Machine*, die Laufzeitumgebung für Java-Programme.

³⁶ Eine Erweiterung wird in der Eclipse-Nomenklatur auch Plug-in oder Package genannt. Der Begriff Plug-in hat zwei Bedeutungen: Die erste bezeichnet eine komplette Erweiterung, wie z.B. das Eclipse CDT. Eine Erweiterung besteht aus mindestens einem sogenannten „Feature“, das wiederum mehrere Plug-ins umfassen kann. Plug-in in der zweiten Bedeutung bezeichnet die eher technische Einheit eines Software-Moduls (das Design der Plug-ins erfolgt nach technischen Gesichtspunkten), während Feature die semantische, installierbare Einheit ist (McAffer & Lemieux, 2005; Pat McCarthy (SDI Corp.), 2003).

³⁷ Dieser Begriff ist unter der Bezeichnung *Bugtracker* (von *bug tracking system*) oder *Fehlerbearbeitungssystem* bekannter. Die Bezeichnung ist aber nicht ganz zutreffend, da neben Fehlern auch Anforderungen an die zu entwickelnde Software und Aufgaben für den Entwickler verwaltet werden.

zuleiten oder gar erst zu eröffnen), wird mit Mylyn auch die Entwicklungsumgebung selbst gestaltet: Das Plug-in merkt sich für einen geöffneten Fall die bearbeiteten Dateien und erfasst die geöffneten Fenster. Bei Weiterleitung oder Wiederaufnahme des bearbeiteten Falles werden dann gleich die relevanten Dateien und Informationen hervorgehoben.

Interessant ist der zweistufige Plug-in-Ansatz: Die einzelnen Fälle werden in externen Fallbearbeitungssystemen gespeichert, von denen es eine Vielzahl gibt. Für die Kommunikation von Mylyn mit einem speziellen Fallverwaltungssystem gibt es die sogenannten *Konnektoren*. Die sind wiederum selbst als Plug-in ausgeprägt ("Eclipse Mylyn Open Source Project," 2012).

Nachdem die Eclipse-Plattform mit dem Prinzip der Erweiterbarkeit und als „Open Tools Plattform“ zu einem der bevorzugten Werkzeuge zur Java-Entwicklung aufgestiegen war, wurde eine Untermenge der verwendeten Features identifiziert³⁸. Sie bildet für sich die Basis einer Software, d.h. sie stellt beispielsweise Funktionen für die Oberflächengestaltung und die Fensterverwaltung innerhalb einer grafischen Benutzeroberfläche bereit. Diese - für die Erreichung des genannten Ziels minimale Menge von Plug-ins - wurde *Rich Client Platform (Eclipse RCP)* genannt und ist ein mächtiges Framework zur Erstellung von modularen Anwendungen - denn die Grundprinzipien wie Modularität und Erweiterbarkeit wurden beibehalten.

Wesentlicher Bestandteil der Eclipse-Plattform ist die OSGi-Implementierung *Equinox* - die Basis für das Komponentenmodell und damit die Erweiterbarkeit von Eclipse. OSGi ist ein von der *OSGi Alliance* definierter Standard für ein Java-Komponentenmodell ("OSGi Alliance", 2012). Kennzeichen ist eine Unterteilung der Software in sogenannte Bundles (oder auch Services) sowie das Vorhandensein einer Verwaltungseinheit, der sogenannten Service-Registry. Mit ihrer Hilfe kann der Lebenszyklus von Bundles bei gestarteter Laufzeitumgebung gesteuert werden, d.h. Bundles können installiert, gestartet und wieder gestoppt werden. Dieser Standard wurde für die Eclipse-Plattform, beginnend mit der Version 3.0 im Jahre 2004, unter dem Namen „Equinox“ implementiert ("Equinox," 2012). Für den Simulationskern auf dem Server ist nur der Equinox-Anteil von Eclipse RCP von Bedeutung, nicht aber die GUI-Unterstützung sowie der Update-Mechanismus.

b) Unity3D

Unity3D wird für die Entwicklung der Spieloberfläche genutzt. Bei der Auswahl der Software für die Spieloberfläche wurden die folgenden Kriterien angelegt:

- **Game Engine:** Es gibt spezialisierte Software für die Erstellung von Spielen, die sogenannten *Game Engines*, die viele bei der Erstellung von Spielen anfallenden Aufgaben schon selbst übernehmen bzw. unterstützen, so dass die eigentliche Entwicklung effizienter wird.
- **Simulation von physikalischen Phänomenen:** Mit Hilfe der zu erstellenden Spielplattform sollen bauphysikalische Phänomene präsentiert werden. Daher ist es sinnvoll, die von Game Engines gewöhnlich standardmäßig implementierten physikalischen Effekte aus Bereichen wie Akustik, Licht und der Physik starrer Körper („rigid body mechanics“) in die Darstellung zu integrieren.

³⁸ Eclipse selbst ist gemäß der Einheiten Feature und Plug-in strukturiert, d.h. es ist eine Ansammlung von Features und Plug-ins.

- **Verbreitung:** Die Software sollte etabliert und verbreitet sein. Damit wird sichergestellt, dass es sich um belastbare, fehlerfreie Technik handelt, die es ermöglicht, die Anstrengungen der Entwicklung auf das eigentliche Problem zu konzentrieren und nicht mit technischen Problemen bei der Umsetzung zu kämpfen zu müssen. Zudem ist es leichter, mit der Technik vertraute Entwickler zu finden.
- **Kosten:** Game Engines gibt es in vielen Preislagen. Einige Spielehersteller haben eigene produziert, die sehr effiziente Entwicklung ermöglichen und zur Refinanzierung dann entsprechend teuer verkauft werden. Auf der anderen Seite gibt es freie Software, die tendenziell leistungsmäßig nicht mit den kommerziellen Vertretern mithalten vermag. Die zu entwickelnde Spielplattform ist für den universitären Betrieb gedacht, so dass zumindest vorerst keine nennenswerten Einnahmen zu erwarten sind. Aus diesem Grunde ist es notwendig, bei der Auswahl der Game Engine zum einen auf den Leistungsumfang, zum anderen aber auch auf Finanzierbarkeit zu achten.

Ohne einen formal strukturierten Auswahlprozess durchgeführt zu haben, ist die Wahl auf Unity3D gefallen: Unity3D ist eine - inzwischen - weitverbreitete Game Engine, die vorzugsweise im Bereich kleiner und mittlerer Spielentwicklungsprojekte eingesetzt wird und in den letzten Jahren enorm an Verbreitung gewonnen hat. Die Lizenzkosten liegen im Rahmen des Projektbudgets³⁹ (DeLoura, 2009, 2011).

Unity3D unterstützt neben den Skriptsprachen JavaScript und Boo die Programmiersprache C#, die ähnlich wie Java vom objektorientierten Programmierparadigma geprägt ist sowie eine Laufzeitumgebung benötigt. Für Java heißt die Laufzeitumgebung *Java Virtual Machine* (JVM), C# basiert auf der .NET⁴⁰-Plattform, deren Laufzeitumgebung *Common Language Runtime* (CLR) heißt. Beide Laufzeitumgebungen werden ergänzt durch eine Reihe von Klassenbibliotheken. Vorteile der Laufzeitumgebung ist die Betriebssystem- und Hardwareunabhängigkeit: Programme in den betreffenden Sprachen laufen auf allen Systemen, für die es eine Implementierung der Laufzeitumgebung gibt.

Unity3D erlaubt die Erstellung von sogenannten Webdeployments, d.h. es werden ausführbare Dateien erstellt, die in einem Webbrowser laufen können. Dazu ist die einmalige Installation eines Plug-ins - des sogenannten *Unity Web Players* - für den Webbrowser notwendig.

c) SmartFoxServer

Die zu erstellende Spielplattform ermöglicht die Zusammenarbeit der Spieler. Merkmale dieser Spielplattform sind u.a. die Kommunikation der Spieler untereinander sowie die Persistenz des Spielerstatus. Diese beiden Funktionen werden über den zentralen Server abgewickelt.

Sogenannte *Middleware* stellt eine Vermittlungsschicht zwischen unterschiedlichen Softwaresystemen dar. Es wird zwischen Client und Server „vermittelt“. Zu den Aufgaben der benötigten Software zählt Lastverteilung und Skalierbarkeit, d.h. in Abhängigkeit der Anzahl von Clients können Server hinzu- oder gegebenenfalls auch abgeschaltet werden. Zusätzlich ermöglicht der Einsatz einer Middleware die Abstraktion der Kommunikation aus Entwicklersicht: Bei der Ent-

³⁹ Eine „Unity-Pro“-Lizenz kostet im Unity3D Online Store € 1050, hinzu kommen noch einmal € 350 für die Team-Lizenz, die unter anderem den *Asset Server* enthält, die Unity3D-eigene Versionsverwaltung (Unity, 2012).

⁴⁰ Gesprochen: Dot Net

wicklung ist die zu bearbeitende Einheit eine Anfrage des Clients („request“) und die Antwort des Servers („response“). Diese können vom Entwickler verarbeitet werden, ohne Details des genutzten Kommunikationsprotokolls zu kennen.

Mehr-Spieler-Online-Spiele stellen die Anforderung einer geringen Latenzzeit, d.h. es sollen schnelle Interaktionen der Spieler untereinander in einer gemeinsamen virtuellen Welt möglich sein. Diese Anforderung gilt somit auch für den Game Server bzw. für die verwendete Middleware. Das ist einer der Haupteinsatzgründe für von speziell auf den Einsatz in Spielumgebungen ausgerichteter Middleware (Feng, Chang, Feng & Walpole, 2005).

Bei der Auswahl der Middleware gab es die folgenden Anforderungen:

- **Einsatz in Spielumgebungen:** Wie oben beschrieben, sind für Mehrspieler-Online-Spiele niedrige Latenzzeiten notwendig, um beim Zusammenspiel über das Netz Interaktion in Quasi-Echtzeit zu erlauben.
- **Server: Unterstützung von Eclipse RCP (Java):** Da auf der Serverseite Eclipse RCP eingesetzt werden soll, muss der Server eine Schnittstelle zur Programmiersprache Java unterstützen.
- **Unterstützung von Unity auf der Client-Seite:** Als Client ist die Game Engine Unity3D vorgesehen (s.o.). Da der Client mit dem Server kommunizieren muss, ist sicherzustellen, dass eine Kommunikationsschnittstelle zwischen beiden System-Komponenten unterstützt wird.

Erster Kandidat für diese Software war das Project *Darkstar* der Firma *Sun Microsystems* (Shankland, 2006). Die Weiterentwicklung dieser sich in einem frühen Lebenszyklus befindlichen Software wurde jedoch mit der Übernahme von Sun durch die Firma Oracle Corporation 2010 eingestellt (Oracle, 2009). Es gab dann zwar noch den Versuch der Weiterentwicklung in einem Open Source Projekt *Red Dwarf*, hier wurde die zur Zeit aktuelle Version jedoch auch 2010 fertiggestellt („RedDwarf Server,” 2010) - so dass die Wartung als fraglich angesehen werden kann.

SmartFoxServer von der Firma *gotoAndPlay()* ist eine Java basierte Middleware - nach firmeneigener Aussage zur Zeit „the leading middleware for creating large scale multiplayer games“ („SmartFoxServer,” 2012). Die angegebenen Referenzen (u.a. *Disney's Club Penguin*) unterstützen diese Aussage. Diese Software beherrscht sowohl die Kommunikation mit Unity3D-Clients als auch die Integration von Java-basierter Server-Logik. Zudem ist sie in einer Basis-Version kostenfrei. Diese genügt, um die Anforderungen dieses Projektes abzudecken.

d) XAMPP

Die Spielplattform soll im Internet zur Verfügung stehen, d.h. ihre Benutzung setzt keine lokale Installation voraus⁴¹. Stattdessen wird ein Webserver genutzt, über den auf den Client zugegriffen werden kann. Als Webserver-Software ist *Apache* von der *Apache Foundation* als Marktführer zu sehen (Apache, 2012). Dieser Webserver ist Bestandteil des frei erhältlichen *XAMPP*-Paketes. *XAMPP* ist ein Bündel freier Software für die gängigen Serverbetriebssysteme, das die wichtigsten Anforderungen für den Betrieb einer Website abdeckt („xampp,” 2012). Es vereinfacht den Installations- und Administrationsaufwand der Software. Ebenfalls enthalten ist die weitverbreitete relationale Datenbanksoftware *MySQL*, die zur Speicherung der Spielerdaten genutzt werden kann (MySQL, 2012). Somit bietet es sich an, das *XAMPP*-Paket zu nutzen.

⁴¹ Abgesehen vom Unity Web Player.

e) Systemanforderungen

Die Systemanforderungen des Clients sind gegeben durch die Anforderungen, die die verwendete Clientsoftware von Unity3D - Webplayer genannt - stellt⁴²: Für *MS Windows* als Betriebssystem werden zur Zeit die Versionen *Windows XP* oder *Vista* und die Webbrowser *Internet Explorer*, *Firefox*, *Chrome*, *Safari* oder *Opera* genannt. *Mac OS X* wird ab Version 10.5 unterstützt, die dort möglichen Webbrowser sind *Chrome*, *Firefox* oder *Safari*.

Auf der Serverseite stellt die Software SmartFoxServer die folgenden Anforderungen⁴³:

- **Betriebssystem:** MS Windows , Mac OS X ab Release 10.5, Linux und weitere Unix-Derivate
- **Hauptspeicher:** 512 MByte
- **Prozessor:** 2 GHz Taktfrequenz

XAMP definiert seine Anforderungen unter MS Windows wie folgt⁴⁴:

- **Betriebssystem:** MS Windows , Mac OS X ab Release 10.5, Linux und weitere Unix-Derivate
- **Hauptspeicher:** 512 MByte
- **Festplattenplatz:** 64 MByte
- **Prozessor:** 2 GHz Taktfrequenz

Alle genannten Systemanforderungen sind als eher moderat einzustufen.

⁴² Quelle: <http://unity3d.com/webplayer/>, zuletzt abgerufen am 10.08.2012

⁴³ Quelle: <http://docs2x.smartfoxserver.com/GettingStarted/installation>, zuletzt abgerufen am 10.08.2012

⁴⁴ Quelle: http://www.apachefriends.org/winxampp/readme_en.txt, zuletzt abgerufen am 10.08.2012

4 Simulationskern

4.1 Design-Grundsätze

In diesem Kapitel werden die wesentlichen Paradigmen des Designs der Spielplattform - Modularität und Erweiterbarkeit - auf der Ebene der Implementierung erläutert. Dieses erfolgt u.a. mit Hilfe von Codefragmenten. Dabei erfolgt die Darstellung gemäß den folgenden Konventionen: Es wird eine vereinfachte Quellcodeform genutzt, um die Verständlichkeit des Codes zu erhöhen und die wesentlichen Details besser herausarbeiten zu können: Feinheiten der Implementierung, die zum Verständnis nicht weiter notwendig sind, wurden gekürzt. Zu den Vereinfachungen gehört auch die Weglassung von qualifizierten Namen, wie z.B. `Package`⁴⁵-Namen und Plug-in-Namen sowie von Prozeduren, deren konkreter Inhalt nicht für das Verständnis des Codes wichtig ist. Der dargestellte Quellcode ist damit nicht immer übersetzbar. Der unveränderte, originale Code kann an der in Anhang G genannten Stelle eingesehen werden.

4.1.1 Modularität

Das Entwicklungsergebnis einer in sich abgeschlossenen Teilaufgabe ist ein Modul. In der Eclipse RCP-Terminologie ist dafür auch der Begriff „Plug-in“ gebräuchlich. Für ein Modul ist zu definieren, welche Module vorausgesetzt werden. In der Gesamtheit aller Module ergibt sich aus diesen Abhängigkeiten ein wohldefinierter, zyklenfreier Modulbaum. In Abbildung 6 ist ein solcher dargestellt. Als Voraussetzung einer effizienten Modulentwicklung wurden eine Taxonomie der zu erstellenden Module definiert:

Core Modules

Die Kernmodule stellen die Basis bereit, die - unabhängig von Fachdomänen - Basisklassen und -mechanismen des Frameworks enthält.

Technical Modules

Die Fachmodule enthalten fachspezifische Klassen. Wichtig ist die Tatsache, dass es innerhalb der Fachmodule ebenfalls eine Hierarchie gibt: Um das Beispielszenario zu realisieren, gibt es das Basis-Modul „Gebäude“ (`TenementBuilding`). Dieses definiert die grundlegenden Objekte eines Wohnhauses: „Wohnungen“ (`Tenement`), „Wände“ (`wall`), „Fenster“ (`window`) usw. Auf diesem Modul bauen bauphysikalische Grundlagen-Module auf: „Akustik“ (`Acoustics`), „Wärme“ (`Heat`) und „Feuchtigkeit“ (`Moisture`) sind hier beispielhaft aufgeführt. Auf der gleichen Ebene steht das Modul „Vertrag“ (`TenementContract`), das zwar keine bauphysikalisch relevanten Fakten modelliert, aber unter dem Aspekt der Spielmechanik wichtig ist. Auf der nächsten Ebene steht das Modul „Schimmel“ (`Mildew`). Dieses setzt die Module „Wärme“ und „Feuchtigkeit“ voraus.

Scenario Modules

Szenario-Module bauen Objekte der Klassen aus den Fachmodulen zu konkreten Szenarien zusammen. Dabei wird die Modulhierarchie durch Szenarien-Module wieder zusammengeführt, d.h. während es für die Fachmodule unerheblich ist, welche Module abgesehen von den direkt

⁴⁵ Ein *Package* ist ein Strukturierungsinstrument für Klassen in Java. Durch die Benutzung von Packages werden die Klassen hierarchisch angeordnet. In Analogie zu den Begriffen Ordner und Datei aus einem Dateisystem entspricht eine Klasse einer Datei und ein Package einem Ordner.

vorausgesetzten noch in der Modulhierarchie vorhanden sind, sind alle Module, die ein Szenario als Voraussetzung definiert, auch notwendig zur Darstellung des Szenarios.

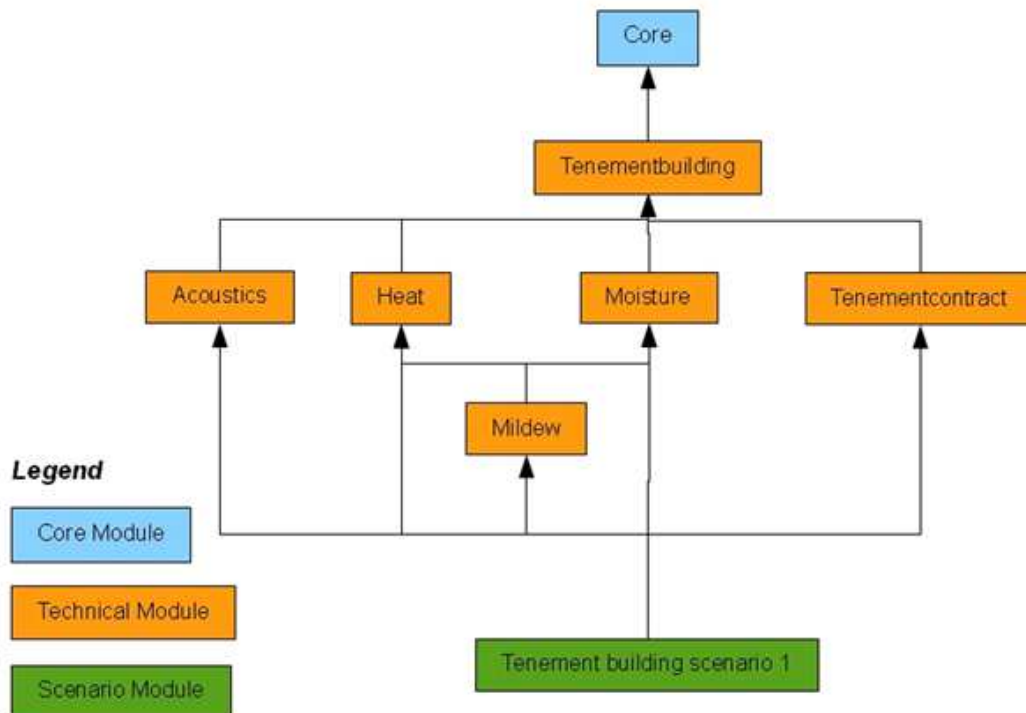


Abbildung 6: Modulbaum nach Modulklassen geordnet

Implementierte Module

Insgesamt wurden im Rahmen dieser Arbeit für den Prototyp 34 Module implementiert. Jedes dieser Module kann jeweils eindeutig einer der obengenannten Modulkategorien (*Core Module*, *Technical Module*, *Scenario Module* sowie zur Kategorie *Test Module*⁴⁶) zugeordnet werden. Eine detaillierte Übersicht der vorhandenen Module findet sich in Anhang G im Kapitel „Liste der Module“.

4.1.2 Erweiterbarkeit

Erweiterungen sind eine bewährte Möglichkeit, Module zu entkoppeln. Es ist eines der Prinzipien, die den Erfolg der Eclipse-Plattform ausmachen (Gamma & Beck, 2003). Ein Modul bietet dabei anderen Modulen die Möglichkeit an, es zu ergänzen. Die Fachbegriffe der Eclipse Plattform sind „extension point“ für die registrierende (ergänzbare) Stelle und „extension“ für die Erweiterung (McAffer & Lemieux, 2005).

Ein wesentliches Kennzeichen der im Rahmen dieser Arbeit entwickelten Kernmodule sind die bereitgestellten Möglichkeiten zur Erweiterung. Es lassen sich alle Elemente des Objektmodells (s. Kap. 4.3) mit Hilfe eines Extension Points ergänzen. Im Folgenden findet sich eine Auswahl der bereitgestellten Extension Points⁴⁷:

⁴⁶ Es wurden einige Test Module implementiert, um das Konzept des automatisierten Tests für dieses System zu validieren.

⁴⁷ Eine vollständige Auflistung der Extension Points wird in Anhang G gegeben.

- **Simulationselement:** Ermöglicht die Deklaration neuer Simulationselemente⁴⁸.
- **Simulationselementbeziehung:** Definiert eine hierarchische Beziehung zwischen zwei Simulationselementen und ermöglicht damit den Aufbau von Simulationselementhierarchiebäumen. Die eigene Definition dieses Extension Points erlaubt es, - im Gegensatz zu einer Definition der Eltern-Kind-Beziehung im Extension Point „Simulationselement“ selbst - Simulationselemente an verschiedenen Stellen in einer Hierarchie einzuhängen und damit eine Wiederverwendung zu ermöglichen.
- **Parameter:** Definiert einen Parameter als eine Kombination aus Wert, Dimension und Einheit.
- **Simulationselement-Parameter-Beziehung:** Ordnet einen Parameter einem Simulationselement zu. Auch hier gilt - ähnlich wie beim Extension Point „Simulationselementbeziehung“ -, dass die separate Definition dieses Extension Points es ermöglicht, einen Parameter mehrmals zu verwenden.
- **Parameterberechnungsalgorithmus:** Legt den Algorithmus zur Berechnung eines Parameterwertes fest. Die Trennung zwischen Parameter und Parameterberechnungsalgorithmus erlaubt es, je nach Bedarf für einen Parameter unterschiedliche Algorithmen auszuprobieren bzw. denselben Algorithmus für verschiedene Parameter zu nutzen.
- **Gewichteter Parameterbeitrag:** Dies ist eine recht spezifische Erweiterungsmöglichkeit, die aber besonders geeignet ist, das Prinzip der Erweiterbarkeit genauer darzustellen: Beispiel ist hier der Parameter „Mieterzufriedenheit“: Dessen Werte liegen in dem Intervall $[-1,1]$, wobei -1 für äußerste Unzufriedenheit und 1 für totale Zufriedenheit steht. Alle durch das System gesteuerten Mieter (NPCs) besitzen diesen Parameter. Beim Unterschreiten einer vorher definierten Schwelle für einen gewissen längeren Zeitraum erfolgt der Auszug des Mieters aus der Wohnung. Fachmodule können einen modulspezifischen Zufriedenheitsparameter besitzen, so besitzt das Akustik-Modul einen Parameter, der die Zufriedenheit des Mieters mit dem Lärmpegel in der Wohnung angibt, das Wärme-Modul trägt die Zufriedenheit mit der Wohnungstemperatur bei und das Modul `TenementContract` gibt mit seiner Zufriedenheit an, wie der Mieter die Miethöhe einschätzt. Programmtechnisch registrieren sich die modulspezifischen Zufriedenheitsparameter beim allgemeinen Zufriedenheitsparameter und werden dann gewichtet gemittelt. Unterschreitet einer der spezifischen Parameter eine bestimmte Schwelle, so wird der allgemeine Zufriedenheitsparameter als Minimum der spezifischen Parameterwerte ermittelt („K.O.-Kriterium“). Diese Lösung hat den Vorteil, dass jedes Modul seinen Beitrag leisten kann, aber nicht muss. So findet keine Kopplung statt.

Ein weiterer Vorteil der beziehungserzeugenden Extension Points wie „Simulationselementbeziehung“, „Simulationselement-Parameter-Beziehung“ und „Gewichteter Parameterbeitrag“ ist die Unterstützung des modularen Systementwurfs: Die Deklaration der Beziehung kann in einem weiteren, dritten Modul stattfinden. Verdeutlicht werden kann dies am Beispiel der Simulationselement-Parameter-Beziehung: Gäbe es die eigene Definition der Beziehung nicht, so müsste die Zuordnung entweder beim Simulationselement selbst oder aber beim Parameter aufgehängt sein:

⁴⁸ Die Definition eines Simulationselementes bzw. weiterer Elemente des Simulationsmodells wird in Kap. 4.3 gegeben.

- **Zuordnung am Simulationselement aufgehängt:** Dies bedeutet, dass das Simulationselement alle Parameter kennen müsste. Somit müssten alle Module, in denen Parameter des Simulationselements definiert sind, schon zur Zeit der Definition des Simulationselements bekannt sein.
- **Zuordnung am Parameter aufgehängt:** Dieses bedeutet, dass zur Zeit der Parameterdefinition alle Simulationselemente bekannt sein müssen, denen der Parameter zuzuordnen ist.

Für beide Fälle bedeutet die Benutzung eines Beziehungs-Extension-Points mehr Freiheiten:

- **Zeitliche Freiheit:** Die beiden zu verbindenden Elemente können zeitlich unabhängig voneinander definiert werden.
- **Örtliche Freiheit:** Die Beziehungserweiterung kann in einem dritten Modul definiert werden.
- **Semantische Freiheit:** Es können unterschiedliche Objektmodelle aufgebaut werden. Die zu verbindenden Elemente müssen nicht in allen Szenarien miteinander verbunden sein.

Gewöhnlich verlangen diese Extension Points, dass bei einer Erweiterung eine Java-Klasse angegeben wird, die eine Extension Point spezifische Schnittstelle⁴⁹ implementiert.

a) Beispiel eines Extension Points

Zur Verdeutlichung sollen hier die wesentlichen Schritte bei der Verwendung eines Extension Points anschaulich dargestellt werden⁵⁰:

Der betrachtete Extension Point ist *Simulationselement* (`simuElement`). Für einen Extension Point wird zunächst eine Beschreibung definiert: Das sind die formalen Angaben, die notwendig sind, um eine Extension zu diesem Extension Point zu definieren. Die Beschreibung wird textuell vorgenommen, die Entwicklungsumgebung stellt aber einem grafischen Editor bereit, mit dem die notwendigen Angaben komfortabel erfasst werden können (Abbildung 7: Extension Point Editor).

⁴⁹ Eine solche Schnittstelle wird in der Programmiersprache Java in sogenannten *Interfaces* implementiert.

⁵⁰ Eine tiefergehende und vollständige Einführung in den Extension Point Mechanismus findet sich z.B. in Gamma & Beck (2003).

The screenshot shows the 'simulationElement' Extension Point Editor. The left pane, titled 'Extension Point Elements', contains a tree structure with 'extension' as the root, which has sub-elements 'point', 'id', 'name', and 'simulationElement'. The 'simulationElement' element has sub-elements 'name', 'description', and 'class'. The right pane, titled 'Attribute Details', shows properties for the 'name' attribute: Name: name, Deprecated: false (selected), Use: optional, Type: string, Translatable: false (selected), and Restrictions: empty. A description field contains 'Name of the simulation element.' The bottom of the editor has tabs for 'Overview', 'Definition', and 'Source'.

Abbildung 7: Extension Point Editor

Die Angaben für eine Erweiterung des Typs `simuElement` sind in Abbildung 7 grafisch dargestellt und werden in Tabelle 4 beschrieben:

Attribut	Typ	Beschreibung
name	String	Name des Extension Points
description	String	Beschreibung des Extension Points
class	ISimuElement	Die Klasse, in der die Erweiterung untergebracht ist. Diese Klasse muss das Interface ISimuElement implementieren.

Tabelle 4: Konfigurationsinformationen der Extension "simuElement"

Nachdem ein Extension Point definiert wurde, kann er verwendet werden, d.h. es kann eine Extension deklariert werden. Für die Erweiterung `Radiator` ist ein Eintrag in der Datei `plugin.xml` nötig. Diese Datei ist eine Eclipse-spezifische Erweiterung eines OSGi-Bundles, das dieses zum „Plug-in“ werden lässt. Sie nimmt u.a. die Informationen über Extension Points und Extensions auf.

Für das Simulationselement „Heizkörper“ werden die folgenden Informationen bereitgestellt:

```

01: <extension
02:     point="edu.uni_weimar.simuframe.simuElement">
03:     <simulationElement
04:         class="edu.uni_weimar.simuframe.tenementbuilding.heat.element.Radiator"
05:         description="A radiator is used to heat a tenement."
06:         name="Radiator">
07:     </simulationElement>
08: </extension>

```

Quellcode 1: plugin.xml: Extension Simulationselement Radiator

In Zeile 6 wird der Name als „Radiator“ definiert, Zeile 5 enthält die Beschreibung, und in Zeile 4 ist der Name der Klasse zu finden, die das Interface `ISimuElement` implementiert und damit das Simulationselement-spezifische Verhalten definiert.

4.2 Ergebnisorientierte Implementierung des Prototyps

Der erstellte Prototyp ist keine ausgereifte Software. Er dient zur Überprüfung der technischen Machbarkeit ausgewählter Konzepte, die in dieser Arbeit vorgestellt werden. Im Folgenden werden einige aktuelle Beschränkungen aufgezeigt und erörtert.

4.2.1 Anforderungen an den Szenario-Editor

Die Idee des Szenario-Editors verlangt, dass erstellte Objekte dynamisch in die laufende Anwendung eingehängt werden können („hot deployment“). Das ist eine Anforderung, die der realisierte Prototyp nicht erfüllt. In diesem Kapitel soll diskutiert werden, weshalb das so ist und welche Maßnahmen notwendig wären, um ein Hot Deployment zu ermöglichen.

Die Deklaration der Objekte über den Plug-in-Mechanismus von Eclipse erfordert, dass diese statisch und persistent in der Datei `plugin.xml` des betreffenden Plug-ins vorliegt. Dies bedeutet auf der anderen Seite, dass bei Änderung eines Plug-ins zur Zeit der Server neu gestartet werden muss.

Es gibt mehrere Möglichkeiten, diese Beschränkung zu umgehen: Das komplette Plug-in könnte zur Laufzeit neu generiert werden. Dies schließt zu generierende Textdateien (u.a. `plugin.xml` und `MANIFEST.MF`) genauso ein wie Java-Quellcode-Dateien. Die Java-Quellcode-Dateien müssten dann zur Serverlaufzeit noch kompiliert werden. Zur sauberen Trennung der verschiedenen Versionen ist für jeden Generierungslauf die Vergabe einer Versionsnummer notwendig, ähnlich wie das schon bei der automatisierte Erstellung von Eclipse-Plug-ins durchgeführt wird (Eclipse.org, 2011). Nach der Erzeugung des Plug-ins könnte es über die OSGi-Mechanismen geladen und gestartet werden. Um die Kompilierung von Java-Klassen zur Server-Laufzeit zu umgehen, wäre es alternativ auch möglich, die Script Engine einer zusätzlichen Skriptsprache (z.B. Ruby (JRuby.org, 2012) oder Python (Python, 2012)) zu benutzen (O’Conner, 2006). Hier entfielen der Kompilierungsvorgang.

Eine Alternative ist die direkte, programmatische Manipulation der Extension-Registry⁵¹: Hier würden die Erweiterungen zur Laufzeit der Extension-Registry bekanntgegeben, ohne vorherige Generierung der deklarierenden Textdateien (`plugin.xml`, `MANIFEST.MF`).

Als Fazit wird festgehalten: Der erstellte Prototyp unterstützt zwar noch nicht das vom Szenario-Editor geforderte Hot Deployment. Technisch gesehen ist dieses jedoch möglich. Es ist eine Frage des notwendigen Realisierungs-Aufwandes, den Prototyp um diese Eigenschaft zu erweitern.

4.2.2 Spezifikationsorientierte Implementierung

Für jedes Szenario, das auf der Plattform zur Verfügung gestellt wird, muss ein Simulationsobjektmodell aufgebaut werden. In der derzeitigen Implementierung werden die Objektmodelle mit Hilfe von Java-Code aufgebaut. Es wird das Factory Pattern genutzt (Gamma, Helm, Johnson & Vlissides, 1995).

Der Einsatz von Java-Code ist auch hier nur eine von mehreren Möglichkeiten. Zusätzlich scheint sie für die im vorigen Abschnitt schon diskutierte Szenario-Editor-Anforderung des Hot Deployments nicht ideal: Zur Laufzeit müsste Java-Code generiert und kompiliert werden. Zu den

⁵¹ Dies ist das Objekt des Eclipse-Frameworks, das zur Laufzeit die Erweiterungen verwaltet.

weiteren Möglichkeiten, ein Objektmodell zu definieren und zu generieren, können beispielsweise das Auslesen einer Datei oder die Benutzung von Extension Points gezählt werden.

Die verschiedenen Möglichkeiten zur Erzeugung eines Objektmodells können nach unterschiedlichen Kriterien kategorisiert werden:

Persistenz Das Model eines Szenarios muss gespeichert werden, damit es nicht immer wieder neu definiert werden muss. Möglichkeiten der Speicherung eines Modells sind u.a.:

- **Datei:** Es sind verschiedene Dateiformate denkbar (z.B. XML, Textdateien, Binärdateien), die vom benutzten Framework abhängig sind.
- **Datenbank**
- **Code-Datei:** Eine Java-Klasse, die auf programmatischem Wege ein Szenario erstellt, ist ebenfalls eine Form der Speicherung.

Modelldefinition Es gibt verschiedene Technologien zur Definition des Modells, d.h. der Festlegung, welche Objekte in einem Modell enthalten sind und wie diese verknüpft sind. Wichtige Anforderung im Kontext der Plattform ist die Erweiterbarkeit des Modells, d.h. die Technologie muss die modulare Entwicklung des Modells unterstützen und darf nicht darauf angewiesen sein, dass das komplette Modell von Beginn an vorhanden ist. Die folgenden Technologien sind Kandidaten zur Definition des Modells:

- **Modellierungs-Framework:** Das Modell kann mit Hilfe von Modellierungsframeworks definiert werden. Diese stellen gewöhnlich einen grafischen Modell-Editor zur Verfügung. Ein mächtiges, im Eclipse-Ökosystem vorhandenes Framework ist das *Eclipse Modeling Framework (EMF)*. Es kann das Modell mit Hilfe verschiedener Persistenzanbieter (engl. *persistence provider*) verwalten und die entsprechenden Java-Klassen generieren.
- **Schema-Dateien:** Als Beispiel für die Verwendung von Dateien mit einem standardisierten Format sei *Java Architecture for XML Binding (JAXB)* genannt. Dies ist ein Framework, bei dem das Datenmodell in sogenannten XML-Schema-Dateien beschrieben ist. Mit Hilfe der Schema-Daten werden entsprechende Java-Klassen generiert.
- **Code-Datei:** Das Modell kann durch manuelles Erstellen von Java-Klassen definiert werden. Dieser Weg ist im Prototyp realisiert.
- **Proprietäres Dateiformat:** Die Modell-Informationen werden in einer Datei in einem selbst definierten Format abgelegt. Das Modell wird aus dieser Datei mit einer Eigenentwicklung gelesen. Beispiel für dieses Vorgehen ist TRNSYS (s. Anhang D).

Generierung Diese Kategorie unterscheidet die Vorgehensweisen, mit denen aus den Modelldaten das Modell erzeugt wird.

- **Persistenz-Framework :** Persistenz-Frameworks ermöglichen den Datentransfer von nichtflüchtigem Speicher (Beispiel: Dateien auf der Festplatte) zu flüchtigem Speicher (Beispiel: Hauptspeicher eines Rechners) und zurück. Bei Benutzung einer Datenbank werden gewöhnlich objektrelationale Abbildungswerkzeuge eingesetzt. Ihre Aufgabe ist es, die (Java-)Objekte des Hauptspeichers auf die Tabellen einer relationalen Datenbank abzubilden. Geläufige Vertreter hier sind *EclipseLink* (EclipseLink, 2012) und *Hibernate* (Hibernate, 2012).

Andere Persistenz-Frameworks setzen Dateien zur Speicherung ein, das obengenannte JAXB nutzt das XML-Format.

- **Proprietäres Framework:** Ein selbstentwickeltes, proprietäres Framework liest die Modelldefinition ein (beispielsweise aus einer Datei mit einem proprietären Format) und erzeugt daraus das Simulationsmodell.
- **Code-Datei:** Das Modell kann mit Hilfe von Java-Code erzeugt werden. Diese Lösung erfordert den geringsten Aufwand, hat dafür jedoch nur eine eingeschränkte Flexibilität. Sie ist die gewählte Lösung im Prototyp.

Mit Hilfe der verschiedenen genannten Möglichkeiten kann eine Spezifikation implementiert werden. Die Implementierungen unterscheiden sich bezüglich verschiedener Merkmale

- **Entwicklungsaufwand:** Die Entwicklung eines eigenen Frameworks ist sehr aufwändig, hingegen ist die Implementierung des Modells mit Code eine sehr effiziente Möglichkeit.
- **Einarbeitungsaufwand:** Frameworks erfordern einen Einarbeitungsaufwand. Bei einem selbst entwickelten Framework fällt dieser nur für neu hinzukommende Entwickler an, diese können von der direkten Zusammenarbeit mit den Erstellern des Frameworks effizienter lernen. Der geringste Einarbeitungsaufwand fällt bei der direkten Implementierung in Code an.
- **Flexibilität durch Konfigurierbarkeit:** Frameworks bieten mitunter eine hohe Flexibilität durch Konfiguration: Oft lässt sich allein durch Bereitstellung von Daten in Dateien oder Datenbanken das Verhalten ändern. Es ist kein kompletter Entwicklungszyklus mit Kompilation, Test⁵² und Auslieferung notwendig - eine wesentliche Aufwandserleichterung. Der Aufwand, ein eigenes Framework zu erstellen ist sehr hoch.
- **Flexibilität der Funktionalität:** Es ist schwierig, Funktionalität in Frameworks zu ergänzen oder zu ändern. Dies ist bei der Eigenentwicklung wesentlich einfacher.

Trotz der Unterschiede der verschiedenen Implementierungsmöglichkeiten bleibt festzuhalten, dass alle Implementierungen derselben Spezifikation genügen und somit dieselbe Funktionalität zur Verfügung stellen. Da die verfügbaren Ressourcen begrenzt waren, wurde für den Prototyp die Implementierung mit dem geringsten Entwicklungsaufwand gewählt, d.h. es wurde der Coding-Ansatz bei der Erstellung der Szenario-Modelle angewendet. Durch den Einsatz von Design Patterns wie *Fassade* und *Factory* (Gamma et al., 1995) ist es aber möglich, bei einer Ausweitung des Prototyps für spezielle Aufgaben Frameworks einzusetzen, ohne einen kompletten Neuentwurf des Prototyps durchzuführen.

4.3 Simulation: Objektmodell

Das grundlegende Objektmodell eines Simulationsszenarios ist einfach aufgebaut. Es wurde in Anlehnung an einen Vorschlag von Merrill (2000) erstellt. Dort werden 4 Typen von „Knowledge Objects“ beschrieben:

“Entities are things (objects). Actions are procedures that can be performed by a learner on, to, or with entities or their parts. Processes are events that occur often as a result of some action. Properties are qualitative or quantitative descriptors for entities, actions, or processes.”

⁵² Ein Test ist sicherlich auch bei der Änderung einer Konfigurationseinstellung notwendig, jedoch entfallen die anderen Teile des Entwicklungszyklus, inklusive eines aufwändigen Systemtests, der bei Neuerstellung der binären Programmdateien durchgeführt werden sollte.

Die weitere Ergänzung des Modells erfolgte im Rahmen einer agilen, d.h. iterativen, inkrementellen und anforderungsgetriebenen Vorgehensweise in einem heuristischen Verfahren (Martin, 2002; Schwaber & Beedle, 2002).

4.3.1 Entity: SimuElement

Ein Simulationselement⁵³ entspricht einem Gegenstand oder einem Teil eines Gegenstandes der realen Welt. Simulationselemente können hierarchisch miteinander verknüpft werden: Ein Simulationselement kann einem Simulationselement zugeordnet sein und seinerseits ein oder mehrere Kind-Simulationselemente besitzen. Zusätzlich kann es beschrieben werden durch ein oder mehrere Parameter. Beispiele von Simulationselementen werden in Tabelle 5 gezeigt. Ein hierarchisches Modell dieser Simulationselemente ist in Abbildung 8 dargestellt: Basiselement der Hierarchie ist ein Gebäude, das aus Wohnungen besteht. Die Wohnungen werden von Wänden umfasst, eine Wand kann ein Fenster enthalten. Das Gebäude kann eine Heizungsanlage enthalten, dann enthalten die Wohnungen auch einen Heizkörper.

Simulationselement	Beschreibung
Building	Gebäude als Ganzes
Environment	Umgebung, diese besitzt Parameter, die zeitabhängig ihren Wert ändern.
HeatingSystem	Heizungsanlage in einem Gebäude
Radiator	Heizung in einer Wohnung
Tenant	Mieter einer Wohnung
Tenement ⁵⁴	Wohnung in einem Gebäude
WallElement	Wand eines Gebäudes bzw. einer Wohnung
Window	Fenster in einer Wand.

Tabelle 5: Simulationselemente im Prototyp

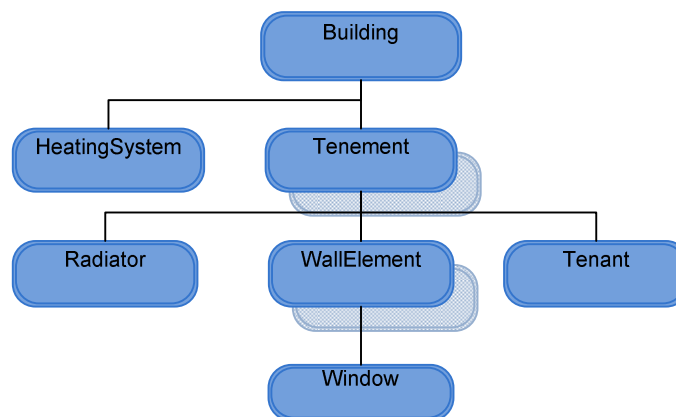


Abbildung 8: Hierarchisches Modell von Simulationselementen

4.3.2 Property: Parameter

Ein Parameter hat einen Typ, einen Wert und eine Einheit. Er ist einem Simulationselement zugeordnet. Er kann einen Berechnungsalgorithmus besitzen, dem gleichen Parameter können in verschiedenen Kontexten auch verschiedene Berechnungsalgorithmen zugeordnet sein. Dieser

⁵³ Die im Quellcode verwendete Bezeichnung ist „SimuElement“.

⁵⁴ In der derzeitigen Modellierung ist Raum und Wohnung gleichgesetzt, d.h. alle Wohnungen bestehen aus nur einem Raum.

Kontext kann das Szenario sein, in dem der Parameter vorkommt, es kann aber auch das Simulationselement sein, dem der Parameter zugeordnet ist.

Parameter, denen kein Algorithmus zugewiesen wurde, sind entweder konstant oder können vom Spieler oder System gesetzt werden. Ein solcher Parameter ist beispielsweise die Miete (`Rent`). Berechnet wird hingegen die Temperatur innerhalb einer Wohnung. Tabelle 6 zeigt eine Auswahl der im Prototyp implementierten Parameter. Eine weitere Diskussion der Eigenschaften von Parametern findet in Anhang G statt.

Parameter	SimuElement	Beschreibung
TenantSatisfaction	Tenant	Mieterzufriedenheit, eine Maßzahl, wie zufrieden der Mieter mit seiner gegenwärtigen Wohnsituation ist.
NoiseLevel	Tenement, Environment	Tatsächliche Lautstärke, in einer Wohnung oder in der Umgebung
SoundReductionIndex	Window, WallElement	Schalldämmmaß
CurrentThermalPower	HeatingSystem	Die von der Heizungsanlage abgegebene aktuelle Wärmeleistung.
HeatLevel	Tenement, Environment	Temperatur

Tabelle 6: Parameter aus dem Prototyp

4.3.3 Action

Eine Action - auch als Benutzer-Aktion bezeichnet - bietet dem Spieler die Möglichkeit, auf das Spiel Einfluss zu nehmen: Ein Spieler kann eine Action anstoßen und gegebenenfalls mit entsprechenden Parametern versorgen. Durch eine angebundene Prozedur wird gewöhnlich eine Statusänderung des Szenarios vorgenommen. Eine Action ist an ein Simulationselement gebunden, die Verfügbarkeit dieser Action kann vom Zustand des Szenarios abhängen - die Action kann auch zeitweise inaktiv sein. Tabelle 7 zeigt im Prototyp implementierte Actions als Beispiele. Die Action `WindowInstaller` muss beispielsweise mit dem Parameter des neuen, einzubauenden Fenstertyps versehen sein. Die Action `TenantHirer` ist ein Beispiel für eine nicht immer verfügbare Benutzer-Aktion: Wenn der Spieler keine weiteren Wohnungen zur Vermietung zur Verfügung hat, ist sie inaktiv.

Action	SimuElement	Beschreibung
OrderFuel	HeatingSystem	Kauft - sofern eine Ölheizung vorhanden ist - neues Heizöl.
TenantHirer	Tenant	Bietet einem Mieter einen Mietvertrag für eine freie Wohnung an.
WallElementExchanger	Tenement, Building	Ermöglicht es, in einem Gebäude oder einer Wohnung neue Wände mit anderen Eigenschaften einzusetzen ⁵⁵ .
WindowInstaller	Tenement, Building	Ermöglicht es, neue Fenster für ein Gebäude oder eine Wohnung auszuwählen und diese gleich auszutauschen.

Tabelle 7: Im Prototyp implementierte Actions

⁵⁵ An dieser Stelle werden die Vorteile eines Simulationsspiels gegenüber der Wirklichkeit deutlich: Es können Aktionen durchgeführt werden, die nicht realistisch sind - denn das Austauschen der (tragenden) Wände ist in Realität eher eine ungewöhnliche Maßnahme - die Häuser werden gewöhnlich abgerissen. Solche Möglichkeiten bieten dem Spieler eine Chance auf Lernerfahrungen.

Das Beispiel des `windowInstaller` (und auch des `wallElementExchanger`) zeigt, dass eine Action auch unterschiedlichen Simulationselementen zugeordnet werden kann. In diesem Fall werden die Fenster entweder für das gesamte Gebäude oder nur für eine spezielle Wohnung ausgetauscht.

Abbildung 9 zeigt ein Beispiel für die Darstellung der Aktionen in der Benutzeroberfläche des Simulationskerns. Das Kontextmenü der Wohnung enthält Einträge, die den hier sinnvollen Benutzeraktionen entsprechen, wie dem Einbau neuer Fenster sowie der Kündigung des Mietvertrags.

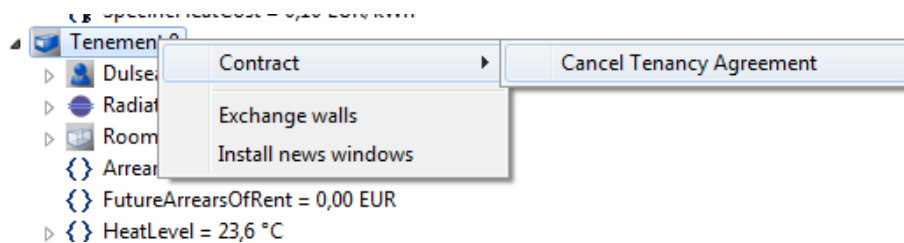


Abbildung 9: Repräsentation der Actions im Kontextmenü

4.3.4 Event

Ein *Event* definiert ein Systemereignis, das entweder mit einer bestimmten Wahrscheinlichkeit oder aufgrund einer Änderung des Szenariostatus eintritt. Es ändert seinerseits den Status des Szenarios und könnte dadurch eine Reaktion des Spielers erfordern. Die Einbeziehung von Wahrscheinlichkeit ist - neben den Spieleraktionen - eine Ursache für einen nicht deterministischen Simulations- und damit Spielverlauf. Das ist eines der Merkmale eines authentischen Lernkontextes (Bröker, 2013). Zu einem Event gehört gleichfalls eine Prozedur, die beim Eintreten des Events ausgeführt wird. Der Unterschied zur *Action* besteht in dem Auslöser: Bei der *Action* ist es der Spieler, ein *Event* wird vom System angestoßen.

Event	Referenz	Beschreibung
TenantChangesJob	Tenant	Der Mieter wechselt seinen Arbeitsplatz so, dass dadurch ein Umzug bedingt ist.
FamilyGrows	Tenant	Die Familie des Mieters wächst durch die Geburt eines neuen Kindes, es gibt ein zusätzliches Haustier oder die Wohnung wird aus anderen Gründen zu klein. Es folgt ein Umzug.
TenantIsUnsatisfied	Tenant-Satisfaction	Die Zufriedenheit des Mieters mit seiner Wohnung ist seit einer bestimmten Zeit unter dem Schwellenwert, die Konsequenz ist der Auszug ⁵⁶ .
TenantDoesNotPayRent	Tenement	Der Mieter kann die fällige Miete nicht zahlen.
RentPayment	Building	Der Mieter zahlt die fällige Miete.

Tabelle 8: Im Prototyp deklarierte Events

Tabelle 8 zeigt die im Prototyp deklarierten Events. Die Spalte *Referenz* verdeutlicht, dass ein Event in das Simulationsmodell integriert ist: In dieser Spalte wird definiert, mit welchem Objekt des Simulationsmodells das Event verbunden ist. Es gibt drei unterschiedliche Kategorien von Events:

⁵⁶ Bei diesem Event handelt es sich um ein parameterbasiertes Event (`ParameterValueSystemEvent`), d.h. es tritt nicht mit einer gewissen Wahrscheinlichkeit ein, sondern abhängig von einem bestimmten Wert eines Parameters.

- **Regular TimedSystemEvent:** Dieses Event tritt in regelmäßigen Abständen auf. Es ist an eine bestimmte Simulationselementklasse geknüpft. Die zugehörige Prozedur wird ohne Einschränkung ausgeführt. Beispiel ist das Event `RentPayment`, mit dessen Hilfe die regelmäßige Zahlung der Miete abgebildet wird.
- **Probabilistic TimedSystemEvent:** Grundsätzlich tritt dieses Event in regelmäßigen Abständen auf und ist ebenfalls an ein bestimmtes Simulationselement gebunden. Bei jedem Eintritt des Events wird jedoch mit einer konfigurierten Wahrscheinlichkeit entschieden, ob die zugehörige Prozedur ausgeführt wird oder nicht. Beispiele für diesen Event-Typ sind die Events `TenantChangesJob`, `FamilyGrows` und `TenantDoesNotPayRent`. Beim Eintreten eines dieser Events wird die Grundgesamtheit der zugeordneten Objekte (also hier `Tenant` und `Tenement`) ermittelt. Für jedes Element der Grundgesamtheit wird einzeln per Zufall bestimmt, ob die Prozedur ausgeführt wird.
- **ParameterValueSystemEvent:** Dieses Event ist an einen Parameter gebunden. Die zugehörige Prozedur wird dann ausgeführt, wenn der zugehörige Parameter bestimmte Werte bzw. Wertebereiche annimmt. Beispiel ist das Event `TenantUnsatisfied`. Dieses ist an den Parameter `TenantSatisfaction` gebunden. Es wird ausgelöst, wenn der Wert dieses Parameters für eine definierte Zeit unter einem Schwellenwert liegt. Die zugehörige Prozedur führt den Auszug des Mieters durch.

4.3.5 Commands

Merrill (2000) benennt Actions und Events als Bestandteile des Objektmodells. Gleichzeitig spricht er von Prozeduren. Im Rahmen des hier vorgestellten Frameworks werden diese Begriffe leicht abweichend eingeordnet: Eine Action hat die Bedeutung einer Benutzeraktion, d.h. der Benutzer interagiert mit dem System und löst dabei eine Prozedur aus. Ein Event tritt in der Form eines Systemevents auf, d.h. von der Plattform ausgelöst tritt dieses Ereignis auf und löst ebenfalls die Ausführung einer Prozedur aus. Die sowohl von Actions als auch von Events ausgelösten Prozeduren werden innerhalb der Spielplattform *Command* genannt.

4.4 Konzept weiterer Frameworkelemente

Im vorigen Kapitel wurden die Modellelemente beschrieben, aus denen ein Szenario aufgebaut ist. Ein Szenario ist der zentrale Bestandteil der Spielplattform. Damit diese korrekt arbeiten kann, sind jedoch noch weitere Plattformbestandteile notwendig. Grundsätzlich wurden diese bereits in Abbildung 5 (S. 31) beschrieben. Im Folgenden werden diejenigen Frameworkelemente näher beschrieben, die im aktuellen Prototyp enthalten sind.

4.4.1 User Manager (UM)

Der User Manager verwaltet die spieler-spezifischen Daten. Er ist die zentrale Instanz, die diese Daten bereitstellt, modifiziert und persistiert (d.h. die Daten werden abgespeichert, so dass sie eine Spielsitzung überdauern). Diese Daten bestehen aus den folgenden Teilelementen:

- **Auszeichnungen („Badges“):** Eine Auszeichnung dokumentiert als eine Art virtueller Orden einen bestimmten Leistungsstand oder eine bestimmte Tat.
- **Geldmenge:** Der Spieler erwirbt durch Aktivitäten im Spiel (Spiel-)Geld („Coins“), das er benutzen kann, um Spielgegenstände zu kaufen.
- **Punktestände:** Ebenfalls werden die Aktivitäten des Spielers mit Punkten belohnt. Eine häufig anzutreffende Form dieser Punkte sind die Erfahrungspunkte („XPs“). Im Gegensatz zu

Geld sind XPs eine akkumulierende Größe, d.h. einmal erworbene Punkte können nicht wieder verloren werden.

- **Fähigkeiten** („Skills“): Skills sind eine besondere Form von Punkten. Ein Skill misst eine virtuelle Fähigkeit des Spielers in Form einer akkumulierenden Größe.
- **Persönliche Spielgegenstände** („Inventory“): Im Inventory sind Spielgegenstände zu finden, die während des Spiels erworben werden. Diese Gegenstände haben gewöhnlich eine Funktion innerhalb des Spiels. Das Inventory umfasst den Besitz von virtuellen Gegenständen des Spielers.

4.4.2 Skill- & Assessment Component (SAC)

Aufgabe der *Skill- & Assessment Component* ist die Steuerung des Lernfortschritts der Spieler. Sie sollen mit Szenarien konfrontiert werden, die für sie herausfordernd sind, sie aber nicht überfordern. Grundsätzlich werden Fähigkeiten („Skills“) definiert. Diese spiegeln die fachlichen Lernziele wider. Eine solche Fähigkeit kennt für jeden Spieler einen aktuellen (Punkte-)Stand, d.h. jede Fähigkeit hat einen ganzzahligen Wert. Für ein Szenario wird definiert, welche fachlichen Eingangsvoraussetzungen es hat, d.h. es werden Fähigkeiten zusammen mit Mindest-Punktezahlen aufgelistet. Im Status eines Spielers wird eine Liste der Fähigkeiten zusammen mit den aktuellen Punktzahlen geführt. Diese Fähigkeits-Punktzahlen konnte er durch das erfolgreiche Lösen von Missionen in den Szenarios erwerben. Einem Spieler werden solche Szenarios zur Bearbeitung angeboten, deren Mindestvoraussetzungen er erfüllen kann und vornehmlich solche, deren Mindestvoraussetzungen in der Nähe seiner aktuellen Fähigkeits-Punktzahlen liegen.

Zu einem Szenario gehören in der Regel auch Missionen („Missions“). Sie definieren Aufgaben für den Spieler. Die Belohnung („Reward“) für das erfolgreiche Lösen einer Mission kann bestehen aus Elementen der folgenden Liste:

- **Fähigkeits-Punkte:** Für das Lösen von Missionen können unterschiedliche Fähigkeiten nötig sein. Die notwendigen, gezeigten Fähigkeiten können mit Fähigkeits-Punkten honoriert werden, mit deren Hilfe virtuelle Fähigkeiten („Skills“) aufgebaut werden können. Diese sollen dann ein Abbild der tatsächlichen Fähigkeiten ergeben⁵⁷.
- **Coins:** Coins können notwendig sein, um bestimmte Spielgegenstände kaufen zu können, mit deren Hilfe sich Missionen lösen lassen. Daher ist es sinnvoll, Gelegenheiten zu schaffen, in denen der Spieler Coins verdienen kann.
- **Erfahrungspunkte:** XPs sind gewöhnlich ein genereller Maßstab für die vom Spieler durchgeführten Aktionen und werden daher bei vielen Gelegenheiten als Belohnung ausgeschüttet⁵⁸.
- **Spielgegenstände:** Die Ausgabe von Spielgegenständen als Belohnung für eine erfolgreich gemeisterte Mission ist aus mehreren Gründen erfolgversprechend: Zum einen dient es als Belohnung zur Motivation des Spielers. Auch ist es eine Möglichkeit, neue Gegenstände und damit möglicherweise neues Wissen dem Spieler zu vermitteln. Bei geschickter Planung der

⁵⁷ In verschiedenen Spielen gibt es auch eine zeitgesteuerte Mechanik zum Aufbau der Fähigkeiten: Der Spieler startet eine Ausbildung (eine Erhöhung eines Fähigkeitspunktestands) gegen die Zahlung von Coins oder anderen Spielgegenständen, nach dem Ablauf einer bestimmten Zeit liegt dann der verbesserte Skill vor. Dieses Vorgehen ist beispielsweise in EVE Online (CCP, 2012) oder Fliplife (Fliplife, 2012) zu finden.

⁵⁸ In einigen Spielen lassen sich Coins in XPs konvertieren: FarmVille (Zynga, 2009) ermöglicht dies beispielsweise durch den Kauf von Spielgegenständen mit Coins: für eine Kaufsumme von 100 Coins wird in der Regel 1 XP ausgeschüttet.

Missionen können diese bezüglich der verteilten Gegenstände aufeinander aufbauen, d.h. die Belohnung einer Mission in Form eines Gegenstands wird für die Lösung der nächsten Mission benötigt.

- **Auszeichnungen:** Als Belohnung für bestimmte Spielerleistungen können Auszeichnungen („Badges“) verliehen werden. Sie sind virtuelle Orden und dokumentieren einen bestimmten Leistungsstand, sie tragen zum Status des Spielers bei. Die erfolgreiche Bewältigung einer spielerischen Herausforderung in Form einer Mission kann eine auszeichnungswerte Leistung sein. Das Erreichen von bestimmten Orden kann die Eingangsvoraussetzung zu einer Prüfung in der formalen Ausbildung sein.

Ein Zertifikat („Certificate“) ist eine ähnliche Auszeichnung wie ein Badge. Es dokumentiert einen bestimmten Ausbildungsstand, d.h. Grundlage für die Erteilung eines Zertifikats an den Spieler ist die Erreichung festgelegter Punktestände für eine definierte Liste von Fähigkeiten. Ebenfalls notwendig sein kann der Besitz bestimmter anderer Badges oder Zertifikate. Zur gezielten Erreichung von Zertifikaten dient ein sogenannter *Certificate Guide*: Der Spieler wählt ein Zertifikat aus, das er erreichen möchte. Der Certificate Guide stellt dann die Aktionen vor, die der Spieler noch ausführen muss, um die Voraussetzungen für dieses Zertifikat zu erfüllen.

Ausgehend von Niegemann (2009b), der auf sogenannte Lernvoraussetzungen und auf Hierarchien derselben hinweist, können Fähigkeiten kombiniert werden zu einer Art „Fähigkeitsbaum“: Bestimmte Fähigkeiten können Unterfähigkeiten von anderen Fähigkeiten sein. Auch kann der Erwerb von Punkten einer Fähigkeit nur dann möglich sein, wenn schon ein entsprechender Punktestand für vorausgesetzte Fähigkeiten erreicht ist.

All die in diesem Kapitel genannten Frameworkelemente unterliegen dem Prinzip der Erweiterbarkeit. Missions, Certificates, Badges und Spielgegenstände können deklarativ der Spielplattform nach deren Erstellung hinzugefügt werden - unter Nutzung des Szenario-Editors.

4.4.3 Szenario Manager (SM)

Aufgabe des Szenario-Managers ist die Verwaltung der Szenarios, d.h. neue Szenarios werden dem Szenario-Manager bekanntgegeben. Dieser ist verantwortlich für die Persistierung der Szenarios. Er beantwortet Anfragen der SAC nach Szenarios, die bestimmte Fähigkeiten trainieren. Daher kennt er die Skill-Voraussetzungen eines Szenarios und die Skill-Punkte, die ein Spieler durch die Bewältigung der Szenario-eigenen Missionen verdienen kann. Der Szenario Manager erlaubt eine Schlagwortsuche („Tags“).

4.4.4 Monitoring Component (MC)

Die Monitoring Component dient zur Datensammlung im Rahmen der Beobachtung von Spielverhalten und Spieler. Ziel ist es Rückmeldungen zu erhalten, mit deren Hilfe sich Schwachstellen der Plattform und ihrer Inhalte beseitigen und Stärken ausbauen lassen. Fragen, die mit Hilfe der MC beantwortet werden sollen, sind:

- **Welche Szenarios werden besonders häufig gespielt?**
Besonders beliebte Szenarien können untersucht werden auf Gemeinsamkeiten. Mit diesem Wissen können Regeln für das Design von Szenarios und Missionen aufgestellt werden.
- **Welche Themen/fachlichen Bereiche sind beliebt?**
Es soll untersucht werden, wieso diese fachlichen Themen so beliebt sind: Sind sie besonders einfach zu lösen oder gut verstanden? Oder liegt das Thema in einem Bereich, der her-

ausfordernd, aber nicht überfordernd ist? Auch hier stellt sich die Frage, ob Regeln extrahiert werden können, die zum Design erfolgreicher Szenarien beitragen.

- **Welche Missionen werden gut angenommen? Welche Spielmechaniken kommen bei den Spielern besonders gut an?**

Es wird versucht, Regelmäßigkeiten und Muster zu finden, die in einen Richtlinienkatalog münden können. Damit lässt sich die Qualität der neu erstellten Szenarien erhöhen.

- **Welche Spieler sind erfolgreich beim Design von neuen Szenarien?**

Die Spieler, deren Szenarien besonders oft und erfolgreich gespielt werden, könnten gezielt belohnt und angesprochen werden, um die Erzeugung qualitativ hochwertiger Szenarien zu fördern.

- **Wie lange dauert eine Spielsitzung?**

Wird mehr in kurzen Zyklen zwischendurch gespielt? Oder gibt es lange Spielsitzungen, die komplexe Szenarien erfordern? In Abhängigkeit von der Dauer der Spielsitzungen kann der Ausbau der Plattform - also des Frameworks - gesteuert werden.

- **Wie ist der Spieler zu typisieren?**

Es gibt verschiedene Spielertypen, die sich sowohl im Lernverhalten als auch in dem was sie als Freude oder Spaß empfinden, unterscheiden. Die MC soll es ermöglichen, den Typ des Spielers zu bestimmen anhand seiner bevorzugten Aktionen. Diese Typisierung ist eine wichtige Eingangsgröße für die automatische Auswahl der Szenarien.

4.4.5 Simulation Engine (SE)

Die Simulation Engine ist der Teil des Frameworks, in dem die Szenarien ausgeführt werden. Sie bedient sich dabei verschiedener weiterer Frameworkelemente wie Time Manager und Event Manager.

a) Time Manager (TM)

Im Social Online Game „FarmVille“ (Zynga, 2009) lässt sich der Reifegrad von Früchten berechnen über den Saatzeitpunkt und die notwendige Reifezeit. Zu jedem Zeitpunkt zwischen Saat und Erntereife kann errechnet werden, welcher Anteil der Reifezeit gerade abgelaufen sind. Hier ist keine kontinuierliche Simulationsrechnung notwendig: Mit Hilfe des Parameters *Zeit* kann mit einer einfachen Rechnung das Ergebnis ermittelt werden.

Anders verhält es sich bei vielen physikalischen Problemen: Betrachtet man die Raumtemperatur während des Aufheizens eines Raumes, so handelt es sich um einen instationären Prozess, wenn nicht von einem idealisierten Gebäude ausgegangen wird: Mit steigender Raumtemperatur nehmen auch die Wärmeverluste an die Umgebung zu. Die Wärmeverluste sind damit eine zeitabhängige Größe in dem System. Für ein solches System lässt sich gewöhnlich eine Differentialgleichung aufstellen. Findet sich keine analytische Lösung für die Differentialgleichung, so bleibt nur die rechenintensivere, numerische Lösung. Sind noch weitere zeitlich abhängige Einflussgrößen vorhanden, beispielsweise die Umgebungstemperatur, dann wird das System komplexer bis zu einem Grad, an dem nur komplette Simulationsläufe noch helfen, das Systemverhalten zu illustrieren. Allein aus diesem Grund ist es nicht sinnvoll, Szenarien, in denen (bau)physikalische Vorgänge involviert sind, beispielsweise nach dem Vorbild der Social Online Games, in Echtzeit ablaufen zu lassen.

Es gibt also eine Unterscheidung von simulierter Zeit und Simulationszeit⁵⁹. Der Time Manager ist für die Verwaltung dieser verschiedenen Zeiten zuständig: Er kann Simulationszeitwerte in Werte der simulierten Zeit umrechnen und umgekehrt - er kennt einen Maßstab und eine Schrittweite für die Simulation des Parameters *Zeit*.

I Implementierung

Der Time Manager ist in der Klasse `TimeManager.java` implementiert. Ein wesentlicher Bestandteil sind Attribute, die in Tabelle 9 beschrieben werden.

Attribut	Typ	Beschreibung
ScaleDenominator	Date ⁶⁰	Dies ist der Zeitmaßstab. Der Wert gibt die Anzahl der Zeiteinheiten der simulierten Zeit an, die in einer Einheit der Simulationszeit ablaufen, d.h. simuliert werden. Ein ScaleDenominator von 3600 bedeutet, dass in einer Sekunde, die die Simulation in Echtzeit läuft, 3600 s (d.h. 1 Stunde) simuliert werden.
SimulationSteps	Long	Dieses Attribut zählt die Simulationsschritte mit, die im Laufe der Simulation durchgeführt werden. Der jeweilige Simulationsschritt wird in den Parametern als Attribut gespeichert, um kenntlich machen zu können, wann der Parameter das letzte Mal geändert wurde.
SimulationStepInterval	Long	Das SimulationStepInterval gibt die Zeitdauer in Millisekunden (ms) an, die abgelaufen sein muss, damit der nächste Simulationsschritt gestartet werden kann. Die hierdurch ausgelöste künstliche Pause in der Berechnung soll die Möglichkeit bieten, die benötigte Rechenleistung zu regulieren - für den Fall, dass große Szenarien oder mehrere Szenarien parallel berechnet werden.
GameStartTime	Date	Mit Hilfe dieses Attributs kann der simulierte Startzeitpunkt gesetzt werden. Das Attribut „GameTime“ wird in Abhängigkeit zu diesem Attribut berechnet.
RealStartTime	Date	Das Attribut RealStartTime wird genutzt, um den Zeitpunkt des Starts der Simulation festzuhalten.
GameTime	Date	Dieses Attribut gibt die aktuelle Zeit in der Simulation an.
RealTime	Date	Dies ist die aktuelle Zeit.

Tabelle 9: TimeManager: Attribute

ScaleDenominator und SimulationStepInterval haben eine Bedeutung für den minimalen Zeitschritt in simulierter Zeit: der minimale Zeitschritt entspricht dem Produkt der beiden Attribute: Beträgt der ScaleDenominator 3600 und wird das SimulationStepInterval auf 5000 ms festgelegt, so beträgt der minimale Zeitschritt der simulierten Zeit $3.600 * 5.000 \text{ ms} = 18.000.000 \text{ ms} = 5 \text{ h}$.

b) Event Manager (EM)

Die Aufgabe des Event Managers ist es, die verschiedenen auftretenden Events zu verwalten und unter den vorgesehenen Bedingungen zur Ausführung zu bringen. In Kap. 4.3.4 (S. 50) werden

⁵⁹ Die Simulationszeit ist die (echte) Zeit, während der die Simulation durchgeführt wird. Die simulierte Zeit ist die Zeit, die in der Simulation benutzt wird (s. a. Glossar).

⁶⁰ Java-Klasse: `java.util.Date`

die unterstützten Eventarten beschrieben. Zum einen gibt es wahrscheinlichkeitsorientierten Events, die mit einer definierten Wahrscheinlichkeit ausgeführt werden, zum anderen sind zustandsbasierte Events vorhanden, die eintreten, wenn Teile des Simulationsmodells einen definierten Status erreicht haben.

I Implementierung

Zentrales Element der Implementierung ist die Klasse `SystemEventManager`. Diese startet einen Hintergrundprozess (definiert in der Klasse `SystemEventJob`), der die Verarbeitung der Events (Klasse `SystemEvent`) übernimmt. In Abbildung 10 werden die zugehörigen Klassen dargestellt.

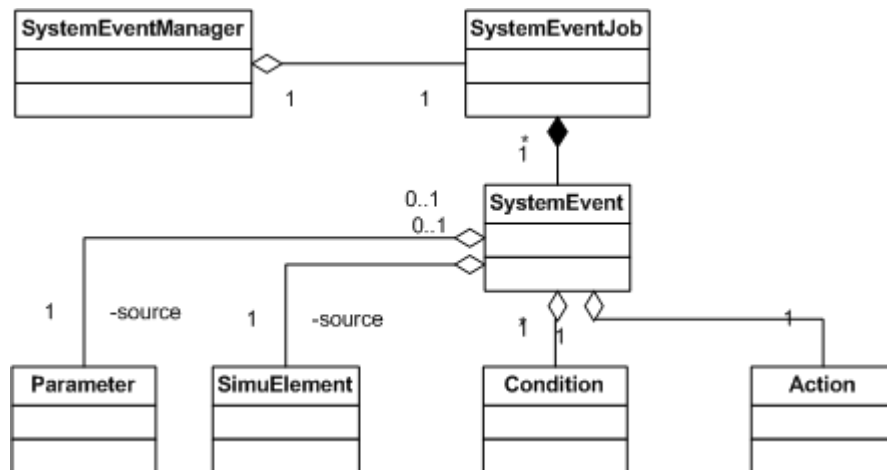


Abbildung 10: Klassendiagramm Event Manager

Die Events selbst werden über den Extension Point `systemEvent` definiert. Abstrakt lassen sich die Informationen, die dieser Extension Point zur Definition einer Extension benötigt, wie in Abbildung 11 gezeigt, abbilden⁶¹.

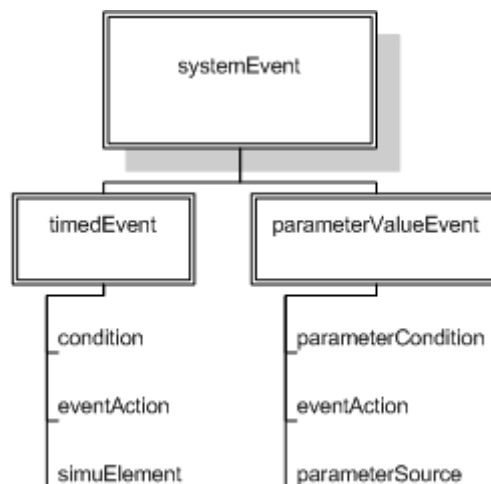


Abbildung 11: Extension Point `systemEvent`

Bei der Definition eines Events muss zunächst unterschieden werden, ob ein zeitorientierter oder ein zustandsbasierter Event vorliegt. Für beiden Event-Arten ist jeweils eine Bedingung, eine Aktion und ein Referenzelement anzugeben. Bei allen drei Elementen handelt es sich um Implementierungen in Java-Klassen. Die Bedingung wird überprüft, wenn der Event durch den

⁶¹ Die originale Dokumentation dieses Extension Points ist in Anhang D enthalten.

Hintergrundprozess behandelt wird: Bei einem zeitorientierten Event gibt es einen festen Zeitpunkt, zu dem das Event ausgeführt wird, bei einem zustandsbasierten Event ist es die Änderung eines Parameters. Die Aktion wird ausgeführt, wenn die Bedingung als erfüllt evaluiert wurde. Das Referenzelement dient dazu, einen Kontext für das Event bereitzustellen: Es wird eine Klasse definiert, mit der dieses Event verbunden ist. Für ein zeitorientiertes Event bedeutet dies, dass es auf der Grundgesamtheit der Objekte dieser Klasse operiert, d.h. bei Eintritt des Events stehen grundsätzlich alle Objekte der Klasse für die Aktion zur Verfügung. Aus Performancegesichtspunkten sollte eine schnelle Einschränkung der Objekte erfolgen, beispielsweise durch eine zufallsbasierte Auswahl. Für ein zustandsbasiertes Event hat der Parameter als Referenzelement die Bedeutung, dass alle Instanzen dieses Parameters auf eine Wertveränderung beobachtet werden.

```

01: <extension point="systemEvent">
02:   <timedSystemEvent
03:     description="The Tenant changes the job and moves out of the build-
04:     ing."
05:     name="TenantChangesJob">
06:       <timedEventCondition
07:         class="TimerCondition"
08:         frequency="3600000"
09:         random="true">
10:       </timedEventCondition>
11:       <eventAction
12:         class="TenantMovesOutAction">
13:       </eventAction>
14:       <simuElement
15:         simuElementId="Tenant">
16:       </simuElement>
17:     </timedSystemEvent>
18:   </extension point>

```

Quellcode 2: Auszug der Datei plugin.xml: Definition eines zeitorientierten Events

Quellcode 2 demonstriert die Definition eines zeitorientierten Events: Dieses Event lässt einen Mieter ausziehen, weil jener die Arbeitsstelle wechselt. In Zeile 14 wird das Referenzelement definiert, dieses Event operiert auf der Menge der Mieter. In der derzeitigen Implementierung wird aus der Menge der Mieter ein Objekt per Zufall ausgewählt. Auf dieses Objekt wird die Prozedur `TenantMovesOutAction` angewendet (Zeile 11). Durch Zufall wird auch das Auftreten des nächsten Ereignisses bestimmt: In Zeile 7 wird die Frequenz definiert, in der dieses Ereignis auftritt: alle 3,6 Millionen Sekunden (also knapp alle 42 Tage) der simulierten Zeit zieht ein Mieter aus. Das Kennzeichen `random` in Zeile 8 sorgt dafür, dass keine äquidistanten Zeitintervalle zwischen aufeinanderfolgenden Ereignissen erzeugt werden, sondern dass das Ereignis zu einem zufälligen Zeitpunkt innerhalb dieses Intervalls auftritt.

In Quellcode 3 ist die Schleife zur Verarbeitung der Events in der Klasse `SystemEventJob` dargestellt: Die Methode `getNextEvent()` in Zeile 4 liefert das als nächstes auszuführende Event zurück. Dieses Event muss aber noch nicht fällig sein, deshalb wird in Zeile 6 die Zeit berechnet, die noch gewartet werden muss, bis das Event eintritt. Das Warten selbst wird durchgeführt in Zeile 8. In Zeile 10 wird geprüft, ob die Bedingung, unter der die Prozedur (`action`) dieses Events aktiv werden soll, erfüllt ist. Ist das der Fall, wird sie ausgeführt. In Zeile 15 wird das Event wiederum in die Warteschlange eingefügt, wenn es denn mehrmals auftritt.

```

01: while (!isCanceled()){
02:   waitingEvents = getWaitingEvents();
03:   synchronized(waitingEvents) {

```

```
04:         ISystemEvent nextEvent = getNextEvent();
05:         long nextAdvisedCall =
            nextEvent.getCondition().nextAdvisedCall(_context);
06:         long timeToWait = nextAdvisedCall - System.currentTimeMillis();
07:         if (timeToWait > 0) {
08:             waitingEvents.wait(timeToWait);
09:         }
10:         if (nextEvent.getCondition().isFulfilled(_context)) {
11:             IEventAction action = nextEvent.getAction();
12:             action.run(_context);
13:         }
14:         if (!nextEvent.runOnce()) {
15:             addEvent(nextEvent);
16:         }
17:     }
18: }
```

Quellcode 3: Klasse SystemEventJob: Schleife zur Eventverarbeitung

5 Szenarien ausgewählter bauphysikalischer Probleme

5.1 Basis des Szenariodesigns

Das Grundelement der Spielplattform sind Szenarien, die einzeln zur Plattform beigetragen werden können. Roger C. Schank stellt in einer grundlegenden Arbeit Richtlinien für den Entwurf von Szenarien auf (Schank, Fano, Bell & Jona, 1994). Seine Arbeit bezieht sich zwar auf Lernumgebungen. Dass die Regeln auf Spiele anwendbar sind, zeigen die zahlreichen Beispiele, die dem Bereich der Computerspiele entstammen. Die Regeln werden hier in ihren wesentlichen Punkten dargestellt und werden als Basis für das Vorgehen beim Szenariodesign benutzt:

- **Authentischer Kontext:** Das Szenario soll ein Problem aus der realen Welt beschreiben. Das erhöht die Chance, dass es auf die Interessen des Studenten trifft, der damit motivierter ist, seine Fähigkeiten am Szenario zu erproben und zu stärken.
- **Ziel:** Das Ziel der Szenarios ist das Erlernen von Fähigkeiten. Eine Fähigkeit ist definiert als „knowing how to do something“.
- **Titelgeschichte:** Ein Szenario wird angeführt von einer Titelgeschichte, die den Inhalt darstellt und dem Szenario einen Kontext verleiht.
- **Mission:** Jede Mission hat einen eindeutigen Schwerpunkt. Der Schwerpunkt kommt aus der Gruppe „Steuerung“, „Entwurf“, „Entdeckung“ oder „Erklärung“. Der Schwerpunkt „Erklärung“ - das Erstellen eines erklärenden Artefakts - ist in einem Spiel weniger gut umsetzbar und zusammen mit dem Schwerpunkt „Entdeckung“ indirekt als Ziel messbar - indem der Spieler die richtigen Entscheidungen trifft und geeignet steuert oder entwirft.
- **Missionsziele:** Eine Mission besitzt Ziele. Deren Erfüllung ist eindeutig messbar.

Die Regeln von Schank geben den Rahmen für den Entwurf eines Szenarios vor, die Detailumsetzung in ein Objektmodell wird durch die Ausführungen in Kap. 4.3 beschrieben.

5.2 Szenariobeschreibung

Die Beschreibung der Szenarien ist in verschiedene Informationsteile strukturiert, die in Tabelle 10 genannt werden. Die in den Folgekapiteln vorgestellten konkreten Szenariobeschreibungen befassen sich mit Grundlagen der Bauphysik und sind relativ gut verständlich. Es liegt nicht im Fokus dieser Arbeit, diese detailliert zu erläutern⁶². Die wichtigsten Begriffe werden im Glossar dargestellt.

Information	Beschreibung
Szenariotitel	Das Szenario trägt einen zum Thema des Szenarios passenden Titel sowie einen Identifikator, der Verwechslungen ausschließen soll.
Lernziele	Es werden die Lernziele des Szenarios beschrieben. Nach dem Ansatz von Schank (Schank et al., 1994) werden Lernziele an Fähigkeiten geknüpft. Fähigkeiten lassen sich in der Form „in der Lage sein, etwas zu tun“ („knowing how to do

⁶² Zur Vertiefung der bauphysikalischen Grundlagen können Übersichtswerke und Lehrbücher wie z.B. Bläsi (2002), Hens (2008) oder Willems, Schild & Dinter (2006) benutzt werden.

	something“) ausdrücken. Um die Beschreibung kurz zu halten, wird in der formalen Beschreibung der „in der Lage sein“-Teil weggelassen und nur die Beschreibung des „etwas zu tun“ gegeben.
Titelgeschichte	Mit der Titelgeschichte des Szenarios soll das Interesse des Spielers geweckt werden. Anhand der Titelgeschichte entscheidet der Spieler, ob das Szenario für ihn attraktiv ist.
Szenarioaufbau	Der Punkt „Szenarioaufbau“ erläutert in textueller Form den Aufbau des Szenarios und dient als Dokumentation für den Szenariodesigner. Es werden auch die wesentlichen Elemente des zugehörigen Objektmodells beschrieben. Zum Objektmodell zählen Simulationselemente, Parameter, Ereignisse, mögliche Spieleraktionen und Missionen.
Wirkungsgeflecht: Systemgrafik	Das dem Szenario zugrundeliegende Wirkungsgeflecht wird in einer Systemgrafik beschrieben, die in einer an <i>System Dynamics</i> angelehnten Symbolik verfasst ist. (s. Kap. 5.2.6.)
Wirkungsgeflecht: Abhängigkeitsmatrix	Eine Abhängigkeitsmatrix zeigt die Abhängigkeiten der verschiedenen Größen untereinander auf. Es wird ersichtlich, welche Größen für die Berechnung anderer Größen von Bedeutung sind, und welchen Einfluss sie auf diese ausüben.
Wirkungsgeflecht: Berechnungsvorschriften	Die Berechnungsvorschriften werden in Form von Formeln dargestellt. In den Fällen, in denen keine einzelne, abgeschlossene Formel angegeben werden kann, wird die Berechnungsvorschrift in Form eines Algorithmus angegeben, der mit Hilfe von Pseudocode dargestellt wird.
Missionen	Mit einem Szenario sind Missionen verbunden. Die Missionen werden dem Spieler in der Regel sequentiell angeboten, d.h. wenn eine gelöst wurde, wird ihm die darauffolgende angeboten. Damit besitzen die Missionen eine definierte Reihenfolge. Die Struktur einer Missionsbeschreibung wird im Folgenden beschrieben:
Mission: Ziele	Hier werden die konkreten Ziele beschrieben. Die Ziele müssen derart formuliert sein, dass algorithmisch überprüft werden kann, ob sie erreicht wurden.
Mission: Beschreibung	Die Mission und die dahinterstehenden Absichten werden beschrieben.
Mission: Belohnungen	Die Belohnungen für eine erfolgreich durchgeführte Mission werden hier angegeben ⁶³ .
Tags	Schlagworte zur Kategorisierung des Szenarios und seiner Lernziele. Sie dienen der schnelleren Auffindung thematisch zusammenhängender Szenarien.
Vorausgesetzte Fähigkeiten	Diese Information dient der Steuerung der Szenarioauswahl: Dem Spieler sollen die der Entwicklung seiner Fähigkeiten passenden Szenarien angeboten werden (s. Kap. 5.2.2).

Tabelle 10: Struktur der Beschreibung eines Szenarios

⁶³ Von höherer Bedeutung bei der Nennung hier ist die Art der Belohnung. Die Quantität kann im Rahmen des Game Balancing einfacher angepasst werden. Zur Belohnung zählen immer Coins und XPs. Gewöhnlich werden auch Fähigkeitspunkte ausgeschüttet. Bisher noch nicht umgesetzt wurde die Prämierung der Spielerbemühungen mit speziellen Gegenständen, die in weiteren, folgenden Missionen genutzt werden können bzw. auch eingesetzt werden müssen.

5.2.1 Definition der Lernziele

Zur formalen Definition der Lernzielen werden die Definitionen von Schank et al. (1994) sowie von Anderson & Krathwohl (2001) benutzt. Schank schlägt vor, die Formulierung des Lernziels in der „in der Lage sein, etwas zu tun“-Form durchzuführen. Dieses lässt sich auch ausdrücken als „Besitz einer Fähigkeit“. Bei Anderson & Krathwohl wird eine Fähigkeit aufgefasst als eine Kombination von Substantiv und Verb: Mit einer Sache etwas tun können. Zusammenfassend wird ein Lernziel definiert als Erwerb einer Fähigkeit. Eine Fähigkeit wiederum bezieht sich auf den Umgang mit einem Objekt. Beispiel ist das Lernziel *F001* aus dem Szenario in Kap. 5.5.1a):

Der Spieler kann den Einfluss von Fenstergröße und Fensterart auf die (Transmissions-) Wärmeverluste einschätzen.

Substantivisch wird hier *Einfluss von Fenstergröße und Fensterart auf (Transmissions-) Wärmeverluste* benutzt, als zugehöriges Verb fungiert *einschätzen*.

Ein weiteres Beispiel stammt aus Kap.: 5.5.3:

[F013] *Der Spieler kann die DIN 4108-2 anwenden.*

Als Substantiv wird in diesem Beispiel die *DIN 4108-2* benutzt, *anwenden* ist das zugehörige Verb.

Die Lernziele sind zu unterscheiden nach fachlichen Lernzielen sowie spielinternen Lernzielen. Fachliche Lernziele dienen dem Erwerb fachlicher Fähigkeiten aus dem Bereich der Bauphysik. Spielinterne Lernziele hingegen dienen dem Umgang mit der Spielplattform. Diese sind entsprechend gekennzeichnet.

5.2.2 Fähigkeiten

Die Aufnahme der geförderten und vorausgesetzten Fähigkeiten in die Szenariobeschreibung dient der adaptiven Szenarioauswahl für den Spieler. Das erfolgreiche Lösen einer Mission eines Szenarios belohnt den Spieler mit einem Zuwachs in der geförderten Fähigkeit. Um eine Arbeitsgrundlage für die Entwicklung der Plattform zu haben, wird festgelegt, dass eine Fähigkeit dann als erworben gilt, wenn drei Missionen gelöst wurden, die diese Fähigkeit fördern. Diese Annahme kann durch Rückmeldungen aus dem Betrieb der Plattform angepasst werden und muss an den Stellen aufgeweicht werden, an denen noch keine 3 Missionen vorhanden sind, die eine Fähigkeit fördern. In diesem Falle müssen alle auf der Plattform vorhandenen Missionen bezüglich dieser Fähigkeit gelöst werden.

5.2.3 Basiskonfigurationen sowie Basis-, Ergänzungs- und Kombinationsszenarien

Gemäß James Paul Gee (2005) ist eines der Lernprinzipien, die in erfolgreichen Spielen unterstützt werden, die Möglichkeit, Laborbedingungen in Spielsituationen zu erzeugen („fish tank“). Bezogen auf Szenarios bedeutet dies, dass sie nur Elemente enthalten, die notwendig sind, um die Inhalte des Lernzieles zu demonstrieren. Dieses wird folgendermaßen umgesetzt:

- **Basisszenario:** Es gibt in der Regel ähnlich strukturierte Basisszenarien, in denen jeweils ein grundlegendes Phänomen demonstriert wird.
- **Basiskonfiguration:** Der Grundaufbau an Simulationselementen dieser Szenarien ist in einer Basiskonfiguration definiert. Die Standardbasiskonfiguration ist ein Einraumhaus.

- **Ergänzungsszenario:** Aufbauend auf einem Basisszenario gibt es weitere Szenarien, die dieses um zusätzliche Phänomene ergänzen, solche Szenarien werden Ergänzungsszenarien genannt.
- **Kombinationsszenario:** Szenarien, die verschiedene Ergänzungsszenarien desselben Basisszenarios bzw. derselben Basiskonfiguration kombinieren, werden Kombinationsszenarien genannt. Sie fordern den Spieler, mehr als eine Systemvariable gleichzeitig im Blick zu haben.
- **Grundsszenario:** Grundsszenario ist der Sammelbegriff für die Szenarien, aus denen ein Kombinationsszenario gebildet wird.

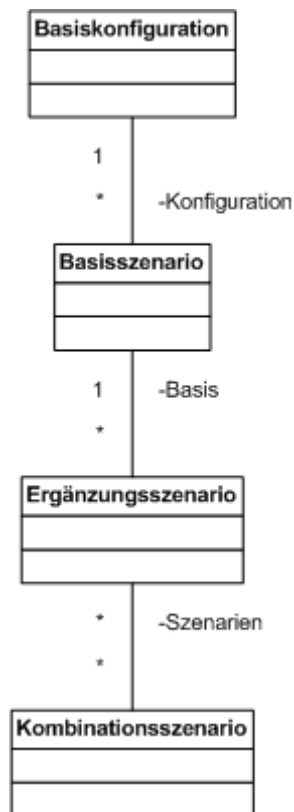


Abbildung 12: Klassendiagramm Szenario und Konfiguration

Abbildung 12 zeigt die Zusammenhänge zwischen den Szenarioarten: Einem Basisszenario ist immer eine Basiskonfiguration zugeordnet. Ein Ergänzungsszenario baut auf einem Basisszenario auf und erweitert dieses bzw. auch die zugrundeliegende Basiskonfiguration. In einem Kombinationsszenario werden mehrere Ergänzungsszenarien zusammengeführt. Voraussetzung ist, dass die Ergänzungsszenarien auf derselben Basiskonfiguration beruhen - und damit dieselbe Modellstruktur besitzen.

5.2.4 Beschreibungsschema einer Basiskonfiguration

Eine Basiskonfiguration wird schematisch beschrieben. Im Folgenden sind die verschiedenen Elemente des Schemas aufgeführt:

Konfigurationsaufbau: Wichtigstes Element der Beschreibung einer Basiskonfiguration ist die Darstellung des Modells der Simulationselemente. Der Konfigurationsaufbau bildet eine Basis und kann, muss aber nicht, in den konkreten Szenarien geändert werden. Zum Konfigurationsaufbau zählen auch Benutzeraktionen und Systemereignisse, sofern diese schon auf dieser Ebene

ne Sinn machen. Ein Beispiel für eine sinnvolle Benutzeraktion ist „Einbau neuer Fenster“, wenn der Konfigurationsaufbau bereits Fenster enthält. Zur Demonstration eines nicht sinnvollerweise im Konfigurationsaufbau zu beschreibenden Systemereignisses kann das Systemereignis „Die gesetzlichen Vorschriften des Wärmeschutzes wurden überarbeitet. Ab sofort ist eine neue Version der EnEV einzuhalten.“ herangezogen werden - ein solches Ereignis ist auf dieser Ebene der Beschreibung nicht richtig platziert, wenn durch den Aufbau des Simulationselementmodells noch nicht klar ist, ob Wärmeschutz enthalten ist.

Größenverzeichnis: In einem Größenverzeichnis werden diejenigen Parameter und Attribute aufgeführt, die schon in einer Basiskonfiguration vorhanden sind. Zum überwiegenden Teil sind dieses Größenangaben.

Missionen: Missionen sind Aufgaben, die die Spieler lösen sollen, um ein Spielziel des jeweiligen Szenarios zu erreichen. Im Rahmen der Basiskonfiguration definierte Missionen beziehen sich auf eine Änderung dieser Basiskonfiguration, wie beispielsweise eine bauliche Änderung. Sie beziehen sich nicht auf Ziele, die mit der Basiskonfiguration noch gar nicht dargestellt werden können, weil eventuell noch notwendige Parameter fehlen, die erst später - z. B. im Rahmen eines Ergänzungsszenarios - definiert werden.

5.2.5 Beschreibungsschema eines Kombinationsszenarios

Die Beschreibung eines Kombinationsszenarios lehnt sich an die eines Basis- oder Ergänzungsszenarios an. Jedoch ergeben sich aus der Kombination mehrerer Szenarien einige Besonderheiten. Diese werden im Folgenden beschrieben:

Information	Bemerkung
Lernziele	Neben den Lernzielen der Grundszenarien haben solche Szenarien weitere Lernziele, die sich aus dem Zusammenspiel der vorher isoliert betrachteten Phänomene ergeben, wie beispielsweise die gegensätzlichen Auswirkungen einer Aktion auf unterschiedliche Phänomene.
Beschreibung (Titelgeschichte)	Ein Kombinationsszenario hat gewöhnlich keine Titelgeschichte. Stattdessen wird die Titelgeschichte eines der Grundszenarien übernommen. In der Beschreibung werden mögliche Zielkonflikte genannt. Sie hebt bestimmte Missionen zur Steuerung des Spielers hervor.
Szenarioaufbau	Der Szenarioaufbau ist grundsätzlich der kumulierte Aufbau der zugrundegelegten Szenarien, d.h. wenn ein Simulationselement in allen Szenarien vorkommt, so wird es beibehalten. Wenn es nur in einem Szenario vorhanden ist, so gehört es auch zum resultierenden Szenarioaufbau.
Wirkungsgeflecht	Das Wirkungsgeflecht wird nur dann erweitert beschrieben, wenn zwischen den Wirkungsgeflechten der Grundszenarien neue Abhängigkeiten entstehen. Dieses ist jedoch gewöhnlich nicht der Fall.
Missionen	Für ein Kombinationsszenario müssen nicht notwendigerweise neue Missionen definiert werden. Vorgeschlagen wird, die Missionen eines Basisszenarios zu übernehmen. Damit die Anforderungen aus den anderen Szenarien ebenfalls in das Kombinationsszenario eingehen, sollten diejeni-

	gen Missionen der anderen Szenarien mit den qualitativ schärfsten Anforderungen zunächst angeboten werden: Zur Lösung dieser Missionen sind Aktionen des Spielers notwendig: Wenn er bei solchen Aktionen gleich die Ziele der anderen Missionen im Auge hat, so lassen sich virtuelle Ressourcen sparen. Beispielsweise ist nur ein Fenstertausch notwendig, wenn das eingebaute Fenster sowohl die Anforderungen an den Schallschutz als auch die Anforderungen an den Wärmeschutz erfüllt.
Mission: Belohnungen	Zu den Belohnungen von Missionen zählen Fähigkeitspunkte. Fähigkeiten, die mit den Missionen der Kombinationsszenarien erworben werden können, sind zum einen die Fähigkeiten der zugehörigen Grundszenarien. Zum anderen gibt es Fähigkeiten, die die gleichzeitige Optimierung mehrerer Zielgrößen anzeigen.
Tags	Tags dieses Szenarios sind zum einen diejenigen der Grundszenarien, zum anderen Tags, die die kombinierte Betrachtung verschiedener Lernziele bzw. hier bauphysikalischer Phänomene belegen, wie beispielsweise „Kombination Wärmeschutz Akustik“.
Vorausgesetzte Fähigkeiten	Als vorausgesetzte Fähigkeiten werden die Lernziele der Szenarien gesehen, die die Grundlage für dieses Szenario bilden. Da diese Fähigkeiten nun kombiniert anzuwenden sind, sollten sie vorher isoliert ausreichend trainiert worden sein.

Tabelle 11: Beschreibung eines Kombinationsszenarios

5.2.6 Identifikatoren

Fähigkeiten und Szenarien werden in textueller Form angegeben. Um Referenzen zu erleichtern, werden Identifikatoren genutzt: Jede Fähigkeit (und damit jedes Lernziel) sowie jedes Szenario besitzt einen Identifikator.

Die hier benutzten Identifikatoren sind teilweise sprechend⁶⁴:

Die **Art des Szenarios** wird im Identifikator wiedergegeben: Der Anfangsbuchstabe eines Szenario-Identifikators ist entweder ein „B“ (Basisszenario), „E“ (Erweiterungsszenario) oder „K“ (Kombinationsszenario). Die Basiskonfiguration wird mit „BK“ benannt.

Die Identifikatoren der Fähigkeiten beginnen mit einem „F“, spielinterne Fähigkeiten mit „FS“.

5.2.7 Tags

Zur Klassifikation der Szenarien werden Tags eingesetzt. Tags sind Schlagworte, mit denen entsprechende Objekte (hier: Szenarien) kategorisiert werden können (Carlin, 2006). Die Vergabe von mehreren Schlagworten für ein Objekt ermöglicht dessen Einordnung zum leichteren Wiederauffinden ohne den Begrenzungen einer möglicherweise künstlichen, nicht immer pas-

⁶⁴ Von einem sprechenden Schlüssel (oder einem klassifizierenden Schlüssel) wird gesprochen, wenn vom Schlüssel bzw. von Teilen des Schlüssels auf das zu identifizierende Objekt geschlossen werden kann. Beispiel ist ein KFZ-Kennzeichen in Deutschland: Der erste Teil identifiziert den Kreis, in dem das KFZ angemeldet ist (Dworatschek, 1989, S. 317).

senden Hierarchie ausgesetzt zu sein. Tags werden insbesondere im Rahmen des Social Tagging im Web 2.0 genutzt (Peters, 2009).

Um beim Entwurf der Szenarien größere Freiheit zu besitzen und nicht auf einen fertigen Katalog von Lernzielen angewiesen zu sein, sollen Tags hier ähnliche, aber nicht identische Lernziele miteinander assoziieren. Damit soll dem Szenariodesigner eine bessere Orientierung ermöglicht werden. Der Spieler hat hierdurch die Möglichkeit, bei erwachendem Interesse ähnliche Szenarien zu finden, in denen er sein Wissen anwenden, vertiefen und festigen kann.

Die Tags werden im Rahmen der Plattform vom Szenariodesigner vergeben. Um Kandidaten für Tags systematisch finden zu können, werden die folgenden Quellen vorgeschlagen:

- **Fachgebiete:** Zur groben Klassifikation ist das Fachgebiet geeignet, z.B. können so alle Szenarien identifiziert werden, die im Fachgebiet des Wärmeschutzes angesiedelt sind.
- **Substantive der Lernziele:** Die Substantive der Lernziele beschreiben in erster Näherung die fachlichen Inhalte. Da es sich bei den Verben tendenziell mit höherer Wahrscheinlichkeit um Beschreibungen kognitiver Vorgänge (z.B. Erinnern, Aufzählen) handelt, sind die Substantive eher zu einer fachlichen Einordnung des übergeordneten Szenarios geeignet.
- **Substantive der Szenario-Titelgeschichte:** Mitunter sind auch in der Titelgeschichte Fachbegriffe enthalten, die zu einer Kategorisierung genutzt werden können.
- **Verben der Lernzielbeschreibung:** Handelt es sich bei den Verben der Lernzielbeschreibung um fachbezogene Tätigkeiten, dann können auch diese zur Kategorisierung herangezogen werden.

5.2.8 Systemnotation

Die Spielplattform dient der Übung des Umgangs mit Systemen. Daher stehen Systeme im Mittelpunkt der Szenarien. Zur Unterstützung des Szenariodesigns müssen die zugrundeliegenden Systeme dargestellt werden. Aus diesem Grund wird hier das benutzte Schema für die Darstellung von Systemen beschrieben.

Die Darstellung der Systeme ist ebenso wie die der Szenarien kumulativ: Bezieht sich ein Basis-szenario auf eine Basiskonfiguration, so werden im Basisszenario lediglich die zusätzlichen Elemente dargestellt. Genauso verhält es sich bei den in Ergänzungsszenarien gegebenen Informationen: Diese werden dort nur aufgeführt, wenn sie im Basisszenario noch nicht vorhanden sind⁶⁵.

a) Systemgrafik

Die Basis des Darstellungsschemas ist das der Methode *System Dynamics*. Sie wurde von Jay Forrester eingeführt und dient als Hilfsmittel zum Untersuchen und Verstehen des Verhaltens komplexer Systeme (Forrester, 1961, 1994)⁶⁶. Es existieren mehrere Softwarepakete zur Modellierung von Systemen unter Nutzung der System Dynamics-Methode. Eines davon ist das Modellierungswerkzeug Vensim®, dessen grafischer Editor zum Erstellen der Diagramme benutzt

⁶⁵ Diese Darstellung neigt auf Papier zur Unübersichtlichkeit. Sie vermeidet aber Redundanz und bietet daher bei der Konsistenzhaltung der verschiedenen Szenarien und Konfigurationen Vorteile.

⁶⁶ Wesentliche Merkmale, die ein System als komplex (und schwer verständlich) erscheinen lassen, sind Rückkopplungseffekte und Zeitverzug.

wird⁶⁷. Tabelle 12 erläutert die in den Grafiken verwendeten Symbole der Notation (Muetzelfeldt & Massheder, 2003; Simulistics, 2003).

Element	Beschreibung	Symbol
Compartment	Status-Variable; Menge; ist nicht Ziel eines <i>Influence</i> -Pfeiles, Änderungen erfolgen nur über <i>Flows</i> . Abbildung: Rechteck, das den Namen einschließt (Synonym: stock, level, dt. Lager)	
Flow	Fluss, entspricht einem Prozess, Einheit entspricht der eines <i>Compartment</i> s pro Zeit. Abbildung: Pfeil mit stilisiertem Ventil (dt. Fluss oder Rate)	
Cloud	Ähnlich wie ein <i>Compartment</i> , nur dass die Menge unbekannt oder nicht wichtig ist für das spezifizierte System, entspricht einer Größe außerhalb des Systems, kann nicht Ziel oder Quelle eines <i>Influence</i> -Pfeiles sein. Abbildung: Wolke	
Variable	Dient der Abbildung von Parametern, Konstanten und Zwischenergebnissen Abbildung: Name als Text in der Grafik	
Influence	Dient der Darstellung des Einflusses einer Größe auf eine andere, Quelle kann ein <i>Compartment</i> , ein <i>Flow</i> oder eine <i>Variable</i> sein, das Ziel nur ein <i>Compartment</i> oder ein <i>Flow</i> . Abbildung: Pfeil	
Equation	Gleichung, die beschreibt, wie der Wert einer Variablen berechnet wird. Abbildung: In der Grafik gibt es keine entsprechende Abbildung, eine Gleichung ist in der textuellen Modellbeschreibung zu finden.	

Tabelle 12: Elemente der Notation zur Systemdarstellung

Abbildung 14 auf S. 77 zeigt ein einfaches Beispiel für eine Systemgrafik.

b) Abhängigkeitsmatrix

Zur Darstellung wird auch eine Tabelle als Abhängigkeitsmatrix verwendet. In dieser Matrix werden alle quantifizierbaren Größen (d.h. *variables* (Variablen), *compartments* und *flows*) des Modells sowohl als Zeilen als auch als Spalten aufgelistet. Die Tabelle bildet die Beziehungen „beeinflusst“ und „wird beeinflusst von“ ab. Eine Zeile enthält den Namen einer Größe und für jede Größe, von der sie direkt beeinflusst wird, einen Eintrag, der entweder „+“ (bei einer positiven Beeinflussung⁶⁸), „-“ (bei einer negativen Beeinflussung) oder „o“ (bei einer ungerichteten, diffu-

⁶⁷ Das Modellierungswerkzeug Vensim wird von der Firma Ventana Systems vertrieben. In der Variante Vensim PLE („Personal Learning Edition“) kann es kostenlos zu Lernzwecken und für den persönlichen Gebrauch genutzt werden (Ventana, 2012).

⁶⁸ „Positive Beeinflussung“ bedeutet, dass für einen höheren Wert der beeinflussenden Größe auch die beeinflusste Größe einen höheren Wert besitzt. Für das Gegenteil - negative Beeinflussung - gilt, dass ein niedriger Wert der beeinflussenden Größe einen höheren Wert der beeinflussten Größe verursacht. Mathematisch ausgedrückt gibt positiv oder negativ das Vorzeichen der Steigung eines Graphen mit der be-

sen Abhängigkeit⁶⁹) sein kann. Somit lässt sich eine Übersicht über die gegenseitigen Einflüsse der Größen erhalten⁷⁰. Um die Übersichtlichkeit zu wahren, sind jedoch nur direkte Abhängigkeiten in der Tabelle aufgeführt, d.h. es existiert nur ein Eintrag in einer Zelle, wenn eine Größe direkt in die Berechnung einer anderen Größe eingeht. Eine solche Abhängigkeitsmatrix kann daher auch rekursiv gelesen werden.

Variablen als eine der betrachteten Größen werden noch weiter kategorisiert nach dem Merkmal der Herkunft ihres Wertes (Simulistics, 2003). In Tabelle 13 werden die verschiedenen Kategorien beschrieben.

Typ	Abkürzung	Beschreibung
Konstante	K	Eine Größe wird dann als konstant markiert, wenn sie sich im Zeitablauf - außer durch den Eingriff des Spielers ⁷¹ - nicht ändert.
Exogene Variable	E	Eine solche Variable hat Einfluss auf das Modell, wird aber nicht durch dieses verändert. Ein Beispiel ist die Außentemperatur.
Ausgabevariable	A	In einer Ausgabevariablen werden Berechnungsergebnisse gespeichert, die aber nicht weiter zur Berechnung des Modells benötigt werden.

Tabelle 13: Kategorisierung von Variablen

Die Kategorisierung erleichtert die Nutzung der beschriebenen Abhängigkeitsmatrix. Konstanten und exogene Variablen werden durch keine anderen Größen des Modells beeinflusst, somit ist eine Zeile mit einer solchen Größe ohne Einträge. Auch um solche Zeilen zu kennzeichnen, wird die Abhängigkeitsmatrix zu Anfang um eine Typ-Spalte erweitert, in der die in Tabelle 13 beschriebenen Abkürzungen eingetragen werden.

Eine Ausgabevariable beeinflusst keine andere Größe des Modells, somit ist die entsprechende Spalte ohne Eintrag.

Ein Beispiel für eine Abhängigkeitsmatrix ist auf S. 78 (Tabelle 21) zu finden. In der Abhängigkeitsmatrix eines Erweiterungsszenarios sind aus Gründen der Übersichtlichkeit nur diejenigen Größen in den Zeilen aufgeführt, die gegenüber dem Basisszenario eine veränderte Herleitung aufweisen.

einflussenden Größe auf der X-Achse und der beeinflussten Größe auf der Y-Achse an - bei ansonsten konstanten anderen Formelbestandteilen.

⁶⁹ Ein Beispiel für eine solche Abhängigkeit entstammt dem Themengebiet der solaren Wärmegewinne: Der *Abminderungsfaktor Himmelsrichtung* ist von der Himmelsrichtung abhängig. Die Himmelsrichtung ist keine Mengengröße, sondern besitzt einen stetigen, abgeschlossenen Wertebereich. Über diesen Wertebereich verläuft der Graph der Funktion des *Abminderungsfaktors Himmelsrichtung* nicht ausschließlich fallend oder steigend.

⁷⁰ Erfolgreich angewendet werden konnte dieses Darstellungselement u.a. bei der Erstellung des an der Bauhaus-Universität Weimar entwickelten Verkehrssimulationsspiels „Mobility“, das auf wissenschaftlich abgesicherten Simulationsmodellen beruht (Brannolte, Griesbach, Harder & Kraus, 2000).

⁷¹ Ein Beispiel für eine vom Spieler verursachte Veränderung einer Konstanten ist der Einbau von neuen Fenstern, der ansonsten als konstant betrachtete Werte wie U-Wert und Schalldämmmaß für das Szenario ändert.

c) Größenverzeichnis

Zu jedem Szenario wird ein Größenverzeichnis angelegt, in dem die verwendeten Größen mit- samt verwendeten Einheiten präzise dargestellt und erläutert werden. Das Größenverzeichnis dient in erster Linie dem Szenarioentwickler zur Dokumentation des Szenarios. Nicht alle hier beschriebenen Größen sind letztendlich auch für den Spieler sichtbar. Die Größenverzeichnisse sind kumulativ. Für jede Größe werden die folgenden Informationen aufgeführt:

- **Name:** Dieser Name wird für die Größe benutzt, u.a. auch in der Systemgrafik, der Ab- hängigkeitsmatrix und der Liste der Berechnungsvorschriften.
- **Symbol:** Diese Formelzeichen wird für die Größe u.a. in den Berechnungsvorschriften benutzt. Obwohl bei der Vergabe des Symbols versucht wurde, etwaige Konventionen zu beachten, können sich im Einzelfall Abweichungen ergeben.
- **Einheit:** Das ist die für diese Größe genutzte Einheit.
- **Definition:** In der Definition wird festgelegt, um welche Größe es sich handelt. Liegt eine in der Fachliteratur eindeutig bekannte Größe vor, so wird die Verbindung an dieser Stel- le hergestellt.

d) Berechnungsvorschriften

Für alle Größen, die Einträge in der zugehörigen Zeile der Abhängigkeitsmatrix aufweisen, wird eine Berechnungsvorschrift für den Wert dieser Größe zum Zeitpunkt t_n angegeben. Sofern in diesen auf andere Größen des Modells Bezug genommen wird, wird der Wert dieser Größen zum Zeitpunkt t_{n-1} genutzt. Um die Übersichtlichkeit zu wahren, werden die Indizes nicht extra dargestellt. Wenn nichts anderes angegeben wurde, gilt, dass auf der rechten Seite der Glei- chung (sowie in Bedingungen, wie z.B. in Pseudocode) immer die Werte der vorherigen Rechen- schritt genutzt werden (Index: $n-1$) und auf der linken Seite einer Gleichung die Werte des aktu- ellen Rechenschrittes (Index: n) gemeint sind.

Zur Darstellung der Berechnungsvorschriften werden unterschiedliche Methoden angewendet. Die erste ist die einer **mathematischen Formel**. Dieses ist die bevorzugte Darstellungsweise, da sie prägnant und übersichtlich ist. In einigen Fällen werden in einer Berechnungsvorschrift Fall- unterschieden durchgeführt. Diese lassen sich nur unübersichtlich in einer Formel ausdrü- cken. Daher wird in solchen Fällen die Berechnungsvorschrift durch Pseudocode dargestellt. Pseudocode ist die verbale Beschreibung eines Algorithmus - oft unter Verwendung von einfa- chen Programmiersprachkonstrukten, wie beispielsweise Wenn-Dann-Strukturen. Dabei ist die Beschreibung aber programmiersprachenunabhängig und durch den Anteil an natürlicher Spra- che leicht verständlich (Dworatschek, 1989; McConnell, 2005).

Schlüsselwort	Bedeutung
IF ... THEN	Beginn einer Abfrage, zwischen IF und THEN steht die Bedingung. Die Anweisungen des darauffolgenden Blockes werden ausgeführt, wenn die Bedingung erfüllt ist. Der folgende Block endet beim zugehörigen Schlüsselwort ELSE - sofern dieses nicht vorhanden ist, beim entspre- chenden Schlüsselwort ENDIF.
ELSE	ELSE ist das Schlüsselwort, mit dem der alternative Block einer IF- THEN-ELSE Anweisung beginnt, d.h. der Block, welcher ausgeführt wird, wenn die Bedingung nicht erfüllt ist. Ein ELSE-Block ist optional.
ENDIF	ENDIF schließt entweder einen IF oder einen ELSE-Block ab.

// ... Kommentar

Tabelle 14: Verwendete Pseudocode-Syntaxelemente

e) Darstellungsschwerpunkte

Die wesentlichen Dinge, die durch die Szenarien vermittelt werden sollen, sind bauphysikalischer Natur. Im Rahmen der Spielmechaniken werden Zahlungsmittelressourcen in die Szenarien mit eingebunden. Oft spielt Geld eine wichtige Rolle bei den analogen Entscheidungen des realen Lebens, d.h. Entscheidungen in echten bauphysikalischen Problemstellungen erfolgen häufig nach ökonomischen Gesichtspunkten.

Da es in den vorgestellten Szenarien jedoch um die Darstellung bauphysikalischer Zusammenhänge geht, und diese oft einen höheren Komplexitätsgrad als die abhängigen Abrechnungssysteme besitzen, werden kostenbasierte Aspekte der betrachteten Systeme weitgehend in der Darstellung nicht berücksichtigt.

5.3 Übersicht der beschriebenen Szenarien

Hier wird zur besseren Orientierung eine kurze Übersicht der beschriebenen Szenarien, der benutzten Basiskonfigurationen sowie der Lernziele gegeben.

5.3.1 Basiskonfigurationen

Bezeichnung	Erläuterung
Einraumhaus [BK001]	Einfachste Basiskonfiguration: Haus mit quadratischem Grundriss, einem Raum und einem Fenster in jeder Wand, geeignet zur Demonstration von einfachen bauphysikalischen Sachverhalten.
Vierwohnungsbungalow [BK002]	Haus mit quadratischem Grundriss, bestehend aus vier quadratischen Räumen mit jeweils eigenem Eingang. Geeignet zur Demonstration von bauphysikalischen Sachverhalten, die sich auf Innenwände beziehen oder die die Einflüsse von Nachbarräumen darstellen.

Tabelle 15: Verwendete Basiskonfigurationen

5.3.2 Szenarien

ID	Bezeichnung	Typ	Gebiet
B001	Heizwärmebedarf in Abhängigkeit von der Fenstergröße und -qualität (S. 76)	Basis	Wärmeschutz
E001	Entwicklung der Transmissionswärmeverluste in Abhängigkeit von der Dämmungsqualität (S. 81)	Ergänzung	Wärmeschutz
E002	Solare Wärmegewinne (S. 87)	Ergänzung	Wärmeschutz
B002	Schallpegel in Abhängigkeit von der Fenstergröße und -qualität (S. 81)	Basis	Schallschutz
E003	Schalldämmung einer Außenwand (S. 92)	Ergänzung	Schallschutz
E004	Schalldämmung von Zwischenwänden (S. 96)	Ergänzung	Schallschutz
K001	Schalldämmung und Wärmedämmung (S. 100)	Kombination	Schallschutz, Wärmeschutz

Tabelle 16: Übersicht der Szenarien

5.3.3 Lernziele

In der folgenden Übersicht werden alle Lernziele dargestellt und ein Verweis auf die Szenarien gegeben, in denen sie vorkommen. Der Identifikator des Lernziels dient als Sortierkriterium. Anzumerken ist, dass die hier aufgeführten Lernziele nicht mit einem Lehrplan abgeglichen wurde, wie er z.B. von Bernt et al. (1999) aufgestellt wurde, sondern aus den Szenarien abgeleitet

wurden. Es wird vorgeschlagen, ein systematisches Lernzieldesign - und daraus hervorgehend einen Entwurf entsprechender Szenarios - dann durchzuführen, wenn sich dieser Ansatz als tragfähig erwiesen hat.

Identifikator	Lernziel	Szenario
F001	Einschätzen des Einflusses von Fenstergröße und Fensterart auf (Transmissions-) Wärmeverluste	B001
F002	Unterscheiden verschiedener Arten von Dämmstoffen	E001
F003	Kalkulieren von Kosten einer Wärmedämmung	E001
F004	Kalkulieren von Wärmeverlusten	E001
F005	Kalkulieren von Heizwärmekosten im Jahresverlauf	E001
F006	Anwenden der Vorschriften der EnEV 2009	E001
F007	Bewerten der Wirtschaftlichkeit einer Wärmedämmung	E001
F008	Kennen des bauteilspezifischen Kennwertes U-Wert	B001
F009	Berechnen des U-Wertes für ein zusammengesetztes Bauteil	B001
F010	Nennen der maßgeblichen Parameter von solaren Wärmegewinnen	E002
F011	Berechnen von solaren Wärmegewinnen	E002
F012	Recherchieren in DINs	E002
F013	Anwenden der DIN 4108-2	E002
F014	Berechnen des maximal zulässigen Sonneneintragswertes	E002
F015	Anwenden des g-Wertes	E002
F016	Unterscheiden verschiedener Arten von Baumaterialien	E001
F017	Bewerten des Baumaterials anhand des Attributs „Wärmeleitfähigkeit“	E001
F021	Kennen des Parameters „Bewertetes Bauschalldämmmaß“	B002, E003, E004
F022	Berechnen des Gesamtschalldämmmaßes eines einfachen, zusammengesetzten Bauteils	B002, E003, E004
F023	Kennen der gegen charakteristische Geräusche erforderlichen Schalldämmmaße	B002
F024	Kennen des rechnerischen Zusammenhangs von Schalldämmmaß und Masse.	E003
F025	Berechnen des Gesamtschalldämmmaßes einer einschaligen Wand anhand der flächenspezifischen Masse	E003, E004
F026	Berechnung des Gesamtschallpegels durch energetische Addition	E004, E004
F030	Kennen des möglichen Zielkonflikts aus Wärmedämmeigenschaft und Schalldämmeigenschaft eines Materials.	K001
F031	Finden einer optimalen Lösung für divergierende Anforderungen von Wärmeschutz und Schallschutz	K001
FS001	Durchführen von baulichen Veränderungen	B001, B002, E003, E004

Tabelle 17: Lernziele der Beispielszenarien

5.3.4 Missionen

Nachfolgend sind die Missionen aufgeführt. Die Tabelle ist sortiert nach den Identifikatoren der Lernziele.

Mission	Lernziele	Szenario
Finde das beste Fenster aus energetischer Sicht! (S. 80)	F001, F008, F009	B001
Statte das Haus mit einer Wärmedämmung aus! (S. 84)	F002, FS001	E001
Wärmedämmung mit begrenzter Dicke der Dämmung (S. 85)	F003, F004, F005	E001
EnEV 2009: Transmissionswärmeverluste der Außenhülle (S. 85)	F003, F004, F006	E001
Wärmedämmung mit begrenztem Budget (S. 85)	F003, F007	E001
Baumaterial als Wärmedämmung (S. 86)	F004, F016, F017	E001
Richte das Haus optimal nach den Vorschriften der DIN 4108-2 aus! (S. 91)	F012, F013, F014	E002
Baue die optimalen Fenster ein! (S. 91)	F013, F014, F015	E002
Keine Sprachfetzen von draußen (S. 95)	F021, F022, F023, FS001	B002
40 dB für die Wohnung (S. 103)	F021, F022, F025, F026, FS001	E004
Bringe das Gesamtschalldämmmaß der Fassadenwand auf 50 dB! (S. 95)	F021, F022, FS001	B002
Bringe das bewertete Schalldämmmaß der Innenwand auf 60 dB! (S. 103)	F021, F025, F026, FS001	E004
Bringe das Gesamtschalldämmmaß der Wand auf 60 dB! (S. 99)	F021, F025, FS001	E003
Nicht mehr als einen halben Meter (S. 99)	F021, F025, FS001	E003
Richte das Haus optimal aus! (S. 90)	F10, F11	E002
Wechsle die Fenster aus! (S. 73, S. 75)	FS001	BK001, BK002

Tabelle 18: Übersicht der Missionen

5.3.5 Fachliche Präzision

Fachliche Präzision bezeichnet hier die Exaktheit, mit der Größen, die in Szenarien verwendet werden, den tatsächlichen, in der Praxis benutzten Größen entsprechen. Wenn zum Beispiel vom Schalldämmmaß gesprochen wird, so ist klar, dass dies eine Größe ist, die eine akustische Eigenschaft eines Bauteils beschreibt. Fachlich präziser wäre es jedoch zu wissen, ob es sich um das Bauschalldämmmaß (mit Berücksichtigung der Schallübertragung über flankierende Bauteile) handelt. Eine weitere Frage, die sich der Experte stellt, ist, ob das bewertete Schalldämmmaß gemeint ist (ein Einzahl-Wert) oder ob es sich um ein frequenzabhängiges Spektrum handelt. Aus der Ingenieurs-Sicht ist Präzision nicht schädlich: Wenn etwas genauer beschrieben wurde, als es notwendig ist, so ist die Beschreibung dennoch korrekt. Aus der Sicht des Lernenden erschweren nicht relevante Informationen jedoch die Abstraktion des zugrundeliegenden Sachverhalts⁷². Die notwendige Präzision eines Szenarios hängt - ähnlich wie die Genauigkeit eines Simulationsmodells durch den Zweck der Simulation bestimmt wird (s. Anhang C) - ab von den Lernzielen, die der Spieler mit Hilfe des Szenarios erreichen soll. Wenn zum Beispiel in Szenario E004 (S. 100) eine Straße als Schallquelle des Umgebungslärms vorhanden ist, der auf eine Wohnung einwirkt,

⁷² Gee (2003) bezeichnet das Vorgehen, nur die relevanten Informationen darzubieten mit dem Prinzip des „fish tanks“.

so ist es zunächst beispielsweise nicht wichtig zu wissen, an welcher Stelle der Lärm nach Maßgabe der RLS-90⁷³ gemessen werden sollte. Vermittelt werden soll in dem Szenario, dass Verkehrslärmemissionen - sofern sie vorhanden sind - in der Berechnung des Schallpegels in einer Wohnung berücksichtigt werden müssen.

Aus vorgenannten Gründen wurde das sogenannte Größenverzeichnis genutzt, um alle beteiligten Größen möglichst präzise zu beschreiben. Die aufgeführten Berechnungsvorschriften sind ebenfalls exakt. Bei Titelgeschichten, Szenario- und Missionsnamen wurden jedoch Abstriche bei der Präzision gemacht - zugunsten möglichst einfacher Erfassbarkeit im Sinne des Einsatzzwecks: Die Szenarien sind für ein Computerspiel vorgesehen. Eigenschaft eines solchen Spiels ist es gewöhnlich, dass exakte Modelle unter einer nach ästhetischen Gesichtspunkten entwickelten Oberfläche arbeiten (Hunicke, LeBlanc, & Zubek, 2004).

Die vorgeschlagenen vereinfachten Berechnungsalgorithmen können ebenfalls zu fehlender Präzision im Sinne einer Abweichung der verwendeten Größenbenennung von der fachlich üblichen führen. Zum Beispiel wird in Szenario B001 (S. 76) die Größe „*Wärmemenge*“ Q benutzt, die nicht typisch ist für die übliche Betrachtung aus bauphysikalischer Sicht, sondern eine Hilfsgröße für die Nutzung eines vereinfachten Berechnungsverfahrens darstellt. An dieser Stelle wurde sie in der Szenariobeschreibung eingeführt, weil die Raumtemperatur nicht mit Hilfe von instationären Berechnungsmethoden simuliert wird, sondern auf Basis eines Bilanzverfahrens: Ein Raum enthält eine definierte Menge an Wärmeenergie. In einem für die Berechnung angenommenen Zeitschritt wird diese Menge vergrößert um zugeführte Heizenergie und vermindert um entstandene Wärmeverluste. Die resultierende Wärmemenge wird dann wieder umgerechnet auf die Lufttemperatur des Raumes.

5.4 Basiskonfigurationen

5.4.1 Basiskonfiguration: Einraumhaus [BK001]

a) Konfigurationsaufbau

Simulationselemente

- **Wände:** Es gibt ein Haus mit 4 Wänden. Eine Wand besitzt eine Tür, die anderen 3 Wände haben je ein Fenster.
- **Fenster:** Kunststofffenster mit Doppelverglasung
- **Haustür**

Ersatzsimulationselemente

- **Fenster:** Eine Datenbank-gespeiste Liste von verschiedenen Fenstern mit unterschiedlicher Verglasung und unterschiedlichen Maßen. Die jeweils dargebotenen Fenster werden per Zufall ermittelt, um den Spieler bei jedem Einbau zu einer neuen Bewertung der Fenster zu bewegen.

Parameter

- **Fenster**

⁷³ Mit RLS-90 werden „Richtlinien für den Lärmschutz an Straßen“ bezeichnet, die 1990 vom Bundesministerium für Bundesministerium für Verkehr, Bau und Stadtentwicklung herausgegeben wurden.

- **Höhe** und **Breite** (sowohl Wanddurchbruch als auch Fenster selbst), aus diesen Angaben folgt die Oberfläche der Fenster.
- **Wand**
 - **Höhe** und **Breite**, aus diesen Angaben werden abgeleitet:
 - die Oberfläche der Wände
 - das Raumvolumen ⁷⁴
 - **Material**
- **Haustür**
 - Höhe und Breite (sowohl Wanddurchbruch als auch Fenster selbst), aus diesen Angaben folgt die Oberfläche der Haustür.

Ereignisse

- **Zeitsprung**: Die Zeit wird ein halbes Jahr vorgespult. Dadurch herrschen aufgrund der wechselnden Jahreszeiten andere Umgebungsbedingungen.

Spieler-Aktionen

- **Fensterdurchbruch anpassen**: Der Fensterdurchbruch wird angepasst auf die Größe eines auszuwählenden Fensters.
- **Fenster einbauen**: Die ausgewählten Fenster werden eingebaut. Voraussetzung ist, dass sie in den Fensterdurchbruch passen.
- **Zeitsprung**: Der Spieler kann einen Zeitsprung von einem Jahr auslösen, er ist dadurch in der Lage, Verbrauchsgrößen über das Jahr zu ermitteln (Das ist u.a. wichtig für die Berechnung des Energieverbrauchs).

b) Größenverzeichnis

Name	Symbol	Einheit	Definition
Oberfläche Wand	A_W	m^2	Oberfläche einer Wand ohne Abzug von Durchbrüchen u.a. für Fenster und Türen.
Oberfläche Wand netto	$A_{W,net}$	m^2	Oberfläche einer Wand mit Abzug von Durchbrüchen u.a. für Fenster und Türen.
Oberfläche Fenster	A_F	m^2	Oberfläche eines Fensters
Raumvolumen	V_R	m^3	Das Volumen des Einraumhauses.

Tabelle 19: Größenverzeichnis Basiskonfiguration Einraumhaus [BK001]

c) Missionen

I Mission „Wechsle die Fenster aus!“

Ziel

Der Spieler hat sämtliche Fenster des Gebäudes ausgewechselt.

Beschreibung

Diese Mission fordert den Spieler, sich mit den Grundmechanismen des Spiels vertraut zu machen. Neben der Aktion „Fenster einbauen“ muss er zuvor auch die Aktion „Fensterdurchbruch anpassen“ ausführen.

⁷⁴ Mit der Festlegung, dass der Raum quadratisch ist, ergibt sich das Volumen gemäß der Formel:
 $V_R = b * b * h$

Belohnung

Fähigkeitspunkte

- **[FS001]** Durchführen von baulichen Veränderungen

d) Tags

Fenster, Wand

5.4.2 Basiskonfiguration: Vierwohnungsbungalow [BK002]

a) Beschreibung

Das stilisierte Haus besitzt 4 Wohnungen, die aus jeweils einem Raum bestehen. Sie sind mit Fenstern und Türen versehen. Eine Skizze des Grundrisses ist in Abbildung 13 zu sehen:

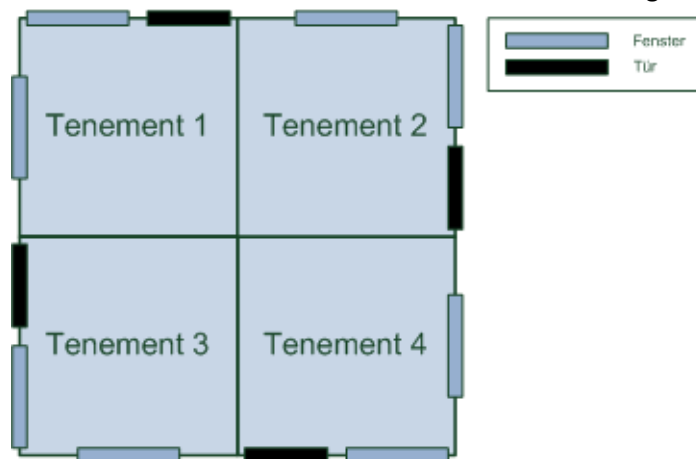


Abbildung 13: Grundriss-Skizze Vierwohnungsbungalow [BK002]

b) Konfigurationsaufbau

Simulationselemente

- **Wände:** Es gibt ein Haus mit 12 Wänden - 8 Außenwände und 4 Innenwände. Jede Außenwand hat ein Fenster, zusätzlich hat eine Wand pro Wohnung eine Tür.
- **Fenster:** Die Fenster besitzen einen Kunststoffrahmen und Doppelverglasung, pro Außenwand gibt es ein Fenster.
- **Haustür:** Jede Wohnung besitzt eine Haustür.

Ersatzsimulationselemente

- **Fenster:** Eine Liste von verschiedenen Fenstern mit unterschiedlicher Verglasung (s. [BK001]).

Parameter

- **Fenster**
 - **Höhe und Breite** (s. [BK001])
- **Wand**
 - **Höhe und Breite** (s. [BK001])
- **Haustür**
 - **Höhe und Breite** (s. [BK001])

Ereignisse⁷⁵

- **Zeitsprung:** Die Zeit wird ein halbes Jahr vorgespult. Dadurch herrschen aufgrund der wechselnden Jahreszeiten andere Umgebungsbedingungen.

Spieler-Aktionen

- **Fensterdurchbruch anpassen:** Der Fensterdurchbruch wird angepasst auf die Größe eines auszuwählenden Fensters.
- **Fenster einbauen:** Die ausgewählten Fenster werden eingebaut. Voraussetzung ist, dass sie in den Fensterdurchbruch passen.
- **Zeitsprung:** Der Spieler kann einen Zeitsprung von einem Jahr auslösen, er ist dadurch in der Lage, Verbrauchsgrößen über das Jahr zu ermitteln (z.B. wichtig für die Berechnung des Energieverbrauchs).

c) Größenverzeichnis

Name	Symbol	Einheit	Definition
Oberfläche Wand	A_W	m^2	Oberfläche einer Wand ohne Abzug von Durchbrüchen wie Fenster und Türen.
Oberfläche Wand netto	$A_{W,net}$	m^2	Oberfläche einer Wand mit Abzug von Durchbrüchen wie Fenster und Türen.
Oberfläche Fenster	A_F	m^2	Oberfläche eines Fensters
Raumvolumen	V_R	m^3	Das Volumen einer Wohnung im Vierwohnungsbungalow.

Tabelle 20: Größenverzeichnis Basiskonfiguration Vierwohnungsbungalow [BK002]

d) Missionen

I Mission „Wechsle die Fenster aus!“

Ziel

Der Spieler hat sämtliche Fenster des Gebäudes ausgewechselt.

Beschreibung

Diese Mission fordert den Spieler, sich mit den Grundmechanismen des Spiels vertraut zu machen. Neben der Aktion „Fenster einbauen“ muss er zuvor auch die Aktion „Fensterdurchbruch anpassen“ ausführen.

Belohnung

Fähigkeitspunkte

- [FS001] Durchführen von baulichen Veränderungen

e) Tags

Fenster, Zwischenwand, Wand

⁷⁵ Viele Elemente dieser Basiskonfiguration (Spieleraktionen, Systemereignisse) stimmen mit der Basiskonfiguration Einraumwohnung [BK001] überein. Dieses deutet daraufhin, dass in einem späteren Schritt die Struktur der Basiskonfigurationen nochmals überarbeitet werden sollte. Es bietet sich an, Ereignisgruppen und Gruppen von Spieleraktionen einzuführen, die dann einer Basiskonfiguration zugeordnet werden. Im jetzigen Stand des Entwurfs wird jedoch die Wiederholung in Kauf genommen - sie findet nur an einer Stelle statt.

5.5 Szenarien

5.5.1 Basisszenario: Heizwärmebedarf in Abhängigkeit von der Fenstergröße und -qualität [B001]

a) Lernziele

[F001] Einschätzen des Einflusses von Fenstergröße und Fensterart auf (Transmissions-) Wärmeverluste

[F008] Kennen des bauteilspezifischen Kennwertes U-Wert

[F009] Berechnen des U-Wertes für ein zusammengesetztes Bauteil

b) Basisszenario

[B000] Basiskonfiguration: Einraumhaus [BK001] (s. S. 69)

c) Titelgeschichte

Der Mieter Herr Emm klagt über exorbitant hohe Heizkosten. Er droht mit Auszug. Die Fenster sind sanierungsbedürftig und müssen ausgetauscht werden. Du hast von einem Freund einen Container mit vermessenen Fenstern geschenkt bekommen. Die Fenster haben unterschiedliche Glasqualitäten und unterschiedliche Größen. Eine eventuell notwendige Anpassung der Wanddurchbrüche für die Fenster verursacht beim ersten Einbau keine zusätzlichen Kosten, da ein anderer Freund, der als Maurer arbeitet, dir noch einen Gefallen schuldet.

d) Szenarioaufbau

Simulationselemente

- **Haustür:** Die Haustür hat einen guten U-Wert (Sie ist im Rahmen der Wärmebilanz kein Problem).
- **Heizungsanlage**

Ersatzsimulationselemente

- **Fenster:** Eine Liste von verschiedenen Fenstern mit unterschiedlicher Verglasung, verschiedenen Rahmenarten und unterschiedlichen Maßen. Nach dem Einbau eines Fensters verschwindet dieses aus der Liste.

Parameter

- **Fenster**
 - **U-Wert**⁷⁶
- **Wand**
 - **U-Wert**

Ereignisse

- **Neue Fenster:** Auf dem Markt sind neue Fenster verfügbar. Zum Teil besitzen sie bessere U-Werte, die dem Spieler zusätzliche Möglichkeiten eröffnen.

⁷⁶ Bei der Betrachtung der möglichen Wärmeverluste wird die Fugendurchlässigkeit der Fenster zunächst außer Acht gelassen. Dieses relevante Phänomen wird in einer Szenariovariante eingeführt.

Spieler-Aktionen

- **Fensterdurchbruch anpassen:** Der Fensterdurchbruch wird angepasst auf die Größe eines auszuwählenden Fensters.
- **Fenster einbauen:** Die ausgewählten Fenster werden eingebaut. Voraussetzung ist, dass sie in den Fensterdurchbruch passen.
- **Zeitsprung:** Der Spieler kann einen Zeitsprung von einem Jahr auslösen, er ist dadurch in der Lage, den Energieverbrauch über das Jahr durch eine Simulation zu ermitteln.

e) Wirkungsgeflecht

I Systemgrafik

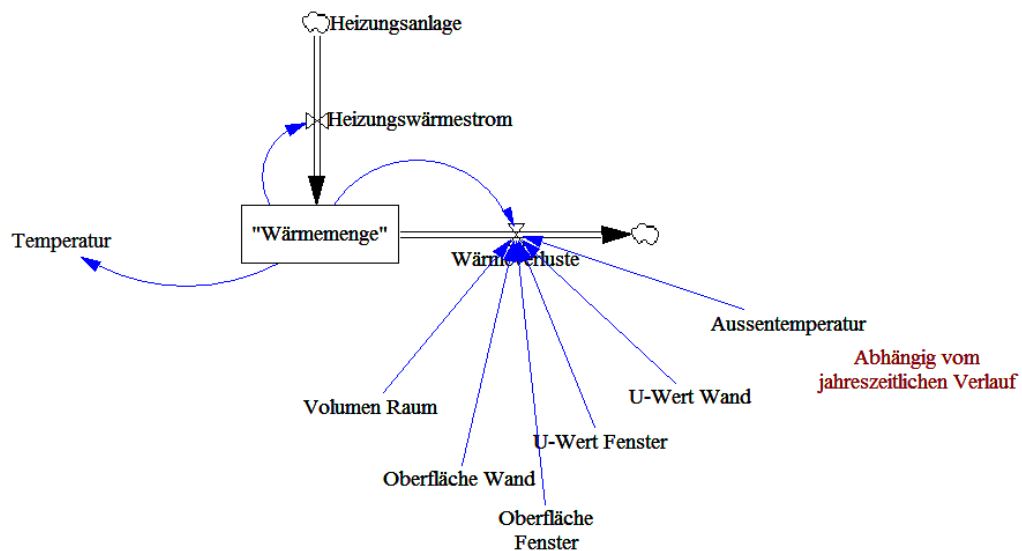


Abbildung 14: Wirkungsgeflecht Basisszenario Wärme [B001]

Die Größe „Wärmemenge“ ist eine Hilfsgröße, die zur Nutzung eines vereinfachten Berechnungsalgorithmus eingeführt wurde.

II Abhängigkeitsmatrix

Typ	beeinflusst	Außentemperatur	Heizungswärmestrom	Maximalleistung Heizkörper	Oberfläche Fenster	Oberfläche Wand	Volumen Raum	Temperatur	U-Wert Fenster	U-Wert Wand	Wärmemenge	Wärmeverluste	Wunschtemperatur
	Wird beeinflusst von												
E	Außentemperatur												
	Heizungswärmestrom							-					
E	Maximalleistung Heizkörper												

K	Oberfläche Fenster												
K	Oberfläche Wand												
K	Volumen Raum												
	Temperatur					-					+		
K⁷⁷	U-Wert Fenster												
K	U-Wert Wand												
	Wärmemenge		+									-	
	Wärmeverluste	-			+	+		+	+	+	+		
E	Wunschtemperatur												

Tabelle 21: Abhängigkeitsmatrix Basisszenario Wärme [B001]

III Größenverzeichnis

Name	Symbol	Einheit	Definition
Außentemperatur	T_A	°C	Lufttemperatur außerhalb des Hauses, vorgegeben durch Klimadaten
Heizungswärmestrom	Φ_H	W	Wärmestrom, der durch die Heizkörper in das Einraumhaus abgegeben wird.
Maximalleistung Heizkörper	$\Phi_{H,max}$	W	Die bauartbedingte maximale Wärmeabgabe des Heizkörpers.
Oberfläche Wand	A_W	m ²	Oberfläche einer Wand ohne Abzug von Durchbrüchen für Fenster und Türen.
Oberfläche Wand netto	$A_{W,net}$	m ²	Oberfläche einer Wand mit Abzug von Durchbrüchen für Fenster und Türen.
Oberfläche Fenster	A_F	m ²	Oberfläche eines Fensters
Raumvolumen	V_R	m ³	Das Volumen des Einraumhauses.
Schrittweite Heizungsregelung	d_H	W	Eine Charakteristik des Heizungswärmestroms ist, dass er nur zeitverzögert dem tatsächlichen Bedarf angepasst wird. Um dieses Verhalten nachzubilden, ändert sich der Heizungsstrom maximal um einen definierten Betrag. In diesem Beispiel wird ein Bruchteil des maximal möglichen Wärmestroms der Heizung vorgeschlagen ($d_H = 0,1 * \Phi_{H,max}$).
Temperatur	T_I	°C	Raumtemperatur
U-Wert Fenster	$U_{Fenster}$	W/m ² K	U-Wert eines Fensters
U-Wert Wand	U_{Wand}	W/m ² K	U-Wert einer Wand
Wärmemenge	Q_R	Ws	Hilfsgröße zur Nutzung eines vereinfachten Berechnungsalgorithmus: Wärmemenge innerhalb der Gebäudehülle.
Wärmeverluste	Φ_V	W	Wärmeverluststrom, der sich ergibt aus Transmission durch Wände und Fenster.
Wunschtemperatur	T_W	°C	Die Wunschtemperatur der Bewohner.

Tabelle 22: Größenverzeichnis Basisszenario Wärme [B001]

⁷⁷ Zur Vereinfachung wird ein konstanter U-Wert angenommen. Diese Annahme erweist sich als ausreichend exakt, um die vorgesehenen Effekte mit Hilfe der Simulation zu demonstrieren. Es ist (bisher) auch die herrschende Vorgehensweise in bauphysikalischen Fragestellungen, obwohl neuere Forschungsergebnisse zeigen, dass der U-Wert unter instationären Verhältnissen eine variable Größe ist (Gürlebeck & Rudolph, 2009).

IV Berechnungsvorschriften

IV.I Heizungswärme­strom Φ_H

```

IF  $T_I < T_W$  THEN
    // Raumtemperatur ist geringer als Wunschtemperatur
     $\Phi_H = \Phi_{H,n-1} + d_H$ 
    IF  $\Phi_H > \Phi_{H,max}$  THEN
        // Begrenzung auf maximale Leistung
         $\Phi_H = \Phi_{H,max}$ 
    ENDIF
ELSE
    IF  $\Phi_{H,n-1} > 0$  THEN
        // Raumtemperatur zu warm und Heizung läuft ->
        // Heizwärme­strom kann reduziert werden.
         $\Phi_H = \Phi_{H,n-1} - d_H$ 
        IF  $\Phi_H < 0$  THEN
            // Die Heizung nimmt keine Wärme auf.
             $\Phi_H = 0$ 
        ENDIF
    ENDIF
ENDIF

```

Pseudocode 1: Heizungswärme­strom

IV.II Temperatur T_I

Die Berechnung der Temperatur benutzt die folgenden vereinfachenden Annahmen:

- Die Temperatur wird überschlagmäßig errechnet durch die Betrachtung der Wärmeströme: Wärmegewinne entstehen durch die Heizung, Wärmeverluste durch Transmission durch die Raumbegrenzungen. Die Differenz der beiden Größen ist ein Wärme­strom, der für das Berechnungsintervall als konstant angenommen wird. Das Produkt mit der Dauer des Berechnungsintervalls ergibt die Differenzwärmemenge. Diese wird umgelegt auf das Volumen des Raumes in eine Temperaturdifferenz.
- Es wird vereinfachender Weise angenommen, dass 1 m^3 Raumluft unter idealisierten Bedingungen $0,35 \text{ Wh/K}$ Wärme speichert.

```

// Berechnung der zugeführten Wärme (Einheit: Wh)
 $Q_{diff} = (\Phi_H - \Phi_V) * (t_n - t_{n-1})$ 
// Umrechnung der Wärmemenge auf das Raumvolumen
 $T_{diff} = \frac{Q_{diff}}{V_R * 0,35}$ 
// Berechnung der neuen Temperatur
 $T_I = T_{I,n-1} + T_{diff}$ 

```

Pseudocode 2: Temperatur im Innenraum

IV.III Wärmemenge Q_R

Die Hilfsgröße „Wärmemenge“ wird nicht separat berechnet, sondern geht in die Berechnung der Raumtemperatur ein (s. Kap. IV.II). Sie wird hier aufgeführt, da die Größe in der Systemgrafik und in der Abhängigkeitsmatrix enthalten ist.

IV.IV Wärmeverluste Φ_V

Die Wärmeverluste werden berechnet über Formel 1, in den Summen werden jeweils alle vier Wände berücksichtigt.

$$T_I = (T_I - T_A) * U_{ges} * \sum A_W \text{ mit } U_{ges} = \frac{\sum U_F A_F + U_W A_{W,net}}{\sum A_W}$$

Formel 1: Wärmeverluste

f) Missionen

I Mission „Wechsle die Fenster aus!“

Ziel

Der Spieler hat sämtliche Fenster des Gebäudes ausgewechselt.

Beschreibung

Diese Mission fordert den Spieler, sich mit den Grundmechanismen des Spiels vertraut zu machen. Neben der Aktion „Fenster einbauen“ muss er zuvor auch die Aktion „Fensterdurchbruch anpassen“ ausführen.

Belohnung

Fähigkeitspunkte

- **[FS001]** Durchführen von baulichen Veränderungen

II Mission „Finde das beste Fenster aus energetischer Sicht!“

Ziel

Der Spieler hat sämtliche Fenster des Gebäudes erneuert, und zwar mit den Fenstern, die insgesamt eine Minimierung des Energieverbrauches - für ein Jahr gesehen - ergeben.

Beschreibung

Mit Hilfe dieser Mission wird der Spieler veranlasst, sich mit dem grundsätzlichen Wirkungssystem auseinanderzusetzen. Er muss die Parameter identifizieren, die für einen minimalen Energieverbrauch verantwortlich sind und sie bewerten. Damit der Spieler sich nicht ausschließlich auf das Ausprobieren beschränkt, werden einem Fensteraustausch Kosten zugeordnet (abgesehen von dem ersten, freien Versuch).

Belohnung

Fähigkeitspunkte

- **[F001]** Einschätzen des Einflusses von Fenstergröße und Fensterart auf (Transmissions-) Wärmeverluste
- **[F008]** Kennen des bauteilspezifischen Kennwertes U-Wert
- **[F009]** Berechnen des U-Wertes für ein zusammengesetztes Bauteil

g) Tags

Fenster, Transmissionswärmeverluste, U-Wert, Wärmeschutz, Wärmeverluste

h) Vorausgesetzte Fähigkeiten

(keine)

i) Szenariovarianten

Das im Szenario benutzte Modell zeigt nur einen Aspekt der Bauphysik und ist sehr vereinfacht. Zu den bisher noch nicht berücksichtigten Effekten, die in das Szenario eingebaut werden können, gehören:

Wärmespeicherfähigkeit der Wände: Dadurch, dass die Wände ebenfalls eine Wärmespeicherefunktion besitzen, ist die Wirkung von Wechseln der Umgebungstemperatur nicht so unmittelbar, sie wird durch die in den Wänden gespeicherte Wärme gepuffert. In einer weiteren Variante

kann auch der Einfluss von Phase Change Materials (PCM) thematisiert werden (Völker, Kornadt, & Ostry, 2008).

Lüftungswärmeverluste: Lüftungswärmeverluste sind eine Größe, die bei der Dimensionierung der Heizungsanlage nicht vernachlässigt werden können (Lutz et al., 2002, S. 151). Dieses Szenario bietet sich an, den Begriff einzuführen.

Fugendurchlässigkeit: Ein nicht unerheblicher Teil von Wärmeverlusten resultiert aus der sogenannten Fugendurchlässigkeit der Fenster. Sie wird verursacht durch den vorhandenen Zwischenraum zwischen Flügelrahmen und Blendrahmen und wird ausgedrückt durch den A-Wert (bzw. Q-Wert). Durch eine Einbeziehung der Fugendurchlässigkeit steigen die Wärmeverluste. Darauf kann der Spieler durch die Auswahl von Fenstern einer höheren Klasse der Fugendurchlässigkeit nach DIN EN 12207 reagieren.

5.5.2 Ergänzungsszenario: Entwicklung der Transmissionswärmeverluste in Abhängigkeit von der Dämmungsqualität [E001]

a) Lernziele

- **[F002]** Unterscheiden verschiedener Arten von Dämmstoffen
- **[F003]** Kalkulieren von Kosten einer Wärmedämmung
- **[F004]** Kalkulieren von Wärmeverlusten
- **[F005]** Kalkulieren von Heizwärmekosten im Jahresverlauf.
- **[F006]** Anwenden der Vorschriften der EnEV
- **[F007]** Bewerten der Wirtschaftlichkeit einer Wärmedämmung
- **[F016]** Unterscheiden verschiedener Arten von Baumaterialien
- **[F017]** Bewerten des Baumaterials anhand des Attributs „Wärmeleitfähigkeit“

b) Basisszenario

[B001] Basisszenario: Heizwärmebedarf in Abhängigkeit von der Fenstergröße und -qualität (S. 76)

c) Titelgeschichte

Mieter Emm hat über hohe Heizkostenrechnungen geklagt. Der Vermieter bekommt im Rahmen eines energetischen Wohnungssanierungsprogramms der Stadt und der damit verbundenen Zuschüsse die relativ preisgünstige Gelegenheit, die Wohnung besser zu dämmen. Die Dämmung soll von außen an die Wände angebracht werden. Der Vermieter möchte die Gelegenheit nutzen, er kann nun über die Art des Dämmmaterials und die aufzutragende Dicke entscheiden. Natürlich hat er dabei seine Kosten im Blick.

d) Szenarioaufbau

Ersatzsimulationselemente

- **Dämmmaterialien**
In diesem Szenario kann der Spieler aus einer Liste (s. Tabelle 23) von Dämmstoffen einen auswählen, mit dem das Haus gedämmt wird.

Parameter

- **Wand**
 - Dicke

- Wärmeleitfähigkeit
- Wärmeübergangswiderstand innen und außen (R_{si} und R_{se})
- **Dämmmaterial**
 - Dicke
 - Spezifische Kosten
 - Wärmeleitfähigkeit

Spieler-Aktionen

- **Dämmung anbringen** Der Spieler kann eine Wand neu dämmen. Dazu muss er das Dämmmaterial auswählen sowie die Dicke, in der dieses aufgetragen werden soll, angeben.
- **Wand neu errichten** Der Spieler kann eine Wand neu errichten. Auch hier wählt er das Material aus und gibt die Dicke der Wand an. Die Neuerrichtung einer Wand entfernt die alte zusammen mit einer eventuell angebrachten Dämmung.

Listen

- **Dämmmaterialien**

Dämmmaterial	Wärmeleitfähigkeit (λ) in W/mK	Spezifische Kosten (k_D) in €/m ³
EPS (Styropor)	0,04	50
Mineralwolle	0,04	55
Holzfaserdämmplatte	0,045	185
XPS Hartschaum	0,04	134
PUR Hartschaum	0,025	165
Hanf/Flachs	0,04	148
Schaumglasschotter	0,09	99

Tabelle 23: Liste der auswählbaren Dämmmaterialien

Tabelle 23 zeigt exemplarisch die Liste der in diesem Szenario auswählbaren Dämmstoffe. Die Dämmstoffe und zugehörigen Werte für die Leitfähigkeit wurden von Plag (2010) übernommen. Die spezifischen Kosten wurden ebenfalls aus den Angaben dieser Quelle entwickelt. Sie unterliegen in der Realität Schwankungen, müssen jedoch zur Ausgestaltung eines Simulationsspielszenarios konkretisiert sein.

Baumaterial	Wärmeleitfähigkeit λ in W(m ² *K)
Beton	1,85 ⁷⁸
Mauerziegel	0,58
Leichtlochziegel	0,33
Kalksandstein	0,99
Porenbeton	0,14
Leichtbetonsteine	0,49

Tabelle 24: Wärmeleitfähigkeit verschiedener Baumaterialien

Tabelle 24 zeigt die Wärmeleitfähigkeiten einer Liste von Baustoffen, aus denen die Wände aufgebaut werden können (Schneider, 2001: Tafel 10.16).

⁷⁸ Der Wert wurde bestimmt durch lineare Interpolation gemäß den dort angegebenen Werten für die Dichte.

e) Wirkungsgeflecht

I Systemgrafik

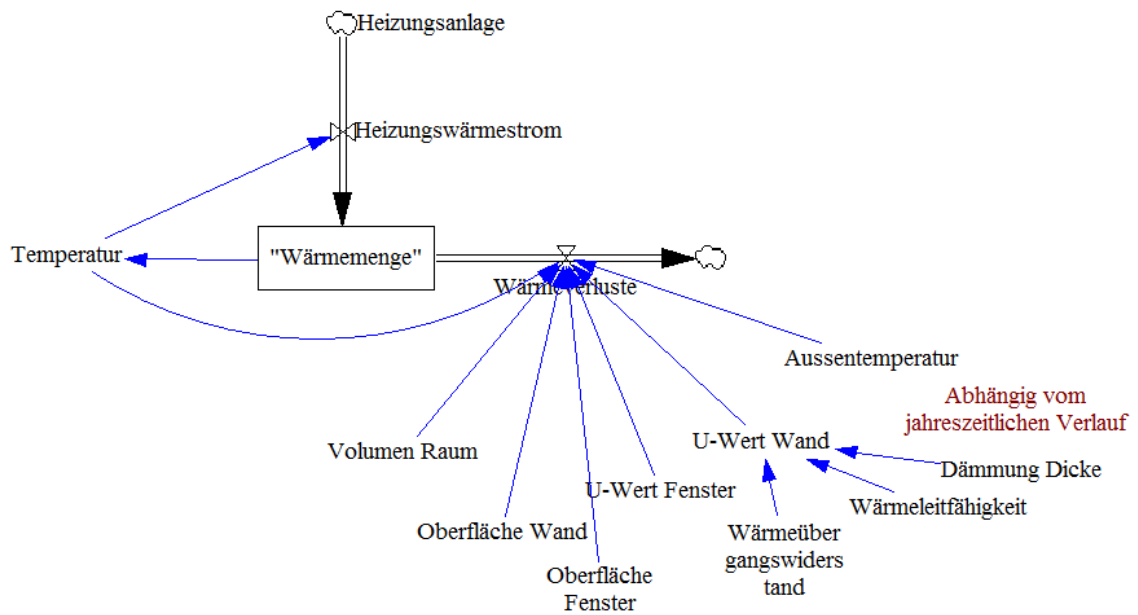


Abbildung 15: Wirkungsgeflecht Erweiterungsszenario Wärmedämmung [E001]

II Abhängigkeitsmatrix

beeinflusst								
Typ	Wird beeinflusst von	Schichtdicke der Dämmung	Schichtdicke der Wand	U-Wert Wand	Wärmeleitfähigkeit Dämmmaterial	Wärmeleitfähigkeit Wand	Wärmeübergangswiderstand (außen)	Wärmeübergangswiderstand (innen)
K	Schichtdicke der Dämmung							
K	Schichtdicke der Wand							
	U-Wert Wand	-	-		+	+	-	-
K	Wärmeleitfähigkeit Dämmmaterial							
K	Wärmeleitfähigkeit Wand							
K	Wärmeübergangswiderstand (außen)							
K	Wärmeübergangswiderstand (innen)							

Tabelle 25: Abhängigkeitsmatrix Ergänzungsszenario Transmissionswärmeverluste [E001]

III Größenverzeichnis

Neben den in Tabelle 22 genannten Größen werden in diesem Ergänzungsszenario die in der folgenden Tabelle 26 genannten Größen verwendet.

Name	Symbol	Einheit	Definition
Schichtdicke der Dämmung	d_D	m	Dicke, in der die Dämmung aufgetragen wurde.
Schichtdicke der Wand	d_W	M	Schichtdicke der ursprünglichen Wand ohne Dämmung.
Spezifische Kosten der Dämmung	k_D	€/m ³	Die Dämmmaterialien werden flächig in einer bestimmten Dicke aufgetragen, der Verbrauch lässt sich mit einem Raummaß messen ⁷⁹ .
U-Wert Wand	U_W	W/m ² K	U-Wert einer Wand, dieser wird berechnet mit Hilfe der Wärmeleitfähigkeit der verwendeten Dämmung.
Wärmeleitfähigkeit Dämmmaterial	λ_D	W/mK	Wärmeleitfähigkeit des verwendeten Dämmmaterials.
Wärmeleitfähigkeit Wand	λ_W	W/mK	Wärmeleitfähigkeit des Wandmaterials
Wärmeübergangswiderstand (außen)	R_{se}	m ² K/W	Wert gemäß Schneider, 2001 (Tafel 10.26): 0,04 m ² K/W
Wärmeübergangswiderstand (innen)	R_{si}	m ² K/W	Wert gemäß Schneider, 2001 (Tafel 10.26): 0,13 m ² K/W
Spezifische Heizkosten	k_H	€/kWh	Kosten pro tatsächlich genutzter kWh Heizenergie.

Tabelle 26: Größenverzeichnis Ergänzungsszenario Transmissionswärmeverluste [E001]

IV Berechnungsvorschriften

IV.I U-Wert Wand U_W

Der U-Wert der Wand wird berechnet gemäß der Formel

$$U = \frac{1}{R_{si} + R + R_{se}} \text{ mit } R = \frac{d_D}{\lambda_D} + \frac{d_W}{\lambda_W}$$

Formel 2: U-Wert mit Hilfe von Wärmedurchlasswiderständen (S. Bläsi, 2002, S. 13)

f) Missionen

I Mission „Statte das Haus mit einer Wärmedämmung aus!“

Ziel

Der Spieler hat das Haus mit einer Wärmedämmung ausgestattet.

Beschreibung

Diese Mission fordert den Spieler, sich mit den Grundmechanismen dieses Szenarios vertraut zu machen. Er muss eine Auswahlentscheidung treffen.

Belohnung

Fähigkeitspunkte:

- **[F002]** Unterscheiden verschiedener Arten von Dämmstoffen
- **[FS001]** Durchführen von baulichen Veränderungen

⁷⁹ Die Einheit €/m³ wurde gewählt, um die Preise unterschiedlicher Dämmmaterialien vergleichen zu können. Ein anderes, in der Praxis gebräuchliches Maß ist Preis pro Quadratmeter bei einer vorgegebenen Dicke von z.B. 10 cm (Plag, 2010)

II Mission „Wärmedämmung mit begrenztem Budget“

Ziel

Der Spieler hat das Haus mit einer Wärmedämmung ausgestattet, deren Kosten nicht höher als eine vorgegebene Grenze ist.

Beschreibung

Diese Mission fordert den Spieler, sich mit den Kosten einer Wärmedämmung vertraut zu machen. Er sollte die Kosten einer Wärmedämmmaßnahme berechnen können.

Belohnung

Fähigkeitspunkte:

- **[F003]** Kalkulieren von Kosten einer Wärmedämmung
- **[F007]** Bewerten der Wirtschaftlichkeit einer Wärmedämmung

III Mission „Wärmedämmung mit begrenzter Dicke der Dämmung“

Ziel

Der Spieler hat das Haus mit einer Wärmedämmung ausgestattet, deren Schichtdicke eine vorgegebene Größe nicht überschreitet, aber auf die Dauer von 10 Jahren die geringsten Heizkosten inklusive der Kosten für die Dämmmaßnahme verursacht.

Beschreibung

Diese Mission fordert den Spieler, sich mit den Kosten einer Wärmedämmung vertraut zu machen. Auch soll er die Auswirkungen des Anbringens einer Dämmung in energetischer Sicht bewerten können. Zusätzlich soll er die Einsparung von Kosten für einen gegebenen Zeitraum unter Berücksichtigung eines ebenfalls gegebenen prognostizierten Kostenverlaufs von Heizwärme für diesen Zeitraum berechnen.

Belohnung

Fähigkeitspunkte:

- **[F003]** Kalkulieren von Kosten einer Wärmedämmung
- **[F004]** Kalkulieren von Wärmeverlusten
- **[F005]** Kalkulieren von Heizwärmekosten im Jahresverlauf.

IV Mission „EnEV 2009: Transmissionswärmeverluste der Außenhülle“

Ziel

Der Spieler hat das Haus mit einer Wärmedämmung ausgestattet, die die Transmissionswärmeverluste der Außenhülle auf ein von der EnEV gefordertes Maß reduziert. Gleichzeitig hat er Grenzen bezüglich der maximalen Schichtdicke der Dämmung sowie der Kosten der Dämmmaßnahme nicht überschritten⁸⁰.

Beschreibung

Diese Mission fordert den Spieler, sich mit den Kosten einer Wärmedämmung vertraut zu machen. Auch soll er die Auswirkungen des Anbringens einer Dämmung in energetischer Sicht bewerten können und diese mit den in der EnEV geforderten Werten vergleichen können.

⁸⁰ In dieser Mission sind nur Teilforderungen der EnEV zu erfüllen, es ist kein kompletter Nachweis in Form eines Energieausweises zu erstellen.

Belohnung

Fähigkeitspunkte:

- **[F003]** Kalkulieren von Kosten einer Wärmedämmung
- **[F004]** Kalkulieren von Wärmeverlusten
- **[F006]** Anwenden der Vorschriften der EnEV

V Mission „Baumaterial als Wärmedämmung“

Ziel

Der Spieler hat das Haus derart mit Außenwänden ausgestattet, dass die resultierenden U-Werte die Anforderungen der EnEV 2009 erfüllen.

Beschreibung

Diese Mission fordert vom Spieler, ein geeignetes Baumaterial zu suchen und die Wand passend zu dimensionieren, um den erforderlichen U-Wert zu erreichen. Der Spieler kann erkennen, mit welcher Dicke eine Wand dimensioniert werden muss - und kann dieses in Bezug setzen zu der notwendigen, äquivalenten Schichtdicke an Dämmmaterial. Der Umgang mit den Vorschriften der EnEV 2009 wird trainiert.

Belohnung

Fähigkeitspunkte:

- **[F004]** Kalkulieren von Wärmeverlusten
- **[F016]** Unterscheiden verschiedener Arten von Baumaterialien
- **[F017]** Bewerten des Baumaterials anhand des Attributs „Wärmeleitfähigkeit“

g) Tags

EnEV 2009, Kosten Wärmedämmung, Wärmedämmung, Wärmeleitfähigkeit, Wärmeverluste, Wärmeübergangswiderstand

h) Vorausgesetzte Fähigkeiten

[FS001] Durchführen von baulichen Veränderungen.

[F001] Einschätzen des Einflusses von Fenstergröße und Fensterart auf (Transmissions-) Wärmeverluste

i) Szenariovarianten

Wesentlich erhöhte Heizkosten: Unter Nutzung von Kostenverläufen können wesentlich erhöhte Heizkosten simuliert werden und deren Auswirkung auf die Rentabilität von Dämmmaßnahmen gezeigt werden.

Lüftungswärmeverluste: Mit Hilfe einer weiteren Mission kann der Begriff der Lüftungswärmeverluste eingeführt werden (s.a. Kap. 5.5.1i)). Dieses Thema kann in zusätzlichen Ergänzungsszenarien vertieft werden.

Auftreten von Schimmel bei Innendämmung: Bei Innendämmung kann es wegen kondensierender Feuchtigkeit zu Schimmelbildung kommen. Dieses Phänomen gehört eigentlich zum Feuchteschutz, kann hier jedoch einführend thematisiert werden.

Entsorgung der Baumaterialien: Recycling von Bauschutt schont Deponiekapazitäten. Um den Spieler für diese Problematik zu sensibilisieren, können Entsorgungskosten bzw. Verkaufserlöse für den durch Umbau anfallenden Bauschutt in das Szenario mit einbezogen werden.

5.5.3 Ergänzungsszenario: Solare Wärmegewinne [E002]

a) Lernziele

- **[F010]** Nennen der maßgeblichen Parameter von solaren Wärmegewinnen
- **[F011]** Berechnen von solaren Wärmegewinnen
- **[F012]** Recherchieren in DINs
- **[F013]** Anwenden der DIN 4108-2
- **[F014]** Berechnen des maximal zulässigen Sonneneintragswertes
- **[F015]** Anwenden des g-Wertes

b) Basisszenario

[B001] Basisszenario: Heizwärmebedarf in Abhängigkeit von der Fenstergröße und -qualität (S. 76)

c) Titelgeschichte

Mieter Emm hat immer noch Probleme mit erhöhten Heizkosten. In der Zwischenzeit hat er mitbekommen, dass es neuartige Wärmeschutzverglasungen auf dem Markt gibt. Die Glasvertriebsfirma hat eine Marketing-Aktion zusammen mit einem Autokranhersteller gestartet: Die ersten 50 Kunden können gegen einen Sonderpreis eine Hausdrehschere mit Hilfe einer brandneuen Krankoordinationssteuerung für 4 Autokräne erwerben. Damit ließe sich dann ein komplettes Haus neu ausrichten. Das kommt dem Mieter Emm doch gerade recht, da er etwas von solaren Wärmegewinnen gehört hat.

d) Szenarioaufbau

Parameter

- **Haus**
 - Geographische Koordinaten
- **Fenster**
 - **g-Wert** (Die Glasscheiben eines jeden Fensters haben unterschiedlich g-Werte)
 - **Ausrichtung**
 - **Breite und Höhe** (die Größe der Fenster variiert von Wand zu Wand, um unterschiedliche solare Wärmeeinstrahlungen zu ermöglichen)

Ereignisse

- **Umgebungssprung:** Der Ort des Haus wird plötzlich verändert. Auch hier ergeben sich andere Umgebungsbedingungen. Insbesondere ist für dieses Szenario die Umgebungstemperatur wichtig, die von der Klimazone abhängt.

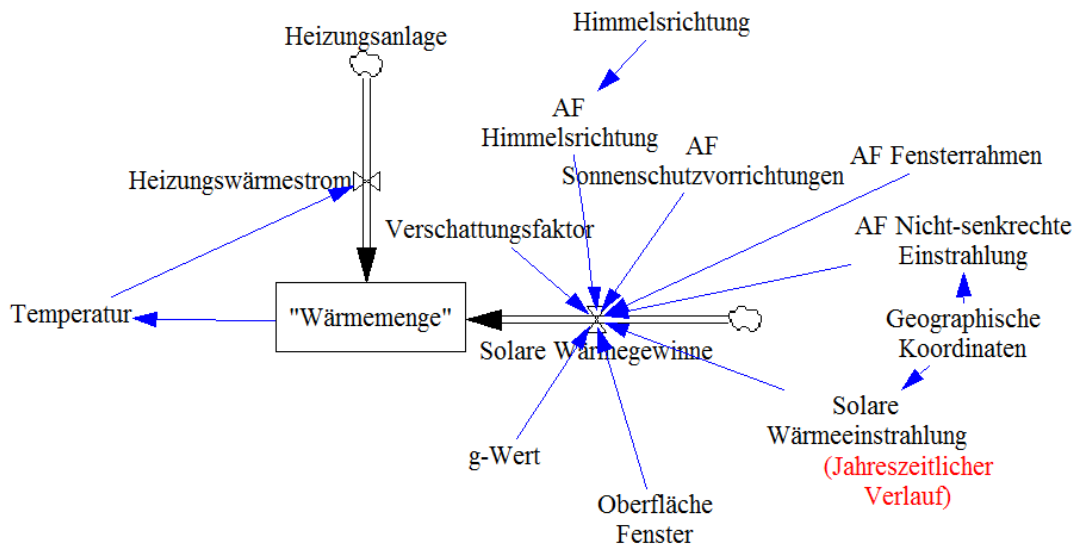
Spieler-Aktionen

- **Ausrichtungsänderung** Mit Hilfe der schon beschriebenen Kransteuerung kann der Spieler die Richtung des Hauses verändern und somit die solaren Wärmegewinne beeinflussen.

- **Anbringen von Sonnenschutzvorrichtungen** Nachdem der Spieler die Ausrichtung des Hauses auf maximale solare Wärmeeinstrahlung im Winter optimiert hat, kann es passieren, dass im Sommer der max. Sonneneintragskennwert überschritten wird. In diesem Fall ist die Montage von Sonnenschutzvorrichtungen notwendig.

e) Wirkungsgeflecht

I Systemgrafik



81

Abbildung 16: Wirkungsgeflecht Ergänzungsszenario Solare Wärmegewinne [E002]

II Größenverzeichnis

Name	Symbol	Einheit	Definition
AF Sonnenschutzvorrichtungen	F_C	-	In diesem Szenario gibt es noch keine Sonnenschutzvorrichtungen, daher wird der Faktor als 1,0 angenommen.
AF Himmelsrichtung	F_D	-	Die solare Wärmeeinstrahlung ist abhängig von der Himmelsrichtung. Mit Hilfe dieses Abminderungsfaktors wird die Himmelsrichtung, in die das Fenster weist, modelliert.
AF infolge Fensterrahmen	F_F	-	Dieser Faktor wird als 0,7 angenommen.
AF infolge nicht senkrechter Einstrahlung	F_W	-	Abminderungen in Abhängigkeit vom Winkel der Sonneneinstrahlung sind abhängig vom geographischen Standort des Gebäudes.
Geographische Koordinaten	λ	-	Der Standort des Gebäudes beeinflusst zum einen die solare Wärmeeinstrahlung, zum anderen den Abminderungsfaktor infolge nicht senkrechter Einstrahlung.
g-Wert	g	-	Gesamtenergiedurchlassgrad der Verglasung
Himmelsrichtung	D	-	Die Himmelsrichtung dient zur Kennzeichnung der Ausrichtung des Gebäudes.

⁸¹ AF ist die Abkürzung für Abminderungsfaktor.

Solare Wärmeeinstrahlung	I_s	W/m^2	Die solare Wärmeeinstrahlung ist wesentlich abhängig von den Wetterbedingungen und dem Tag-Nacht -sowie dem jahreszeitlichen Rhythmus. Zusätzlich geht die geographische Lage hier ein.
Solare Wärmegewinne	Φ_S	W	Solarer Wärmegewinnstrom
Verschattungsfaktor	F_S	-	Bauliche Besonderheiten wie Überbauten oder Blenden werden mit Hilfe des Verschattungsfaktors abgebildet.

Tabelle 27: Größenverzeichnis Ergänzungsszenario Solare Wärmegewinne [E002]

III Abhängigkeitsmatrix

Typ	Wird beeinflusst von	beeinflusst									
		AF Fensterrahmen	AF Himmelsrichtung	AF Nicht-senkrechte Einstrahlung	AF Sonnenschutzvorrichtungen	Geographische Koordinaten	g-Wert	Himmelsrichtung	Solare Wärmeeinstrahlung	Solare Wärmegewinne	Verschattungsfaktor
K	AF Fensterrahmen										
	AF Himmelsrichtung							0			
K	AF Nicht-senkrechte Einstrahlung					0					
K	AF Sonnenschutzvorrichtungen										
K	Geographische Koordinaten										
K	g-Wert										
K	Himmelsrichtung										
	Solare Wärmeeinstrahlung		+								
	Solare Wärmegewinne	+		+	+		+		+		+
K	Verschattungsfaktor										

Tabelle 28: Abhängigkeitsmatrix Ergänzungsszenario Solare Wärmegewinne [E002]

IV Berechnungsvorschriften

 IV.I Solare Wärmegewinne Φ_S

$$\Phi_S = F_F * F_C * F_S * F_W * I_s * g * A_s$$

Formel 3: Solare Wärmegewinne

In die Formel gehen sogenannte Abminderungsfaktoren ein, mit denen überschlagweise verschiedene Gegebenheiten berücksichtigt werden (Bläsi, 2002, S. 31). Zudem wird die solare Wärmeeinstrahlung I_s berücksichtigt.

IV.II Solare Wärmeeinstrahlung I_S

In der Regel wird die solare Wärmeeinstrahlung als Summenwert bezogen auf ein Jahr angegeben. Damit in diesem Szenario eine kontinuierliche Simulation stattfinden kann, wird die solare Wärmeeinstrahlung als zeitabhängige Funktion modelliert.

Als Basis für diese Funktion werden TRY-Daten genutzt, in denen stundenweise Werte für solare Wärmeeinstrahlung - allerdings auf eine waagerechte Fläche - enthalten sind (Bundesministerium für Verkehr, Bau- und Stadtentwicklung, 2011). Es wird die Funktion $B_{TRY}(t)$ definiert. Diese Funktion liefert aus den gegebenen TRY-Daten den der Zeit t entsprechenden Wert für die direkte Sonnenbestrahlungsstärke B zurück. Da die TRY-Daten für die Dauer eines Jahres vorliegen, wird der Zeitpunkt t_{TRY} , mit dem in den TRY-Daten gesucht wird, als Rest der Division von t und T_A (Zeitdauer eines Jahres) definiert - t_{TRY} ist der unterjährige Anteil von t ($t_{TRY} \in [0, T_A)$).

Die Ergebnisse der Funktion $B_{TRY}(t)$ werden auf die Himmelsrichtung der Fenster angepasst. Für die Anpassung wird gemäß Bläsi (2002:S. 31, Bild 1), eine 2π -periodische Funktion $f_{FD}(D)$ mit $D \in (-\pi, \pi]$ vorgeschlagen, die die Werte aus Tabelle 29 linear interpoliert⁸²:

Fensterausrichtung	Abminderungsfaktor
$-\pi/2$ (Ost)	0,574 (155/270)
0 (Süd)	1
$\pi/2$ (West)	0,574 (155/270)
π (Nord)	0,370 (100/270)

Tabelle 29: Abminderungsfaktor: Himmelsrichtung

Somit ergibt sich zur Berechnung der solaren Wärmestrahlung Formel 4.

$$I_S(D, t) = f_{FD}(D) * B_{TRY}(t)$$

Formel 4: Solare Wärmeeinstrahlung

f) Missionen

I Mission „Richte das Haus optimal aus!“

Ziel

Der Spieler hat das Haus derart gedreht, dass die solaren Wärmegewinne maximiert werden.

Beschreibung

Diese Mission fordert den Spieler, sich mit den Prinzipien und den bestimmenden Parametern (g-Wert, Fenstergröße, Ausrichtung) der solaren Wärmegewinne vertraut zu machen. Der Spieler führt die Aktion „Ausrichtungsänderung“ durch.

Belohnung

Fähigkeitspunkte

⁸² Hier kann sich eine Abweichung zwischen modelliertem System und Realität ergeben: In der Realität wird die Wärmeeinstrahlung über den Tag mit dem Lauf der Sonne variieren: Morgens werden die Fenster Richtung Osten hauptsächlich die Strahlung abbekommen, während am Nachmittag über die Fenster in Richtung Osten der größte Teil der Wärme eingetragen wird. In dem Modell sorgt der (zeitunabhängige) Abminderungsfaktor Himmelsrichtung F_D für eine gleichmäßige Verteilung der unterschiedlichen Wärmeeinträge über den Tagesverlauf.

- **[F010]** Nennen der maßgeblichen Parameter von solaren Wärmegewinnen
- **[F011]** Berechnen von solaren Wärmegewinnen.

II Mission „Richte das Haus optimal nach den Vorschriften der DIN 4108-2 aus!“

Ziel

Der Spieler hat das Haus derart gedreht, dass die solaren Wärmegewinne optimiert werden, dabei aber der maximal zulässige Sonneneintragswert nach DIN 4108-2 nicht überschritten wird.

Beschreibung

Die Ausrichtung des Hauses zur Maximierung der solaren Wärmegewinne führt dazu, dass es im Sommer im Haus für die Bewohner zu warm ist. In dieser Mission soll geprüft werden, ob die Bewohner durch eine weitere Drehung des Hauses auch im Sommer zufriedengestellt werden können. Maßstab der Zufriedenheit ist dabei der maximal zulässige Sonneneintragswert, der in der DIN 4108-2 definiert ist.

Belohnung

Fähigkeitspunkte

- **[F012]** Recherchieren in DINs
- **[F013]** Anwenden der DIN 4108-2
- **[F014]** Berechnen des maximal zulässigen Sonneneintragswertes

III Mission „Baue die optimalen Fenster ein!“

Ziel

Der Spieler hat für die aktuelle Ausrichtung des Hauses ein Fenster mit passender Größe und g-Wert eingebaut, das die solaren Wärmegewinne optimiert, dabei aber nicht den maximalen Sonneneintragswert überschreitet.

Beschreibung

Das Haus wird wieder auf die ursprüngliche Position gedreht - die Drehung mit der Kransteuerung gibt es natürlich nicht, das war ein Traum des Spielers. Dennoch hat der Spieler Möglichkeiten, die solaren Wärmegewinne zu optimieren: Er kann neue Fenster einbauen, dabei darf er sowohl die Größe des Fensters anpassen als auch Glasscheiben mit entsprechenden g-Wert benutzen. Der Spieler muss sowohl die Berechnung des maximal zulässigen Sonneneintragswertes beherrschen als auch durch Variation von g-Wert und Größe des Fensters ein optimales Fenster finden.

Belohnung

Fähigkeitspunkte

- **[F013]** Anwenden der DIN 4108-2
- **[F014]** Berechnen des maximal zulässigen Sonneneintragswertes
- **[F015]** Anwenden des g-Wertes

g) Tags

Abminderungsfaktor, DIN 4108-2, g-Wert, Maximal Zulässiger Sonneneintragswert, Recherche in DINs, Sonneneintragswert, Solare Wärmegewinne, Solare Wärmeeinstrahlung, Verschattungsfaktor

h) Vorausgesetzte Fähigkeiten

- **[FS001]** Durchführen von baulichen Veränderungen.
- **[F001]** Einschätzen des Einflusses von Fenstergröße und Fensterart auf (Transmissions-) Wärmeverluste

i) Szenariovarianten

Änderung des Standorts: Solare Wärmegewinne können abgesehen von der Hausausrichtung und Art der Verglasung durch weitere Faktoren beeinflusst werden. Einer davon ist die Sonnenscheindauer. Mit einer plötzlichen Änderung des Standortes können unterschiedliche Sonnenscheindauern in verschiedenen Klimazonen verarbeitet werden.

Anpassung des Rahmenanteils: Der Rahmenanteil hat ebenfalls Einfluss auf die solaren Wärmegewinne. Durch die Vorgabe eines wesentlich geänderten Rahmenanteils (begründet zum Beispiel durch eine gesetzlich Vorschrift oder eine produktionstechnische Notwendigkeit) kann es notwendig sein, andere Fenster einzubauen, um solare Wärmegewinne zu optimieren (Faktor F_r).

Verschattung: Durch das Anbringen eines Außenbalkons sowie den Neubau eines Hauses in unmittelbarer Nähe werden die solaren Wärmegewinne wesentlich gemindert. Dem Mieter wird die Miete um die nun zusätzlich anfallenden Heizkosten vermindert (Faktor F_s).

5.5.4 Basisszenario: Schallpegel in Abhängigkeit von der Fenstergröße und -qualität [B002]



a) Lernziele

- **[F021]** Kennen des Parameters „Bewertetes Bauschalldämmmaß“
- **[F022]** Berechnen des Gesamtschalldämmmaßes eines einfachen, zusammengesetzten Bauteils
- **[F023]** Kennen der gegen charakteristische Geräusche erforderlichen Schalldämmmaße

b) Basisszenario

[B000] Basiskonfiguration: Einraumhaus [BK001] (S. 69)

c) Titelgeschichte

Der Mieter Herr Emm klagt über unzumutbaren Lärm in der Wohnung von der Straßenseite her. Er hat eine Mietminderung vorgenommen und droht mit dem Auszug. Die Fenster sind sanierungsbedürftig und müssen ausgetauscht werden. Ein Fensterhersteller bietet an, Fenster beliebiger Qualität für Versuche zur Verfügung zu stellen. Diese Fenster haben unterschiedliche Glasqualitäten und unterschiedliche Größen. Eine eventuell notwendige Anpassung der Wanddurchbrüche für die Fenster verursacht beim ersten Einbau keine zusätzlichen Kosten, da ein anderer Freund, der als Maurer arbeitet, dir noch einen Gefallen schuldet.

d) Szenarioaufbau

Simulationselemente

- **Fenster:** Die Fenster besitzen einen Kunststoffrahmen und Doppelverglasung.

Parameter

- **Fenster**
 - Schalldämmmaß

- **Wand**
 - Schalldämmmaß

Listen

- **Bewertetes Schalldämmmaß und das Durchhören von Sprache bzw. Geräten.**
Diese Liste wird benötigt, damit algorithmisch entschieden werden kann, ob die Schalldämmung für die Anforderungen ausreicht.

Senderraum	Empfangsraum	R'_w ⁸³
Normale Sprache	nicht zu hören	55
Laute Sprache	nicht mehr zu hören	60

Tabelle 30: Bewertetes Schalldämmmaß und das Durchhören von Sprache bzw. Geräten
(Quelle: Bläsi (2002, S. 211))

e) Wirkungsgeflecht

I Systemgrafik

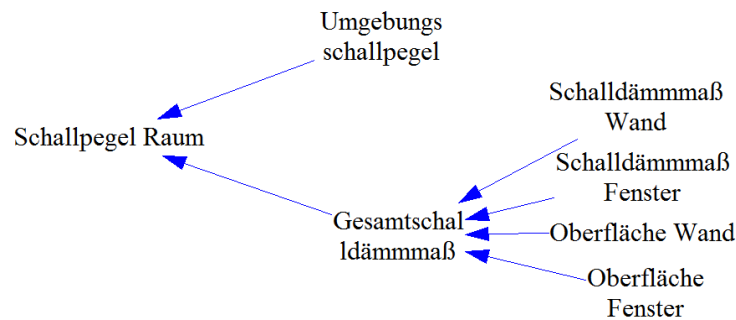


Abbildung 17: Wirkungsgeflecht Basisszenario Akustik [B002]

II Abhängigkeitsmatrix

Typ	beeinflusst							
	Wird beeinflusst von	Gesamtschalldämmmaß	Oberfläche Fenster	Oberfläche Wand	Schalldämmmaß Fenster	Schalldämmmaß Wand	Schallpegel Raum	Umgebungsschallpegel
K	Gesamtschalldämmmaß							
K	Oberfläche Fenster							
K	Oberfläche Wand							
K	Schalldämmmaß Fenster							
K	Schalldämmmaß Wand							
	Schallpegel Raum		+	-	+	+		+

⁸³ Annahme: Grundgeräusch von 30 dB im Empfangsraum.

E Umgebungsschallpegel

Tabelle 31: Abhängigkeitsmatrix Basisszenario Akustik [B002]

III Größenverzeichnis

Name	Symbol	Einheit	Definition
Gesamtschalldämmmaß	R_{ges}	dB	Das Gesamtschalldämmmaß der Wand mit einem Fenster (Bewertetes Schalldämmmaß)
Schalldämmmaß Fenster	R_F	dB	Das Schalldämmmaß des Fensters.
Schalldämmmaß Wand	R'_W	dB	Das bewertete Schalldämmmaß der Wand ⁸⁴ .
Schallpegel Raum ^{85,86}	$L_{P,R}$	dB	Der resultierende Schallpegel im Raum.
Umgebungsschallpegel	$L_{P,A}$	dB	Schallpegel, der außen vor der Wohnung herrscht. Dieser wird - ähnlich wie die Außentemperatur des Basisszenarios [B001] - entweder durch Messdaten oder durch eine simulierte Kurve vorgegeben.

Tabelle 32: Größenverzeichnis Basisszenario Akustik [B002]

IV Berechnungsvorschriften

IV.I Gesamtschalldämmmaß R_{ges}

Das Gesamtschalldämmmaß der Wand wird berechnet gemäß der Formel

$$R_{ges} = R'_W - 10 * \lg \left(1 + \frac{A_F}{A_W} * \left(10^{\frac{R'_W - R_F}{10}} - 1 \right) \right)$$

mit:

A_F : Oberfläche Fenster

A_W : Oberfläche Wand

(Aus: Tabelle 22: Größenverzeichnis Basisszenario Wärme)

Formel 5: Gesamtschalldämmmaß einer Wand (S. Bläsi, 2002, S. 220)

IV.II Schallpegel Raum $L_{P,R}$

Der Schallpegel im Raum wird im Rahmen einer groben Abschätzung berechnet gemäß der Formel

$$L_{P,R} = L_{P,A} - R_{ges}$$

Formel 6: Schallpegel im Raum

Der Einfluss der äquivalenten Schallabsorptionsfläche im Raum wird einfachheitshalber hier vernachlässigt.

⁸⁴ Hier wird das sogenannte Bauschalldämmmaß benutzt, das die Schallleitung über flankierende Bauteile mitberücksichtigt.

⁸⁵ Die physikalisch korrekte Bezeichnung ist *Schalldruckpegel*, der Einfachheit halber wird in dieser Arbeit das Wort *Schallpegel* genutzt.

⁸⁶ Der Unterschied zwischen (physikalischem in dB gemessenem) Schallpegel und (subjektivem, in dBA gemessenem) Lautstärkepegel wird in diesem Beispiel nicht dargestellt. Für die grundsätzliche Darstellung des Einflusses des Schalldämmmaßes hat sich die Verwendung des Schallpegels als ausreichend erwiesen.

f) Missionen

I Mission „Bringe das Gesamtschalldämmmaß der Fassadenwand auf 50 dB!“

Ziel

Der Spieler bringt das Gesamtschalldämmmaß der Fassadenwand auf mind. 50 dB durch den Austausch der Fenster.

Beschreibung

Neben der Durchführung von baulichen Veränderungen in Form des Austausches der Fenster mit Anpassung des Durchbruchs ist der Spieler gefordert, das Gesamtschalldämmmaß der Fassade zu berechnen. Die Fassade besteht aus einer Wand, in der ein Fenster eingelassen ist. Der Spieler muss ein passendes Ersatzfenster auswählen.

Belohnung

Fähigkeitspunkte

- **[F021]** Kennen des Parameters „Bewertetes Bauschalldämmmaß“
- **[F022]** Berechnen des Gesamtschalldämmmaßes eines einfachen, zusammengesetzten Bauteils
- **[FS001]** Durchführen von baulichen Veränderungen

II Mission „Keine Sprachfetzen von draußen“

Ziel

Die Wohnung ist soweit gegen Lärm isoliert, dass die Sprachfetzen der an der Wohnung vorbeihastenden Passanten in der Wohnung nicht mehr zu hören sind.

Beschreibung

Diese Mission fordert den Spieler, Verständnis dafür zu entwickeln, welches Schalldämmmaß erforderlich ist, um Schutz gegen die verschiedenen Arten von Sprache zu bieten (sprechende Menschen sowie Radio und Fernseher).

Belohnung

Fähigkeitspunkte

- **[F021]** Kennen des Parameters „Bewertetes Bauschalldämmmaß“
- **[F022]** Berechnen des Gesamtschalldämmmaßes eines einfachen, zusammengesetzten Bauteils
- **[F023]** Kennen der gegen charakteristische Geräusche erforderlichen Schalldämmmaße
- **[FS001]** Durchführen von baulichen Veränderungen

g) Tags

Fenster, Schalldämmmaß, Zusammengesetzte Bauteile

h) Vorausgesetzte Fähigkeiten

(keine)

i) Szenariovarianten

Das im Szenario benutzte Modell ist sehr vereinfacht. Zu den bisher noch nicht berücksichtigten Effekten, die in das Szenario eingebaut werden könnten, gehören:

Lautstärkeempfinden / Bewerteter Schallpegel: Das menschliche Gehör hat eine frequenzabhängige Empfindlichkeit. Diese führt zur Einführung der Größe „Lautstärke“. Der Unterschied zwischen beiden Größen kann in einem Szenario verdeutlicht werden. Eine mögliche Idee ist ein Probelauf der lokalen Verkehrsbetriebe, die neue Straßenbahnwagen testen mit jeweils unterschiedlichen Frequenzspektren der Betriebsgeräusche. Der Spieler darf dem Mieter Emm, vor dessen Haustür eine Straßenbahnlinie verläuft, beratend zur Seite stehen: Er misst den Schall mit den entsprechenden Messgeräten und gibt schließlich ein bewertendes Votum ab.

Frequenzabhängigkeit des Schalldämmmaßes: Durch geeignete Wahl eines dominant vorherrschenden Außenlärms, der sich vornehmlich in bestimmten Frequenzbereichen auswirkt, kann eine stärkere Schalldämmung als eigentlich mit Hilfe des bewerteten Schalldämmmaßes berechnet wurde, notwendig werden. Auch hier können im Spiel virtuelle Messgeräte zur Verdeutlichung eingesetzt werden.

Nachhallzeit: Die Nachhallzeit kann thematisiert werden, indem eine Nutzung des Raumes als Großraumbüro angenommen wird. In einem solchen Büro darf die Nachhallzeit einen bestimmten Wert (typischerweise 0,5 s) nicht überschreiten. Der Spieler muss eine geeignete Größe bzw. sonstige Umbaumaßnahmen für sein neues Büro vorschlagen.

5.5.5 Ergänzungsszenario: Schalldämmung einer Außenwand [E003]

a) Lernziele

- **[F021]** Kennen des Parameters „Bewertetes Bauschalldämmmaß“
- **[F022]** Berechnen des Gesamtschalldämmmaßes eines einfachen, zusammengesetzten Bauteils
- **[F024]** Kennen des rechnerischen Zusammenhangs von Schalldämmmaß und Masse.
- **[F025]** Berechnen des Gesamtschalldämmmaßes einer einschaligen Wand anhand der flächenspezifischen Masse

b) Basisszenario

[BK001] Basiskonfiguration: Einraumhaus (S. 69)

c) Titelgeschichte

Der Mieter Herr Emm hat wieder ein Problem mit dem Lärm in seiner Wohnung. Dieses Mal kommt der Lärm von der Rückseite seines Hauses aus einem Industriebetrieb. Der Wohnungsbesitzer, Herr Beh, hat gute Verbindungen zu einem Referenten des Studiengangs *eLBau* der Bauhaus-Universität Weimar. Dieses bauphysikalische Problem wird nun zum Gegenstand eines Praktikums im Rahmen des Kurses „Bauakustik“. Die Studenten beschäftigen sich mit dem Fall und stellen fest, dass für die Wand ein bewertetes Schalldämmmaß von 60 dB zu veranschlagen ist. Gleichzeitig bemerken sie, dass die Wände durch Risse beschädigt sind. Konsequenz ist, dass die alte Wand durch eine neue Wand ersetzt werden soll. Das in der Wand vorhandene Fenster soll nicht wieder erneuert werden, es wird ersatzlos gestrichen, da es im Raum ausreichend hell ist. An dieser Stelle wird die praktische Übung unterbrochen, mit einer Fortführung ist erst im nächsten Semester zu rechnen. Mieter Emm erwartet aber eine alsbaldige Lösung.

d) Szenarioaufbau

Simulationselemente

- **Fenster:** Die Fenster besitzen einen Kunststoffrahmen und Doppelverglasung.

Parameter

- **Fenster**
 - Schalldämmmaß
- **Wand**
 - Schalldämmmaß

Spieler-Aktionen

- **Entfernen des Fensters:** Das Fenster wird entfernt, der Wanddurchbruch wird mit dem Material, aus dem auch die Wand besteht, geschlossen.
- **Einbauen einer neuen Wand:** Die Wand wird neu erstellt. Der Spieler muss dazu Dicke und Material der neuen Wand auswählen.

Listen

- Baumaterial

Baumaterial	Dichte ρ in kg/m^3
Beton	2.300
Mauerziegel	1.400
Leichthochlochziegel	800
Kalksandstein	1800
Porenbeton	400
Leichtbetonsteine	1000

Tabelle 33: Dichte verschiedener Baumaterialien (Quelle: Bläsi (2002, S. 216))

e) Wirkungsgeflecht

I Systemgrafik

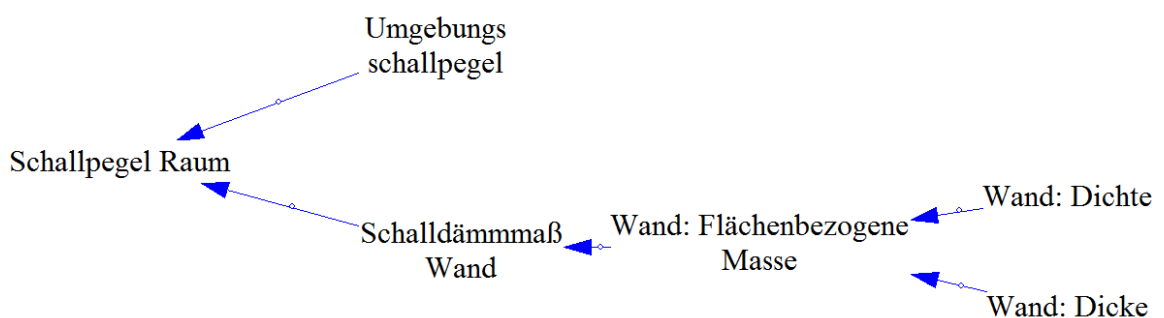


Abbildung 18: Wirkungsgeflecht Ergänzungsszenario Akustik [E003]

II Abhängigkeitsmatrix

Typ	beeinflusst Wird beeinflusst von					
		Schalldämmmaß Wand	Schallpegel Raum	Umgebungsschallpegel	Wand: Dichte	Wand: Dicke
	Schalldämmmaß Wand					+
	Schallpegel Raum	-		+		
E	Umgebungsschallpegel					
K	Wand: Dichte					
K	Wand: Dicke					
	Wand: Flächenbezogene Masse				+	+

Tabelle 34: Abhängigkeitsmatrix Ergänzungsszenario Akustik [E003]

III Größenverzeichnis

Name	Symbol	Einheit	Definition
Schalldämmmaß Wand	R'_w	dB	Das bewertete Schalldämmmaß der Wand
Wand: Dichte	ρ	kg/m ³	Die Dichte des Materials, aus dem die Wand aufgebaut ist
Wand: Dicke	d	m	Die Dicke der Wand
Schallpegel Raum	$L_{P,R}$	dB	Der resultierende Schallpegel im Raum
Umgebungsschallpegel	$L_{P,A}$	dB	Schallpegel, der außen vor der Wohnung herrscht. (s. [B002])
Wand: Flächenbezogene Masse	m'	kg/m ²	Die Masse der Wand pro Fläche

Tabelle 35: Größenverzeichnis Ergänzungsszenario Akustik [E003]

IV Berechnungsvorschriften

IV.I Flächenbezogene Masse m'

Die flächenbezogene Masse der Wand wird berechnet gemäß Formel 7:

$$m' = d * \rho$$

Formel 7: Flächenbezogene Masse einer Wand (S. Bläsi, 2002, S. 216)

IV.II Schalldämmmaß Wand R'_w

Das Schalldämmmaß der Wand wird als bewertetes Schalldämmmaß näherungsweise gemäß der folgenden Formel berechnet:

$$R'_w = 25 * \lg \frac{m'}{m_0} - 12 \text{ dB}$$

mit m_0 : bezugsflächenbezogene Masse (1 kg/m³)

Formel 8: Bewertetes Schalldämmmaß für einschalige Wände (S. Bläsi, 2002, S. 217)

Diese Formel berücksichtigt die Schallübertragung durch flankierende Bauteile, wenn diese eine Masse von mind. 300 kg/m^3 besitzen. Sie gilt nur für Wände ohne Öffnungen (u.a. für Türen oder Fenster).

IV.III Schallpegel Raum $L_{P,R}$

Der Schallpegel im Raum wird näherungsweise berechnet gemäß der Formel

$$L_{P,R} = L_{P,A} - R'_W$$

Formel 9: Schallpegel im Raum

Ein Grundgeräusch im Raum wird vernachlässigt.

f) Missionen

I Mission „Bringe das Gesamtschalldämmmaß der Wand auf 60 dB!“

Ziel

Der Spieler bringt das Gesamtschalldämmmaß der Fassade auf mind. 60 dB durch Entfernung des Fensters und Austausch der Wand.

Beschreibung

Neben der Durchführung von baulichen Veränderungen in Form der Entfernung des Fensters ist der Spieler gefordert, für die Wand ein Ersatzmaterial auszuwählen. Ebenso muss er die Wand dimensionieren - mit dem Ziel, das geforderte Gesamtschalldämmmaß zu erreichen.

Belohnung

Fähigkeitspunkte

- **[F021]** Kennen des Parameters „Bewertetes Bauschalldämmmaß“
- **[F025]** Berechnen des Gesamtschalldämmmaßes einer einschaligen Wand anhand der flächenspezifischen Masse
- **[FS001]** Durchführen von baulichen Veränderungen

II Mission „Nicht mehr als einen halben Meter“

Ziel

Das Material der Wand soll so gewählt werden, dass die Dicke der Wand nicht mehr als 50 cm beträgt und trotzdem ein Schalldämmmaß von mind. 60 dB erreicht wird.

Beschreibung

Mit dieser Mission soll das Augenmerk des Spielers auf die Wanddicke gelenkt werden. Mit jedem der genannten Baustoffe kann die Wand so dimensioniert werden, dass das geforderte Schalldämmmaß erfüllt werden kann, allerdings auf Kosten der Dicke der Wand.

Belohnung

Fähigkeitspunkte

- **[F021]** Kennen des Parameters „Bewertetes Bauschalldämmmaß“
- **[F025]** Berechnen des Gesamtschalldämmmaßes einer einschaligen Wand anhand der flächenspezifischen Masse
- **[FS001]** Durchführen von baulichen Veränderungen

g) Tags

Fenster, Schalldämmmaß, Einschalige Wände, flächenbezogene Masse

h) Vorausgesetzte Fähigkeiten

(keine)

i) Szenariovarianten

Zusammengesetzte Bauteile: Das Szenario kann gespielt werden unter der Maßgabe, dass das Fenster erhalten bleibt. Dadurch werden andere Rechenwege als Entscheidungsgrundlage vom Spieler gefordert.

Zeitlich schwankende Geräusche: Gewöhnlich ist eine Schallbelastung nicht durch einen gleichbleibenden Schallpegel gekennzeichnet. Um eine Beurteilung einer Schallbelastung zu ermöglichen, wird deshalb der sogenannte **Mittelungspegel** genutzt, der ein Maß für die über den Betrachtungszeitraum gemittelte Schallenergie ist. Dieser kann u.a. zur Beurteilung von Verkehrslärm eingesetzt werden. Daher bietet es sich beispielsweise an, in einem Szenario die Auswirkungen der Änderung der erlaubten Höchstgeschwindigkeit auf der Straße vor der Wohnung von 30 km/h auf 50 km/h zu hinterfragen (Lutz et al., 2002, S. 11).

5.5.6 Ergänzungsszenario: Schalldämmung von Zwischenwänden [E004]

a) Lernziele

- **[F021]** Kennen des Parameters „Bewertetes Bauschalldämmmaß“
- **[F022]** Berechnen des Gesamtschalldämmmaßes eines einfachen, zusammengesetzten Bauteils
- **[F025]** Berechnen des Gesamtschalldämmmaßes einer einschaligen Wand anhand der flächenspezifischen Masse
- **[F026]** Berechnung des Gesamtschallpegels durch energetische Addition

b) Basisszenario

[B002] Schallpegel in Abhängigkeit von der Fenstergröße und -qualität (S. 92), allerdings unter Verwendung der Basiskonfiguration [BK002] (S.74)

c) Titelgeschichte

Der Mieter Herr Emm klagt neben dem Lärm von der Straße auch über unzumutbaren Lärm aus der Nebenwohnung, wenn seine Nachbarin - eine Liebhaberin lauter Rockmusik - ihre Stereoanlage benutzt. Aufgrund ähnlicher Probleme in einer früheren Wohnung weiß Herr Emm, dass er sich ab einem Schallpegel von 50 dB (das entspricht dem Schallpegel einer Unterhaltung) gestört fühlt, er kann aber mit 40 dB sehr gut leben. Von der Straßenseite ist mit einem Schallpegel von 70 dB zu rechnen, die Stereoanlage seiner Nachbarin schafft 80 dB. Da der Vermieter Herr Beh aus persönlichen Gründen die Nachbarin nicht zu einer Reduzierung der Lautstärke verpflichten möchte, spendiert dieser Herrn Emm einen Umbau seiner Wohnung. Zunächst sollte es reichen, die Zwischenwand auszutauschen. Erst wenn das nicht genügt, werden die Mittel für einen Austausch der Fenster bereitgestellt. Sollte auch das nicht zum Ziel führen, so kann die Fassade wand ausgetauscht werden.

d) Szenarioaufbau

Simulationselemente

- **Fenster:** Die Fenster besitzen einen Kunststoffrahmen und Doppelverglasung.

Parameter

- **Fenster**
 - Schalldämmmaß
- **Wand**
 - Schalldämmmaß

Spieler-Aktionen

- **Einbauen einer neuen Wand:** Die Wand wird neu erstellt. Der Spieler muss dazu Dicke und Material der neuen Wand auswählen. Diese Aktion bezieht sich sowohl auf Innen- als auch auf Außenwand.

Listen

- **Baumaterialien für Innen- und Außenwände**
Die Baumaterialien für Innen- und Außenwände sind aufgeführt in der Tabelle 33 auf S. 60.

e) Wirkungsgeflecht

I Systemgrafik

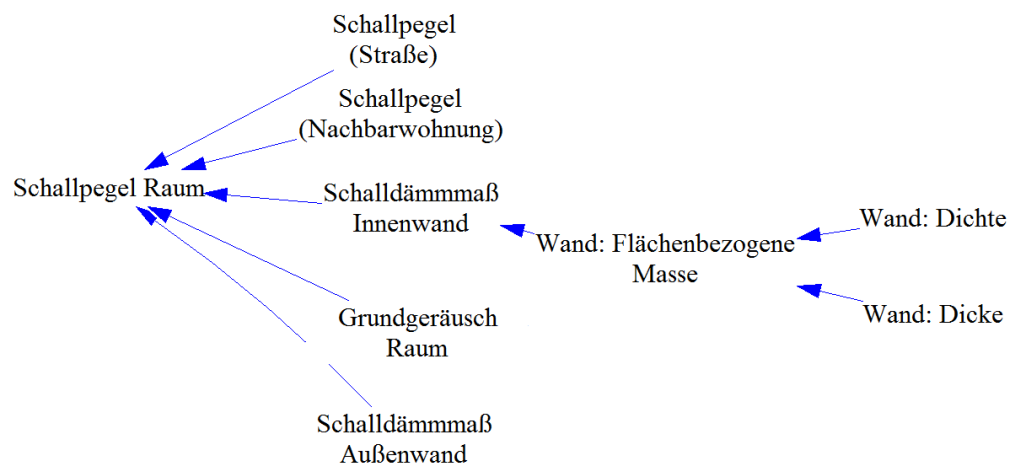


Abbildung 19: Wirkungsgeflecht Ergänzungszenario Akustik [E004]

II Abhängigkeitsmatrix

Typ	Wird beeinflusst von	beeinflusst								
		Grundgeräusch Raum	Schalldämmmaß Außenwand	Schalldämmmaß Innenwand	Schallpegel Nachbarwohnung	Schallpegel Raum	Schallpegel Straße	Wand: Dichte	Wand: Dicke	Wand: Flächenbezogene Masse
K	Grundgeräusch Raum									
K	Schalldämmmaß Außenwand									
K	Schalldämmmaß Innenwand									+
E	Schallpegel Nachbarwohnung									
	Schallpegel Raum	+	-	-	+		+			
E	Schallpegel Straße									
K	Wand: Dichte									
K	Wand: Dicke									
	Wand: Flächenbezogene Masse							+	+	

Tabelle 36: Abhängigkeitsmatrix Ergänzungsszenario Akustik [E004]

III Größenverzeichnis

Name	Symbol	Einheit	Definition
Grundgeräusch Raum	$L_{P,R,G}$	dB	Ein im Raum vorhandener Grundgeräuschpegel, wird als konstant angenommen.
Schalldämmmaß Außenwand	$R'_{w,A}$	dB	Das bewertete Bauschalldämmmaß der Außenwand. Die Außenwand ist fensterlos ⁸⁷ .
Schalldämmmaß Innenwand	$R'_{w,I}$	dB	Das bewertete Bauschalldämmmaß der Wohnungstrennwand zur Nachbarin
Schallpegel Nachbarwohnung	$L_{P,N}$	dB	Der Schallpegel in der Nachbarwohnung wird aus einem Profil (einem zeitlichen Verlauf des Schallpegels) gespeist (s. [B002], Umgebungsschallpegel), kann mitunter 80 dB betragen.
Schallpegel Raum	$L_{P,R}$	dB	Der Schallpegel im Raum, die zu betrachtende Zielgröße.
Schallpegel Straße	$L_{P,A}$	dB	Schallpegel, der außen vor der Wohnung herrscht (s. [B002], dort Umgebungsschallpegel genannt) ⁸⁸ .
Wand: Dichte	ρ	kg/m ³	Die Dichte des Materials, aus dem die Wand aufgebaut ist.

⁸⁷ In einer möglichen Szenariovariante (s.S. 103) kann diese vereinfachende Annahme aufgehoben werden. Sie berührt hier aber nicht die Lernziele des Szenarios.

⁸⁸ Diese Größe wird im Szenario mit Hilfe eines Profils simuliert. In einer Szenariovariante ist es denkbar, den durch die Straße verursachten Geräuschpegel aufgegliedert zu betrachten (s. Szenariovarianten, S. 102)

Wand: Dicke	d	m	Die Dicke der Wand
Wand: Flächenbezogene Masse	m'	kg/m ²	Die Masse der Wand pro Fläche

Tabelle 37: Größenverzeichnis Ergänzungsszenario Akustik [E004]

IV Berechnungsvorschriften

IV.I Schalldämmmaße Innenwand $R'_{W,I}$ und Außenwand $R'_{W,A}$ sowie flächenbezogene Masse m'

Die Schalldämmmaße der Innenwand und Außenwand berechnen sich nach Formel 8 unter Verwendung der flächenbezogenen Masse (Formel 7 auf S. 98). Dabei werden Dichte und Dicke der jeweiligen Wand genutzt.

IV.II Schallpegel Raum $L_{P,R}$

Der Schallpegel im Raum wird aus allen Einzelschallpegeln berechnet, es wird „die Summe unter dem Logarithmus gebildet“ (Lutz et al., 2002, S.9). Somit ergibt sich der resultierende Schallpegel aus dem Schallpegel in der Nachbarwohnung $L_{P,N}$ und dem Schallpegel auf der Straße vor dem Raum $L_{P,A}$ (vermindert jeweils um das entsprechende Schalldämmmaß) sowie dem Schallpegel des Grundgeräusches.

$$L_{P,R} = 10 * \lg \left(10^{\frac{L_{P,N} - R'_{W,I}}{10}} + 10^{\frac{L_{P,A} - R'_{W,A}}{10}} + 10^{\frac{L_{P,R,G}}{10}} \right)$$

Formel 10: Schallpegel Raum

f) Missionen

I Mission „Bringe das bewertete Schalldämmmaß der Innenwand auf 60 dB!“

Ziel

Der Spieler bringt das bewertete Schalldämmmaß der Wohnungstrennwand auf mind. 60 dB durch den Austausch der Trennwand.

Beschreibung

Der Spieler tauscht die Trennwand gegen eine solche Wand aus, deren bewertetes Schalldämmmaß mindestens 60 dB erreicht. Der Spieler muss ein geeignetes Baumaterial auswählen und dieses dimensionieren.

Belohnung

Fähigkeitspunkte

- **[F021]** Kennen des Parameters „Bewertetes Bauschalldämmmaß“
- **[F025]** Berechnen des Gesamtschalldämmmaßes einer einschaligen Wand anhand der flächenspezifischen Masse
- **[F026]** Berechnung des Gesamtschallpegels durch energetische Addition
- **[FS001]** Durchführen von baulichen Veränderungen

II Mission „40 dB für die Wohnung“

Ziel

Bringe den maximalen Schallpegel - bezogen auf die gegenwärtigen Bedingungen - in der Wohnung auf 40 dB.

Beschreibung

Der Spieler kann sowohl Innenwand als auch Außenwand austauschen. Er hat darauf zu achten, dass die Innenwand aus baulichen Gründen nicht dicker als 50 cm wird. Er wählt das entsprechende Baumaterial für Innenwand und ggfs. auch Außenwand aus und dimensioniert die jeweilige Wand. Unter Umständen muss auch das Fenster in der Außenwand zur Straßenseite dimensioniert werden. Der Gesamtschallpegel wird durch die energetische Addition der Einzelschallpegel bestimmt.

Belohnung

Fähigkeitspunkte

- **[F021]** Kennen des Parameters "Bewertetes Bauschalldämmmaß"
- **[F022]** Berechnen des Gesamtschalldämmmaßes eines einfachen, zusammengesetzten Bauteils
- **[F025]** Berechnen des Gesamtschalldämmmaßes einer einschaligen Wand anhand der flächenspezifischen Masse
- **[F026]** Berechnung des Gesamtschallpegels durch energetische Addition
- **[FS001]** Durchführen von baulichen Veränderungen

g) Tags

Fenster, Schalldämmmaß, Zusammengesetzte Bauteile, Gesamtschallpegel, Flächenspezifische Masse

h) Vorausgesetzte Fähigkeiten

(keine)

i) Szenariovarianten

Doppelschalige Trennwände: Um die gewünschten Schalldämmmaße zu erreichen, müssen die Innenwände sehr dick dimensioniert werden. Eine der möglichen Abhilfe sind doppelschalige Trennwände mit Dämmplatten in der Fuge (Lutz et al., 2002, S. 47).

Vorsatzschalen: Damit die Außenwände aus Schallschutzgründen weniger mächtig ausgelegt werden müssen, können Vorsatzschalen eingesetzt werden. Darauf aufbauend kann das Merkmal „Resonanzfrequenz“ (ab dieser Frequenz besitzen die Vorsatzschalen eine Dämmwirkung) eingeführt werden.

Außenwand mit Fenstern: Grundsätzlich können die zu zeigenden Effekte des Szenarios mit einer Außenwand demonstriert werden, in der weder Fenster noch Türen eingelassen sind. Um zu einem realistischeren Szenario zu gelangen, kann die Außenwand mit Fenstern und/oder Türen ausgestattet werden.

Differenzierung der Schallpegelquelle Straße: In den *RLS-90 - Richtlinien für den Lärmschutz an Straßen* sind Berechnungsvorschriften enthalten, die es u.a. erlauben, aus Parametern der Straßenbeschaffenheit sowie der Verkehrsdichte und -zusammensetzung einen zu erwartenden Schallpegel an einem exakt definierten Messpunkt zu berechnen. Das vorliegende Szenario kann derart erweitert werden, dass dem Spieler die Grundlagen der Schallquelle „Straße“ vermittelt werden.

5.5.7 Kombinationsszenario: Schalldämmung und Wärmedämmung [K001]

a) Lernziele

In diesem Szenario hat der Spieler bei seinen Überlegungen und den resultierenden Aktionen sowohl auf Aspekte der Wärmedämmung als auch auf solche der Schalldämmung zu achten.

- **[F030]** Kennen des möglichen Zielkonflikts aus Wärmedämmeigenschaft und Schalldämmeigenschaft eines Materials.
- **[F031]** Finden einer optimalen Lösung für divergierende Anforderungen von Wärmeschutz und Schallschutz

b) Grundszenarien

- Ergänzungsszenario: Entwicklung der Transmissionswärmeverluste in Abhängigkeit von der Dämmungsqualität [E001]
- Ergänzungsszenario: Schalldämmung einer Außenwand [E003]

c) Titelgeschichte

Als Titelgeschichte wird diejenige des Ergänzungsszenarios zur Schalldämmung genutzt (s. S. 65). Zusätzlich wird der Hinweis gegeben, dass der Wärmeschutz nicht vernachlässigt werden sollte. Dazu wird als erste zu lösende Mission eine des Szenarios zum Wärmeschutz ausgeschrieben: Die Mission „Baumaterial als Wärmedämmung“ fordert vom Spieler, die Anforderungen der EnEV einzuhalten. Danach kann der Spieler durch die Lösung der weiteren Schallschutz-Missionen seine Fähigkeiten ausbauen.

d) Szenarioaufbau

Kumulativer Aufbau der Grundszenarien

Listen

Tabelle 33: Dichte verschiedener Baumaterialien auf S. 97 wird erweitert um eine Spalte für die Wärmeleitfähigkeit (aus: Schneider (2001, Tafel 10.16)).

Baumaterial	Dichte ρ in kg/m ³	Wärmeleitfähigkeit in W(m ² *K)
Beton	2.300	1,85
Mauerziegel	1.400	0,58
Leichtlochziegel	800	0,33
Kalksandstein	1800	0,99
Porenbeton	400	0,14
Leichtbetonsteine	1000	0,49

Tabelle 38: Dichte und Wärmeleitfähigkeit verschiedener Baumaterialien

e) Missionen

Die erste Mission soll den Spieler sensibilisieren, die Aspekte des Wärmeschutzes nicht außer Acht zu lassen. Dazu wird die Mission „Baumaterial als Wärmedämmung“ ausgewählt.

Die weiteren Missionen sind die des Ergänzungsszenarios zum Schallschutz (E003).

Werden die Ziele der Mission „Baumaterial als Wärmedämmung“ eingehalten, während die anderen Missionen gelöst werden, werden Fähigkeitspunkte für das Lernziel „[F031] Finden einer optimalen Lösung für divergierende Anforderungen von Wärmeschutz und Schallschutz“ sowie

für das Lernziel „[F030] Kennen des möglichen Zielkonflikts aus Wärmedämmeigenschaft und Schalldämmeigenschaft eines Materials“ vergeben. Werden die Wärmeschutz-bezogenen Ziele hingegen nicht eingehalten, so werden nur die Punkte der Fähigkeit F030 vergeben und der Spieler auf die Nichterfüllung des gewünschten Wärmeschutzes hingewiesen.

f) Vorausgesetzte Fähigkeiten

Lernziele der Grundszenarien

g) Tags

Tags der Grundszenarien + „Kombination Wärme - Schall“

5.5.8 Weitere Ansätze für Szenarien

An dieser Stelle werden Ansätze für zusätzliche Szenarien dokumentiert, die während der Erstellung dieser Arbeit betrachtet wurden, letztlich aber nicht detailliert dargestellt wurden.

Zusätzliche Schall-Kennwerte zur Bewertung der Lärmbelästigung in Räumen: Anhand des Beispiels einer Intensivstation im Krankenhaus können weitere Kennwerte zur Charakterisierung des Schalls eingeführt werden (Salandin, Arnold, & Kornadt, 2011). Die Aktionen des Spielers bestehen darin, für eine Verminderung der Schallemissionen zu sorgen. Dies kann durch Austausch von Maschinen erfolgen, durch die Anbringung von Schallreduzierenden Schutzabdeckungen oder sonstige bauliche Maßnahmen. Die Lernziele für den Spieler umfassen das Kennen und Anwenden der zusätzlichen Schallkennwerte sowie der möglichen Maßnahmen, um die Kennwerte auf ein erträgliches Maß zu senken.

Thermische Behaglichkeit: Um die thermische Behaglichkeit der Nutzer zu gewährleisten, wird das Raumklima u. a. in Bezug auf Temperatur, rel. Luftfeuchte und Strömung optimiert (Völker & Kornadt, 2011; Lichtenheld, 2011). Wesentliche Einflussgrößen sind beispielsweise die Raumgeometrie und -ausrichtung, der Wandaufbau oder aber die Klimatechnik. Diese kann der Spieler entsprechend verändern, um die thermische Behaglichkeit eines fiktiven Bewohners sicherzustellen. Lernziele solcher Szenarien sind das Wissen um entsprechende Kennwerte, das Kennen und Anwenden verschiedener Heizungstypen sowie anderer Raumklimaanlagen.

Anwendung von Raumklimata: Wie bereits in einigen Szenarien dargestellt, können standortbezogene Außenklimata mit Hilfe von *Test Reference Years* (TRY) abgebildet werden. Derzeit werden beispielsweise im Rahmen des Forschungsprojekts *Referenzinnenraumklima* Temperatur- und Feuchteverläufe in Innenräumen aufgezeichnet (Hofmann, Geier, & Kornadt, 2011). Diese eröffnen ein breites Anwendungsspektrum: Es sind Szenarien denkbar, in denen der Spieler das vorherrschende Innenraumklima wählen kann, um die Auswirkungen auf das Wärme- und Feuchteverhalten verschiedener Bauteilkonstruktionen zu simulieren. Ebenso kann demonstriert werden, welchen Heizenergiebedarf ein bestimmtes Innenraumklima erfordert. Im Zusammenspiel mit Prognosemodellen zur Schimmelpilzbildung können zusätzlich Szenarien erstellt werden, die den Spieler mit einem geeigneten Lüftungskonzept experimentieren lassen.

Schalleinleitung und -weiterleitung: Für den Bau von Gebäuden wird die Beachtung akustischer Aspekte immer wichtiger: u.a. durch gestiegene Nutzeranforderungen auf der einen Seite und der Nutzung zusätzlicher Haustechnik, neuartiger Baumaterialien sowie leichter Baukonstruktionen auf der anderen Seite. Zu aktuellen Forschungsthemen gehören die Einleitung und Weiter-

leitung von Körperschall. Die Größe der von einer Schallquelle ins Gebäude eingetragenen Schallenergie lässt sich durch die Parameter Quellleistung und Quellimpedanz sowie der Eingangsimpedanz des Bauteils charakterisieren, an welche die Schallquelle angekoppelt ist (Vogel, Kornadt, Wittsock, & Scholl, 2012). Für die Beschreibung der Schallweiterleitung können sogenannte Übertragungsfunktionen herangezogen werden (Kornadt, Arnold, Wittstock, & Scholl, 2012). Um dem Spieler die Einfluß dieser Größen zu verdeutlichen, werden Szenarien vorgeschlagen, in denen Wandstrukturen und haustechnische Anlagen als einleitende Körperschallquellen variiert werden können.

Konvektiver Feuchtetransport: Schimmelbildung kann eine negative Folge von Feuchteeintrag durch Leckagen in der Gebäudehülle sein (J. Schmidt & Kornadt, 2010). Um dem Spieler einen Eindruck dieses Phänomens zu vermitteln, können Szenarien entwickelt werden, in denen die Gebäudehülle durch unterschiedliche Konstruktionen dargestellt wird. Diese sind verschieden stark anfällig für Leckagen oder permeabel (z.B. Dämmstoffe). Durch einen Zeitraffer im Ablauf des Szenarios können Hüllflächeninfiltrationen und deren Folgen veranschaulicht werden. Der Spieler kann verschiedene bauliche Maßnahmen zur Reduzierung von Leckagen bzw. zur Erhöhung der Luftdichtheit der Gebäudehülle ergreifen. Zur realistischen Simulation von Druckdifferenzen können real gemessene Druckdifferenzprofile in die Szenarien integriert werden (Völker et al., 2013).

6 Zusammenfassung und Diskussion

6.1 Ergebnisse

Im Rahmen der Arbeit wurde ein Konzept für eine Spielplattform entwickelt, die zur Begleitung der akademischen bauphysikalischen Ausbildung genutzt werden kann. Dabei wurden in einem multidisziplinären Ansatz verschiedene Teilkonzepte unter Verwendung von Vorgehensweisen u.a. aus den Disziplinen Informatik, Didaktik und Bauphysik erstellt und zu einem Gesamtkonzept integriert:

- **Systemarchitektur:** Es wurden die notwendigen Komponenten einer Mehrspieler-Spielplattform entwickelt, um sie in eine Client-Server-Systemarchitektur einzubetten. Die Systemeigenschaften *Erweiterbarkeit* und *Modularität* wurden als essentiell erkannt und sind prägend für das Konzept. Nicht zuletzt hierdurch ermöglicht es die Entwicklung einer nachhaltigen Plattform.
- **Simulationskern & Simulationsmodell:** Aus den Eigenschaften bestehender erfolgreicher bauphysikalischer Simulationssoftware wurden Anforderungen an bauphysikalische Simulationen abgeleitet. Dieses Anforderungsprofil wurde in den Entwurf eines Simulationskerns umgesetzt. Jener kann ein Simulationsmodell verarbeiten, das speziell den Eigenschaften von Computerspielen Rechnung trägt. Das Szenario wurde als zentrale Struktureinheit identifiziert.
- **Modul- und Erweiterungskonzept:** Um beliebige benutzerdefinierte Szenarien in der Spielplattform zu ermöglichen, wurde ein Modulkonzept entwickelt. Es basiert auf den Eigenschaften der Basissoftware *Eclipse RCP* und erlaubt es, Module zu entwickeln, die die Plattform mit Szenarien sowie Objekten des Simulationsmodells erweitern.
- **Pädagogisches Modell:** Zur Versorgung des Lerners mit Aufgaben, die seinem Fähigkeitsstand angemessen sind, wurde ein lernzielorientiertes Modell entwickelt. Fähigkeiten entsprechen den Lernzielen. Mit der erfolgreichen Absolvierung von Missionen erhält der Spieler Fähigkeitspunkte. Szenarien können die Erreichung von Lernzielen voraussetzen. Damit werden Szenarios nach Schwierigkeitsgrad abgegrenzt.
- **Spielkonzept:** Es wurde ein übergreifendes Spielkonzept definiert, in das die Szenarios integriert werden können. Insbesondere eine narrative Einbettung hilft bei der Erstellung eines Kontextes, der es dem Spieler erleichtert, sich mit den Aufgaben der Szenarios zu identifizieren.
- **Viralkonzept:** Es wurde eine Reihe von Maßnahmen vorgeschlagen, die zu einer Stärkung der Anziehungskraft der Plattform führen und die Plattform immer wieder direkt oder indirekt dem potentiellen Spieler in Erinnerung rufen. Dazu gehören die Einbettung in einen Social Network Service (SNS), die Nutzung der Mechanismen herkömmlicher Social Network Games (SNGs) sowie dedizierte Mehrspieler-Mechanismen.

Sämtliche oben genannten Konzepte wurden prototypisch in Software umgesetzt und damit die Anwendbarkeit validiert.

Zusätzlich wurden durch die Betreuung von Abschlussarbeiten am Lehrstuhl für Bauphysik weitere Themenbereiche beleuchtet, die in das Konzept eingeflossen sind bzw. noch aufgenommen werden können. Dazu zählen:

- **Entwurf von fachspezifischen SNGs**

In diesen Arbeiten werden Antworten gegeben auf die Frage, wie fachspezifische Inhalte in SNGs eingebunden werden müssen, damit die Spieler diese erlernen können (Kühnert, 2011; Laub & Tetzner, 2011).

- **Taxonomie von sozialen Interaktionen**

Diese Arbeit untersucht, welche sozialen Interaktionen in SNGs vorkommen und wie sich diese nutzen lassen, um die Spieler in Kontakt miteinander zu bringen. Ziel ist es, den sozialen Kontext in den Spielen zu stabilisieren, damit dieser zum Lernergebnis beitragen kann (Luthardt, 2012).

- **Automatisierte Qualitätssicherung in Crowdsourcing-basierten Systemen**

Die vorgeschlagene Plattform hängt stark von benutzererzeugten Inhalten ab. Um den Aufwand für die manuelle Qualitätssicherung dieser Inhalte in Grenzen zu halten, sollten automatisierte Mechanismen zur Qualitätssicherung genutzt werden (Schürenberg, 2012).

- **Erweiterte Methoden zur Bewertung des Spielerverhaltens**

Neben der reinen Erfüllung der aufgetragenen Missionen lässt sich die Leistung eines Spielers auch durch sein Verhalten im Spiel messen. Für ein kommerzielles SNG wurde diesbezüglich ein Konzept erstellt (Naira Müller et al., 2012; Söbke, Hadlich et al., 2012).

- **Spielmechanismen in Nicht-Spiel-Anwendungen**

Die Bauphysik als eine Ingenieurwissenschaft beschäftigt sich mit Systemen der realen Welt. Computerspiele sprechen in dieser Domäne nicht alle Benutzer an. Daher ist es eine Möglichkeit, Gamification-Elemente in Nicht-Spiel-Anwendungen zur Steigerung der Benutzermotivation zu integrieren. Es wurde untersucht, welche dieser Elemente von anderen Anwendungen zu dem Zweck erfolgreich genutzt werden (Borowski, 2012).

- **Anpassung etablierter Software und Spiele**

Die Entwicklung von Spielen ist aufwändig. Aus diesem Grund sollte ein Lösungsansatz immer sein, die Anpassungsmöglichkeit vorhandener kommerzieller Computerspiele auf Fachinhalte zu überprüfen. Hier konnten ermutigende Ergebnisse gefunden, aber auch Irrwege entdeckt werden (Aubel, 2012; Franke, 2012; Großmann, 2012; Müller, 2012).

Mit Hilfe der im Rahmen dieser Arbeit entstandenen Ergebnisse konnte eine profunde Wissensbasis aufgebaut werden, die bei der Durchführung zukünftiger Projekte helfen wird. Als nächster Schritt auf dem Weg zu einem einsatzbereiten System zur Ausbildungsbegleitung ist zum einen der Ausbau der Plattform zu einer robusten Software mit integriertem Szenarioeditor zu nennen. Zum anderen ist darauf aufbauend eine Evaluierung der tatsächlichen mit dieser Plattform erzielbaren Lerneffekte wünschenswert. Es ist zu prüfen, inwieweit die Evaluierung zumindest teilweise mit Hilfe des SNG *Easy Gold* durchgeführt werden kann. Für die Erweiterung von *Easy Gold* sind bedeutend geringere Aufwände zu erwarten.

6.2 Bewertung und Thesen

Die Arbeitsergebnisse werden in diesem Kapitel in Form von Thesen zusammengefasst. Jede These wird begründet.

These 1: Die aus der Softwareentwicklung bekannten Prinzipien *Modularität* und *Erweiterbarkeit* lassen sich auch auf die Spielentwicklung anwenden.

Dieses originäre Ziel der Arbeit wurde erreicht: Es konnte eine Verbindung geschaffen werden zwischen der Spielentwicklung und der Eclipse RCP-Plattform. Das Modell einer Spielsimulation lässt sich in (Java-)Klassen abbilden. Gleichzeitig konnten die verschiedenen Elemente des Simulationsmodells aspektspezifisch in Module aufgeteilt werden. Damit wird zum einen die Entwicklung einer wartbaren Software unterstützt: Ein Modul lässt sich leichter durch eine andere Implementierung mit derselben Schnittstelle austauschen als ein monolithisches Softwaregebilde. Zum anderen kann die Softwareentwicklung durch die Definition klarer Modulschnittstellen auf verschiedene Partner verteilt werden. Desweiteren ermöglicht die Unterteilung in Module auch, die Szenarien vom Einfachen zum Komplexen aufzubauen, indem sukzessive Module hinzu geschaltet werden.

Nicht demonstriert werden konnte das tatsächliche Hinzufügen von Szenarios und Modulen mit Hilfe des Szenario-Editors. An dieser Stelle ist erheblicher Konzeptions- und Implementierungsaufwand zu erwarten.

These 2: Die strenge Unterteilung in Module kostet Entwicklungsgeschwindigkeit.

Während der prototypischen Implementierung einiger Szenarien hat sich gezeigt, dass die strenge Einhaltung der Modularisierung aufwändig ist und die Effizienz der Entwicklung bezogen auf den Prototyp verringert. Die Entwicklung der serverseitigen Funktionalität des Spiels *Easy Gold* (s. *Anhang F*) konnte weit schneller durchgeführt werden, da dort weniger Aufwand in die Modularisierung investiert wurde.

Der erhöhte Aufwand ist auf der einen Seite sicherlich der prototypischen Vorgehensweise zuzuordnen, in der viele Einzelheiten noch zu erkunden waren. Auf der anderen Seite fällt jedoch konzeptioneller Aufwand bei der Definition der Abhängigkeiten der Module und bei der Auswahl der wiederverwendbaren Simulationselemente an. Mit wachsender Erfahrung sollte der Aufwand wieder geringer werden, er ist aber nicht zu unterschätzen. Es ist notwendig, eine stabile Basis zu etablieren.

These 3: Das Szenario ist eine geeignete Struktureinheit für die Plattform.

Das Artefakt „Szenario“ hat sich in vielerlei Hinsicht als geeignete Größe für den Einsatz auf der Plattform herausgestellt:

- **Unterschiedliche Komplexitäten:** Ein Szenario kann in verschiedenen Komplexitäten definiert werden: Für die Vermittlung von grundlegenden Prinzipien reichen einige wenige Simulationselemente mit wenigen Parametern. Gleichzeitig können auch komplexe Systeme mit umfangreichen Hierarchien von Simulationselementen mit zahlreichen Parametern in einem Szenario untergebracht werden.
- **Modulare Erweiterungseinheit:** Mit Hilfe des Szenarios als Struktureinheit wird die Plattform skalierbar. Sie kann nach und nach erweitert werden kann. Ein Szenario ist in sich abgeschlossen und kann einzeln betrachtet werden. Bei Fehlern ist nur das Szenario selbst betroffen, das korrigiert, deaktiviert oder ausgetauscht werden kann. Das ist ein großer Vorteil gegenüber einem einzigen zusammenhängenden Softwarepaket.
- **Lerneinheit:** Als Lerneinheit ist ein Szenario ebenfalls sehr geeignet. Es ist übersichtlich für den Spieler. Bei Problemdiskussionen kann sich der Spieler auf das Szenario bezie-

hen. Hier gibt es eine Analogie zu Aufgaben in der herkömmlichen Lehre. Ebenso wird wenig Zeit benötigt, um ein Szenario zu spielen. Dadurch wird beiläufiges Spielen („casual game play“) unterstützt.

- **Artefakteinheit:** Ein Szenario kann einfach und schnell durch den Spieler erstellt werden. Mit wachsender Erfahrung kann der Spieler auch umfangreichere und komplexere Szenarien entwickeln. Zusätzlich kann jedes Szenario einzeln bewertet werden - durch das Spielverhalten anderer Spieler.

These 4: Bauphysikalische Phänomene lassen sich durch qualitative Simulation darstellen.

Mit Hilfe der gewählten und teilweise implementierten Beispiele konnte gezeigt werden, dass sich die zu vermittelnden Phänomene der Bauphysik durch qualitative Simulation darstellen lassen. Unter Verzicht auf exakte numerische Ergebnisse wurden vereinfachte Berechnungsalgorithmen bzw. die in den einschlägigen Normen der Bauphysik vorgeschlagenen Berechnungsalgorithmen benutzt. Damit konnten Ergebnisse erzielt werden, deren Änderungen in Abhängigkeit von verschiedenen Parametern das korrekte Vorzeichen besaßen und in der korrekten Größenordnung lagen. Die für die Lerner wichtigen Systemzusammenhänge können mit Hilfe von vereinfachten, ressourcenschonenden Berechnungsverfahren ausreichend demonstriert werden.

These 5: Die gewählte Methodik lässt sich auch auf andere Ingenieursdisziplinen ausweiten.

Mit Hilfe der Wissensdomäne Bauphysik konnte die dargestellte Methodik entwickelt und getestet werden. An keiner Stelle wurden jedoch Annahmen getroffen, die nur durch die Bauphysik erfüllt werden. Damit ist die gewählte Methodik nicht auf die Domäne Bauphysik beschränkt, sondern kann in vielen anderen Ingenieursdisziplinen zur Anwendung kommen.

These 6: Lernziele sind eine geeignete Leitgröße für den Aufbau eines Szenarios.

Das Ziel einer Spielplattform, die in zur Ausbildungsbegleitung eingesetzt wird, ist neben der Freude beim Spielen der Lernerfolg. Lernziele geben die Fähigkeiten vor, die erlernt werden sollen. Lernen bedeutet eine dauerhafte Änderung der Verhaltensweise einer Person aufgrund von Erfahrungen. Um diese gewünschten Erfahrungen machen zu können, wird in den Szenarios eine maßgeschneiderte Umgebung erstellt. Das Lernziel⁸⁹ gibt den Aufbau des Szenarios vor. Aus der minimalen⁹⁰ Struktur des Szenarios ergibt sich ein auf das Wesentliche reduzierter Versuchsaufbau⁹¹. Abhängig von der Komplexität des Szenarios können noch weitere Modellelemente hinzugefügt werden. Diese zusätzlichen Elemente mögen lediglich der Motivation des Spielers dienen. Grundsätzlich gilt jedoch, dass das Lernziel ein geeignetes Simulationsmodell für das Szenario treibt.

These 7: Spielentwicklung ist eine multidisziplinäre Tätigkeit und besitzt daher eine erhöhte Komplexität.

Während der Entwicklung der Prototypen - insbesondere der Entwicklung von *Easy Gold* - wurde deutlich, dass die Entwicklung eines Spiels erhöhte Anforderungen an das Entwicklungsteam stellt - verglichen mit der Entwicklung einer herkömmlichen Anwendungssoftware. Die Konzep-

⁸⁹ Es ist auch möglich, in einem Szenario mehrere Lernziele zu verfolgen.

⁹⁰ Minimal in dem Sinne, dass jedes Entfernen von Objekten aus dem Szenario die gewünschte Erfahrung unmöglich machen würde.

⁹¹ Gee (2005) spricht vom Prinzip des „fish tank“.

tion eines Spiels verlangt nicht nur den Entwurf eines stimmigen Spielkonzepts - sondern für Serious Games auch die pädagogisch sinnvolle Integration der Fachinhalte. Das Spielkonzept ist algorithmisch nicht zu validieren - es sind Spieltests erforderlich (s. These 9). Zudem arbeitet ein Computerspiel mit Grafiken und Audioelementen. Für deren Erstellung sind künstlerische Fertigkeiten notwendig, ebenso wie für den vorher genannten Spielentwurf. Dieses steht in Kontrast zur normalen Softwareentwicklung, die in der Regel fast ausschließlich ingenieurwissenschaftliche Fähigkeiten verlangt. Da Personen, die sowohl gute künstlerische als auch ingenieurwissenschaftliche Fähigkeiten besitzen, sehr selten sind⁹², sind Entwicklungsteams aus Mitgliedern mit verschiedenen Tätigkeitsprofilen notwendig. Die Mitglieder müssen in der Lage sein, miteinander zu kommunizieren.

These 8: Die gezielt Entwicklung eines spannenden Serious Game in der Graduiertenausbildung stellt eine Herausforderung dar und ist derzeit nicht Stand der Technik.

In These 7 werden die Herausforderungen der Entwicklung eines Serious Games genannt. Diese führen dazu, dass es kaum Serious Games gibt, die gleichzeitig großen Spielspaß bieten. Die meisten Serious Games wirken wie „chocolate-covered broccoli“ (Laurel, 2001) - als nicht gelungene Kombination von Spielmechaniken und Lerninhalten. Die Folge ist, dass mit diesen Computerspielen weder das Spielen Spaß macht noch das Lernen gelingt. Zur Entwicklung guter und spannender Serious Games ist viel Talent der beteiligten Teammitglieder notwendig - dieser Prozess ist nicht unabhängig von den Personen reproduzierbar. Der künstlerische Anteil am Entwicklungsprozess ist gegenüber dem handwerklichen noch sehr hoch. Zur weiteren Verbreitung von Spielen im Lernprozess müssen Methoden entwickelt werden, die deren Entwicklung einfacher und kostengünstiger werden lassen. Die Spielentwicklungen im Bereich der Serious Games können in den seltensten Fällen durch Verkaufserlöse refinanziert werden.

These 9: Spielentwicklung verlangt zwingend Spieltests.

Die Attraktivität eines Spieles kann nicht algorithmisch geplant werden. Sicherlich kann ein Spiel aufgebaut werden aufgrund von Erfahrungen mit anderen Spielen. Damit ist jedoch noch nicht sichergestellt, dass ein Spiel funktioniert und Spieler es anziehend finden. Dieses muss in Entwicklungs-Iterationen mit Hilfe von Spieltests geprüft werden. Ergebnisse des Spieltests sind Aussagen, welche Spielmechaniken funktionieren und welche nicht. Diese sollten dann in der nächsten Entwicklungsiteration angepasst werden. Ansatzweise konnten diese Iterationen bei der Entwicklung von *Easy Gold* durchgeführt werden. Bei der Entwicklung von SNGs sind die Spieltests zum Teil in die Produktivphase ausgelagert - bei der kontinuierlichen Weiterentwicklung der Spiele kommen A/B-Tests zum Einsatz (Nutt, 2011).

These 10: Die Attraktivität eines Spiels hängt stark vom Typ des Spielers ab.

Ein Computerspiel ist nicht objektiv attraktiv - die Attraktivität eines Spiels ist eine subjektive Wertung des Spielers. Dieses ist analog zu Filmen und Belletristik zu sehen: Auch dort hat jeder Konsument Vorlieben für bestimmte Genres und Abneigungen gegenüber anderen. Um sich der Subjektivität methodisch zu nähern, werden die Vorlieben kategorisiert: Es werden Taxonomien von Spielertypen aufgestellt. Für Spiele bedeutet das, dass ein Spiel gewöhnlich nicht alle Menschen erreichen kann. Ein Computerspiel ist aufgrund seiner Wirkungsweise, die auf intrinsischer Motivation aufbaut, ein Lernmedium für den optionalen Einsatz. Gleichwohl kann es auch ver-

⁹² Abgesehen davon, dass die Arbeitsmenge in der Regel nicht von einer Person zu bewältigen ist.

pflichtend eingesetzt werden. Ein Weg, der Typabhängigkeit zu begegnen, ist unterschiedliche Spielmechaniken für möglichst viele Spielertypen in ein Spiel zu integrieren.

These 11: Social Network Games (SNGs) sind ein geeignetes Lernmedium

SNGs haben einige Eigenschaften, die ihre Eignung als Lernmedium unterstützen. Dazu gehört die leichte Zugänglichkeit. Im Gegensatz zu herkömmlichen Computerspielen muss keine Software installiert oder gar besondere Hardware benutzt werden, sondern es reicht, wenn der Benutzer in der Lage ist, einen Webbrowser zu bedienen. Gleichzeitig ist der zeitliche Kontext des Spiels ein anderer: Viele Spielmechaniken von SNGs ermöglichen kurze Spielzyklen⁹³.

Durch die Einbettung in Social Network Services ist zugleich ein sozialer Kontext gegeben: Bekannte, die dasselbe Spiel spielen, fördern die Bereitschaft des Spielers, sich im Spiel zu engagieren. Das ist eines der Prinzipien, die sich kommerzielle SNGs zunutze machen: Durch das Spiel werden immer wieder Mitteilungen und Anfragen erzeugt, die andere Spieler fortgesetzt dazu animieren, in das Spiel einzusteigen.

Die Herstellungskosten für ein SNG sind in der Regel wesentlich geringer als für herkömmliche kommerzielle Spiele. Dadurch werden die Möglichkeiten größer, ein solches Spiel zu realisieren.

Durch die systemimmanente Client-Server-Kommunikation ist es möglich, das Spielverhalten ständig zu beobachten. Damit kann das Spiel gegebenenfalls angepasst werden, um den Lernerfolg zu verbessern.

These 12: Es ist lohnend, attraktive kommerzielle Spiele auf Anpassungsmöglichkeiten zu untersuchen, um fachliche Lerninhalte zu transportieren.

Die große Komplexität der Erstellung von Lernspielen in Verbindung mit zum Teil bescheidenen Lernerfolgen lässt es sinnvoll erscheinen, Konzepte attraktiver Spiele auf Lernspiele zu übertragen bzw. kommerzielle Spiele zu untersuchen, ob und wie sie mit Fachinhalten angereichert werden können. Squire (2003) hat dieses mit Civilization demonstriert. Es gibt eine Reihe von Spielen, für die ein solches Vorgehen - im Kontext der Bauphysik - lohnenswert erscheint: Müller (2012) hat ein Konzept für die Anreicherung des SNG Fliplife mit bauphysikalischen Inhalten vorgelegt, die Anpassung des Tablet-Computer-Spiels „Jagd nach dem Katzenkönig“ auf andere als juristische Inhalte ist durch das Design bereits vorgesehen (Lernfreak GbR, 2012). Triviador ist ein bereits spielbares SNG, bei dem das Beantworten von Fragen nicht wie ein Fremdkörper wirkt, sondern ein Teil des Spiels ist (Nelson, 2011; THX_Games_PLC, 2011).

⁹³ Kurze Spielzyklen könnten problematisch sein, um eine gute Basis für die Vermittlung von komplexen Systemzusammenhängen zu bieten. Andererseits sind kurze Zyklen von Aktion und Rückmeldung auch das Merkmal von herkömmlichen, komplexeren Spielen.

Epilog

Mit dieser Arbeit liegt ein Konzept vor für die methodische Umsetzung des Prinzips des spielbasierten Lernens in der Graduiertenausbildung. Die Belastungsfähigkeit der Konzeption wurde mit Hilfe von prototypischen Implementierungen demonstriert. Für die weitere erfolgreiche Umsetzung ist zu beachten, dass Lernspiele von Spielern - wenn auch nur unterschwellig - mit kommerziellen, erfolgreichen Computerspielen verglichen werden. Als Konsequenz sinkt die Spielbereitschaft, wenn der Vergleich ungünstig ausfällt. Am ehesten sind Erfolge mit niedrigeren Entwicklungsbudgets zu erreichen, sofern sich dieser unbewusste Vergleich abschwächen lässt. Zu nennen sind hier SNGs wie Fliplife oder Mobile Device Games mit Angry Birds als Beispiel.

Größeres Potential sehe ich persönlich in der Anpassung von kommerziellen Spielen: Einige von ihnen lassen sich zumindest konzeptionell mit Fachinhalten ausstatten. Bei anderen ist dieses auch vom Design so vorgesehen und tatsächlich möglich - wie bei der „Jagd nach dem Katzenkönig“.

Generell ist es notwendig, die Erstellung von Spielen methodisch zu einem Handwerk zu machen - heute ist es zu einem großen Teil ein künstlerischer Prozess. In diesem Sinne ist es Voraussetzung, die Zielgruppe näher zu analysieren: Welche Spielertypen herrschen vor? Wie sieht die prozentuale Zusammensetzung der Spielerschaft aus? Welche Erwartungen haben diese? Wie lassen sie sich motivieren? Auf der Seite des Spieldesigns ist es notwendig, die Wirkung von Spielmechaniken genauer kennenzulernen. Welche Mechanismen führen zu welchen Effekten und Lernerfolgen? Mit Hilfe eines solchen Wissens lassen sich Kataloge aufstellen, die beim Entwurf eines Spiels angewendet werden können. Muster, mit denen die Unterbringung von fachlichen Inhalten in Spielen kategorisiert wird, sind ebenfalls hilfreich.

Neben diesen methodischen Basisaufgaben besteht für mich die Arbeit auf dem Gebiet der Lernspielentwicklung zu einem großen Teil auch aus einer aufmerksamen Grundhaltung, mit der wir uns Schritt für Schritt vorarbeiten können. Eine Grundhaltung, die Chancen erkennt, wann funktionierende Spiele mit wenig Aufwand zur Erfüllung konkreter Lernziele adaptiert werden können. Und natürlich wird immer ein großes Maß an Kreativität hilfreich sein, um vorhandene Mittel geschickt kombinieren zu können - es müssen nicht immer Spiele sein, auch Teilaspekte wie Gamification und Simulation erweisen sich als zielführend. Gleichzeitig bin ich mir bewusst, dass Spiele nur ein Hilfsmittel für die Lehre sein können - weitaus wichtiger ist der Einfluss einer Lehrperson oder Bezugsperson einzuschätzen. Kritisch zu beobachten sind eventuelle Abnutzungseffekte, pointiert ausgedrückt wird dieses durch die Frage „Wer kann schon immer Spaß vertragen?“. Der Zukunft von Spielen in der Ausbildung blicke ich - ausgestattet mit dem genannten, auch persönlichen, Aufgabenplan - optimistisch entgegen: Im Laufe der Zeit hat jedes Medium seinen Platz in der Lehre gefunden - ich habe bisher keinen Grund entdeckt, der das für Spiele ausschließt.

Tabellenverzeichnis

Tabelle 1: Auswahl von Softwarepaketen zur Simulation bauphysikalischer Probleme	24
Tabelle 2: Faktische Dimension der überarbeiteten Lernzieltaxonomie nach Bloom.....	26
Tabelle 3: Lernzieltaxonomie nach Anderson & Krathwohl	26
Tabelle 4: Konfigurationsinformationen der Extension "simuElement"	44
Tabelle 5: Simulationselemente im Prototyp	48
Tabelle 6: Parameter aus dem Prototyp	49
Tabelle 7: Im Prototyp implementierte Actions	49
Tabelle 8: Im Prototyp deklarierte Events	50
Tabelle 9: TimeManager: Attribute	55
Tabelle 10: Struktur der Beschreibung eines Szenarios.....	60
Tabelle 11: Beschreibung eines Kombinationsszenarios	64
Tabelle 12: Elemente der Notation zur Systemdarstellung	66
Tabelle 13: Kategorisierung von Variablen	67
Tabelle 14: Verwendete Pseudocode-Syntaxelemente.....	69
Tabelle 15: Verwendete Basiskonfigurationen	69
Tabelle 16: Übersicht der Szenarien	69
Tabelle 17: Lernziele der Beispielszenarien	70
Tabelle 18: Übersicht der Missionen	71
Tabelle 19: Größenverzeichnis Basiskonfiguration Einraumhaus [BK001]	73
Tabelle 20: Größenverzeichnis Basiskonfiguration Vierwohnungsbungalow [BK002]	75
Tabelle 21: Abhängigkeitsmatrix Basisszenario Wärme [B001].....	78
Tabelle 22: Größenverzeichnis Basisszenario Wärme [B001].....	78
Tabelle 23: Liste der auswählbaren Dämmmaterialien	82
Tabelle 24: Wärmeleitfähigkeit verschiedener Baumaterialien	82
Tabelle 25: Abhängigkeitsmatrix Ergänzungsszenario Transmissionswärmeverluste [E001]	83
Tabelle 26: Größenverzeichnis Ergänzungsszenario Transmissionswärmeverluste [E001]	84
Tabelle 27: Größenverzeichnis Ergänzungsszenario Solare Wärmegewinne [E002].....	89
Tabelle 28: Abhängigkeitsmatrix Ergänzungsszenario Solare Wärmegewinne [E002].....	89
Tabelle 29: Abminderungsfaktor: Himmelsrichtung.....	90
Tabelle 30: Bewertetes Schalldämmmaß und das Durchhören von Sprache bzw. Geräten	93
Tabelle 31: Abhängigkeitsmatrix Basisszenario Akustik [B002].....	94
Tabelle 32: Größenverzeichnis Basisszenario Akustik [B002].....	94
Tabelle 33: Dichte verschiedener Baumaterialien	97
Tabelle 34: Abhängigkeitsmatrix Ergänzungsszenario Akustik [E003].....	98
Tabelle 35: Größenverzeichnis Ergänzungsszenario Akustik [E003].....	98
Tabelle 36: Abhängigkeitsmatrix Ergänzungsszenario Akustik [E004].....	102
Tabelle 37: Größenverzeichnis Ergänzungsszenario Akustik [E004].....	103
Tabelle 38: Dichte und Wärmeleitfähigkeit verschiedener Baumaterialien.....	105

Abbildungsverzeichnis

Abbildung 1: Künstlerisch gestaltete FarmVille-Welt.....	17
Abbildung 2: Anzeigen der Rangliste nach Lösung eines Puzzles in EteRNA	22
Abbildung 3: CDT: Matrix von Inhaltstyp und kognitivem Anforderungsniveau	27
Abbildung 4: Architektur des Systems.....	31
Abbildung 5: Server-Komponenten.....	31
Abbildung 6: Modulbaum nach Modulklassen geordnet.....	41
Abbildung 7: Extension Point Editor.....	44
Abbildung 8: Hierarchisches Modell von Simulationselementen	48
Abbildung 9: Repräsentation der Actions im Kontextmenü.....	50
Abbildung 10: Klassendiagramm Event Manager	56
Abbildung 11: Extension Point systemEvent.....	56
Abbildung 12: Klassendiagramm Szenario und Konfiguration	62
Abbildung 13: Grundrisskizze Vierwohnungsbungalow [BK002].....	74
Abbildung 14: Wirkungsgeflecht Basisszenario Wärme [B001]	77
Abbildung 15: Wirkungsgeflecht Erweiterungsszenario Wärmedämmung [E001].....	83
Abbildung 16: Wirkungsgeflecht Ergänzungsszenario Solare Warmegewinne [E002]	88
Abbildung 17: Wirkungsgeflecht Basisszenario Akustik [B002]	93
Abbildung 18: Wirkungsgeflecht Ergänzungsszenario Akustik [E003]	97
Abbildung 19: Wirkungsgeflecht Ergänzungsszenario Akustik [E004]	101

Formelverzeichnis

Formel 1: Wärmeverluste	79
Formel 2: U-Wert mit Hilfe von Wärmedurchlasswiderständen	84
Formel 3: Solare Wärmegewinne	89
Formel 4: Solare Wärmeeinstrahlung	90
Formel 5: Gesamtschalldämmmaß einer Wand	94
Formel 6: Schallpegel im Raum.....	94
Formel 7: Flächenbezogene Masse einer Wand.....	98
Formel 8: Bewertetes Schalldämmmaß für einschalige Wände.....	98
Formel 9: Schallpegel im Raum.....	99
Formel 10: Schallpegel Raum.....	103

Verzeichnis der Quellcodeauszüge

Quellcode 1: plugin.xml: Extension Simulationselement Radiator	44
Quellcode 2: Auszug der Datei plugin.xml: Definition eines zeitorientierten Events	57
Quellcode 3: Klasse SystemEventJob: Schleife zur Eventverarbeitung.....	58

Verzeichnis der Pseudocodedarstellungen

Pseudocode 1: Heizungswärmestrom	79
Pseudocode 2: Temperatur im Innenraum.....	79

Abkürzungen

Abkürzung	Bedeutung
AF	Abminderungsfaktor
API	Application Programming Interface (Programmierschnittstelle einer Software)
CDT	Component Display Theory, eine Theorie des Instructional Designs von M. David Merrill.
CLR	Common Language Runtime, Laufzeitumgebung für .NET-Programme.
COTS	Commercial off-the-shelf, in großer Stückzahl gefertigte Produkte, hier z.B. handelsübliche Computerspiele.
DEVS	Discrete Events System Specification, ein Beschreibungsformalismus für Simulationsmodelle
DLL	Dynamic Link Library, Binärdatei mit ausführbarem Code unter dem Betriebssystem MS Windows.
dt..	deutsch
DV	Datenverarbeitung
EnEV	Energieeinsparverordnung, diese beschreibt bautechnische Anforderungen zum effizienten Betriebsenergieverbrauch von Gebäuden.
engl.	Englisch
HLA	High Level Architecture, Standard (IEEE 1516) im Bereich der verteilten Simulation
ID	Instructional Design (s. Glossar)
IDE	Integrated Development Environment (Integrierte Entwicklungsumgebung)
IT	Information Technology
JVM	Java Virtual Machine, Laufzeitumgebung für Java-Programme
K-12	Bezeichnung für die formale Bildung vom Kindergarten bis zur 12. Klasse in den USA (s.Glossar:K-12)
M&S	Modeling and Simulation
MMOG	Massively Multiplayer Online Game: Genre eines Computerspiels, das mit Hilfe des Internets gespielt wird und seinen Spielern eine persistente virtuelle Welt bietet. Derartige Spiele sind auf eine möglichst große Anzahl von Spielern ausgelegt, die sich gleichzeitig in der virtuellen Welt aufhalten können. Beispiel eines derartigen Spiels ist <i>World of Warcraft</i> .
MS	Microsoft, Softwarehersteller
MUD	Multi User Dungeon, ein textbasiertes digitales Rollenspiel.
NPC	Non-Player Character: Das sind Figuren in Computerspielen, die nicht durch einen Spieler, sondern vom Computerspiel selbst kontrolliert werden.
OSGi	Open Services Gateway Initiative, Standard eines Komponentenmodells der OSGi Alliance für Java.
Q&A	Question and Answer in Bezug auf webbasierte Software, die das Stellen von Fragen und deren gemeinschaftliche Beantwortung unterstützt.
s.	siehe
S.	Seite
SFX	Sound Effekts: (Künstliche) Geräusche, die u.a. in Computerspielen eingesetzt werden.
SNG	Social Network Game, auch Social Game genannt: Computerspiel auf Basis eines SNS
SNS	Social Networking Service, Plattform für Social Networks, Beispiele sind Facebook und LinkedIn.
SQL	Structured Query Language, eine Sprache zum Umgang mit Datenbanken.
SUI	system under investigation (s. SUS)
SUS	system under study: Das mit Hilfe einer Simulation betrachtete System der realen Welt

Abkürzungen

u.a.	unter anderem
UI	user interface: Benutzerschnittstelle (hier: Bedienoberfläche eines Programmes)
VV&A	verification, validation, and accreditation: Vorgehensmodell zur Qualitätssicherung von Simulationen.

Glossar

Abminderungsfaktor	Abminderungsfaktoren werden im Sinne von Schätz-faktoren, die beispielsweise durch technische Normen vorgegeben werden, in ingenieurmäßigen Berechnungen eingesetzt, wenn eine exakte Berechnung zu aufwändig ist. Beispiel für Abminderungsfaktoren in dieser Arbeit sind die zur Berechnung der solaren Wärmegewinne.
Badge	Innerhalb eines Computerspiels Bezeichnung für eine Auszeichnung für eine erbrachte Leistung. Ein Badge ist eine Art virtueller Orden.
Component Display Theory	Ein Theorie des Instructional Designs, die von M. David Merrill entwickelt wurde.
EVE Online	EVE Online ist ein MMOG des isländischen Spielehertellers CCP Games. Schwerpunkte dieser Weltraumsimulation sind Rohstoffabbau, Veredelung und Produktion sowie Handel und Kampf.
Extension Point	Begriff aus der Plug-in-Entwicklung mit der IDE Eclipse: ein Extension Point beschreibt eine Erweiterungsmöglichkeit des definierenden Softwaresystems, er wird genutzt durch Extensions (s. Extension)
Extension	Begriff aus der Plug-in-Entwicklung mit der IDE Eclipse: Eine Extension ist eine Erweiterung eines bestehenden Softwaresystems, sie nutzt dabei einen Extension Point (s. Extension Point)
Free-to-play	Vertriebsmodell für Spiele: Die Spiele können zunächst kostenlos gespielt werden. Der Hersteller finanziert sich über andere Wege, beispielsweise den Verkauf von zusätzlichen Spielinhalten oder die Einblendung von Werbung.
Game Engine	Laufzeitumgebung für Computerspiele; oft bieten Game Engines spezielle Unterstützung für grafische Effekte bzw. für die Simulation von physikalischen Phänomenen wie Licht, Schwerkraft und Schall an.
g-Wert	Begriff aus der Bauphysik: Der Gesamtennergiedurchlassgrad ist ein Maß für die Energiedurchlässigkeit von transparenten Bauteilen, wie z.B. Fenster. Ein Energiedurchlassgrad von 0,7 bedeutet, dass 70% der einstrahlten Energie durch das Bauteil hindurchgehen.
Hot Deployment	Technik, mit der Serveranwendungen ganz oder teilweise im laufenden Betrieb aktualisiert oder installiert werden können.
Identifikator	Merkmal zur eindeutigen Identifizierung eines Objekts.
Instructional Design	Unter Instructional Design (ID) wird ein methodisches Vorgehen zur Erstellung von Ausbildungseinheiten verstanden. Ziel für die entstehenden Lernmodule ist eine Steigerung des Lernerfolges - sei es durch schnelleres Lernen oder die Erreichung eines höheren Verständnisgrades.
Integrierte Entwicklungsumgebung	Eine Software, in der möglichst viele Arbeiten zur Ent-

	wicklung von Software unter einer gemeinsamen Oberfläche durchgeführt werden können. Zu den wesentlichen Bestandteilen einer Integrierten Entwicklungsumgebung sind Editor, Compiler und Debugger. Eclipse gehört zu den führenden Java IDEs.
Inventory	Innerhalb eines Computerspiels: der spielerbezogene Bestand an spielrelevanten Gegenständen.
K-12	In den USA ist K-12 eine zusammenfassende Bezeichnung für die formale Bildung vom Kindergarten bis zur 12. Schulklasse.
Latenzzeit	Die Zeit zwischen Aktion und Reaktion, bei Online-Computerspielen die Zeit zwischen Absenden einer Anfrage (engl. request) und dem Eintreffen der Antwort (engl. response). Für Online-Computerspiele ist eine niedrige Latenzzeit wichtig, um Interaktionen in Echtzeit zu erlauben.
Listenfeld	In einer grafischen Benutzeroberfläche ein Eingabefeld, das mehrere auswählbare Einträge bereithält. Der Benutzer kann einen dieser Einträge als Wert des Eingabefeldes auswählen.
Mission	Innerhalb eines Computerspiels ist Mission die Bezeichnung für die Aufgabe, die ein Spieler im Spiel erhält und deren Erfüllung belohnt wird.
Monitoring	(dt. Überwachung, Kontrolle) andauernde, systematische Darstellung des Status eines Systems.
Partikelemitter	Ein Partikelemitter (auch Partikelsystem, engl. particle emitter) ist ein (grafischer) Animationseffekt, bei dem eine große Anzahl von kleinen Objekten von einer Quelle ausgestoßen werden und einen konfigurierbaren Weg durch den Raum nehmen. Beispiele sind die Darstellung von Rauch und Explosionen (auch Partikelsystem genannt).
Reward	Innerhalb eines Computerspiels bezeichnet Reward die Belohnung, die ein Spieler für die Lösung einer Aufgabe erhält.
Shooter-Spiele	Gehören zur Kategorie der Action-Spiele, bei der gewöhnlich mit einer Waffe Gegner oder andere Ziele bekämpft werden. Das erfolgreiche Spielen fordert in hohem Maße Reaktions- und Aktionsgeschwindigkeit vom Spieler.
SimuFrame	Arbeitsname des im Rahmen dieser Arbeit als Serverkomponente erstellten Simulationskerns
Simulation Engine	Laufzeitumgebung für Simulationsläufe
Simulation Executive	S. simulation engine
Simulationszeit	In der dynamischen Simulation werden zwei Zeiten unterschieden: Simulationszeit und simulierte Zeit. Simulationszeit ist die echte Zeit, zu der die Simulation gerade läuft, simulierte Zeit der Wert der Zeitvariablen in der Simulation selbst.
Simulierte Zeit	s. Simulationszeit
Skill	Innerhalb eines Computerspiels Bezeichnung für eine

	(virtuelle) Fähigkeit.
Social Network Game (SNG)	Digitales Spiel, das zwischen den Mitgliedern eines Social Networking Service gespielt wird. Wesentliches Merkmal eines solchen Spiels ist die Asynchronizität, d.h. die Spieler müssen nicht gleichzeitig online sein. Eines der bekanntesten Beispiele für ein solches Spiel ist FarmVille der Firma Zynga, das im Juni 2009 gestartet wurde und in der Spitze über 80 Millionen Spieler hatte.
Social Networking Service (SNS)	Webplattform, auf der sich die Mitglieder untereinander verknüpfen und Gemeinschaften bilden können. Einer der größten und bekanntesten Social Networking Services ist Facebook (www.facebook.com).
Software as a Service (SaaS)	Ein Ansatz der Softwarebereitstellung, bei der die Software über ein Kommunikationsnetz bei Bedarf zur Verfügung gestellt wird. Die Verarbeitung erfolgt gewöhnlich auf Rechner des Softwareanbieters.
Spiel-Engine	s. Game Engine
Spielmechanik	Hier: Satz von Regeln in einem Computerspiel, die einen (Eingabe-)Status eines Computerspiels und eine Folge von Aktionen des Spielers in einen (Ausgabe-)Status transferieren. Ein Computerspiel enthält gewöhnlich mehrere Spielmechaniken.
Template-Szenario	Jedes Szenario der Plattform SimuFrame liegt als prototypisches Objektgeflecht vor, das kopiert wird, um daraus lauffähige Simulationsszenarien zu erzeugen. Das prototypische Objektgeflecht wird Template-Szenario genannt.
Test Reference Year	Test Reference Year (TRY) ist eine Datenstruktur, in der die Werte der klimatischen Parameter für eine Klimaregion und ein Jahr festgehalten werden. Die Daten werden in der Regel für Simulationen genutzt.
Tooltip	In Bereich der grafischen Benutzungsoberfläche von Rechnerprogrammen ein kurzer beschreibender Text, der beim Überfahren eines Oberflächenelementes mit einem grafischen Eingabegerät (z.B. Maus) angezeigt wird.
U-Wert	Begriff aus der Bauphysik: Auch Wärmedurchgangskoeffizient genannt, ist der U-Wert ein Maß für den Wärmestrom durch ein Bauteil hindurch. Je kleiner der U-Wert eines Bauteiles ist, desto besser ist die Dämmeigenschaft dieses Bauteiles. Die Einheit des U-Werts ist $W/(m^2 \cdot K)$ (Watt pro Quadratmeter und Kelvin).
Versionsverwaltung	In der Softwareentwicklung ein System, dass zur Protokollierung der Änderungen an Dateien dient. Mit Hilfe der Versionsverwaltung kann der Stand eines Systems an Dateien für einen beliebigen Zeitpunkt wiederhergestellt werden. Bekannt Versionsverwaltung in der Softwareentwicklung sind Subversion, CVS und Git.
Wärmeübergangskoeffizient	Wärmemenge (Einheit: Ws), die pro Zeiteinheit (s)

	zwischen einer Einheit der Oberfläche (m^2) eines festen Stoffes und der ihn berührenden Luft ausgetauscht wird.
Wärmeübergangswiderstand	Der Wärmeübergangswiderstand R_s ist der Kehrwert des Wärmeübergangskoeffizienten h . Es wird unterschieden zwischen den Wärmeübergangswiderständen an der Außen- und der Innenseite einer Wand (R_{si} und R_{se}).
XP	Experience Points (Erfahrungspunkte), oft eine Art der Belohnung innerhalb eines Computerspiels.
Zustandsvariable	Größe in einem Simulationsmodell, die einen Wert annehmen kann.

Übersetzungen

Ausdruck	Übersetzung
achievements	Die Errungenschaften eines Spielers in einem Spiel, z.B. Auszeichnungen (Badges) und Fähigkeiten (Skills)
badge	Auszeichnung (s. Glossar: Badge)
experience point	Erfahrungspunkt
extension	Erweiterung (s. Glossar: Extension)
extension point	Erweiterungspunkt (s. Glossar: Extension Point)
game engine	Spiel-Engine (s. Glossar: Game Engine)
hot deployment	Aktivierung von Software im laufenden Betrieb (s. Glossar: Hot Deployment)
Instructional design	Instruktionsdesign
inventory	Warenbestand, Lager (s.: Glossar: Inventory)
mechanic (of a simulation engine)	eigentlich Mechanik Wirkungsweise oder Funktionsweise
reward	Belohnung (s. Glossar: Reward)
simulation engine	Simulation Engine; Laufzeitumgebung für Simulationsläufe
simulation executive	s. simulation engine

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Bei der Auswahl und Auswertung folgenden Materials haben mir die nachstehend aufgeführten Personen in der jeweils beschriebenen Weise entgeltlich/unentgeltlich geholfen:

1. Janina Held: Unterstützung bei der Programmierung von Server- und Clientfunktionalität (Hilfswissenschaftliche Tätigkeit)
2. Anna Beyer: Unterstützung bei der Programmierung von Serverfunktionalität (Hilfswissenschaftliche Tätigkeit)
3. Adriana Cabrera: Unterstützung beim Entwurf von Grafiken für Easy Gold bzw. BuildVille (Hilfswissenschaftliche Tätigkeit)
4. Philipp Blaschke: Unterstützung bei der Recherche zur Entwicklung bauphysikalischer Szenarien (Hilfswissenschaftliche Tätigkeit)
5. Daniel Schwarz: Unterstützung bei der Erstellung und Auswahl der in Easy Gold bzw. BuildVille verwendeten Audiodateien (Hilfswissenschaftliche Tätigkeit)

Weitere Personen waren an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt. Ich versichere ehrenwörtlich, dass ich nach bestem Wissen die reine Wahrheit gesagt und nichts verschwiegen habe.

Weimar, 28.03.2013

Referenzen

- (Blocon). (2012). HEAT3. Lund, Sweden: BLOCON SWEDEN.
- (CD-adpco). (2012). Star-CCM+.
- (DataKustik). (2012). BASTIAN. Greifenberg: DataKustik GmbH.
- (DWD). (2012). Wettervorhersage Messen - Berechnen - Interpretieren. Retrieved March 20, 2012, from http://www.dwd.de/bvbw/generator/DWDWWW/Content/Presse/Broschueren/Wettervorhersage__PDF,templateId=raw,property=publicationFile.pdf/Wettervorhersage__PDF.pdf
- (IBP). (2012). WUFI PRO, 2D, PLUS. Stuttgart: Das Fraunhofer-Institut für Bauphysik IBP.
- (M&SCO). (2012). VV&A Recommended Practices Guide. Retrieved March 16, 2012, from <http://vva.msco.mil/Default.htm>
- (USC University of Southern California). (2012). The Network Simulator ns-2: Validation Tests. *Information Science Institute*. Retrieved March 7, 2012, from <http://www.isi.edu/nsnam/ns/ns-tests.html>
- Abbott, M. (2010). Portal on the booklist. Retrieved January 18, 2011, from http://www.brainygamer.com/the_brainy_gamer/2010/08/portal-booklist.html
- Achtman, R. L., Green, C. S., & Bavelier, D. (2008). Video games as a tool to train visual skills. *Restorative Neurology and Neuroscience*, 26(4-5), 435–446.
- Adams, P. C. (1998, March). Teaching and Learning with SimCity 2000. *Journal of Geography*. doi:10.1080/00221349808978827
- Alexander, C. (1978). *The Timeless Way of Building* (p. 552). Oxford University Press.
- Ambler, S. W. (2010). 2010 IT Project Success Rates. *Dr.Dobbs*. Retrieved May 10, 2012, from <http://www.drdobbs.com/architecture-and-design/2010-it-project-success-rates/226500046#>
- Andersen, E., Liu, Y.-E., Snider, R., Szeto, R., Cooper, S., & Popovic, Z. (2011). On the harmfulness of secondary game objectives. *FDG '11 Proceedings of the 6th International Conference on Foundations of Digital Games* (pp. 30–37). ACM New York, NY, USA. doi:10.1145/2159365.2159370
- Anderson, L. W., & Krathwohl, D. R. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. (L. W. Anderson & D. R. Krathwohl, Eds.) *Theory Into Practice* (Vol. Complete e, p. xxix, 352 p.). Longman. doi:10.1207/s15430421tip4104_2
- Anderson, S. (2012). Just One More Game ... *The New York Times Magazine*. Retrieved from http://www.nytimes.com/2012/04/08/magazine/angry-birds-farmville-and-other-hyperaddictive-stupid-games.html?_r=2
- Anne. (2012). Gerücht: Rovio bringt Level-Editor für Angry Birds. *Pocket Gamer*. Retrieved August 7, 2012, from <http://de.pocketgamer.com/content/geruecht-rovio-bringt-level-editor-fuer-angry-birds>
- Apache. (2012). The Apache HTTP Server Project. Retrieved April 4, 2012, from <http://httpd.apache.org/>
- Appdata. (2010). Facebook Application Metrics. *Appdata, Independent, Accurate Application Metrics and Trends from Inside Network*. Retrieved December 3, 2010, from <http://www.appdata.com/leaderboard/>
- Arthorne, J., & Laffra, C. (2004). *Official Eclipse 3.0 FAQs*. (E. Gamma, L. Nackman, & J. Wiegand, Eds.) (p. 386). Addison Wesley.
- Atherton, J. (2011). Deep and surface learning. *Learning and Teaching*. Retrieved October 29, 2012, from <http://www.learningandteaching.info/learning/deepsurf.htm>
- Aubel, M. (2012). *Evaluation von NetLogo zur Abbildung bauphysikalischer Szenarien (Bachelorarbeit am Lehrstuhl Bauphysik, betr. v. O. Kornadt, H. Söbke)*. Bauhaus-Universität Weimar.

- Baker, L. B. (2011). Factbox: A look at the \$65 billion video games industry. *Reuters*. Retrieved September 10, 2012, from <http://uk.reuters.com/article/2011/06/06/us-videogames-factbox-idUKTRE75552I20110606>
- Banks, C. M. (2009). What Is Modeling and Simulation? In J. A. Sokolowski & C. M. Banks (Eds.), *Principles of Modeling and Simulation A Multidisciplinary Approach* (pp. 3–23). Hoboken, New Jersey: John Wiley & Sons.
- Banks, J. (1998). Principles of Simulation. In J. Banks (Ed.), *Handbook of Simulation Principles, Methodology, Advances, Applications and Practice* (pp. 3–30). John Wiley & Sons.
- Bartle, R. A. (1996). Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs. *The Journal of Virtual Environments*. Retrieved from <http://www.citeulike.org/user/drakkos69/article/3474752>
- Bartle, R. A. (2003). *Designing Virtual Worlds* (p. 741). New Riders.
- Baumgartner, Peer. (2011). *Taxonomie von Unterrichtsmethoden: Ein Plädoyer für didaktische Vielfalt* (p. 376). Waxmann.
- Baumgartner, Peter, Maier-Häfele, K., & Häfele, H. (2004). *Content Management Systeme in e-Education* (p. 480). Studienverlag GmbH.
- Beaton, W., Helming, J., & Kögel, M. (2011). Was ist Eclipse? *jaxenter*. Retrieved April 5, 2012, from <http://it-republik.de/jaxenter/artikel/Was-ist-Eclipse-4007.html>
- Becker, K., & Parker, J. R. (2011). *The Guide to Computer Simulations and Games* (1st ed., p. 446). Indianapolis: John Wiley & Sons.
- Bernt, A., Bogoslovsky, V., Cziesielski, E., Dreyer, J., Ehlbeck, J., Ertel, H., Filippi, M., et al. (1999). Memorandum: Lehrinhalt Bauphysik für Universitäten und wissenschaftliche Hochschulen. *Bauphysik*, 21(4), 171–175.
- Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-ricci, M., & Rumble, M. (2012). Defining Twenty-First Century Skills. In P. Griffin, B. McGaw, & E. Care (Eds.), *Assessment and Teaching of 21st Century Skills* (pp. 17–66). Springer Netherlands. doi:10.1007/978-94-007-2324-5
- Birta, L. G., & Arbez, G. (2007). *Modelling and Simulation - Exploring Dynamic System Behaviour* (p. 454). Springer Science & Business Media.
- Björk, S., & Holopainen, J. (2005). *Patterns in Game Design. ISBN1584503548* (Vol. 54, p. 423). Charles River Media. doi:10.1.1.10.4097
- Bläsi, W. (2002). *Bauphysik* (5. Auflage., p. 304). Europa-Lehrmittel.
- Bloom, B. S. (1956). *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain*. (Benjamin S Bloom, Ed.) New York (Vol. 16, p. 207). Longmans.
- Boeing, A., & Bräunl, T. (2007). Evaluation of real-time physics simulation systems. *GRAPHITE '07 Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia* (Vol. 1, pp. 281–288). doi:10.1145/1321261.1321312
- Bogost, I. (2004). Asynchronous multiplayer: Futures for casual multiplayer experience. In M. Sicart & J. H. Smith (Eds.), *Proceedings of Other Players Conference 2004, Copenhagen, Denmark*.
- Bogost, I. (2011a). *How to Do Things with Videogames (Electronic Mediations)* (p. 192). University of Minnesota Press.
- Bogost, I. (2011b). Persuasive Games: Exploitationware. *Gamasutra*. Retrieved July 8, 2012, from http://www.gamasutra.com/view/feature/6366/persuasive_games_exploitationware.php
- Bohannon, J. (2008). The Gonzo Scientist. Slaying monsters for science. *Science*, 320(5883), 1592.
- Bonabeau, E. (2002). Agent-based modeling: methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99 Suppl 3, 7280–7. doi:10.1073/pnas.082080899
- Borowski, M. (2012). *Spielprinzipien in Nicht-Spiel-Kontexten (Bachelorarbeit am Lehrstuhl Bauphysik, betr. v. O. Kornadt, H. Söbke)*. Bauhaus-Universität Weimar.

- Bossel, H. (1994). *Modellbildung und Simulation: Konzepte, Verfahren und Modelle zum Verhalten dynamischer Systeme ; ein Lehr- und Arbeitsbuch* (p. 402). Vieweg.
- Brannolte, U., Griesbach, W., Harder, R., & Kraus, T. (2000). *Aktualisierung und Erweiterung von Planspielansätzen im Verkehrswesen im Hinblick auf die Erstellung von Mobilitätsspielen. Schlussbericht* (p. 49). Weimar.
- Breslau, R. (2011). The 10 Best Moments in Pro-Gaming History. *kotaku*. Retrieved November 11, 2011, from <http://kotaku.com/5820907/the-10-best-moments-in-pro-gaming-history>
- Bröker, T. (2013). *Authentische Lernkontexte in Simulationspielen (Dissertation in Bearbeitung)*. Bauhaus-Universität Weimar.
- Bröker, T., & Kornadt, O. (2009). Purposeful Problem Generation in Simulation Games - An approach to extend the target group of complex simulation games in engineering education. *3rd European Conference on Games Based Learning*.
- Bröker, T., Söbke, H., & Kornadt, O. (2011). Close the gap — Obstacles and solutions for the missing educational games in graduate education. *5th European Conference on Games Based Learning* (pp. 74–80).
- Brown, J. S., & Duguid, P. (2000). Balancing Act: Capturing Knowledge Without Killing It. *Harvard Business Review*, 78(3), 73–80, 212.
- Bundesministerium für Verkehr, Bau- und Stadtentwicklung. (2011). *TRY-Handbuch. Forschungsinitiative Zukunft Bau* (p. 74).
- Bungartz, H.-J., Zimmer, S., Buchholz, M., & Pflüger, D. (2009). *Modellbildung und Simulation: Eine anwendungsorientierte Einführung. Management* (Vol. 0, p. 429). Berlin Heidelberg: Springer.
- Burton, M. (2003). Scorched Parabolas: A History of the Artillery Game. *Armchair Arcade*. Retrieved August 7, 2012, from <http://www.armchairarcade.com/aamain/content.php?article.51>
- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. (J. W. Sons, Ed.)Wiley (Vol. Vol. 1, p. 476). Wiley.
- Campbell-Kelly, M. (2007). Number Crunching without Programming: The Evolution of Spreadsheet Usability. *IEEE Annals of the History of Computing*, 29(3), 6–19. doi:10.1109/MAHC.2007.43
- Carlin, S. (2006). *Schlagwortvergabe durch Nutzende (Tagging) als Hilfsmittel zur Suche im Web. Ansatz, Modelle, Realisierungen*. Hochschule Darmstadt.
- Carron, T., & Marty, J. (2011). Using Profiling to Optimise a Collaborative Session in a Learning Game. In D. Gouscas & M. Meimaris (Eds.), *ECGBL 2011 - 5th European Conference on Games Based Learning* (pp. 88–97). Athens, Greece: Academic Publishing International. doi:ISBN 978-1-908272-19-5 CD
- Cassandras, C. G., & Lafortune, S. (2007). *Introduction to discrete event systems* (Vol. 11, p. 776). Springer Science & Business Media.
- Castronova, E. (2006). On the Research Value of Large Games Natural Experiments in Norrath and Camelot. *Games and Culture*.
- Castronova, Edward. (2001). Virtual worlds: A first-hand account of market and society on the cyberian frontier. *CESifo Working Paper Series No. 618*. Retrieved November 8, 2012, from <http://ssrn.com/abstract=294828>
- CCP. (2012). EVE Online. *EVE Online*. Retrieved June 19, 2012, from <http://www.eveonline.com/CCPGames>.
- CCPGames. (2007). EVE ONLINE ernennt In-Game-Ökonom. *openPR*. Retrieved August 20, 2012, from <http://www.openpr.de/news/143278/EVE-ONLINE-ernennt-In-Game-Oekonom.html>
- Cederholm, H., Hilborn, O., Lindley, C., & Sennersten, C. (2011). The Aiming Game : Using a Game with Biofeedback for Training in Emotion Regulation. In A. Copier, M. Kennedy, & H. Waern (Eds.), *Think Design Play: The fifth international conference of the Digital Research Association (DIGRA)*. DiGRA/Utrecht School of the Arts.

- Checkland, P. (1999). *Systems Thinking, Systems Practice: includes a 30-year retrospective. System* (Vol. 1, p. 416). John Wiley and Sons Ltd.
- Chung, C. A. (2004). *Simulation modeling handbook : a practical approach*. (H. P. R, Ed.) New York (p. 574). CRC Press.
- Cifaldi, F. (2012). Breaking down SimCity's Glassbox engine. *Gamasutra*. Retrieved August 9, 2012, from http://www.gamasutra.com/view/news/164870/gdc_2012_breaking_down_simcitys_.php
- Colwell, R. R. (1999). Complexity and Connectivity: A New Cartography for Science and Engineering. *Remarks from the American Geophysical Union 's fall meeting*. San Francisco, CA.
- Corbett, S. (2010). Learning by Playing: Video Games in the Classroom. *The New York Times Magazine*. Retrieved October 12, 2010, from <http://www.nytimes.com/2010/09/19/magazine/19video-t.html>
- CQ. (2012). Chicago Quest Schools. CQ. Retrieved October 10, 2012, from <http://www.chicagoquest.org/about-us/quest-to-learn.html>
- Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience*. (H. Collins, Ed.) *Annals of Physics* (Vol. 54, p. 303). Harper & Row. doi:10.1145/1077246.1077253
- Cummings, J. (2011). To Bartle or Not to Bartle?: Capturing Player Behavior in Social Games. *Motivate Play*. Retrieved June 4, 2012, from <http://www.motivateplay.com/2011/06/to-bartle-or-not-to-bartle-capturing-player-behavior-in-social-games/>
- Dahl, O.-J., Dijkstra, E. W., & Hoare, C. A. R. (1972). *Structured programming* (p. 220). New York, New York, USA: Academia Press Inc.
- Deen, M., & Schouten, B. (2011). Games that Motivate to Learn: Design Serious Games by Identified Regulations. In P. Felicia (Ed.), *Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches* (pp. 330–351). IGI Global. doi:10.4018/978-1-60960-495-0
- DeLoura, M. (2009). The Engine Survey : General results. *Gamasutra: Mark DeLoura's Blog*. Retrieved March 28, 2012, from http://www.gamasutra.com/blogs/MarkDeLoura/20090302/581/The_Engine_Survey_General_results.php
- DeLoura, M. (2011). Game Engine Survey 2011. *Game Developer Magazine*, (5), 7–12.
- Deterding, S. (2011). A Quick Buck by Copy and Paste. *Gamification Research Network*. Retrieved June 12, 2012, from <http://gamification-research.org/2011/09/a-quick-buck-by-copy-and-paste/>
- Dormans, J. (2012). *Engineering emergence: applied theory for game design*. Universiteit van Amsterdam.
- Dörner, D. (1981). Über die Schwierigkeiten menschlichen Umgangs mit Komplexität. *Psychologische Rundschau*, XXXI(2), 163–179.
- Dörner, D. (2003). *Die Logik des Mißlingens. Strategisches Denken in komplexen Situationen*. (p. 352). Reinbek bei Hamburg: Rowohlt.
- Dörner, D. (2010). "Killerspiele" und Gewalt. *Clash of Realities -3rd International Computer Game Conference Cologne*.
- Dr.EyjoG, C. (2012). What is going on with mineral prices? *EVE Insider Dev Blog*. Retrieved August 20, 2012, from <http://community.eveonline.com/devblog.asp?a=blog&nbid=28612>
- Dworatschek, S. (1989). *Grundlagen der Datenverarbeitung* (8. Auflage., p. 607). Berlin New York: Walter de Gruyter.
- Ebbinghaus, H. (1992). *Über das Gedächtnis: Untersuchungen zur experimentellen Psychologie. Neue unveränderte und ungekürzte Ausgabe nach der 1. Auflage 1885* (p. 109). Wissenschaftliche Buchgesellschaft.
- Eclipse. (2012). About the Eclipse Foundation. *Eclipse Foundation*. Retrieved April 5, 2012, from <http://www.eclipse.org/org/>

- Eclipse CDT. (2012). Retrieved April 5, 2012, from <http://www.eclipse.org/cdt/>
- Eclipse Mylyn Open Source Project. (2012). Retrieved April 5, 2012, from <http://www.eclipse.org/mylyn/index.php>
- Eclipse.org. (2011). Using Version Qualifiers. *Help Eclipse Platform*. Retrieved September 5, 2011, from http://help.eclipse.org/helios/index.jsp?topic=/org.eclipse.pde.doc.user/tasks/pde_version_qualifiers.htm
- EclipseLink. (2012). EclipseLink Home. *Eclipse.org*. Retrieved May 9, 2012, from <http://eclipse.org/eclipselink/>
- Egenfeldt-Nielsen, S. (2007). *Educational Potential of Computer Games (Continuum Studies in Education)* (p. 232). Continuum.
- Eickelmann, N. (1999). ACM Fellow Profile: David Lorge Parnas. *Software Engineering Notes*, vol. 24 no. 3. Retrieved June 25, 2012, from <http://www.sigsoft.org/SEN/parnas.html>
- Eldon, E. (2011). US Virtual Goods Market To Hit \$ 2 . 9 Billion In 2012 , With Facebook Games Maturing , Mobile Booming. *TechCrunch*. Retrieved September 11, 2012, from <http://techcrunch.com/2011/12/07/us-virtual-goods-market-to-hit-2-9-billion-in-2012-with-facebook-games-maturing-mobile-booming/>
- Engeln-Müllges, G., Niederdrenk, K., & Wodicka, R. (2005). *Numerik-Algorithmen* (9. Auflage., p. 677). Berlin Heidelberg: Springer-Verlag.
- Equinox. (2012). *Eclipse.org*. Retrieved April 10, 2012, from <http://www.eclipse.org/equinox/>
- Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. (H. Jørgensen & A. C. Lehmann, Eds.) *Psychological Review*, 100(3), 363–406. doi:10.1037//0033-295X.100.3.363
- EteRNA. (2011). EteRNA | Played by Humans, Scored by Nature. *Carnegie Mellon University Stanford University*. Retrieved February 24, 2011, from <http://eterna.cmu.edu/content/EteRNA>
- Facebook. (2012). Requests Overview. *Facebook Developers*. Retrieved March 8, 2012, from <http://developers.facebook.com/docs/requests/>
- Feng, W. F. W., Chang, F., Feng, W. F. W., & Walpole, J. A traffic characterization of popular on-line games. , 13 *IEEE/ACM Transactions on Networking* 488–500 (2005). IEEE Press. doi:10.1109/TNET.2005.850221
- FlightSimulatorX. (2010). Flight Simulator X. Retrieved December 12, 2010, from <http://www.microsoft.com/games/flightsimulatorx/>
- Fliplife. (2012). Fliplife. Retrieved April 26, 2012, from <http://fliplife.com/>
- FoldIt. (2010). FoldIt. *Solve Puzzles for Science | FoldIt*. Retrieved October 23, 2010, from <http://fold.it/portal/>
- Forrester, J. W. (1961). *Industrial Dynamics* (p. 479). Waltham, MA: Pegasus Communications.
- Forrester, J. W. (1994). System dynamics, systems thinking, and soft OR. *System Dynamics Review*, 10(2-3), 245–256. doi:10.1002/sdr.4260100211
- Fowler, M. (1999). *Analysemuster - Wiederverwendbare Objektmodelle* (p. 386). München: Addison-Wesley.
- Franke, C. (2012). *Bauphysikalisches Quartett (Bachelorarbeit am Lehrstuhl Bauphysik, betr. v. O. Kornadt, H. Söbke)*. Bauhaus-Universität Weimar.
- Friedrichsen, U. (2009). Der Mythos Wiederverwendung: “Design für Wartung” als eigentliches Ziel. *OBJEKTSpektrum*, (04), 42–48.
- Fritz, C., Passey, D., & Morris, P. (2009). *Independent Research Report: outcomes of using phase-6* (p. 50). Lancaster.
- Fullerton, T. (2008). *Game Design Workshop: A Playcentric Approach to Creating Innovative Games* (2nd Revise., p. 496). Morgan Kaufmann.
- G, A. (2012). A Bayesian Approach to A / B Testing. *Custora Blog*. Retrieved October 11, 2012, from <http://blog.custora.com/2012/05/a-bayesian-approach-to-ab-testing/>

- Gaber, J. (2007). Simulating Planning: SimCity as a Pedagogical Tool. *Journal of Planning Education and Research*, 27(2), 113–121. doi:10.1177/0739456X07305791
- GameDesk. (2012). GameDesk | PlayMaker School. *PlayMaker School*. Retrieved October 10, 2012, from <http://www.gamedesk.org/playmaker-school/>
- Gamma, E., & Beck, K. (2003). *Contributing to Eclipse: Principles, Patterns, and Plugins*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns*. (A.-W. P. Co, Ed.) *Elements* (Vol. 47, pp. 1–429). Addison Wesley. doi:10.1016/j.artmed.2009.05.004
- Gardner-Medwin, A. (2006). Confidence-Based Marking - towards deeper learning and better exams. In C. Bryan & K. Clegg (Eds.), *Innovative assessment in higher education* (Vol. 37, pp. 977–978). Routledge.
- Garrido, J. M. (2001). *Object-oriented discrete-event simulation with Java: a practical introduction* (p. 251).
- Gee, J. P. (2003). What Video Games Have to Teach Us About Learning and Literacy. *Comput. Entertain.*, 1(1), 20–20.
- Gee, J. P. (2005). Learning by design: Games as learning machines. *E-Learning and Digital Media*, 2(1), 5–16.
- Gee, J. P. (2008). Learning and Games. In K. Salen (Ed.), *The Ecology of Games: Connecting Youth, Games, and Learning* (The John D., Vol. 42, pp. 21–40). Cambridge, MA: The MIT Press. doi:10.1162/dmal.9780262693646.021
- Gee, J. P. (2009). Deep learning properties of good digital games: How far can they go. *Serious games: Mechanisms and effects*, 63–80.
- Gieselmann, H. (2012). Microsoft stoppt Free-to-Play-Simulation "Flight". *heise online*. Retrieved August 7, 2012, from <http://www.heise.de/newsticker/meldung/Microsoft-stoppt-Free-to-Play-Simulation-Flight-1653421.html>
- Gilmore, A. (2010). China's new gold farm. *Journal of Virtual Worlds Research*, 2(4).
- Gipser, M. (1999). *Systemdynamik und Simulation [Taschenbuch]*. *Simulation* (p. 307). Stuttgart, Leipzig: Teubner Verlag.
- Gladwell, M. (2009). *Überflieger: Warum manche Menschen erfolgreich sind - und andere nicht* (p. 272). Campus-Verlag.
- Gogg, T. J., & Mott, J. R. A. . (1993). Introduction to simulation. *WSC '93 Proceedings of the 25th conference on Winter simulation* (p. 9). Winter Simulation Conference.
- Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. (K. Eckerle, P. Hofer, K. P. Masuhr, W. Bohnenschäfer, S. Damberger, K. P. Möller, U. Fritsche, et al., Eds.) *ACM Computing Surveys*, 23(1), 5–48. doi:10.1145/103162.103163
- Gorges, K., Bröker, T., & Kornadt, O. (2007). eLearning als Weiterbildungschance für Ingenieure. *Bauphysik*, 29(2), 138–141. doi:10.1002/bapi.200790017
- Gorges, K., & Kornadt, O. (2011). Student ist nicht gleich Student - die Bedeutung der Zielgruppenorientierung bei der Konzeption und Durchführung von eLearning-Weiterbildungsangeboten. In S. Hambach, A. Martens, & B. Urban (Eds.), *eLearning Baltics 2011 - Proceedings of the 4th International eIBA Science Conference* (pp. 182–197). Rostock: Fraunhofer Verlag.
- Green, C. S., & Bavelier, D. (2003). Action video game modifies visual selective attention. *Nature* (Vol. 423, pp. 534–537). Nature Publishing Group.
- Gross, D. (1999). *Report from the fidelity implementation study group. Fall Simulation Interoperability Workshop Papers* (pp. 1–88).
- Großmann, F. (2012). *Capturing Sheep With Minecraft (Projektarbeit am Lehrstuhl Bauphysik, betr. v. H. Söbke)*. Bauhaus-Universität Weimar, Weimar.

- Gürlebeck, K., & Rudolph, R. (2009). Das Temperaturfeld - Grundlage für fundierte thermische Aussagen. *wksb - Zeitschrift für Wärmeschutz * Kälteschutz * Schallschutz * Brandschutz*, (60), 37–47.
- Hadlich, C. (2013). *Entwicklung von Verfahren zur Beschleunigung bauphysikalischer Berechnungen (Dissertation in Bearbeitung)*. Bauhaus-Universität Weimar.
- Hall, R., & Novak, J. (2008). *Online Game Development* (p. 288). Cengage Learning Services.
- Hassan, A. E., & Holt, R. C. (2004). Predicting change propagation in software systems. *20th IEEE International Conference on Software Maintenance 2004 Proceedings* (pp. 284–293). Ieee. doi:10.1109/ICSM.2004.1357812
- Hda. (2011). Gamer klären Struktur eines Virus-Enzyms auf. *Spiegel Online*. Retrieved July 8, 2012, from <http://www.spiegel.de/wissenschaft/mensch/3-d-spiel-gamer-klaren-struktur-eines-virus-enzym-auf-a-787069.html>
- Held, J. (2011). *Entwurf eines Spieler-Modells für eine erweiterbare Spielplattform zur Ausbildung in der Bauphysik (Bachelorarbeit am Lehrstuhl Bauphysik, betr. v. O. Kornadt, H. Söbke)*. Bauhaus Universität Weimar.
- Hens, H. (2008). *Building Physics - Heat Air and Moisture* (p. 284). Wiley-VCH.
- Herczeg, M. (2005). *Software-Ergonomie: Grundlagen der Mensch-Computer-Kommunikation [Broschiert]* (Auflage: v., p. 203). Oldenbourg Wissenschaftsverlag;
- Heron, K. (2012). GameDesk Opens New PlayMaker School in Los Angeles. *Spotlight On - Digital Media and Learning | Blog*. Retrieved October 10, 2012, from spotlight.macfound.org/blog/entry/GameDesk-Opens-New-PlayMaker-School/
- Herr, R. (2009). A Model of Learning Objectives. *Effective Educational Practice*. Retrieved August 24, 2012, from <http://www.celt.iastate.edu/teaching/RevisedBlooms1.html>
- Hertel, J. P., & Millis, B. (2002). *Using Simulations to Promote Learning in Higher Education* (p. 160). Stylus Publishing.
- Hibernate. (2012). Hibernate - JBoss Community. *hibernate.org*. Retrieved May 9, 2012, from <http://www.hibernate.org/>
- Hicks, A. (2010). Towards social gaming methods for improving game-based computer science education. *Proceedings of the Fifth International Conference on the Foundations of Digital Games - FDG '10*, 259–261. doi:10.1145/1822348.1822386
- Hofmann, M., Geier, C., & Kornadt, O. (2011). Jahresgänge des Raumklimas in Wohnungen - Zwischenstand einer Langzeitmessung. *Weimarer Bauphysiktagung 2011* (pp. 35–37). Verlag der Bauhaus-Universität Weimar. doi:ISBN 978-3-86068-455-9
- Holopainen, J., & Björk, S. (2008). Gameplay Design Patterns for Motivation. *ISAGA 2008* (pp. 1–5).
- Howe, J. (2006). The Rise of Crowdsourcing. *Wired*, (14).
- Hs. (2009). Interaktiver IDE-Vergleich: Eclipse vs. NetBeans vs. IntelliJ. *jaxenter*. Retrieved April 5, 2012, from <http://it-republik.de/jaxenter/news/Interaktiver-IDE-Vergleich-Eclipse-vs.-NetBeans-vs.-IntelliJ-052829.html>
- Hunnicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A Formal Approach to Game Design and Game Research. doi:10.1.1.79.4561
- Hunt, A., & Thomas, D. (2003). *Der Pragmatische Programmierer* (p. 307). München Wien: Carl Hanser Verlag.
- Irons, J. L., Remington, R. W., & McLean, J. P. (2011). Not so fast: Rethinking the effects of action video games on attentional capacity. *Australian Journal of Psychology*, 63(4), no–no. doi:10.1111/j.1742-9536.2011.00001.x
- ITWissen. (2012). *Rich Client*. Retrieved October 4, 2012, from <http://www.itwissen.info/definition/lexikon/Rich-Client-rich-client.html>
- Izurieta, C., & Bieman, J. M. (2008). Testing Consequences of Grime Buildup in Object Oriented Design Patterns (pp. 171–179).

- Izurieta, Clemente, & Bieman, J. M. (2007). How Software Designs Decay: A Pilot Study of Pattern Evolution. *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, 449–451. doi:10.1109/ESEM.2007.55
- J.A. Clarke (ESRU University of Strathclyde). (2012). ESP-r. Glasgow: ESRU.
- Jackson, S. (2011). New Games-Based Charter School to Open in Chicago. *Spotlight On - Digital Media and Learning*. Retrieved May 8, 2012, from <http://spotlight.macfound.org/blog/entry/new-games-based-charter-school-to-open-in-chicago>
- Jantke, K. P. (2009). *Taxonomien digitaler Spiele: Von Ilmenau nach Erfurt*. Erfurt.
- Jantke, K. P. (2010). Toward a taxonomy of game based learning. *IEEE International Conference on Progress in Informatics and Computing (PIC)* (pp. 858–862). Shanghai.
- Järvinen, A. (2009). Game design for social networks: interaction design for playful dispositions. In B. Perron & M. J. P. Wolf (Eds.), *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games Sandbox 09* (Vol. 1, pp. 95–102). ACM Press. doi:10.1145/1581073.1581088
- Jercic, P., Astor, P., Adam, M., & Hilborn, O. (2012). A serious game using physiological interfaces for emotion regulation training in the context of financial decision-making. *ECIS 2012 Proceedings* (p. Paper 207).
- Jones, K., Piccard, B., & Jones, G. (1995). *Simulations: A Handbook for Teachers and Trainers* (3rd ed., p. 145). Taylor & Francis.
- JRuby.org. (2012). Home — JRuby.org. Retrieved May 9, 2012, from <http://jruby.org/>
- Kauhanen, M., Tran, M., & Biddle, R. (2007). Examining Authoring Tools for Serious Games. In T. Bastiaens & S. Carliner (Eds.), *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2007* (pp. 6091–6097). Quebec City.
- Kehrer, T. (2008). A NOTE ON EXTENSIBLE SOFTWARE SYSTEMS IN LIGHT OF MAINTENANCE EFFORTS. In Hans Weghorn & A. P. Abraham (Eds.), *IADIS International Conference Informatics 2008* (pp. 213–217).
- Kharbach, M. (2011). Bloom’s Taxonomy: The 21st Century Version. *Educational Technology and Mobile Learning*. Retrieved April 4, 2012, from <http://www.educatorstechnology.com/2011/09/blooms-taxonomy-21st-century-version.html>
- Khatib, F., DiMaio, F., Cooper, S., Kazmierczyk, M., Gilski, M., Krzywda, S., Zabranska, H., et al. (2011). Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nat Struct Mol Biol*, 18(10), 1175–1177. doi:10.1038/nsmb.2119
- Kim, J., Lee, E., Thomas, T., & Dombrowski, C. (2009). Storytelling in new media: The case of alternate reality games, 2001–2009. *First Monday*, 14(6).
- King, E. M. (2011). *Guys and Games: Practicing 21st Century Workplace Skills in the Great Indoors*. University of Wisconsin - Madison.
- Kirk, W. J. (2005). *Design Patterns of Successful Role-Playing Games* (p. 255).
- Kirriemuir, J. (2002). Video gaming, education and digital learning technologies: Relevance and opportunities. *DLib Magazine*, 8(2), 0.
- Kirriemuir, J., & McFarlane, A. (2004). *Literature Review in Games and learning*. Futurelab.
- Klein, J. (2005). How Does the Architect’s Role Change as the Software Ages? *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture* (p. 141–). Washington, DC, USA: IEEE Computer Society. doi:10.1109/WICSA.2005.38
- Klein, S. A., Beckman, W. A., Mitchell, J. W., Duffie, J. A., Duffie, N. A., Freeman, T. L., Mitchell, J. C., et al. (2010). TRNSYS 17: A Transient System Simulation Program. *Solar Energy Laboratory, University of Wisconsin, Madison, USA*. Thermal Energy Systems Specialists, LLC. Retrieved January 7, 2010, from <http://sel.me.wisc.edu/trnsys/>
- Kornadt, O., Arnold, J., Wittstock, V., & Scholl, W. (2012). Prediction of the sound pressure level caused by technical equipment in buildings by analyzing transfer functions. *INTER-NOISE*

- and NOISE-CON Congress and Conference Proceedings (Vol. 2012, pp. 6292–6301). Institute of Noise Control Engineering.
- Kornadt, O., & Gorges, K. (2004). Lernnetz Bauphysik — Auralisierungen als Lernhilfe. *Bauphysik*, 26(3), 153–156. doi:10.1002/bapi.200490040
- Koskinen, J. (2010). Software maintenance costs. Retrieved June 25, 2012, from <http://users.jyu.fi/~koskinen/smcosts.htm>
- Koster, R. (2004). *A Theory of Fun for Game Design* (p. 256). Paraglyph Press.
- Kühnert, C. (2011). *Entwicklung von Spielmechaniken für ein bauphysikalisches Social Game (Bachelorarbeit am Lehrstuhl Bauphysik, betr. v. O. Kornadt, H. Söbke)*. Bauphysik. Bauhausuniversität Weimar.
- Laub, K., & Tetzner, T. (2011). *Social Game Environmental Management (Projektarbeit am Lehrstuhl Bauphysik, betr. v. H. Söbke)*. Bauhaus Universität Weimar, Weimar.
- Laurel, B. (2001). *Utopian Entrepreneur* (p. 112). The MIT Press.
- Lave, J., & Wenger, E. (1991). *Situated learning: legitimate peripheral participation*. Cambridge University Press.
- Law, A. M. (1991). *Simulation Modeling & Analysis* (p. 759). McGraw-Hill, Inc.
- LBNL. (2012). THERM. Berkeley, CA: Lawrence Berkeley National Laboratory (LBNL).
- Lehdonvirta, V., & Ernkivist, M. (2011). *Converting the Virtual Economy into Development Potential: Knowledge Map of the Virtual Economy*. infoDev / World Bank (p. 58). Washington: The World Bank.
- Leitner, S. (2011). *So lernt man lernen. Der Weg zum Erfolg*. (p. 320). Herder, Freiburg.
- Lernfreak GbR. (2012). Jura Shooter: Jagd nach dem Katzenkönig. *Lernfreak*. Retrieved July 16, 2012, from <http://lernfreak.de/produkte>
- Lewis, C., Wardrip-Fruin, N., & Whitehead, J. (2012). Motivational game design patterns of 'ville games. *Proceedings of the International Conference on the Foundations of Digital Games - FDG '12*, 172–179. doi:10.1145/2282338.2282373
- Li, R., Polat, U., Makous, W., & Bavelier, D. (2009). Enhancing the contrast sensitivity function through action video game training. *Nature Neuroscience*, 12(5), 549–551. doi:10.1038/nn.2296
- Lichtenheld, T. (2011). *Messtechnische Untersuchung des Raumklimas bei Flächenkühlung (Masterarbeit am Lehrstuhl Bauphysik, betr. v. O. Kornadt, C. Völker)*. Bauhaus-Universität Weimar.
- Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 140(140), 1–55. doi:10.1111/j.1540-5834.2010.00585.x
- LogoFoundation. (2010). Logo Foundation. Retrieved December 17, 2010, from <http://el.media.mit.edu/logo-foundation/>
- Ludewig, J., & Lichter, H. (2010). *Software Engineering* (2. überarb., p. 688). Dpunkt.Verlag GmbH.
- Lund, H. (2001). Design Reference Years and Test Reference Years in Europe. *Electronic CIB Publication No. 262*, (262), 1–21.
- Luthardt, L. (2012). *Systematik von Interaktionen zwischen mehreren Spielern in Social Network Spielen (Bachelorarbeit am Lehrstuhl Bauphysik, betr. v. O. Kornadt, H. Söbke)*. Bauhaus-Universität Weimar.
- Lutz, Jenisch, Klopfer, Freymuth, Krampf, & Petzold. (2002). *Lehrbuch der Bauphysik*. (H.-M. Fischer & E. Richter, Eds.) (5. überarb., p. 730). Stuttgart, Leipzig, Wiesbaden: Vieweg+Teubner Verlag.
- Macedonia, M. (2001). Games, Simulation, and the Military Education Dilemma. *Simulation*, 157–167.
- MacTechNews. (2010). Facebook hat erstmals mehr Seitenaufrufe als Google. *MacTechNews.de*. Retrieved September 1, 2011, from <http://www.mactechnews.de/news/index/Facebook-hat-erstmal-mehr-Seitenaufrufe-als-Google-146303.html>

- Mahajan, A. (2010). Rapidly Developing FarmVille. *GDC 2010*. Retrieved September 10, 2011, from <http://de.slideshare.net/amittmahajan/rapidly-building-farmville-how-we-built-and-scaled-a-1-facebook-game-in-5-weeks>
- Malone, T. W., & Lepper, M. R. (1987). Making learning fun: A taxonomy of intrinsic motivations for learning. *Aptitude, learning, and instruction*, 3, 223–253.
- Martin Hermann. (2006). *Numerische Mathematik* (2. Auflage., p. 522). Oldenbourg Wissenschaftsverlag.
- Martin, R. C. (2002). *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall.
- Marton, F., & Saljö, R. (1976). ON QUALITATIVE DIFFERENCES IN LEARNING: I-OUTCOME AND PROCESS. *British Journal of Educational Psychology*, 46(1), 4–11. doi:10.1111/j.2044-8279.1976.tb02980.x
- Mattern, F. (1996). Modellbildung und Simulation. *Informatik: Grundlagen-Anwendungen-Perspektiven*, 56–64.
- Mayer, R. (1998). Cognitive, metacognitive, and motivational aspects of problem solving. *Instructional science*, 26, 49–63.
- McAffer, J., & Lemieux, J.-M. (2005). *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java(TM) Applications*. Addison-Wesley Professional.
- McCardle, M. (2003). Fan fiction, fandom, and fanfare: What's all the fuss? *BUJ Sci. & Tech. L.*, 9(433).
- McConnell, S. (2005). *Code Complete* (2nd Editio., p. 940). Microsoft GmbH.
- McGee, P. (2003). Learning objects: Bloom's taxonomy and deeper learning principles. *7th annual E-learn conference. Norfolk: Association for the Advancement of Computing in Education*.
- McHaney, R. (1991). *Computer Simulation - A practical perspective* (p. 276). San Diego, CA, USA: Academia Press.
- McHaney, R. (2009). *Understanding Computer Simulation. Simulation* (Vol. 77, pp. 1–172). Roger McHaney & Ventus Publishing ApS.
- Mens, T., Wermelinger, M., Ducasse, S., Demeyer, S., Hirschfeld, R., & Jazayeri, M. (2005). Challenges in Software Evolution. *Eighth International Workshop on Principles of Software Evolution IWPS05, 0(April)*, 13–22.
- Merriënboer, J. J. G. van, Jelsma, O., & Paas, F. G. W. C. (1992). Training for reflective expertise: A four-component instructional design model for complex cognitive skills. *Educational Technology Research and Development*, 40(2), 23–43. doi:10.1007/BF02297047
- Merriënboer, J. J. G. van, & Kirschner, P. A. (2007). *Ten Steps to Complex Learning: A Systematic Approach to Four-Component Instructional Design* (p. 306). Lawrence Erlbaum Assoc Inc.
- Merrill, M. D. (1994). *Instructional Design Theory*. (D. G. Twitchell, Ed.) (p. 465). Educational Technology Publications.
- Merrill, M. D. (2000). Knowledge objects and mental models. *Proceedings International Workshop on Advanced Learning Technologies IWALT 2000 Advanced Learning Technology Design and Development Issues*, 2, 244–246. doi:10.1109/IWALT.2000.890621
- Merrill, M. D., Drake, L., & Lacy, M. (1966). Reclaiming instructional design. *Educational Technology*, 3&(5), 5–7.
- Mertins, K., Rabe, M., & Jaekel, F.-W. (2000). Neutral template libraries for efficient distributed simulation within a manufacturing system engineering platform. *Proceeding of the 32nd conference on Winter simulation* (pp. 1549–1557).
- Meyer, M. (2011). Per Spiel zum Traumjob? *FORUM - Das Wochenmagazin*. Retrieved May 21, 2012, from <http://www.magazin-forum.de/per-spiel-zum-traumjob/>
- Mims, C. (2010). Honors Course Using StarCraft Is for Gamers Only. *Technology Review*. Retrieved May 10, 2012, from <http://www.technologyreview.com/view/420274/honors-course-using-starcraft-is-for-gamers-only/>

- Mitra, S., Dangwal, R., & Thadani, L. (2008). Effects of remoteness on the quality of education: A case study from North Indian schools. *Australasian Journal of Educational ...*, 24(2), 168–180.
- Mittermeir, R. (2001). Software evolution: let's sharpen the terminology before sharpening (out-of-scope) tools. *Workshop on Principles of Software Evolution*, 114–121.
- MobyGames. (2012). Genre Definitions. *Moby Games*. Retrieved August 9, 2012, from <http://www.mobygames.com/glossary/genres/>
- Morgan, G. (2009). Challenges of Online Game Development: A Review. *Simulation & Gaming*, 40(5), 688–710. doi:10.1177/1046878109340295
- MPAA. (2012). *Theatrical Market Statistics 2011* (p. 21). Motion Picture Association of America, Inc.
- Muetzelfeldt, R., & Massheder, J. (2003). The Simile visual modelling environment. *European Journal of Agronomy*, 18(3-4), 345–358. doi:10.1016/S1161-0301(02)00112-0
- Müller, Naira, Hennig, C., Aubel, M., Hesse, T., & Schneider, S. (2012). *FlipLife als Mitarbeiterrekrutierungsquelle (Projektarbeit am Lehrstuhl Bauphysik, betr. v. H. Söbke)*. Bauhaus-Universität Weimar.
- Müller, Naria. (2012). *Erweiterung von Fliplife mit bauphysikalischen Inhalten (Bachelorarbeit am Lehrstuhl Bauphysik, betr. v. O. Kornadt, H. Söbke)*. Bauhaus-Universität Weimar.
- Muntean, C. (2011). Raising engagement in e-learning through gamification. *Proc. 6th International Conference on Virtual Learning ICVL* (pp. 323–329).
- Murphy, K., & Spencer, A. (2009). Playing video games does not make for better visual attention skills. *Journal of Articles in Support of the Null Hypothesis*, 6(1), 1–20.
- MySQL. (2012). MySQL :: The world's most popular open source database. Retrieved April 4, 2012, from <http://www.mysql.com/>
- Nelson, R. (2011). Triviador is a Facebook Strategy Game That Asks Who Will Rule the World? *Inside Social Games*. Retrieved May 5, 2012, from <http://www.insidesocialgames.com/2011/11/10/triviador-is-a-facebook-strategy-game-that-asks-who-will-rule-the-world/>
- Nicol, D. J., & Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. *Studies in Higher Education*, 31(2), 199–218. doi:10.1080/03075070600572090
- Niegemann, H. M. (2009a). Lernen durch Computerspiele : Das spielende Klassenzimmer? [Vortrag im Rahmen der Ringvorlesung der Universität Erfurt "Spielemedien - Medienspiele" am 08.12.2009]. Retrieved October 9, 2012, from <http://www.db-thueringen.de/servlets/DerivateServlet/Derivate-19610/niegemann.pdf>
- Niegemann, H. M. (2009b). Instructional Design. In M. Henninger & H. Mandl (Eds.), *Handbuch Medien- und Bildungsmanagement* (1. Auflage., pp. 354–358). Beltz.
- Niegemann, H. M. (2011). Interaktivität in Online-Anwendungen. In P. Klimsa & L. Issing (Eds.), *Online-Lernen: Planung, Realisation, Anwendung und Evaluation von Lehr- und Lernprozessen online* (2. verbess., pp. 125–137). München: Oldenbourg.
- Niegemann, H. M., Domagk, S., Hessel, S., Hein, A., Hupfer, M., & Zobel, A. (2008). *Kompendium multimediales Lernen* (p. 679). Berlin Heidelberg: Springer-Verlag. doi:10.1007/978-3-540-37226-4
- Nutt, C. (2011). A Philosophy That Extends Eastward: Social Games Zynga-Style. *Gamasutra*. Retrieved April 6, 2011, from http://www.gamasutra.com/view/feature/6280/a_philosophy_that_extends_.php
- O'Conner, J. (2006). Scripting for the Java Platform. *Sun Developer Network*. Retrieved May 8, 2012, from <http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/>
- Oldenburg, R. (1999). *The Great Good Place: Cafes, Coffee Shops, Bookstores, Bars, Hair Salons, and Other Hangouts at the Heart of a Community* (3rd Editio., p. 368). Marlowe & Company.

- Oracle. (2009). Project Darkstar. *Oracle Labs*. Retrieved April 4, 2012, from <http://labs.oracle.com/projects/dashboard.php?id=168>
- Oracle. (2012). javadoc tool home page. *Oracle Technology Network*. Retrieved June 28, 2012, from <http://www.oracle.com/technetwork/java/javase/documentation/javadoc-137458.html>
- Osborne, J. (2011). Gardens of Time nabs Best Social Network Game at GDC Online 2011. *Games.com*. Retrieved July 16, 2012, from <http://blog.games.com/2011/10/17/gardens-of-time-best-social-game-gdc-online-2011/>
- OSGi Alliance | About / Benefits of Using OSGi. (2012). *OSGi Alliance*. Retrieved April 10, 2012, from <http://www.osgi.org/About/WhyOSGi>
- Overmars, M. (2005). Learning object-oriented design by creating games. *IEEE Potentials*, 23(5), 11–13. doi:10.1109/MP.2005.1368910
- Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. ACM Classic Books Series. New York: Basic Books.
- Parnas, D. L. (1994). Software Aging. *Proceedings of the 16th international conference on Software engineering* (Vol. 81, pp. 279–287). IEEE Computer Society Press. doi:10.1109/ICSE.1994.296790
- Pat McCarthy (SDI Corp.). (2003, October 14). Put Eclipse features to work for you. *developerWorks*. Retrieved April 5, 2012, from <http://www.ibm.com/developerworks/opensource/library/os-ecfeat/>
- Peters, I. (2009). *Folksonomies : Indexing and retrieval in Web 2.0. Science* (Vol. 2, p. 443). De Gruyter/Saur. doi:10.1177/016555158000200203
- Phase-6. (2012). vokabeltrainer / lernprogramm fürs vokabel-lernen. *Phase-6*. Retrieved July 14, 2012, from <http://www.phase-6.de/opencms/Homepage/>
- Phenomedia. (2012). MOORHUHN.de. *phenomedia publishing GmbH*. Retrieved July 11, 2012, from <http://www.moorhuhn.de/>
- PHP Development Tools. (2012). Retrieved April 5, 2012, from <http://www.eclipse.org/projects/project.php?id=tools.pdt>
- Plag, R. (2010). Die wichtigsten 10 Dämmstoffe im Vergleich. *u-wert.net*. Retrieved August 9, 2012, from <http://www.u-wert.net/10-daemmstoffe-im-vergleich/>
- Playdom. (2012). Gardens of Time. *Gardens of Time*. Retrieved July 16, 2012, from <http://www.facebook.com/GardensofTime>
- Plumer, B. (2012, September). The economics of video games. *The Washington Post*. Retrieved November 9, 2012, from <http://www.washingtonpost.com/blogs/ezra-klein/wp/2012/09/28/the-economics-of-video-games>
- Poling, N. (2010). 21st Century Skills in Starcraft. Retrieved January 18, 2011, from <http://www.honors.ufl.edu/courses/coursesfall10.html>
- Powell, E. M., Finkelstein, S., Hicks, A., Phifer, T., Charugulla, S., Thornton, C., Barnes, T., et al. (2010). SNAG: social networking games to facilitate interaction. *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems* (pp. 4249–4254). ACM.
- Powell, E., Stukes, F., Barnes, T., & Lipford, H. R. (2011). Snag'em: Creating Community Connections through Games. *2011 IEEE Third Intl Conference on Privacy Security Risk and Trust and 2011 IEEE Third Intl Conference on Social Computing* (pp. 591–594). IEEE. doi:10.1109/PASSAT/SocialCom.2011.229
- Pree, W. (1994). *Design Patterns for Object-Oriented Software Development* (p. 268). Addison-Wesley.
- Premsky, M. (2001). Digital Natives, Digital Immigrants Part 1. *On the Horizon*, 9(5), 1–6. doi:10.1108/10748120110424816

- Prensky, M. (2004). Proposal for educational software development sites: an open source tool to create the learning software we need. *On the Horizon Vol 12 Iss 1*, 12(1), 41–44. doi:10.1108/10748120410699585
- Putnam, R. D. (1995). Bowling Alone: America's Declining Social Capital. *Journal Of Democracy*, 6(1), 65–78. doi:10.1353/jod.1995.0002
- Python. (2012). Python Programming Language – Official Website. *python.org*. Retrieved May 9, 2012, from <http://www.python.org/>
- Quest to Learn. (2010). Quest to Learn. Retrieved December 23, 2010, from <http://q2l.org/>
- Rao, V. (2008). Facebook Applications and playful mood: the construction of Facebook as a third place. *Proceedings of the 12th International MindTrek Conference - MindTrek'08* (pp. 8–12). New York: ACM Press.
- RedDwarf Server. (2010). Retrieved April 4, 2012, from <http://www.reddwarfserver.org/>
- Reiser, R., & Dempsey, J. V. (2007). *Trends and Issues in Instructional Design and Technology* (p. 408). Prentice Hall.
- Reynolds, P. F. J. (2009). The Role of Modeling and Simulation. In J. A. Sokolowsky & C. M. Banks (Eds.), *Principles of Modeling and Simulation A Multidisciplinary Approach* (pp. 25–43). John Wiley & Sons. doi:10.1002/9780470403563.ch2
- Robinson, S. (2004). *Simulation: The Practice of Model Development and Use* (Vol. 67, p. 336). Wiley.
- Rockwell, G. M., & Kee, K. (2011). Game Studies - The Leisure of Serious Games: A Dialogue. *Game Studies - the international journal of computer game research*, 11(2).
- Rosser, J. C., Lynch, P. J., Cuddihy, L., Gentile, D. A., Klonsky, J., & Merrell, R. (2007). The impact of video games on training surgeons in the 21st century. *Archives of surgery Chicago Ill* 1960, 142(2), 181–6; discussion 186. doi:10.1001/archsurg.142.2.181
- Rubinstein, R. Y., & Kroese, D. P. (2008). *Simulation and the Monte Carlo Method. Wiley Series in Probability and Mathematical Statistics* (p. 372). Wiley-Interscience.
- Ryan, M., Costello, B., & Stapleton, A. (2012). Deep Learning Games through the Lens of the Toy. *Meaningful Play 2012*. Retrieved October 27, 2012, from http://meaningfulplay.msu.edu/proceedings2012/mp2012_submission_6.pdf
- Ryan, R., & Deci, E. (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary educational psychology*, 25(1), 54–67. doi:10.1006/ceps.1999.1020
- Salandin, A., Arnold, J., & Kornadt, O. (2011). Noise in an intensive care unit. *The Journal of the Acoustical Society of America*, 130(6), 3754–3760.
- Scacchi, W. (2010). Computer game mods, modders, modding, and the mod scene. *First Monday*, 15(5).
- Schank, R. C., Fano, A., Bell, B., & Jona, M. (1994). The Design of Goal-Based Scenarios. *Journal of the Learning Sciences*, 3(4), 305–345. doi:10.1207/s15327809jls0304_2
- Schell, J. (2008). *The Art of Game Design: A book of lenses* (1st ed., p. 512). Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.
- Schmidt, C. (2008). Die Monte-Carlo- Methode zur Bestimmung der Kreiszahl Pi. *Zentrale für Unterrichtsmedien im Internet e.V.* Retrieved March 19, 2012, from http://www.zum.de/Faecher/Inf/RP/Java/java_1.htm
- Schmidt, J., & Kornadt, O. (2010). Convective Moisture Transfer through Walls and Wall Components. In K. Kupfer (Ed.), *MFWA Weimar, Aquametry 2010. First European Conference on Moisture Measurement. Weimar* (pp. 527–534).
- Schmolitzky, A., & Schümmer, T. (2002). Entwurfsmuster zur Betreuung von Abschlussarbeiten. *e-teaching.org*. Retrieved March 9, 2011, from <http://www.e-teaching.org/praxis/erfahrungsberichte/EntwurfsmusterFuerAbschlussarbeiten.pdf>
- Schneider, K.-J. (Ed.). (2001). *Bautabellen für Architekten - mit Berechnungshinweisen und Beispielen* (14. ed.). Düsseldorf: Werner Verlag GmbH.

- Schriber, T. J., & Brunner, D. T. (2005). Inside discrete-event simulation software: how it works and why it matters. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, & J. A. Joines (Eds.), *Proceedings of the 2005 Winter Simulation Conference* (pp. 167–177).
- Schürenberg, P. (2012). *Automatisierte fachliche Qualitätssicherung in einem Crowdsourcing-basierten bauphysikalischen Lernspiel (Bachelorarbeit am Lehrstuhl Bauphysik, betr. v. O. Kornadt, H. Söbke)*. Bauhaus-Universität Weimar.
- Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum (Series in Agile Software Development)* (p. 158). Prentice Hall.
- Schwetman, H. (1996). CSIM18 - THE SIMULATION ENGINE. *Proceedings of the 28th conference on Winter simulation* (pp. 517–521). IEEE Computer Society.
- Selic, B. (2003). The pragmatics of model-driven development. *Software, IEEE*, 20(5), 19–25. doi:10.1109/MS.2003.1231146
- Semendjajew, K. A., Musiol, G., Muehlig, H., & Bronstein, I. (2008). *Taschenbuch der Mathematik*. Harri Deutsch.
- Senge, P. M. (1990). *The Fifth Discipline. Business* (Vol. 1, p. 424). Doubleday.
- Shankland, S. (2006). Sun's Project Darkstar aims for gaming services. *News Blogs - CNET News*. Retrieved April 4, 2012, from http://news.cnet.com/8301-10784_3-6054494-7.html
- Shapado.com. (2012). Shapado. www.shapado.com. Retrieved January 8, 2012, from <https://shapado.com/>
- SimCity. (2010). SimCity Societies. Retrieved January 7, 2011, from <http://simcitysocieties.ea.com/creator/>
- Simulistics. (2003). Simulistics - Model Life. *Ecological modelling with Simile*. Retrieved from <http://www.simulistics.com/files/documents/SimilePresentations/SimileCourse1/SimileCourse1.html>
- SISO. (2012). SISO Standards. *Simulation Interoperability Standards Organization*. Retrieved March 16, 2012, from <http://www.sisostds.org/ProductsPublications/Standards/SISOStandards.aspx>
- SmartFoxServer. (2012). Retrieved April 4, 2012, from <http://www.smartfoxserver.com/>
- Snow, S. (2010). "FarmVille" vs. Real Farms. *mashable.com*. Retrieved October 16, 2010, from <http://mashable.com/2010/09/10/farmville-vs-real-farms-infographic/>
- Söbke, H., Bröker, T., & Kornadt, O. (2012). Social Gaming – Just Click and Reward? In P. Felicia (Ed.), *Proceedings of the 6th European Conference on Games Based Learning* (pp. 478–486). Academic Publishing Limited.
- Söbke, H., Corredor, J. A., & Kornadt, O. (2011). Learning, Reasoning and Modeling in Social Gaming. In S. Hambach, A. Martens, & B. Urban (Eds.), *eLearning Baltics 2011 - Proceedings of the 4th International eIBA Science Conference* (pp. 106–122). Rostock: Fraunhofer Verlag.
- Söbke, H., Hadlich, C., Bröker, T., & Kornadt, O. (2011). Concept of a Gaming Platform for Domain-Specific User-Created Content. In D. Gouscos & M. Meimaris (Eds.), *5th European Conference on Games Based Learning* (pp. 759–766). Athens, Greece: Academic Publishing Limited.
- Söbke, H., Hadlich, C., Müller, N., Hesse, T., Hennig, C., Schneider, S., Aubel, M., et al. (2012). Social Game Fliplife : Digging for talent – an analysis. In P. Felicia (Ed.), *Proceedings of the 6th European Conference on Games Based Learning* (pp. 487–494). Academic Publishing Limited.
- Solarski, C. (2012). *Drawing Basics and Video Game Art: Classic to Cutting-Edge Art Techniques for Winning Video Game Design* (p. 240). Watson-Guptill.
- Squire, K. (2003). *Replaying History: Learning World History through playing Civilization III*. Indiana University.
- Squire, K. (2006). Videogames as Designed Experience From Content to Context. *Educational Researcher*, 35(8), 19–29. doi:10.3102/0013189X035008019

- Squire, K. (2008). Video Game-Based Learning: An Emerging Paradigm for Instruction. *Performance Improvement Quarterly*, 21(2), 7–36.
- Squire, K. (2011). *Video Games and Learning: Teaching and Participatory Culture in the Digital Age* (p. 312). Teachers College Press.
- Stackoverflow.com. (2012). Stack Overflow. *stackoverflow.com*. Retrieved July 17, 2012, from <http://stackoverflow.com/>
- Steinhausen, D. (1994). *Simulationstechniken* (p. 126). München: Oldenbourg.
- Steinkuehler, C. A. (2006). Why game (culture) studies now? *Games and Culture*, 1(1), 97.
- Steinkuehler, C. A., & Duncan, S. (2008). Scientific Habits of Mind in Virtual Worlds. *Journal of Science Education and Technology*, 17(6), 530–543. doi:10.1007/s10956-008-9120-8
- Steinkuehler, C. A., & Williams, D. (2006). Where Everybody Knows Your (Screen) Name: Online Games as “Third Places”. *Journal of Computer-Mediated Communication*, 11(4), 885–909. doi:10.1111/j.1083-6101.2006.00300.x
- Stempell, I. (2008). Desmo-J – Framework für ereignisorientierte diskrete Simulation in neuer Version. *Stempell*. Retrieved August 7, 2012, from <http://stempell.com/2008/12/desmo-j-framework-fur-ereignisorientierte-diskrete-simulation-in-neuer-version/>
- Stroustrup, B. (1991). What is “Object-Oriented Programming”? (1991 revised version). *IEEE Software*, 5(May 1988), 10–20. doi:10.1109/52.2020
- Sulistio, A., Yeo, C. S., & Buyya, R. (2004). A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *Software Practice and Experience*, 34(7), 653–673. doi:10.1002/spe.585
- Superdata. (2012). Worldwide virtual goods market reaches \$ 15 billion . Monetization still a four letter word. *Superdata News*. Retrieved August 11, 2012, from <http://www.superdataresearch.com/monetization-is-a-four-letter-word/>
- Swaim, M. (2009). 10 Video Games That Should Be Considered Modern Art. *Cracked.com*. Retrieved November 10, 2012, from <http://www.cracked.com/blog/defending-the-habit-10-video-games-as-modern-art/>
- Tagg, J. (2003). *The Learning Paradigm College* (p. 379). Bolton: Anker Publishing Company.
- Taylor, T. L. (2012). *Raising the Stakes: E-Sports and the Professionalization of Computer Gaming* (1st Printi., p. 336). The MIT Press.
- Thomas, D. (2005). Agile programming: design to accommodate change. *Software, IEEE*, 22(3), 14–16.
- Thompson, J. R. . (2000). *Simulation: a modeler's approach* (p. 297). John Wiley & Sons.
- THX_Games_PLC. (2011). Triviador on Facebook. *Facebook*. Retrieved October 10, 2012, from <http://apps.facebook.com/triviador>
- Tolk, A., Balci, O., Diallo, S. Y., Fishwick, P. A., Hu, X., Loper, M., Petty, M. D., et al. (2011). TOWARDS A METHODOLOGICAL APPROACH TO IDENTIFY FUTURE M&S STANDARD NEEDS. In S. Jain, R. R. Creasey, J. Himmelspace, K. P. White, & M. Fu (Eds.), *Proceedings of the 2011 Winter Simulation Conference* (pp. 2980–2997). doi:10.1109/WSC.2011.6147999
- Treml, A. K., & Becker, N. (2010). Lernen. In H.-H. Krüger & W. Helsper (Eds.), *Einführung in Grundbegriffe und Grundfragen der Erziehungswissenschaft* (9. Auflage., pp. 103–114). Obladen: Verlag Barbara Budrich.
- Unity. (2012). UNITY: Store. *Unity Technologies*. Retrieved April 2, 2012, from <https://store.unity3d.com/products?currency=EUR>
- Varga, A., & Hornig, R. (2008). An overview of the OMNeT++ simulation environment. *Simutools '08 Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*.
- Varga, A., & Hornig, R. (OpenSim L. . (2012). OMNeT++ IDE Webdemo. *omnetpp.org*. Retrieved March 7, 2012, from <http://omnetpp.org/webdemo/ide/>
- Venners, B. (2005a). How to Use Design Patterns. *artima developer*. Retrieved July 28, 2009, from <http://www.artima.com/lejava/articles/gammapP.html>

- Venners, B. (2005b). Erich Gamma on Flexibility and Reuse. *artima developer*. Retrieved July 24, 2009, from <http://www.artima.com/lejava/articles/reuseP.html>
- Ventana. (2012). Vensim. *Ventana Systems Inc.* Retrieved from <http://www.vensim.com/>
- Vogel, A., Kornadt, O., Wittsock, V., & Scholl, W. (2012). Measurement and prediction of structure-borne sound power in plate-shaped building elements. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings (Vol. 5)* (Vol. 2012, pp. 6247–6258). Institute of Noise Control Engineering.
- Völker, C., & Kornadt, O. (2011). Thermal comfort--simulation and measurement using a thermal manikin. *Roomvent 2011, Trondheim, Norway*. doi:ISBN 9788251928120
- Völker, C., Kornadt, O., & Ostry, M. (2008). Temperature reduction due to the application of phase change materials. *Energy and Buildings*, 40(5), 937–944. doi:10.1016/j.enbuild.2007.07.008
- Völker, C., Schmidt, J., Arnold, J., Bode, K.-A., Baldy, F., Thurow, T., Braunes, J., et al. (2013). *Methoden und Baustoffe zur nutzerorientierten Bausanierung (nuBau)*. Abschlussbericht (in Druck).
- W3stfa11. (2012). Most expensive video games. *Video Game Sales Wiki*. Retrieved August 3, 2012, from http://vgsales.wikia.com/wiki/Most_expensive_games
- Wagner, M. G. (2008). Interaktionstechnologie im gesellschaftlichen Spiel : eine Grundsatzthese zur kulturellen Bedeutung von digitalen und hybriden Spielen. In K. Mitgutsch & H. Rosenstingl (Eds.), *Faszination Computerspielen : Theorie - Kultur - Erleben* (pp. 47–55). Wien: Braumüller.
- Wagner, M. G. (2009). Serious Games: Spielerische Lernumgebungen und deren Design. In L. J. Issing & P. Klimsa (Eds.), *Online-Lernen: Handbuch für Wissenschaft und Praxis*. München: Oldenbourg Wissenschaftsverlag;
- Wagner, M. G., & Mitgutsch, K. (2008). *Didaktische Szenarien des Digital Game Based Learning*. Projektendbericht (p. 46).
- Wainer, G. (2009). *Discrete-event modeling and simulation: a practitioner's approach*. Design (p. 520). Boca Raton, FL: CRC Press.
- Wei, S. (2010). Decorated Farm in FarmVille. *Facebook*. Retrieved April 24, 2012, from https://fbcdn-sphotos-a.akamaihd.net/hphotos-ak-snc6/224850_208017265886951_100000360370627_655563_2452168_n.jpg
- Wenger, E. (2006). Communities of practice - a brief introduction. *Etienne Wenger*. Retrieved September 10, 2012, from <http://www.ewenger.com/theory/>
- Willems, W. M., Schild, K., & Dinter, S. (2006). *Vieweg Handbuch Bauphysik 2: Schall- und Brandschutz, Fachwörterglossar deutsch-englisch, englisch-deutsch: TEIL 2* (p. 556). Vieweg+Teubner Verlag.
- Willis, D., & Clift, J. C. (1983). Approaches to study adopted by school-leavers. *Proceedings of the AARE Conference 1983 Canberra1* (pp. 435–441). Canberra.
- Wright, W. (2007). Will Wright makes toys that make worlds. Retrieved December 23, 2010, from http://www.ted.com/talks/will_wright_makes_toys_that_make_worlds.html
- xampp. (2012). *apache friends*. Retrieved April 4, 2012, from <http://www.apachefriends.org/de/xampp.html>
- Yee, N. (2006). Motivations for play in online games. *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, 9(6), 772–5. doi:10.1089/cpb.2006.9.772
- Yourdon, E., & Constantine, L. L. (1979). *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design* (p. 473). Prentice Hall.
- YOYOGames. (2010). Gamemaker. *YOYO Games*. Retrieved December 14, 2010, from <http://www.yoyogames.com/gamemaker>
- Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000). *Theory of Modeling and Simulation. Simulation* (Vol. 132, p. 510). San Diego, CA, USA: Academic Press. doi:10.1159/000074301

- Zichermann, G., & Cunningham, C. (2011). *Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps* (1st editio., p. 208). O'Reilly Media.
- Zimbardo, P. G., & Gerrig, R. J. (2004). *Psychologie* (16., aktua., p. 943). München: Pearson.
- Zimmermann, A. (2008). *Stochastic Discrete Event Systems* (p. 391). Berlin Heidelberg New York: Springer.
- Zynga. (2009). FarmVille. Retrieved from <http://apps.facebook.com/onthefarm/>

Anhang A: Simulationsspielplattform: Verwandte Software

Inhaltsverzeichnis

1 Einleitung.....	2
2 Kategorien	3
2.1 Spielportale	3
2.2 Physik-affine Spiele.....	3
2.3 Fachspezifische Portale	4
2.4 Simulationsportale	4
2.5 Lernplattformen	4
2.6 Bildende (Spiel-)Entwicklungsumgebungen	5
2.7 Spielentwicklungsumgebungen.....	5
2.8 Simulationsframeworks.....	6
2.9 Simulationsplattformen.....	6
2.10 Simulationsspiele.....	6
2.11 Simulatoren	7
2.12 Crowdsourcing-basierte Spiele.....	7
3 Softwareübersicht	8
4 Softwarebeschreibungen	10
5 Schlussfolgerung.....	17

1 Einleitung

Ein wesentlicher Teil der Ingenieurwissenschaften besteht aus dem Umgang mit Systemen. Daraus erwachsen komplexe Probleme, deren Lösungen weder algorithmisch zu bestimmen sind noch eindeutig definiert sind. Simulation ist eine Möglichkeit, mit solchen Problemen umzugehen. Sie kann helfen, Probleme des realen Lebens zu analysieren und zu lösen. In der Ausbildung können simulationsbasierte Computerspiele eingesetzt werden, die ebenfalls gleichartige Probleme bereitstellen und damit authentische Lernkontexte erzeugen können (Bröker & Kornadt, 2009). Computerspiele werden damit zu einem wertvollen Lernmedium, nicht auch zuletzt, weil sie sich zu einem Kristallisationspunkt für Gemeinschaften entwickeln, die das gemeinschaftliche Lösen von Problemen kultivieren (Steinkuehler, 2006). Es entstehen sogenannte *Communities of Practice* (Lave & Wenger, 1991).

Seymour Papert unterstreicht für die Lerntheorie des Konstruktivismus die Bedeutung des aktiven Handelns für den Aufbau von Wissen (Papert, 1980). Computerspiele bieten Möglichkeiten zum aktiven Handeln. Jedes Spiel fordert Handlungen vom Spieler. Darüber hinaus stellen einige Spiele dem Spieler auch die Möglichkeit bereit, Inhalte selbst zu entwickeln und damit zum Spiel beizutragen. Das ist gleichfalls eine Form des aktiven Handelns.

Die vorherigen Überlegungen führten zu der Idee eines „Bauphysikalischen MMOG¹“ zur Ausbildungsbegleitung von Studenten. Ein solches Spiel hat die folgenden Eigenschaften:

1. Es ist ein **Spiel**.
2. Es ist **simulationsbasiert**.
3. Es besitzt einen **Mehrspielermodus**, damit die Spieler gemeinsame Aktionen durchführen können.
4. Es bietet (auch) **bauphysikalische** Inhalte, die vermittelt werden sollen².
5. Es ist offen für **Erweiterungen** durch die Spieler, die bei der Erstellung neuer Inhalte lernen.
6. Es besitzt einen gemeinsamen thematischen Bogen, in den die Inhalte **integriert** werden können.

Aufzählung 1 Gewünschte Eigenschaften der angestrebten Spielplattform

In dieser exakten Konfiguration gibt es bisher noch keine Spielplattform zur Ausbildungsunterstützung. Die Erstellung eines solchen Spiels von Grund auf ist jedoch sehr aufwändig und damit auch teuer. Möglicherweise gibt es jedoch Software, die die Basis für ein solches Spiel bietet und entsprechend ausgebaut werden kann. Daher wurde eine Recherche nach einer derartigen Software durchgeführt, deren Ergebnisse hier präsentiert werden. Die sicherlich nicht allumfassende Recherche wurde mit Suchbegriffen wie

- *Game platform*
- *Simulation platform*
- *Serious Game / Educational Game*
- *Crowdsourcing*
- *IDE / Integrated Development Environment*

¹ MMOG: Massively Multiplayer Online Game

² Die Anforderung eines umfassenden pädagogischen Konzepts wird nicht gestellt. Es wird angenommen, dass Spiele von Grund auf als Lernmaschinen aufgebaut sind (Gee, 2005) und eine offensichtliche pädagogische Ausrichtung die Spielerschaft eher verschreckt (Squire, 2011).

- *Portal*

gestartet. Die Ergebnisse werden in diesem Dokument dargestellt³.

In Kapitel 2 werden Kategorien für die gefundenen Softwarepakete aufgestellt und beschrieben. Danach werden in Kapitel 3 die Pakete in Übersichtstabellen dargestellt und kategorisiert. Kapitel 4 legt detaillierte Informationen für die einzelnen Programme mitsamt einer Einschätzung der Eignung für den oben gegebenen Zweck dar. Schließlich werden in Kapitel 5 die Ergebnisse zusammengefasst.

2 Kategorien

Die Kategorien werden mit Hilfe von Beispielen erläutert. Sind diese ohne Referenz genannt, werden sie detaillierter in Kapitel 3 unter Nennung einer solchen beschrieben.

2.1 Spielportale

Sehr früh fallen Web-Portale für webbasierte Spiele auf: In diesen sind die Spiele oft thematisch sortiert, einige dieser Webportale bieten die Spiele zur kostenlosen Nutzung an. Mitunter gibt es Physik-Kategorien in diesen Portalen bzw. auch spezialisierte Physik-Spiele-Webportale: Der Spieler kann das Spiel durch Anklicken starten. Er kann gegebenenfalls ein Tutorial durchführen und dann spielen.

Bezüglich der in Aufzählung 1 genannten Anforderungen wird in der Regel nur der Spielcharakter (Punkt 1) sowie die thematische Ausrichtung (Punkt 4) erfüllt. Die Spielplattformen sind nicht erweiterbar, es sind gewöhnlich Spiele für den Einzelspielermodus. Der fachliche Aspekt, der dargestellt wird, geht oft eher in Richtung Physik als in Richtung Bauphysik. Ein Beispiel für eine solche Sammlung ist unter <http://www.freewebarcade.com/physics-games.php>⁴ zu finden.

2.2 Physik-affine Spiele

Die thematische Ausrichtung (Punkt 4) wird durch Spiele unterstützt, die um einen ausgewählten Aspekt des Fachgebiets als zentrales Element der Spielmechaniken entwickelt wurden⁵. Zu den Spielen, die einen physikalischen Effekt thematisieren gehören beispielsweise *Angry Birds* und *Portal*. Um *Angry Birds* erfolgreich spielen zu können, muss ein Spieler die Grundlagen der Ballistik beachten (siehe Abbildung 1). *Portal* verknüpft den (fiktiven) Effekt der Teleportation mit Impuls und Drehimpuls: Drehimpuls und Impuls werden durch die Teleportation nicht verändert, sie werden nur beeinflusst durch die neue Umgebung, in die der Avatar nach der Teleportation gelangt. In der neuen Umgebung kann sich die Bewegungsrichtung geändert haben – die Wirkung der Schwerkraft auf die Bewegung hat sich geändert.

Gewöhnlich werden diese Spiele kommerziell entwickelt und lassen sich nicht erweitern.

³ Die Recherche wurde im Dezember 2010 durchgeführt, die Reinschrift in Form dieses Dokuments erfolgte erst im August 2012. Daraus ergeben sich teilweise Quellangaben, auf die nach Dezember 2010 zugegriffen wurde.

⁴ Letzter Zugriff: 02.10.2011

⁵ Da kommerzielle Spiele, die Phänomene aus dem Bereich der Bauphysik nutzen, nicht gefunden wurden, wurde die Suche ausgedehnt auf das umfassendere Gebiet der Physik.



Abbildung 1 Angry Birds: Typische Spielszene

2.3 Fachspezifische Portale

Der gemeinsame thematische Bogen (Punkt 6 aus Aufzählung 1) ist insbesondere bei fachspezifischen Portalen gegeben. In solchen Portalen wird das Wissen zu einem bestimmten Themengebiet, dargestellt mit den unterschiedlichsten Medien, gesammelt. Oft gibt es für den Teilnehmer ein Benutzerkonto, unter dem er auch Inhalte beitragen kann (Punkt 5). *HUBZero* ist ein Beispiel für Software, die zur Bereitstellung eines solchen Portals genutzt werden kann – sie wurde themenunabhängig entwickelt.

Obwohl in einigen Portalen auch Spiele als Lernmedium enthalten sind, fehlt häufig das Merkmal des Spiels.

2.4 Simulationsportale

Ebenso wie Spielportale gibt es Sammlungen von Simulationen. Diese dienen der Ausbildung und sind fachlich geordnet. Ein solches Simulationsportal stellt die Firma *ExploreLearning* unter www.explorelearning.com bereit.

Ähnlich wie die fachspezifischen Portale, lassen Lernportale den Charakter des übergreifenden Spiels vermissen.

2.5 Lernplattformen

Der Begriff webbasierte Lernplattform wird durch Baumgartner, Maier-Häfele, & Häfele (2004) definiert als „eine serverseitig installierte Software [...], die beliebige Lerninhalte über das Internet zu vermitteln hilft und die Organisation der dabei notwendigen Lernprozesse unterstützt“. Grundsätzlich war es nicht das Ziel der Recherche, Lernplattformen zu untersuchen, da bei diesen der Anteil an spielerischen Komponenten nicht ausreichend ist. Der Aspekt einer Lernplattform, der von Interesse ist, ist die adaptive Führung der Lernenden zu einem bestimmten Lernziel. Die Lernplattform *IKnow!*, die Sprachkurse anbietet, versorgt den Lernenden jeweils mit zielgerichteten Angeboten – für die Spielplattform von Bedeutung sind die Methoden und die Werkzeuge, mit der die passenden Inhalte für den Lerner ermittelt werden.

2.6 Bildende (Spiel-)Entwicklungsumgebungen

Punkt 5 der Aufzählung 1, das Erstellen und Beitragen eigener Artefakte, wird in zahlreichen Beispielen zu Spielentwicklungsumgebungen oder Entwicklungsumgebungen unterstützt. Diese benötigen geringe Einstiegskenntnisse und können auch zur Erlangung von Programmierkenntnissen im Unterricht eingesetzt werden. Solche Entwicklungsumgebungen sind zwar besonders einfach zu bedienen, in einigen Fällen – wie dem von *NetLogo* – sind sie gleichzeitig auch ein mächtiges Werkzeug zur Unterstützung des Entwicklers bei der Lösung von anspruchsvollen Problemen. Teilweise sind sie speziell auf die Erstellung von Spielen ausgerichtet, meistens erlauben sie die Lösung von Programmieraufgaben.

Kennzeichen derartiger Entwicklungsumgebungen sind:

- **Eigene GUI:** Es gibt eine benutzerfreundliche und unterstützende graphische Benutzeroberfläche.
- **Start auch ohne Programmierkenntnisse:** Die Entwicklungsumgebung soll auch durch Benutzer ohne Programmierkenntnisse genutzt werden können. Bei der Erstellung von Spielen lernen diese dann zu programmieren.

Gewöhnlich besitzen diese Spielentwicklungsplattformen keinen gemeinsamen thematischen Bogen. Sie sind jedoch ein gutes Beispiel für die Möglichkeit der Erstellung eigener Inhalte.

2.7 Spielentwicklungsumgebungen

Neben den Entwicklungsumgebungen, die in erster Linie das Erlernen von Programmierfähigkeiten unterstützen sollen, sind auch solche zu finden, deren Schwerpunkt in der Erstellung von Computerspielen liegt. Im Unterschied zu den in Kapitel 2.6 betrachteten Entwicklungsumgebungen bieten auf Spiele spezialisierte Entwicklungsumgebungen größere Unterstützung zur Erstellung von spieltypischen Inhalten. Dazu gehören graphische Effekte - wie Partikelsystem⁶ – und akustische Effekte. Im Allgemeinen ist die Simulation physikalischer Phänomene, wie beispielsweise Schwerkraft und Festkörpermechanik, schon integriert. Sie erlaubt die Erstellung realitätsnäherer virtueller Welten.

Eines der bekanntesten Beispiele für eine Spielentwicklungsumgebung ist *GameMaker*: Ursprünglich auch als Ausbildungswerkzeug konzipiert (Overmars, 2005), hat diese Spielentwicklungsumgebung inzwischen einen Reifegrad erlangt, die sie auch als IDE für kommerzielle Spiele qualifiziert.

Bezüglich der Anforderungen aus der Aufzählung 1 unterstützen Spielentwicklungsumgebungen hervorragend den Spielaspekt (Punkt 1). Auch ermöglichen sie dem Benutzer, eigene Inhalte zu erzeugen (Punkt 4). Zur Konzentration auf spezifische Inhalte einer Fachdomäne wäre die Erstellung einer eigenen Bibliothek an (Fach-)Objekten nötig. Die Erstellung von Simulationsmodellen ist in der Regel ebenfalls nicht eingebaut, hier wäre die Erstellung eines eigenen Frameworks notwendig.

⁶ Ein Partikelsystem (engl. particle emitter) ist ein (grafischer) Animationseffekt, bei dem eine große Anzahl von kleinen Objekten von einer Quelle ausgestoßen wird. Diese nehmen einen konfigurierbaren Weg durch den Raum. Beispiele sind die Darstellungen von Rauch und Explosionen.

2.8 Simulationsframeworks

Simulation, der Punkt 2 der Aufzählung 1, wird besonders unterstützt durch die zahlreich anzutreffenden Simulationsframeworks im Sinne von speziellen Entwicklungsumgebungen für Simulationen und Simulationsmodelle⁷. Vornehmlich arbeiten solche Frameworks im Bereich der diskreten Ereignissimulation, da diese sich relativ gut formalisieren lässt - im Vergleich zur kontinuierlichen Simulation. Oft sind solche Frameworks unterteilt in eine Laufzeitumgebung („Simulation-Engine“), die für die Ausführung der Simulationen zuständig ist und in eine Modellierungsumgebung, die eine graphische Benutzeroberfläche für die Erstellung von Simulationsmodellen bietet.

Häufig bieten Simulationsframeworks eine Bibliothek an Modellen und Teilmodellen, die für weitere Simulationen wiederverwendet werden können. Ebenso sind die Simulationsframeworks teilweise auf bestimmte Klassen von Simulationsmodellen spezialisiert. Beispiel ist hier *OM-Net++*, das sich besonders zur Simulation von Kommunikationsnetzwerken eignet.

Simulationsframeworks zeichnen sich oftmals durch eine ansprechende Benutzeroberfläche aus. Aus der Liste der wünschenswerten Eigenschaften fehlt gewöhnlich das Spiel-Attribut.

2.9 Simulationsplattformen

Ein Simulationsframework wird gewöhnlich auf einem Rechner, den der Benutzer steuern kann, installiert. Darüber hinaus gibt es auch Websites, die sowohl als Modellierungsumgebung als auch als Laufzeit- und Präsentationsumgebung für Simulationen dienen. Hier ist keine lokale Installation von Software nötig, und die Simulationen können im Web zur Verfügung gestellt werden. Zusätzlich können durch die Bereitstellung auf einem Server mehrere Teilnehmer in einer Simulation agieren. Eingeschränkt ist jedoch die Bereitstellung von simulationsspezifischen Erweiterungen, da diese projektbezogen auf dem zentralen Server integriert werden müssen. Aufgrund des verwandten Zweckes der Software gelten für die Eignung bezüglich der Anforderungsliste dieselben Ausführungen wie sie schon im vorigen Kap. 2.8 für Simulationsframeworks gemacht wurden.

Forio ist ein Beispiel für eine solche Simulationsplattform. Sie enthält sowohl eine Entwicklungsumgebung als auch eine Präsentationsumgebung, in der der Benutzer die Simulationen starten kann und in der diese visualisiert werden.

2.10 Simulationsspiele

Jedes Computerspiel ist eine Simulation. Das dem Spiel zugrundeliegende Modell wurde in Programmcode umgesetzt und wird während zur Ausführungszeit simuliert. Wenn das Basismodell des Spiels eine Abbildung eines weiteren, außerhalb der Spielumgebung – gewöhnlich in der Realität - existierenden Modells ist, dann wird das Spiel Simulationsspiel genannt. Simulationsspiele sind somit eine Untermenge der Computerspiele (s.a. Abbildung 2)⁸.

⁷ Eine umfassende Sammlung von Simulationsframeworks ist zu finden unter <http://www.topology.org/soft/sim.html> (Letzter Zugriff: 15.08.2012).

⁸ Jones, Piccard & Jones (1995, S. 21) zitieren die folgende Definition für ein Simulationsspiel: A simulation-game combines the features of a game (competition, co-operation, rules, players) with those of simulation (incorporation of features of the real world).

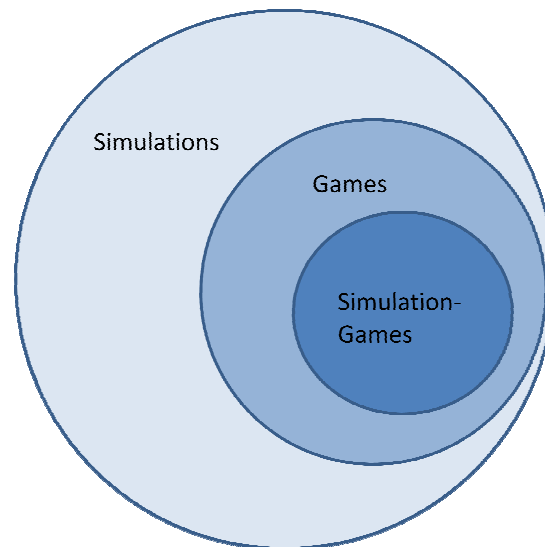


Abbildung 2 The Technical View of the Simulations - Games Relationship nach Becker & Parker, 2011, fig. 3–3

Simulationsspiele erfüllen somit per Definition die Punkte 1 und 2 der Aufzählung 1: Sie arbeiten mit Simulation und enthalten Spielmechaniken. Beispiele für solche Simulationsspiele sind *MS Flight Simulator*, *SimCity* und *Civilization*. Diese Spiele zeigen auch, dass der Grad der Realitäts-treue einer Simulation unterschiedlich sein kann und vom Ziel der Simulation abhängt.

Simulationsspiele sind sehr geeignet als Lernmedium, auch für die Ausbildung im Studium (Hertel & Millis, 2002). Ein umfassendes Computerspiel für die Domäne der Bauphysik konnte jedoch im Rahmen der Recherche nicht gefunden werden.

2.11 Simulatoren

Flugsimulatoren stehen für eine spezielle Klasse von Simulationen: Sie werden in unterschiedlichen Kontexten genutzt: Einmal als Mittel zur Ausbildung (Macedonia, 2001) - unterstützt u.a. durch die hohe Detailgenauigkeit -, desweiteren aber auch als Freizeitaktivität im Sinne eines Spiels. Damit werden in idealer Weise die Punkte Ausbildung, Simulation und (Spiel-)Spaß miteinander verknüpft.

Oft lassen sich Simulatoren erweitern: Beispielsweise gewinnt der Flugsimulator durch die Bereitstellung neuer Flugzeugtypen und Flughäfen an Vielseitigkeit.

2.12 Crowdsourcing-basierte Spiele

Punkt 5 umfasst Erweiterungen durch den Spieler: Die Produkte der Bemühungen der Spieler können zur Erweiterung des Spieles genutzt werden. Dieses ist das Prinzip des Crowdsourcing: die Erfüllung einer Aufgabe wird auf eine Masse von Freiwilligen unter Nutzung des Internets übertragen (Howe, 2006). Beispiele für Spiele, in denen Crowdsourcing erfolgreich eingesetzt wird, sind FoldIt und EteRNA. Bei FoldIt geht es darum, die räumliche Struktur von Proteinen zu ermitteln, deren Molekülzusammensetzung bekannt ist. Dem Spieler werden mit ernstem wissenschaftlichem Hintergrund Molekülketten vorgesetzt, die nach bestimmten Regeln gefaltet werden sollen. Für die Lösung gibt es einen Bewertungsmaßstab, so dass sie vergleichbar ist. Die Aufgabe, die an die Freiwilligen verteilt wird, ist das Finden einer Lösung zu einem gegebenen Puzzle mit einer möglichst hohen, nach objektiven Maßstäben ermittelbaren Bewertungszahl.

Eine weitere Stufe des Crowdsourcing ist es, wenn die Produkte eines Spielers Aufgaben für die anderen Spieler darstellen und somit das Spiel erweitern. Ein übliches Hilfsmittel für eine solche Erweiterung ist ein Level-Editor: Mit dessen Hilfe kann können neue Aufgaben für andere Spieler geschaffen werden. Vorteile dieser Stufe des Crowdsourcings sind die geringeren Kosten für das Gesamtsystem.

3 Softwareübersicht

Tabelle 1 enthält eine Übersicht der im Rahmen der Recherche näher untersuchten Softwarepakete und der jeweils zugeordneten Kategorie aus Kapitel 2. Die Softwarepakete sind alphabetisch geordnet und in den Zeilen aufgeführt. Die Kategorie ist mit einem „X“ markiert.

Kategorie \ Software	Spielportal	Physik-affines Spiel	Fachspezifisches Portal	Simulationsportal	Lernplattform	Bildende (Spiel-) IDE	Spiel-IDE	Simulationsframework	Simulationsplattform	Simulationsspiel	Simulator	Crowdsourcing ⁹
AgentSheets						X						
Alice						X						
Angry Birds		X										
Civilization										X		
DESMO-J								X				
easy java simulations								X				
EteRNA												X
Etoys						X						
ExploreLearning				X								
ExtendSim								X				
Flight Simulator X											X	
FoldIt												X
Forio									X			
GameMaker							X					
iKnow!					X							
Kodu Game Lab						X						
Nanohub			X									
NASA Simulation – Interactive Media											X	
NetLogo						X						
OKSIMO								X				
OMNet++								X				
Physics Games on freewebarcade.com	X											
Portal			X									
Ptolemy								X				
Scratch						X						
SimCity										X		
World of Goo			X									

Tabelle 1: Kategorisierung der gefundenen Softwarepakete

Tabelle 3 zeigt eine zusammenfassende Wertung bezüglich der von der Spielplattform geforderten Aspekte aus der Aufzählung 1. Es wurde eine 3-teilige Wertung vorgenommen, die in Tabelle 2 beschrieben ist.

Wertung	Beschreibung	Symbol
Hauptaspekt	Das ist der Schwerpunkt der betrachteten Software.	X
Nebenaspekt	Diesen Aspekt kann die Software teilweise erfüllen.	o
Keine Unterstützung	Der Aspekt wird von der Software nicht oder nur zu einem geringen Grad unterstützt.	-

Tabelle 2 Grad der Unterstützung eines Aspekts

⁹ Crowdsourcing-basierte Spiele

Da die Wertungen subjektiv nur durch eine Person erfolgten, sind sie nicht absolut und endgültig. Sie sind jedoch ein Hinweis darauf, dass keines der gefundenen Softwarepakete alle Anforderungen abdecken kann.

Aspekt Software	Spiel	Simulation	Gemeinschaft	Bauphysik-Inhalte	Erweiterbarkeit	Integrierte Plattform
AgentSheets	O			O	X	
Alice	O			O	X	
Angry Birds	X	O		O		O
Civilization	X	O			O	O
DESMO-J		X			X	
easy java simulations		X		O	X	
EteRNA		X	O		O	O
Etoys	O				X	
ExploreLearning		X	O			
ExtendSim		X				
Flight Simulator X	O	X			O	O
FoldIt		X	O		O	O
Forio		X	O	O	X	
GameMaker	O		O		X	
iKnow!			O			X
Kodu Game Lab	X				X	
Nanohub			X			X
NASA Simulation – Interactive Media		X				O
NetLogo	O	O	O		X	
OKSIMO		X			X	
OMNet++		X			X	
Physics Games on freewebarcade.com	X	O		O		
Portal	X	O		O		
Ptolemy		X			X	
Scratch	O				X	
SimCity	X	X				O
World of Goo	X	O		O		O

Tabelle 3 Software und unterstützte Aspekte (X: Hauptaspekt, O: Nebenaspekt)

4 Softwarebeschreibungen

In diesem Kapitel werden die betrachteten Softwarepakete einzeln beschrieben. Gleichzeitig wird neben Hinweisen auf Stärken, Schwächen und beispielhaften Aspekten auch eine persönliche Einschätzung formuliert, inwieweit sich die Software für den gegebenen Einsatzzweck eignet.

Name	Bemerkungen
AgentSheets	<p>Beschreibung: AgentSheets ist eine Entwicklungsumgebung, mit der sich Simulationen, Simulationswerkzeuge und Spiele erzeugen lassen. Gleichzeitig kann mit ihr Programmentwicklung gelernt werden. Ein wesentlicher Zweck ist die schnelle Entwicklung von Programmen für (Noch-)Nicht-Programmierer („Thought Amplifier“). Die Hauptzielgruppe ist die des K-12¹⁰. Die Ergebnisse lassen sich für den Zugriff über das Internet exportieren.</p> <p>AgentSheets wird von der gleichnamigen Firma unter der Leitung von Dr. Alexander Repening in Boulder im US-Bundesstaat Colorado hergestellt, es ist im universitären Umfeld entstanden. Die aktuelle Entwicklung nennt sich <i>AgentCubes Creativity</i>^{3D}.</p> <p>Einschätzung¹¹: Der Schwerpunkt liegt eher auf einzelnen Spielen, die Spiele sind von Grund auf nicht miteinander verbunden. Es ist keine Plattform für Spiele eines Themas – die gemeinsame Basis fehlt.</p> <p>Website: http://www.agentsheets.com/¹²</p>
Alice	<p>Beschreibung: Alice ist ebenfalls eine Entwicklungsumgebung zum Erlernen von ersten Programmierkenntnissen. Sie wird entwickelt an der Carnegie Mellon University (Pittsburgh, US-Bundesstaat Pennsylvania)¹³.</p> <p>Einschätzung: Auch hier liegt der Schwerpunkt auf der Entwicklung von einzelnen Programmen und Spielen. Es fehlt die gemeinsame fachliche Basis für eine Spielplattform.</p> <p>Website: http://www.alice.org/</p>
Angry Birds	<p>Beschreibung: Angry Birds ist ein populäres Computerspiel aus dem Genre <i>Artillery</i>¹⁴ des finnischen Computerspielherstellers Rovio: Der Spieler schießt mit Vögeln auf Schweine und versucht diese durch Wahl des richtigen Anstellwinkels sowie der richtigen Feuerkraft zu zerstören. Das Spiel ist insbesondere geeignet für Geräte mit Touchbedienung, wie z.B. das iPad von Apple.</p> <p>Einschätzung: Angry Birds beschäftigt sich mit einem Teilbereich der Physik, der Ballistik. Die Ausweitung auf andere Effekte ist im Spielkonzept zunächst nicht vorgesehen. Ein Level-Editor zur Beteiligung der Spieler wäre sehr gut möglich, es gibt darüber erste Berichte (Anne, 2012).</p> <p>Website: http://www.rovio.com/en/our-work/games/view/1/angry-birds</p>

¹⁰ In den USA ist K-12 eine zusammenfassende Bezeichnung für das formale Bildungssystem vom Kindergarten bis zur 12. Schulklasse (gesprochen: „Kay through twelve“).

¹¹ Die Einschätzung stellt eine Diskussion dar, inwieweit die Software geeignet ist, die Anforderungen einer erweiterbaren Spielplattform zu erfüllen.

¹² Sofern nicht anders vermerkt, wurde auf die in dieser Tabelle genannten Websites zuletzt während der Endredaktion dieses Textes am 7. und 8. August 2012 zugegriffen.

¹³ Das Projekt Alice wurde mitbegründet von Prof. Randolph „Randy“ Frederick Pausch, der mit seinem Buch „Last Lecture“ und seiner darin beschriebenen Erkrankung an Bauchspeicheldrüsenkrebs weltweit Anteilnahme erregte.

¹⁴ Aufgabe von Artillery-Spielen ist es, gegnerische Stellungen mit Projektilen zu zerstören. Die Herausforderung für den Spieler ist die richtige Bestimmung der Schuss-Parameter, also der Umgang mit ballistischen Sachverhalten (Burton, 2003).

Civilization	<p>Beschreibung: Civilization ist ein rundenbasiertes Strategiespiel, in dem der Spieler ein Volk durch die Geschichte lenken muss. Der Spieler kann u.a. Städte ausbauen, in Forschung investieren und Diplomatie betreiben mit dem Ziel, größer und stärker zu werden als die Nachbarvölker. Die erste Version des Spiels wurde von Sid Meier 1991 veröffentlicht. 2010 wurde die bisher letzte Weiterentwicklung <i>Civilization 5</i> herausgegeben</p> <p>Einschätzung: Civilization lässt sich mit eigenen Szenarien, sogenannten Mods ergänzen. Diese können genutzt werden, um die Erreichung spezieller Lernziele zu unterstützen. Kurt Squire benutzte einen Mod für Civilization III im Rahmen des <i>Social Studies</i>-Unterrichts und zeigte damit, wie die Faszination kommerzieller, zu Unterhaltungszwecken erstellter Spiele auf die Vermittlung von schulrelevanter Lernziele ausgedehnt werden kann (Squire, 2003).</p> <p>Bisher konnte im Rahmen dieser Arbeit kein Spiel entdeckt werden, das modifiziert Probleme der Bauphysik enthalten kann. Ansatzweise wurde das mit Minecraft versucht (Großmann, 2012).</p> <p>Website: http://www.civilization.com/</p>
DESMO-J	<p>Beschreibung: DESMO-J („Discrete-Event Simulation and MOdelling in Java“) ist ein Framework zur Entwicklung von ereignisorientierten Simulationen in der Programmiersprache Java. Es wird entwickelt an der Universität Hamburg im Fachbereich Informatik unter der Leitung von Prof. Dr.-Ing. Bernd Page.</p> <p>Einschätzung: DESMO-J ist zunächst ein Framework für die Diskrete-Event-Simulation. Es ist also nicht geeignet für die kontinuierliche Simulation wie sie für die Abbildung physikalischer Effekte benötigt wird. Als Simulationsframework ist DESMO-J im Rahmen dieser Arbeit aufgefallen, weil die Entwicklung einer Eclipse-Version angekündigt war. (Stempell, 2008). In der aktuellen Version wird Eclipse jedoch lediglich als Entwicklungsumgebung unterstützt, die Architektur des Systems beruht nicht auf OSGi.</p> <p>Website: http://desmoj.sourceforge.net/home.html</p>
Easy Java Simulations	<p>Beschreibung: Easy Java Simulations (EJS) ist ein auf der Programmiersprache Java basierendes Werkzeug zur Erstellung physikalischer Simulationen zu Lehrzwecken. Es wurde von Prof. Francisco Esquembre von der Universität Murcia in Spanien im Rahmen des <i>open source physics</i> (OSP)-Projektes¹⁵ ins Leben gerufen.</p> <p>Einschätzung: Ähnlich wie <i>AgentSheets</i> lassen sich mit Hilfe dieses Werkzeuges Simulationen im Bereich Physik erstellen, es fehlt jedoch eine gemeinsame Basis und Verknüpfung der verschiedenen Szenarios.</p> <p>Website: http://www.um.es/fem/EjsWiki/index.php/Main/-WhatIsEJS?</p>
EteRNA	<p>Beschreibung: EteRNA ist eine browserbasierte Anwendung, die erstellt wurde von der Carnegie Mellon Universität und der Stanford Universität. Die Spieler finden räumliche Anordnungen von RNA¹⁶-Molekülen und haben dabei aufgrund von Intuition und Intelligenz wesentliche Geschwindigkeitsvorteile gegenüber Computersimulationen.</p> <p>Einschätzung: Das Spiel beruht – ebenso wie FoldIt – auf dem Prinzip des Crowdsourcing: Mit Hilfe der Intelligenz formal unbeteiligter Personen werden Lösungen für gegebene Probleme gefunden. Die Erweiterung auf bauphysikalische Problemstellungen ist erst einmal nicht möglich.</p> <p>Website: http://eterna.cmu.edu/</p>

¹⁵ Website: <http://www.compadre.org/osp/>

¹⁶ Ribonukleinsäure (engl. ribonucleic acid) ist ein Träger genetischer Informationen in der (biologischen) Zelle.

Etoys	<p>Beschreibung: Etoys ist eine objektorientierte Programmiersprache und eine Entwicklungsumgebung zu Lehrzwecken. Sie wurde u.a. von Alan Kay entwickelt und erschien 1996 zum ersten Mal.</p> <p>Einschätzung: Ähnlich wie bei <i>Alice</i> und <i>Scratch</i> liegt der Schwerpunkt von Etoys auf den Einsatz als Lernwerkzeug. Die einzelnen Szenarios müssen nicht fachspezifisch miteinander verbunden sein und es fehlt eine gemeinsame Basis und Spielumgebung.</p> <p>Website: http://www.squeakland.org/</p>
ExploreLearning	<p>Beschreibung: ExploreLearning ist ein Portal, das zu Lehrzwecken von der gleichnamigen Firma betrieben wird. Es stellt sogenannte Gizmos bereit: das sind Simulationen aus dem mathematischen und naturwissenschaftlichen Bereich mit den Klassen 3 bis 12 als Zielgruppe.</p> <p>Einschätzung: Die einzelnen Simulationen stehen nebeneinander, es gibt keine gemeinsame Einbettung. Die Erstellung eigener Simulationen durch die Lerner ist nicht möglich.</p> <p>Website: http://www.explorelearning.com/</p>
ExtendSim	<p>Beschreibung: ExtendSim ist ein Simulationsprogramm für diskrete, kontinuierliche und agentenbasierte Simulationen. Es wird von der Firma <i>Imagine That Inc.</i> hergestellt und vertrieben.</p> <p>Einschätzung: Kontinuierliche Simulationen eignen sich zur Darstellung physikalischer Vorgänge, es fehlt jedoch der Charakter einer Spielplattform.</p> <p>Website: http://www.extendsim.com/</p>
Flight Simulator X	<p>Beschreibung: Im Microsoft Flight Simulator ist der Spieler der Pilot eines Flugzeuges. Er hat verschiedene Missionen zu erfüllen. Durch die relativ detailgenaue Nachbildung von Flugzeugen und Flughäfen kann dieses Simulationsspiel auch zu Ausbildungszwecken genutzt werden. Die erste Version erschien schon 1982, die aktuelle Version ist MS Flight Simulator X. Die Entwicklung einer Free-to-play-Version (<i>MS Flight</i>) wurde im Juli 2012 eingestellt (Gieselmann, 2012).</p> <p>Einschätzung: MS Flight Simulator ist ein gutes Beispiel für ein Simulationsprogramm, das sowohl als Spiel als auch als Ausbildungswerkzeug genutzt wird. Darüber hinaus ist es möglich, Inhalte für den MS Flight Simulator zu erzeugen, es werden sowohl Editoren als auch APIs¹⁷ angeboten.</p> <p>Gegen einen Einsatz als Spielplattform spricht die fehlende Assessment-Komponente sowie fachliche Fokussierung auf den Flugbereich.</p> <p>Website: http://www.microsoft.com/games/flightsimulatorx/</p>
FoldIt	<p>Beschreibung: FoldIt ist - ebenso wie <i>EteRNA</i> – ein Crowdsourcing-basiertes Spiel mit wissenschaftlichem Hintergrund, entwickelt an der <i>University of Washington</i> in Seattle im US-Bundesstaat Washington: Aufgabe des Spielers ist es, vorgegebenen Eiweiß-Molekülketten nach bestimmten „Faltungsregeln“ eine räumliche Struktur zu geben, die diese Moleküle in der Natur einnehmen. Auch hier wieder ist menschliche Intuition und Intelligenz der Simulation durch Computerprogramme zeitlich und fachlich überlegen.</p> <p>Einschätzung: Das Prinzip des Crowdsourcing funktioniert bei dieser Software sehr gut, jedoch ist sie fachlich begrenzt und nicht auf die Bauphysik erweiterbar.</p> <p>Website: http://fold.it/portal/</p>
Forio	<p>Beschreibung: Forio ist eine Simulationssoftware, die es einem Benutzer ermöglicht, selbst zu programmierende Simulationen mit einer gestaltbaren und ansprechenden Benutzeroberfläche online zu präsentieren. Dieser Webservice wird von der Firma Forio Online Simulations betrieben.</p> <p>Einschätzung: Forio sticht aus der Menge der betrachteten Software heraus durch die Realisierung als Webservice (SaaS). Es fehlt die Infrastruktur einer Spielplattform.</p> <p>Website: http://forio.com/</p>

¹⁷ API (Application Programming Interface): definierte (und) dokumentierte Programmierschnittstellen einer Software, die es Dritten erlauben, Erweiterungen für diese Software zu programmieren.

GameMaker	<p>Beschreibung: GameMaker ist eine spezielle Entwicklungsumgebung für Spiele – entworfen und entwickelt durch Mark Overmars, Professor an der Universität Utrecht in den Niederlanden. GameMaker wird auch als Lernwerkzeug eingesetzt und ermöglicht die Entwicklung von Spielen auch mit nur geringen Programmierkenntnissen durch eine grafische Benutzeroberfläche. Es besitzt einen Upload-Bereich, in dem die Designer ihre Spiele anderen Spielern zur Verfügung stellen können. Zusätzlich ist es auch um binäre Bibliotheken¹⁸ erweiterbar.</p> <p>Einschätzung: GameMaker ist eine ausgewachsene Spielentwicklungsumgebung. Um es als Spielplattform einzusetzen, ist die Definition einer gemeinsamen Spielinfrastruktur notwendig sowie die Entwicklung einer fachspezifischen Bibliothek für bauphysikalische Objekte. Vermisst wird allerdings ein OSGi(-ähnliches) Komponentenmodell, das den Entwickler konstruktiv und auf Modellebene bei der Einhaltung und Nutzung von Entwicklungsprinzipien wie Erweiterbarkeit und Modularität unterstützt.</p> <p>Website: http://www.yoyogames.com/gamemaker</p>
iKnow!	<p>Beschreibung: iKnow! ist eine Lernplattform für Kurse in chinesischer oder japanischer Sprache. Sie zeichnet sich aus durch starke Steuerung des Lernenden. Diese erfolgt durch Einstufungstests, durch Wochenpläne, durch Statistiken und durch Badges und Achievements, die dem Lerner verliehen werden.</p> <p>Einschätzung: Diese Website ist ein Beispiel für die Steuerung des Lernenden. Die anderen benötigten Eigenschaften wie Erzeugung eigener Inhalte sowie spielerische Elemente fehlen hier jedoch.</p> <p>Website: http://iknow.jp/</p>
Kodu Game Lab	<p>Beschreibung: Das Kodu Game Lab ist eine integrierte Entwicklungsumgebung für Spiele von den Microsoft FUSE Labs. Sie kann auch durch einen Game Controller bedient werden, benötigt also keine Tastatur und läuft u.a. auch auf der XBOX, der Spielkonsole von Microsoft. Das Kodu Game Lab kann wie <i>AgentSheets</i>, <i>Alice</i> und andere Spielentwicklungsumgebungen eingesetzt werden, um Schülern erste Programmiererfahrungen zu verschaffen.</p> <p>Einschätzung: Dies ist eine Entwicklungsumgebung für Spiele. Wie auch bei <i>AgentSheets</i> und <i>Alice</i> fehlt die gemeinsame Infrastruktur für eine Spielplattform.</p> <p>Website: http://www.kodugamelab.com/</p>
Nanohub	<p>Beschreibung: Nanohub ist ein Webportal zum Thema Nanotechnologie. Neben einer Social-Network-Komponente und Materialien zum Thema Nanotechnologie bietet es auch einige Simulationswerkzeuge für dieses Fachgebiet an.</p> <p>Einschätzung: Nanohub ist zwar keine Spielplattform, es ist aber ein exzellentes Beispiel für eine fachlich fokussierte Plattform¹⁹. Beiträge von anderen Teilnehmern sind willkommen, sofern sie in den fachlichen Rahmen passen. Es gibt viele weitere Beispiele für fachlich fokussierte Webportale, einige von ihnen basieren auf der Software <i>HUBzero</i> (http://hubzero.org/) von der <i>Purdue University</i>, auf der auch Nanohub aufbaut.</p> <p>Website: http://nanohub.org/</p>
NASA Simulation – Interactive Media	<p>Beschreibung: Diese Simulationen der NASA sind in einen Lehrplan eingebettet. Sie simulieren fachspezifische Dinge wie beispielsweise einen Raketenstart.</p> <p>Einschätzung: Die Simulationen demonstrieren sehr fachspezifische Dinge. Sie sind nicht erweiterbar.</p> <p>Website: http://www.nasa.gov/offices/education/programs/national/-summer/education_resources/engineering_grades7-9/E_simulations.html</p>

¹⁸ Sogenannte DLLs (Dynamic Link Libraries) unter Microsoft Windows

¹⁹ Ein Unterschied zu fachlichen Portalen wie www.physics.org liegt in der Möglichkeit, auch eigene Inhalte beizutragen.

NetLogo	<p>Beschreibung: NetLogo ist sowohl eine Programmiersprache als auch eine Entwicklungsumgebung. Es ist eine Weiterentwicklung der funktionalen Programmiersprache Logo. Logo wurde u.a. von Seymour Papert geschaffen als Programmiersprache, die es auch Nicht-Programmierern erlaubt, Programme zu erstellen und bei der Erstellung von Artefakten Lernerfahrungen zu machen (Konstruktivismus (Papert, 1993)). NetLogo wird unter der Leitung von Prof. Uri Wilensky an der <i>Northwestern University</i> in Chicago im US-Bundesstaat Illinois seit 1999 entwickelt. Der Grundsatz "low threshold and no ceiling" verdeutlicht, dass diese Anwendung keine Einstiegshürden aufbauen will, gleichzeitig aber auch fortgeschrittenen Entwicklern ein mächtiges Werkzeug sein möchte.</p> <p>Einschätzung: NetLogo hat eine hohe Verbreitung. Es ermöglicht seinen Benutzern, Inhalte zu erzeugen. Diese können anderen Benutzern als Modelle zur Verfügung gestellt werden. Es unterstützt diskrete und kontinuierliche Simulationsmodelle. Nicht vorhanden ist die Infrastruktur einer Spielplattform.</p> <p>Website: http://ccl.northwestern.edu/netlogo/</p>
OKSIMO	<p>Beschreibung: OKSIMO ist die Abkürzung für <i>Open Knowledge Simulation Modeling</i>. Es ist eine Simulationsumgebung, die als Open Source Software unter Führung des <i>Frankfurter Instituts für neue Medien</i> (INM, Prof. Döben-Henisch) entwickelt wird. Vornehmlich werden diskrete Simulationsmodelle unterstützt. Zentrale Komponente ist ein Modellkatalog, in dem die mit einem Modell-Editor zusammengestellten Modelle öffentlich gespeichert werden können.</p> <p>Einschätzung: Beispielhaft ist der Ansatz einer Austauschplattform für Modelle. Hier kann der Benutzer selbst erstellte Modelle zur allgemeinen Verfügbarkeit unterbringen. Fehlend in dieser Software ist die Spielinfrastruktur.</p> <p>Website: http://oksimo.inm.de/</p>
OMNet++	<p>Beschreibung: OMNet++ ist eine in C++ geschriebene Open Source²⁰-basierte Simulationsumgebung für ereignisorientierte Simulationen: Sie eignet sich insbesondere zur Nachbildung von Netzwerken, im Wesentlichen sind dies Kommunikationsnetzwerke, aber auch Warteschlangenmodelle sind möglich. Neben einer auf Eclipse aufbauenden Entwicklungsumgebung gibt es eine grafische Laufzeitumgebung. OMNet++ ist Grundlage einer Reihe von weiteren Simulationsprojekten.</p> <p>Einschätzung: OMNet++ ist ein gutes Beispiel für ein modulare Simulationsframework, es können auch Module anderer Programmiersprachen eingebunden werden. Die über eine Erweiterung für die Eclipse-Plattform erstellte IDE ist ein Beleg für die Belastbarkeit der Eclipse RCP Software. Es fehlt die spielerische Komponente.</p> <p>Website: http://www.omnetpp.org/</p>
Physics Games on freewebarcade.com	<p>Beschreibung: Unter der Rubrik „Physics Games“ sind auf freewebarcade.com eine Reihe von browserspielbaren Spielen verfügbar, die in der Regel jeweils einen physikalischen Aspekt in ihre Spielmechanik eingebunden haben.</p> <p>Einschätzung: Es sind unterschiedliche Physik-basierte Spiele, die jedoch jedes für sich allein stehen – es fehlt ein übergreifender Rahmen.</p> <p>Website: http://www.freewebarcade.com/physics-games.php/²¹</p>

²⁰ Für kommerzielle Verwendung ist eine Lizenz notwendig.

²¹ Es gibt eine Reihe von weitere Sammlungen von physik-affinen Spielen, beispielsweise unter www.physicsgames.net oder www.physics-games.net.

Portal	<p>Beschreibung: Portal (bzw. die Nachfolgerversion Portal 2) ist ein Computerspiele des Genres <i>Action-Adventure</i>²². Auf seinen Missionen steuert der Spieler einen Avatar und löst dabei Rätsel, die mit Hilfe der Spielmechanik und des theoretischen Konstrukts der Teleportierung angegangen werden. Die Teleportierung selbst unterliegt physikalischen Gesetzmäßigkeiten, beispielsweise bleibt die Geschwindigkeit konstant.</p> <p>Einschätzung: Portal ist ein Computerspiel, das in erster Linie zur Unterhaltung gedacht ist. Gleichwohl werden durch die Integration physikalischer Gesetzmäßigkeiten diese dem Spieler vermittelt. Für den Spielerfolg muss er diese anwenden und damit auch lernen. Durch einen Mehrspielermodus wird auch der Austausch in der Gruppe gefördert. Ein Level-Editor erlaubt das Erzeugen eigener Artefakte. Leider ist das Spiel thematisch beschränkt, d.h. weitere Effekte der Physik lassen sich nicht integrieren.</p> <p>Website: http://www.thinkwithportals.com/</p>
Ptolemy	<p>Beschreibung: Ptolemy II ist eine Simulationsumgebung für Computersysteme mit erhöhten technischen Anforderungen wie z.B. Nebenläufigkeit und Echtzeit oder eingebettete Systeme, die an der <i>University of California, Berkeley</i>, entwickelt wird. Schwerpunkte der Arbeit sind eine komfortable Modellierungsumgebung und die Hierarchisierung von Modellen aus unterschiedlichen Quellen.</p> <p>Einschätzung: Ptolemy zeichnet sich aus durch die Integration von heterogenen Modellen. Das ist eine Notwendigkeit unter der Annahme, dass es das allumfassende Modellierungswerkzeug oder Modell niemals geben wird. Es ermöglicht damit die Nutzung der Ergebnisse vieler Beitragender. Auch unterstützt es verschiedene Arten der Simulation, neben ereignisorientierter auch kontinuierliche Simulation.</p> <p>Es fehlt allerdings die spielerische Komponente.</p> <p>Website: http://ptolemy.eecs.berkeley.edu/</p>
Scratch	<p>Beschreibung: Scratch ist eine visuelle Entwicklungsumgebung, mit deren Hilfe erste Programmierkenntnisse vermittelt werden können. Es wurde entwickelt von der MIT Media Lab Lifelong Kindergarten Group. Unter dem Slogan "imagine, program, share" hat sich eine Gemeinschaft entwickelt, die die erstellten Produkte – u.a. auch Spiele – in einem Portal der Allgemeinheit zur Verfügung stellt. Die für 2012 angekündigte Version 2.0 wird online auf Flash basieren.</p> <p>Einschätzung: Scratch ist auf die Erstellung eigener Produkte ausgelegt und kann auf eine große Gemeinschaft bauen. Die Artefakte existieren jedoch unabhängig voneinander, es fehlt eine gemeinsame fachliche Basis.</p> <p>Website: http://scratch.mit.edu/²³</p>

²² Im Genre *Action* kommt es auf Geschicklichkeit und schnelle Reaktionen des Spielers an, ein Spiel des Genres *Adventure* wird eher von einer Geschichte getrieben. Der Spieler steuert gewöhnlich einen Avatar, der Aufgaben und Rätsel löst (MobyGames, 2012).

²³ Das Portal ist ebenfalls zu finden unter der Projekt-Website <http://scratch.mit.edu/>.

SimCity	<p>Beschreibung: SimCity ist ein Computerspiel, das zum Genre der Wirtschaftssimulationen gehört. Der Spieler hat die Aufgabe, eine Stadt aufzubauen, er muss dabei Faktoren wie Umwelt, Verkehr, Bildung und Kriminalität steuern. Ein komplexes Wirkungsgefüge setzt die Aktionen des Spielers in einen Entwicklungsstatus der Stadt um. Das Spiel existiert seit 1989, wurde von Will Wright entwickelt und erfährt regelmäßige neue Versionen. Die nächste Version SimCity 5 ist für 2013 geplant²⁴. Auf Facebook ist inzwischen die um Spielmechaniken des Social Gaming erweiterte Version <i>SimCity Social</i>²⁵ verfügbar.</p> <p>Einschätzung: SimCity ist eines der erfolgreichsten und damit auch - bezogen auf den Spielspaß – eines der attraktivsten Simulationsspiele, das auch zu pädagogischen Zwecken eingesetzt werden kann (Adams, 1998; Gaber, 2007). Es gibt ebenfalls Erweiterungspakete, mit denen zusätzliche Szenarien gebaut werden können (SimCity, 2010). Jedoch ist der fachliche Rahmen sehr beschränkt, er lässt sich nicht in einfacher Weise auf bauphysikalische Sachverhalte ausdehnen.</p> <p>Website: http://www.simcity.com/</p>
World of Goo	<p>Beschreibung: <i>World of Goo</i> ist eine Puzzle-Spiel. Der Spieler muss der Schwerkraft unterliegende Spielsteine („Balls of Goo“) durch eine Landschaft zu einem Ausgang bewegen. Hierzu ist es mitunter notwendig, aus den Spielsteinen selbst größere Strukturen, wie z.B. Brücken, zu bauen, die dann als Hilfskonstruktionen zum Transport eingesetzt werden können.</p> <p>Einschätzung: <i>World of Goo</i> thematisiert den physikalischen Aspekt der Schwerkraft und ist damit ein Beispiel für ein physikaffines Spiel. Es bietet großen Spielspaß, ist jedoch thematisch beschränkt. Zudem gibt es keinen offiziellen Leveleditor.</p> <p>Website: http://www.worldofgoo.com/</p>

5 Schlussfolgerung

Die Recherche nach einer simulationsbasierten Spielplattform zur Einsatz in der ingenieurwissenschaftlichen Ausbildung brachte zwar eine Vielzahl von verwandter Software zum Vorschein, es konnte aber keine Software gefunden werden, die alle Anforderungen wenigstens teilweise abdecken konnte. Einzelne Softwarepakete konnten als hervorragende Beispiele bestimmter Teilaspekte identifiziert werden, so dass sie dort als Vorbild für die Konzeption einer Spielplattform dienen können.

Zusätzlich gab es auch weitere Erkenntnisse, die Hinweise für das weitere Vorgehen liefern können: Einer dieser Hinweise ist das Finden von zahlreichen Frameworks für ereignisbasierte Simulation: Allein durch die Anzahl der gefundenen Frameworks wurde deutlich, dass es leichter möglich zu sein scheint, ereignisbasierte Simulationsmodelle zu formalisieren als kontinuierliche Simulationsmodelle. Für kontinuierliche Simulationen werden gewöhnlich Spezialentwicklungen durchgeführt.

Eine gemeinsame Spielinfrastruktur fehlt bei den meisten betrachteten Softwarepaketen. Auch wenn Spiele entwickelt werden können, so sind es meistens Einzelentwicklungen, die isoliert nebeneinander stehen. Es findet keine Integration statt.

²⁴ Die Version *SimCity 5* benutzt die speziell entwickelte Simulations-Engine *Glassbox*. Auf ihr können auch weitere Simulationsspiele aufgebaut werden (Cifaldi, 2012).

²⁵ <http://apps.facebook.com/simcitysocial>

Anhang B: Konstruktive Maßnahmen zur Verminderung des Software Aging am Beispiel von bauphysikalischen Simulationsprogrammen

Inhaltsverzeichnis

1 Einführung	2
2 Überblick.....	2
3 Software Aging	2
4 Ansatz der Maßnahmen gegen Software Aging	3
4.1 Änderungen vermeiden.....	3
4.2 Auswirkungen von Änderungen	3
5 Software Wellness.....	5
6 Wie kann Software Wellness durch konstruktive Maßnahmen erreicht werden?	5
6.1 Bewusstsein und Rückhalt auf der Ebene der Entscheidungsträger	5
6.2 Wandlungsorientiertes Design	5
6.2.1 Identifikation der instabilen Systemelemente	6
6.3 Verbesserung des Prozesses der verzerrten Modellaufnahme.....	6
6.3.1 Keine Modellaufnahme	6
6.3.2 Verringerung der Größe des aufzunehmenden Modells	7
6.3.3 Steuerung der Modellaufnahme	8
6.4 Reviews	9
6.4.1 Manuelles Review	9
6.4.2 Automatisiertes Review	9
6.4.3 Automatisierte Tests	9
6.5 Frühes Gegensteuern	9
7 Zukünftige Ansätze	10

1 Einführung

Bei der Beschäftigung mit bauphysikalischen Berechnungs- und Simulationsprogrammen ist uns aufgefallen, dass einige dieser Programme mehr als 30 Jahre alt sind – und immer noch als Stand der Technik eingesetzt werden. Auf den ersten Blick scheint sich das nicht mit der rasanten Entwicklung in der Informatik zu vertragen, es tauchen verschiedene Fragen auf:

- Weshalb wird die Software so alt?
- Was lässt sich machen, damit Software so alt wird?
- Welche konstruktiven Maßnahmen erhöhen die Zeitbeständigkeit von Software?
- Worauf sollte ein Verantwortlicher achten, was sollte er fordern vom Softwareentwicklungsprozess?

Zur Abgrenzung soll erwähnt werden, dass der Begriff „Software“ hier jeweils auf eine Codebasis bezogen verstanden wird. Nicht betrachtet werden solche Softwaregenerationen, die auf Neuentwicklungen beruhen, aber aus Marketing-Gründen mit demselben Namen bezeichnet werden.

Ebenfalls nicht Gegenstand des Artikels ist das Altern von laufender Software durch fehlerhaftes Blockieren von Systemressourcen (z.B. Speicherlecks).

2 Überblick

In Kapitel 3 wird der Begriff Software Aging eingeführt und Ursachen sowie Auswirkungen beschrieben, in Kapitel 4 wird versucht, Ansatzpunkte gegen Software Aging zu finden, wobei das Modell der verzehrten Modellaufnahme genutzt wird. Der Begriff „Software Wellness“ wird in Kapitel 5 eingeführt, im darauf folgenden Kapitel 6 wird – im wesentlichen basierend auf dem Ansatz der verzehrten Modellaufnahme – eine Liste von Maßnahmen erstellt, die die Software Wellness fördern. Kapitel 7 ist Maßnahmen vorbehalten, die noch nicht generell umsetzbar sind.

3 Software Aging

Auf der Suche nach Modellen für das Altern von Software stößt man auf den Begriff des „Software Aging“. Zentrale und prägende Arbeit ist hier „Software Aging“ von David L. Parnas (1994). Dort werden zwei wesentliche Gründe für das Altern von Software identifiziert:

1. Software altert, wenn sie nicht mehr die Bedürfnisse des Benutzers erfüllt bzw. wenn sich die Probleme des Benutzers verändern.
2. Aus dem ersten Grund wird versucht, die Software anzupassen. Anpassungen von Software führen aber zu „brüchigem“ Code, so dass jede Anpassung die folgenden Anpassungen schwerer macht. Zusätzlich wird die Zuverlässigkeit der Software vermindert.

Symptome des Software Aging umfassen u.a. nach Kehler (2008)

- wachsende Komplexität
- unstrukturierten Code
- unkoordiniertes Wuchern von Funktionalität
- ungenügende Dokumentation

4 Ansatz der Maßnahmen gegen Software Aging

4.1 Änderungen vermeiden

Software Aging ist ein Ergebnis des Änderungsdrucks, der aus sich wandelnden Anforderungen des Benutzers an die Software ergibt. Diesem Druck kann man widerstehen, geht dabei aber das Risiko ein, dass sich der Anwender andere Software sucht, sobald diese verfügbar ist und der Nutzen den Lernaufwand übersteigt (Parnas, 1994). In der Regel ist der Änderungsdruck so groß, dass Änderungen an der Software unumgänglich sind.

Gründe für Ausnahmen, also das Weiterbestehen von Software, die wesentliche Defizite gegenüber ihren Anforderungen an hat, könnten sein:

- Es gibt wenig Alternativen:
 - Die Benutzerschaft ist gering – es gibt wenig (wirtschaftliche) Anreize für den Programmierer
 - die zu erledigenden Aufgaben sind sehr anspruchsvoll – die Aufgabe ist für den Programmierer zu komplex.
 - der Aufwand, die Software zu erstellen, ist hoch.
- Der Leidensdruck ist gering
 - die Benutzer kommen eher aus dem technischen Umfeld und werden von einer veralteten Oberfläche und komplizierten Bedienung nicht abgeschreckt.
- Die Verbindung mit dem Produkt ist hoch.
 - In die Software wurde viel Geld und Aufwand investiert.
 - Der Benutzer beherrscht die Technologie und hat vielleicht schon eine Erweiterung für die Software erstellt.

4.2 Auswirkungen von Änderungen

Änderungen von Software sind im Rahmen der Wartung nichts Außergewöhnliches. Bei derartigen Änderungen läuft nach Parnas (1994) der folgende Prozess ab:

- Die Software hat ein bestimmtes Systemmodell¹ SM_0 . Dieses Systemmodell ist entstanden aus einem mentalen Modell² MM_0 des erstellenden Entwicklers E_0 .
- Entwickler E_1 bekommt den Auftrag, die Änderung A durchzuführen.
- Entwickler E_1 versucht nun das Modell SM_0 aufzunehmen und zu verstehen. Dieses gelingt ihm mehr oder weniger weitgehend, er hat danach ein mentales Modell MM_1 in seinem Kopf. Gewöhnlich entsprechen die beiden Modelle einander nicht komplett, sie weichen von einander ab. Ebenso weichen die beiden mentalen Modelle MM_0 und MM_1 voneinander ab.
- Entwickler E_1 ändert die Software gemäß seinem mentalen Modell MM_1 . Da dieses nicht mit dem Systemmodell SM_0 übereinstimmt, entsteht ein neues Systemmodell SM_1 ^{3,4}.

¹Systemmodell: „Die im Anwendungssystem technisch abgebildeten, d.h. programmierten Modelle nennen wir systemtechnische Modelle oder auch Systemmodelle.“ (nach Herczeg (2005, S. 39))

²Ein mentales Modell ist die Repräsentation eines Gegenstandes oder eines Prozesses im Bewusstsein eines Lebewesens (aus: http://de.wikipedia.org/wiki/Mentales_Modell, Abruf am 03.08.2009).

³Hier wird die Änderung des Systemmodells durch die Softwareänderung außer Acht gelassen – wichtig allein ist die Änderung, die dadurch entsteht, dass beide Entwickler (der Ersteller der Software E_0 sowie der ändernde Entwickler E_1) ein unterschiedliches mentales Modell zu Grunde legen.

- Entwickler E_2 bekommt den Auftrag, die Änderung B durchzuführen.
- Entwickler E_2 versucht das Modell SM_1 aufzunehmen, er entwickelt das mentale Modell MM_2 . Dabei hat er es schwerer als Entwickler E_1 das Systemmodell zu verstehen. Denn dieses wurde ja schon durch die Änderungen des Entwicklers E_1 verzerrt, so dass das ursprüngliche Systemmodell SM_0 nicht mehr so klar zu erkennen ist – es ist immer die Frage zu stellen, was gehört zum Ursprungsmodell SM_0 und was wurde durch die Änderungen des Entwicklers E_1 verursacht. Diese Frage ist aber nur schwer zu beantworten.
- Mit jeder weiteren Änderungsanforderung wird es für einen neuen Entwickler immer schwerer, ein schlüssiges mentales Modell zu erstellen⁵.
- Konsequenzen dieses Prozesses der verzerrten Modellaufnahme sind:
 - Keiner der beteiligten Entwickler versteht das existierende Systemmodell korrekt. Der erstellende Entwickler nicht, weil sein Modell durch die Änderungen der nachfolgenden Entwickler verzerrt wurde, und die nachfolgenden Entwickler nicht, weil sie ihr jeweiliges Basis-Systemmodell schon nicht richtig verstehen konnten.
 - Die Software wird immer brüchiger, es wird immer schwerer, Änderungen durchzuführen.
 - Um diesen Prozess entgegenzuwirken, muss der Prozess der Modellbildung durch den Entwickler verbessert werden.

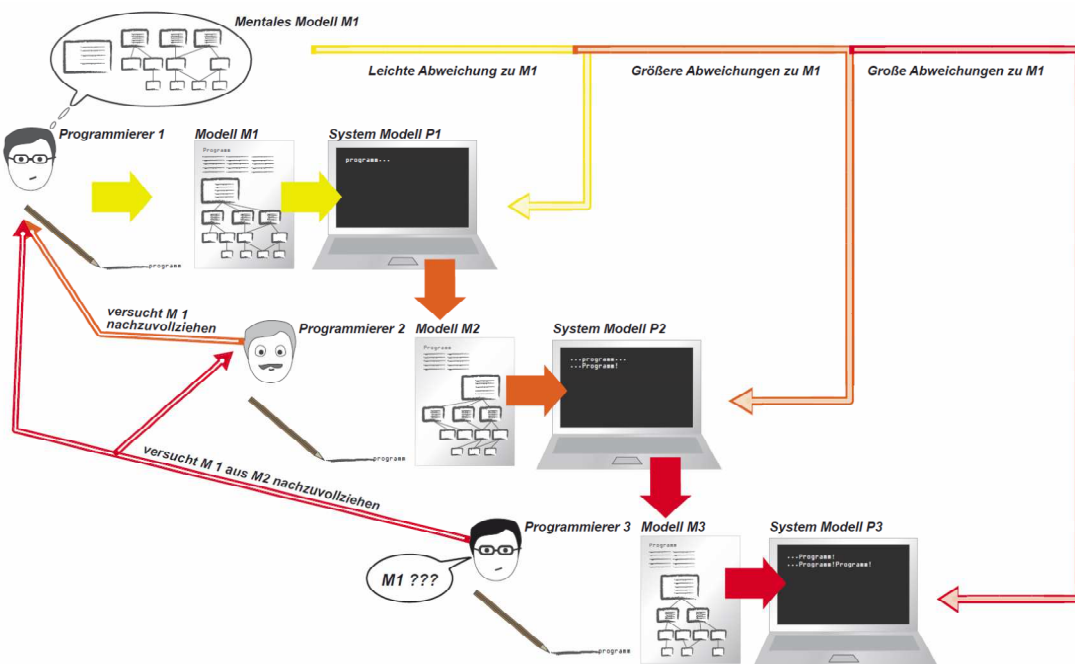


Abbildung 1: Auswirkungen der verzerrten Systemaufnahme in der Software-Wartung

⁴ An dieser Stelle merkt Parnas (1994) an, dass es nun niemanden mehr gibt, der das aktuelle Systemmodell versteht: der erzeugende Entwickler E_0 versteht es nicht mehr, da das Modell ja geändert wurde. Der ändernde Entwickler E_1 hat das ursprüngliche Modell schon nicht verstanden.

⁵ *It's harder to read code than to write it.* (aus: <http://www.joelonsoftware.com/articles/fog0000000069.html>, Abruf am 03.08.2009)

5 Software Wellness

Zentrales Element zur Verringerung des Software Agings ist eine Verbesserung des Modellbildungsprozesses durch den Entwickler. Um hier zu einem positiven Ausdruck zu kommen (und nicht Maßnahmen **gegen** etwas aufzählen zu müssen), wird der Begriff **Software Wellness** definiert.

In Anlehnung an den Begriff Wellness, der für einen Menschen Wohlbefinden und leben in ausgewogener Balance ausdrückt⁶, bedeutet Software Wellness eine Balance zwischen den Anforderungen der Benutzer und den technischen Möglichkeiten, diese zu realisieren. Das umfasst auch die Punkte:

- Neue Anforderungen der Benutzer können in vernünftiger Zeit verwirklicht werden.
- Fachlich als vernünftig erscheinende Änderungswünsche werden nicht aufgrund technischer Restriktionen abgelehnt.
- Im Wartungsprozess ist Zeit für Refactoring.

Software Wellness erscheint vor allem wichtig vor dem Hintergrund, dass die Mehrheit der Kosten eines Softwareprodukts in der Wartungsphase anfallen (u.a. Ludewig & Lichter, 2010). Eine erleichterte Wartung hat also direkte ökonomische Vorteile.

6 Wie kann Software Wellness durch konstruktive Maßnahmen erreicht werden?

Mit konstruktiv ist hier alles gemeint, was im Voraus von Verantwortlichen für ein Softwareprojekt festgelegt werden kann, als Handlungsanweisung oder Strategie, ohne dass es nötig ist, selbst bei der Softwareerstellung programmierend oder modellierend einzugreifen – also persönliche Kompetenz einzubringen.

6.1 Bewusstsein und Rückhalt auf der Ebene der Entscheidungsträger

Um Software Wellness zu erreichen, sind Ressourcen und Maßnahmen notwendig. Es entstehen Aufwände, für die jemand aufkommen muss. Daher ist es notwendig, das Bewusstsein bei Entscheidungsträgern zu wecken, um dort Rückhalt und Ansporn zu erhalten (Mens et al., 2005).

6.2 Wandlungsorientiertes Design

Betrachtet man die Kosten von Software während ihres Lebenszyklus so stellt man fest, dass die Mehrheit (bis zu 90%) der Kosten erst in der Wartungsphase anfällt (Koskinen, 2010). Daher besteht nach vielfältiger Aussage (u.a. Friedrichsen (2009) und Kehrer (2008)) die eigentliche Herausforderung in der Softwareentwicklung darin, die Wartung so einfach wie möglich zu machen, den nachfolgenden Wartungsprozess gleich in der Entwicklung zu berücksichtigen. Parnas (1994) nennt es „*Design for Success*“, Friedrichsen (2009) „*Design for Wartung*“ und Mittermeir (2001) „*Design for volatility*“.

Die Anforderungen an eine einfache Wartung definiert Friedrichsen (2009) wie folgt:

- Umsetzung der Änderungen lokal/wenige Quellcodeeinheiten
- keine Schnittstellenänderungen

⁶Aus http://en.wikipedia.org/wiki/Wellness_%28alternative_medicine%29, Abruf am 03.08.2009

- keine globalen Seiteneffekte

Wesentliches Merkmal eines wandlungsorientierten Designs ist das Trennen der Systemelemente nach der potentiellen Änderungshäufigkeit: Die stabilen Systemelemente können durch ein Framework abgedeckt werden.

6.2.1 Identifikation der instabilen Systemelemente

a) Entwurfsphase

Generell ist die Unterscheidung der stabilen und instabilen Systemelemente durch die Erfahrung des Architekten zu leisten, Friedrichsen (2009) nennt als eine Heuristik, dass schwammige und generische Anforderungen Änderungspotential vermuten lässt. Ebenso wichtig ist eine Trennung der fachlichen und der technischen Aspekte, da die Fachlichkeit sich schneller ändert als die Technologie. Die Technologie ändert sich eher schubweise, während fachliche Anforderungen laufend eingearbeitet werden müssen.

Fachliche Schnittstellen sollten sauber strukturiert und voneinander entkoppelt sein. So ist die Chance höher, dass Änderungen lokal beschränkt bleiben.

b) Wartungsphase

Um sich häufig ändernde Quellcodedateien während der Wartungsphase zu entdecken, ist – neben dem Wissen der Entwickler – eine Auswertung der Daten des Versionsverwaltungssystem möglich, wie es von Hassan & Holt (2004) gemacht wurde.

Sind die sich häufig ändernden Dateien identifiziert, so lässt sich nach weiterer Analyse entscheiden, ob ein Refactoring vorgenommen werden sollte.

6.3 Verbesserung des Prozesses der verzerrten Modellaufnahme

Wichtiger Eingriffspunkt ist der Prozess der verzerrten Modellaufnahme. Hier lassen sich verschiedene Ansätze finden. Grundsätzliche Strategie ist es, die aufzunehmenden Modelle zu verkleinern oder die Modellaufnahme ganz zu eliminieren – aus der Sicht von Codeänderungen ausgedrückt: Die Änderungen der Quellcodes sind möglichst lokal zu begrenzen oder ganz zu vermeiden.

6.3.1 Keine Modellaufnahme

a) Stabiles Projektteam

Die Modellbildung ist dann nötig, wenn sich ein neuer Entwickler in den Code einarbeiten muss. Macht hingegen ein mit der Software vertrauter Entwickler die Änderungen, so kann er sein schon vorhandenes mentales Modell nutzen⁷, das Modell wird nicht weiter verzerrt (Kehrer, 2008).

b) Konfiguration statt Codeänderung

Lässt sich eine Benutzeranforderung durch Konfiguration der Software erreichen, so ist keine Quellcodeänderung notwendig, die Software bleibt stabil (Thomas, 2005).

⁷ Der Entwickler hat sicherlich auch einiges vergessen, das mentale Modell ist aber weitaus exakter als das eines nicht mit der Software vertrauten Entwicklers.

Zu beachten ist hier jedoch, dass die für die Konfiguration notwendige zusätzliche Abstraktion die Entwicklung der Software komplizierter und fehleranfälliger gemacht haben könnte.

6.3.2 Verringerung der Größe des aufzunehmenden Modells

a) Modularität

Ein Modul ist eine abgeschlossene Komponente einer Software⁸. Jede Komponente hat ihr eigenes Systemmodell. Nimmt man an, dass eine Änderung auf ein Modul beschränkt ist, dann muss ein Entwickler nur ein mentales Modell für das Modul entwickeln (sowie evtl. zusätzlich ein Modell für das Zusammenspiel der Module). Auch wenn die Änderung nicht lokal auf ein Modul beschränkt ist, sondern sich über mehrere Module erstreckt, ist das Erstellen der mentalen Modelle für die betroffenen Module immer noch weniger Aufwand als ein Modell des kompletten Softwaresystems aufzunehmen. Modularität verringert die Komplexität der Modelle wesentlich, daher entstehen auch weniger Modellverzerrungen.

Ein weiterer Vorteil modularer Software ist, dass Module, gekennzeichnet durch eine klare Schnittstelle und die dahinter stehende Implementierung, leicht ersetzt werden können: Wenn ein Modul so brüchig geworden ist, dass eine weitere Wartung mehr Aufwand verursacht als das Neuentwickeln des kompletten Moduls, muss der Entwickler nur die Schnittstelle verstehen und kann dann die Implementierung des Moduls neu durchführen.

b) Erweiterbarkeit

Ein Sonderfall der Modularität ist die Erweiterbarkeit von Software: Die Software bietet wohldefinierte Anschlussstellen für Module, zur Laufzeit wird dann der Kontrollfluss an die angeschlossenen Module weitergeleitet. Enormer Vorteil ist, "that you can customize behaviour without having to touch existing code" (Venners, 2005a). Schon existierender Code muss nicht geändert werden, trotzdem können die Benutzerbedürfnisse erfüllt werden durch das Erstellen eines Moduls mit den vom Benutzer gewünschten Eigenschaften. Der Entwickler muss nur ein Modell der Schnittstelle entwickeln, nicht aber die Einzelheiten existierender Software verstehen. Zusätzlich werden keine neuen Fehler in schon bestehende Software eingebaut.

Populäres Beispiel für Erweiterbarkeit ist das Plug-in-Konzept der Software-Plattform Eclipse⁹. Anschlussstellen heißen hier „*extension point*“, die erweiternden Module „*extension*“.

c) Standardisierte Softwarebausteine

Die Bildung eines mentalen Modells wird einfacher, wenn wohlbekannte Softwarebausteine genutzt werden. Sobald diese Standards entdeckt werden, ist es einfach, ein schon bekanntes Modell in der Software wiederzufinden. Zu den Softwarebausteinen im weiteren Sinne gehören Frameworks und Design Patterns

I Frameworks

Nachdem die Funktionsweise eines speziellen Frameworks einmal vom Entwickler verstanden wurde, ist für die Komponenten keine Modellaufnahme mehr notwendig. Der Entwickler kann sich schnell in die Anwendung einfinden. Zudem wird das zu verstehende Modell durch die Trennung zwischen Fachlichkeit und Technik verkleinert: Gewöhnlich muss nur das fachliche

⁸Aus http://de.wikipedia.org/wiki/Modul_%28Software%29, Abruf am 04.05.2009

⁹Siehe <http://www.eclipse.org>

Modell neu aufgenommen werden. Zu achten ist allerdings darauf, dass Frameworks nicht zu komplex werden, da ansonsten das Framework selbst sehr schwer zu verstehen ist (Venners, 2005b).

II Design Patterns

Design Patterns sind ebenfalls Softwarestrukturen, die bei der Modellaufnahme schnell erkannt werden können. Izurieta und Bieman fanden heraus, dass Patterns bevorzugt erhalten bleiben – auch für den nachfolgenden Entwickler (Clemente Izurieta & Bieman, 2007). Mit Hilfe der Namen geläufiger Design Patterns wird die Kommunikation der Entwickler untereinander effizienter.

d) KISS

KISS ist ein Akronym, das in diesem Kontext für *Keep It Simple and Stupid* steht. Als Anforderung an den Entwicklungsprozess in der Informatik lässt es sich im Sinne von Einfachheit verwenden. Wenn Software einem einfachen Systemmodell folgt, so lässt sich daraus auch einfacher wieder ein mentales Modell ableiten. Einfachheit ist daher als Entwurfsprinzip zu fordern.

6.3.3 Steuerung der Modellaufnahme

a) Training der Entwickler

Werden Entwickler für den Modellbildungsprozess sensibilisiert, wird ihnen die Bedeutung einer sauberen Arbeit für die Software Wellness vermittelt. Werden ihnen Mittel und Wege aufgezeigt, ihren Nachfolgern die Modellbildung zu erleichtern, so werden die Effekte der verzerrten Modellaufnahme verringert. Parnas beklagt ein allgemeines Ausbildungsdefizit, das es zu beseitigen gilt (Eickelmann, 1999).

b) Coaching der Entwickler

Sind die Entwickler bei der Modellaufnahme nicht allein auf ihr eigenes Urteilsvermögen angewiesen, sondern können sie erfahrene Kollegen in Zweifelsfällen um Rat fragen, so ist das entstehende mentale Modell von höherer Qualität. Klein spricht vom „key developer“, der die Ideen des Architekten weiter an die Entwickler vermittelt (Klein, 2005).

c) Dokumentation

Adäquate Dokumentation hilft dem Entwickler bei der Modellaufnahme und dem Verständnis. Fehlende, unvollständige oder veraltete Dokumentation ist jedoch ein weitverbreitetes Phänomen vieler Softwareentwicklungen (Kehrer, 2008; Parnas, 1994) – wohl auch weil das Erstellen von Dokumentation nicht zu den beliebtesten Aufgaben im Leben eines Entwicklers gehört.

Mögliche Abhilfen sind u.a.:

- Reviews (siehe Kapitel 6.4.1)
- Projektmitgliedsrolle *Dokumentator*: Eine Person, die für die Dokumentation zuständig ist und das fehlende Wissen erfragen muss.
- Dokumentation durch Tests (siehe Kapitel 6.4.3)
- Dokumentation im Quellcode¹⁰

¹⁰ Dokumentation im Quellcode hilft, der Aktualitätsfalle zu entgehen: Quellcodedateien werden geändert, die Dokumentation aber nicht, weil sie gerade nicht greifbar ist, es vergessen wird oder zu aufwändig erscheint.

6.4 Reviews

Reviews stellen generell eine Möglichkeit dar, Schwachstellen in Software zu finden, um sie dann beseitigen zu können. Parnas hält Reviews für dringend notwendig, da für ihn eine wesentliche Motivation der Entwickler lauffähige Programme sind (Parnas, 1994) – für die Erreichung dieses Ziels werden gerne auch Verstöße gegen das Ziel der Software Wellness in Kauf genommen.

6.4.1 Manuelles Review

Beim manuellen Review werden die Arbeitsergebnisse durch Dritte begutachtet. Hier kann überprüft werden, ob noch die Bedingungen und Richtlinien der Software Wellness vorliegen, oder ob verbesserungsbedürftige Konstrukte („Smells“) auffallen.

6.4.2 Automatisiertes Review

Während des Erstellungsprozesses der ausführbaren Programmdateien können automatisierte Reviews durchgeführt werden.

a) Code Quality Checker

Sogenannte *Code Quality Checker*, wie z.B. *PMD*¹¹ oder *Findbugs*¹² ermöglichen es, eine Reihe von verständniserschwerenden Code-Konstruktionen zu entdecken. Wird der automatisierte Erstellungsprozess der ausführbaren Dateien („Daily Build“) nach Entdeckung mindestens eines solchen Problems als fehlgeschlagen deklariert, dann entsteht die notwendige Dringlichkeit, das Problem auch zu beheben.

b) Software Analyse und Visualisierung

Es gibt eine Reihe von Analyse- und Visualisierungswerkzeugen¹³, die den Quellcode eines Softwaresystems analysieren und Metriken darauf anwenden. Anhand der Ergebnisse lassen sich kritische, für eine Überarbeitung zu empfehlende Stellen identifizieren.

6.4.3 Automatisierte Tests

Werden während des Erstellungsprozesses der ausführbaren Programmdateien auch automatisierte Tests (zum Beispiel über JUnit¹⁴) ausgeführt, so wird dadurch das Risiko verkleinert, dass mit einer Quellcodeänderung ein neuer Fehler entsteht – und damit auch die Notwendigkeit einer nochmaligen Quellcodeänderung mit weiterer Fehlergefahr. Gleichzeitig können automatisierte Tests auch als Dokumentation des Systemverhaltens dienen.

6.5 Frühes Gegensteuern

Erkannte Schwächen sollten so früh wie möglich beseitigt werden, denn je früher im Entwicklungsprozess die Schwäche eliminiert wird, desto weniger Aufwand fällt dabei an (C. Izurieta & Bieman, 2008).

¹¹Siehe <http://pmd.sourceforge.net/>

¹²Siehe <http://findbugs.sourceforge.net/>

¹³Beispiele: Sotograph (<http://www.hello2morrow.com/products/sotograph>, abgerufen am 05.08.2009) und SonarJ (<http://www.hello2morrow.com/products/sonarj>, abgerufen am 05.08.2009)

¹⁴Siehe <http://www.junit.org/>

7 Zukünftige Ansätze

Mens et al. (2005) fordern die toolbasierte Unterstützung des Refactoring auch für die Modell-Ebene. Der Ansatz des Model-Driven-Developments geht in dieselbe Richtung: Quellcode wird größtenteils aus mit grafischen Benutzeroberflächen arbeitenden Werkzeugen generiert, die der Verwaltung der Modelle dienen - nur an einigen wenigen Stellen erfolgt die manuelle Ergänzung von Codefragmenten.

Anhang C: Grundlagen der Simulation

Inhaltsverzeichnis

1 Einleitung.....	2
2 Definition	2
2.1 Der Begriff „Simulation“	2
2.2 Motivation: Weshalb wird simuliert?	3
2.3 Anwendungen: Was lässt sich simulieren?	5
3 Der Weg zur Simulation.....	5
3.1 Problemstellung	5
3.2 System	5
3.3 Systemanalyse	6
3.4 Modellierung	6
3.5 Implementierung.....	7
3.6 Verifikation und Validierung.....	8
4 Grenzen und Fehlerquellen	8
5 Taxonomien	9
5.1 Grundlegende Kategorisierungen	9
5.2 Werkzeugunterstützung	10
5.3 Exaktheit des Simulationsmodells.....	13
5.4 Arten von Simulationen.....	14
5.4.1 Monte-Carlo-Simulation.....	14
5.4.2 Ereignisorientierte Simulation.....	15
5.4.3 Kontinuierliche Simulation	15
5.4.4 Hybride Simulation	15
6 Standards, Normen und Formalismen	16
7 Abbildungsverzeichnis	17

What happens if a big asteroid hits Earth? Judging from realistic simulations involving a sledge hammer and a common laboratory frog, we can assume it will be pretty bad.

- Dave Barry

Simulations may not lead to Nobel prizes, but they can significantly increase student learning and motivation

- John P. Hertel & Barbara J. Millis (2002)

1 Einleitung

Simulation ist ein weitverbreitetes Mittel, um anschaulich und begreifbar den Umgang mit Systemen zu erfahren. In der wissenschaftlichen Fachliteratur ist es ein vielbehandeltes Thema. Gleichzeitig ist es die Grundlagen von sogenannten Simulationsspielen. Mit deren Hilfe kann auf spielerische Art der Umgang mit Systemen trainiert werden. Die Grenzen zwischen Simulation und Spiel sind oft fließend, wie der Microsoft Flugsimulator zeigt (FlightSimulatorX, 2010). Dieser wird als Unterhaltungsspiel eingesetzt, er ist aber auch als Trainingsinstrument geeignet.

Da Simulation eine wichtige Grundlage von Simulationsspielen ist, soll hier ein komprimierter Abriss über Wesen und Sinn von Simulation gegeben werden. Die wichtigen Begriffe werden kurz erläutert. Das Augenmerk der Arbeit im Bereich der Simulation liegt auf erweiterbaren und modularen Simulationsmodellen, es wird in einem eigenen Kapitel der Hauptarbeit besonders vertieft. Um einen fließenden Aufbau dieses Anhangs zu ermöglichen, werden einige Begriffe schon benutzt, bevor sie definiert oder beschrieben werden. Dies geschieht nur dann, wenn dadurch das Verständnis der Ausführungen nicht gefährdet ist.

2 Definition

In seinem „*Handbook of Simulation*“ definiert Jerry Banks Simulation mit den folgenden Sätzen:

Simulation is the imitation of the operation of a real-world process or system over time. Simulation involves the generation of an artificial history of the system and the observation of that artificial history of the real system that is represented. Simulation is an indispensable problem solving methodology for the solution of many real-world problems. Simulation is used to describe and analyze the behavior of a system, ask what-if questions about the real system, and aid in the design of real systems. Both existing and conceptual systems can be modeled with simulation (J. Banks, 1998).

Geführt durch diese Definition soll in den nächsten Abschnitten der Begriff der Simulation als eine wesentliche Grundlage dieser Arbeit näher erläutert werden. Geführt durch eine Unterteilung von Robinson (2004) wird dabei behandelt, was Simulation ist, weshalb sie angewendet wird und für welche Problemstellungen sie zum Einsatz kommt.

2.1 Der Begriff „Simulation“

In der obigen Definition heißt es: „*Simulation is the imitation of the operation of a real-world process or system over time.*“ Hier werden drei wesentliche Eigenschaften einer Simulation genannt:

- Zum ersten ist Simulation immer mit einem System oder Prozess verbunden. Banks fordert zwar zunächst ein System oder einen Prozess aus der realen Welt, im letzten Satz aber lässt er auch konzeptionelle Systeme zu
- Der zweite charakteristische Begriff ist „imitation“: Das zugrundeliegende System wird nachgeahmt, es wird nicht das originale System benutzt. Der Übergang zwischen beiden Systemen wird Modellierung genannt („can be modeled with simulation“).
- Ein dritter wichtiger Begriff ist „over time“: Es wird das System im Zeitverlauf betrachtet¹.

2.2 Motivation: Weshalb wird simuliert?

Die Motivation, Simulationen durchzuführen, liegt laut Banks darin, dass es eine Methode zur Problemlösung ist. Mit Hilfe von Simulationen kann das Verhalten von Systemen beschrieben und analysiert werden. Dörner (2003) führt aus, dass es eine wesentliche, inhärente Schwäche des Menschen ist, das Verhalten eines Systems im Zeitablauf zu erfassen und vorherzusagen. Hier können Simulationen dabei helfen, ein grundlegendes Verständnis für die Zusammenhänge in einem System und Auswirkungen von Veränderungen in einem System über den Zeitablauf zu schaffen. Simulationen werden auch als Spielzeuge betrachtet, mit denen sich der Umgang mit Systemen und deren längerfristige Dynamik erfassen lässt (Wright, 2007). Reynolds (2009) sieht neben dem spielerischen Ansatz, mit dem sich Verständnis für ein System entwickeln lässt², auch die zweite wichtige Rolle der Simulation: Die Simulation als Mittel zur Abwägung von Alternativen, und damit auch Hilfe bei der Problemlösung. Unter diesem Funktionsbereich sind beispielsweise Simulationen der Seuchenausbreitung, die Wettervorhersage und der Vergleich verschiedener Prozessorarchitekturen für Computer einzugliedern. Hier wird zwischen den Rollen Entwickler und Benutzer der Simulation unterschieden. Beim spielerischen bzw. experimentellen Ansatz der Simulation verschmelzen diese beiden Rollen: Der Entwickler kann auf die Simulation Einfluss nehmen und sie ändern, um damit Fragen nach Gründen und Zusammenhängen zu beantworten. Reynolds nennt die Simulation neben der Theoriebetrachtung und dem Experiment auch das inzwischen dritte Standbein der Forschung³.

¹ Wie später beschrieben wird, gibt es auch Definitionen, bei denen Zeit eine nicht notwendige Komponente der Simulation ist.

² An dieser Stelle zählt Reynolds die Simulationen zum Zwecke der Lehre und des Trainings explizit zur Gruppe der Simulationen zur Problemlösung.

³ Diese Aussage findet sich auch bei C. M. Banks (2009). Dort wird sie zurückgeführt auf einen Beitrag von Colwell (1999).

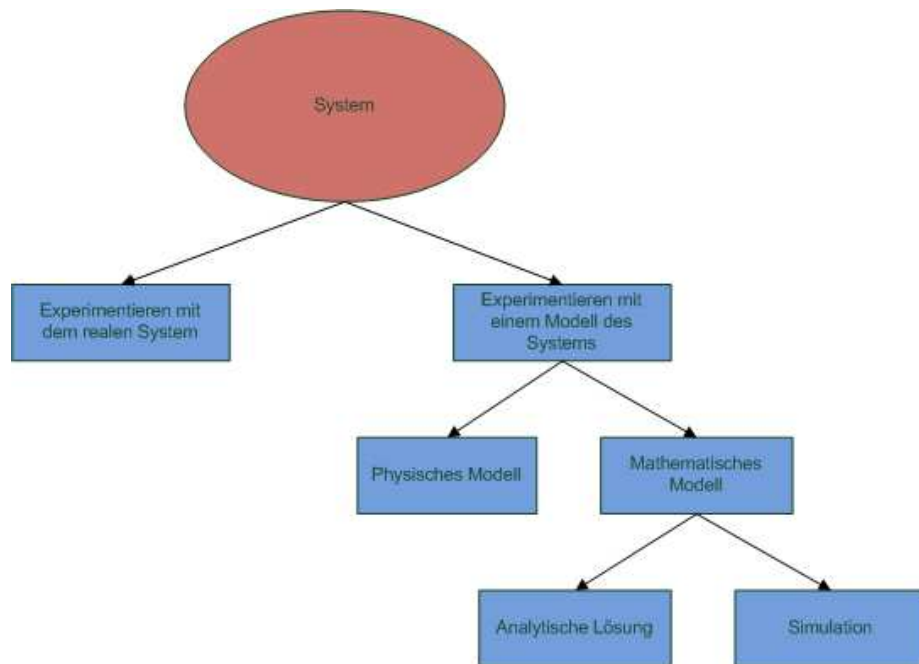


Abbildung 1: Ansätze der Untersuchung eines Systems (Law, 1991)

In Abbildung 1 ist zu sehen, dass Simulation ein Ansatz von mehreren ist, ein System zu erforschen. Simulation ist eine Alternative zum Experiment mit dem System in der realen Welt oder mit einem physischen Modell dieses Systems. Auch ist sie ein Mittel für den Fall einer fehlenden analytischen Lösung.

Da die Vorhersage der dynamischen Entwicklung eines Systems für den Menschen ohne Hilfsmittel nur schwer möglich ist, stellt Robinson (2004) drei Eigenschaften von Systemen heraus, die den Einsatz von Simulationen fordern: Schwankungen der Eingangsgrößen, Vernetzung der Wirkungsgeflechte der Systemkomponenten und Komplexität. Die Vernetzung der Komponenten trägt dazu bei, dass die schwankenden Eingangsgrößen intuitiv nicht vorhersagbare unterschiedliche Wirkung auf das Systemverhalten haben können. Bei der Komplexität von Systemen unterscheidet Robinson zwischen kombinatorischer und dynamischer Komplexität. Die kombinatorische Komplexität hängt ab von der Anzahl der Systemkomponenten: Je höher die Anzahl der Systemkomponenten, desto komplexer ist das System tendenziell. Die dynamische Komplexität eines Systems rührt demgegenüber vom Zusammenspiel der Komponenten im Zeitablauf.

Senge (1990) beschreibt drei Effekte, die dazu beitragen, dass die dynamische Komplexität ansteigt:

- Eine Handlung hat kurzfristig eine andere Wirkung als langfristig.
- Eine Handlung wirkt sich in verschiedenen Komponenten des Systems unterschiedlich aus.
- Eine Handlung hat nicht naheliegende Auswirkungen, das System verhält sich kontraintuitiv

Begünstigt durch den Fortschritt in den letzten Jahrzehnten sowohl bei Hardware als auch bei Software (McHaney, 1991, S. 34) wird es zunehmend einfacher, Simulationen zu erstellen. Dadurch werden die Einsatzmöglichkeiten für die Simulation als Werkzeug immer vielfältiger.

Die Gründe für den Einsatz von Simulationen sind vielfältig, einer der wesentlichsten ist die einfachere Handhabung gegenüber dem Experiment mit dem realen System (Robinson, 2004). Es können Ressourcen eingespart werden, sei es in Form von Kosten oder in Form von Zeit. Zusätz-

lich ist es einfacher, mit dem System zu experimentieren: Die Eingangsgrößen in software-basierten Systemen lassen sich leicht verändern, ebenso wie die Zusammenhänge der Systemkomponenten untereinander. Existiert das zu simulierende System (noch) nicht ⁴oder haben Messungen verfälschende Einflüsse auf das System⁵, so sind Simulationen unvermeidlich zur Untersuchung des Systems. Bei noch nicht existierenden Systemen bekommt die Simulation die Bedeutung eines Planungshilfsmittels (Gogg & Mott, 1993). Auch die zeitliche Dimension kann ein Grund für den Einsatz der Simulation sein: Laufen Prozesse sehr schnell oder sehr langsam ab so können sie durch Simulation auf ein für den Menschen zugängliches Zeitmaß angepasst werden. Beispiel für einen sehr langandauernden Prozess ist die Entstehung einer Galaxie, sehr kurze Prozesse sind beispielsweise chemische Reaktionen (Mattern, 1995).

2.3 Anwendungen: Was lässt sich simulieren?

In seiner Zusammenfassung führt J. Banks (1998) aus, dass sich wirkliche und konzeptionelle Systeme zur Simulation eignen. Er stellt als Anwendungsbereiche Fertigungssysteme und Produktionsprozesse vor. Gesundheitssysteme, Transportsysteme und Vegetationsvorhersagesysteme gehören genauso zum großen Bereich der Simulationen wie Luftfahrt, Computersystemdurchsatz sowie die Leistung von Mobilfunknetzen. Hinzu kommen die weiten Anwendungsgebiete von Monte-Carlo-Simulationen, die in der Physik oder in der Mathematik genutzt werden. Mit ihrer Hilfe lassen sich u.a. mathematische Probleme wie Differentialgleichungen lösen (Thompson, 2000). Dieser umfassende Bereich zeigt, dass sich die Anwendungsbereiche von Simulation thematisch nicht eingrenzen lassen, sondern dass Simulationen immer dann zum Einsatz kommen, wenn sich die essentiellen Komponenten eines Systems modellieren lassen und Vorteile der Simulation genutzt werden können, wie zum Beispiel die Einsparung von Kosten.

3 Der Weg zur Simulation

3.1 Problemstellung

Ausgangspunkt einer jeden Simulation ist eine Problemstellung. Die Simulation ist Werkzeug, mit dem das Problem näher untersucht werden kann. Das Problem setzt den Kontext für die Simulation, das heißt, es ist Leitfaden und Maßstab bei der Erstellung der Simulation: abhängig vom Problem werden unterschiedliche Aspekte der realen Welt simuliert.

3.2 System

Ein zentraler Begriff im Bereich der Simulation ist der des Systems. Garrido (2001) definiert den Begriff System wie folgt:

A system is part of the real world under study and that can be identified from the rest of its environment for a specific purpose. Such a system is called the real system because it is physically [...] part of the real world. [...] A system is composed of a set of entities (or components) that interact among themselves and with the environment to accomplish the system's goal.

⁴Bei der Rekonstruktion eines Verkehrsunfalls existiert das betrachtete System nicht mehr.

⁵Gipser (1999) nennt an dieser Stelle das Beispiel der Druckmesssonde bei einer Strömungssimulation.

Diese Definition ist schon auf den Kontext der Simulation zugeschnitten, wenn das System als Teil der realen Welt, der beobachtet wird, gesehen wird. Wichtig ist die Zusammensetzung des Systems aus mehreren Komponenten und deren Zusammenspiel zueinander.

Checkland (1999) unterteilt Systeme weiter in 4 Hauptklassen:

- **Natürliche Systeme** („natural systems“) sind all die Systeme, die ihren Ursprung im Universum haben, wie z.B. Atome, Wetter und Sternensysteme.
- **Entworfenen physischen Systeme** („designed physical systems“) sind körperliche Systeme, die vom Menschen entworfen wurden. Beispiele sind hier: Autos, Häuser und Fabriken.
- **Entworfenen abstrakten Systeme** („designed abstract systems“) sind Systeme ohne körperliche Manifestation, die vom Menschen entworfen wurden, wie z.B. Mathematik und Literatur.
- **Systeme menschlicher Aktivitäten** („human activity systems“) sind solche, die durch menschliche Interaktionen charakterisiert sind, wie beispielsweise eine Familie, eine Stadt oder ein politisches System.

Hier wird klar, dass der Begriff des Systems ziemlich weit gefasst werden kann und dass Systeme sich beileibe nicht auf physisch vorhandene Systeme einschränken lassen, sondern dass der Begriff des Systems auch immer vom Kontext abhängt, in dem es genutzt wird. Für diese Arbeit relevant sind die ersten beiden Hauptklassen.

3.3 Systemanalyse

Ein System, wie es im vorigen Abschnitt vorgestellt wurde, ist ein Teil der realen Welt. Soll es simuliert werden, muss es in eine Form überführt werden, die eine Simulation durch einen Rechner erlaubt. Diese „Überführung“ nennt sich Systemanalyse. Dabei wird das zu modellierende System, oft ein System der realen Welt, beobachtet (Birta & Arbez, 2007) und die für die Simulation relevanten Objekte und ihre Beziehungen werden identifiziert. Ziel ist es, diese in ein formales Simulationsmodell, **konzeptionelles Modell** genannt (Garrido, 2001), zu überführen, das exakt beschrieben ist und in ein Modell für den Rechner (**Simulationsmodell**) umgesetzt werden kann.

3.4 Modellierung

Wainer (2009) definiert ein Modell als eine verständliche, abstrakte und konsistente Darstellung eines Systems, das von Menschen genutzt wird mit dem Ziel, das zugrundeliegende System besser zu verstehen. Als Modellierung bezeichnet er den Prozess des Durchdenkens eines Systems mit dem Ziel, ein abstraktes Modell dieses Systems zu schaffen. Die Modellierung erfolgt jeweils unter einem bestimmten Gesichtspunkt, der sich aus der Problemstellung ableitet. Dementsprechend ist das Modell auch nur eine beschränkt – bezüglich des Modellierungsgesichtspunktes – gültige Abbildung des realen Systems (Bossel, 1994). Ein wichtiger Aspekt des Modells ist die Wahl einer geeigneten, d.h. der Problemstellung angepassten Größenordnung: So ist die Molekülebene für die numerische Wetterprognose zu detailliert, gleichzeitig ist aber die Aussage „In Europa scheint größtenteils die Sonne.“ auch nicht hilfreich. In ersterem Fall ist der Aufwand zu hoch, im zweiten Fall ist die Güte der Aussage nicht ausreichend (Bungartz, Zimmer, Buchholz & Pflüger, 2009).

Im Rahmen der Modellierung werden die Ergebnisse der Systemanalyse zunächst in ein konzeptionelles Modell und dann in ein Simulationsmodell umgesetzt. Die formal ausgedrückten Ergebnisse der Analysephase (z. B. in Form von mathematischen Gleichungen, grafischen Darstellungen für Petri-Netze oder Zustandsmaschinen, regelbasierte Beschreibungen oder Pseudocode) werden in ein konsistentes Modell überführt. Das Modell kann aus mehreren Teilmodellen bestehen und die Darstellungsform muss nicht einheitlich sein (Birta & Arbez, 2007).

Sowohl Systemanalyse als auch Modellierung sind Tätigkeiten, bei denen überwiegend die menschliche Intelligenz und Intuition gefragt ist. Sie können (noch nicht) nicht durch Computerprogramme durchgeführt werden. Dennoch gibt es hierfür Ansätze. Gipser (1999) bezeichnet mit „Systemidentifikation“ die rechnergestützte Suche nach dem zugrundeliegenden System. Technisch werden dabei Parameter eines Modells solange variiert, bis die Ergebnisse des Modells mit den Messungen der Realität übereinstimmen. Von „Systemoptimierung“ spricht er, wenn mit diesem Verfahren die Modellparameter solange variiert werden, bis größtmögliche Übereinstimmung mit einem Sollverhalten vorhanden ist.

Oft gehen die Phasen Analyse und Modellierung gleitend ineinander über, in der Analysephase liegt der Schwerpunkt auf Gewinnung eines Verständnisses des Systems (Fowler, 1999). Für die Ersteller einer Simulation wird daher die Notwendigkeit einer ausreichenden Schulung gesehen (Gogg & Mott, 1993). Auch werden ganze Berufsbilder im Bereich der Simulation angesiedelt, beispielsweise der Simulations-Analyst (McHaney, 1991).

3.5 Implementierung

Ein Simulationsmodell kann mit verschiedenen Hilfsmitteln in eine rechnerausführbare Form gebracht werden. Besondere Anforderungen an solche Hilfsmittel sind die Unterstützung statistischer und wahrscheinlichkeitstheoretischer Funktionen, Event- und Listenverarbeitung, Sammlung und Analyse von Daten, Berichtsfunktionen und Fehlerbehandlung (Law, 1991).

Robinson (2004) nennt die drei Alternativen Tabellenkalkulationsprogramme, Programmiersprachen und spezialisierte Simulationssoftware für die Implementierung von Simulationen. Tabellenkalkulationen wie Microsoft Excel sind eher für einfache, schnell mit relativ wenig Programmierkenntnissen entwickelbare Modelle geeignet. Viele der Anforderungen komplexer Modelle können zwar erfüllt werden, müssten aber mühsam programmiert werden. Gleichwohl existieren sogenannte Add-Ons (d.h. zusätzliche, einzufügende Softwarepakete) zur Bereitstellung von Simulationsfunktionalität.

Java, C#, C++ und Visual Basic sind Beispiele für Programmiersprachen, die der Entwicklung von Simulationen große Flexibilität bieten, aber im Gegenzug auch ein hohes Maß an Programmierkenntnissen erwarten. Aus den besonderen Anforderungen, die aus der Implementierung von Simulationen erwachsen und zu immer wiederkehrenden Programmieraufgaben führen, sind spezialisierte Simulationsprogrammiersprachen und -pakete wie z.B. GPSS/H oder Simulink erwachsen. Für die Erstellung einer Simulation bieten diese Pakete eine hohe Mächtigkeit, erfordern auf der anderen Seite vom Entwickler der Simulation aber entsprechende Einarbeitung. Zusätzlich sind die Kosten für die Softwarelizenzen meist nicht unerheblich.

In der Literatur wird zusätzlich noch der Simulator genannt. Law (1991) definiert einen Simulator als ein Programmpaket, das die Erstellung einer Simulation aus einer bestimmten Klasse von

Simulationen mit nur geringen Programmierkenntnissen erlaubt. McHaney (1991) diskutiert die Vor- und Nachteile eines Simulators detailliert.

3.6 Verifikation und Validierung

Zur Erstellung einer Simulation gehören Verifikation und Validierung. In Abbildung 2 ist zu sehen, wie Law (1991) diese Aufgaben im Lebenszyklus einer Simulation anordnet.

Verifikation bezeichnet die Überprüfung der Implementierung des Simulationsmodells auf Fehlerfreiheit (McHaney, 1991). Genutzt werden hier dieselben Techniken wie bei der Überprüfung einer Software auf Fehlerfreiheit, so z.B. Traces, Debugging oder Unit-Tests (Law, 1991).

McHaney (1991) führt zusätzliche Methoden der konstruktiven Qualitätssicherung wie die Wahl eines geeigneten Programmierparadigmas⁶ und die sorgfältige Dokumentation des Quellcodes an. Er empfiehlt auch den gesunden Menschenverstand zur Verifikation: Das sorgfältige Beobachten von Diagrammen und Animationen sowie das Lesen von Simulationsergebnisberichten kann bei der Fehlersuche helfen.

Validierung hingegen bezeichnet das Prüfen des Simulationsmodells auf fachliche Richtigkeit. Dies geschieht meist durch Experten und durch einen Vergleich von Messungen oder Beobachtungen der realen Welt mit den Ergebnissen des Programmes (Garrido, 2001). Die Vergleichsdaten können auch im Nachhinein („a-posteriori“) erhoben werden. Das ist bei Simulationen zur Prognose – Beispiele sind Börsenkurse und Wettervorhersagen – eine geeignete Möglichkeit, um die Qualität des Modells zu bewerten (Bungartz et al., 2009). Validierung ist notwendig, da nur Teile der realen Welt modelliert wurden – die Modellierung der realen Welt als Ganzes ist schon aus Kostengründen nicht möglich. Validierung überprüft nun, ob die Selektion der zu modellierenden Aspekte der realen Welt funktioniert hat und ob das erzeugte Modell im Kontext der Simulation zu plausiblen Ergebnissen kommt. Ist dies nicht der Fall könnten aus den weiteren Simulationsergebnissen keine belastbaren Schlussfolgerungen gezogen werden.

Abhängig vom Einsatz der Simulation können Verifikation und Validierung so hohe Bedeutung zukommen, dass sie mit Richtlinien und Standards unterstützt werden. Als Beispiel seien Richtlinien für den VV&A⁷-Prozess des *Modeling & Simulation Coordination Office* (MSCO) im amerikanischen Militärbereich genannt ((M&SCO), 2012).

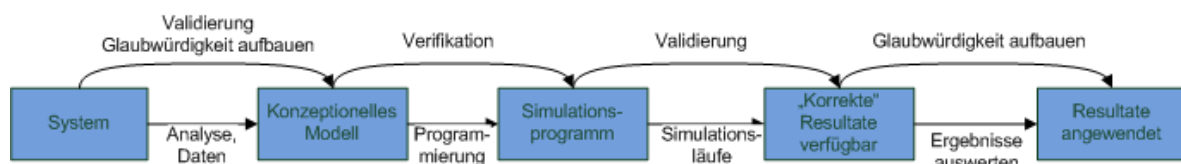


Abbildung 2: Prozess zur Erzeugung einer Simulation (Law, 1991)

4 Grenzen und Fehlerquellen

Simulationen sind ein mächtiges Werkzeug für die Arbeit mit Systemen. Gleichwohl sollte der Anwender sich beim Einsatz bewusst sein, dass sie Grenzen besitzen. Law (1991) weist darauf-

⁶ Er spricht 1991 vom Ansatz der strukturierten Programmierung, dieser wird heute – gut 20 Jahre später – ergänzt oder abgelöst durch weitere Ansätze, zum Beispiel durch das Paradigma der objektorientierten Programmierung.

⁷ VV&A: steht für *verification, validation, and accreditation* – Verifizierung, Validierung und Zertifizierung.

hin, dass eine analytische Lösung des Problems – sofern diese möglich ist – vorzuziehen ist, da jede Simulation nur eine Annäherung des betrachteten Systems ist. Auch sollte man sich durch die scheinbare Präzision – oft unterstrichen durch den Einsatz von Animationen - der Ergebnisse nicht verführen lassen: Fehler in der Modellierung machen die Simulationsergebnisse wertlos. Gleichzeitig ist die Validierung des Simulationsmodells eine nicht triviale Aufgabe (McHaney, 1991). Auch die Qualität der Eingabedaten ist zu beachten: Stimmt diese nicht, sind auch keine aussagekräftigen Ergebnisse zu erwarten (Chung, 2004). Die Ergebnisse selbst sind zu interpretieren, Simulationen liefern keine einfachen Antworten auf komplexe Probleme, in einer möglichen Lösung sollten alle beteiligten Systemelemente betrachtet werden (Chung, 2004).

Nicht zu vernachlässigen sind auch die teilweise großen Aufwände, die bei der Erstellung einer Simulation anfallen: Ausgaben für Soft- und Hardware sowie die Gehälter der Mitarbeiter für Analyse, Implementierung, Betrieb und Auswertung können hoch sein (McHaney, 1991).

Law (1991, S. 114) diskutiert ausführlich die möglichen Fehlerquellen bei der Arbeit mit Simulationen, hierunter fallen fehlerhafte Modellierungen, mangelnde Ausbildung der beteiligten Personen und Unterschätzung des konzeptionellen und organisatorischen Aufwands eines komplexen Simulationsprojekts.

5 Taxonomien

5.1 Grundlegende Kategorisierungen

Sulistio, Yeo & Buyya (2004) unterscheiden - wie in Abbildung 3 dargestellt - Simulationen nach 3 Merkmalen:

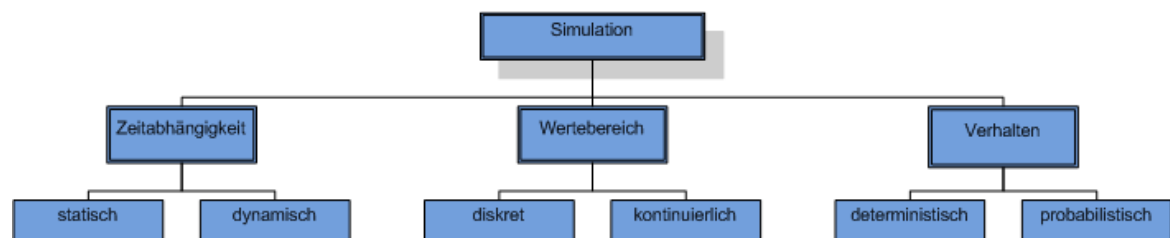


Abbildung 3: Taxonomie der Simulation nach Sulistio et al. (2004)

Zeit

Ein Unterscheidungsmerkmal ist die Berücksichtigung der Größe *Zeit* in der Simulation. Haben die Größen einer Simulation eine zeitliche Abhängigkeit, so wird sie **dynamisch** genannt. Zeigen die Größen einer Simulation keine zeitliche Änderung, so heißt sie **statisch**.

Beispiel: Die Suche eines optimalen Pfades in einem Graphen kann statisch simuliert werden.

Der Temperaturverlauf in einer Wohnung unter Einfluss von Außentemperatur und Heizleistung wird gewöhnlich dynamisch simuliert.

Zusätzlich wird zwischen zeitdiskreten und zeitkontinuierlichen Systemen unterschieden: **Zeitkontinuierlich** heißt ein System dann, wenn alle Zustandsgrößen stetige Funktionen der Zeit sind. Ändern die Zustandsgrößen ihre ansonsten konstanten Werte sprunghaft, spricht man von einem **zeitdiskreten** System. Beispiel für erstere Systeme sind physikalische Phänomene wie Wärmeausbreitung, ein zeitdiskretes System ist ein Computerprogramm mit seinen Variablen.

Wertebereich

Die Wertebereiche der Zustandsvariablen einer Simulationen können entweder **diskret** oder **kontinuierlich** sein. Der Flugweg einer Rakete ist eine kontinuierliche Simulation. Im Schachspiel nehmen die Steine nur bestimmte, festgelegte Positionen ein, der Wertebereich ist also diskret.

Verhalten

Führt eine Simulation bei denselben Eingabewerten immer zu den gleichen Ausgabewerten, so verhält sich die Simulation **deterministisch**. Treten bei denselben Eingabewerten gewöhnlich unterschiedliche Ausgabewerte auf, so zeigt die Simulation ein **probabilistisches** Verhalten. Es gibt verschiedene Gründe für die Nutzung von Zufallselementen in einer Simulation: Mit Hilfe eines Zufallsgenerators kann Rauschen erzeugt werden, es können damit Unsicherheiten abgebildet werden. Die Simulation stochastischer Prozesse – wie z.B. die *Brownsche Molekularbewegung* – ist ebenfalls essentiell abhängig von Zufallselementen (Bungartz et al., 2009).

Ein Beispiel zur Darlegung des Unterschieds: Die Simulation des Flugweges einer Rakete bei gleichen äußeren Bedingungen und innerem Zustand ist deterministisch. Treten unterwegs immer wieder zufällig⁸ Turbulenzen auf, so ist die Simulation probabilistisch.

5.2 Werkzeugunterstützung

Ebenfalls bei Sulistio et al. (2004) findet sich eine Kategorisierung der zur Durchführung von Simulationen benutzten Werkzeuge.

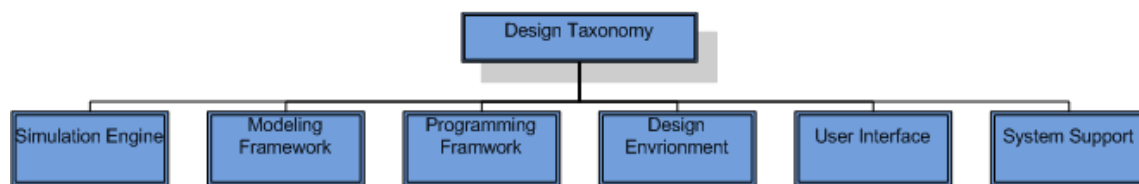


Abbildung 4: Taxonomie der Werkzeuge zur Erstellung und Durchführung einer Simulation nach Sulistio et al. (2004)

Simulation Engine

Die Simulation Engine ist zuständig für die Ausführung des Simulationsmodells. Schwetman (1996) definiert sie als Sammlung von Komponenten, besonderen Eigenschaften und Unterstützungsfunktionen, die für die Implementierung eines Simulationsmodells notwendig sind.

Nach Sulistio et al. (2004) wird zwischen paralleler und serieller Ausführung unterschieden. Parallele Ausführung verkürzt die Gesamtlaufzeit der Simulation, gleichzeitig werden erhöhte Anforderungen an die Hardware gestellt: Es müssen mehrere Prozessoren zur Verfügung stehen. Auch das Simulationsmodell muss so aufgebaut sein, dass seine Ausführung sich parallelisieren lässt. Die Implementierung von Simulationen mit paralleler Ausführung ist auch aus softwaretechnischer Sicht schwieriger als die entsprechende serielle Variante.

Eine weitere Kategorie zur Einordnung von Simulation Engines ist deren Funktionsweise: Es werden die drei Kategorien **kontinuierlich**, **diskret-eventbasiert** und **hybrid** unterschieden (Law, 1991). Der **kontinuierliche** Simulationsansatz zeichnet sich dadurch aus, dass der Zustandswech-

⁸ Nach dem Ansatz des Deterministischen Chaos sind die Turbulenzen nicht als zufällig zu bewerten. Um den Flugweg der Rakete simulieren zu können, ist die Betrachtung der Turbulenzen als probabilistisches Simulationselement eine geeignete Systemvereinfachung. Im anderen Fall würde der Aufwand für die Simulation auf ein nicht beherrschbares Maß ansteigen.

sel des Systems stetig erfolgt - stetig in dem durch die Verwendung von Rechnern begrenztem Sinne: So wie Fließkommazahlen binär nur mit endlicher Genauigkeit dargestellt werden können (Goldberg, 1991), können Rechner nicht kontinuierliche Zustandsänderungen verarbeiten, sie erfolgen immer in kleinen, diskreten Zeitschritten (Robinson, 2004, S. 25).

Basis des kontinuierlichen Simulationsansatzes sind oft Differentialgleichungen, mit denen das System der realen Welt beschrieben werden kann. In jedem Simulationsschritt werden diese Gleichungen numerisch gelöst. Weniger aufwändig ist der Fall, in dem die Gleichungen eine analytische Lösung besitzen.

Kennzeichen des **diskret-eventbasierten** Simulationsansatzes sind die nur zu bestimmten Zeitpunkten auftretenden Zustandsänderungen des Simulationsmodells. Zu den Zeitpunkten treten Ereignisse („Events“) ein, die zu einer Zustandsänderung führen. Ein Beispiel ist die Simulation der Nutzung eines Geldautomaten: Kunden kommen, benutzen den Geldautomaten und gehen wieder (McHaney, 2009). In der Zwischenzeit gibt es keine problemrelevante Zustandsänderung des Geldautomaten.

Hybrid wird ein Simulationsansatz genannt, wenn ein Simulationsmodell mit beiden vorher genannten Ansätzen arbeitet. Das sind zum Beispiel alle Systeme, in denen physikalische Prozesse und Ereignisse gleichzeitig auftreten (Cassandras & Lafortune, 2007).

Modeling Framework

Das Modeling Framework dient als Werkzeug zur Erstellung des Simulationsmodells. Es liefert einen Rahmen zur Modellierung. Näherungsweise ausgedrückt beschreibt es die Verteilung des Kontrollflusses zwischen der Simulation Engine und den Komponenten des installierten Simulationsmodells. Sulistio et al. (2004) unterscheiden für die Kategorie der diskret-eventbasierten Simulation zwischen entitäts- und ereignisbasierten Ansätzen, die auch in einem hybriden Ansatz gemeinsam auftauchen können. Garrido (2001) differenziert drei verschiedene Ansätze: Neben dem entitätsorientierten Ansatz, den er prozess-interaktionsorientiert nennt, und dem eventorientierten Ansatz kennt er auch den aktivitätsorientierten Ansatz: Aktivitäten werden immer dann ausgeführt, wenn mit der Aktivität verknüpfte Bedingungen erfüllt sind.

Der entitätsorientierte Ansatz zeichnet sich dadurch aus, dass mehrere Entitäten unabhängig voneinander agieren und gegebenenfalls miteinander über Nachrichten kommunizieren. Weiter entwickelt findet sich dieser Ansatz in der agentenbasierten Modellierung (ABM) wieder: Hier wird ein System modelliert als eine Menge von selbständig entscheidenden Entitäten, die Agenten genannt werden. Die Weiterentwicklung des ABM in letzter Zeit wurde durch den Anstieg der Leistungsfähigkeit der Rechner möglich gemacht (Bonabeau, 2002).

Programming Framework

Als Programming Framework definieren Sulistio et al. (2004) das Programmier-Paradigma, das im Rahmen der Implementierung der Simulation zur Anwendung kommt. Unterschieden wird zwischen dem objektorientierten Ansatz und dem strukturierten Ansatz. Grundsätzlich kommen für das Programming Framework nur diejenigen Ansätze in Frage, die auch für herkömmliche Softwareentwicklung benutzt werden können, da die Entwicklung von Simulationen auch Softwareentwicklung ist. Hier sind objektorientierter und strukturierter Ansatz die vorherrschenden. Das wird durch eine Bestandsaufnahme der gängigen Simulationsentwicklungsumgebungen

bestätigt. Der Ansatz der strukturierten Programmierung versucht, durch den Einsatz von abgegrenzten Codeeinheiten (Prozeduren, Funktionen und Modulen) und klaren Kontrollstrukturen (Sequenz, Iteration und Selektion) die Klarheit und Verständlichkeit des Programmcodes zu gewährleisten, um damit Wartbarkeit und Wiederverwendung zu ermöglichen (Dahl, Dijkstra & Hoare, 1972). Beim objektorientierten Ansatz – zeitlich jünger und teilweise aufbauend auf den Prinzipien der strukturierten Programmierung – erfolgt die Programmierung auf Basis eines Objekt-Modells. Wichtige Prinzipien sind Data Hiding und Vererbung (Stroustrup, 1991). Die Kommunikation der Objekte miteinander erfolgt durch den Aufruf von Methoden.

Eine weitere Alternative ist der generative Ansatz. Hier wird ein Modell mit zumeist grafischen Editoren erstellt, und aus dem Modell erfolgt schließlich die Generierung des Programmcodes. Anpassungen und Erweiterungen des generierten Programmcodes, die gelegentlich noch fällig werden können, sind dann jedoch auch mit Hilfe der Programmiersprache zu erledigen, in die generiert wird.

Design Environment

Mit dem Begriff Design Environment bezeichnen Sulistio et al. (2004) die Art und Weise, mit der die Erstellung neuer Simulationsmodelle durch die Entwicklungsumgebung unterstützt wird. Er sieht zwei Ansätze: Auf der einen Seite stehen, wie in Kap. 3.5 schon dargestellt, spezialisierte Simulationsprogrammiersprachen, die Sprachkonstrukte bereitstellen, mit deren Hilfe immer wiederkehrende Aufgaben aus dem Bereich der Simulation gelöst werden können. Zum anderen gibt es die Möglichkeit einer Bibliothek: Elemente aus Bibliotheken können bei der Entwicklung einer Simulation genutzt werden. Je nach unterstütztem Programming Framework können das sowohl Objekte als auch andere Codeelemente sein. Beide Ansätze schließen einander nicht aus und können in einer Entwicklungsumgebung kombiniert werden.

Analog zum Programming Framework können auch für das Design Environment Techniken aus der der herkömmlichen Softwareentwicklung eingesetzt werden. Dazu zählt die Nutzung von Templates für Simulationsmodelle (Mertins, Rabe & Jaekel, 2000). Templates sind Kopier-Vorlagen, deren Kopien vom Benutzer abgeändert werden können und nach dem eigenständigen Abspeichern keine Verbindung mehr zur Vorlage haben.

User Interface

Die Benutzeroberfläche- *das User Interface* - legt fest, wie der Benutzer mit dem Simulationswerkzeug interagiert. Grundsätzlich wird unterschieden zwischen dem Designwerkzeug und der Ausführungsumgebung. Für beide Komponenten sind grafische Benutzeroberflächen möglich: Bei Designwerkzeugen kann das Simulationsmodell grafisch dargestellt werden und es kann grafisches Modifizieren des Modells („Drag and Drop“) möglich sein. Die Ausführungsumgebung kann Grafiken und Animationen einbetten, um den Verlauf und das Ergebnis der Simulation anschaulich darzustellen. Von nichtgrafischem Design wird gesprochen, wenn Programmcode geschrieben wird und/oder Konfigurationsdateien geändert werden. Sulistio et al. (2004) vermissen integrierte Entwicklungsumgebungen („IDE“), wie sie in der Softwareentwicklung eingesetzt werden und fordern sie für die Zukunft: Eine einzige grafische Umgebung, in der sowohl Simulationen entwickelt werden als auch ausgeführt werden können. Inzwischen gibt es derartige IDEs: Von Varga & Hornig (2012;2008) wird beispielsweise demonstriert, wie eine solche IDE (hier

ursprünglich erstellt für die Netzwerksimulation, jetzt aber auch zu anderen Zwecken nutzbar) aussehen kann.

System Support

Unter dem Begriff *System Support* sortieren Sulistio et al. (2004) die Funktionen einer Entwicklungsumgebung für Simulationen ein, die die Arbeit mit Simulationen erleichtern. Dazu gezählt wird Code-Generierung aus dem Modell heraus. Dieser Ansatz erlaubt die Erstellung eines Simulationsmodells auf einer höheren Abstraktionsebene, das „Schreiben“ des Codes in der Programmiersprache erfolgt automatisch durch einen Code-Generator. Diese Vorgehensweise ist auch in der herkömmlichen Softwareentwicklung bekannt, es wird hier *Model Driven Development* (MDD) genannt (Selic, 2003).

Ein Online-Debugger ist ein wertvolles Hilfsmittel, um effektiv Fehler in der Simulationsimplementierung zu finden, da sich der Entwickler den Programmcode im Kontext der laufenden Simulation anschauen kann.

Ebenso bieten manche Simulationswerkzeuge die Möglichkeit, das Modell zu validieren, d.h. es wird mit den Beobachtungen aus der realen Welt abgeglichen. Von der University of Southern California (2012) werden Validierungstests für einen Netzwerksimulator beschrieben. Hier ist die Validierung technisch recht einfach umsetzbar mit Hilfe der Aufzeichnung von realen Netzwerknachrichten. Mit den Aufzeichnungsdaten werden dann die simulierten Nachrichten verglichen.

Auch wird eine Statistik-Schnittstelle zum System Support gezählt: Diese hilft dabei, Daten zu erzeugen, mit deren Hilfe sich die Simulationsläufe auswerten lassen. Sie steht repräsentativ für die schon im Kap. 3.5 genannten simulationsbezogenen Aspekte einer speziellen Simulationssprache.

5.3 Exaktheit des Simulationsmodells

Simulationsmodelle können einen unterschiedlichen Grad der Exaktheit des abzubildenden Systems besitzen. Mit zunehmender Detaillierung des Modells steigt der notwendig Aufwand (in Form von Zeit und Kosten). Eine iterative Verfeinerung des Simulationsmodells ermöglicht es, Erfahrungen mit den auftretenden Problemen zu sammeln (McHaney, 1991). Gleichzeitig kann besser der Kompromiss zwischen Aufwand und Zweckerfüllung des Simulationsmodells getroffen werden⁹.

McHaney (1991) sieht eine Konzeptsimulation als Ergebnis einer frühen Phase in der Entwicklung von Simulationen: Mit Hilfe von Rapid Prototyping-Werkzeugen wie zum Beispiel einem Tabellenkalkulationsprogramm wird ein grobes Simulationsmodell aufgestellt. Das Vorgehen kann auch Züge einer Machbarkeitsstudie haben („proof of concept“). Andere Bezeichnungen für diesen Typ der Simulation sind „Rough Cut Simulation“ und „Rapid Model“. Eines der möglichen Probleme ist eine unzulässig starke Vereinfachung des Modells, die zu keiner Lösung für das zugrundeliegende Problem führt.

⁹ Dies ist auch ein Merkmal des agilen Softwareentwicklungsprozesses: Software so genau wie nötig zu entwickeln, aber nicht genauer (Martin, 2002).

Low Fidelity und *High Fidelity* sind die Begriffe, mit denen in der Literatur die Originaltreue einer Simulation beschrieben wird¹⁰. *Fidelity* wird definiert als Grad, mit dem ein Modell den Zustand und das Verhalten eines Systems der realen Welt wiedergibt (Gross, 1999).

Reynolds (2009) bezieht die Genauigkeit des Simulationsmodells auch auf den Zweck der Simulation: Eine Simulation, die der Lösung eines konkreten Problems der realen Welt im Sinne einer Abwägung von Alternativen dient, benötigt eine hohe Genauigkeit. Alle wichtigen Elemente sind modelliert und das Modell ist sehr stabil. Hingegen muss ein Modell, das dem Erzeugen von Verständnis dient, weder komplett noch in allen Einzelheiten exakt sein. Es kann der ständigen Weiterentwicklung durch den Benutzer unterliegen und dient dem Lernprozess. Daher liegt der Schwerpunkt von Game Engines, in denen physikalische Vorgänge simuliert werden, gewöhnlich auch auf die Einhaltung der Zeitanforderungen. Die Exaktheit der Simulation ist ein zweitrangiges Ziel, dessen Erfüllungsgrad von der zur Verfügung stehenden Rechenleistung abhängt (Boeing & Bräunl, 2007).

5.4 Arten von Simulationen

Nachdem bisher eine Kategorisierung der Simulationen nach bestimmten Merkmalen erfolgte, wird in diesem Kapitel ein Überblick der in der Literatur vorherrschend anzufindenden Simulation gegeben.

5.4.1 Monte-Carlo-Simulation

Der Typ *Monte-Carlo-Simulation* gehört zu den statischen Simulationen, sie ist nicht zeitabhängig. Probleme, die mit der Monte-Carlo-Simulation gelöst werden können, sind beispielsweise mathematische oder physikalische Fragestellungen, zu denen eine analytische Lösung nur schwer oder gar nicht zu finden ist (Rubinstein & Kroese, 2008). Aufgaben derartig hoher Komplexität liegen nicht im Fokus dieser Arbeit. Daher wird die Monte-Carlo Simulation hier nicht detailliert betrachtet.

Das grundlegende Prinzip ist, dass Zufallsvariablen definiert werden, für diese eine Reihe von Zufallsexperimenten durchgeführt werden und die Ergebnisse aufgezeichnet und akkumuliert werden. Auf Basis des Gesetzes der großen Zahlen kann dann eine Lösung zum Ausgangsproblem gefunden werden. Ein schönes Beispiel ist die Bestimmung der Zahl Pi (Schmidt, 2008; Steinhausen, 1994): Zufallsgesteuert mit zwei Gleichverteilungen wird ein Punkt im Einheitsquadrat verteilt. Für den jeweiligen Punkt wird festgestellt, ob er inner- oder außerhalb des Einheitskreises liegt, und dieses entsprechend verbucht. Mit zunehmender Anzahl von Wiederholungen dieses Experiments kann aufgrund der Formel für den Kreisflächeninhalt Pi immer genauer berechnet werden. An diesem Beispiel lässt sich auch erkennen, dass die Benutzung der Monte-Carlo-Simulation stark mit dem Einsatz von Rechnern verknüpft ist: Die Durchführung ist so arbeitsaufwändig, dass sie nicht manuell durchgeführt werden kann.

Die Monte-Carlo-Simulation ist nicht zu verwechseln mit anderen Simulationsarten, bei denen Eingabedaten mit stochastischen Methoden generiert werden. Beispiele für letztere sind einige Arten der *Discrete Event Simulation*, bei denen Ereignisse oft zufallsgesteuert auftreten.

¹⁰ Diese Begriffe werden auch für nicht-digitale Simulationen, z.B. zur Ausbildung im medizinischen Bereich angewendet, die Erläuterung hier bezieht sich aber auf digitale Simulationen.

5.4.2 Ereignisorientierte Simulation

Kennzeichen der ereignisorientierten Simulation, die zu den dynamischen Simulationen zählt, sind zu bestimmten Zeitpunkten auftretende Ereignisse. Sie sind von der Simulation zu verarbeiten. Statusänderungen des Simulationsmodells werden verursacht durch das Auftreten der Ereignisse (Wainer, 2009; Zimmermann, 2008). Andere Bezeichnungen sind *Discrete Event Systems* oder *Discrete Event Simulation*. Anwendungsfälle sind beispielsweise die Simulation von Produktions- oder Transportabläufen. Die ereignisorientierte Simulation nutzt häufig Mittel der Wahrscheinlichkeitsrechnung, um eintretende Ereignisse zu simulieren. Für die Kapazitäts-Auslegung eines Grills eines Verkaufsstandes von echten Thüringer Bratwürsten als Beispiel kann eine Simulation modelliert werden, bei der die Ankunftsrate von neuen Kunden durch eine Verteilungsfunktion gesteuert wird. Diese Verteilungsfunktion kann von Tages- und Jahreszeit, vom Wetter und anderen Parametern abhängig sein.

In einer ereignisorientierten Simulation können auch kontinuierliche Prozesse integriert werden. In obigem Beispiel ist der Garprozess einer Bratwurst ein solcher (Schriber & Brunner, 2005).

Ein besonderes Merkmal dieser Simulationsart ist die Abarbeitung der Zeit: Zustandsänderungen erfolgen nur zu den Zeitpunkten, an denen Ereignisse auftreten. Somit kann nach der Abarbeitung eines Ereignisses zum nächsten – bereits zeitlich terminierten - Ereignis gesprungen werden. Die dazwischenliegende simulierte Zeit wird ausgelassen. Das ist ein wesentlicher Unterschied zur kontinuierlichen Simulation, in der die komplette simulierte Zeit einmal durchlaufen wird¹¹.

5.4.3 Kontinuierliche Simulation

Kontinuierliche Simulationen gehören ebenfalls zu den dynamischen Simulationen – die Werte der Statusvariablen ändern sich kontinuierlich über die Zeit. Gewöhnlich wird das zugrundeliegende Modell mit Hilfe von Differentialgleichungen ausgedrückt, für die eine analytische Möglichkeit nur sehr schwer oder gar nicht möglich ist. Im einfachen Fall ist der Wert einer Statusvariablen nur vom Zeitpunkt t abhängig und lässt sich ohne die numerische Lösung berechnen (Law, 1991). Anwendungsgebiete der kontinuierlichen Simulation beinhalten stetige Prozesse. Die sogenannte numerische Wettervorhersage ist ein sehr bekanntes Anwendungsgebiet dieser Simulationsart. Anhand dieses Beispiels lässt sich eines der dringlichsten Probleme der kontinuierlichen Simulation verdeutlichen: fehlende Rechenleistung. Die Genauigkeit der Wettervorhersage ließe sich bei verfeinerten Modellen steigern, diese würden aber die gegenwärtigen Rechnerkapazitäten sprengen (DWD, 2012).

5.4.4 Hybride Simulation

Simulationen, die sowohl ereignisorientierte als auch kontinuierliche Simulationsmechaniken aufweisen, werden hybrid genannt. Gibt es in einer ereignisorientierten Simulation eine Statusvariable, die kontinuierlich Werte annimmt und die nicht nur vom Wert der simulierten Zeit abhängig ist, die also kontinuierlich numerisch berechnet werden muss, dann muss diese Simulation auch nach Prinzipien der kontinuierlichen Simulation arbeiten und wird zu einer hybriden Simulation.

¹¹ In der jeweiligen Schrittweite für die simulierte Zeit (Die Simulation ist in diesem Sinne auch nicht kontinuierlich, sondern diskret – das ist aber eine generelle Eigenschaft bei der Benutzung von Rechnern (Robinson, 2004)).

Cassandras & Lafortune (2007) zeigen ein schönes Beispiel, wie die Entscheidung, ob ein Simulationsmodell ereignisorientiert oder zeitgetrieben (d.h. nach kontinuierlichen Simulationsmechaniken) aufgebaut werden soll, vom Zweck des Modells, also von der Problemstellung, abhängt: Betrachtet man den technischen Datenfluss im Internet, so ist dieser in hohem Masse ereignisorientiert: Ein Datenpaket wird gesendet und wird empfangen, eine Anfrage wird gestellt und wird beantwortet. Möchte man aber Aussagen zu Transportkapazitäten des Internets machen, so werden die einzelnen Datenpakete zu einem Datenfluss zusammengefasst, der wiederum kontinuierlich ist, d.h. zu jedem Zeitpunkt lässt sich eine Aussage zu seiner Stärke machen. Dieses Beispiel lässt sich noch erweitern: Der Transport der Datenpakete beruht auf physikalischen Phänomenen (z.B. Elektrizität oder Licht), die wiederum mit kontinuierlichen Mechaniken simuliert werden können.

6 Standards, Normen und Formalismen

Obwohl jedes Simulationsmodell zunächst sehr individuell erscheint, gibt es Bestrebungen, allgemeine Regeln für das Gebiet der Simulation zu entwickeln (Tolk et al., 2011). Zu den Standardisierungsorganisationen gehören:

- Simulation Interoperability Standards Organization (SISO)
- Institute of Electrical and Electronics Engineers (IEEE)
- International Organization for Standardization (ISO)

Die meisten Normen sind angesiedelt im Bereich der Kommunikation in verteilten Simulationen sowie bei wiederverwendbaren Objektmodellen (SISO, 2012).

Zeigler et al. (2000) schlagen eine M&S¹²-Mehrschichtenarchitektur vor, die die Zusammenarbeit bei der Entwicklung von Simulationsmodellen und die Durchführung von verteilten Simulationen ermöglichen soll. Zu den sechs verschiedenen Schichten gehören neben dem Network-, dem Simulation- und dem Modeling-Layer auch:

- **Search Layer:** Unterstützt die Suche nach geeigneten Modellen und assistiert intelligent bei der Zusammensetzung von Teilmodellen.
- **Decision Layer:** Ermöglicht unter anderem die Auswertung verschiedener Simulationsläufe, die Durchführung von Wenn-Dann-Analysen und von Optimierungen.
- **Collaboration Layer:** Erlaubt die Zusammenarbeit von Fachleuten, die mit unterschiedlichem (Teil-)Wissen über das zu lösende Problem zu einer gemeinsamen, umfassenden Lösung beitragen können.

Im Simulation-Layer werden Kommunikationsprotokolle der HLA (High Level Architecture) genutzt. Diese ist in der IEEE Norm 1516 definiert und legt u.a. das Format der Datenpakete fest, mit denen der Status von Simulationsobjekten ausgetauscht wird.

Ebenfalls gibt es Bestrebungen, formale Beschreibungen eines Simulationsmodells zu erstellen. Als Beispiel kann die *Discrete Event System Specification* (DEVS) genannt werden (Zeigler et al., 2000). Eine sogenannte *Atomic DEVS* besteht aus:

- einer Menge von Eingabe-Ereignissen

¹² M&S: Modeling and Simulation

- einer Menge von Ausgabe-Ereignissen
- einer Menge von Zuständen
- einer Funktion für die Lebensdauer der Zustände
- einer Funktion für die Statusänderung des Systems nach dem Eintritt eines Eingabeereignisses
- einer Funktion für die Statusänderung des Systems nach dem Ablauf der Lebensdauer eines Zustandes
- einer Funktion, die aus dem Systemzustand ein Ausgabe-Ereignis ableitet.

7 Abbildungsverzeichnis

Abbildung 1: Ansätze der Untersuchung eines Systems	4
Abbildung 2: Prozess zur Erzeugung einer Simulation	8
Abbildung 3: Taxonomie der Simulation	9
Abbildung 4: Taxonomie der Werkzeuge zur Erstellung und Durchführung einer Simulation	10

Anhang D: Fallstudie: Designanforderungen an bauphysikalische Simulationssoftware

Inhaltsverzeichnis

1 Einleitung.....	2
2 Anpassbarkeit am Beispiel von TRNSYS.....	2
2.1 Anpassungsmöglichkeiten von TRNSYS für den Benutzer.....	5
2.1.1 Projekt	5
2.1.2 Konfigurierbare Komponenten	5
2.1.3 Komponentenbibliotheken	6
2.1.4 Benutzerspezifische Komponenten.....	7
2.2 Schlussfolgerungen.....	7
3 Lernszenarien am Beispiel von CASAnova.....	8
3.1 Das Lernprogramm CASAnova	8
3.2 Schlussfolgerungen.....	10
4 Abbildungsverzeichnis	10

1 Einleitung

Ingenieurmässiges Vorgehen in der Bauphysik wird bereits mit Hilfe von Simulationssoftware unterstützt. In diesem Anhang werden zwei Beispiele von Softwarepaketen dargestellt, die bezüglich jeweils eines Aspekts als Vorbild – und damit als Grundlage - für die Spielplattform angesehen werden können. Zum einen wird die Konfigurierbarkeit von TRNSYS diskutiert, zum anderen wird die Abbildung von Lernszenarien mit Hilfe von CASAnova untersucht.

2 Anpassbarkeit am Beispiel von TRNSYS

Das Softwarepaket TRNSYS ist ein Werkzeug, das vornehmlich zur dynamischen Gebäudesimulation und zur thermischen Anlagensimulation genutzt wird. Es wurde an der *University of Wisconsin – Madison* entwickelt und existiert seit 1975 in einer kommerziellen Version. Sein Anwendungsbereich wurde seitdem stark erweitert: es eignet sich generell zur Modellierung von dynamischen Systemen: So lassen sich auch Verkehrsflusssimulationen und biologische Prozesse mit diesem Werkzeug abbilden.

Dieses Softwarepaket hat einen ähnlichen fachlichen Anwendungsbereich wie die zu konzipierende Spielplattform. Daneben ist es sehr vielseitig einsetzbar und durch den Anwender erweiterbar. Es hat verglichen mit gewöhnlichen Softwarelebenszyklen ein sehr hohes Alter erreicht und scheint noch immer aktuell zu sein – es wird u.a. in Normen als Referenz-Simulationswerkzeug genannt¹. Dies sind alles Gründe, einen tieferen Blick auf die Architektur und das Konzept von TRNSYS zu werfen, um Strukturen zu identifizieren, die auch für die geplante Spielplattform erfolgversprechend sein könnten.

¹ S. <http://sel.me.wisc.edu/trnsys/faq/faq.htm>, letzter Zugriff am 10.11.2012

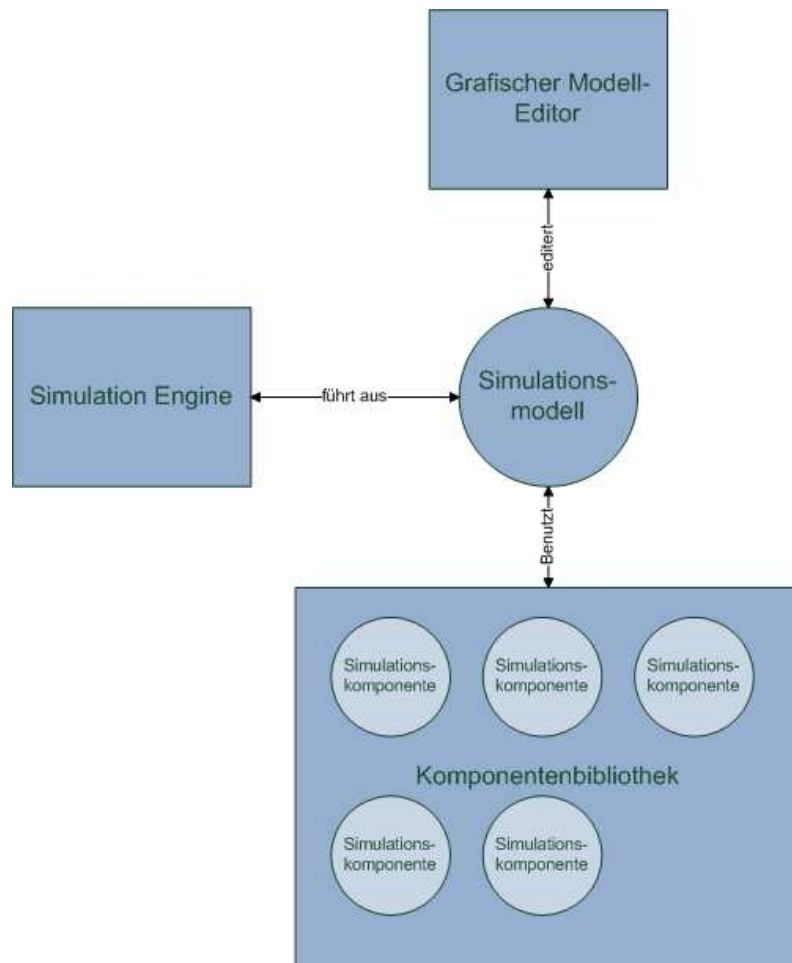


Abbildung 1: Architektur TRNSYS

Abbildung 1 zeigt die prinzipielle Struktur von TRNSYS: Zentrales Element ist das Simulationsmodell, das in Form einer Datei vorliegt. Dieses Modell wird mit Hilfe eines grafischen Editors bearbeitet, der in Abbildung 2 dargestellt ist. Das Modell selbst liegt in Form einer lesbaren, auch manuell änderbaren Textdatei vor, ein Ausschnitt ist in Abbildung 3 zu sehen. Es enthält Verweise auf Komponenten, die in Form von Komponentenbibliotheken existieren. Gewöhnlich werden diese Komponenten eigenständig entwickelt, sie genügen einer Programmierschnittstelle (API) und werden in FORTRAN programmiert, einer relativ alten Programmiersprache, die vorzugsweise zu numerischen Berechnungen eingesetzt wird. Das Modell wird mit Hilfe einer Simulation Engine ausgeführt. Simulationsergebnisse werden in Grafiken, Berichten oder Dateien ausgegeben. Schlüsseleigenschaft ist die beliebige Verknüpfung von Ausgabevariablen einer Komponente mit den Eingabevariablen einer anderen Komponente.

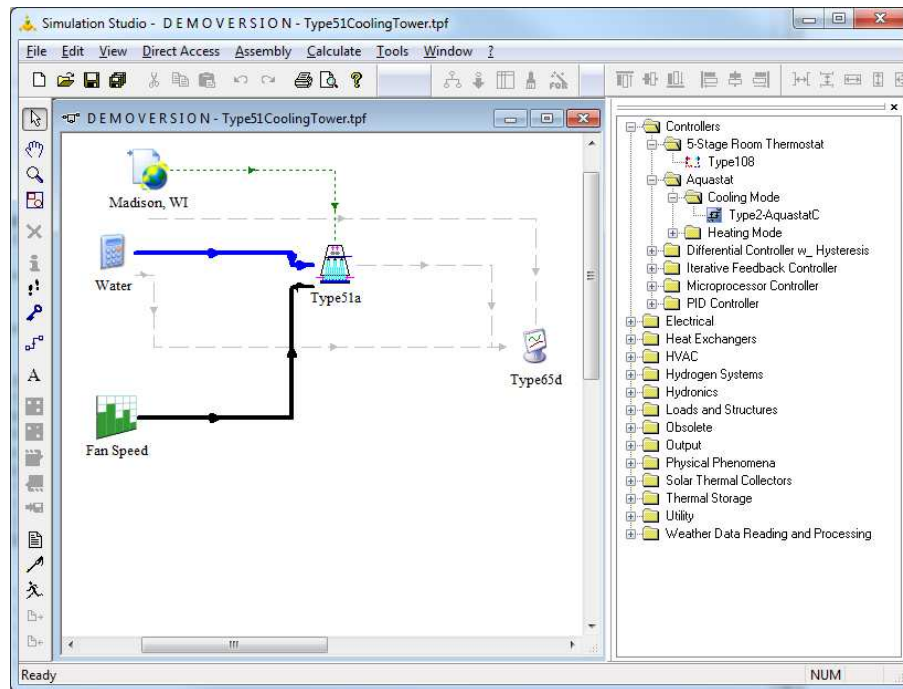


Abbildung 2: Bildschirmkopie des grafischen Modell Editors von TRNSYS („Simulation Studio“)

```

*****
*** Control cards
*****
* START, STOP and STEP
CONSTANTS 3
START=4444
STOP=4468
STEP=.125
SIMULATION  START      STOP  STEP ! Start time      End time      Time step
TOLERANCES 0.001 0.001      ! Integration      Convergence
LIMITS 30 30 30      ! Max iterations  Max warnings  Trace limit
DFQ 1      ! TRNSYS numerical integration solver method
WIDTH 80      ! TRNSYS output file width, number of characters
LIST      ! NOLIST statement
          ! MAP statement
SOLVER 0 1 1      ! Solver statement      Minimum relaxation factor
NAN_CHECK 0      ! Nan DEBUG statement
OVERWRITE_CHECK 0      ! Overwrite DEBUG statement
TIME_REPORT 0      ! disable time report
EQSOLVER 0      ! EQUATION SOLVER statement
* User defined CONSTANTS

* Model "Type51a" (Type 51)
*
UNIT 2 TYPE 51      Type51a
*$UNIT NAME Type51a
*$MODEL .\HVAC\Cooling Towers\External Performance File\Type51a.tmf
*$POSITION 396 168
*$LAYER Water Loop #
PARAMETERS 11
2      ! 1 Calculation mode
1      ! 2 Flow geometry
1      ! 3 Number of tower cells
106681.422472      ! 4 Maximum cell flow rate
7.457122      ! 5 Fan power at maximum flow
9.999999      ! 6 Minimum cell flow rate
1.0      ! 7 Sump volume
15.0      ! 8 Initial sump temperature
30      ! 9 Logical unit
24      ! 10 Number of data points
1      ! 11 Print performance results?
INPUTS 6
EWT      ! Water:EWT ->Water inlet temperature
m_dot_water      ! Water:m_dot_water ->Inlet water flow rate
6,1      ! Madison, WI:Dry bulb temperature ->Dry bulb temperature
6,3      ! Madison, WI:Wet bulb temperature ->Wet bulb temperature
0,0      ! [unconnected] Sump make-up temperature
    
```

Abbildung 3: TRNSYS: Ausschnitt einer Simulationsmodelldatei

2.1 Anpassungsmöglichkeiten von TRNSYS für den Benutzer

Der Benutzer von TRNSYS kann an verschiedenen Stellen eingreifen, um seine Anforderungen in ein Simulationsszenario umzusetzen. Im Folgenden werden die verschiedenen Möglichkeiten diskutiert.

2.1.1 Projekt

Ein Projekt wird mit Hilfe eines grafischen Editors - dem sogenannten „Simulation Studio“ - angelegt. Ein Projekt enthält das zu simulierende System in Form von Komponenten, Datenquellen und –senken sowie den entsprechenden Verbindungen. Die wahlfreie Zuordnung von Ausgabevariablen einer Komponente zu Eingabevariablen einer verbundenen Komponente (ein Beispiel ist in Abbildung 4 dargestellt) erlaubt zum einen einen hohen Freiheitsgrad bei der Erstellung des Modells, zum anderen aber auch die unabhängige Entwicklung der Komponenten: Jede Komponente kann für sich entwickelt werden, sie kommuniziert lediglich über die Ein- und Ausgabevariablen mit anderen Komponenten, muss aber keinerlei Informationen über deren internen Aufbau verarbeiten.

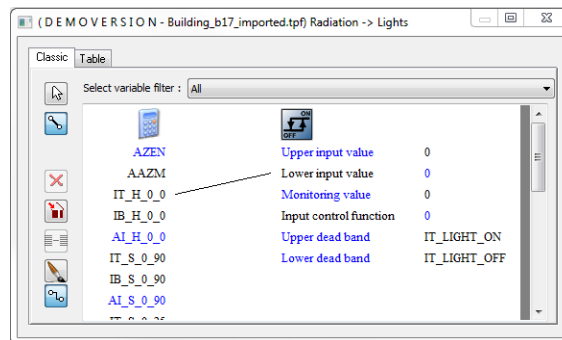


Abbildung 4: TRNSYS: Zuweisung von Aus- und Eingabevariablen durch den Verknüpfungsektor

2.1.2 Konfigurierbare Komponenten

Jede Komponente für sich kann selbst noch einmal konfigurierbar sein, d.h. es können charakteristische Merkmale einer Komponente von außen durch den Ersteller des Szenarios vorgegeben werden. Im Extremfall kann eine so weitgehende Konfiguration möglich sein, dass ein eigener Editor benötigt wird. Beispiel ist hier eine Komponente, die im Projekt als „Gebäude“ erscheint, bei der Konfiguration aber einen speziellen Editor² öffnet, der sogar eine Strukturänderung des Gebäudes ermöglicht (Abbildung 5).

² Der Editor ist TRNBuild, eine wesentliche Komponente von TRNSYS zur Modellierung eines Gebäudes.

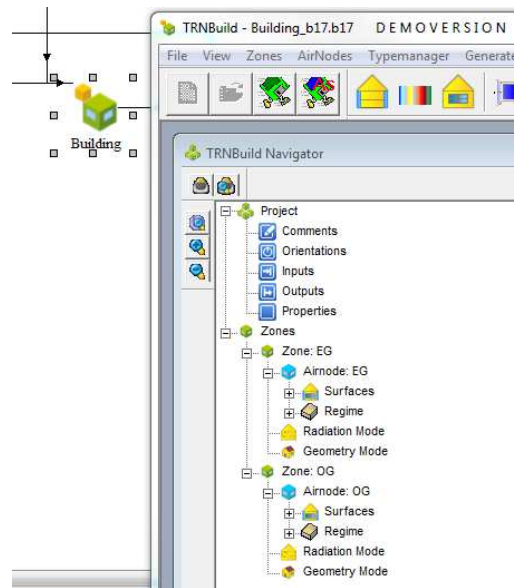


Abbildung 5: TRNSYS: Gebäude-Komponente und zugehöriger Editor

2.1.3 Komponentenbibliotheken

Komponenten werden zu Bibliotheken zusammengefasst. Es haben sich in der Zwischenzeit verschiedene Bibliotheken etabliert, im Folgenden findet sich eine Auswahl³:

- **TRNSYS Standard Components**
Diese Bibliothek wird mit zusammen TRNSYS ausgeliefert.
- **TESS Libraries**
Dies sind Komponenten, die von der Firma TESS⁴ entwickelt wurden und als eigenständige Bibliothek vertrieben werden.
- **TRNLIB Components**
Diese Bibliotheken sind aus der Benutzer-Gemeinschaft von TRNSYS entstanden. Benutzer von TRNSYS lösen ihr Problem durch die Eigenentwicklung von Komponenten oder durch die Erstellung von Konfigurationsdaten und stellen diese wieder der Allgemeinheit zur Verfügung⁵.
- **TRNAUS Components**
In dieser Bibliothek finden sich Komponenten, die ursprünglich zur Simulation von thermischen Solaranlagen in Australien zur Anwendung gekommen sind.
- **Transsolar Non-Standard Components**
Die Firma Transsolar ist der deutsche Betreuer von TRNSYS. Sie bietet auf ihrer Website ebenfalls TRNSYS-Komponenten verschiedener Autoren zum Kauf an⁶.

³ Diese Bibliotheken werden auf der TRNSYS-Homepage genannt.

(<http://www.trnsys.com/features/libraries.php>, letzter Zugriff 11.09.2012)

⁴ TESS (Thermal Energy System Specialists, LLC) mit Sitz in Madison, WI, USA ist das kommerzielle Unternehmen, dass für die Weiterentwicklung von TRNSYS nach der Ausgliederung aus dem universitären Bereich zuständig ist.

⁵ Das *Solar Energy Laboratory* der *University of Wisconsin – Madison* fungiert als Makler für diese Art von Komponenten: Es bietet eine Website mit Beschreibungen und Download-Möglichkeiten an (<http://sel.me.wisc.edu/trnsys/trnlib/library16.htm>, letzter Zugriff am 10.09.2012).

⁶ S. http://trnsys.de/docs/komponenten/komponenten_ts_en.htm, letzter Zugriff am 10.09.2012.

Die Betrachtung der Bibliotheken zeigt, dass aus der Erweiterbarkeit von TRNSYS benutzerspezifische Lösungen entstehen und diese dann wieder anderen potentiellen Benutzern zur Verfügung gestellt werden. Auf diesem Weg fördert die Erweiterbarkeit eine Benutzer-Gemeinschaft und schließlich Wiederverwendung. Konsequenz ist eine höhere Verbreitung und größerer Einfluss der Software.

2.1.4 Benutzerspezifische Komponenten

Sind die vorher beschriebenen Möglichkeiten nicht ausreichend, um ein geeignetes Simulationsmodell zu erstellen, bietet die Erstellung einer spezifischen Komponente durch Programmierung größtmögliche Flexibilität. Hierzu bietet TRNSYS eine Programmierschnittstelle an, die im mitgelieferten „Programmer’s Guide“ beschrieben wird. Die Komponenten werden in der Sprache FORTRAN programmiert: der Entwickler erstellt Quellcode für eine Funktion, deren Signatur⁷ durch TRNSYS vorgegeben wird. Aus dem Quellcode wird mit Hilfe eines Übersetzungsprogramms (engl. compiler) eine Binärdatei (Dynamic Link Library (DLL)) für das Betriebssystem MS Windows erstellt. Die erzeugte Komponente kann in Projekten genutzt werden und in eine Bibliothek integriert werden.

2.2 Schlussfolgerungen

Am Beispiel von TRNSYS lassen sich mehrere Faktoren aufzeigen, die zum Erfolg einer Simulationsplattform beitragen. Diese werden im Folgenden beschrieben.

Erweiterbarkeit

TRNSYS bietet eine Schnittstelle, die eine Anpassung von TRNSYS an die eigenen Bedürfnisse erlaubt. Es gibt nicht den Anspruch auf die allumfassende Lösung, stattdessen gibt die Software einen Rahmen vor, der zur Erstellung spezifischer Lösung erweitert werden kann.

Modularität

Die Zusammenarbeit verschiedener Teile der Software wird über Schnittstellen sichergestellt. Ein Beispiel ist das Format der Projektdatei – dieses ist die Schnittstelle sowohl für den Projekt-Editor („Simulation Studio“) als auch für die Simulation Engine: Beide Softwareeinheiten können unabhängig voneinander gepflegt oder sogar ausgetauscht werden – es muss lediglich sichergestellt sein, dass eine Projektdatei gelesen bzw. geschrieben werden kann.

Die Komponenten sind ein weiteres Beispiel für das Prinzip der Modularität: Sie werden jede für sich unabhängig voneinander entwickelt. Es ist lediglich gefordert, dass sie Parameter beschreiben und diese – im Falle von Eingabeparametern - verwerten können sowie – im Falle von Ausgabeparametern – mit Werten belegen.

Konfigurationsmöglichkeiten

Die Plattform lässt sich weitestgehend konfigurieren: Die Komponenten innerhalb eines Projektes können frei gewählt werden. Die Verbindungen der Komponenten untereinander in Form von Ausgabeparameter-Eingabeparameter-Zuordnungen sind ebenfalls wahlfrei. Zudem bieten die Komponenten selbst teilweise sehr weitgehende Möglichkeiten der Konfiguration. Die Möglichkeit zur Konfiguration hat gegenüber Programmierung die Vorteile, dass sie weniger techni-

⁷ Mit Signatur wird in der Programmierung die Schnittstelle einer Funktion bezeichnet: Sie enthält den Namen der Funktion sowie die Anzahl, den Typ und die Reihenfolge der Parameter. Zusätzlich können sprachspezifische Besonderheiten wie der Übergabemodus der Parameter enthalten sein.

sches Wissen in Form von Programmierfähigkeiten verlangt und damit den Kreis der potentiellen Anwender deutlich erhöht. Zum anderen trägt sie – im Vergleich zur Programmierung als Alternative – auch zur Stabilität der Plattform bei, da viele Möglichkeiten zur Fehlkonfiguration durch die Plattform und die Komponenten selbst schon abgefangen werden.

Programmierung durch den Fachexperten

Reichen die Möglichkeiten der Konfiguration nicht aus und sind spezifischere Lösungen nötig, so erlaubt die Plattform die gezielte Erweiterung durch eigenständige Programmierung einer Komponente. Diese Arbeit ist zwar technisch anspruchsvoller, jedoch durchführbar – wie die vielen realisierten Nicht-Hersteller-Komponenten zeigen.

Bibliotheken

Die Sammlung bereits entwickelter Komponenten in Bibliotheken ermöglicht die Wiederverwendung und erhöht das Einsatzpotential der Plattform. Zudem verteilt es die Entwicklungsaufwände und erlaubt teilweise die Refinanzierung der Entwicklungskosten.

Benutzergemeinschaft

Das Entstehen einer Benutzergemeinschaft der Plattform – wie es durch die vielen selbstentwickelten Komponenten sowie eine stark genutzte Mailingliste⁸ belegt wird – ermöglicht den fachspezifischen Austausch der Benutzer, es entsteht eine *Community of Practice*.

3 Lernszenarien am Beispiel von CASAnova

3.1 Das Lernprogramm CASAnova

CASAnova ist ein an der Universität Siegen entwickeltes Softwarepaket, das Lernszenarien aus dem Bereich Wärmeschutz darstellt. Der Benutzer kann ein vorgegebenes Modell eines Hauses konfigurieren und sieht sofort die Auswirkungen. In Abbildung 6 ist die Oberfläche des Programmes direkt nach dem Start zu sehen: Die Bildschirmmaske ist zweigeteilt. Auf der linken Seite können Eingaben gemacht werden, die sich direkt nach Eingabe auf der rechten Bildschirmhälfte auswirken: Die Berechnung des statischen Simulationsmodells wird sofort durchgeführt. Die Konfigurationsdaten, die in die Simulation einfließen, sind in die Gruppen *Geometrie*, *Fenster*, *Dämmung*, *Gebäude*, *Klima* und *Energie* unterteilt. Die Ausgabe (dargestellt sowohl als numerische Werte als auch in Grafiken) wird zusätzlich zu der in Abbildung 6 dargestellten Übersicht in den Kategorien *Klima/Gebäude*, *Energieflüsse*, *Heizen* und *Kühlen* dargestellt.

⁸ Die TRNSYS-Mailing-Liste (<https://mailman.cae.wisc.edu/listinfo/trnsys-users>, letzter Zugriff 11.09.2012) weist für 2011 über 2000 Einträge auf.

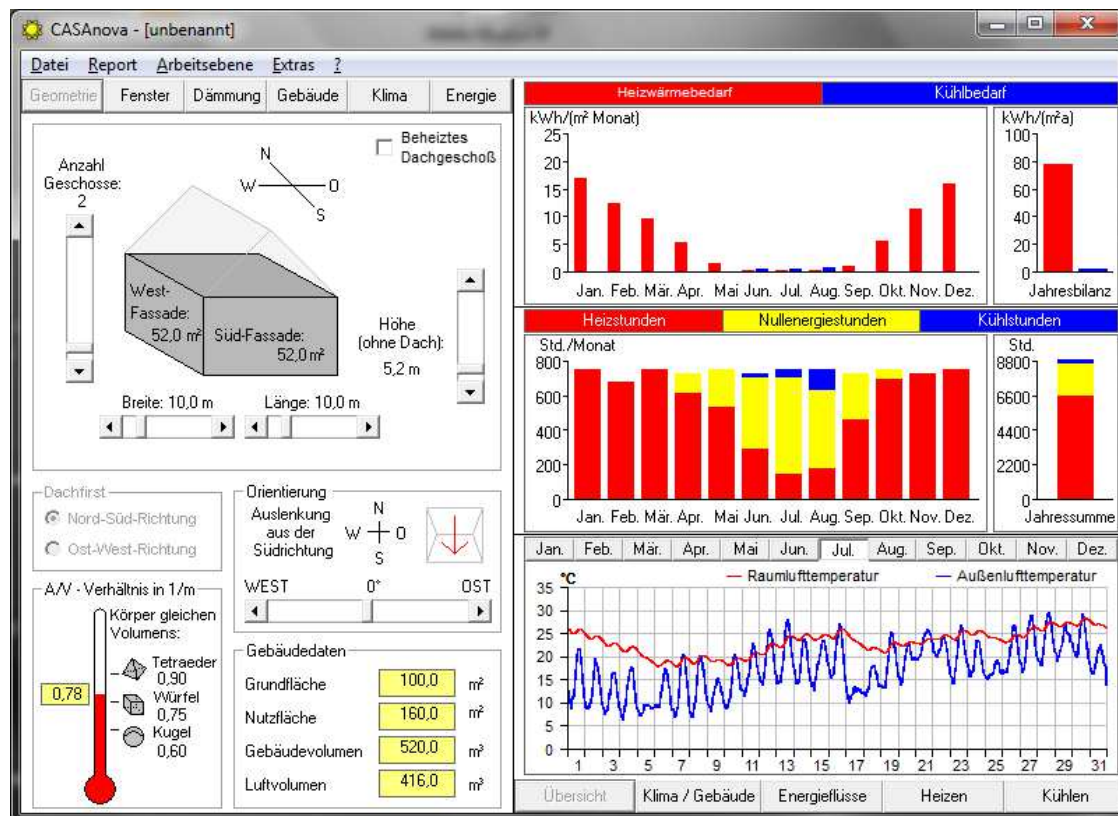


Abbildung 6: CASAnova: Start-Ansicht

Zu jedem Zeitpunkt der Simulation ist es möglich, jeden Eingabeparameter zu ändern und sofort die Auswirkungen zu sehen. Zwischen den einzelnen Gruppen von Informationen kann ohne Einschränkung umgeschaltet werden.

In der Hilfe zu dieser Software sind die Formeln des dargestellten Modells hinterlegt – wie in Abbildung 7 gezeigt. Der Benutzer kann sich jederzeit bei weitergehender Notwendigkeit darüber informieren, auf welche Weise bestimmte Werte errechnet werden.

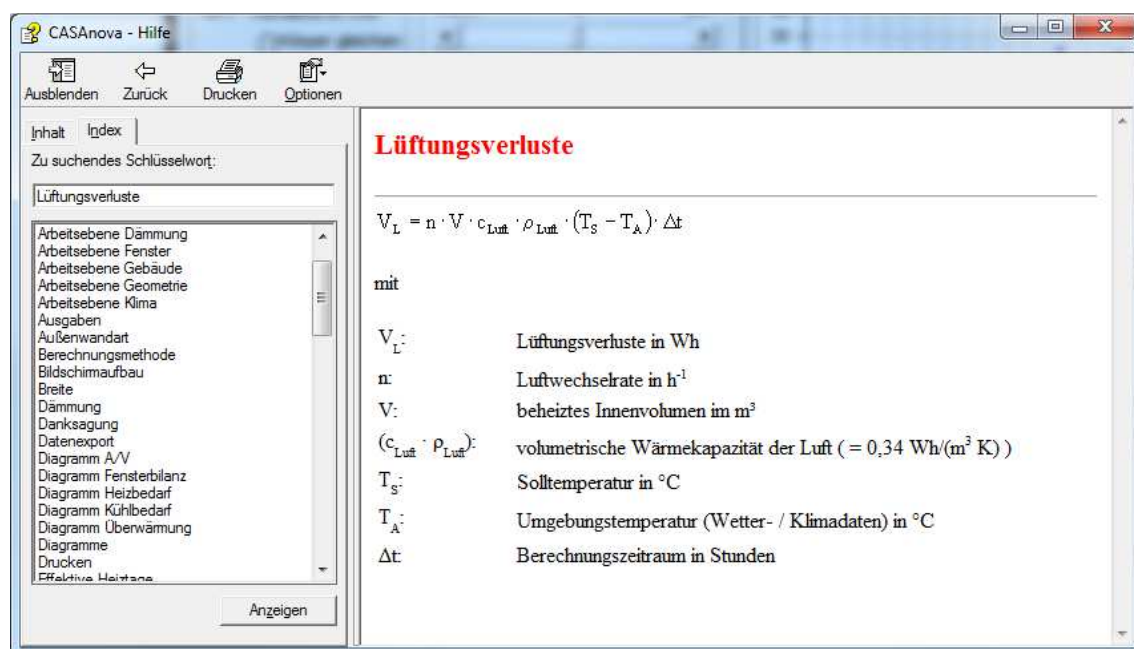


Abbildung 7: CASAnova: Ausschnitt Online-Hilfe mit Beschreibung der verwendeten Formel

3.2 Schlussfolgerungen

Im Gegensatz zu der zuvor vorgestellten Software TRNSYS bietet CASAnova keine Möglichkeit zur Erweiterung. Das zugrundeliegende Modell ist fest programmiert – der Benutzer muss die Software so nutzen, wie sie erstellt wurde. Die einzige Möglichkeit der benutzerspezifischen Anpassung der Standardkonfiguration ist das Abspeichern eines Satzes von Eingabedaten in einer Datei zur späteren Wiederverwendung.

Dennoch demonstriert diese Software durch ihre unmittelbare Rückmeldung in Form von Grafiken und Größen die Möglichkeiten der spielerischen, experimentellen Simulation. Mit Hilfe von CASAnova ist der Benutzer in der Lage, den Einfluss von verschiedenen Größen auf ein komplettes System auszutesten und damit letztendlich zu lernen. Die Einschränkung, dass mit CASAnova nur stationäre Prozesse simuliert werden können, wird ausgeglichen durch die Möglichkeit der schnellen Rückmeldung aufgrund von weniger aufwändigen Berechnungen. In dieser Hinsicht ist CASAnova ein herausragendes Beispiel und Vorbild für die geplante Spielplattform.

4 Abbildungsverzeichnis

Abbildung 1: Architektur TRNSYS.....	3
Abbildung 2: Bildschirmkopie des grafischen Modell Editors von TRNSYS („Simulation Studio“) ..	4
Abbildung 3: TRNSYS: Ausschnitt einer Simulationsmodelldatei	4
Abbildung 4: TRNSYS: Zuweisung von Aus- und Eingabevariablen durch den Verknüpfungseditor	5
Abbildung 5: TRNSYS: Gebäude-Komponente und zugehöriger Editor.....	6
Abbildung 6: CASAnova: Start-Ansicht.....	9
Abbildung 7: CASAnova: Ausschnitt Online-Hilfe mit Beschreibung der verwendeten Formel.....	9

Anhang E: Computerspiele als Lernmedium

Inhaltsverzeichnis

1 Einleitung.....	2
2 Grundlagen	2
2.1 Lernen & Computerspiele	2
2.2 Deep Learning.....	2
2.3 Situiertes Lernen und Communities of Practice.....	3
2.4 Taxonomie	4
2.5 Kommerzielle Spiele in der Lehre: Vermittelbare Fähigkeiten.....	5
2.6 Modifikation von kommerziellen Computerspielen.....	6
2.7 Interest Capturing.....	6
2.8 Selbstbestimmung	6
2.9 Laborumgebung	7
2.10 Artefakte und konstruktionistisches Lernen	7
3 Computerspiele und Lernen in der Praxis	8
3.1 Eine Schule baut auf Spiele: Quest to Learn.....	8
3.2 Computerspiel „Portal“ auf der Buchliste eines Seminars.....	8
3.3 StarCraft zur Schulung von „21st Century Skills“	9
3.4 Emotion und Kognition.....	9
3.5 Voreingenommenheit	10
3.6 Motorik und Wahrnehmungsfähigkeit.....	11
3.7 Spielen für die Wissenschaft	11
3.8 Replaying History: Civilization III im Klassenraum.....	12
3.9 Alternate Reality Games.....	12
4 Weitere aktuelle Fragestellungen	12
4.1 Reduzierung der Aufwände	12
4.2 Adaptabilität von Computerspielen	14
4.3 Zielgerichteter Einsatz in der Lehre.....	15
5 Abbildungsverzeichnis	17

1 Einleitung

Der Einsatz von Computerspielen als Hilfsmittel zum Lernen – sowohl in formellen als auch in informellen Kontexten – stellt ein faszinierendes Gebiet dar, das im Rahmen dieser Arbeit untersucht werden konnte. In diesem Anhang werden interessante Grundlagen und Erkenntnisse im Umfeld des spielbasierten Lernens dokumentiert, die nicht direkt in die Hauptarbeit eingeflossen sind.

2 Grundlagen

2.1 Lernen & Computerspiele

Zimbardo & Gerrig (2004, S. 243) definieren Lernen als „erfahrungsbasierten Prozess, der in einer relativ überdauernden Veränderung des Verhaltens [...] resultiert“. Trembl & Becker (2010: S. 107) verstehen unter Lernen „alle nicht direkt zu beobachtenden Vorgänge in einem Organismus, vor allem in seinem zentralen Nervensystem (Gehirn), die durch Erfahrung (aber nicht durch Reifung o.ä.) bedingt sind und eine relativ dauerhafte Veränderung bzw. Erweiterung des Verhaltensrepertoires zur Folge haben“. Beide Definitionen weisen darauf hin, dass Erfahrungen ein wichtiger Auslöser für dauerhafte Verhaltensänderungen – und damit für Lernen – sind.

Squire (2008) sieht Computerspiele als das Medium, das prägende Lernerfahrungen für die Spieler ermöglicht, er spricht von „designed experiences“ (Squire, 2006). Gee (2008) unterstützt die These und nennt als weitere Elemente einer guten Lernerfahrung Ziele, Deutungen, Praxis, Erklärungen, Auswertungen und Rückmeldungen. Jedes dieser Elemente kann durch Computerspiele geleistet oder angestoßen werden, wenn man die um Computerspiele entstehende Kultur bzw. sozialen Gruppen – er nennt sie *affiliation groups*, bei Lave & Wenger (1991) werden sie als *Communities of Practice (CoP)* bezeichnet – einbezieht. Als Resultat attribuiert er Spiele als Lernmaschinen (Gee, 2005).

2.2 Deep Learning

Lernen, das mit Hilfe von Computerspielen stattfindet, hat den Ruf des sogenannten *Deep Learning*. Marton & Säljö (1976) haben die Ansätze der gedanklichen Aufnahme eines zu lesenden Artikels beobachtet und in zwei Arten unterteilt. „Surface Processing“ zeichnet sich dadurch aus, dass der Artikel nur isoliert gesehen wird und der Leser versucht, die Fakten und Details zu behalten. Hingegen ist das Ziel beim „Deep Processing“, die Inhalte des Artikels mit schon bekanntem Wissen in Verbindung zu setzen und so dem Text eine Bedeutung zu geben und seine Aussagen einzuordnen. Mit vernetztem Wissen ist der Lerner in der Lage, höhere Verständnisniveaus zu erreichen. Bezogen auf eine Lernzieltaxonomie¹ bedeutet dies, dass Surface Processing die unteren Ebenen abdecken kann (*Wissen* und *Verstehen*), während Deep Processing notwendig ist, um die höheren Ebenen (3 bis 6) zu erreichen (McGee, 2003).

Deep Learning ist eher korreliert mit intrinsischen Motivationsansätzen – der Student möchte den Sachverhalt verstehen (*Meaning Approach*), während Surface Learning mehr extrinsisch motiviert zu sein scheint – der Student möchte die nächste Klausur bestehen (*Reproducing Approach*) (Atherton, 2011; Willis & Clift, 1983). Tagg (2003, S. 70) unterscheidet zwischen Deep und

¹ In der übergeordneten Arbeit wird die Lernzieltaxonomie von Bloom genutzt (Anderson & Krathwohl, 2001).

Surface Learning wie folgt: „... *the knowledge that students use in practical problem solving and thinking is the knowledge framed in the context of what they really believe and find meaningful. Deep learning is learning that takes root in our apparatus of understanding, in the embedded meanings that define us and that we use to define the world. Surface learning floats lightly on a matrix of repetition and reinforcement, abstracted from who we are and what we deeply know.*“

Gee (2009) postuliert, dass gerade Computerspiele als Medium Deep Learning fördern, weil sie u.a. den Spieler emotional binden. Er beschreibt eine Menge von Prinzipien von Computerspielen, die diese erfüllen, um Lernen fördern (Gee, 2003). Erfolgreiche Computerspiele müssen gemäß diesen Prinzipien aufgebaut sein, damit die Spieler eine Chance haben, die oft komplizierten, zugrundeliegenden Systeme zu erfassen. M. Ryan et al. (2012) verbinden diese Aussagen mit dem Grundsatz, dass einem faszinierendem Spiel ein Spielzeug zugrundeliegen muss („Lens of a toy“, (Schell, 2008)) und sie identifizieren 3 Kernthemen, die ein Spiel besetzen sollte, um Deep Learning zu fördern:

- **Abstraktion:** Einem Spiel liegt unter der Oberfläche ein abstraktes Modell zugrunde. Für den Spieler ist es eine Herausforderung, dieses Modell aus dem im Spiel konkret dargestellten System zu abstrahieren. Koster (2004) bemerkt hierzu, dass Spieler „Pattern Matcher“ sind. Ein Spiel ist so lange interessant, wie der Spieler das zugrundeliegende abstrakte System modellieren kann.
- **Steuerung („Control“):** Die Möglichkeit zur Steuerung lässt Spieler mit dem Werkzeug „Spiel“ verschmelzen. Dieses Phänomen ist auch als „Immersion“ bekannt, d.h. der Spieler geht in der Spielwelt auf.
- **Eigentümerschaft („Ownership“):** Spiele, die ein individuelles Lernerlebnis erlauben, erzeugen dadurch eine tiefe Verbindung des Spielers mit dem Gelernten, das er als sein Eigentum betrachten kann. Gee (2009) und Squire (2011) nennen dies auch „player-enacted trajectories“.

2.3 Situiertes Lernen und Communities of Practice

Die Lerntheorie des Situierten Lernen wurde von Lave & Wenger (1991) beschrieben. Annahme ist, dass Lernen einen authentischen Kontext benötigt: es wird dann gefördert, wenn eine tatsächliche Aufgabe gelöst werden soll. Das Lernen passiert eher zufällig als absichtlich. Wichtige Elemente des Kontextes, in dem Lernen stattfindet, sind dieser Theorie zufolge soziale Interaktion und Zusammenarbeit mit anderen, die in *Communities of Practice* (CoP) stattfinden. Wenger (2006) nennt drei Merkmale einer CoP:

- **Fachgebiet („domain“):** Eine CoP bezieht sich auf einen fachlichen Problembereich, der das Interesse aller Mitglieder der Gemeinschaft hat.
- **Gemeinschaft:** Die Community besteht aus mehreren Mitgliedern mit Interesse, Aufgabenstellungen aus demselben Problembereich („domain“) zu lösen.
- **Anwendung („practice“):** Die Problemstellungen entstehen nicht aus theoretischen Betrachtungen, sondern aus praktischen Notwendigkeiten. Es sollen konkrete Ziele erreicht werden.

Ein Beispiel ist eine Gruppe von Alkoholsüchtigen, die sich austauschen über Wege, ihrer Sucht zu entkommen (Lave & Wenger, 1991). Ein weiteres Beispiel ist der informale Austausch von Kopiergerätetechnikern untereinander, wie bestimmte Probleme mit den Geräten am besten zu

lösen sind (Brown & Duguid, 2000). In Abgrenzung zu anderen Erscheinungen ist für eine CoP wichtig, dass es ein Austausch über Lösungsmöglichkeiten konkreter Probleme besteht. Die Mitglieder der Gruppe sind selbst betroffen und sie tauschen sich aus – situiertes Lernen ist nach Lave & Wenger auch immer ein sozialer Prozess.

Computerspiele sind Treiber einer CoP: die Spieler haben konkrete Probleme eines bestimmten Fachgebietes (nämlich die zu lösenden Aufgaben des Spiels) und kommunizieren darüber (beispielsweise über Foren oder Online-Chat) – sie lernen dabei nach der Theorie des situierten Lernens. Sie machen Lernerfahrungen², die prägend sind und sich tief im Bewusstsein des Spielers verankern³.

2.4 Taxonomie

Es wurden verschiedene Taxonomien für Computerspiele entwickelt, eine ist die *Erfurt Taxonomie* (Jantke & Gaudl, 2010; Jantke, 2009, 2010): In ihr werden Computerspiele in einem 3-dimensionalen Raum angeordnet. Den Dimensionen D1, D2 und D3 werden die folgenden Bedeutungen zugewiesen:

- **D1:** Das Spiel wird als Computerprogramm gesehen. Alle Eigenschaften des Spiels als Computerprogramm werden untersucht, z.B. Anzahl der Benutzer und Netzwerkfähigkeit.
- **D2:** Das Genre des Spiels als Kunstgegenstand und Medium.
- **D3:** Ein Spiel ist höchst interaktiv. Daher hat es auch einen Einfluss auf die Psyche und das soziale Verhalten.

Als Arbeitsgrundlage für diese Arbeit soll zunächst eine andere Taxonomie verwendet werden, die nur für die Computerspiele gültig ist, die in der Lehre eingesetzt werden. Das oberste Kriterium dieser Taxonomie ist der Zweck, für den das Computerspiel geschaffen wurde. Es findet eine Unterteilung in Spielen statt, die kommerziell zu Unterhaltungszwecken erstellt wurde und Lernspielen, die allein zu diesem Zweck erstellt wurden (s. Abbildung 1).

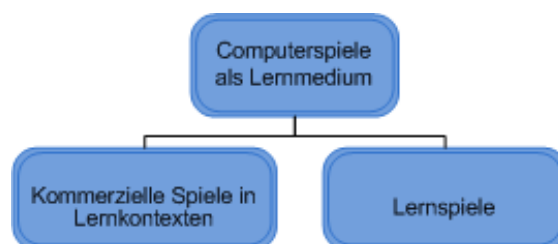


Abbildung 1: Taxonomie von Computerspielen als Lernmedium

Diese Unterscheidung wird getroffen im Bewusstsein, dass die Herstellung eines Lernspiels kein leichtes Unterfangen ist: Kirriemuir & McFarlane (2004, S. 4) berichten, dass viele Versuche, Lernspiele zu erstellen, fehlgeschlagen sind. Ähnlich argumentiert Egenfeldt-Nielsen (2007), wenn er feststellt, dass „*Edutainment started as a serious attempt to create computer games that taught children different subjects. Arguably, it ended up as a caricature of computer games and a reactionary use of learning theory.*“ Laurel (2001) nennt eine solche Kombination auch

² Squire (2011) nennt sie „aesthetically enlivening experiences“.

³ „Erkläre mir, und ich vergesse. Zeige mir, und ich erinnere. Lass es mich tun, und ich verstehe.“ Dieses Zitat wird Konfuzius zugeschrieben. Es verdeutlicht den Wert der selbstgemachten Erfahrung im Lernprozess den Menschen. Mit Computerspielen bzw. -simulationen existieren nun Medien, die an dieser Stelle unterstützen können.

“chocolate-covered broccoli”: Dieser Ausdruck ist inzwischen zu einer stehenden Redewendung geworden, mit der Lernspiele bezeichnet werden, in denen Lerninhalte und Spielmechaniken nicht stimmig ineinander integriert wirken und als Fremdkörper erscheinen.

Projekte zur Entwicklung von Computerspielen unterliegen dem Projektrisiko des Scheiterns. Generell ist dieses für IT⁴-Projekte – wie es ein Projekt zur Erstellung eines Computerspiels auch ist – erheblich. Eine Studie nennt eine Fehlschlag-Quote von ca. 50% – nur die Hälfte der gestarteten Projekte werden erfolgreich abgeschlossen (Ambler, 2010). Hinzu kommt bei Computerspielen die Herausforderung, ein spannendes Spiel zu entwickeln, d.h. die Lerninhalte in Spielmechaniken einzubetten, so dass ein Computerspiel entsteht, das die Spieler anzieht und nicht eines, für das die Spieler verpflichtet werden müssen, da sie während des Spiels keinen Spaß empfinden und keine intrinsische Motivation entsteht. Diese Anforderung erhöht das Projektrisiko bzw. die Projektaufwände nochmals erheblich. Daher ist es eine vertretbare Alternative im Feld des spielbasierten Lernens, auf kommerzielle Spiele⁵, die zu Unterhaltungszwecken entwickelt wurden, zurückzugreifen.

2.5 Kommerzielle Spiele in der Lehre: Vermittelbare Fähigkeiten

Teilweise können kommerzielle Spiele Fachwissen vermitteln, wie es beispielsweise beim Microsoft Flugsimulator der Fall ist, der eingesetzt werden kann, um Wissen über Flughäfen und Flugzeugtypen weiterzugeben (FlightSimulatorX, 2010). Zusätzlich können mit Spielen fachbereichsübergreifende Fähigkeiten gefördert werden. Diese werden auch als Metaskills⁶ bzw. 21st Century Skills bezeichnet. 21st Century Skills werden die Fähigkeiten genannt, die als bedeutend gesehen werden, um unter den Rahmenbedingungen des 21. Jahrhunderts effektiv Aufgaben durchführen zu können. Das 21. Jahrhundert ist gekennzeichnet durch Globalisierung und einer hohen Änderungsgeschwindigkeit von Wissen und Arbeitsumwelt. Eine Taxonomie von 21st Century Skills stammt von Binkley et al. (2012). Sie umfasst 10 Fähigkeiten, die in 4 Gruppen angeordnet werden. Sie wird im Folgenden aufgeführt, um einen Eindruck der unter diesem Schlagwort subsummierten Fähigkeiten zu geben. Derartige Fähigkeiten gelten als förderbar durch Computerspiele.

1. Denken
 - Kreativität und Innovation
 - Kritisches Denken, Problemlösungsverhalten, Entscheidungsverhalten
 - Lernen zu lernen, Metakognition
2. Arbeiten
 - Kommunikation
 - Zusammenarbeit
3. Werkzeuge für die Arbeit
 - Informationsnutzung
 - Nutzung von Informations- und Kommunikationstechnologie
4. Reales Leben („Living in the World“)

⁴ IT: Information Technology

⁵ Diese werden auch als *Commercial off-the-shelf* (COTS)-Spiele bezeichnet.

⁶ Mayer (1998) definiert den Begriff als *„Metaskills (or metacognitive knowledge) involves knowledge of when to use, how to coordinate, and how to monitor various skills in problem solving“*.

- Mündiger Bürger
- Leben und Beruf („Live and career“)
- Persönliche und soziale Verantwortung (mit kulturellem Bewusstsein und Kompetenz)

Die von Mayer als Metaskills bezeichneten Fähigkeiten sind in der obigen Taxonomie im Bereich „Denken“ einzuordnen. In verschiedenen Studien wurde nachgewiesen, dass derartige Fähigkeiten von Spielern diverser Computerspiele gefordert werden und damit auch das Erlernen dieser Fähigkeiten gefördert wird (King, 2011; Steinkuehler & Duncan, 2008). Ein Beispiel für den Einsatz eines kommerziellen Spiels mit dieser Absicht wird in Kap. 3.3 *StarCraft* zur Schulung von „21st Century Skills“ beschrieben.

2.6 Modifikation von kommerziellen Computerspielen

Kommerzielle Computerspiele können auf verschiedene Weisen angepasst werden, um einen Einsatz in der Lehre effektiver zu gestalten. Scacchi (2010) unterscheidet zwischen „User interface customizations“ und „Game conversion mods“⁷. Erstere Gruppe steht eher für Änderungen an den Spielen, die durch Konfigurationen entstehen. Die Möglichkeit der Konfiguration ist durch den Hersteller des Spiels vorgesehen und wird durch ihn gegebenenfalls auch unterstützt durch die Bereitstellung von Editoren, so enthält das Spiel *Civilization* in der Standardversion bereits einen Editor, mit dem neue Spielwelten erzeugt werden können („map editor“). Auch können benutzerdefinierte Szenarios erstellt werden. Mit diesen konnte Squire (2003) *Civilization* als Medium für den Geschichtsunterricht nutzen (s. Kap. 3.8).

„Game conversion mods“ sind solche Modifikationen, die der Hersteller nicht direkt vorgesehen hat. Sie werden zwar teilweise durch die Bereitstellung von API-Informationen und Scripting-Schnittstellen unterstützt. Der sogenannte „Modder“, also die Person, die die „Mods“ (Abkürzung für „modification“) durchführt, braucht Wissen über den technischen Aufbau der Spielsoftware und/oder Programmierkenntnisse. Für das Open-World-Spiel *Minecraft* existieren zahlreiche Mods, zum Beispiel erstellte Großmann (2012) einen Mod zur Darstellung bauphysikalischer Szenarien in *Minecraft*.

2.7 Interest Capturing

Ein Effekt von Computerspielen ist es, dass sie Interessen in den Spielern erwecken können: Spieler entwickeln Neugier zu bestimmten Themen und suchen selbständig nach weiteren Informationen. Squire (2011) nennt das *Interest Capturing*. Eindrucksvoll ist die Erzählung von einem seiner Schüler, der sich durch das Spielen von *Civilization*, einem Strategie-Computerspiel, inspiriert fühlte, das Buch „Über die Kriegskunst“ von Sun Tzu, einem chinesischen General und Philosophen zu lesen. Dieser Effekt, der auch von M. Wagner & Mitgutsch (2008) beschrieben wird, ist hoch zu bewerten in Bezug auf Lernen, da er intrinsische Motivation auslöst.

2.8 Selbstbestimmung

Wesentliches Merkmal des Lernprozesses in Spielen ist die Selbstbestimmung des Spielers: Er selbst kann über die Inhalte entscheiden, mit denen er sich beschäftigt und er kann auch das

⁷ Exakt genommen nennt er auch noch „Machinima and Art Mods“ und „Custom gaming PCs“, aber diese Kategorien sollen hier nicht betrachtet werden, da sie als nicht zielführend für die Ausbildung in Ingenieursdisziplinen angesehen werden (*Machinima* sind mit Hilfe von Game Engines produzierte Filme).

Tempo bestimmen, in dem er sich mit den Inhalten beschäftigt. Auch hier ist die Motivation wieder intrinsisch. Squire (2011) spricht auch vom *Interest-Driven Learning und Learning Trajectories* – jeder Spieler hat die Freiheit zu entscheiden, was ihn interessiert und welche Probleme des Spiels er in welcher Zeit lösen möchte. Die Selbstbestimmung findet allerdings Grenzen, wenn Spiele in formalen Lernkontexten eingesetzt werden: Die Pflicht der Einhaltung von Lehrplänen führt zu zeitlichen und inhaltlichen Beschränkungen.

2.9 Laborumgebung

Computerspiele bieten die Möglichkeit das Verhalten der Spieler aufzuzeichnen: Alle Aktionen des Spielers müssen durch ein Eingabegerät dem Spiel mitgeteilt werden und können damit in eine digitale Form umgewandelt werden.

Damit ist es nach Squire (2011) möglich, das menschliche Lernverhalten zu studieren ohne eine künstliche Laborumgebung zu schaffen. Es kann getestet werden, welche pädagogischen Konstruktionen erfolgreich funktionieren – und welche nicht funktionieren.

Auch unter anderen Aspekten kann die Laborumgebung des Spiels genutzt werden: Castronova (2006) nutzte die virtuelle Welten von MMOGs, um das ökonomische Verhalten von Spielern zu untersuchen – er konnte keine Hinweise darauf finden, dass sie im virtuellen Leben andere ökonomische Maßstäbe anlegen als im richtigen Leben.

Über das SNG „Fliplife“ wird berichtet, dass Spieler aufgrund des Verhaltens im Spiel Einladungen zu Vorstellungsgesprächen erhalten (Meyer, 2011), es kann zur Mitarbeiterrekrutierung eingesetzt werden (Söbke, Hadlich et al., 2012) – weil sich das Spielerverhalten digital aufzeichnen und größtenteils automatisiert auswerten lässt.

A/B-Testing ist ebenfalls eine Maßnahme, die insbesondere in SNGs einfach durchgeführt werden kann⁸: Zwei Varianten einer zu testenden Spielmechanik werden parallel an zwei Gruppen von Spielern ausgeliefert, mit Hilfe der Aufzeichnungen wird die gemäß vorher definierter Kriterien erfolgreichere Variante gefunden. Insbesondere Zynga nutzt diese Möglichkeit, Spiele als Laborumgebung zu sehen (Andersen et al., 2011; G, 2012).

2.10 Artefakte und konstruktionistisches Lernen

Zwei Grundlagen der konstruktionistischen Lerntheorie sind erstens, dass Wissen nicht transferiert werden kann in den Kopf des Lernenden, sondern dort neu aufgebaut werden muss und zweitens, dass der Aufbau des Wissen dann sehr gut gelingt, wenn der Lernende ein für ihn sinnvolles Produkt („Artefakt“) erzeugt (Papert, 1980). In diesem Sinne sind Spielplattformen zu sehen, die den Rahmen für gestalterische Prozesse des Spielers vorgeben: Der Spieler erhält ein Werkzeug, mit dem er eigene Gedanken in ein Produkt überführen kann – und dabei beste Bedingungen zum Lernen hat. Nicht alle Computerspiele erlauben eigenes Gestalten, aber es gibt sehr erfolgreiche Beispiele⁹:

⁸ SNGs bieten ideale Voraussetzungen, da bei jedem Aufruf die ausführbaren Dateien von Server neu geladen werden – der Spieler muss lokal keine Software installieren. Auf dem Server kann konfiguriert werden, welchem Spieler welche Variante des Spiels ausgeliefert wird.

⁹ Weitere Beispiele von Plattformen, die es erlauben, eigene Inhalte zu erzeugen, werden in Anhang A vorgestellt.

- Seymour Papert selbst initiierte *Logo*, eine Programmiersprache für den Einsatz in der Lehre (LogoFoundation, 2010).
- *Spore* ist ein Simulationsspiel des Herstellers Maxis– entworfen von Will Wright – das dem Spieler erlaubt, Lebewesen nach eigenen Vorstellungen zu entwickeln.
- *LittleBigPlanet* ist ein Puzzle-Spiel, bei der man die Spielfigur über einen aus Podesten bestehenden Parcours bewegen muss, um dabei einen möglichen Weg zu entdecken. Jeder Spieler kann selbst Puzzles entwerfen und diese anderen Spielern zur Verfügung stellen.

3 Computerspiele und Lernen in der Praxis

In diesem Kapitel soll ein Eindruck vermittelt werden, wie Computerspiele heute schon zu Lernzwecken eingesetzt werden. Dazu werden Beispiele aus verschiedenen Bereichen und Facetten herangezogen, um die große Bandbreite der möglichen und tatsächlichen Nutzung zu zeigen.

3.1 Eine Schule baut auf Spiele: Quest to Learn

Quest to Learn (Q2L) ist der Name einer New Yorker Schule für die Jahrgänge 6 bis 12, die im Schuljahr 2009/10 gestartet wurde. Zentrales Element des Lehrplans ist Systemdenken („systems thinking“). Wesentliche pädagogische Mittel sind Spiele. Sie werden in drei verschiedenen Ausprägungen eingesetzt (Corbett, 2010; Quest to Learn, 2010):

- **Unterrichtsorganisation als Spiel**
Der Unterricht wird entsprechend eines Spiels strukturiert: Die Schüler erhalten Forschungsaufgaben, bei denen sie lernen können, in Form von Missionen – das erfolgreiche Absolvieren wird entsprechend belohnt.
- **Erstellung von Spielen**
Die Erstellung von Spielen fordert die Schüler, sich mit den fachlichen Grundlagen der Spiele auseinanderzusetzen, Systeme der Realität in die den Spielen zugrundeliegenden Systeme zu übertragen und den Umgang mit neuen Medien zu lernen.
- **Spiele als Inspiration**
Das Spielen geeigneter Computerspiele dient als Inspiration. Gleichzeitig haben die Spieler die Chance, sich mit den den Spielen zugrunde liegenden Systemen der realen Welt auseinanderzusetzen.

In der Zwischenzeit wurde eine weitere Schule basierend auf diesem Konzept in Chicago eröffnet (CQ, 2012; Jackson, 2011). In Los Angeles fußt die *PlayMaker*-Schule auf einem ähnlichen Ansatz (GameDesk, 2012; Heron, 2012).

3.2 Computerspiel „Portal“ auf der Buchliste eines Seminars

Das Computerspiel *Portal*¹⁰ – inzwischen liegt eine Folgeversion *Portal 2*¹¹ vor - ist eine Plattform für Puzzles, bei denen der Spieler in jedem einzelnen Puzzle einer Art Labyrinth unter Berücksichtigung von Hilfsmitteln, der Schwerkraft und der Fähigkeit zur Teleportierung entkommen muss.

¹⁰ orange.half-life2.com/portal.html, letzter Zugriff am 09.10.2011

¹¹ www.thinkwithportals.com/, letzter Zugriff am 05.06.2012

Portal steht auf der Buchliste des 2010 eingeführten Seminars „Enduring Questions“, das philosophische Fragen behandelt und obligatorisch für alle Erstsemester des Wabash College (Indiana, USA) ist (Abbott, 2010).

3.3 StarCraft zur Schulung von „21st Century Skills“

*StarCraft*¹² bzw. die Nachfolgeversion *StarCraft II*¹³ ist ein Echtzeit-Strategiespiel: Spieler bzw. Spielergruppen bauen Ressourcen im Spiel auf und versuchen mit deren Hilfe, den Spielgegner zu vernichten. Es war Gegenstand des Kurses „21st Century Skills in StarCraft“ an der *University of Florida* (USA). Ziel des Kurses war es, das Geschehen in Spielen der Kursteilnehmer zu beobachten und zu analysieren. Damit sollen Fähigkeiten trainiert werden, die als Schlüsselqualifikation für das 21. Jahrhundert gelten. Hierzu werden u.a. kritisches Denken, Entscheidungsfähigkeit, Ressourcenmanagement und Problemlösungsverhalten gezählt (s. Kap. 2.5, S. 5). Die Studenten sollen sich der Fähigkeiten bewusst werden, die sie benötigen, um erfolgreich StarCraft zu spielen. Dadurch wird die Übertragung ähnlicher Problemlösungsansätze in das spätere Berufsleben erleichtert (Mims, 2010; Poling, 2010). Aktuell erfolgt die Auswertung der Ergebnisse in Rahmen der Dissertation von Nathaniel Poling¹⁴.

3.4 Emotion und Kognition

Börsenhändler erzielen bessere Handelsergebnisse, wenn sie ihre Emotionen einschätzen können und die Mechanismen von kognitiven Verzerrungen kennen. Das *Aiming Game* (Cederholm, Hilborn, Lindley & Sennersten, 2011) und das *Auction Game* (Jercic, Astor, Adam & Hilborn, 2012) sind zwei Ansätze, den Spieler zu trainieren, sich seiner Erregung bewusst zu werden und diese aus dem Entscheidungsprozess herauszuhalten bzw. sie zu unterdrücken oder durch Neubewertung der Situation zu beseitigen. Bei beiden Spielen gibt es eine grundlegende Spielmechanik, die vom Spieler schnelle Entscheidungen verlangt und damit Druck ausübt. Das *Aiming Game* ist ein *Shooter-Spiel*¹⁵, in dem Vögel erlegt werden sollen, beim *Auction Game* muss ein virtueller Marktpreis schnell ermittelt werden und entweder ein Verkauf oder Kauf getätigt werden. Zur Hardware für die Spiele gehören Sensoren, die Herz- und Hirntätigkeit messen können. Daraus wird der emotionale Zustand des Spielers abgeleitet. Melden die Sensoren steigende Erregung zurück, so hat dies Einfluss auf das Spiel: Zum einen wird der Erregungsgrad auf dem Bildschirm angezeigt, zum anderen werden die Aufgaben schwieriger, wenn der Spieler erregt ist, weil er sich beispielsweise über vorangegangene fehlerhafte Aktionen ärgert. Die Schwere der Aufgaben wird zum Beispiel dadurch erhöht, dass die Vögel undeutlicher abgebildet werden oder die Beispielpreise, aus denen der Marktpreis ermittelt werden soll, weiter auseinanderliegen und damit der Durchschnittspreis schwerer im Kopf zu errechnen ist. Beide Studien ziehen zunächst ein positives Fazit: Es ist möglich, mit derartigen Spielen das Entscheidungsverhalten der Spieler zu verbessern.

¹² eu.blizzard.com/de-de/games/sc/, letzter Zugriff am 06.06.2010

¹³ eu.battle.net/sc2/de/, letzter Zugriff am 07.12.2010

¹⁴ Persönliche E-Mail-Kommunikation, Nov. 2012

¹⁵ Shooter Spiele gehören zur Kategorie der Action Spiele, bei der gewöhnlich mit einer Waffe Gegner oder andere Ziele bekämpft werden, das erfolgreiche Spiel fordert in hohem Maße Reaktionsgeschwindigkeit und Geschwindigkeit vom Spieler.

3.5 Voreingenommenheit

Pathfinder¹⁶ ist ein Computerspiel über Voreingenommenheit und Diskrimination im Berufsleben, hier am Beispiel der akademischen Arbeitswelt gezeigt. Gespielt wird die Karriere eines Studenten, der aufgrund seiner Herkunft verschiedenen unterschwelligen Vorurteilen seiner Person gegenüber ausgesetzt ist. Grundlegende Spielmechanik ist, typische akademische Aufgaben (Schreiben von Forschungsanträgen, Durchführen der Forschung und Erstellen der Berichte) zu erfüllen (s. Abbildung 2). Begleitend verbessert der Spieler seine Vernetzung in der akademischen Welt und begegnet dabei immer wieder verschiedenen Arten von Voreingenommenheit. Diese werden in begleitenden Texten erklärt – der Spieler wird auf die Mechanismen aufmerksam gemacht und kann sich in Zukunft im realen Leben anders verhalten. Die numerische Darstellung von abstrakten Begriffen wie Freundlichkeit, Respekt und des Wissensstandes um unterbewusste Voreingenommenheit („Bias Literacy“) hilft dem Spieler bei der Erfassung des Themas (s. Abbildung 3). In diesem Spiel werden Erfahrungen gemacht und mit der Hoffnung auf eine dauerhafte Verhaltensveränderung verknüpft – das Spiel dient als Medium zum Erlernen von abstrakten Systemen.

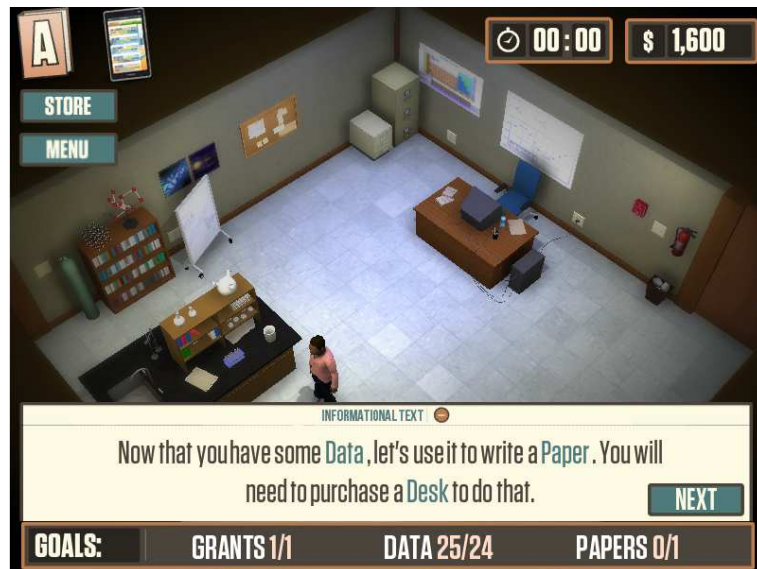


Abbildung 2: Pathfinder: Arbeitsumgebung

¹⁶ www.eriainteractive.com/project_Pathfinder.php, letzter Zugriff am 19.09.2012

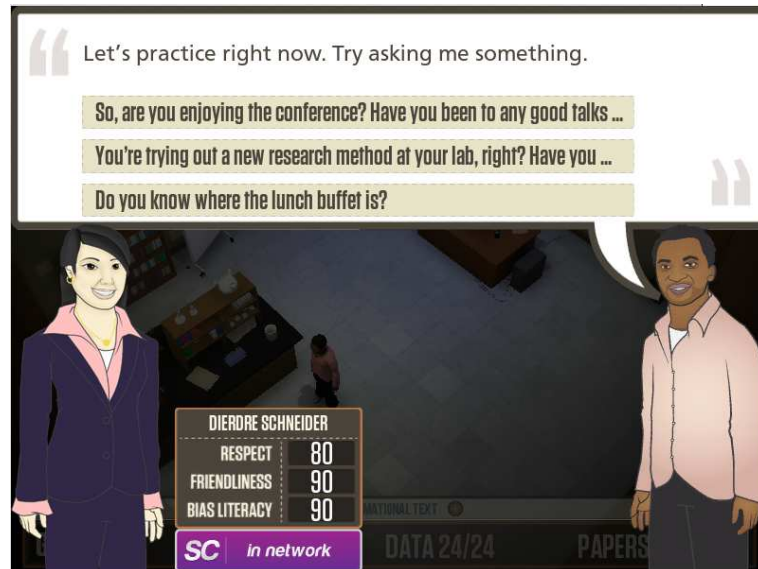


Abbildung 3: Pathfinder: Dialoge und Feedback

3.6 Motorik und Wahrnehmungsfähigkeit

Computerspiele sind interaktiv, d.h. die Spieler reagieren auf die Umgebung, die das Computerspiel ihnen bietet. Dabei finden Anpassungseffekte statt: die körperlichen Fähigkeiten verändern sich aufgrund der Reize, denen der Spieler durch fortgesetztes Spielen ausgesetzt werden. Computerspiele dienen als Trainingsgeräte für Funktionen des Körpers.

Li, Polat, Makous & Bavelier (2009) wiesen nach, dass die Kontrastempfindlichkeit mit Hilfe von Actionspielen trainierbar ist. Durch Green & Bavelier (2003) wurde festgestellt, dass verschiedene visuelle Fähigkeiten durch den Einsatz von Computerspielen verbessert werden können. Dazu zählen räumliches Sehen, eine Vergrößerung des Sichtfeldes als auch die Verbesserung der Wahrnehmung von zeitlich nur eingeschränkt sichtbaren Objekten. Achtman, Green & Bavelier (2008) merken an, dass verschiedene Fähigkeiten trainierbar sind durch Computerspiele, es ist aber noch nicht verstanden, welche Prozesse genau ablaufen, d.h. man weiß nicht, durch welche Spielaktionen welche Fähigkeiten trainiert werden. So ist es zurzeit noch nicht möglich, ein selektives und zielgerichtetes Training durchzuführen. Murphy & Spencer (2009) konnten die Ergebnisse der Studie von Green & Bavelier jedoch nicht replizieren. Ebenso konnten Irons, Remington & McLean (2011) keinen Effekt von Computerspielen auf die Aufnahmefähigkeit nachweisen. Es lässt sich daher nicht eindeutig schlussfolgern, dass die Wahrnehmungsfähigkeit mit Hilfe von Spielen trainierbar ist.

Rosser et al. (2007) fanden heraus, dass Chirurgen bessere Ergebnisse bei endoskopischen Operationen erzielen konnten, wenn sie mit Hilfe von ausgewählten handelsüblichen Unterhaltungs-Computerspielen (COTS Games) trainiert wurden. Die Autoren schließen, dass die Fähigkeiten zur Bedienung von Computerspielen mit einigen für die endoskopische Chirurgie notwendigen Fähigkeiten korrelieren. Sie sehen die Ergebnisse als Hinweis, dass Computerspiele ein geeignetes Trainingsmittel für Chirurgen sein könnten.

3.7 Spielen für die Wissenschaft

„3-D-Spiel: Gamer klären Struktur eines Virus-Enzyms auf“ ist die Überschrift eines Artikels (Hda, 2011), in dem berichtet wird, wie mit Hilfe eines Spiels und der Mithilfe von mehr als einhundert

Spielern die räumliche Struktur eines Eiweißes ermittelt werden konnte. Die Kenntnis der räumlichen Struktur eines Enzyms, das für die Reproduktion des AIDS-Virus von Bedeutung ist, hilft bei der Entwicklung von Medikamenten, die den Reproduktionszyklus unterbrechen können (Khatib et al., 2011).

Das Werkzeug, das für diese Leistung genutzt wurde, ist das Programm FoldIt (FoldIt, 2010). Es wurde bereichsübergreifend an der *University of Washington* vom *Center for Game Science* und dem *Department of Biochemistry* entwickelt – mit dem Ziel, menschliche Intuition einzusetzen für eine Aufgabe, die nur mit großem Aufwand durch Rechner erledigt werden kann: Das Finden der räumlichen Struktur von Molekülen. Die Motivation der freiwillig mithelfenden Personen – FoldIt benutzt die Mechanismen von Crowdsourcing – wird durch eine Einbettung der nutzbringenden Aktionen in Spielmechaniken unterstützt.

3.8 Replaying History: Civilization III im Klassenraum

Im Rahmen seiner Dissertation untersuchte Kurt Squire (2003), wie sich das Strategiespiel Civilization III im Rahmen des Geschichtsunterrichts als Lernmedium nutzen lässt. Er ließ modifizierte Szenarien spielen und testete Methoden der pädagogischen Begleitung aus – so fand er heraus, dass nachfrageorientierte Mini-Lektionen den Erfolg erhöhen: treffen die Spieler auf ein Problem, so kann dieses zeitpunktgerecht erörtert werden. Einen Teil der Gruppe konnte er nicht erreichen, dafür entwickelte der andere Teil der Gruppe umso mehr Interesse an den Unterrichtsinhalten und vertiefte eigenständig das Wissen („interest driven learning“).

3.9 Alternate Reality Games

Alternate Reality Games (ARG) sind ein Spielgenre, das eine Verbindung zwischen Realität und Fiktion herstellt: Eine fiktive Geschichte wird mit realen Hilfsmitteln und verschiedenen Medien in der Realität entfaltet, sie vermitteln den Mitspielern Lernerfahrungen (Kim, Lee, Thomas & Dombrowski, 2009). ARGs können zu Marketingzwecken eingesetzt werden, bekannt wurden sie aber auch durch die Verbindung mit gesellschaftlichen Zielen. Ein bekanntes Beispiel ist das ARG „world without oil“¹⁷, das u.a. von der Spieldesignerin Jane McGonigal durchgeführt wurde: Als fiktives Szenario diente eine drohende Verknappung des Erdöls. Die Spieler waren für den Zeitraum des Spiels von ungefähr einem Monat aufgerufen, Ideen zu entwickeln, wie sie ihr Leben auf die Verknappung des Erdöls einstellen können, diese Ideen auch teilweise durchzuführen und darüber medial zu berichten. Dadurch wurde das Spiel zu einem großen Crowdsourcing von Ideen, die auch dokumentiert und damit verbreitet wurden und somit einer weiteren Bewertung offenstehen.

4 Weitere aktuelle Fragestellungen

4.1 Reduzierung der Aufwände

Die Herstellung von Computerspielen ist relativ aufwändig. Für populäre, weltweit vermarktete Spiele werden Produktionsbudgets von 100 Mill. US Dollar angesetzt (W3stfa11, 2012). Bartle sieht die Finanzierung eines MMOGs als schwierigste Aufgabe im Produktionsprozess an (Hall & Novak, 2008, Seite 74). Auf der anderen Seite stellen Bröker & Kornadt (2009) sowie Bröker,

¹⁷ www.worldwithoutoil.org, zuletzt abgerufen am 9.8.2012

Söbke & Kornadt (2011) fest, dass das Problem der Finanzierung bei kleineren Zielgruppen, wie sie bei Serious Games vorherrschen, ebenfalls vorhanden ist und sogar noch größer wird.

Es werden verschiedene Lösungsmöglichkeiten für dieses Dilemma vorgeschlagen (Söbke, Hadlich, Bröker & Kornadt, 2011). Ein Ansatz ist die Verbesserung der Technik im Sinne einer Erhöhung der Produktivität. So sehen Kauhanen, Tran & Biddle (2007) die Bereitstellung von Werkzeugen („authoring tools“) zur Erstellung von Serious Games als zentrale Herausforderung an, um die Effizienz der Entwickler zu erhöhen. Die Software „*RealmCrafter*“¹⁸ ist ein Beispiel für einen MMOG-Editor: sie unterstützt die Herstellung von MMOGs u.a. auch durch Bereitstellung von grafischen Bibliotheken. Darüber hinaus gibt es eine Vielzahl von Entwicklungsumgebungen für Spiele, die versuchen, dem Entwickler eine möglichst hohe Produktivität zu verleihen. GameMaker (YOYOGames, 2010) ist eines der prominenten Beispiele.

Ein weiterer Ansatz ist die Beauftragung und Einbindung von Fachexperten mit der Entwicklung von Spielszenarien. Dazu ist es notwendig, Frameworks, Plattformen und Werkzeuge bereitzustellen, die die Fachexperten in die Lage versetzen, Spielszenarien zu erzeugen¹⁹. Notwendig ist dazu beispielsweise ein komfortabler Editor zur Erzeugung von neuen Objekten und Szenarios. Wenn der Fachexperte zur Erstellung der Inhalte angeregt werden soll, so benötigt er einen einfach zu bedienenden Editor, in dem er sich nicht mit Programmierung auseinandersetzen muss. Der (grafische) Editor setzt die Benutzereingaben derart um, dass die Schnittstellen der Spielplattform bedient werden. Vorbild für einen solchen Editor kann Tabellenkalkulationssoftware wie MS Excel sein. Mit Hilfe solcher Software ist der Endanwender heute in der Lage, Kalkulationsprobleme zu lösen, die in früheren Zeiten die Kapazität mehrerer Entwickler benötigten (Campbell-Kelly, 2007)²⁰.

Ein Vorschlag von Prensky (2004) ist die Bündelung der Aktivitäten: Für jede Fachdisziplin gibt es eine zuständige Stelle, die die Aktivitäten im Bereich der Lern-Software (zu der auch Computerspiele gehören) nach dem Prinzip von Open Source Software koordinieren und integrieren kann. Dadurch gibt es zu jeder Zeit eine gute Übersicht über die vorhandene Lernsoftware eines Themengebietes. Damit ist es möglich, die erfolgreichen Ansätze weiter auszubauen und ihnen eine größere Verbreitung zu verschaffen. Das Ergebnis ist ein Lehrsystem auf dem neuesten Stand, das ohne Kosten für alle zur Benutzung offen ist und durch eine Gemeinschaft nach dem Beispiel von Linux gepflegt wird. Ein derartiges Vorgehen erfordert jedoch den einmaligen Aufwand einer gemeinsamen Basissoftware.

Eine Synthese beider Vorschläge für den Bereich des spielbasierten Lernens ist im Rahmen dieser Arbeit entstanden und wird von Bröker et al. (2011) sowie Söbke, Hadlich et al. (2011) beschrieben: Es wird eine Softwareplattform vorgeschlagen, die von Fachexperten um Szenarien erweitert werden kann.

¹⁸ www.realmcrafter.com, zuletzt abgerufen am 9.8.2012

¹⁹ Das ist der Ansatz, zu dem diese Arbeit beitragen soll.

²⁰ Weiteres Beispiel für die Möglichkeit des Fachexperten, DV-Systeme unmittelbar zu nutzen, sind relationale Datenbanken, die mit Hilfe von SQL (Structured Query Language) abgefragt werden können. In den Anfängen von Datenbanken waren von DV-Experten erstellte Programme notwendig, um Änderungen an den Daten durchführen zu können. Unter Nutzung von SQL ist es einem (versierten) Fachexperten möglich, flexibel Änderungen selbst durchzuführen.

Morgan (2009) sieht Optimierungsmöglichkeiten aus softwaretechnischer Sicht, so fehlen bislang Standards u.a. bei der Modellierung und Beschreibung des Spielgeschehens und der benutzten Software (z.B. der Middleware). Ebenfalls verweist er auf *SecondLife*²¹ als ein Beispiel für eine Plattform, die vom Benutzer erstellte Inhalte aufnimmt. Produktivitätssteigerungen lassen sich auch durch die Optimierung der im Produktionsprozess verwendeten Methodik erzielen. Dazu zählt der Einsatz von Mustern – hier sind in der letzten Zeit einige Kataloge (z. B. Björk & Holopainen, 2004; Kirk, 2005) entstanden. Auch beschreibt Dormans (2012) in *Machinations* ein Werkzeug, mit dessen Hilfe Spielmechaniken modelliert werden können.

4.2 Adaptabilität von Computerspielen

Ein bedeutsamer Unterschied des Lernmediums „Computerspiel“ gegenüber anderen in der Lehre eingesetzten Medien ist die Notwendigkeit der intrinsischen Motivation. Computerspiele erwecken intrinsische Motivation im Spieler, sich mit den Inhalten des Spiels tiefer zu beschäftigen, der Spieler lernt letztendlich. Das ist ein Alleinstellungsmerkmal gegenüber anderen Medien und Methoden. Voraussetzung ist allerdings, dass der Spieler sich tatsächlich für das Spiel interessiert. Das ist personenabhängig – ähnlich wie Personen Vorlieben für Film- oder Literaturgenres haben. Für Computerspiele besteht daher eine – gegenüber anderen Lernmedien – erhöhte Notwendigkeit, die Bedürfnisse der Spieler zu erfüllen. Computerspiele sollten sich den Bedürfnissen des Lerners anpassen, sie sollten adaptiv sein.

Ein Lösungsansatz beginnt damit, die Vorlieben der Spieler zu definieren. Dazu werden die Spieler zunächst typisiert, um eine übersichtliche Anzahl von Gruppen zu schaffen, für deren Bedürfnisse dann Spiele entworfen werden können: es werden Spielertaxonomien aufgestellt. Bartle (1996) hat eine Vierteilung der Spielerschaft vorgenommen. Dazu erfragte er in einem MUD²² von den Spielern die Faktoren und Aktivitäten, die ihnen Spaß bereiteten. Er erkannte wiederkehrende Muster:

- **Achiever:** Der Spieler versucht, selbstgesteckte Ziele zu erreichen und Spielerrungenschaften in großen Mengen zu horten.
- **Explorer:** Das Ziel des Spielers ist, möglichst viel Wissen über das Spiel zu bekommen. Er möchte alles ausprobieren.
- **Socializer:** Die Kommunikations-Werkzeuge des Spiels werden genutzt, um mit Mitspielern zu kommunizieren und gemeinsam zu handeln.
- **Killer:** Dieser Spielertyp versucht, anderen Mitspielern seine Macht aufzuerlegen und sie zu dominieren oder sie durch seine Hilfe an sich zu binden.

Die angegebenen Typen kommen nicht in ihrer Reinform im Spiel vor, sondern ein Spieler hat gewöhnlich Züge aller vier Typen, von denen jedoch einer dominant ist²³. Yee (2006) hat Bezugnehmend auf dieses Modell und empirischen Daten eine Taxonomie der Motivationen von Spielern eines MMOG erstellt, die als Werkzeug zu quantitativen Forschungen genutzt werden kann. Cummings (2011) stellt fest, dass die Taxonomie von Bartle für die Spieler eines MUDs entwickelt wurde, sich aber nicht notwendigerweise für alle Genres von Spielen verallgemeinern lässt, was er anhand von SNGs erläutert. Für diese Spiele – und generell für alle Computerspiele, die

²¹ www.secondlife.com, letzter Zugriff am 3.8.2009

²² *Multi User Dungeon*, ein textbasiertes digitales Rollenspiel.

²³ Bartle hat in einer späteren Arbeit die Einteilung in 8 Gruppen verfeinert (2003).

regelmäßig mit einem Server kommunizieren – gibt es eine wertvolle Möglichkeit, das Verhalten der Spieler zu beobachten: Das Profiling. Dabei werden die Aktionen der Spieler an den Server gesendet und dort aufgezeichnet. Diese Daten können ausgewertet werden, um Präferenzen der Spieler zu erkennen. Das Design des Spiels kann entsprechend der Benutzervorlieben ausgerichtet werden. Zynga, der erfolgreichste Hersteller von Facebook-Spielen, nutzt diese Methodik für das Design ihrer Spiele (Nutt, 2011). Carron & Marty (2011) beschreiben die Nutzung von Profiling-Daten ergänzt um eine per Fragebogen erhobene Selbsteinschätzung. Mit Hilfe dieser Methoden soll der Spieler typisiert werden und die Spielmechaniken auf ihn zugeschnitten werden. Deen & Schouten (2011) beschreiben verschiedene Lernstile, die mit Spielstilen korrespondieren.

Es lässt sich zusammenfassen, dass Computerspiele – wenn sie den Anspruch haben, einen Großteil der Lerner zu erreichen – sich mehr dem Stil der Lerner anpassen müssen als herkömmliche Medien. Um eine derartige Anpassung zu erreichen, wird versucht, die Bedürfnisse und Eigenschaften der Spieler zu kategorisieren, um dann für diese Kategorien geeignete Spielmechaniken zu finden. Dieses erhöht die Komplexität zur Erstellung von Lernspielen weiter: Es reicht nicht nur, funktionierende Software zu erstellen, deren Benutzung Spielspaß erzeugt. Zusätzlich muss dieses auch für verschiedene Spielertypen geschehen.

4.3 Zielgerichteter Einsatz in der Lehre

Dass durch Spiele gelernt werden kann, wurde wissenschaftlich wiederholt nachgewiesen. Ein Beispiel ist die Arbeit von Squire (2003), in der er Civilization für den Geschichtsunterricht nutzt. Söbke, Corredor et al. (2011) fanden heraus, dass auch SNGs zum gemeinschaftlichen Wissensaufbau genutzt werden können. Hier zeigt sich aber auch die Problematik eines Spiels: Das aufgebaute Wissen ist zum einen spezifisch für das Spiel und damit im richtigen Leben nicht direkt nutzbar, zum anderen aber auch ungerichtet: D.h. ein kommerzielles Spiel ist selten direkt in der Lehre einsetzbar, da die Inhalte des Spiels und die Lernziele nicht notwendigerweise eine große Schnittmenge aufweisen.

Spiele mit dem Anspruch, von allen Lernenden eines Kurses gespielt zu werden müssen verschiedene Bedingungen erfüllen:

- **Effizienz:** Spielerisches Lernen hat den Ruf, dass es nicht genauso effizient ist wie herkömmliche Unterrichtsmethoden. Langsameres Lernen würde verhindern, dass die Ziele von straffen Lehrplänen erreicht werden können²⁴.
- **Hohe Reichweite:** Es sollten alle Lernenden mit dem Spiel erreicht werden. Im Beispiel von Squire war es so, dass einige Schüler überragende Lernerfolge erzielt haben, eine nicht vernachlässigbar große Gruppe aber mit den Unterrichtseinheiten nicht erreicht werden konnte. In der überwiegenden Mehrheit der Studien, die den Erfolg von spielbasiertem Lernen zeigen, bezieht sich der Lernerfolg nur auf eine Auswahl der Grundgesamtheit – bei Söbke, Corredor et al., (2011) beispielsweise nur auf die Personen, die freiwillig in ihrer Freizeit spielen und gleichzeitig an einem Wiki mitarbeiten.
- **Lehrplan-Relevanz:** Werden Spiele im Unterricht eingesetzt, so müssen Spiele eingesetzt werden, die vom Lehrplan verlangte Fähigkeiten vermitteln. Spannende, in den jeweili-

²⁴ Bei einer Bewertung der Effizienz der Lernmethode sollte jedoch die Qualität der Lernergebnisse ebenfalls beachtet werden.

gen Kontext passende Spiele²⁵ zu finden bzw. zu erstellen ist jedoch ein schwieriges Unterfangen.

Verschiedene Szenarien für den Einsatz von Spielen im Unterricht wurden überprüft, z.B. von M. Wagner & Mitgutsch (2008). Wichtige Erkenntnisse sind:

- **Vor- und Nachbereitung** Beim Einsatz von Spielen in formalen Bildungskontexten hat sich herausgestellt, dass eine Vorbereitung („Briefing“) und Auswertung („Debriefing“) der Spielsitzungen dem Lernerfolg zuträglich sind – dadurch wird Reflexion angeregt.
- **Schulung der begleitenden Personen** Die Personen, die die Spielsitzungen begleiten, müssen Kompetenzen im Medium *Computerspiel* erwerben. Ihre Aufgabe ändert sich vom Lehrer, der das Wissen präsentiert, zu einem Begleiter, der den Lernenden beistehen kann, wenn diese sich mit den Spielen auseinandersetzen²⁶.

Wenn es eine Herausforderung ist, Spiele in die herkömmlichen Strukturen der Wissensvermittlung einzubetten, so ist eine Lösungsmöglichkeit, nach alternativen Einsatzformen für Spiele zu suchen. Da der zeitliche Aufwand für Spiele im Unterricht – wie von M. Wagner & Mitgutsch (2008) beschrieben – sich oft als zu hoch herausstellt, so können Spiele auch als zusätzliches Lernmedium in der Nicht-Unterrichtszeit der Spieler eingesetzt werden – als Ausbildungsbegleitung. Auch sind andere, vom herkömmlichen Unterrichtsparadigma abweichende Lernformen denkbar. Vielleicht verlangt das Medium *Computerspiel* nach einer eigenen, auf das Medium zugeschnittenen Lernform.

²⁵ Hiermit sind Spiele gemeint, die fachliche Fähigkeiten vermitteln. Spiele, die Metaskills fördern, sind unter den kommerziellen Spielen (COTS) einfacher zu finden.

²⁶ Dass – ungeachtet aller Technik – dem Lehrer die entscheidende Rolle bei der Entwicklung von Lernenden zukommt, zeigt eine Studie von Mitra, Dangwal & Thadani (2008): Hier wurde ein Zusammenhang zwischen der Zufriedenheit der Lehrer mit der gegenwärtigen Lebenssituation und den Noten ihrer Schüler festgestellt.

5 Abbildungsverzeichnis

Abbildung 1: Taxonomie von Computerspielen als Lernmedium	4
Abbildung 2: Pathfinder: Arbeitsumgebung.....	10
Abbildung 3: Pathfinder: Dialoge und Feedback.....	11

Anhang F: *Easy Gold* – Eine Quizanwendung für Social Networks

Inhaltsverzeichnis

1 Einleitung.....	3
2 Grundlagen	4
2.1 Spielprinzip	4
2.2 Oberflächenmodul.....	6
2.3 Einbettung in die Spielplattform	6
2.4 Status und Rechte: Administrator	7
3 Spielbestandteile	7
3.1 Spielmodus	7
3.1.1 Lernmodus.....	8
3.1.2 Battle-Modus.....	8
3.1.3 Turnier-Modus.....	8
3.1.4 Team-Match-Modus.....	9
3.2 Fragenstruktur	9
3.3 Fragenliste	10
3.4 Thematische Gruppierung.....	11
3.4.1 Zuordnung von Tags	11
3.4.2 Fähigkeiten und Tags.....	12
3.4.3 Automatische Auswahl der Fragen	12
3.5 Fähigkeiten und Zertifikate	13
3.6 Zertifikate	14
3.7 Passender Schwierigkeitsgrad	15
3.8 Fragenstapel.....	16
4 Interaktionen	18
4.1 Beantwortung einer Frage.....	18
4.2 Erstellen von Fragen	19
4.3 Rückmeldung auf Fragen	20
4.4 Weiterleitung von Fragen.....	21
5 Technische Systemelemente	22
5.1 Forum	22
5.2 Nachrichtenstrom und Schwarzes Brett.....	23
5.3 E-Mail.....	24
6 Anforderungsabgleich	25
6.1 Definition einer guten Leistung	25
6.2 Entwicklung von Selbsteinschätzung und Reflektionen	25
6.3 Informationen über den Lernprozess.....	26
6.4 Dialog.....	26
6.5 Motivierendes Feedback	26
6.6 Lernziel	27

6.7 Feedbackschleife.....	27
7 Schlussbetrachtung.....	27

1 Einleitung

Tests, die aus Multiple-Choice-Fragen bestehen, sind ein oft benutztes Mittel zur Lernzielkontrolle. Multiple Choice Tests sind – im Vergleich zu Lernszenarios – einfach zu entwickeln: Auch der Aufwand einer softwaretechnischen Umsetzung ist wesentlich geringer. Es müssen keine Simulationsmodelle aufgestellt werden und Simulationsläufe durchgeführt werden. Dieser Umstand wird zum Anlass genommen, die Prinzipien der erweiterbaren bauphysikalischen Spielplattform (Söbke, Hadlich, Bröker, & Kornadt, 2011) mit Hilfe des Inhaltes „Fragen nach dem Antwort-Wahl-Verfahren“ zu demonstrieren. Grundlegende Eigenschaften der bauphysikalischen Spielplattform sind auch Bestandteile dieses Spiels¹ und können damit unabhängig von der Realisation von Simulationsszenarien getestet werden:

- **Spiel in der Gemeinschaft:** Durch die Einbettung des Spiels in ein soziales Netzwerk² nimmt die Interaktion der Spieler untereinander einen hohen Stellenwert ein. Damit soll gemeinschaftliches Lernen ermöglicht werden.
- **Spieler als Inhalte-Erzeuger (Content Provider):** Spieler können selbst Inhalte beisteuern. Das senkt zum einen die Erstellungskosten, zum anderen lernt der Spieler durch die Konstruktion von Inhalten.
- **Anpassung an die aktuellen Fähigkeiten der Spieler:** Dem Spieler sollen zu seinen aktuellen Fähigkeitsniveau passende Probleme präsentiert werden, die weder eine Überforderung noch eine Unterforderung auslösen.
- **Gezielte Ausbildung von Fähigkeiten** Das System bietet Unterstützung bei der Auswahl von zu einem Ausbildungsziel passenden Problemen.

Als Leitfaden zur Überprüfung des Designs für das Spiel wird die Arbeit von David Nicol genutzt: Er überträgt die 7 Prinzipien guten Feedbacks auf Multiple-Choice-Tests (Tabelle 1). Auf die Erfüllung der einzelnen Kriterien wird in einem abschließenden Kapitel eingegangen.

Prinzip
1 Definiert eine gute Leistung (Kriterien, Ziel und Standards).
2 Nützt der Entwicklung der Fähigkeit zur Selbsteinschätzung und Reflektion beim Lernen.
3 Liefert Informationen hoher Qualität über den Lernprozess.
4 Fördert den Dialog um das Lernen.
5 Das Feedback ist motivierend.
6 Führt den Lerner an das Lernziel.
7 Feedback beeinflusst die Lerninhalte.

Tabelle 1 7 Prinzipien guten Feedbacks (Nicol & Macfarlane-Dick, 2006)

Dieses Dokument beschreibt das Spiel, das bisher bereits als Prototyp *Easy Gold* in der Facebook-Anwendung *BuildVille* mit einigen der beschriebenen Eigenschaften und Leistungsmerkma-

¹ Im Folgenden wird für die Fragen-Anwendung die Bezeichnung „Spiel“ genutzt: Das reine Beantworten von Fragen wird sicherlich nicht von vielen Menschen grundsätzlich als Spiel empfunden, jedoch sind in der Anwendung einige spielerische Mechanismen und Elemente enthalten, die die Bezeichnung „Spiel“ rechtfertigen. Jesse Schell (2008) definiert „A game is a problem-solving activity, approached with a playful attitude.“. Sieht man die Beantwortung einer Frage als Lösen eines Problems und geht man davon aus, dass die Benutzer der Anwendung diese nicht nur benutzen, um Fragen zu beantworten, sondern auch die spielerischen Elemente genießen können, dann ist die Bezeichnung „Spiel“ durchaus gerechtfertigt.

² Mit dem Begriff „Soziales Netzwerk“ wird hier die lose Verbindung von Personen in einer Netzgemeinschaft verstanden. Im Englischen gibt es hierfür auch den Begriff *Social Network Service (SNS)*. Erfolgreiche und populäre SNS sind z.B. *Facebook* oder *Google+*.

len existiert und ersten Tests unterzogen werden konnte³. In diesem Artikel werden zusätzliche Eigenschaften beschrieben, um die der Prototyp erweitert werden kann und die seine Leistungsfähigkeit als Lernspiel erhöhen könnten.

Das soziale Netzwerk, in das der Prototyp eingebettet ist und auf dem er bereitgestellt wurde, ist Facebook⁴. Es ist eines der am weitesten genutzten sozialen Netzwerke (MacTechNews, 2010). Zusätzlich ist es Basis für einige sehr erfolgreiche Spiele (Appdata, 2010). Bei der Entwicklung des Prototyps wurde darauf geachtet, dass die Funktionalität zur Integration in ein soziales Netzwerk in einem Modul gekapselt ist. Das soll die Portierung des Systems in ein anderes soziales Netzwerk erleichtern.

Grundsätzliches Ziel ist die Erstellung eines weitestgehend selbstorganisierenden Systems: Die Inhalte – abgesehen von einer initialen Bestückung – sollen nicht durch den Betreiber der Plattform erstellt werden, sondern durch die Spieler selbst. Selbstorganisierend bedeutet, dass die durch die Spieler erzeugten Fragen auch wieder durch die Spieler bewertet werden und damit eine Qualitätssicherung weitestgehend ohne administrativen Aufwand stattfindet.

Anzumerken ist, dass der existierende Prototyp noch nicht ausbalanciert ist. Daher sind an einigen Stellen Angaben zur Spielbalance recht lückenhaft – beispielsweise was Menge und Art der Belohnungen angeht. Ebenso führt die Vermischung von schon bestehenden Funktionalitäten und vorgeschlagenen Erweiterungen zu einem ungleichmäßigen Detaillierungsgrad der Beschreibung, der aber die Verständlichkeit der Darstellung nicht beeinträchtigen sollte. Auch ist die Vermengung der Metapher „Schatzsuche“ mit dem Mittel der Multiple-Choice-Frage auf den ersten Blick kein ausgewogenes Spieldesign, sondern der Notwendigkeit geschuldet, ein Lernspiel zu entwickeln. Mit *Easy Gold* bzw. *BuildVille* konnte – trotz aller genannten Defizite – die technische und methodische Machbarkeit eines SNG mit Lerninhalten demonstriert werden. In der Zwischenzeit konnte im kommerziellen Kontext das Facebook-Spiel *Triviador* entdeckt werden (Nelson, 2011; THX_Games_PLG, 2011). *Triviador* benutzt ebenfalls die Spielmechanik Multiple-Choice-Frage bzw. Schätzfrage, verknüpft dieser mit der Spielidee des Brettspiels *Risiko* – an den Stellen, an denen im herkömmlichen Spiel Würfel über den nächsten Spielzug entscheiden, werden in *Triviador* Fragen benutzt – und zeigt damit, dass sich Fragen durchaus zu spannenden Spielmechaniken verarbeiten lassen.

2 Grundlagen

2.1 Spielprinzip

Im Spiel wird die Metapher der Schatzsuche genutzt. Auf dem Spielfeld sind Schätze in Form von Schatztruhen versteckt. Darüber liegt eine Bedeckung, die in einzelne Kacheln eingeteilt ist und Erde symbolisiert. Diese kann durch Benutzung eines Spatens kachelweise freigekratzt werden.

³ Der Prototyp steht zur allgemeinen Verfügung unter <http://apps.facebook.com/buildville>.

⁴ Facebook ist erreichbar unter <http://www.facebook.com>.

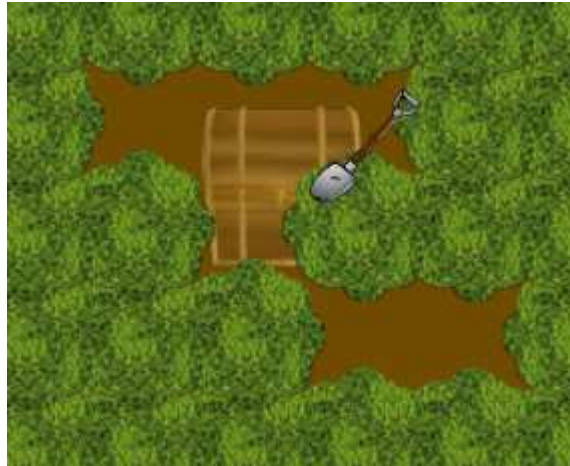


Abbildung 1: Schatzfeld mit Schatzkiste, Spaten und Abdeckung

Jeder Spatenstich erfordert den Einsatz von Coins und ergibt Erfahrungspunkte (XPs). Damit wird Aktivität des Spielers belohnt. Trifft der Spieler auf eine Schatztruhe, so wird dies optisch angezeigt: es sind noch weitere Spatenstiche notwendig, um die Schatztruhe komplett freizulegen. In Abbildung 1 ist eine teilweise freigelegte Schatzkiste mit Resten der Abdeckung sowie dem Spaten dargestellt. Erst nach der kompletten Freilegung lässt sich die Schatztruhe mit einem Klick öffnen. Bevor die Inhalte der Schatztruhe entnommen werden können, muss jedoch noch eine Multiple-Choice-Frage beantwortet werden. In Abbildung 2 ist eine solche dargestellt.



Abbildung 2: Beispiel einer Frage

Wird die Frage korrekt beantwortet, hat der Spieler freien Zugriff auf die „Schätze“ in der Schatztruhe: Er kann Coins, XPs und Spielgegenstände an sich nehmen. Die Spielgegenstände sind auch in der übergeordneten Spielplattform nutzbar.

2.2 Oberflächenmodul

Die im vorigen Kapitel „Spielprinzip“ beschriebene Oberfläche kann als separates Modul mit einer definierten Aufgabe aufgefasst werden. Daher lässt es sich durch Alternativen ersetzen. Beispielsweise ist es denkbar, eine Oberfläche nach dem Prinzip des Spieles *Moorhuhn* zu nutzen (Phenomedia, 2012): Der Spieler erlegt die Moorhühner, nach dem Erlegen kann er die Reste des Moorhuhns nach Belohnungen durchsuchen. Bevor dies möglich ist, muss er aber zunächst eine Frage beantworten.

In ähnlicher Weise lassen sich weitere Oberflächen konzipieren. Diese haben die Aufgabe, eine Spaß machende Grundtätigkeit zu ermöglichen, die dann Gelegenheiten bereitstellt, die Beantwortung von Multiple-Choice-Fragen in den Spielablauf einzuweben. Durch die Entwicklung verschiedener Oberflächenmodule kann eine Anpassung an die Vorlieben unterschiedlicher Spielertypen erfolgen und damit die Gruppe derjenigen, die ein solches Lernspiel mit Spaß freiwillig spielen, möglicherweise erhöht werden (Bartle, 1996; Cummings, 2011; Yee, 2006).

2.3 Einbettung in die Spielplattform

Das hier vorgestellte Konzept des Mini-Spiels *Easy Gold* eignet sich, um es in die übergeordnete Spielplattform⁵ zu integrieren. Dabei entstehen einige Schnittstellen:

- **Aufruf von *Easy Gold* aus der Spielplattform:** Grundsätzlich gibt es einen Menüpunkt, mit dessen Hilfe es möglich ist, *Easy Gold* aufzurufen (s. Abbildung 3). Da eine Funktion von *Easy Gold* – wie der Name sagt – die schnelle Bereitstellung von spielnotwendigen Währungen – z.B. Coins – ist, kann der Aufruf von *Easy Gold* an einigen weiteren Stellen möglich gemacht werden. Vornehmlich sind es solche Stellen, an denen der Spieler akuten Bedarf an Coins haben könnte.
- **Gemeinsame Währungen:** *Easy Gold* benutzt dieselben Währungen wie die übergeordnete Spielplattform: Coins und XPs. Coins werden benötigt, um graben zu können. Als Belohnung für das Graben werden XPs ausgeschüttet. Ebenfalls sind in den Schatzkisten Coins und XPs verborgen.
- **Spezifische Belohnungen:** Belohnungen, die in *Easy Gold* für das erfolgreiche Beantworten von Fragen verdient werden können, haben in der übergeordneten Spielplattform eine Bedeutung und können dort eingesetzt werden: Beispielsweise können Messinstrumente, die sich in bauphysikalischen Problemstellungen einsetzen lassen, oder Baumaterialien als Preise gewonnen werden.
- **Fertigkeiten:** Mit der korrekten Beantwortung von Fragen vergrößert der Spieler den Wert spezifischer fachlicher Fähigkeiten, denen diese Frage zugeordnet ist. Diese Fähigkeiten sind auch in der übergeordneten Spielplattform benutzbar.
- **Freunde:** Auch für *Easy Gold* werden assoziierte Spieler benötigt. Diese rekrutieren sich aus den „Freunden“ des sozialen Netzwerks, das von der Spielplattform genutzt wird⁶.

⁵ Diese wird beschrieben in der übergeordneten Hauptarbeit dieses Anhangs (Söbke, Hadlich, et al., 2011; Söbke, 2013).

⁶ In der derzeitigen Implementierung ist dies der Social Network Service *Facebook*.



Abbildung 3: Aufruf von *Easy Gold* aus der übergeordneten Plattform

2.4 Status und Rechte: Administrator

Obwohl das Spiel selbstorganisierend ist, kann zu manchen Zeitpunkten ein Administrator⁷ notwendig sein. Dieser ist für die Auflösung von Konflikten zuständig, deren automatisierte Lösung übermäßig viel Aufwand verursachen würde. Ein Beispiel für den Einsatz eines Administrators ist die Beurteilung einer Frage, die durch die Spieler unterschiedlich bewertet wird. Der Administrator kann die Frage mit Hilfe seines Fachwissens persönlich begutachten, ein Urteil fällen und sie gegebenenfalls auch editieren, damit sie verständlicher wird.

An diesem Beispiel wird gleichfalls deutlich, dass die Rolle des Administrators nicht allein durch einen anderen Administrator zugeteilt werden muss, sondern dass die Erteilung entsprechender Rechte auch automatisch nach der Steigerung fachlicher Kompetenzen zugeteilt werden kann, also aus der Aktivität im System erwächst (s.a. Tabelle 3: Berechtigungen auf S. 10).

Ein weiteres Element des Spiels ist es, den Status des Spielers transparent für alle anderen Spieler zu machen. Zum Status gehören alle Informationen, die während des Spiels gesammelt werden können. Beispiele sind Anzahl der beantworteten Fragen, Anzahl der weitergeleiteten Fragen, Anteil der richtig beantworteten Fragen, errungene Berechtigungen und Grade der unterschiedlichen Fähigkeiten. An geeigneten Stellen werden auch Hitlisten in das Spiel eingebunden.

Die Absicht hinter der Anzeige des öffentlichen Status ist es, den zielgerichteten Wettbewerb zwischen den Spielern zu unterstützen. Daher werden nur Statusinformationen öffentlich dargestellt, die den gesunden Wettbewerb fördern. Beispiele für nicht zielgerichtete Statusinformationen sind die Anzahl der Logins und die Gesamtzeit der Aktivität auf der Plattform. Es ist das Ziel der Plattform, die Spieler zu unterhalten und dabei Lerninhalte zu vermitteln, nicht aber die Spieler möglichst lange und möglichst oft in der Plattform zu halten – daher soll es beispielsweise keinen Wettbewerb in Anzahl der Login-Vorgänge sowie der Online-Zeit geben.

3 Spielbestandteile

3.1 Spielmodus

Das Spiel kann in verschiedenen Varianten gespielt werden, die unterschiedliche Ziele besitzen: Einmal steht Lernen im Vordergrund, zum anderen wird die Unterhaltungs- und Spaßkomponente betont.

⁷ In Forensoftware wird diese Rolle auch häufig "Moderator" genannt.

3.1.1 Lernmodus

Im Lernmodus werden dem Spieler die Fragen nach dem Prinzip der Lernkartei präsentiert (Leitner, 2011): Eine Frage, die korrekt beantwortet werden konnte, wird in die nächste Kategorie eingeordnet. Dies bedeutet, dass sie erst nach einem größeren zeitlichen Abstand (in der ersten Kategorie ist dieses ein Tag) erneut gestellt wird. Kann sie auch dann wieder richtig beantwortet werden, wird sie wiederum eine Kategorie höhergestuft. Dieses wird solange mit erhöhten zeitlichen Intervallen durchgeführt, bis die Endkategorie erreicht ist. Dann gilt die Frage mit ihrer Antwort als gelernt und als endgültig im Langzeitgedächtnis verankert. Eine Frage, die falsch beantwortet wird, wird wieder der ersten Kategorie zugeteilt, unabhängig von ihrer vorherigen Kategorie. Diese Prinzipien sind in der Lernsoftware Phase-6 (2012) implementiert, die auf der Vergessenskurve von Hermann Ebbinghaus aus dem 19. Jahrhundert basiert (Ebbinghaus, 1992) und deren Effizienz von Fritz, Passey, & Morris (2009) untersucht wurde.

3.1.2 Battle-Modus

Mit dem sogenannten Battle-Modus ist die Spielmechanik der Zeitbeschränkung verbunden. Gibt es im Lernmodus keine Zeitbeschränkung, d.h. dort kann der Spieler bei Bedarf die richtige Lösung auch mit Hilfe von externen Quellen ermitteln, so gibt es hier mehrere Arten von Zeitbegrenzung: Zum einen ist die Bearbeitung eines kompletten Schatzfeldes (das heißt, alle im Schatzfeld versteckten Kisten müssen gefunden werden) in einer bestimmten Zeit notwendig. Zum anderen hat man für die Beantwortung einer Frage nur eine begrenzte Zeitdauer. Werden alle Fragen des Schatzfeldes innerhalb der vorgegebenen Zeit bearbeitet und korrekt beantwortet, so wird eine zusätzliche Belohnung ausgeschüttet. Der Spieler kann die Zeit, innerhalb derer er eine Frage beantworten möchte, selbst wählen. Eine kleinere Maximalzeit bringt eine höhere Belohnung. Die Zeit, die er für die Beantwortung des gesamten Schatzfeldes zur Verfügung hat, wird berechnet aus der Zeit pro Frage multipliziert mit der Anzahl der Schatzkisten. Dies scheint eine gute erste Näherung zu sein, denn in dieser Zeit müssen die Schatzkisten zusätzlich zur Beantwortung noch freigelegt werden. So ist die Einhaltung der Gesamtzeit auch eine Herausforderung, wenn alle Einzelzeiten zur Beantwortung einer Frage nicht überschritten werden.

Der Battle-Modus wird durch entsprechende Hintergrundmusik und funkensprühende Grafik beim Ausgraben der Schatzkisten untermalt.

Es ist zu untersuchen, welchen Einfluss die knappe Zeit auf den Lernerfolg hat. In einigen Spielen ist Zeitdruck ein elementares Prinzip des Spieles (z.B. im juristischen Lernspiel „*Jagd nach dem Katzenkönig*“ (Lernfreak GbR, 2012)). Negativ auf die Lerneffektivität könnte sich jedoch auswirken, dass durch den Zeitdruck keine Zeit zur Reflektion und damit zur Verankerung des Gelernten bleibt.

3.1.3 Turnier-Modus

Nach dem Vorbild der Challenge in dem Wimmelbildspiel „Gardens of Time“ (Playdom, 2012), das über das soziale Netzwerk Facebook spielbar ist⁸, gibt es einen Turniermodus. Hierbei wählt ein Spieler ein Themengebiet, eine Anzahl und einen Schwierigkeitsgrad der zu stellenden Fragen für ein Turnier. Daraufhin schickt er Einladungen zu diesem Turnier an verschiedene Freunde. Er selbst startet dann das Turnier und versucht, alle Schatzkisten auf dem ihm präsentierten

⁸ Das Spiel „*Gardens of Time*“ wurde 2011 als „Best Social Network Game“ im Rahmen der Game Developers Choice Online Awards gekürt (Osborne, 2011).

Schatzfeld möglichst schnell zu finden und die damit verbundenen Fragen korrekt zu beantworten. Die erreichte Leistung wird mit einer Punktzahl bewertet, um eine Rangfolge der Turnierteilnehmer herstellen zu können. In die Punktzahl geben Parameter der Problemlösung wie die benötigte Zeit, die Anzahl der richtigen Antworten, der Schwierigkeitsgrad der Fragen und die Anzahl der umgegrabenen Schollen zur Entdeckung der Schatzkisten ein. Alle eingeladenen Teilnehmer haben die Möglichkeit, ebenfalls das Schatzfeld zu bewältigen. Sie bekommen nicht notwendigerweise dieselben Fragen präsentiert, sondern Fragen, die vom Schwierigkeitsgrad und von der Thematik her ähnlich sind.

Nach einer vorher festgelegten Zeit kann der Organisator das Turnier als beendet erklären – unabhängig davon, wie viele der eingeladenen Spieler letztendlich die Herausforderung angenommen haben und ein Ergebnis geliefert haben. Der Spieler mit der höchsten Punktzahl gewinnt das Turnier, und erhält eine Belohnung. In seiner persönlichen Statistik als Ausdruck des Status wird ein weiterer Turniersieg verbucht. Gleichzeitig wird der Bilanz zweier Spieler eine weitere Entscheidung hinzugefügt. Damit können sich zwei Spieler untereinander vergleichen.

3.1.4 Team-Match-Modus

Eine weitere Variante des Turnier-Modus ist der des Team-Match-Modus: Dieser folgt einer Spielmechanik, die u.a. in *StarCraft II*⁹ funktioniert. Es finden sich Gruppen von Spielern zusammen, die ein Team bilden und sich für einen Wettkampf beim System anmelden. Das System sucht nun einen passenden Gegner mit Hilfe der Parameter Gruppenstärke, Gruppenfähigkeit (abgeleitet aus den Fähigkeiten der jeweiligen Teammitglieder) und gewünschtem Themenbereich. Ist ein Gegner gefunden, so kann das Match starten. Alle Spieler beginnen, ihr Schatzfeld abzusuchen und die Fragen zu beantworten. Die aktuellen Spielstände der Teams werden angezeigt, so dass jederzeit Klarheit über den Status des Spiels herrscht. Befinden sich die Spieler in einem Raum so wird eine rege Diskussion über die Frageninhalte stattfinden, die auch nach dem Match anhält und zu Reflektion führt¹⁰.

3.2 Fragenstruktur

Die Elemente einer Frage sind in Tabelle 2: Elemente einer Frage aufgeführt. Es sind verschiedene Typen von Fragen möglich: Fragen mit genau einer richtigen Antwort, mit mehreren richtigen Antworten oder mit einer Freitextantwort. In Abhängigkeit vom Typ der Frage müssen nicht alle Elemente einer Frage gefüllt sein.

⁹ Zugriff über <http://us.blizzard.com/en-us/games/sc2/> (letzter Zugriff am 6.8.2012)

¹⁰ Während meiner Zeit als Visiting Scholar an der *University of Wisconsin – Madison* wurde in meiner Projektgruppe zur Ablenkung und Entspannung oder zur Abrundung des Tages gerne eine Partie *StarCraft II* eingelegt (s. <http://bauhausgames.blogspot.de/search/label/StarCraft>, letzter Abruf am 6.8.2012). Wesentliche und erstaunlichste Erkenntnisse waren die Diskussionen, die sich nach einem Match einstellten: Es wurden die unterschiedlichen Spielmanöver analysiert sowie Fehler und Wendepunkte im Spielverlauf beschrieben. Es war ein Ort der Reflektion. Zudem konnte ein großes Gemeinschaftsgefühl beobachtet werden sowie eine Identifikation mit den Spielergebnissen: Bei einem Sieg gab es gegenseitige Glückwünsche, bei einer Niederlage herrschte gedämpfte Stimmung.

Element	Beschreibung
Frage	Text der Frage
Unterstützende Grafiken	Eine oder mehrere Grafiken, auf die sich die Frage bezieht ¹¹ (Optional).
Typ	Typ der Frage, von dem auch die Darstellung abhängt: Ist eine vorgegebene Antwort richtig? Sind mehrere Antworten richtig? Freitexteingabe möglich?
Mindestens 2 Antwortmöglichkeiten	Eine Liste von möglichen Antworten. Zu jeder Antwort gehört auch ein Kennzeichen, das angibt, ob die Antwort richtig ist oder nicht (Optional, wenn Freitextantwort möglich ist).
Freitextantwort	Kennzeichen, ob eine Freitextantwort möglich ist.
Korrekte Antwort (Text)	Korrekte Freitext-Antwort (nur notwendig, wenn Kennzeichen „Freitextantwort“ gesetzt ist).
Rückmeldungstexte für jede der Antwortmöglichkeiten	In den Rückmeldungstexten können – sofern es sich anbietet – Hinweise gegeben werden, weshalb eine Antwort nicht richtig ist. Für verschiedene „Schlüsselwörter“ in der Freitextantwort können ebenfalls Rückmeldungstexte hinterlegt werden (Optional).
Tags	Tags zur fachlichen Einordnung der Frage

Tabelle 2: Elemente einer Frage

3.3 Fragenliste

Im Rahmen des Spiels gibt es einen Einstiegspunkt zu einer Liste der Fragen nach dem Vorbild der Software von Stackoverflow.com (2012). Hier kann der Spieler bestimmte Fragen suchen und einsehen. Zur Suche kann der Spieler Tags, Freitext und Identifikatoren verwenden. Je nach seiner Rolle (Administrator, Fachexperte oder Spieler) hat er verschiedene Berechtigungen im Umgang mit der Frage. Diese sind in Tabelle 3 exemplarisch dargestellt. Sie können noch weiter verfeinert werden, eine Orientierung können hier die je nach Status des Benutzers vergebenen Rechte in der Software von Stackoverflow.com sein.

Berechtigung
Frage ändern
Frage ansehen
Kommentar erstellen
Frage deaktivieren
Frage aktivieren
Tag zuordnen
Tag erstellen

Tabelle 3: Berechtigungen

In der Detailansicht einer Frage finden sich alle mit dieser Frage verbundenen Informationen. Diese sind in Tabelle 4 dargestellt.

Information	Bemerkung
Fragentext	
Antwortmöglichkeiten	
Richtige Antwort	
Fragenersteller	Verweis auf den Spieler, der die Frage erstellt hat.

¹¹ Eine Erkenntnis der bisher erstellten und getesteten Prototypen ist, dass die Ersteller von Fragen sich häufig die Einbindung einer Grafik gewünscht haben.

Antwort-Statistiken	Hierzu zählen die Anzahl der Auslieferungen der Frage, die Anzahl der richtigen und die Anzahl der falschen Antworten sowie die Belohnungen, die der Ersteller bekommen hat.
Tags	Kategorien, denen die Frage angehört
Diskussion um die Frage	Diskussionen, die sich um die Fragen entwickelt haben.
Likes/Dislikes	Wie oft haben Spieler eine Bewertung für die Frage abgegeben?

Tabelle 4: Informationsdetails einer Frage

3.4 Thematische Gruppierung

Eine thematische Gruppierung der Fragen erfolgt mit Hilfe von sogenannten Tags¹²: Jeder Frage können ein oder mehrere¹³ Tags zugeordnet werden. Die Tags entsprechen einer inhaltlichen Gruppierung der Fragen. Bei der Suche nach Schatzkisten gibt es den sogenannten *Magic Spade*: Der Spieler kann vor dem Graben aus einer Liste der vergebenen Tags wählen (s. Abbildung 4). Die Fragen, die ihm dann vor der Öffnung der gefundenen Schatzkisten präsentiert werden, tragen dieses Tag.



Abbildung 4: "Magic Spade": Auswahl eines Tags

3.4.1 Zuordnung von Tags

Es gibt verschiedene Möglichkeiten, einer Frage Tags zuzuordnen:

- **Bei der Erstellung:** Tags können einer Frage zugeordnet werden bei der Erstellung derselben. Dabei erfolgt die Auswahl aus der Menge der schon existierenden Tags¹⁴.
- **Nach der Beantwortung:** Nach der Beantwortung einer Frage hat der Spieler die Möglichkeit des Feedbacks (s.a. Kap. 4.3). Als Teil des Feedbacks wird den Spieler die Gele-

¹² Das Wort „tag“ kommt aus dem Englischen und hat hier die Bedeutung eines Schlagwortes. Mit Hilfe von Tags werden die Fragen indiziert. Die Tätigkeit des Indizierens wird „tagging“ genannt und ist durch das Social Tagging im Web 2.0 bekanntgeworden, also der Verschlagwortung von benutzererzeugten Inhalten durch die Benutzer selbst (Peters, 2009).

¹³ Es können pro Frage beliebig viele Tags verteilt werden. Dies hat die Konsequenz, dass durch die richtige Beantwortung der Frage auch beliebig vielen Fähigkeiten Punkte zugewiesen werden können (s. Kap. 3.4.2).

¹⁴ Die Erstellung eines Tags bedarf einer separaten Berechtigung, die sich der Spieler mit wachsendem Status erarbeiten muss.

genheit geboten, weitere Tags der Frage zuzuordnen – sofern er der Meinung ist, dass noch einige fehlen. In Abhängigkeit vom Status des Spielers werden die Tags direkt zugeordnet oder – bei fehlenden Berechtigungen - erst nachdem mehrere Spieler jeweils ein übereinstimmendes Tag vorgeschlagen haben.

- **Edit-Modus:** Hat der Spieler die Erlaubnis, die Frage über die Fragenliste im Edit-Modus zu öffnen, kann er der Frage auch neue Tags zuordnen.

3.4.2 Fähigkeiten und Tags

Die Tags werden gleichzeitig auch als Fähigkeiten genutzt: Beantwortet ein Spieler eine Frage richtig, so wird für jedes Tag, den diese Frage besitzt, eine entsprechende virtuelle Fähigkeit ermittelt, dieser Fähigkeit wird dann ein entsprechender Punktwert zugeschlagen (s. Tabelle 6: Aktionen und Punktezahlen auf S. 14). Die konkrete Punktzahl für eine Fähigkeit wird im Rahmen dieser Arbeit auch Fähigkeitsgrad genannt.

Dass ein Tag einer Fähigkeit entspricht, ist die Standardkonfiguration. Von dieser Regel kann explizit abgewichen werden: In einem Editor können Tags bestimmten Fähigkeiten zugeordnet werden, so dass mehrere Tags zu einer Fähigkeit beitragen können oder auch Fähigkeitshierarchien aufgebaut werden können.

Mit Hilfe dieses Tag-Fähigkeits-Mechanismus ist es möglich, unterrichts- und kursspezifische „Fähigkeiten“ zu erstellen, deren Erwerb als Spielziel ausgegeben werden kann: Thematisch zu den Lernzielen des Unterrichts passende Fragen werden mit einem kurs-spezifischen Tag versehen. Unter Einsatz des *Magic Spade* (s. Kap. 3.4) kann der Spieler genau diese Fragen anfordern und die notwendigen Punkte zur Erreichung der Fähigkeit erzielen. Definierte Stufen von bestimmten Fähigkeiten können zu einem Zertifikat gebündelt werden (s.a. Kap. 3.6).

3.4.3 Automatische Auswahl der Fragen

Sofern der Spieler nicht mit Hilfe des *Magic Spade* die Fragen auf einen bestimmten Tag einschränkt, gibt es verschiedene Mechanismen, die die Auswahl der Fragen automatisiert durchführen. Zum einen ist dies die Historie der bisher beantworteten Fragen: Die werden zunächst präsentiert, damit sie durch den Lernzyklus gemäß des Karteikastenprinzips durchgeschleust werden können und in das Langzeitgedächtnis übergehen. Für die Auswahl der neuen Fragen werden die Tags der bisher beantworteten Fragen des Spielers genutzt: Es werden thematisch verwandte Fragen mit diesen Tags präsentiert.

Weiterhin werden auch die Vorlieben der Freunde des übergeordneten sozialen Netzwerks ausgewertet: Grundlage dieses Vorgehens ist die Annahme, dass Freunde in sozialen Netzwerken die gleichen Interessen haben könnten¹⁵.

Bei der Auswahl der Fragen ist durch eine Zufallskomponente zu gewährleisten, dass die Reihenfolge der Fragen nicht deterministisch und vorhersagbar ist. Ebenfalls wird der Status der Fragen

¹⁵ Das Internetradio Last.fm (www.last.fm) nutzt einen umgekehrt arbeitenden Algorithmus: Die vom Teilnehmer gehörte Musik wird registriert und mit den musikalischen Profilen anderer Teilnehmer verglichen. Bei einer hohen Übereinstimmung zweier Teilnehmer werden diese zu „musikalischen Nachbarn“ im Rahmen des sozialen Netzwerkes von Last.fm. Nachbarn machen dann innerhalb einer kleineren Gemeinschaft beispielsweise Musikvorschläge. Die Wahrscheinlichkeit, dass solche Hinweise erfolgreich aufgenommen werden von den Nachbarn ist aufgrund des ähnlichen Musikgeschmacks höher.

berücksichtigt: Um eine hohe Qualität der Fragen zu gewährleisten, werden bei der Auswahl Fragen mit einer guten Bewertung höher gewichtet.

Der Schwierigkeitsgrad der Fragen wird bei der Auswahl gleichfalls berücksichtigt.

Ein Algorithmus zur Auswahl der Fragen besitzt die folgende Form:

1. Bestimme die nach dem Prinzip der Lernkartei aktuell fälligen Fragen.
2. Bestimme die Tags der bisher beantworteten Fragen.
3. Bestimme die Freunde des sozialen Netzwerks.
4. Bestimme die Tags der durch die Freunde (Schritt 3) bisher beantworteten Fragen.
5. Mische die Tags aus Schritt 2 und Schritt 4. Reduziere die Tags um die Tags aus der Sperrliste für den Spieler (s. Kap. 4.3). Die Tags aus Schritt 2 erhalten ein höheres Gewicht, d.h. es werden im Folgeschritt relativ mehr Fragen gefunden, die dieses Tag tragen.
6. Bestimme Fragen, die bisher noch nicht beantwortet wurden, mit Hilfe der Tags aus Schritt 5. Fragen erhalten ein Gewicht relativ zu ihrer Bewertung, d.h. besser bewertete Fragen werden bevorzugt gestellt. Ebenfalls erhalten vom Schwierigkeitsgrad passende Fragen ein höheres Gewicht (s.a. Kap. 3.7).
7. Bestimme die zu stellende Frage aus einer gewichteten Mischung der Fragen aus Schritt 1 und der Fragen aus Schritt 6¹⁶.

3.5 Fähigkeiten und Zertifikate

Mit Hilfe von Tags werden die Fragen unterschiedlichen Fähigkeiten zugeordnet: Wird eine Frage richtig beantwortet, so erhält der Spieler für die Fähigkeiten, die durch die Tags bezeichnet werden, Punkte. Bestimmte Punktezahlen entsprechen bestimmten Stufen dieser Fähigkeiten. Diese sind in Tabelle 5 dargestellt. Bei der Erstellung des Punkteschemas wurde darauf geachtet, dass die ersten Stufen möglichst schnell erreicht werden können, um dem Spieler schnell eine positive Rückmeldung zu geben.

Stufe	notwendige Punkte
Beginner	3
Learner	10
Intermediate	30
Advanced	70
Expert	250
Genius	750

Tabelle 5: Fähigkeitsstufen und notwendige Punktzahlen

Die Zahl der Punkte, die ein Spieler durch die richtige Beantwortung einer Frage erhält, hängt von dem Modus ab, in der die Frage beantwortet wird. Mit dem Modus verknüpft ist die Nachhaltigkeit des Lerneffektes. Der Punktestand soll möglichst ein Abbild des Wissenstandes sein, daher werden richtige Antworten, die auf einen hohen Wissenstand hinweisen, besonders stark belohnt. Hierunter fallen insbesondere die höchsten Stufen des Zyklus nach dem Prinzip der Lernkartei. Ein Vorschlag für das Punkteschema ist in Tabelle 6 zu finden. Die dort aufgeführten Punkte werden pro Fähigkeit verteilt, mit der die jeweilige Frage über ein Tag verbunden ist. Da

¹⁶ Obwohl Fragen, die schon im Lernzyklus enthalten sind, bevorzugt werden sollen, sollten auch neue Fragen untergemischt werden, damit bei wenigen alten Fragen die Reihenfolge vom Spieler nicht vorhergesagt werden kann.

beliebig viele Tags möglich sind, können auch zu beliebig vielen Fähigkeiten mit der Beantwortung einer einzigen Frage Punktezuwächse erzielt werden.

Aktion	Punkte
Battle-Modus: Richtige Antwort	1
Turnier-Modus: Richtige Antwort	1
Lernmodus: Frage der Stufen 1 und 2	2
Lernmodus: Frage der Stufen 3 und 4	4
Lernmodus: Fragen der Stufe 5	6
Lernmodus: Fragen der Stufe 6 (Ende des Lernzyklus)	10

Tabelle 6: Aktionen und Punktezahlen

3.6 Zertifikate

Ein Zertifikat ist eine Auszeichnung für den Spieler. Es wird diesem verliehen, weil der Spieler eine Menge exakt definierter Fähigkeitsstufen im Spiel erreicht hat. Ein Zertifikat dokumentiert damit einen bestimmten Wissensstand. Als Platzhalter für eine Gruppe von Fähigkeitsstufen vereinfacht es die Statusdarstellung und –erkennung eines Spielers: Die Wahrnehmung eines bestimmten Zertifikats ist schneller als die Prüfung, ob ein Spieler eine äquivalente Menge von verschiedenen Fähigkeitsstufen erreicht hat.

Zertifikate können eine Leitfunktion für den Spieler besitzen: Wenn dieser ein bestimmtes Zertifikat anstrebt, werden auch die notwendigen Maßnahmen erkennbar, die er zur Erreichung des Zieles ergreifen muss: Es ist klar, aus welchen Themengebieten die zu beantwortenden Fragen stammen müssen, um die notwendigen Fähigkeitspunkte zu erreichen.

Grundlage für die Erteilung eines Zertifikates können nicht nur bestimmte Fähigkeitsstufen sein, sondern auch andere Zertifikate. Diese stehen jedoch selbst wieder für eine Menge von Fähigkeitsstufen, so dass letztendlich ein Zertifikat immer auf Fähigkeitsstufen zurückzuführen ist.

Ein Beispiel für die Steuerung von Spielern durch Zertifikate ist der *Certification Planner* des MMORPGs¹⁷ „EVE Online“. Abbildung 5 zeigt den Bildschirm: Auf der linken Seite gibt es den Baum der erwerbbaaren Zertifikate, von denen eines ausgewählt werden kann. Dieses wird dann auf der rechten Seite in einer Detailansicht dargestellt: Neben der Bedeutung des Zertifikats wird klar gezeigt, welche Fähigkeiten bereits erworben wurden und welche Fähigkeitsstufen für die Erteilung des Zertifikats noch fehlen.

¹⁷ **Massively Multiplayer Online Role Playing Game** (Mehrspieler-Internet-Rollenspiel)

Abbildung 5: EVE Online: Certification Planner¹⁸

3.7 Passender Schwierigkeitsgrad

Ein wesentliches Merkmal einer spielergerechten Aufgabe (hier: Frage) ist ein passender Schwierigkeitsgrad. Der Spieler soll weder durch die Aufgaben gelangweilt werden, noch soll er dermaßen überfordert werden, dass er ohne Hoffnung auf eine Lösung die Bemühungen frustriert einstellt. Um dieses Ziel zu erreichen, ist es einerseits nötig, jeder Frage einen Schwierigkeitsgrad zuzuordnen, andererseits jedem Spieler einen Fähigkeitsgrad zuzuweisen, d.h. eine Zahl, die ein Maß für die Problemlösungskompetenz des Spielers bezüglich in einem definierten Themengebiet ist. Diese beiden Maßzahlen werden dann benutzt, um geeignete Fragen für einen Spieler zu ermitteln¹⁹.

Als Maßzahl für die Fähigkeiten des Spielers wird der Einfachheit halber dessen Wert für die entsprechende Fähigkeit genutzt, hier Fähigkeitsgrad genannt. Es entfallen aufwändige Berechnungen.

Der Schwierigkeitsgrad einer Frage wird mit Hilfe verschiedener Daten festgelegt: Zunächst bestimmt der Fragersteller einen Schwierigkeitsgrad SG_1 nach eigener Einschätzung auf einer Skala zwischen 0 und 1. Dies führt dazu, dass die Frage an gemäß dieser Einschätzung geeignete Spieler ausgeliefert wird. Diese haben dann die Möglichkeit über die Feedbackfunktion ihre eigene Einschätzung zum Schwierigkeitsgrad dieser Frage abzugeben. Die Einschätzungen der Spieler werden - gewichtet durch den Status des jeweiligen Spielers – in die Berechnung des Schwierigkeitsgrades einbezogen. Es wird der gewichtete Durchschnitt SG_2 gebildet.

¹⁸ Snapshot verfügbar unter http://www.eve-wiki.net/images/b/bd/Certificate_planner.png (letzter Zugriff am 2.8.2012)

¹⁹ Dies gilt nicht für den Turniermodus. Dort wählt der Organisator einen festen Bereich für den Schwierigkeitsgrad der zu beantwortenden Fragen.

Eine weitere Einflussgröße ergibt sich aus der Korrektheit der Antworten: Es wird der Anteil der falschen Antworten bestimmt. Die Fähigkeitswerte der Spieler, die diese Frage beantwortet haben, werden aufsteigend sortiert. Als Schwierigkeitsgrad gilt der Wert, der an der Grenze zwischen falschen und richtigen Antworten liegt. Beispiel: Die Frage A trägt zur Fähigkeit B bei und wird zu 75% falsch beantwortet. Es wird nun in der sortierten Liste der Werte der Fähigkeit B der antwortenden Spieler der erste Wert gesucht, der gerade größer ist als 75% aller anderen Werte der Fähigkeit B. Dieser Wert wird auf das Intervall $[0,1]$ normiert. Zusammen mit dem Wert SG_2 wird ein gewichteter Durchschnittswert gebildet: der letztendliche Schwierigkeitsgrad SG_3 .

Einer Frage können mehrere Tags zugeordnet sein. Daher kann eine Frage auch unterschiedliche Schwierigkeitsgrade bezüglich mehrerer Fähigkeiten haben²⁰. Zur Auswahl der Fragen wird das folgende Vorgehen genutzt: Ziel ist es, den Fähigkeitsgrad des Spielers auf die höchste Stufe zu bringen. Daher wird bevorzugt der Fähigkeitsgrad verbessert, der den geringsten Abstand von der höchsten Stufe besitzt. Fähigkeitsgrade der höchsten Stufe werden nicht mehr gezielt weiter verbessert.

3.8 Fragenstapel

Der Fragenstapel ist eine Liste von Fragen, die der Spieler zu beantworten hat bzw. die er bereits beantwortet hat. Diese Liste ist zugänglich über den Haupt-Bildschirm von *Easy Gold* (s. Abbildung 6).



Abbildung 6: Schaltfläche zum Aufruf des Fragenstapels

Fragen erscheinen aus verschiedenen Gründen im Fragenstapel (Darstellung in Abbildung 7):

- **Zurückstellen:** Wird dem Spieler eine Frage gestellt und er kann sie nicht direkt beantworten, so gibt es für ihn die Möglichkeit, über die „Park“-Schaltfläche die Frage zu zurückzustellen. Dies bedeutet, dass er den Aufwand, den er aufbringen musste, bis ihm die Frage gestellt wurde, nicht verliert. Wurde eine Frage zurückgestellt, so erscheint sie im Fragenstapel und kann jederzeit beantwortet werden.
- **Eigenes Weiterleiten:** Leitet der Spieler eine Frage weiter (s. Kap. 4.4), so muss es für ihn eine Möglichkeit geben, sich die Antworten seiner Freunde anzuschauen und seine eigene Antwort zu geben. Dieses ist über den Fragenstapel möglich, zu dem eine Frage automatisch hinzugefügt wird im Falle einer Weiterleitung.

²⁰ Nach dem obigen Algorithmus können unterschiedliche Schwierigkeitsgrade einer Frage bezüglich verschiedener Fähigkeiten nur durch die unterschiedlichen Fähigkeitsgrade der Spieler zustande kommen, die sich durch die zufällige Fragenauswahl ergeben. Daher sind keine systematischen Abweichungen zu erwarten. Das realisierte System sollte beobachtet werden, ob solche systematischen Abweichungen entstehen oder ob sich die Schwierigkeitsgrade einer Frage auf längere Sicht in der Regel gleich entwickeln.

- **Weiterleitung:** Fragen, die an einen Spieler weitergeleitet werden, erscheinen in seinem Fragenstapel. Dort hat der Spieler die Möglichkeit zu einer Antwort.



Abbildung 7: Fragenstapel

Jede Frage des Fragenstapels hat einen Status. Der Status bestimmt die Aktionen, die der Spieler aus der Fragenliste heraus mit dieser Frage durchführen kann. Status und zugehörige Aktionen sind in Tabelle 7 beschrieben.

Status	Beschreibung	Mögliche Aktionen
forwarded	Der Spieler hat die Frage an andere Spieler weitergeleitet, keiner dieser Spieler hat die Frage bisher beantwortet.	Weiterleiten Beantworten Anschauen
parked	Der Spieler hat die Frage geparkt, damit er sie später beantworten kann.	Weiterleiten Beantworten
answeredPartly	Der Spieler hat die Frage weitergeleitet, einige der Adressaten haben die Frage schon beantwortet.	Weiterleiten Beantworten
ownAnswerNeeded	Der Spieler hat die Frage weitergeleitet, alle Adressaten haben die Frage beantwortet, es fehlt nur noch die Antwort des Spielers.	Weiterleiten Beantworten
gotForwarded	Diese Frage wurde an den Spieler weitergeleitet, er hat noch keine Antwort gegeben ebenso wie derjenige, der die Frage weitergeleitet hat.	Weiterleiten Beantworten
gotForwardedForwarder-Answered	Diese Frage wurde an den Spieler weitergeleitet, er hat noch keine Antwort gegeben. Der weiterleitende Spieler hat bereits eine Antwort gegeben, die aber nicht sichtbar ist, um den Spieler nicht zu be-	Weiterleiten Beantworten

	einflussen.	
gotForwardedAnd-Answered	Diese Frage wurde an den Spieler weitergeleitet und er hat bereits geantwortet. Der weiterleitende Spieler hat noch nicht geantwortet.	
rewardReady	Diese Frage wurde an den Spieler weitergeleitet. Sowohl er als auch der weiterleitende Spieler haben eine Antwort gegeben. Eine zusätzliche Belohnung steht für den Spieler bereit. Sobald er diese abholt, wird diese Frage aus dem Fragenstapel entfernt.	Belohnung abholen

Tabelle 7: Fragenstapel: Status

Aus dem Fragenstapel entfernt werden die Fragen durch das Geben einer Antwort. Hier gibt es eine Ausnahme: Beantwortet der Spieler eine Frage, die an ihn weitergeleitet wurde und die der weiterleitende Spieler noch nicht beantwortet hat, so kann noch nicht entschieden werden, ob die zusätzliche Belohnung, die bei richtiger Antwort des weiterleitenden Spielers an den antwortenden Spieler ausgeschüttet wird, diesem auch zusteht. Eine solche Frage hat den Zustand *gotForwardedAndAnswered*. Antwortet der weiterleitende Spieler richtig, so geht diese Frage in den Zustand *rewardReady* über. Holt der Spieler die Belohnung einer Frage mit diesem Zustand ab, so verschwindet sie aus der Liste des Fragenstapels. Bei einer falschen Antwort des weiterleitenden Spielers wird die Frage schon vorher aus dem Fragenstapel entfernt.

4 Interaktionen

4.1 Beantwortung einer Frage

Bei der Beantwortung einer Frage muss der Spieler eine der vorgeschlagenen Antworten auswählen. Zusätzlich kann er eine Sicherheit angeben, mit der er diese Antwort gibt²¹. Dadurch soll die Reflektion über die Antwort gefördert werden (Gardner-Medwin, 2006). Eine richtige Einschätzung der möglichen Richtigkeit der eigenen Antwort wird als belohnungswürdig angesehen. Eine falsche Antwort wird auch dann belohnt, wenn der Spieler einen relativ niedrigen Wert für die Sicherheit angegeben hat.

In Abbildung 2 (S. 5) sind die möglichen Aktionen zur Beantwortung einer Frage zu sehen:

- **Answer:** Die Schaltfläche ist aktiv, sobald eine Antwort ausgewählt oder eingegeben wurde. Im Turnier- und Battle-Modus wird sie mit fortlaufender Zeit kleiner. Ist die Zeit abgelaufen (Standardwert ist derzeit 30 Sekunden), wird sie ersetzt durch eine *Show Answer*-Schaltfläche: Diese ermöglicht es dem Spieler gegen Zahlung von einigen (im Prototypen sind es zurzeit 30) Coins die richtige Antwort einzusehen. Die normale Rückmeldung bei einer falschen Antwort beinhaltet nicht die Anzeige der korrekten Antwort. Dies soll den Spieler dazu veranlassen, über eine möglicherweise richtige Antwort nachzudenken anstatt sich die angezeigte richtige Antwort zu merken.
- **Forward:** Mit Hilfe der *Forward*-Schaltfläche wird die Frage an Freunde weitergeleitet. Die Frage selbst wird in den Fragenstapel überführt und kann nach den Antworten der Freunde wieder aktiviert und beantwortet werden. In Abbildung 8 wird die Weiterleitung einer Frage gezeigt: Dazu wird ein Dialog des genutzten sozialen Netzwerks (*Face-*

²¹ Die Bewertung der Sicherheit erfolgt auf einer Skala nach dem Vorbild der Likert-Skalen (Likert, 1932): z.B. Sehr sicher – sicher – zuversichtlich – unsicher – sehr unsicher

book) eingesetzt. Die Frage kann je nach Auswahl des Spielers an alle Facebook-Freunde oder an die Spielbenutzer (hier: *BuildVille*) weitergeleitet werden. Auswirkungen einer Weiterleitung sind zum einen ein Eintrag auf dem schwarzen Brett der Spieler, an die weitergeleitet wurde sowie das Erscheinen der Fragen in deren Fragenstapel.

- **Park:** Die *Park*-Schaltfläche sorgt für eine Speicherung der Frage im Fragenstapel. Damit wird der Aufwand des Findens der Schatztruhe gesichert, die Frage muss nicht noch einmal freigeschaufelt werden.
- **Skip:** Mit der Schaltfläche *Skip* zeigt der Spieler an, dass ihn die Frage nicht weiter reizt, er übergeht sie. Das hat die Konsequenz, dass die Frage nicht wieder angezeigt wird und dass bei der weiteren Ermittlung von Fragen die Tags, die mit dieser Frage verbunden sind, weniger stark gewichtet werden (s. Kap. 3.4.3). Werden mehrere Fragen desselben Tags übergangen, so werden schließlich keine Fragen des Tags mehr ausgewählt.

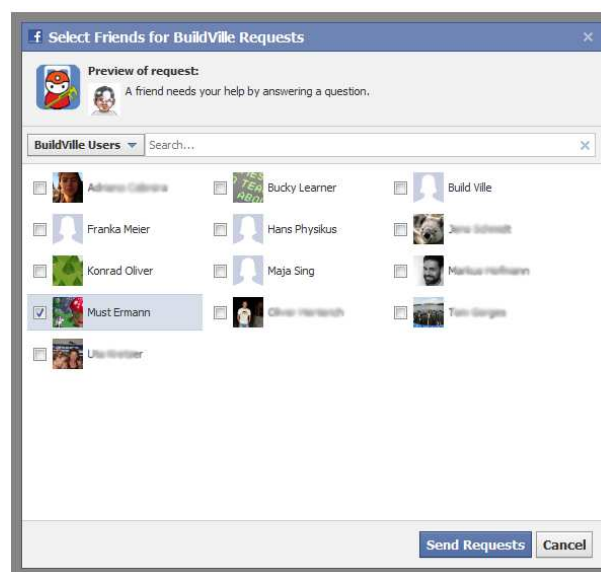


Abbildung 8: Weiterleitung einer Frage mit Hilfe eines Facebook-Dialogs

4.2 Erstellen von Fragen

Die Lerntheorie des Konstruktivismus (Papert, 1980) misst der Herstellung von Artefakten eine wichtige Bedeutung im Wissensaufbau bei. Das Erstellen von Multiple-Choice-Fragen mit der Auswahl von falschen Antworten kann als eine solche Artefakt-Herstellung gesehen werden. Daher wird dem Spieler die Möglichkeit geboten, eigene Fragen in das System einzustellen. Für die Konstruktion einer Frage müssen neben dem Fragentext mehrere Antwortalternativen gefunden werden, von denen eine als richtig markiert ist. Zusätzlich sollten verschiedene Schlagworte („Tags“) angegeben werden, die die Frage kategorisieren.

Bei der Erstellung einer Frage werden alternative Fragen angezeigt, die schon im System vorhanden sind. So werden ungewollte Mehrfacheingaben²² vermieden. Vorbild für diese Funktionalität ist die Software von Stackoverflow.com (2012). In Abbildung 9 ist eine Bildschirmkopie der Eingabe einer Frage zu sehen: Während die Frage eingegeben wird (1), sucht das System mit Hilfe der eingegebenen Zeichen ähnliche, schon existierende Fragen und präsentiert sie (2).

²² Es kann sinnvoll sein, dieselbe Frage mit unterschiedlichen Alternativantworten zu stellen, um den Fokus von der Auswahl der richtigen Antwort durch das Ausschlussverfahren auf das tatsächliche Wissen der richtigen Antwort zu verschieben.

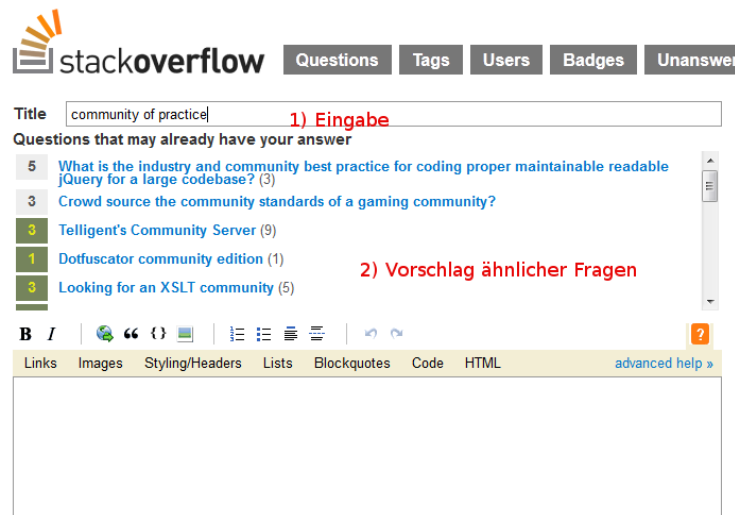


Abbildung 9 Stackoverflow.com: Erstellen einer neuen Frage

Die Motivation für den Spieler, Fragen zu erstellen, wird auf verschiedene Weisen gefördert:

- **Tantiemen:** Für die Beantwortung der von ihm erstellten Fragen durch andere Spieler erhält er Coins. Dies kann zu einem stetigen Zuwachs an Coins führen ohne weiteres Zutun des Spielers – ein Anreiz zur Erstellung von Fragen.
- **Fähigkeits-Punkte:** Stellen sich vom Spieler gestellte Fragen als sinnvoll heraus, so werden dem Spieler Fähigkeitspunkte erteilt, d.h. genauso wie durch das erfolgreiche Beantworten von Fragen erhält er für das Stellen von Fragen einen Zuwachs an virtuellen Fähigkeiten.
- **Status:** Die Anzahl der vom Spieler gestellten Fragen sowie die zugehörige Anzahl der Antworten auf diese Fragen werden in seinen Statusdaten angezeigt. Zudem gibt es eine Rangliste bezüglich dieser Merkmale.
- **Wettbewerb:** Es gibt monatlich einen Wettbewerb, in dem die am besten bewertete Frage gekürt wird. Die Bewertung erfolgt im Rahmen des Feedbacks.

Der Spieler hat die Möglichkeit, auf die von ihm erstellen Fragen in einer Liste zuzugreifen (s. Kap. 3.3). Über diese Liste kommt er in den Edit-Modus für eine Frage. Die Frage lässt sich ändern, aber auch deaktivieren. Zusätzlich sieht der Spieler die Statistiken.

4.3 Rückmeldung auf Fragen

Nach Beantwortung einer Frage hat der Spieler die Möglichkeit, ein Feedback auf die Frage zu geben. Dazu gehören verschiedene Informationen, die aber allesamt und jede für sich freiwillig gegeben werden. Es ist nicht verpflichtend, eine bestimmte Rückmeldung zu geben. Das Feedback darf nicht das Spiel behindern:

Information	Beschreibung
Like	Wenn der Spieler die Frage für sinnvoll hält, dann drückt er eine <i>Like</i> -Schaltfläche.
Dislike	Hält der Spieler die Frage für fehlerhaft oder für sinnlos, kann er die <i>Dislike</i> -Schaltfläche betätigen.
Kommentar	In einem Kommentar hat der Spieler die Möglichkeit, Freitext einzugeben. Diese Freitexte werden angezeigt, in einer Detail-Ansicht der Frage. Sie werden in das Forum eingestellt.
Identifikator	Jede Frage hat einen Identifikator. Mit Hilfe dieses Identifikators soll es dem

	Spieler möglich sein, die Frage in der Fragenliste aufzufinden und evtl. an einer Diskussion teilzunehmen, Fragen zu stellen und Unklarheiten zu beseitigen.
Tags	Es können zusätzliche Tags vergeben werden, die die fachliche Einordnung der Frage vergrößern.
Schwierigkeitsgrad	Wert zwischen 0 und 1, der den fachlichen Schwierigkeitsgrad der Frage angibt (0: sehr einfach, 1: äußerst schwer).
Darstellungsgenauigkeit	Wert zwischen 0 und 1, der angibt, wie sorgfältig und verständlich die Frage erstellt wurde. Schlechte Bewertungen in diesem Punkt machen eine Überarbeitung der Frage notwendig.

Tabelle 8: Feedback-Informationen

Die Feedback-Informationen werden vom System zur Bewertung der Frage benutzt. Gibt es einen zu hohen Anteil an „Dislikes“, wird die Frage nicht mehr weiter an reguläre Spieler ausgeliefert. Stattdessen erscheint die Frage auf einer Liste der problematischen Fragen, die von Spielern mit hohem Status eingesehen werden kann (s. Kap. 2.4). Diese können die Frage sperren, korrigieren oder an zusätzliche Fachexperten weiterleiten, die dann gegebenenfalls die Frage bearbeiten können.

4.4 Weiterleitung von Fragen

Kennt der Spieler die Antwort auf eine ihm gestellte Frage nicht, so hat er die Möglichkeit, diese an alle oder ausgewählte Freunde des zugrundeliegenden sozialen Netzwerks weiterzuleiten. Die Weiterleitung erfolgt in Form eines Eintrags im Nachrichten-Strom („News-Feed“) (s. Abbildung 10) des sozialen Netzwerkes und durch Erscheinen der Frage im Stapel der zu beantwortenden Fragen der anderen Spieler (s. Kap. 3.8 und Abbildung 8 auf S. 19). Durch das Anklicken des Nachrichtenstrom-Eintrags gelangt der gefragte Spieler zu der gestellten Frage und kann diese beantworten, ist dazu aber nicht verpflichtet. Vor einer Antwort kann er sie selbst seinerseits weiterleiten und die Ergebnisse seiner Freunde abwarten. Für den fragenden Spieler wird die Frage in seinem Fragenstapel sichtbar (s. Abbildung 7 auf S. 17). Er kann dort jederzeit die Antworten seiner Freunde einsehen und gegebenenfalls auf diesen basierend seine eigene Antwort geben.

**Abbildung 10: BuildVille: Anwendungsanfrage in Facebook²³**

Die Beantwortung einer Frage, die an den Spieler weitergeleitet wurde, erbringt ihn genauso eine Belohnung wie die Beantwortung einer Frage, die ihm selbst durch das Öffnen einer Schatz-

²³ Der aktuelle Prototyp gibt die Anfrage nicht im Nachrichtenstrom aus, sondern nutzt dazu den Mechanismus „Anwendungsanfrage“ in Facebook. Das ist eine Methode, die Spieler persönlich und direkt außerhalb des Spiels, aber innerhalb des sozialen Netzwerkes, mit dem Ziel anzusprechen, ihn wieder zum Spielen zu motivieren.

kiste präsentiert wurde. Zusätzlich erhält er ergebnisabhängig noch eine weitere Belohnung: Hat der Spieler, der die Frage an ihn weitergeleitet hat, die richtige Antwort gegeben, so erhält er dessen Belohnung ebenfalls – unabhängig von der Richtigkeit seiner Antwort – nur abhängig vom Geben einer Antwort.

In die Präsentation der Frage eingebettet ist ein Verweis auf die Diskussion der Frage im zugehörige Forum und eine Kurzansicht der letzten dort stattfindenden Aktionen bezüglich dieser Frage. Das soll den Spieler – sofern er sich seiner Antwort nicht ganz sicher ist – neugierig machen und zu einem Blick in das Forum verleiten, in dem er dann unter Umständen selbst einen Diskussionsbeitrag leisten kann.

5 Technische Systemelemente

5.1 Forum

Das System wird durch ein Forum unterstützt. In diesem Forum können die Mitspieler die Fragen und verbundene Themen diskutieren und durch gegenseitige Rückmeldungen zu einer gemeinsamen Wissensgenerierung beitragen.

Jede Frage trägt einen Identifikator. Mit dessen Hilfe ist es möglich, im Forum den passenden Eintrag zu einer Frage zu finden. Damit nicht vor der Antwort auf eine Frage das Forum nach der richtigen Antwort durchsucht wird und diese kritiklos übernommen wird, wird der Identifikator erst in der Reaktion auf die Antwort des Spielers gezeigt, nicht aber in der Frage selbst. Abbildung 11 zeigt den Identifikator („Q7“) in der Rückmeldung auf eine falsch beantwortete Frage.



Abbildung 11: Falsch beantwortete Frage: Identifikator in der Rückmeldung

Für die technische Umsetzung des Forums wird die Software von Stackoverflow.com (2012) vorgeschlagen. Diese Software bietet mehrere Merkmale, die sie von anderer Forensoftware²⁴ abhebt:

²⁴ Streng genommen ist die Software von Stackoverflow kein Forum, sondern eine „community driven Q&A site“ (Q&A: Question and Answer) (Stackexchange.com, 2012). Zentrales Strukturelement ist die Frage. Durch die Möglichkeit, jeden Eintrag (also entweder eine Frage oder eine Antwort) zu kommentieren, können sich auch hier Diskussionen entwickeln.

- **Gamification:** Aktivität des Benutzers wird registriert und belohnt, die Vorteile von Gamification (z.B. hohe Motivation) können genutzt werden.
- **Statusabhängige Rechte:** Die Software ist selbstorganisierend: Der Benutzer erarbeitet sich seinen Status durch Aktionen, mit steigendem Status erhält er auch zusätzliche Rechte.
- **Zugriff durch Webbrowser:** Auf das Forum kann mit Hilfe des Webbrowsers zugegriffen werden, d.h. ebenso wie beim Spiel muss keine Software auf dem Rechner installiert werden, das Forum ist von jedem Rechner mit Internetverbindung erreichbar.
- **Fortschrittliche, unterstützende Oberfläche:** Obwohl die Oberfläche durch den Webbrowser bereitgestellt wird, bietet sie hohen Benutzerkomfort. Beispielsweise werden beim Anlegen eines neuen Themas vergleichbare Themen während des Tippens gesucht und eingeblendet, so dass der Benutzer entscheiden kann, ob seine Frage bereits vorher ausreichend beantwortet wurde.

Da die Software von Stackoverflow.com (2012) nicht frei verfügbar ist, wurde für den Prototypen ein Forum mit Hilfe der einen ähnlichen Leistungsumfang bietenden Software von Shapado (2012) eingerichtet²⁵. Abbildung 12 zeigt eine Bildschirmkopie einer Frage.



Abbildung 12: Shapado: Bildschirmkopie

5.2 Nachrichtenstrom und Schwarzes Brett

Soziale Netzwerke bieten gewöhnlich die Möglichkeit, Nachrichten an andere Mitglieder zu verschicken („Messaging“) sowie Beiträge auf dem eigenen schwarzen Brett oder dem eines anderen Mitgliedes zu platzieren („Wall Posts“). *Easy Gold* nutzt diese Optionen, um das Spiel zwischen Freunden zu unterstützen:

- **Beiträge auf dem eigenen Schwarzen Brett:** Auf dem eigenen schwarzen Brett werden Statusmeldungen des Spiels platziert. Dazu zählen Meldungen über eigene Erfolge und Aufrufe zur Mithilfe. Meldungen über eigene Erfolge sind gewöhnlich mit einer Beloh-

²⁵ Das Forum ist zu finden unter <http://buildville.shapado.com/>.

nung für den reagierenden Mitspieler verbunden, d.h. sie enthalten eine Verknüpfung („Link“), dessen Anklicken belohnt wird und gleichzeitig in das Spiel führt. Hierdurch sollen Freunde des sozialen Netzwerks in das Spiel hineingezogen werden. Beispiel für eine solche Mitteilung über einen eigenen Erfolg ist das Erreichen einer bestimmten Fähigkeitsstufe. Der Klick auf die zugehörige Verknüpfung wird belohnt, beispielsweise mit Coins.

Aufrufe zur Mithilfe bieten dem Spieler eine Aktion an, mit der er seinen Mitspieler unterstützen kann. Sie enthalten ebenfalls eine Verknüpfung in das Spiel und werden gleichfalls belohnt. Beispiel hier ist der allgemeine Aufruf zur Beantwortung einer Frage. Die Belohnung besteht bei richtiger Antwort aus Coins und Fähigkeitspunkten. Auch bei einer falschen Antwort bleibt immer noch das Fortkommen im Status: Die Anzahl der beantworteten Fragen wird hochgezählt.

- **Beiträge auf dem Schwarzen Brett eines Mitspielers:** Auf dem schwarzen Brett eines Mitspielers wird ein Eintrag nur dann platziert, wenn dieser direkt angesprochen werden soll. Das ist zum Beispiel der Fall, wenn eine Frage an ihn weitergeleitet wird. Durch eine Verknüpfung in der Mitteilung wird der Spieler zu der Frage geführt, so dass er sie sogleich lesen und beantworten kann.

Aus einem spielgenerierten Beitrag auf einem schwarzen Brett können sich Diskussionen und Interaktionen zwischen den Spielern entwickeln. Diese sind Basis einer gemeinschaftlichen Wissenskonstruktion im Sinne einer *Community of Practice* (Lave & Wenger, 1991).

Der Nachrichtenstrom ergibt sich aus den Inhalten der Schwarzen Bretter aller im sozialen Netzwerk mit dem Spieler verknüpften Personen. Verschiedene Social Network Services bieten unterschiedliche Varianten des schwarzen Bretts bzw. der Ansprache aller Mitspieler oder eines einzelnen Mitspielers. Der hier gewollte Nutzen eines solchen Mechanismus ist die Kontaktaufnahme mit potentiellen oder ehemaligen Spielern außerhalb des Spiels auf der Plattform des sozialen Netzwerks mit dem Ziel, sie wieder zum Spielen zu motivieren. In Facebook ist eine Variante die der Anwendungsanfrage („Application Request“, (Facebook, 2012)): Mit ihr können auswählbare Spieler direkt aus einer Anwendung (als eine solche ist das Spiel realisiert) erreicht werden.

Das Erstellen eines Beitrages auf einem Schwarzen Brett muss aktiv durch den Spieler bestätigt werden. Damit soll gewährleistet werden, dass er das Spiel auch spielen kann, ohne für andere Mitspieler sichtbare Spuren von Aktivitäten zu hinterlassen. Somit werden auch zurückhaltende Spieler nicht vom Spielen abgehalten.

5.3 E-Mail

E-Mails werden dann eingesetzt, wenn sich Spieler schon längere Zeit im Spiel bzw. dem zugrundeliegenden Sozialen Netzwerk nicht mehr angemeldet haben. Damit sind die Kontaktmöglichkeiten über das soziale Netzwerk und das Spiel nicht mehr gegeben. Als Ersatz erinnern E-Mails den Spieler an mögliche Aktivitäten in *Easy Gold* und versuchen ihn, für das Spiel zu reaktivieren. Im Folgenden werden einige Anlässe aufgezählt:

- **Fällige Fragen:** Wenn der Spieler im Lern-Modus gespielt hat, werden Fragen in bestimmten Zeitabständen fällig. Häufen sich diese Fragen bzw. sind Fragen auf den letz-

ten Stufen zu beantworten, so sind das besondere Gelegenheiten, über eine E-Mail mit dem Spieler zu kommunizieren.

- **Neue Fragen zu Spezial-Themengebieten des Spielers:** Möglich ist, dass der Spieler nur an bestimmten Themengebieten interessiert ist und zu diesen Themengebieten schon alle Fragen beantwortet hat. In einem solchen Fall kann der Spieler mit E-Mails über neue Fragen zu seinem Themengebieten informiert werden. Auch um ein bestimmtes Themengebiet im Spiel zu erweitern, kann eine E-Mail-Aktion gestartet werden: Die Spezialisten dieses speziellen Themengebietes werden eingeladen, neue Fragen zu formulieren.
- **Angebote zu Turnieren:** Einladungen durch andere Spieler zu Turnieren (s. Kap. 3.1.3) können ebenfalls an den erwünschten Spieler per E-Mail weitergeleitet werden, sofern sich dieser seit längerer Zeit nicht eingeloggt hat bzw. die Einladung nicht abgelehnt hat.
- **Weitergeleitete Fragen:** Weitergeleitete Fragen sind genauso eine Aufforderung, sich wieder am Spiel zu beteiligen. Ebenso wie bei den Einladungen zu den Turnieren könnte der Spieler durch den Verweis auf die persönliche Einladung eines anderen Spielers wieder motiviert werden, an dem Spiel teilzunehmen.

In den E-Mails sind immer Verknüpfungen zu den entsprechenden Spielaktionen enthalten. Das ermöglicht den Empfängern einen bequemen Eintritt in das Spiel.

E-Mails sollten fein dosiert eingesetzt werden, damit sie nicht als Belästigung und Spam empfunden werden. Dann nämlich ist die Gefahr groß, dass der Spieler das Senden von E-Mails im Spiel deaktiviert und somit diese wertvolle Kontaktmöglichkeit beendet.

6 Anforderungsabgleich

Die in Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.** dargestellten von David Nicol genutzten Regeln guten Feedbacks für Multiple-Choice-Tests sollen an dieser Stelle mit dem vorgestellten Design des Spiels abgeglichen werden.

6.1 Definition einer guten Leistung

Eine gute Leistung ist deutlich definiert über mehrere Aspekte, von denen im Folgenden einige genannt werden:

- Jede Frage hat eine exakt festgelegte Antwort.
- Zertifikate geben ein klares Ziel vor.
- Der Lernzyklus nach dem Prinzip der Lernkartei ist ebenfalls ein klares Ziel für jede Frage.
- Battle- und Turnier-Modus definieren gleichfalls eine gute Leistung: Im Turnier-Modus ist es der Sieg bzw. eine gute Platzierung und im Battle-Modus das Freiräumen des Schatzfeldes in der vorgegebenen Zeit.

6.2 Entwicklung von Selbsteinschätzung und Reflektionen

Selbsteinschätzung ist dann gefordert, wenn bei der Beantwortung einer Frage eine Sicherheit angegeben werden soll, mit der die Frage beantwortet werden kann. Auch die Wahl einer Zeitspanne im Battle-Modus, in der die Fragen beantwortet werden müssen, ist mit einer Selbsteinschätzung verbunden: Der Spieler setzt sich ein Ziel, von dem er hofft, es erreichen zu können.

Reflektion wird u.a. auch gefördert, wenn bei einer falsch beantworteten Frage die richtige Antwort nicht sofort preisgegeben wird, sondern erst auf Nachfrage und dann gegen Zahlung einer Art Schutzgebühr: Dem Spieler wird vermittelt, dass die richtige Antwort einen Wert besitzt. Auch wenn die Schutzgebühr nicht groß ist, wird er gewöhnlich darüber nachdenken, ob er die richtige Antwort nicht selbst erkennen kann (z. B. wenn er zwischen zwei Antworten geschwankt hat).

6.3 Informationen über den Lernprozess

Der Status des Spielers ist jederzeit sichtbar: es ist erkenntlich, welche Fähigkeiten er aufgebaut hat, welche Fragen er beantwortet hat, welche Fragen auf welcher Stufe der Lernkartei stehen, wie viele Fragen er beantwortet hat. Selbst bekommt er bei jeder kleinsten Einheit des Lernens – dem Beantworten einer Frage – eine Rückmeldung über seine Kompetenzen: Er konnte die Frage richtig beantworten oder eben nicht.

6.4 Dialog

Der Dialog zwischen den Spielern und über das Lernen wird mehrfach gefördert:

- Das Forum ist ein Ort des Dialogs: Spieler stellen Fragen und sie bekommen Antworten – oft mit Kommentaren. Eine Diskussion entwickelt sich: Es wird gemeinsam Wissen zusammengetragen und damit eine Lösung konstruiert. Eine solche gemeinschaftliche Wissenserzeugung lässt sich auch in Foren anderer Spiele beobachten (Steinkuehler & Duncan, 2008; Söbke, Corredor & Kornadt, 2011).
- Die Feedback-Funktion liefert ebenfalls Rückmeldungen der Spieler auf die Fragen: Alle Rückmeldungen sind sowohl über die Frage zugänglich als auch über die Frage im Forum. Damit bieten sie Anreiz zum Start einer Diskussion oder zu einer Beteiligung an einer bereits laufenden Diskussion.
- Die Weiterleitung von Fragen kann auch zu privater Kommunikation zwischen den beiden beteiligten Spielern²⁶ führen, mit Hilfe der verschiedenen technischen Kommunikationsmöglichkeiten wie Chat, Messaging oder Video-Konferenz des einbettenden sozialen Netzwerkes.

6.5 Motivierendes Feedback

Zumindest für einen Teil der Spieler stellen erreichbarer Status und dauernde Belohnungen eine willkommene Motivation dar – auch ersichtlich aus dem anhaltenden Erfolg des Social Gaming (Söbke, Bröker, & Kornadt, 2012). Motivation ergibt sich auch aus den gemeinsamen Aktionen innerhalb einer Gruppe – im Turnier-Modus sowie im Team-Match-Modus können sich interessante Diskussionen zwischen den Spielern entwickeln: Vor, während und nach dem Spiel.

Als Einschränkung ist das Zurückstufen einer Frage in der Lernkartei zu sehen: Wenn der Spieler sie falsch beantwortet hat, so startet der Lernzyklus von der ersten Kategorie an neu – das ist erst einmal kein positives Feedback für den Spieler.

²⁶ Wird eine Frage weitergeleitet, so sind daran meistens mehr als zwei Spieler beteiligt. An dieser Stelle liegt jedoch der Schwerpunkt auf der zweiseitigen Kommunikation zweier Spieler – dem Sender und dem Empfänger einer weitergeleiteten Frage.

6.6 Lernziel

Mit Hilfe von Zertifikaten können Lernziele klar definiert werden. Ein Zertifikat definiert die Fähigkeiten, die ein Spieler lernen soll: Er weiß damit, welche Fragen er zu beantworten hat.

6.7 Feedbackschleife

Durch die Feedback-Funktion ist es dem Spieler möglich, Fragen zu kommentieren. Fragenersteller können damit Anmerkungen der Spieler in die Frage einfließen lassen und somit schrittweise für eine Verbesserung der Fragen sorgen.

Durch die umfangreichen statistischen Daten, die zu einer Frage gesammelt werden, ist es leicht möglich, unpassende Fragen zu identifizieren bzw. systematische Probleme zu erkennen (z.B. zur Spielbalance). Derartiges Feedback kann ebenfalls zu einer Gesamtverbesserung des Systems genutzt werden.

7 Schlussbetrachtung

Das vorgestellte System ist ein Ansatz, die Aspekte Social Gaming, Game Based Learning und Multiple-Choice-Fragen miteinander zu einer faszinierenden Lernanwendung zu verknüpfen. Die Anwendung ist in Teilen bereits realisiert, die Machbarkeit ist damit erwiesen. Erste Tests mit Studenten haben Ansätze zur Optimierung ergeben, die auch in dieses Konzept eingeflossen sind, insbesondere im Bereich Spielspaß und technischer Zuverlässigkeit.

Anhang G: Plattformbeschreibung

Inhaltsverzeichnis

1 Einleitung.....	3
2 Ausgewählte Aspekte der Implementierung.....	3
2.1 Berechnungsalgorithmen für Parameter	3
2.1.1 Anforderungen	3
2.1.2 Einflussgrößen für den Wert eines Parameters	3
2.2 Benutzung von Entwurfsmustern.....	7
2.2.1 Observer Pattern	8
2.2.2 Singleton Pattern.....	9
2.2.3 Visitor Pattern	9
2.2.4 Weitere Entwurfsmuster	11
2.3 Designaspekte zur Unterstützung von Modularität und Erweiterbarkeit.....	12
2.3.1 Darstellungsschema	12
2.3.2 Parameter Contribution	13
2.3.3 Dynamische Erweiterung des Datenmodells.....	15
2.3.4 Szenarioänderungsbasierte Systemereignisse	18
2.3.5 Simulationselement bezogene Änderungsdialoge	20
2.3.6 Verbindungsmodule	25
2.4 Designaspekte zur Unterstützung von Simulation	27
2.4.1 Automatisierte Erzeugung von individualisierten Simulationselementen.....	27
2.4.2 Konfiguration im Szenariomodul.....	35
2.5 Numerische Aspekte.....	37
2.5.1 Problemstellung	37
2.5.2 Qualitätseigenschaften von numerischen Algorithmen.....	40
2.5.3 Lösungsmöglichkeiten	40
2.5.4 Schlussfolgerungen.....	43
3 Oberfläche des Szenario-Editors	44
3.1 Beschreibungsinhalte	44
3.2 Perspective	45
3.3 Views der Kern-Module.....	46
3.3.1 Scenario Outline	46
3.3.2 Navigator	49
3.3.3 Parameter Types.....	50
3.3.4 System Events.....	51
3.3.5 Parameter Monitor.....	52
3.3.6 Parameter Comparison	53
3.3.7 Action Log	55
3.4 Views des Skill-Manager-Moduls	55
3.4.1 Skill Tree Admin	56
3.4.2 Skill Tree	57
3.5 Views des Moduls Missionsmanager	58

3.5.1 Mission Administration	58
3.5.2 Mission	58
4 Module	59
4.1 Namenskonventionen	59
4.2 Liste der Module	60
4.3 Statistik	60
4.4 Einordnung der Module	61
4.4.1 Kern-Module	61
4.4.2 Fachliche Module	62
4.4.3 Client-Server-Unterstützung	63
5 Extension Points	64
5.1 Beschreibungsschema	64
5.2 simulationElementParameter	65
5.3 parameterSource	66
5.4 weightedAverageContributor	68
5.5 systemEvent	69
5.6 scenarioFactory	70
5.7 simulationElement	72
5.8 listenerSupplier	72
5.9 mission	73
5.10 badge	74
5.11 skill	75
6 Quellcode und Anwendungen	76
7 Abbildungsverzeichnis	77
8 Tabellenverzeichnis	78

1 Einleitung

In diesem Dokument wird die technische Dokumentation zu erstellten Artefakten zusammengefasst. Zusätzlich werden Verweise auf die Online verfügbaren Arbeitsergebnisse gegeben. Im folgenden Kapitel werden jedoch zunächst getroffene Designentscheidungen in ihren Grundlagen sowie Auswirkungen beschrieben.

2 Ausgewählte Aspekte der Implementierung

In diesem Kapitel werden Implementierungsdetails dargestellt, die spezielle Anforderungen der Spielplattform erfüllen und zum Erkenntnisgewinn der Arbeit beitragen.

2.1 Berechnungsalgorithmen für Parameter

2.1.1 Anforderungen

Einer der Grundsätze des Designs der Plattform ist Modularität, d.h. die Plattform besteht aus vielen kleinen, in sich abgeschlossenen Bausteinen. Jeder dieser Bausteine definiert exakt die Voraussetzungen, unter denen er aktiviert werden kann. Je schmaler die Voraussetzungen eines Modules ausfallen, desto vielseitiger einsetzbar ist es. Gleichzeitig soll die erforderliche Rechenleistung für den Betrieb der Plattform möglichst gering sein, da viele Szenarien verschiedener Spieler parallel ausführbar sein sollen bzw. geringere Ansprüche an Hardware und Software die Verbreitung der Plattform erhöhen¹. Hieraus ergeben sich die Anforderungen an die Berechnungsalgorithmen der Parameter: Die Berechnung sollte einfach und nicht aufwändig sein. Es sollten wenige Abhängigkeiten zu anderen Parametern und Modulen bestehen, um die Einsetzbarkeit zu erhöhen.

2.1.2 Einflussgrößen für den Wert eines Parameters

Eine Grundlage zum Entwurf von Richtlinien, wie Berechnungsalgorithmen für Parameter entworfen werden müssen, damit die obigen Anforderungen bestmöglich erfüllt werden können, ist eine Analyse der potentiellen Einflussgrößen auf den Wert von Parametern:

- **Konstante:** Der Parameter kann eine Konstante sein, d.h. er wird nicht berechnet, er ändert sich nicht über die Zeit. Beispiel ist das Schalldämmmaß eines Fensters oder einer Wand (`SoundReductionIndex`). Dies ist ein bauteilbedingter, zeitunabhängiger Wert.
- **Benutzerdefinierter Wert:** Bestimmte Parameter können durch den Spieler gesetzt werden. Auch hier gibt es keine Berechnungsvorschrift für den Parameter. Beispiel ist die Miete einer Wohnung (`Rent`). Sie kann frei durch den Spieler gewählt werden.
- **Systemdefinierter Wert:** Systemdefinierte Werte sind solche, die durch die Ausführung von Prozessen gesetzt werden, indirekt ausgelöst durch Spieleraktionen oder Systemereignisse. Diese Werte werden nicht kontinuierlich berechnet, sondern behalten ihren Wert im Sinne einer Zustandsgröße.
Beispiel: Der Parameter Kassenstand (`Cash`) des Gebäudes ändert immer dann seinen Wert, wenn Ein- und Auszahlungen erfolgen, beispielsweise bei Mietzahlung oder Brennstoffkauf.
- **Werte anderer Parameter:** Der Wert eines Parameters kann aus den Werten anderer Parameter bestimmt werden.

¹ Die Möglichkeit der Auslagerung von rechenleistungsintensiven Algorithmen (z.B. auf die Grafikkarte) wird in einer parallelen Dissertation von Christiane Hadlich (Hadlich, 2013) untersucht.

Beispiel 1: Der Schallpegel in einer Wohnung kann mit Hilfe einer Formel berechnet werden, in der die Schallpegel anderer Schallquellen - als Parameter abgebildet - eingehen.

Beispiel 2: Die Mieterzufriedenheit ist eine Kombination der aspektspezifischen Mieterzufriedenheiten, d.h. beispielsweise der Zufriedenheit mit dem Schallschutz oder der Zufriedenheit mit dem Wärmeschutz.

- **Zeit:** Der Zeit kommt eine hervorgehobene Rolle in der Berechnung des Wertes von Parametern zu. Ein einfaches Beispiel ist die zu zahlende Miete, die sich aus der Miete pro Monat und der Dauer der Wohnungsbenutzung ergibt.
- **Frühere Werte des Parameters:** Werte von Parameter, die Mengengrößen beschreiben, werden gewöhnlich unter Einbeziehung von früheren Werten dieses Parameters berechnet.
Beispiel: Der Inhalt des Heizöltanks wird berechnet aus dem vorherigen Inhalt abzüglich des Verbrauchs im Simulationsintervall.

Zusammenfassend lässt sich feststellen, dass die meisten Einflussfaktoren auf Parameterwerte relativ einfach zu handhaben sind. Komplexer stellt sich in einigen Fällen der Einfluss der Zeit dar, wie im folgenden Kapitel dargestellt wird.

a) Zeit als Einflussgröße für Parameterwerte

In der Spielplattform sollen physikalische Systeme simuliert werden. Die Simulation erfolgt kontinuierlich, d.h. gewöhnlich müssen Differentialgleichungen gelöst werden, um Parameterwerte ermitteln zu können. Das ist in der Regel nur numerisch möglich, da für viele Differentialgleichungen keine analytischen Lösungen gefunden werden können.

b) Parameterberechnungsalgorithmus

Parameter, die berechnet werden, besitzen einen Parameterberechnungsalgorithmus. Der Berechnungsalgorithmus wird während des Simulationslaufs periodisch aufgerufen. Durch die Trennung zwischen Parameter und Berechnungsalgorithmus wird die Möglichkeit der Wiederverwendung für die Parameter erhöht: Bei der Definition eines Szenarios kann einem Simulationselement ein Parameter zugeordnet werden. Normalerweise hat dieser Parameter einen Standard-Parameterberechnungsalgorithmus. In Szenarien, in denen der Parameter in abweichender Weise berechnet werden soll, wird bei der Zuordnung von Simulationselement und Parameter zusätzlich ein anderer Parameterberechnungsalgorithmus angegeben.

Beispiel: Zuordnung unterschiedlicher Parameterberechnungsalgorithmen zu einem Parameter

Der Parameter *Raumtemperatur* (`HeatLevel`) beschreibt die Temperatur in einem Raum. Es ist denkbar, dass der Wert dieses Parameters nach alternativen, in ihrer Genauigkeit voneinander abweichenden Algorithmen berechnet werden kann, die auch von unterschiedlichen zu berücksichtigenden Phänomenen ausgehen. In verschiedenen Szenarien kann dann der für das jeweilige Szenario passende Parameterberechnungsalgorithmus konfiguriert werden.

Beispiel: Austauschbare Parameterberechnungsalgorithmen zur Konfiguration unterschiedlicher Datenquellen

Ein weiteres Beispiel ist die mögliche Konfiguration verschiedener Datenquellen: Ein Szenario, das ein Modul zur Wärme enthält, hat gewöhnlich einen Parameter *Umgebungstemperatur* (`Environment.HeatLevel`). Dieser Parameter kann seine Werte auf verschiedene Weise beziehen:

- **Konstanten:** Dies ist die einfachste Möglichkeit, die aber für die Simulation gewöhnlich nicht sehr ergiebig ist.

- **Berechnungen:** Im Prototyp wird die Umgebungstemperatur mit Hilfe einer periodischen Funktion dargestellt: Eine Überlagerung von zwei periodischen Funktionen zur Simulation der Temperaturschwankungen im Tages- und im Jahresablauf erscheint als eine erste Annäherung eines realistischen Temperaturverlaufs. Die Implementierung dieses Vorgehens wird im nächsten Kap. I beschrieben.
- **Tabellen:** Eine Möglichkeit, einen Temperaturwert für einem bestimmten Zeitpunkt zurückzuliefern, ist der Rückgriff auf eine Tabelle. In dieser Tabelle wird einem Zeitpunkt ein Wert für die Temperatur zugewiesen. Diese Möglichkeit kann genutzt werden, um beispielsweise sogenannte *Test Reference Year* (TRY) – Daten, die stündliche Temperaturwerte über ein Jahr enthalten, in die Simulation einzuspeisen (Lund, 2001).

I Parameterberechnungsalgorithmus Umgebungstemperatur

Wärmeschutz gehört zu den bauphysikalischen Teilgebieten. Um diesbezügliche Fragestellungen in Simulationen zu untersuchen, ist die Außentemperatur eine wichtige Eingangsgröße: Viele thermodynamische Vorgänge unter Einbeziehung von Gebäuden hängen von der Außentemperatur ab. In einem Simulationsmodell, mit dem das Verhalten von Gebäuden untersucht wird, gehört die Außentemperatur zu den Eingangsgrößen. Das heißt, dass keine Berechnung im Rahmen des Simulationsmodells stattfindet, sondern der Verlauf der Außentemperatur als gegeben betrachtet wird. Es gibt verschiedene Möglichkeiten, einen Außentemperaturverlauf zu erhalten: U.a. kann sie als konstant angenommen werden oder aus Messdaten abgeleitet werden. Die erste Möglichkeit hat den Nachteil, dass Effekte, die durch Temperaturschwankungen sichtbar werden, in der Simulation nicht auftreten werden. Die zweite Möglichkeit erfordert den Aufwand der Messung. Als effizient zu implementierende Lösung wurde die Außentemperatur für den Prototyp über eine periodische Funktion berechnet: Eine grobe Analyse des Verlaufs der Temperatur ergibt, dass diese im Winter ihr Minimum erreicht und im Sommer ihr Maximum. Grob betrachtet kann der Verlauf über eine Sinus-Funktion modelliert werden. Dasselbe Modell wurde für den Tagesverlauf der Temperatur angenommen: Am Mittag, wenn die Sonne am höchsten steht, ist es gewöhnlich am wärmsten, in der Nacht, wenn ohne Sonne eine Abkühlung stattgefunden hat, ist die Temperatur am niedrigsten. Auch dieses Verhalten kann durch eine Sinusfunktion angenähert werden. Die Zielfunktion ergibt sich durch die Überlagerung (d.h. Summation) zweier Sinusfunktionen.

Die Sinusfunktion ist gegeben mit

$$f(x) = a \sin(bx + c)$$

bzw. in der physikalischen Darstellung

$$f(t) = A \sin(\omega t + \varphi)^2$$

Dabei gelten die folgenden Symbolbedeutungen:

A : Amplitude der Schwingung

ω : Kreisfrequenz, gibt die Anzahl der Schwingungen in 2π Zeiteinheiten an/ $T = \frac{2\pi}{\omega}$ gibt die Schwingungsdauer an.

φ : Phasenverschiebung

² Semendjajew, Musiol, Muehlig & Bronstein (2008)

Um die annähernde Formel für die Außentemperatur angeben zu können, sind pro Sinussummand 3 Parameter festzulegen:

Jahreszeitliche Temperaturschwankung

Summand 1 bildet die jahreszeitlich bedingte Temperaturschwankung ab.

- **Amplitude:** Die Amplitude wird berechnet als die Hälfte der Differenz zwischen angenommener höchster und angenommener niedrigster Temperatur eines Jahres.
- **Schwingungsdauer:** Die Schwingungsdauer ist hier festgelegt als 1 Jahr.
- **Phasenverschiebung:** Die Phasenverschiebung ist so zu wählen, dass die Sinusfunktion ihr Minimum zu einem Zeitpunkt T_{DL} erreicht. Dieser wird als 30. Januar gewählt.

Tägliche Temperaturschwankung

Summand 2 bildet die Temperaturschwankung im Tagesverlauf ab.

- **Amplitude:** Es wird eine Tagesschwankung von 10 K angenommen, die Amplitude beträgt daher 5 K.
- **Schwingungsdauer:** Die Schwingungsdauer beträgt hier 1 Tag.
- **Phasenverschiebung:** Die Phasenverschiebung ist so zu wählen, dass die Sinusfunktion ihr Minimum zu einem Zeitpunkt T_{HL} erreicht. Dieser wird hier auf 2:00 Uhr morgens festgelegt.

Fasst man die oben gegebenen Annahmen zusammen, so ergibt sich die folgende Formel:

$$f(t_D, t_T) = T_{\min} + \left(\sin\left(\frac{t_D - T_{DL} - 365/4}{365} * 2\pi\right) + 1 \right) * \frac{T_{\max} - T_{\min}}{2} + A_D * \sin\left(\frac{t_T - T_{HL} - 6}{24} * 2\pi\right)$$

Symbole:

t_D : $t_D \in \mathbb{N} \wedge t_D \in [1, 365]$ (Tag des Jahres)

t_T : $t_T \in \mathbb{R} \wedge t_T \in [0, 24[$ (Zeit des Tages)

A_D : Tagesamplitude (konstant, hier: 5 K (angenommene Differenz zwischen höchster und niedrigster Temperatur eines Tages: 10 K))

T_{DL} : Tag mit der niedrigsten Temperatur (konstant, hier: 30 (30. Januar))

T_{HL} : Stunde mit der niedrigsten Temperatur (konstant, hier 2: (2:00 Uhr morgens))

T_{\min} : angenommene Minimaltemperatur des Jahres

T_{\max} : angenommene Maximaltemperatur des Jahres

Formel 1: Nachbildung des Jahresverlaufs der Außentemperatur

Die Formel zur Annäherung der Umgebungstemperatur ist in der Klasse

`EnvironmentTemperatureSource`³ implementiert.

```
public class EnvironmentTemperatureSource extends AbstractParameterSourceAlgorithm<Double> {
    private static double DAY_WITH_LOWEST_TEMPERATURE = 30.0;
    private static double HOUR_WITH_LOWEST_TEMPERATURE = 2.0;
    private static double MIN_AVERAGE_TEMP = -3;
    private static double MAX_AVERAGE_TEMP = 18;
    private static double DAILY_AMPLITUDE = 5;
```

³ Im Package `edu.uni_weimar.simuframe.tenementbuilding.heat.algorithm`.

```

@Override
public void calculate(ISimulationContext context) {
    ITimeManager timeManager = context.getTimeManager();
    if (timeManager != null) {
        Date gameTime = timeManager.getGameTime();
        if (gameTime != null) {
            getTargetParameter().setValue(
getEnvironmentTemperature(gameTime));
        }
    }
}

public static double getEnvironmentTemperature(Date gameTime) {
    Calendar instance = Calendar.getInstance();
    instance.setTime(gameTime);
    int hour = instance.get(Calendar.HOUR_OF_DAY);
    int dayOfYear = instance.get(Calendar.DAY_OF_YEAR);
    double temperature = calculateBaseTemp(dayOfYear);
    temperature += calculateDailyAdjustment(hour);
    return temperature;
}

public static double calculateDailyAdjustment(int hour) {
    double tempDiff = Math.sin((hour - HOUR_WITH_LOWEST_TEMPERATURE -
6.0) /
                                24.0 * 2.0 * Math.PI);
    double delta = tempDiff * DAILY_AMPLITUDE;
    return delta;
}

public static double calculateBaseTemp(int dayOfYear) {
    double temperature = Math.sin(((dayOfYear -
DAY_WITH_LOWEST_TEMPERATURE -
                                91.25) / 365.0 * 2.0 * Math.PI));
    double diff_temp = (MAX_AVERAGE_TEMP - MIN_AVERAGE_TEMP)/2.0;
    temperature = temperature * diff_temp + (diff_temp +
MIN_AVERAGE_TEMP) ;
    return temperature;
}
}

```

Quellcode 1: Klasse EnvironmentTemperatureSource

2.2 Benutzung von Entwurfsmustern

Entwurfsmuster (engl. *design patterns*) bestehen aus erprobten Lösungen für typische Probleme. Sie haben zum einen den Vorteil, dass sie fertige Lösungen liefern, sofern das Problem richtig erkannt und eingeordnet werden konnte. Desweiteren bieten sie die Grundlage für eine gemeinsame Kommunikation der Beteiligten: Wenn jemand den Namen eines verwendeten Entwurfsmuster nennt, hat ein anderer Experte es wesentlich einfacher, die implementierte Lösung zu verstehen. Ein wesentlicher Meilenstein im Bereich der Entwurfsmuster war die Arbeit von Christopher Alexander (1978) – auf dem Gebiet der Architektur. Bezugnehmend auf seine Arbeit wurde erstmals von Gamma et al. (1995) ein Katalog von Entwurfsmustern für die Softwareentwicklung definiert. Dieser Katalog etablierte Entwurfsmuster als Werkzeug bei Entwurfsentscheidungen: eine ganze Reihe von weiteren Entwurfsmusterkataloge sind seitdem in – erstaunlich unterschiedlichen - Fachgebieten entstanden, zudem wird die Bedeutung der Entwurfsmuster diskutiert:

- Martin Fowler beschäftigt sich mit Analysemustern und wiederverwendbaren Objektmodellen (1999).
- Whitson John Kirk III findet Design Patterns für Rollenspiele (2005).

- Staffan Björk und Jussi Holopainen stellen einen Katalog an allgemeinen Mustern für die Spielentwicklung zusammen (2005).
- Axel Schmolitzky und Till Schümmer transformieren die gesammelten Erfahrungen bei der Erstellung von akademischen Abschlussarbeiten in Entwurfsmuster (2002).
- Lewis et al. stellen einen Katalog an Entwurfsmustern zusammen, deren Benutzung in SNGs die Motivation der Spieler fördern soll (2012).
- Buschmann et al. beschreiben eine Pattern-orientierte Softwarearchitektur (1996).
- Wolfgang Pree beschreibt den Einsatz und die Bedeutung von Entwurfsmustern für die Softwareentwicklung – zeitlich vor Gamma et al. (1994).

Entwurfsmuster sind zu einem wichtigen Werkzeug der Softwareentwicklung geworden. Auch in dem im Rahmen dieser Arbeit entwickelten Simulationskern *SimuFrame* wurden an den Stellen, an denen es angebracht erschien, Entwurfsmuster eingesetzt. Im Folgenden wird eine Auswahl der Entwurfsmuster, die in *SimuFrame* verwendet werden und von Gamma et al. beschrieben wurden, zusammen mit Beispielen der Anwendung vorgestellt.

2.2.1 Observer Pattern

Gamma et al. (1995) definieren das *Observer Pattern* als

„Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.“

Beteiligt an diesem Muster sind mehrere Klassen: Das als *Publisher* fungierende Objekt ist die Quelle der Zustandsänderung. Es gibt seine Statusänderungen als Ereignisse an alle interessierten Objekte (in der Abbildung 1 *Subscriber* genannt) weiter. Dazu pflegt es eine Liste der Subscriber. Die Subscriber sind selbst dafür verantwortlich, sich beim Publisher zu registrieren.

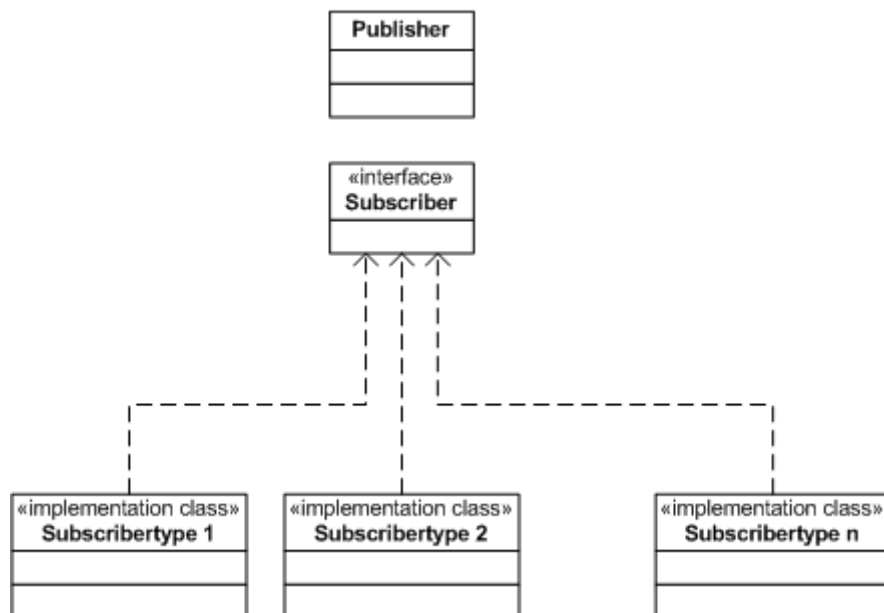


Abbildung 1: Klassendiagramm des Publisher-Subscriber Patterns.

Einer der Vorteile des Musters ist es, dass es auch zur Entkopplung von Modulen dienen kann: Modul A definiert die *Publisher*-Klasse und eine *Subscriber*-Schnittstelle (Interface). Die aktuellen *Subscriber*-Implementierungsklassen können in anderen Modulen liegen und müssen dem

Modul A nicht bekannt sein. Lediglich die anderen Module, die auf Ereignisse des Publishers reagieren möchten, besitzen Modul A als Voraussetzung. Damit ist das Observer Pattern eine wertvolle Hilfe bei der Entkopplung von Modulen.

Ein Beispiel ist die Aktion `WindowInstaller` (Fenstertausch), mit Hilfe dieser Aktion lassen sich neue Fenster einbauen, die u.a. andere (bessere) Werte für die Parameter `NoiseReductionIndex` (Schalldämmmaß) und `HeatTransferCoefficient` (Wärmedurchgangskoeffizient, U-Wert) haben. Der Fenstertausch kostet natürlich Geld, das aus der Kasse (Parameter `Cash`) der Gebäudeverwaltung entnommen wird. Dies soll den Spieler dazu veranlassen, seine Entscheidung gut abzuwägen. Die Aktion `WindowInstaller` ist im Modul `TenementBuilding` definiert und agiert als Publisher, der Parameter `Cash` ist ein Objekt des Moduls `TenementContract`, das das Modul `TenementBuilding` voraussetzt und ist ein Subscriber. Somit ist der Aktion `WindowInstaller` der Parameter `Cash` nicht bekannt. Damit trotzdem eine Verbuchung der Kosten stattfinden kann, löst die Aktion `WindowInstaller` ein Ereignis aus, das Modul `Contract` kennt das Modul `TenementBuilding` und somit auch die potentiellen Ereignisse dieses Moduls und hat sich daher dort zuvor als Zuhörer registriert. Daher wird das Modul `Contract` über jeden Fenstertausch unterrichtet und kann eine entsprechende Verbuchung vornehmen.

2.2.2 Singleton Pattern

„Ensure a class only has one instance, and provide a global point of access to it“
- Gamma et al. (1995)

Das *Singleton Pattern* wurde im vorliegenden Prototyp immer dann angewendet, wenn es darum ging, einen eindeutigen Zugriffspunkt für Informationen zu sichern. Das ist insbesondere in Klassen zur Verwaltung von Objekten der Fall.

Klasse	Aufgabe
<code>BadgeManager</code>	Verwaltung der Auszeichnungen
<code>IdentifizierManager</code>	Zuteilung der Identifikatoren für Elemente des Simulations-Modells, damit diese auch über Servergrenzen hinweg eindeutig sind.
<code>ParameterTypeManager</code>	Verwaltung der Parametertypen, Instanziierung von Parametern
<code>ScenarioManager</code>	Verwaltung der Szenarios
<code>SimuElementManager</code>	Verwaltung der Simulationselemente
<code>SkillManager</code>	Verwaltung der Fähigkeiten, Berücksichtigung von Fähigkeitsbäumen
<code>SystemEventManager</code>	Verwaltung und Hintergrundverarbeitung der Systemereignisse
<code>UnitManager</code>	Verwaltung der Einheiten von Größen; Service zur einheitenberücksichtigenden Berechnung von Parametern
<code>UserManager</code>	Verwaltung der Daten der Spieler
<code>WeightedAverageSourceContributorManager</code>	Verwaltung von Parametern, die zu anderen Parametern mit Hilfe des Mechanismus <i>WeightedAverageSourceContributor</i> beitragen.

Tabelle 1: Manager-Klassen

2.2.3 Visitor Pattern

„Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.“
- Gamma et al. (1995)

Das *Visitor Pattern* kann gewählt werden, um die Struktur eines Objekt-Modells und die Verarbeitung⁴ der einzelnen Modell-Elemente voneinander zu entkoppeln. Wie in der Definition schon beschrieben wurde, erlaubt dieses das Hinzufügen von Operationen in einem späteren Stadium der Entwicklung, ohne schon bestehende Klassen ändern zu müssen. Bei einem herkömmlichen Lösungsansatz müssten die schon bestehenden Klassen der Modellelemente jeweils für eine weitere Operation erweitert werden – mit einem erhöhten Risiko, dass sich Fehler einschleichen⁵.

Beispiel für die Benutzung des Visitor Patterns in SimuFrame ist die Abarbeitung des Simulationsmodells. Das Simulationsmodell besteht im Grundsatz aus einer hierarchischen Struktur von Simulationselementen mit angehängten Parametern (Abbildung 2). Es gibt mehrere Operationen, die im Verlaufe der Simulation auf alle Simulationselemente angewendet werden. Darunter sind solche Operationen wie `SimuElementRemover` (entfernt ein als Parameter übergebenes `SimuElement` aus dem Objektgeflecht des Simulationsmodells), `SimuElementCounter` (zählt die Simulationselemente eines bestimmten Typs im Simulationsmodell) und `SimuElementVisitor` (stößt einen Schritt in der Berechnung für jedes Simulationselement und jeden Parameter an). Diese Operationen implementieren das Interface `ISimuElementVisitor`, das sie verpflichtet, die Methode `visit(ISimuElement)` bereitzustellen. Das Entwurfsmuster sieht vor, dass diese Methode für jedes `SimuElement` der Modellhierarchie aufgerufen wird. In der Klasse des `SimuElements` (bzw. der abstrakten Basisklasse `AbstractSimuElement`) gibt es die Methode `accept()`, die für eine Traversierung des Simulationsmodells zuständig ist, d.h. dafür sorgt, dass alle beteiligten Objekte mit der Schnittstelle `ISimuElement` „besucht“ werden. Mit Hilfe dieser Methode können dann beliebige Operationen vom Typ `ISimuElementVisitor` zur Ausführung gebracht werden (Quellcode 2).

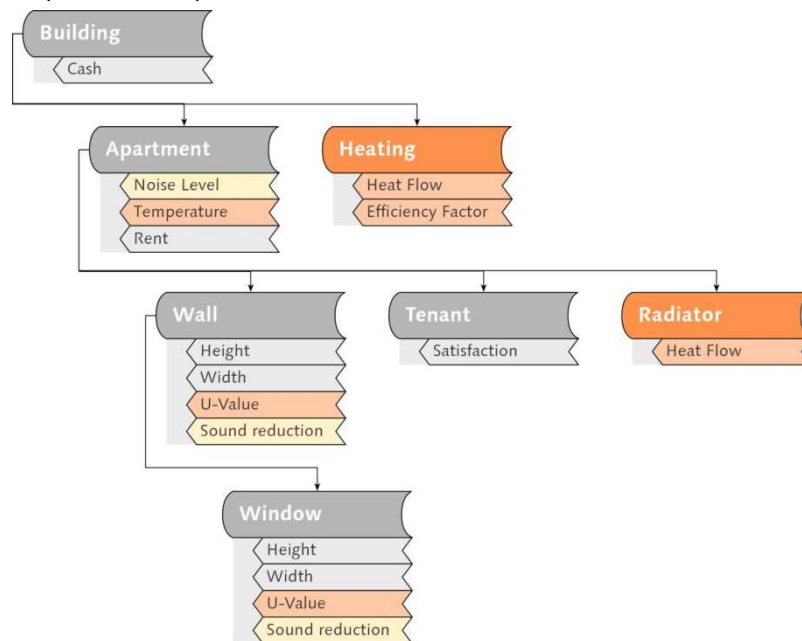


Abbildung 2: Hierarchie von Simulationselementen mit angehängten Parametern

```

01: public void accept(ISimuElementVisitor visitor) {
02:     boolean goOn = visitor.visit(this);
03:     if (goOn) {

```

⁴ In Form von Operationen, die auf die Modellelemente angewendet werden.

⁵ Die Wartbarkeit von Software wird in Anhang B behandelt.

```

04:     Collection childrenToVisitLater = new Vector();
05:     Collection childrenVisited = new Vector ();
06:     childrenVisited.add(this);
07:     Collection childrenToVisit = new Vector(getChildren());
08:     addDynamicSimuElements(childrenToVisit, this);
09:     while (goOn) {
10:         for (ISimuElement ele: childrenToVisit) {
11:             goOn = visitor.visit(ele);
12:             childrenVisited.add(ele);
13:             if (goOn){
14:                 Collection children = ele.getChildren();
15:                 if (children != null) {
16:                     childrenToVisitLater.addAll(children);
17:                     addDynamicSimuElements(childrenToVisitLater, ele);
18:                 }
19:             }
20:         }
21:         childrenToVisit = childrenToVisitLater;
22:         childrenToVisit.removeAll(childrenVisited);
23:         childrenToVisitLater = new Vector<ISimuElement>();
24:         if (childrenToVisit.size() == 0) {
25:             goOn = false;
26:         }
27:     }
28: }
19:}

```

Quellcode 2: Klasse AbstractSimuElement: Methode zur Traversierung der SimuElement-Hierarchie

Die Methode `accept(ISimuElementVisitor)` ist die zentrale Codeeinheit, die zu ändern ist, wenn sich der Aufbau des Simulationsmodells ändert. Sie navigiert durch das Simulationsmodell und ruft für alle Simulationselemente (Typ `ISimuElement`) die Methode `visit(ISimuElement)` des entsprechenden `visitor`-Objektes mit dem Simulationselement als Parameter auf (Zeilen 2 und 11).

Die leichte Wartbarkeit des Codes im Falle einer Strukturänderung des Simulationsmodells wurde deutlich mit der Einführung eines dynamisch erweiterbaren Datenmodells (s. Kap. 2.3.3.). Zu diesem Zeitpunkt wurden zusätzliche Simulationselemente in die Simulationselementhierarchie eingehängt. Um die Verarbeitung des Modells weiter korrekt zu gewährleisten, war lediglich eine Änderung in dieser Methode notwendig: In den Zeilen 8 und 17 wurden Aufrufe der ebenfalls neugeschriebenen Methode `addDynamicSimuElements(Collection, ISimuElement)` eingefügt, damit wurde das Modell trotz Strukturänderung wieder korrekt traversiert.

2.2.4 Weitere Entwurfsmuster

Bei der Entwicklung des Simulationskerns `SimuFrame` wurden weitere Entwurfsmuster eingesetzt, die hier in einer Zusammenfassung erwähnt werden. Sie bedürfen keiner detaillierten Erläuterung, da die Nennung der Bezeichnung des Entwurfsmusters allein als beschreibende Referenz dient.

Entwurfsmuster	Beschreibung (lt. Gamma et al. (1995))	Beispiel
AbstractFactory	<i>Provide an interface for creating families of related or dependent objects without specifying their concrete classes.</i>	<code>IScenarioFactory</code> : Dieses Interface müssen alle Klassen, die Szenarios bereitstellen, implementieren.
Prototype	<i>Specify the kinds of objects to</i>	<code>ScenarioManager.createScenario-</code>

	<i>create using a prototypical instance, and create new objects by copying this prototype.</i>	(String): Diese Methode kopiert Szenarien-Prototypen, um ein ausführbares Szenario, zu erzeugen.
Adapter	<i>Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.</i>	Durch den hierarchischen Aufbau des Simulationsmodells bedingt werden zurzeit Nachbarschaftsbeziehungen zwischen Wohnungen nicht unterstützt. <code>INeighbourTenements</code> ist ein Interface, das dennoch eine Nachbarschaftsbeziehung modelliert.
Composite	<i>Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.</i>	Die Simulationselemente werden zu einer Hierarchie aufgebaut, d.h. sie besitzen oft untergeordnete Simulationselemente, und meistens ein übergeordnetes Simulationselement. <code>ISimuElement</code> ist das Interface, in dem die Methoden <code>getChildren()</code> und <code>getParent()</code> modelliert sind.

Tabelle 2: In SimuFrame verwendete Entwurfsmuster

2.3 Designaspekte zur Unterstützung von Modularität und Erweiterbarkeit

Software-Entwicklung besteht aus einer Folge von Entscheidungen zur Struktur der Software und deren Umsetzung – getrieben von den Anforderungen an die Software. Bei der Entwicklung dieser Frameworks wurde Wert darauf gelegt, an den Stellen, an denen aus der konkreten Anforderung generalisierbare Problemstellungen erwachsen sind, wiederverwendbare Lösungen zu schaffen, die an anderer Stellen im Rahmen dieses Frameworks erneut eingesetzt werden können. Der Prozess der Entwicklung von wiederverwendbaren Lösungen wurde dokumentiert. Im Folgenden werden die wichtigsten Designentscheidungen dargelegt. Dies ist eine Möglichkeit, das Framework in seinen Einzelheiten darzustellen.

2.3.1 Darstellungsschema

Die hier vorgestellten Designentscheidungen haben den Charakter von Mustern oder Musterlösungen. Nachdem Gamma et al. (1995) mit ihrem Buch den Einsatz von Mustern im Softwaredesign etabliert haben, wurde in Anlehnung an dieses Vorbild einige weitere Lösungskataloge entwickelt⁶. Gemeinsames Merkmal der meisten dieser Kataloge ist eine formalisierte Musterbeschreibung. So wurde auch zur Darstellung der hier verwendeten Muster ein Beschreibungsgerüst gewählt, das die relevanten Informationen zu einer Designentscheidung wiedergeben soll (s. Tabelle 3). Dieses Beschreibungsgerüst muss in der konkreten Darstellung nicht vollständig ausgefüllt sein, so dass im Einzelfall auch einige Beschreibungselemente übergangen werden können. Basis des Entwurfs dieser Darstellung waren schon existierende Musterkataloge, wie die schon erwähnten von Gamma et al. (1995) und Björk & Holopainen (2004) sowie von Martin (2002).

Beschreibungselement	Beschreibung
Konkretes Problem	Eine Designentscheidung wird gewöhnlich durch ein konkretes Problem initiiert: Die Notwendigkeit einer Designentscheidung wird anhand dieses Problems deutlich.

⁶ Eine Übersicht wird in Kap. 2.2 Benutzung von Entwurfsmustern (S. 95) gegeben.

Generelles Problem	Das konkrete Problem wird abstrahiert, um zugrundeliegende Strukturen herauszuarbeiten. Eine mögliche Lösung kann dadurch für weitere Probleme nutzbar werden.
Lösungsansatz	Im Lösungsansatz werden die Grundlagen für die Lösung des generellen Problems beschrieben.
Anforderungen	Die Anforderungen diskutieren die sich aus dem Lösungsansatz möglicherweise ergebenden Forderungen an die zu lösenden Probleme bzw. an die Lösung selbst.
Implementierung	In der Implementierung wird die konkrete technische Umsetzung der Lösung beschrieben.
Grenzen und Erweiterungsmöglichkeiten	Eine Designentscheidung entsteht gewöhnlich als Ergebnis einer Abwägung möglicher Vor- und Nachteile. Auch ist es ratsam, nicht „auf Vorrat“ zu entwickeln, um mit einer Designentscheidung theoretisch möglichen Anforderungen gerecht zu werden, die in der Praxis aber noch gar nicht vorhanden sind. An dieser Stelle werden bewusst akzeptierten Grenzen der Designentscheidung zusammen mit eventuellen Erweiterungsmöglichkeiten dargelegt.

Tabelle 3: Formales Beschreibungsschema einer Designentscheidung

2.3.2 Parameter Contribution

a) Konkretes Problem

Bei der Berechnung der Mieterzufriedenheit, die durch einen Zahlenwert im Bereich von -1 und 1 ausgedrückt wird, sollen Teilfaktoren berücksichtigt werden: Z.B. die Zufriedenheit mit dem Mietpreis, die Zufriedenheit mit dem Lärmpegel in der Wohnung. Jedes noch zu schaffende Modul soll einen Beitrag zur Mieterzufriedenheit liefern können. Es sollen alle Module beitragen können, die für das gewählte Szenario aktiviert sind und einen Beitrag zur Mieterzufriedenheit leisten.

b) Generelles Problem

Wie kann ein Parameter A zum Wert eines anderen Parameters B beitragen, wenn zum Zeitpunkt der Implementierung von Parameter B der Parameter A noch gar nicht bekannt ist?

c) Lösungsansatz

Es gibt einen Parameter `TenantSatisfaction`, der jedem Mieter zugeordnet wird.

Dieser Parameter bietet eine Erweiterungsmöglichkeit an, d.h. andere Module können Beziehungen zu weiteren Parametern anlegen, die bei der Berechnung des Parameters genutzt werden. Generelles Merkmal sowohl des übergeordneten Parameters als auch der untergeordneten Parameter ist der normierte Wertebereich von $[-1, 1]$. Ein Wert von -1 ist der schlechtestmögliche und ein Wert von +1 der bestmögliche Wert des jeweiligen Parameters.

Ein einfacher Fall für eine Erweiterung sieht wie folgt aus:

Das Modul `Acoustics` trägt den Parameter `ComfortableNoiseLevel` zum Simulationselement `Tenant` bei, gleichzeitig bekommt das Simulationselement `Tenement` einen Parameter `NoiseLevel`.

Den beiden Parametern kommt die folgende Bedeutung und Rolle zu:

Parameter	Rolle	Beschreibung
-----------	-------	--------------

ComfortableNoiseLevel	Soll-Wert	Wohlfühl-Geräuschpegel
NoiseLevel	Ist-Wert	Geräuschpegel

Die zwei Werte werden zu einem Parameter-Beitrag zusammengeführt, d.h. aus den zwei Werten wird eine Zahl aus dem Wertebereich von $[-1, 1]$ errechnet, die die modulspezifische Zufriedenheit darstellt.

Dieser Parameter-Beitrag einer modulspezifischen Zufriedenheit wird dem übergeordneten Parameter `TenantSatisfaction` zugeordnet. Dies geschieht über einen Extension Point. Bei der Zusammenfassung zum Wert für die Zufriedenheit werden alle zugeordneten Parameter-Beiträge der Module arithmetisch gemittelt. Gleichzeitig gibt es sogenannte *K.O.-Werte* für die Parameter-Beiträge: Wenn eine Schwelle für einen Beitrag unterschritten wird, wird der Wert des übergeordneten Parameters `TenantSatisfaction` auch auf diesen Wert gesetzt – es wird nicht das Mittel berechnet. Hintergrund ist die Überlegung, dass das deutliche Unterschreiten einer Toleranzschwelle bezüglich eines Merkmals die komplette Zufriedenheit des Mieters stark beeinträchtigt: Ein weiterer Verbleib in der Wohnung ist auf Dauer nicht möglich. Als Beispiel sei hier eine defekte Heizung angeführt. Im Winter ist dies ein inakzeptabler Zustand, der auch durch beste Werte in allen anderen Bereichen nicht ausgeglichen werden kann.

d) Anforderungen

Die Wertebereiche von übergeordnetem Parameter und untergeordneten Parametern müssen übereinstimmen bzw. eindeutig aufeinander abgebildet werden können.

e) Implementierung

Zunächst wird der übergeordnete Parameter als Extension des Extension Points `parameterSource` definiert und über den Extension Point `simuElementParameter` dem Simulationselement `Tenant` zugeordnet. Dann wird dem Parameter ein Parameterberechnungsalgorithmus zugewiesen. Dies ist eine für diese Problemstellung erstellte Java-Klasse `WeightedAverageSource`. Sie ist für die Berechnung zuständig. Dabei wendet sie sich an eine zentrale Klasse `WeightedAverageSourceContributorManager`: Das ist eine zentrale Instanz, an der über einen Extension Point alle beitragenden Parameter angemeldet werden. Über die Methode `getContributors(String)` kann diese nach den Parametern gefragt werden, die zu dem übergeordneten Parameter beitragen. Aus den Werten dieser Parameter wird dann der aktuelle Wert des übergeordneten Parameters berechnet. Das geschieht zur Laufzeit: damit ist die Anforderung erfüllt, dass die Zuordnung der Parameter je nach aktiviertem Modul stattfinden kann.

Es gibt einen Mechanismus, der dafür sorgt, dass ein beitragender Parameter der zentralen Klasse `WeightedAverageSourceContributorManager` bekanntgegeben wird. Dies geschieht ebenfalls über einen Extension Point: Der Extension Point `weightedAverageContributor` benötigt den beitragenden Parameter und den übergeordneten Parameter. Mit Hilfe dieser Information kann die Zuordnung der beiden Parameter durchgeführt werden.

f) Grenzen und Erweiterungsmöglichkeiten

Die aktuell implementierte Variante arbeitet für die vorgesehene Zielstellung, mehreren Modulen variabel einen Beitrag zum Parameter `TenantSatisfaction` zu erlauben. Für zukünftige Entwicklungen sind jedoch einige Erweiterungen denkbar.

- **Parameter müssen am selben Simulationselement aufgehängt sein:** Der Berechnungsalgorithmus `WeightedAverageSource` sucht die ihm mitgeteilten Parameter am aktuellen Simu-

lationselement (Für das Beispiel ist das `Tenant`): Hier ist es denkbar, dass auch Navigationswege (mit den Rollennamen untergeordneter Simulationselemente) zum Parameter mitgeteilt werden oder die Suche generell auf untergeordnete Simulationselemente ausgedehnt wird.

- **Aggregationsfunktion:** Eine Schwäche der vorgestellten Lösung ist die fehlende Gewichtungsmöglichkeit der beitragenden Parameter. Im Moment ist es nicht möglich, einen Parameter höher zu gewichten als andere Parameter, alle Parameter leisten den gleichen Beitrag durch die Bildung des arithmetischen Mittelwertes. Dem Fall, dass einige beitragende Parameter von höherer Bedeutung als andere sein könnten, wird nicht Rechnung getragen. Eine Möglichkeit der Erweiterung wäre ein Gewichtungsfaktor, der bei der Extension zum `weightedAverageContributor` angegeben werden müsste. So könnte beispielsweise dem Parameter `HeatSatisfaction` die doppelte Gewichtung gegeben werden. Eine weitere Lösungsmöglichkeit, die in Teilen schon realisiert ist, sind K.O.-Werte (s.o.): extrem schlechte Werte in Teilen ziehen den Gesamtparameter auf einen schlechten Wert. Insgesamt bleibt jedoch die Frage, ob es noch weitere Anwendungsmöglichkeiten für solche K.O.-Werte gibt. Denkbare Erweiterungen sind auch Parameterberechnungsalgorithmen des übergeordneten Parameters, die zusätzlichen Aggregationsfunktionen implementieren, z.B. die Summe der beitragenden Parameterwerte.

2.3.3 Dynamische Erweiterung des Datenmodells

a) Konkretes Problem

Wie lassen sich im Simulationsmodell Nachbarschaftsbeziehungen zwischen Wohnungen abbilden?

Auf den ersten Blick verwundert diese Problemstellung: Eigentlich sollte sich die Nachbarschaft von Wohnungen aus der Geometrie des aufnehmenden Gebäudes ergeben. Jedoch in Anhang C: Grundlagen der Simulation dargelegt, dass der Entwurf des Simulationsmodells getrieben wird vom Zweck der Simulation. Da im Rahmen dieser Arbeit versucht wird, qualitative Simulationen mit begrenztem Modellierungs- und Berechnungsaufwand zu entwerfen, wurde auf eine exakte Geometrie des Gebäudes verzichtet. In der aktuellen Modellierung enthält ein Gebäude (`Building`) mehrere Wohnungen (`Tenement`), von denen die Größe bekannt ist, aber nicht die Lage. Dies ist auch eine Folge des Ansatzes eines hierarchischen Simulationsmodells: Einem Gebäude sind mehrere Wohnungen untergeordnet, ohne dass die Wohnungen miteinander verbunden sind. Für die Simulation der gegenseitigen Beeinflussung der Wohnungen (z.B. durch akustische oder thermodynamische Vorgänge) ist es jedoch notwendig, Nachbarwohnungen zu kennen.

b) Generelles Problem

Das generelle Problem, das hier gelöst werden soll, ist das dynamische Einfügen von Beziehungen der Simulationselemente untereinander, in anderen Worten eine dynamische Erweiterung des Datenmodells um neue Simulationselemente.

c) Lösungsansatz

Für jeden Fall, in dem eine solche Anforderung auftritt, gibt es eine Klasse, die für die Verwaltung der zusätzlichen Beziehungen und Simulationselemente zuständig ist. Diese Klasse wird gemäß dem Adapter-Entwurfsmuster (s. Kap. 2.2.4) entworfen. Die Abbildung einer zusätzlichen

Beziehung erfolgt mit Hilfe von generischen Attributen, d.h. es müssen keine Zugriffsmethoden programmatisch definiert werden, sondern die Speicherung von Referenzen in den Objekten ist möglich allein mit Hilfe von unspezifischen Methoden und Variablen. Das Wissen über den Zugriff auf die dynamischen Elemente ist allein in der Adapter-Klasse vorhanden.

d) Implementierung

Die Begrenzungselemente werden als Simulationselemente modelliert⁷. Sie werden dem System über den Extension Point `simuElement` definiert, der die Attribute `className` (Name der Simulationselement-Klasse, diese Klasse muss das Interface `ISimuElement` implementieren), `description` und `name` (übernimmt die Funktion eines Identifikators) besitzt. Die Beziehung zwischen einem konkreten Begrenzungselement und einem konkreten Wandelement wird programmatisch abgebildet, da sie mit der Definition der hierarchischen Beziehung schon vorgegeben ist⁸.

Zur programmatischen Abbildung wird ein Adapter über den vom Eclipse RCP-Framework definierten Extension Point `org.eclipse.core.runtime.adapters` genutzt. Dazu muss dann eine Factory definiert werden, die den Adapter bereitstellt.

```
01: <extension
02:     point="org.eclipse.core.runtime.adapters">
03:     <factory
04:         adaptableType="Tenement"
05:         class="NeighbourAdapterFactory">
06:         <adapter
07:             type="INeighbourTenements">
08:         </adapter>
09:     </factory>
10:</extension>
```

Quellcode 3: Deklaration eines Adapters in der plugin.xml

Quellcode 3 weist das RCP-Framework an, dass – wann immer es auf ein Simulationselement vom Typ `Tenement` (Zeile 4) trifft – es die Klasse `NeighbourAdapterFactory` (Zeile 5) nach einem Adapter vom Typ `INeighbourTenements` (Zeile 7) für das betreffende `Tenement` fragen kann. In Quellcode 4 ist ein Einsatz des Adapters dokumentiert. Zum Verständnis des Codes ist in Abbildung 3 die Lage der Wohnungen zueinander dargestellt. In den Zeilen 1 bis 4 werden zwei Wandelemente erzeugt. In Zeile 5 wird von der Variablen für Wohnung 1 (`tenementsA[0]`) der Adapter zur Behandlung der Nachbarschaftsbeziehungen erfragt. Mit Hilfe dieses Adapters können dann in Zeile 6 und 7 die zwei Wandelemente für Wohnung 1 gesetzt werden. In Zeile 8 und 9 wird die Wohnung 1 mit der Wohnung 2 und Wohnung 3 nachbarschaftlich verbunden, d.h. das für Wohnung 1 schon gesetzte Wandelement wird auch für die andere Wohnung zum Wandelement.

```
01: WallElement we1And2 = WallElementFactory.createWallElement(null);
02: we1And2.setName("Wall between 1 and 2");
03: WallElement we1And3 = WallElementFactory.createWallElement(null);
04: we1And3.setName("Wall between 1 and 3");
```

⁷ Hintergrund: Ein Wandelement kann die Begrenzung für mehrere Wohnungen bilden, es kann also nicht im Rahmen einer „Ist Teil von“-Beziehung existieren, da dann nicht klar wäre, Teil welcher Wohnung das Wandelement denn nun ist.

⁸ Zu dem Zeitpunkt, an dem die hierarchische Beziehung zwischen Gebäude und Wohnungen definiert wird, sind auch die Lagen der Wohnungen zueinander bekannt. Daher muss das Datenmodell an dieser Stelle nicht erweiterbar sein.

```

05: INeighbourTenements nTenement1 = ↵
        tenementsA[0].getAdapter2(INeighbourTenements.class);
06: nTenement1.setWallElement(INeighbourTenements.DIRECTION_EAST, we1And2);
07: nTenement1.setWallElement(INeighbourTenements.DIRECTION_SOUTH, we1And3);
08: nTenement1.setNeighbour(INeighbourTenements.DIRECTION_EAST, tenementsA[1]);
09: nTenement1.setNeighbour(INeighbourTenements.DIRECTION_SOUTH, tenementsA[2]);

```

Quellcode 4: Szenario-Generierung: Zusätzliche Beziehungen mittels Adapter

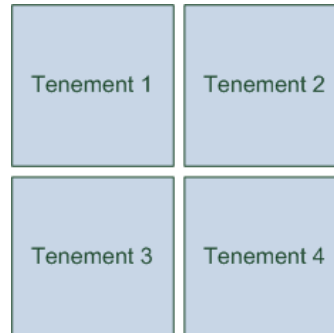


Abbildung 3: Lage der Wohnungen zueinander

```

01: public void setNeighbour(int direction, Tenement neighbourTenement) {
02:     RoomBoundary boundary = (RoomBoundary)_tenement.getDynamicSimuElement(↵
        COMPOSITE_NAME_ROOM_BOUNDARIES, Integer.valueOf(direction));
03:     WallElement wallElement = boundary.getWallElement();
04:     int opDir = getOppositeDirection(direction);
05:     INeighbourTenements neighbourAdapterOfNeighbour = neighbourTenement↵
        .getAdapter2(INeighbourTenements.class);
06:     neighbourAdapterOfNeighbour.setWallElement(opDir, wallElement);
07: }

```

Quellcode 5: Setzen der Nachbarschaftsbeziehung

Die Methode `setNeighbour (int, Tenement)` ist eine Methode des `INeighbourTenements`-Adapter der Klasse `Tenement`. In Zeile 2 wird das Simulationselement `RoomBoundary`, dass dynamisch mit dieser Wohnung verbunden ist, bestimmt. Dieses dynamische Simulationselement ist der eigentliche Grund, weshalb das vorgestellte Beispiel zum Thema „Dynamische Erweiterung des Datenmodells“ aufgeführt wird: Die nachträglich modellierte Beziehung wird mit Hilfe eines dynamischen Simulationselementes realisiert⁹: `RoomBoundary` ist eine Art Halterung für das Simulationselement `WallElement`. Somit kann dann in Zeile 3 das `WallElement` bestimmt werden. Zeile 4 bestimmt die Richtung, in der das `WallElement` in der Nachbarwohnung eingetragen werden muss. In Zeile 5 wird der entsprechende Adapter der Nachbarwohnung bestimmt, mit dessen Hilfe dann das eben bestimmte `WallElement` für die Nachbarwohnung gesetzt wird.

e) Grenzen und Erweiterungsmöglichkeiten

Eine Adapter-Klasse - und zusätzlich die Nutzung des Adapter-Mechanismus innerhalb des Eclipse RCP-Frameworks – ist eine konzeptionell relativ anspruchsvolle Lösung. Das bringt Probleme bei der Umsetzung mit sich. Deshalb ist für einen breiteren Einsatz noch eine Vereinfachung der Lösung vorteilhaft.

⁹ Dynamisch wird das Simulationselement `RoomBoundary` hier auch genannt, weil es im übergeordneten Simulationselement `Tenement` keine Zugriffsfunktionen (z.B. `getRoomBoundary()` und `setRoomBoundary(...)`) gibt, sondern die Beziehung über ein generisches Attribut erfolgt (`getDynamicSimuElement(...)` und `setDynamicSimuElement(...)`).

Desweiteren kommt die Benutzung eines hierarchischen Simulationsmodells an einigen Stellen – wie hier beispielhaft gezeigt – an seine Grenzen. Das Framework sollte erweitert werden, so dass es netzwerkartige Simulationsmodelle unterstützt.

2.3.4 Szenarioänderungsbasierte Systemereignisse

a) Konkretes Problem

Das Modul `TenementContract` (Vertrag) fügt zwei Parameter zum Simulationselement `Tenement` (Wohnung) hinzu: `TenancyAgreementStart` und `TenancyAgreementEnd`. Der Einzug und Auszug eines Mieters erfolgt jedoch ohne das Wissen um diese Parameter. Dennoch sollte das Modul `TenementContract` die Möglichkeit haben, auf das System-Ereignis *Mietereinzug* sowie *Mieterauszug* reagieren zu können, um dann gegebenenfalls die entsprechenden Parameter setzen zu können.

b) Generelles Problem

Erweiterungen des Szenarios (seien es Simulationselemente oder auch Parameter) können Informationen über Änderungen des Szenarios während der Simulation benötigen, um entsprechend reagieren zu können.

c) Lösungsansatz

Die Änderungen des Szenarios werden mit Hilfe von Systemereignissen weitergeleitet. Die Systemereignisse werden an den entsprechenden Stellen (also hier: Auszug eines Mieters (Prozedur `CmdMoveTenant`)) erzeugt und an die Zuhörer weitergereicht.

Für die Bekanntgabe wird das Observer-Muster (s. Kap. 2.2.1) genutzt. Dazu müssen Ereignisse ausgelöst werden. Zuhörer, die an diesem Ereignis interessiert sind, werden benachrichtigt, da sie sich zuvor für das Ereignis als Zuhörer eingetragen haben. Im Falle des Ereignisses wird dann gegebenenfalls¹⁰ eine Prozedur ausgeführt.

d) Implementierung

Zur Demonstration der Implementierung wird das in der Problemstellung dargelegte Beispiel genutzt: Die Details des Mietvertrages und der damit verbundenen Mietzahlungen werden im Modul `TenementContract` abgebildet. Der Ein- und Auszug in eine Wohnung ist eine Prozedur, die im grundlegenden Modul `TenementBuilding` definiert ist. Aufgabe der Implementierung ist es, das Modul `TenementContract` über einen Ein- oder Auszug zu informieren. Dabei ist zu beachten, dass dem Modul `TenementBuilding` das Modul `TenementContract` nicht direkt bekannt ist.

Schritt 1: Das Modul `TenementContract` registriert sich als Zuhörer für Events der Klasse `Tenement`

Dies geschieht in der Methode `start(...)`¹¹ der Klasse `TenementContractPlugin`. Die Klasse leitet von der Klasse `Plugin` ab. Das ist die zentrale Klasse eines Plug-ins des Eclipse-Frameworks.

```
01: public void start(BundleContext context) throws Exception {
```

¹⁰ Es können noch weitere Bedingungen getestet werden, von denen abhängig ist, ob die Prozedur wirklich ausgeführt werden soll oder nicht.

¹¹ Die Notation mit drei Punkten innerhalb der Klammern soll andeuten, dass diese Methode Parameter besitzt, diese jedoch im Sinne einer kurzen Darstellung weggelassen wurden, da Typ und Anzahl der Parameter für das Verständnis im aktuellen Kontext nicht notwendig sind.

```

02:   ListenerRegistry.getInstance().addListenerSupplier(↵
           new InitializeTenementAttributesAfterMoveIn(), Tenement.class);
03: }

```

Quellcode 6: Registrieren eines Zuhörers

Die `ListenerRegistry` ist die zentrale Instanz zur Verwaltung von Zuhörern innerhalb des Eclipse-Frameworks. Der Parameter `Tenement.class` gibt an, dass nur auf Ereignisse gehört werden soll, die durch die Klasse `Tenement` ausgelöst werden. Die Methode `addListenerSupplier` deutet an, dass nicht direkt der Zuhörer angegeben wird, sondern ein Objekt, das den Zuhörer kennt. Dies ist eine Instanz der Klasse `InitializeTenementAttributesAfterMoveIn`.

Schritt 2: Definition des Zuhörers

Der Zuhörer ist hier in der Klasse `InitializeTenementAttributesAfterMoveIn` definiert. Diese Klasse erfüllt zwei Funktionen: Die erste Funktion ist die des `ListenerSuppliers`, d.h. des Erzeugers eines Zuhörers. Dies ist eine Spezifik des Eclipse-Frameworks: Nicht direkt den Zuhörer anzugeben, sondern eine weitere Abstraktionsebene zwischen Ereigniserzeuger und –zuhörer zu legen. Der Vorteil dieses Vorgehens ist die Möglichkeit der zeitlichen Entkopplung der Registrierung eines Zuhörers und der Erzeugung des eigentlichen Zuhörer-Objekts. Zur Erfüllung dieser Funktion implementiert die Klasse das Interface `IListenerSupplier`. Die zweite Funktion dieser Klasse ist die Bereitstellung der eigentlichen Zuhörerfunktionalität: Dazu wird die Methode `simuElementEventOccured(...)` genutzt:

```

01: public void simuElementEventOccured(ISimuElementEvent event) {
02:   Tenement t = (Tenement)event.getSource();
03:   parameterDate = (ParameterDate)t.getParameter(↵
           TenementContractPlugin.↵
           PARAMETER_NAME_TENANCY_AGREEMENT_START);
04:   parameterDate.setValue(t.getScenario())↵
           .getContext().getTimeManager().getGameTime());
05: }

```

Quellcode 7: Ereignisbehandlungsprozedur

Diese Methode (Quellcode 10) bekommt das aufgetretene Ereignis als Parameter übergeben. Mit dem Ereignis ist der Auslöser- ein `Tenement` - verknüpft (Zeile 2). Das `Tenement` besitzt einen Parameter `TenancyAgreementStart`, der mit dem Datum der aktuellen simulierten Zeit belegt wird (Zeile 4).

An dieser Stelle ist die Prozedur, die im Falle des Ereignisses ausgeführt wird, im Zuhörer direkt implementiert. Für den Fall, dass eine Prozedur ausgeführt wird, die als Reaktion auch auf andere Systemereignisse oder Benutzeraktionen notwendig ist, sollte der entsprechende Code im Sinne einer Wiederverwendung in eine Prozedur-Klasse (`Command`) ausgelagert werden.

Schritt 3: Erzeugen und Senden des Ereignisses

Das Ereignis ist in diesem Fall der Einzug eines Mieters. Ein- und Auszüge eines Mieters werden durch eine Prozedur vorgenommen (`CmdMoveTenant`). Dadurch ist es möglich, bei jedem Einzug eines Mieters ein Ereignis zu erzeugen und an die angemeldeten Zuhörer weiterzureichen. An der Stelle in dieser Prozedur (Quellecode 11), in der ein Einzug stattfindet (Zeile 1), wird das Ereignis erzeugt (Zeile 2) und verteilt (Zeile 3):

```

01: _target.moveIn(tenant);
02: ISimuElementEvent simuElementEvent =
           new SimuElementEvent(EVENT_TENANT_HAS_MOVED_IN, _target);

```

```
03: AbstractSimuElement.fireSimuElementEvent(simuElementEvent);
```

Quellcode 8: Erzeugen eines Ereignisses

Bei der Erzeugung des Ereignisses wird die Quelle des Ereignisses als Parameter übergeben. Als Quelle gilt hier das `Tenement` (Zeile 2: `_target`), in das der Mieter (Zeile 1: `tenant`) einzieht. Das Senden des Ereignisses erfolgt durch die statische¹² Methode `fireSimuElementEvent(...)`. Intern wird dabei ein Verzeichnis aller – auch für andere Ereignisse - registrierten Zuhörer genutzt, um das Ereignis nur an die speziell auf dieses Ereignis ausgerichteten Zuhörer weiterzureichen.

e) Grenzen und Erweiterungsmöglichkeiten

Für die Entwicklung weiterer Zuhörer werden Informationen über die Ereignisse benötigt, auf die ein Zuhörer reagieren könnte. Im Moment ist es nur schwer möglich, einen Überblick der vorhandenen Ereignisse zu erhalten. Bisher sind die Ereignisse nur im Quellcode vorhanden, der durchsucht werden sollte¹³. Eine wesentlich komfortablere Lösung wäre es, die beobachtbaren Ereignisse programmatisch erfragen zu können. Das würde es ermöglichen, die Ereignisse im Szenario-Editor verfügbar zu machen.

2.3.5 Simulationselement bezogene Änderungsdialoge

a) Konkretes Problem

Fenster besitzen unterschiedliche Werte für den Parameter `NoiseReductionIndex` (Schalldämmmaß). Um dem Spieler den Umgang damit zu erlauben, besteht die Möglichkeit, die Fenster in einem Szenario auszutauschen, d.h. die bestehenden Modelle durch andere zu ersetzen. Das ist eine dedizierte Entscheidung des Spielers und soll von ihm in einer *Action* durchgeführt werden.

b) Generelles Problem

Ein Szenario besteht auch aus Simulationselementen. Um die Effekte unterschiedlicher Simulationselemente mit voneinander abweichenden Parameterwerten erfahrbar zu machen, sollte es für den Spieler generell möglich sein, diese Simulationselemente gegen andere Simulationselemente derselben Klasse auszutauschen.

c) Lösungsansatz

Es muss eine Menge von verfügbaren Ersatzbauelementen in Form von Simulationselementen geben. Diese werden im Szenario selbst verwaltet: Die Ersatzbauelemente sind mit Parametern ausgestattet, die durch die für das Szenario aktivierten Module bestimmt werden. So sind die Ersatzbauteile szenariospezifisch zu erstellen.

Es gibt einen allgemeinen Auswahldialog: Dieser präsentiert dem Spieler die verfügbaren Ersatzteile mit den entsprechenden spezifischen Parametern und ermöglicht eine Auswahl.

Für diejenigen Klassen von Simulationselementen, für die Ersatzbauelemente definiert sind, gibt es Spieleraktionen zum Austausch der Simulationselemente.

¹² Eine statische Methode ist eine Methode der Klasse. In der objektorientierten Programmierung beziehen sich Methoden normalerweise auf ein Objekt – die Instanz einer Klasse. Statische Methoden hingegen können auch ohne konkrete Instanz aufgerufen werden.

¹³ Alternativ kann die mit Hilfe von *Javadoc* (Oracle, 2012) generierte Quellcode-Dokumentation benutzt werden. Auch hier bleibt jedoch das Problem, dass der Suchende wissen muss, welche Klassen möglicherweise ein Ereignis produzieren.

d) Anforderungen

Die Ersatzbauelemente müssen eine Struktur (in Form von Parametern und untergeordneten Simulationselementen) besitzen, die derjenigen der initial vorhanden Bauelemente entspricht. Ansonsten würde ein Austausch das Simulationsmodell strukturell verändern.

e) Implementierung

Ersatzbauelemente:

Das Szenario erhält die Möglichkeit, einen Katalog von Ersatzbauelementen aufzunehmen. Dies geschieht durch die Definition einer Variablen, die zu jeder Typangabe eine Menge von Ersatzbauelementen aufnehmen kann:

```
01: private Map<String, Collection<ISimuElement>> _replacementParts;
```

Quellcode 9: Variable zur Aufnahme der Ersatzbauteile

In der Initialisierungsphase des Szenarios werden die Ersatzbauelemente hinzugefügt:

```
01: createReplacementWindow(scenario, "PHW-TG", ↵
02:     "Passive House Window, Triple Glazing",
03:     0.8, 43.0, 650);
```

Quellcode 10: Methodenaufruf zur Erzeugung eines Ersatzbauteils

Die Methode `createReplacementWindow(...)` erzeugt aus den Methodenparametern ein Simulationselement der Klasse `Window` und fügt es den Ersatzbauteilen hinzu (Quellcode 14).

```
01: private static Window createReplacementWindow(↵
    TenantBuildingScenario scenario, ↵
    String name, ↵
    String description, ↵
    double uvalue, ↵
    double soundReductionIndex, ↵
    double cost) {
02:     Window window = WindowFactory.createWindow(null);
03:     window.setName(name);
04:     window.setDescription(description);
05:     window.setParameterValue(↵
        HeatPlugin.PARAMETER_HEAT_TRANSFER_COEFFICIENT, uvalue);
06:     window.setParameterValue(↵
        AcousticsPlugin.PARAMETER_SOUND_REDUCTION_INDEX,
        soundReductionIndex);
07:     window.setParameterValue(↵
        TenementContractPlugin.PARAMETER_NAME_PIECECOST, cost);
08:     scenario.addReplacementPart(window);
09:     return window;
10: }
```

Quellcode 11: Methode zur Erzeugung eines Ersatzbauteils

(Spieler-)Aktionen:

Dem Spieler muss die Aktion angeboten werden, Fenster auszuwechseln. Im Eclipse-RCP-basierten Prototyp funktioniert dies über den Extension Point `org.eclipse.ui.popupMenus`¹⁴ und einer sogenannten `objectContribution`:

```
01: <extension point="org.eclipse.ui.popupMenus">
02:     <objectContribution
03:         adaptable="false"
```

¹⁴ Die Dokumentation für den Extension Point `popupMenus` findet sich unter <http://help.eclipse.org/indigo/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Freference%2Fextension-points%2Forg.eclipse.ui.popupMenus.html> (letzter Zugriff: 05.08.2012)

```
04:      id="edu.uni_weimar.simuframe.tenementbuilding.ui.WindowInstaller"
05:      objectClass="edu.uni_weimar.simuframe.interfaces.element.ISimuElement">
06:      <action
07:          class="edu.uni_weimar.simuframe.tenementbuilding↵
08:              .ui.useractions.WindowInstaller"
09:          enablesFor="+↵
10:          id="edu.uni_weimar.simuframe.tenementbuilding↵
11:              ui.useractions.WindowInstaller"
12:              label="Install news windows"
13:              menubarPath="edu.uni_weimar.simuframe.tenementbuilding↵
14:                  .ui.popupMenu"
15:                  tooltip="Installs new windows">
16:      </action>
17:      <visibility>
18:      <and>
19:      <or>
20:      <objectClass
21:          name="edu.uni_weimar.simuframe.tenementbuilding↵
22:              .element.Tenement">
23:      </objectClass>
24:      <objectClass
25:          name="edu.uni_weimar.simuframe.tenementbuilding↵
26:              .element.Building">
27:      </objectClass>
28:      </or>
29:      </and>
30:      </visibility>
31:  </objectContribution>
32: </extension>
```

Quellcode 12: plugin.xml: Definition eines Kontext-Menüeintrag als Benutzeraktion

Dieser Ausschnitt aus der Datei `plugin.xml` ist ein geeignetes Beispiel, um die Mächtigkeit des Eclipse RCP-Frameworks zu zeigen: Deklarativ, d.h. ohne Programmierung wird hier in der Oberfläche ein Kontextmenüeintrag hinzugefügt und mit einer Aktion verbunden. In Zeile 5 wird definiert, dass dieses Popup-Menü für alle Objekte des Typs `ISimuElement` gilt, also für alle Simulationselemente, da sie dieses Interface implementieren. Zeile 7 beschreibt die Klasse, in der die Aktion definiert ist. Zeile 10 legt den Text fest, der im Menü erscheint, Zeile 12 den Tooltip-Text. In Zeile 11 wird das Menü ausgewählt, d.h. festgelegt, in welcher Menüebene und gegebenenfalls in welchem Untermenü dieser Eintrag erscheint. Ab Zeile 14 wird definiert, wann dieser Menüeintrag sichtbar ist: wenn der Kontext entweder zu einem `Tenement` (Zeile 18) oder (Zeile 16) zu einem `Building` (Zeile 21) gehört und (Zeile 15) für dieses Objekt das Attribut `isElementofRunningScenario` den Werte `true` besitzt (Dieser Kontext-Menü erscheint nur dann, wenn das Szenario gerade simuliert wird.)¹⁵. Abbildung 4 zeigt das erstellte Kontext-Menü für das Objekt `Tenement` in der Baumdarstellung des Simulationsmodells im Szenario-Editor.

¹⁵ Das Attribut `isElementofRunningScenario` ist kein Attribut einer Klasse, sondern wird mit Hilfe der Framework Klassen `ActionFilterAdapterFactory` und `SimuElementActionFilter` und des Adapter Patterns (s. Kap. 2.2.4) dynamisch der aktuellen Klasse hinzugefügt.

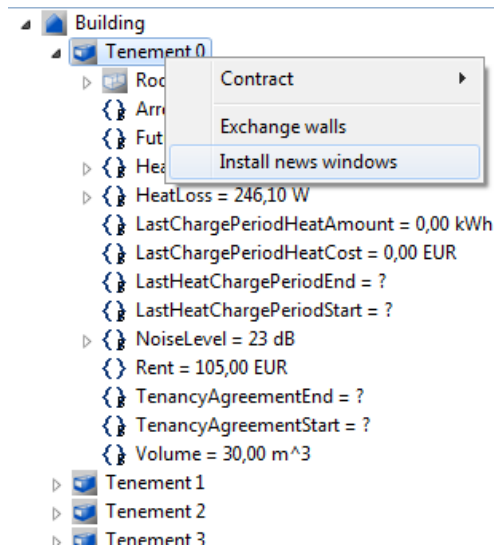


Abbildung 4: Szenario-Editor: Kontext-Menü

In der Aktion selbst erfolgt die Verbindung zwischen Benutzeroberfläche und Prozedur: Über die Benutzeroberfläche wird die Art des einzusetzenden Fensters abgefragt. Das Ergebnis wird an die Prozedur weitergegeben:

```
01: window = askWindowType(scenario);
02: if (window != null) {
03:     CmdInstallWindow cmd = new CmdInstallWindow(tenementOrBuilding, ↵
        window.getName());
04: }
04: cmd.execute();
```

Quellcode 13: Aufruf der Prozedur zur Installation der Fenster

Dieser Ausschnitt aus der Action `WindowInstaller` zeigt das Vorgehen: In Zeile 1 wird die Prozedur `askWindowType(...)` aufgerufen, die einen Dialog öffnet, in der der Typ der einzubauenden Fenster ausgewählt werden kann (s. nachfolgenden Abschnitt „Auswahldialog“). Zurückgegeben wird ein Identifikator des ausgewählten Fenstertyps. Nachfolgend wird die Prozedur `CmdInstallWindow` aufgerufen. Diese simuliert den Austausch der Fenster in der übergebenen Wohneinheit (Gebäude oder Wohnung).

Auswahldialog:

Der Einbau anderer Simulationselemente erfordert deren Auswahl durch den Spieler. Dies bedingt eine Interaktion über die Benutzeroberfläche. In der IDE des Szenario-Editors erfolgt dieses mit Hilfe eines Dialoges (s. Abbildung 5). Der Dialog ist generisch für ein Simulationselement angelegt: Er zeigt einen für die konkrete Klasse des Simulationselements spezifischen Text („Choose a Window“). Die verschiedenen Typen der Simulationselemente sind in einem Listenfeld „Type“ enthalten. Wählt der Spieler einen Typ aus, werden die Attribute dieses Typs angezeigt: eine Beschreibung sowie die Parameter.

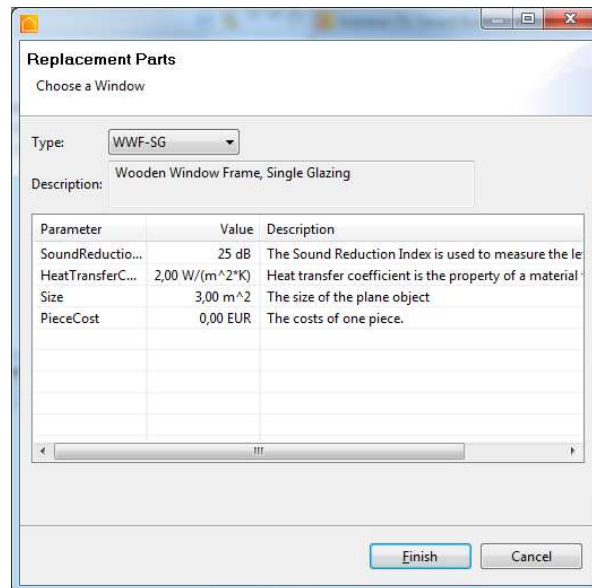


Abbildung 5: Dialog zum Austausch der Fenster in einem Szenario

Im folgenden Quellcode 14 ist die Methode `askWindowType(...)` dargestellt, deren Aufruf schon im vorigen Abschnitt besprochen wurde.

```

01: Window askWindowType(IScenario scenario) {
02:     Wizard wizard = new Wizard(SimuElementManager.getInstance().←
        getSimuElementId(Window.class) , scenario);
03:     WizardDialog dlg = new
        WizardDialog(Display.getCurrent().getActiveShell(),←
        wizard);
04:     dlg.create();
05:     int retCode = dlg.open();
06:     if (retCode == Dialog.OK) {
07:         result = (Window) wizard.getChoosenSimuElement();
08:     }
09:     return result;
10: }

```

Quellcode 14: Funktion zur Abfrage des Fenstertyps

Die Klasse `wizard` in Zeile 2 wurde abgeleitet von einer Eclipse-RCP-Framework-Klasse¹⁶ und definiert den Aufbau des anzuzeigenden Dialogs. Da die Ersatzbauelemente mit dem Szenario verknüpft sind, können innerhalb des Dialoges die anzuzeigenden Informationen ermittelt werden (Das Szenario wird als Parameter `scenario` übergeben.). Die Klasse `wizardDialog` in Zeile 3 ist eine Eclipse RCP-Framework-Klasse, die für den Dialogaufbau und die Dialoganzeige und –beendigung zuständig ist. Die explizite Typumwandlung in Zeile 7 (`(Window)`) und der Methodenname `getChoosenSimuElement()` deuten auf die generische Konzeption des Dialogs hin: Es werden Simulationselemente (`ISimuElement`) dargestellt. Zusammenfassend zeigt sich hier wieder, dass auch die Verknüpfung mit der Benutzeroberfläche zu einem großen Teil von der Funktionalität des zugrundeliegenden Eclipse RCP-Frameworks übernommen werden kann.

f) Grenzen und Erweiterungsmöglichkeiten

Die dargestellte Lösung enthält einen großen Anteil an Benutzeroberfläche, der im Szenario-Editor realisiert wurde. Die Spieleraktionen sind jedoch auch im Gaming Client notwendig. Um das zu erlauben, wäre es möglich, die Konfigurationsdaten der Spieleraktionen auszulesen und

¹⁶ `org.eclipse.jface.wizard.Wizard`

vom Server an den Gaming-Client weiterzuleiten. Damit wäre eine einheitliche Datenbasis gewährleistet und die Einhaltung des DRY¹⁷-Prinzips gewährleistet: eine Spieleraktion muss nur an einer Stelle definiert werden.

Im Moment gibt es keine generelle Austauschmöglichkeit von Simulationselementen. Für jede Klasse von Simulationselementen muss diese explizit konfiguriert und programmiert werden. Denkbar ist es jedoch, eine Änderungsaktion für jede Simulationselementklasse anzubieten, die im Simulationsmodell enthalten ist. Hier äußert sich ein Vorteil der Simulation: Simulationselemente können ausgetauscht werden, auch wenn das in der Realität nicht oder nur schwer möglich wäre.

Zurzeit werden die Ersatzbauteile durch Quellcode generiert. Mehr Flexibilität ergäbe sich, wenn diese aus Datenbanken gelesen werden würden.

2.3.6 Verbindungsmodule

a) Konkretes Problem

Das Modul `Heat` (Wärme) definiert das Simulationselement `HeatingSystem` (Heizung) sowie die Benutzeraktion `OrderFuel` (Brennstoff kaufen). Diese Benutzeraktion hat Auswirkungen auf die Kosten (also das Modul `TenementContract` (Vertrag)), zumindest müssen die durch den Kaufvorgang entstehenden Kosten verbucht werden. Eine erste Lösung wäre, das Modul `TenementContract` abhängig zu machen vom Modul `Heat`. Das ist aber nicht sinnvoll: Jedes fachliche Modul wäre nach demselben Schema dann Voraussetzung für das Modul `TenementContract`. Mit zunehmender Anzahl an fachlichen Modulen funktioniert das Modul `TenementContract` nur noch, wenn alle fachlichen Module ebenfalls vorhanden sind. Das aber untergräbt die Ziele der Modularisierung. Beispielsweise müsste bei Hinzufügen eines neuen fachlichen Moduls ein bestehendes (nämlich hier `TenementContract`) geändert werden, was die Stabilität dieses Moduls potentiell gefährden würde.

b) Generelles Problem

Lose Kopplung und enge Bindung gehören zu den wichtigen Grundsätzen der Modularisierung von Software (McConnell, 2005). Lose Kopplung bedeutet, dass Module im Wesentlichen über definierte Schnittstellen miteinander kommunizieren und dass modulinterne Änderungen keine Änderungen in anderen Modulen nach sich ziehen, solange die Schnittstellen unverändert bleiben. Enge Bindung innerhalb eines Moduls (*Kohäsion*) sorgt dafür, dass das Modul eine logische Einheit bildet und verständlicher bleibt. Somit bleibt die Wartung des Codes einfacher, da fachliche Änderungen weniger Module betreffen (Yourdon & Constantine, 1979). Bezogen auf die Spielplattform bedeutet dies, dass die Einführung eines neuen fachlichen Aspekts durch ein Modul nicht eine wesentliche fachliche Änderung bzw. Erweiterung eines anderen Modules nach sich ziehen sollte. Zusätzliche Module sind vorzuziehen.

c) Lösungsansatz

Die logische Verbindung der beiden Module erfolgt über ein sogenanntes Verbindungsmodul. Das heißt, das Modul `Heat` und das Modul `TenementContract` bilden die Grundlage für das

¹⁷ DRY: **Don't repeat yourself** – ein Grundsatz in der Softwareentwicklung: „Jedes Stück Wissen muss eine einzige, eindeutige und maßgebliche Repräsentation in einem System haben.“ (Hunt & Thomas, 2003)

Modul `TenementContract_Heat`. Dieses Modul nimmt die fachlichen Aspekte auf, die auf beiden Modulen gleichzeitig aufbauen.

d) Implementierung

Modulabhängigkeiten

Das Modul `TenementContract_Heat` ist das Verbindungsmodul der beiden Module `TenementContract` und `Heat` und setzt diese beiden Module daher voraus.

Aufgaben des Moduls

- **Definition von Parametern:** Das Modul definiert Parameter, die der Abrechnung von Heizkosten dienen und unterschiedlichen Simulationselementen zugeordnet sind.
- **Definition von Prozeduren:** Die Prozeduren dienen der Verarbeitung auftretender Systemereignisse. Beispielhaft wird die Arbeitsweise der Prozedur `BookFuelPurchase` erläutert.

Parameter	SimuElement	Beschreibung
<code>FuelCost</code>	<code>Environment</code>	Spezifische Kosten des Brennstoffes
<code>LastChargePeriodHeatAmount</code>	<code>Tenement</code>	In der letzten Heizperiode verbrauchte Wärmemenge
<code>LastChargePeriodHeatCost</code>	<code>Tenement</code>	Betrag der für die letzte Heizperiode abgerechneten Kosten
<code>LastHeatChargePeriodEnd</code>	<code>Tenement</code>	Ende der zuletzt abgerechnete Heizperiode
<code>LastHeatChargePeriodStart</code>	<code>Tenement</code>	Start der zuletzt abgerechneten Heizperiode
<code>SpecificHeatCost</code>	<code>HeatingSystem</code>	Kosten der erzeugten Wärme pro Einheit

Tabelle 4: Parameter im Modul `TenementContract_Heat`

Prozedur „BookFuelPurchase“

Die Aufgabe dieser Prozedur ist es, den Kauf von Brennstoff finanziell zu verarbeiten. Angestoßen wird diese Prozedur durch ein Systemereignis, das durch die Befüllung des Brennstofftanks ausgelöst wird. Die Befüllung wird durch den Spieler initiiert, sie wird durch die Prozedur `CmdFillFuelReservoir` in der Simulation umgesetzt. In der Prozedur wird ein Systemereignis `EVENT_POST_FUEL_TANK_FILLMENT` erzeugt (Wirkungsmechanismus wie in Kap. 2.3.4. beschrieben): Für dieses Ereignis ist die Prozedur `BookFuelPurchase` als Zuhörer eingetragen. In der werden die Kosten des Brennstoffs ermittelt und vom aktuellen Kassenstand des Gebäudes abgezogen:

```
01: public void simuElementEventOccured(ISimuElementEvent event) {
02:     HeatingSystem source = (HeatingSystem)event.getSource();
03:     double price = getPriceForFuel(source);
04:     Double amount = getFuelAmount(event);
05:     Double totalcost = amount * price;
06:     ParameterDouble cash = getCash(source);
07:     ParameterUtils.subtract(cash, totalcost);
08: }
```

Quellcode 15: Ereignisprozedur `EVENT_POST_FUEL_TANK_FILLMENT`

In Zeile 2 wird das Objekt vom Typ `HeatingSystem` ermittelt. Dies ist die Quelle des Ereignisses, gehört zum Modul `Heat` und dient als Element des Simulationsmodells dazu, das Simulationselement `Environment` aufzufinden. Das besitzt den Parameter `FuelCost` (wie oben beschrieben, ein Beitrag des Verbindungsmoduls `TenementContract_Heat`), dessen Wert in Zeile 3 über die

Methode `getPriceForFuel()` ermittelt wird. Der Wert des Parameters `cash` (aus dem Modul `TenementContract`) wird in Zeile 7 um die Kosten für die Tankfüllung vermindert und neu gesetzt.

2.4 Designaspekte zur Unterstützung von Simulation

Abseits von der reinen informationstechnischen Umsetzung gibt es auch generelle Anforderungen aus dem Bereich der Simulation, die Designentscheidungen notwendig machen. Diese werden exemplarisch in den folgenden Abschnitten beschrieben.

2.4.1 Automatisierte Erzeugung von individualisierten Simulationselementen

a) Konkretes Problem

Das Spielkonzept beruht in wesentlichen Punkten auf unterschiedlichen Bedürfnissen verschiedener virtueller Mieter (als NPC modelliert), die damit gegebenenfalls in Konflikt mit ihrer Umgebung gelangen – und somit Problemlösungsbedarf für den Spieler erzeugen. Damit dieses Konzept aufgehen kann, ist die automatisierte Erzeugung von Mietern so zu gestalten, dass Mieter mit unterschiedlichen Bedürfnisprofilen entstehen.


b) Generelles Problem

Es ist möglich, aber aufwändig, ein Szenario komplett vom ersten Simulationselement bis zum letzten Parameter manuell zu erstellen, d.h. beispielsweise Programmcode zu schreiben oder Konfigurationsdateien zu editieren. Eine weniger aufwändige Variante ist es, ausgehend von einer Grundkonfiguration zufallsbasierte Abweichungen in das Szenario einzubauen. Auch hiermit lassen sich voneinander abweichende Probleme erstellen. Mit Hilfe unterschiedlicher Probleme soll vermieden werden, dass bei Wiederholungen der Lösung die notwendigen Schritte auswendig gelernt werden. Stattdessen soll der Spieler jedes Szenario fachlich analysieren und auf dieser Grundlage die notwendigen Maßnahmen zur Lösung erarbeiten.

c) Lösungsansatz

Die hier vorgestellte Lösung bezieht sich auf die Generierung des Simulationselementes `Tenant`. Die Mieter der Spielplattform unterscheiden sich durch die Werte verschiedener Parameter, hingegen ist die Struktur immer gleich¹⁸. Abbildung 6 zeigt die Darstellung eines potentiellen Mieters aus der Baumdarstellung des Simulationsmodells im Szenario-Editor. Der Mieter (`Tenant`) wird charakterisiert durch Parameter, die in Tabelle 5 dargestellt werden.

¹⁸ Dieser Lösungsansatz zeigt nur einige grundlegende Prinzipien auf, eine tiefergehende Untersuchung führt Thomas Bröker in seiner parallel zu dieser Arbeit entstehenden Dissertation durch (Bröker, 2013).

 Brynne Goelz

- { } AnzahlMietparteiMitglieder = 2
- { } Archetype = Paar
- { } ArchetypeDetail = Arbeitsuchend
- { } ComfortableHeatLevel = 23,0 °C
- { } ComfortableNoiseLevel = 61 dB
- { } Einkommen = 24890,00 EUR
- { } HeatLevelSatisfaction = 0,00
- { } Lueftungsverhalten = 0,88
- { } Milieu = DDR-Nostalgische
- { } NoiseLevelSatisfaction = 0,00
- { } RentalPriceSatisfaction = 0,00
- { } SourceNoiseLevel = 60 dB
- { } SustainableRent = 108,00 EUR
- { } TenantSatisfaction = 0,00

Abbildung 6: Mieter in Simulationsmodell des Szenario-Editors

Parameter	Modul	Beschreibung
AnzahlMietpartei-Mitglieder	Tenant	Anzahl der Personen in der Wohnung
Archetype	Tenant	Art der Mietparte (Single, Paar, Kleinfamilie, Großfamilie)
ArchetypeDetail	Tenant	Primäre Erwerbsquelle der Mietpartei (Ruhestand, Arbeitend, In Ausbildung, Arbeitsuchend) (s. Archetype)
ComfortableHeatLevel	Heat	Bevorzugte Temperatur, Abweichungen von dieser Temperatur senken die spezifische Zufriedenheit. (s. HeatLevelSatisfaction)
ComfortableNoiseLevel	Acoustics	Bevorzugter Geräuschpegel, ein höherer Geräuschpegel senkt die spezifische Zufriedenheit (s. NoiseLevelSatisfaction)
Einkommen	Tenant	Einkommen des Mieters
HeatLevelSatisfaction	Heat	Spezifische Zufriedenheit mit der Temperatur in der Wohnung
Lueftungsverhalten	Tenant	Anzahl der Stoßlüftungen pro Tag
Milieu	Tenant	Kategorisierung nach Sinus®-Milieus ¹⁹
NoiseLevelSatisfaction	Acoustics	Spezifische Zufriedenheit mit dem Geräuschpegel in der Wohnung.
RentalPriceSatisfaction	Tenement-Contract	Spezifische Zufriedenheit mit der Miete für die Wohnung.
SourceNoiseLevel	Acoustics	Geräuschpegel, für den der Mieter verantwortlich ist.
SustainableRent	Tenement-Contract	Höhe der Miete, die der Mieter nachhaltig bereit ist zu zahlen. Höhere Mieten senken die spezifische Zufriedenheit (s. RentalPriceSatisfaction).
TenantSatisfaction	Tenement	Generelle Zufriedenheit des Mieters als Ergebnis der spezifischen Zufriedenheiten (hier: RentalPriceSatisfaction, NoiseLevelSatisfaction, HeatLevelSatisfaction)

Tabelle 5: Simulationselement Tenant: Parameter

¹⁹ Mit Hilfe von Sinus-Milieus werden Menschen nach Lebensauffassung und Lebensweise kategorisiert. Eine stetige Beobachtung der Gruppen und Aktualisierung der Kategorien erfolgt durch das Sinus-Institut, Heidelberg (<http://www.sinus-institut.de/>, letzter Zugriff 12.09.2012).

Betrachtet man die Parameter des `Tenants` in Tabelle 5, so fällt auf, dass einige davon berechnet werden (`HeatLevelSatisfaction`, `NoiseLevelSatisfaction`, `RentalPriceSatisfaction` und `TenantSatisfaction`) und daher nicht bei der Generierung eines Mieters belegt werden müssen. Die Werte der weiteren Parameter sollen generiert werden. Im Folgenden werden grundsätzliche Prinzipien, die dabei genutzt werden, beschrieben.

I Verteilungen

Grundlegendes Ziel bei der Generierung von Parameterwerten ist es, dass die erstellten Werte ein Abbild der Wirklichkeit sind. Es wird angenommen, dass die Werte von Parametern stochastisch mit Hilfe von Verteilungsfunktionen beschrieben werden können. Es gibt unterschiedliche Verteilungen²⁰, für die es jeweils einen charakteristischen Satz von Parametern²¹ gibt. Die wohl am häufigsten verwendete der stetigen Verteilungsfunktionen ist die Normalverteilung, deren Dichtefunktion auch als *Gauß'sche Glockenkurve* bezeichnet wird. Sie hat zwei Parameter: μ bezeichnet den Erwartungswert und σ^2 steht für die Varianz (Semendjajew et al., 2008).

Ist nun ein Parameter gegeben, so ist festzulegen, welcher Verteilung er unterliegt und welche Werte die Verteilungsparameter besitzen. Für die betrachteten Parameter (`ComfortableHeatLevel`, `ComfortableNoiseLevel`, `SustainableRent`²²) wurden die Normalverteilung und die Gleichverteilung gewählt. Zusätzlich wurde eine allgemeine Wahrscheinlichkeitsfunktion einer diskreten Zufallsgröße gewählt, d.h. jedem annehmbaren Wert kann eine eigene Wahrscheinlichkeit zugeordnet werden, mit der dieser Wert angenommen wird, alle Einzelwahrscheinlichkeiten addieren sich zu 1.

II Profile/Abhängige Parameterwerte

Ausgangspunkt der Überlegungen, die zu Profilen geführt haben, war die nicht immer gegebene fehlende Unabhängigkeit der verschiedenen Parameter. In Abhängigkeit vom Wohlstand variieren beispielsweise Ansprüche und Toleranzgrenzen der Mieter für die verschiedenen Parameter (Bröker, 2013).

Eine Möglichkeit ist es, Kategorien von Mietern zu bilden, und diesen Profile zuzuweisen. Profile sind als bestimmte Mengen von Parametern zu verstehen, denen jeweils eine passende Verteilungsfunktion mit geeigneten Verteilungsparametern zugeordnet wurde. Dieser Ansatz wurde zunächst realisiert. Da die Plattform aber erweiterbar ist und so Parameter hinzukommen und wegfallen können, wurde letztendlich die Lösung der voneinander abhängigen Parameter gewählt. Dies bedeutet, dass bei der Erzeugung von Parameterwerten die Werte anderer Parameter herangezogen werden.

²⁰ Für eine Zufallsgröße X ist die Verteilungsfunktion $F(x)$ definiert als $F(x) = P(X < x)$ mit P als Wahrscheinlichkeit für das Ereignis $X < x$. In Worten ausgedrückt :

„Der Wert der Verteilungsfunktion an der Stelle x_0 ist also gleich der Wahrscheinlichkeit dafür, dass die Zufallsgröße einen Wert annimmt, der kleiner als x_0 ist.“ (Semendjajew et al., 2008, S. 660)

Es werden stetige und diskrete Zufallsgrößen unterschieden. Für stetige Zufallsgrößen ist die Verteilungsfunktion das Integral der Dichtefunktion.

²¹ Parameter im Sinne von Koeffizient/Kennzahl im Gegensatz zu den vorher genannten Simulationsparametern.

²² Für die Höhe der Miete wird vereinfachend angenommen, dass diese unabhängig von der Anzahl der Personen der Mietpartei festgelegt ist.

d) Implementierung

Die Belegung der Parameter mit initialen Werten erfolgt einmal zu Beginn des Simulationslaufs. In Abbildung 7 sind die grundlegenden Objektbeziehungen dargestellt: Ein Parameter kann mit einem `InitialValueProvider` verknüpft sein. Dieser generiert zufallsbasierte Werte gemäß einer auszuwählenden und zu konfigurierenden Verteilung.

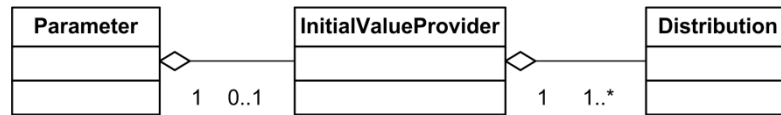


Abbildung 7: Grundsätzliches UML-Klassendiagramm: Parameter

Im Folgenden werden dazu einzelne Aspekte der technischen Grundlagen vorgestellt.

I Initiale Parameterwerte

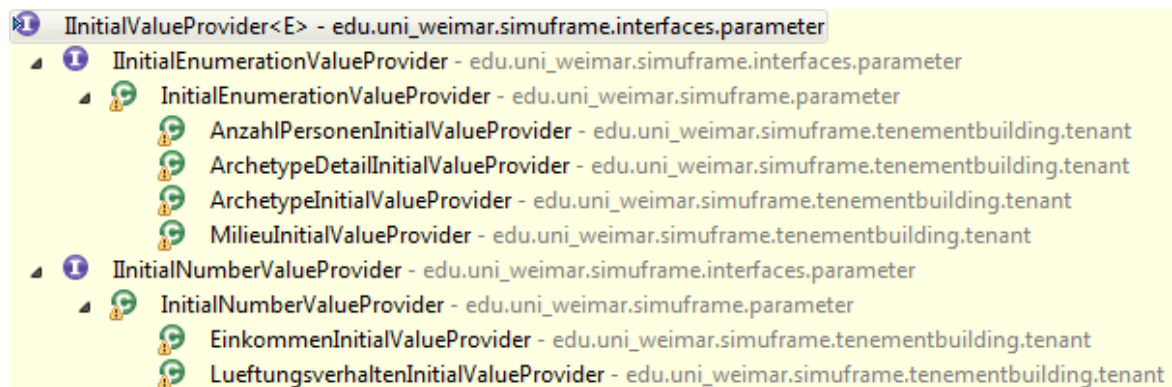
Für die Belegung mit einem initialen Wert gibt es das Attribut `initialValueProvider` im Extension Point `simuElementParameter`, mit dessen Hilfe ein Parameter deklariert wird. Dieses Attribut wird mit einer Klasse belegt, die das Interface `IInitialValueProvider` implementiert (s. Quellcode 16, Zeile 9). Das Interface enthält die Methode `createValue(...)`. Sie wird aufgerufen im Laufe der Erzeugung eines Parameters und liefert einen initialen Wert zurück.

```

01: <simuElementParameter
02:     defaultValue="24000"
03:     parameterId="Einkommen"
04:     unit="EUR">
05:   <simuElement
06:       simuElementId="Tenant">
07:     </simuElement>
08:     <initialValue2 initialValueProvider="EinkommenInitialValueProvider">
09:   </initialValue2>
10: </simuElementParameter>
  
```

Quellcode 16: Deklaration eines Parameters in der plugin.xml: `initialValueProvider`

Parameter haben jeweils einen expliziten Typ, d.h. es gibt Parameter mit einem numerischen Wert (z.B. Einkommen, TenantSatisfaction) und es gibt Parameter vom Typ Zeichenfolge. Weiterhin kann der Wertebereich kontinuierlich oder diskret sein. In Abhängigkeit hiervon ist der initiale Wert zu wählen. Um diese Anforderung zu erfüllen, wurden verschiedene Klassen und Interfaces entwickelt, die untereinander in einer Vererbungshierarchie (s. Abbildung 8) in Beziehung stehen.

Abbildung 8: Vererbungshierarchie `IInitialValueProvider`

Das Interface `IInitialValueProvider` steht am Anfang der Hierarchie. Von diesem abgeleitet werden zwei weitere Interfaces: `IInitialEnumerationValueProvider` und `IInitialNumberValueProvider`. Diese unterscheiden sich im Typ des erzeugten Wertes: `IInitialEnumerationValueProvider` gibt einen Wert aus einer Aufzählung zurück, der zurückgegebene Wert hat den Typ `String` (Zeichenfolge). Hingegen dient `IInitialNumberValueProvider` zur Erzeugung eines numerischen Wertes (`Number`). Von beiden Interfaces ist jeweils eine Klasse abgeleitet. Aufgabe dieser Klassen ist es, interne Variablen mit Objekten (z.B. Verteilungen, vgl. Kap. 2.4.1d)) zu belegen und sie allen abgeleiteten Klassen zur Verfügung zu stellen.

II Verteilungen

Zur Erzeugung von initialen, zufallsverteilten Parameterwerten werden Verteilungen eingesetzt. Da - wie schon im vorigen Kapitel beschrieben - diskrete und kontinuierliche Werte generiert werden müssen, entspricht die Vererbungshierarchie die der `IInitialValueProvider` (Abbildung 9): Elementares Interface ist `IDistribution`. Dieses enthält die Methode `createValue()`, die einen mit Hilfe eines Zufallsgenerators erzeugten Wert zurückgibt. In dem Interface ist der Typ des erzeugten Wertes noch nicht festgelegt, er ist generisch. Die von dieser Methode zurückgelieferten Werte unterliegen in ihrer Gesamtheit einer Verteilung. Die Verteilung und der Typ des zu erzeugenden Wertes werden erst in der konkreten Klasse festgelegt.

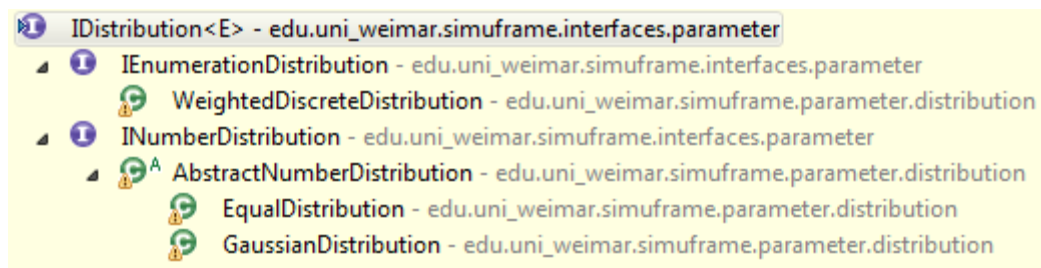


Abbildung 9: Vererbungshierarchie `IDistribution`

Abgeleitet vom `IDistribution` werden zwei Interfaces, die eine Spezialisierung bezüglich diskreter und kontinuierlicher Verteilung vornehmen: Das erste Interface, `IEnumerationDistribution`, liefert Werte entsprechend einer kontinuierlichen Verteilung zurück. Der Unterschied zum übergeordneten Interface ist, dass nun der Typ des Rückgabewert festgelegt wird: Die Methode ist deklariert als `String createValue()` (im Vergleich zu `E createValue()` mit `E` als generischem Typ(-platzhalter)). Dieses Interface wird implementiert von der Klasse `WeightedDiscreteDistribution`.

Das zweite Interface wird `INumberDistribution` genannt und modelliert die Verteilung einer kontinuierlichen Zufallsgröße. Über die abstrakte Klasse `AbstractNumberDistribution`, die zur Aufnahme gemeinsamer Implementierungsdetails eingeführt wurde, erben zwei konkrete Klassen: `EqualDistribution` liefert Werte entsprechend der Gleichverteilung zurück, `GaussianDistribution` simuliert die Normalverteilung.

Im Folgenden werden für alle drei Klassen Implementierungsdetails dargestellt:

WeightedDiscreteDistribution

Diese Klasse erstellt Werte einer diskreten Zufallsgröße. Sie kennt eine Liste der diskreten Werte, die den Typ `String` besitzen. Jedem Wert ist ein numerischer Wert („Gewicht“) zugeordnet,

der die Wahrscheinlichkeit für das Auftreten dieses Wertes angibt. Die Summe der Gewichte muss nicht 1 ergeben, bei der zufallsgesteuerten Wertgenerierung werden sie normalisiert.

EqualDistribution

Diese Klasse versorgt eine kontinuierliche numerische Zufallsgröße mit Werten. Der Wertebereich lässt sich durch einen Minimal- und einen Maximalwert einschränken. Die zugehörige Dichtefunktion ist konstant.

GaussianDistribution

Auch diese Klasse bedient eine kontinuierliche numerische Zufallsgröße. Als Parameter dienen ebenfalls ein Minimal- und ein Maximalwert sowie der Durchschnittswert. Bei der Ermittlung des Zufallswertes erfolgt eine lineare Abbildung des errechneten, annähernd $N(0, 1)$ normalverteilten Wertes auf die Intervalle [Minimalwert, Mittelwert] sowie [Mittelwert, Maximalwert], je nachdem ob der Wert kleiner oder größer als 0 ist²³.

III IInitialValueProvider und Verteilungen

Die Verbindung zwischen `IInitialValueProvider` und `IDistribution` besteht darin, dass einem `IInitialValueProvider` eine oder mehrere Instanzen vom Typ `IDistribution` zugewiesen werden können. Abhängig von der Konfiguration wird die entsprechende Verteilung bei der Erzeugung eines Parameterwertes genutzt. In Quellcode 17 ist eine solche Konfiguration dargestellt: Innerhalb einer Extension des Extension Points `simuElementParameter` werden über das Element `initialValue` die Parameter für die Verteilung angegeben (Zeile 6 bis 12). In Zeile 8 wird die Verteilung gewählt, in Zeile 9 und 10 werden Maximal- und Minimalwert genannt und in Zeile 7 wird der Erwartungswert definiert.

```
01: <simuElementParameter
02:     parameterId=
03:         "edu.uni_weimar.simuframe.tenementbuilding.heat.ComfortableHeatLevel">
04:     <simuElement
05:         simuElementId="edu.uni_weimar.simuframe.tenementbuilding.Tenant">
06:     </simuElement>
07:     <initialValue
08:         averageValue="22"
09:         distribution="Gaussian"
10:         maxValue="26"
11:         minValue="18">
12:     </initialValue>
13: </simuElementParameter>
```

Quellcode 17: Konfiguration eines initialen Wertes in der plugin.xml

IV Abhängige Parameter

Die Generierung von Werten für Parameter, deren initiale Werte in Abhängigkeit von den Werten anderer Parameter erzeugt werden, erfolgt durch der Zuweisung von Verteilungen zu Gruppen von Parametern. Prinzip ist, dass die Parameter, deren Werte zur Selektion der passenden Verteilung genutzt werden, schon belegt sein müssen. Im Quellcode 18 (ein Ausschnitt aus der

²³ Hier liegen Abweichungen zur herkömmlichen Beschreibung einer Normalverteilung vor: Diese kennt als Parameter den Erwartungswert μ und die Standardabweichung σ . Auch die Berechnung erfolgt mit der Simulation einer Normalverteilung durch die Addition von 12 Zufallswerten nach einer Heuristik („Add 12 uniform numbers“, <http://stackoverflow.com/questions/2325472/generate-random-numbers-following-a-normal-distribution-in-c-c/2325913#2325913>, zuletzt abgerufen am 10.10.2010). Diese Vorgehensweise scheint zu funktionieren, jedoch bietet es sich mit dem Stand meines heutigen Wissens an, auf Standardbibliotheken zurückzugreifen (z.B. liefert in Java `Random.nextGaussian()` das gewünschte Ergebnis).

Klasse `EinkommenInitialValueProvider`) ist das Vorgehen beispielhaft dargestellt: Der initiale Wert des Parameters Einkommen hängt von drei weiteren Parametern ab (`Archetype`, `ArchetypeDetail` und `Milieu`²⁴). In den Zeilen 2 bis 4 werden diese Parameterwerte bestimmt. Mit diesen Werten wird die passende Verteilungsfunktion gewählt in der Zeile 6 – mit Hilfe der Funktion `getLocalDistribution(...)`, die in den Zeilen 10 – 19 definiert ist. Die verschiedenen Verteilungen werden mit Hilfe der Methode `initDistributions()` erzeugt: Sie werden in einer `Hashtable`²⁵ gespeichert: Der Schlüssel wird aus den Werten der drei Parameter generiert und dient zum Auffinden (Methode `getLocalDistribution(...)`) des entsprechenden Verteilungsobjekts. Gibt es zu einem gegebenen Schlüsselwert kein explizites Verteilungsobjekt, wird eine Standardverteilung genutzt (s. Zeilen 14 bis 17). Die nicht abgebildete Methode `getKey(...)` erzeugt aus den drei Parameterwerten einen Schlüsselwert. In den Zeilen 28 bis 31 wird demonstriert, wie für einen Schlüsselwert ein eigenes Verteilungsobjekt abgespeichert wird.

```

01: public Long createInitialValue() {
02:     String archetypeValue = ↵
        getParameterValue(ArchetypeInitialValueProvider.ARCHETYPE);
03:     String archetypeDetailValue = getParameterValue(↵
        ArchetypeDetailInitialValueProvider.ARCHTYPE_DETAIL);
04:     String milieuValue = getParameterValue(MilieuInitialValueProvider.MILIEU);
05:
06:     Number result = getLocalDistribution(archetypeValue, ↵
        archetypeDetailValue, milieuValue).createValue();
07:     return result.longValue();
08: }
09:
10: private INumberDistribution getLocalDistribution(String archetypeValue, ↵
        String
        archetypeDetailValue, ↵
        String milieuValue) {

11:     if (_localDistributions.size() == 0) {
12:         initDistributions();
13:     }
14:     INumberDistribution distribution = _localDistributions.get(↵
        getKey(archetypeValue, archetypeDetailValue, milieuValue));
15:     if (distribution == null) {
16:         distribution = _localDistributions.get("Default");
17:     }
18:     return distribution;
19: }
20: private void initDistributions() {
21:     setDistributionStr(DISTRIBUTION_NAME_GAUSSIAN);
22:
23:     GaussianDistribution distribution = ↵
        (GaussianDistribution) createDistribution();
24:     distribution.setAverageValue(24000);
25:     distribution.setMaximumValue(100000);
26:     distribution.setMinimumValue(1000);
27:
28:     _localDistributions.put(getKey(ArchetypeInitialValueProvider.AT_GROSSFAMI
IE,
29:         ArchetypeDetailInitialValueProvider.ATD_ARBEITEND,
30:         MilieuInitialValueProvider.DDR_NOSTALGISCHE
31:         ), distribution);
32:
33:     distribution = (GaussianDistribution) createDistribution();

```

²⁴ Zur Bedeutung s. Tabelle 5: Simulationselement Tenant: Parameter.

²⁵ Eine `Hashtable` ermöglicht es, Schlüssel-Wert-Paare abzuspeichern. Der Wert kann dann mit Hilfe des Schlüssels wieder gelesen werden.

```
34:    distribution.setAverageValue(23000);
35:    distribution.setMaximumValue(120000);
36:    distribution.setMinimumValue(300);
37:
38:    _localDistributions.put("Default", distribution);
39: }
```

Quellcode 18: Auszug aus der Klasse EinkommenInitialValueProvider

V Namensparameter

Das zu generierende Simulationselement `Tenant` besitzt das Attribut `name`. Dieses soll ebenfalls mit einem sinnvollen Wert gefüllt werden, d.h. einem Namen, der als solcher zu erkennen ist. Die Lösungsidee sieht vor, dass Listen von Nachnamen und Vornamen erstellt werden. Wird ein neuer Name generiert, dann wird aus jeder Liste ein Eintrag zufällig gewählt, die Kombination wird als Name zurückgegeben.

Zur Speicherung der Namen wird jeweils eine Textdatei genutzt, die komplett in den Speicher eingelesen wird. Damit ist dann wahlfreier Zugriff (Methode `getRandomEntry(...)` in den Zeilen 6 bis 10 in Quellcode 19) auf einen Eintrag der in einer `ArrayList` enthaltenen Namen (Zeile 1 und 2) möglich. Der Name wird erzeugt durch die Methode `getRandomName()` (Zeile 3 bis 5).

```
01: private ArrayList<String> _names = new ArrayList<String>(8000);
02: private ArrayList<String> _surnames =
                                new ArrayList<String>(3500);
03: public String getRandomName() {
04:     return getRandomEntry(_names) + " " + getRandomEntry(_surnames);
05: }

06: private String getRandomEntry(ArrayList<String> container) {
07:     int size = container.size();
08:     int index = (int) (Math.random()* size);
09:     return container.get(index);
10: }
```

Quellcode 19: Generierung von Namen in TenantNames.java

e) Grenzen und Erweiterungsmöglichkeiten

Die beschriebene Lösung bezieht sich nur auf die Erzeugung von Parametern und deren Werte. Es ist aber denkbar, dass eine individualisierte Erzeugung in die Struktur eines Simulationsmodells eingreift, d.h. Simulationselemente hinzufügt oder weglässt.

Bezüglich der Vererbungshierarchie des `IInitialValueProvider` ist vorstellbar, dass es Parameter gibt, deren Werte numerisch, aber diskret sind. Die Vererbungshierarchie kann entsprechend erweitert werden.

Die Erzeugung der Namens kann geschlechtsspezifisch durchgeführt werden. Die Grundlagen sind bereits gelegt, da es eine Datei mit weiblichen und eine mit männlichen Vornamen gibt. Im Moment werden diese allerdings in eine Liste eingelesen. Ebenfalls ist es möglich, auch für die Verteilung der Namen eine diskrete Verteilungsfunktion zu benutzen, die der Realität näher kommt. Im Moment sind alle Namen gleichverteilt. Diese Lösung scheint dem derzeitigen Simulationsziel angemessen zu sein.

2.4.2 Konfiguration im Szenariomodul

a) Generelles Problem

Die Plattform nutzt in hohem Maße die Grundsätze der Erweiterbarkeit und Modularität und ist sehr flexibel, wenn es darum geht, neue Module und Simulationselemente einzuführen. An einer Stelle jedoch kann die Flexibilität nicht aufrecht erhalten werden: Wenn es darum geht, ein konkretes Szenario aufzubauen, müssen die Module festgelegt werden, die notwendig sind, um das Szenario zu betreiben. Dies zeigt sich u.a. an Berechnungsalgorithmen für Parameter. Es gibt zwar Ansätze, die Werte von Parametern unabhängig von einem bestimmten Modul festzulegen (s. Kap. 2.3.2 Parameter Contribution), aber in einigen Fällen kann nicht derart offen programmiert werden. Ein Beispiel ist die Klasse `LueftungsverhaltenInitialValueProvider`: Diese erzeugt initiale Werte für den Parameter `Lueftungsverhalten` in Abhängigkeit von einer demographischen Klassifizierung des Mieters (Parameter `Milieu`)²⁶. Diese beiden Parameter gehören fachlich nicht direkt zusammen und wären daher auch in zwei verschiedenen Modulen unterzubringen²⁷. Durch diese Verbindung würden die beiden Module zwingend in einem Szenario zusammengehören. Sie könnte aufgehoben werden, wenn die initiale Belegung des Parameters über eine andere Klasse erfolgen würde, die nicht den Wert des Parameters `Milieu` benutzt.

Für einige Module gibt es ebenfalls eine Liste von erforderlichen Modulen. Gehört ein solches Modul zu einem Szenario, dann müssen auch dessen erforderliche Module in die Liste der vorausgesetzten Module für das Szenario-Modul mit aufgenommen werden. Abbildung 10 zeigt einen solchen Abhängigkeitsbaum.

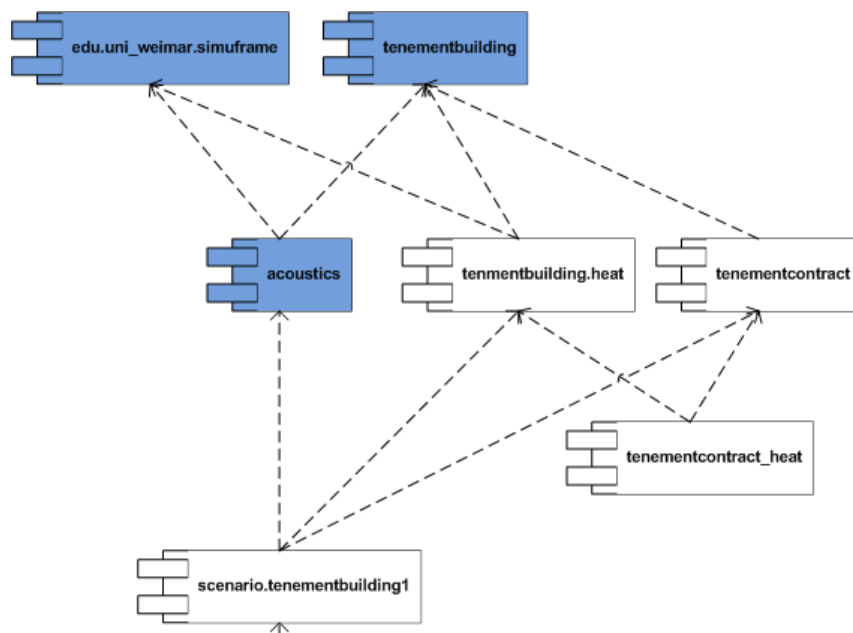


Abbildung 10: Modulabhängigkeiten für ein Szenario-Modul

²⁶ Dies geschieht nach einem ähnlichen Muster wie für die Klasse `EinkommenInitialValueProvider` in Quellcode 18 beschrieben wurde.

²⁷ Im derzeitigen Prototyp sind sie zwar vorerst aus pragmatischen Gründen zusammen in einem Plug-in untergebracht, die Erstellung eines weiteren Plug-ins zur Thematik der Lüftung ist aber noch ein offener Punkt.

Das Modul `acoustics`²⁸ setzt die Module `tenementbuilding` und `edu.uni_weimar.simuframe` voraus. Das Modul `scenario.tenementbuilding1` setzt direkt das Modul `acoustics` voraus, indirekt aber auch diejenigen Module, von denen dieses Modul abhängt, also auch `tenementbuilding` und `edu.uni_weimar.simuframe`. Bemerkenswert ist, dass auch das Modul `tenementcontract_heat` zu den Modulen des Szenarios gehört, aber nicht direkt vom Szenario-Modul vorausgesetzt wird. An dieser Stelle erweisen sich die Maßnahmen zur Modularisierung und Reduzierung von Abhängigkeiten als wirksam.

Es lassen sich verschiedene Arten von Modulabhängigkeiten unterscheiden:

- **Direkte Abhängigkeit**
Diese wird in der Modulbeschreibung (`MANIFEST.MF`) vermerkt.
- **Indirekte Abhängigkeit**
Diese ergibt sich aus dem Baum der Modulabhängigkeiten. Eine indirekte (oder transitive) Abhängigkeit ist gegeben, wenn Modul C Modul B voraussetzt und Modul B seinerseits Modul A benötigt. Hier ist Modul C indirekt von Modul A abhängig.
- **Semantische Abhängigkeit**
Eine solche Abhängigkeit liegt im Falle des Moduls `tenementcontract_heat` vor: Weder der Compiler noch die Eclipse IDE²⁹ erkennen die Abhängigkeit, für die Funktionalität des Szenarios ist sie jedoch von Bedeutung.

b) Lösungsansatz

Es werden spezielle Szenario-Module erstellt. Zu den Aufgaben der Szenario-Module gehört zum ersten die Festlegung der beteiligten Module und zum zweiten die Bereitstellung von einem oder mehreren Szenarien. Für die Generierung von Szenarien ist es von Bedeutung, welche Module eingeschlossen sind und genutzt werden können, da hier die konkreten Simulationselemente instanziiert werden. Auch muss bei der Erstellung der Ersatzbauteile feststehen, welche Parameter mit Werten zu belegen sind. Die Szenario-Module sind der Ort, an dem die Paradigmen Erweiterbarkeit und Modularität in den Hintergrund treten: Die Module, die für das Szenario-Modul zur Verfügung stehen, bilden einen Baukasten, aus dem das Szenario ausschließlich zusammengesetzt werden muß.

c) Implementierung

Zunächst werden die Module festgelegt, die zur Realisierung eines Szenarios benötigt werden. Im Prototyp geschieht dies mit Hilfe der Eclipse IDE (s. Abbildung 11). Darauf aufbauend werden die Simulationselemente und sonstigen Bestandteile der Module zusammengebaut und konfiguriert.

²⁸ Die Modulnamen werden hier kleingeschrieben. Dies entspricht den tatsächlichen, technischen Namen der Module. An anderen Stellen der Arbeit werden sie aus ästhetischen Gründen großgeschrieben. Es sind in beiden Fällen dieselben Module gemeint.

²⁹ Die Eclipse IDE bietet eine Werkzeugunterstützung für das Management der Abhängigkeiten.

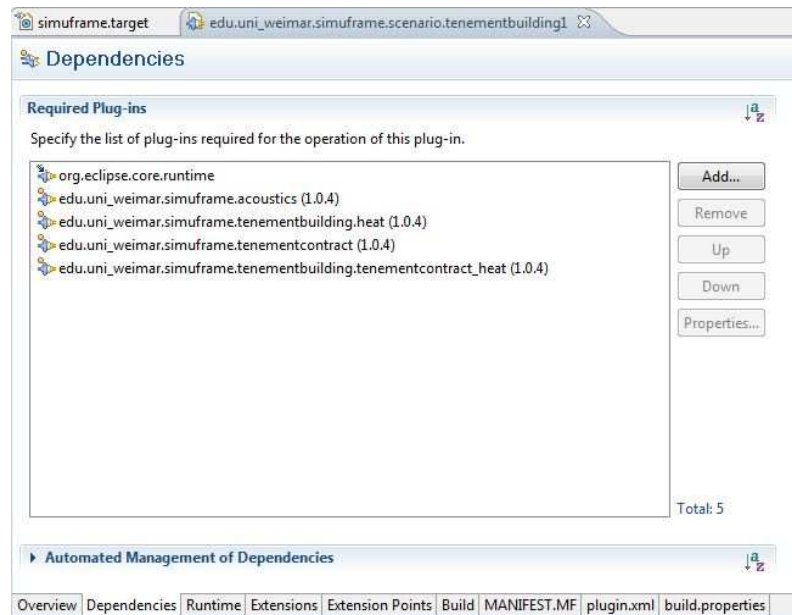


Abbildung 11: Abhängigkeiten des Moduls tenementbuilding1

d) Grenzen und Erweiterungsmöglichkeiten

In der realisierten Version lassen sich die Module noch nicht pro Szenario-Modul ein- oder ausschalten, es sind immer alle Module geladen, die über die Laufzeit-Konfiguration (engl. „Runtime Configuration“) der Eclipse IDE eingebunden sind. Dies bedeutet, dass z.B. immer alle Parameter aktiviert sind, auch wenn diese für ein bestimmtes Szenario nicht benötigt werden oder nicht vorgesehen sind. Durch die OSGi-Konsole ist es zwar möglich, einzelne Module ein- oder auszuschalten, dies gilt dann aber für die komplette OSGi-Instanz. Die Anforderung ist jedoch, dass auf dem Server unterschiedliche Szenarien mit jeweils einer eigenen Modulkonfigurationen parallel laufen können. Eine Möglichkeit, das zu realisieren ist eine eigene Implementierung: die Manager-Instanzen, also diejenigen Klassen, in denen die Simulationselemente, Parameter und sonstigen Erweiterungen verwaltet werden, können um deren Herkunft, also den Modulen, ergänzt werden. Beim Aufbau des Szenarios werden dann gezielt nur die Simulationselemente gefiltert, die aus vorausgesetzten Modulen für das Szenario-Modul stammen.

2.5 Numerische Aspekte

2.5.1 Problemstellung

Im Rahmen der Verifikation eines Simulationsmodells ist zu prüfen, ob die errechneten Werte eines Parameters dem zu simulierenden Funktionsverlauf entsprechen. Da bei der vorgeschlagenen Plattform gewöhnlich jeder Parameter in einem vereinfachten Verfahren berechnet wird und die jeweilige Berechnungsvorschrift durch den Szenariodesigner vorgegeben wird, ist die Verifikation der Berechnungsvorschrift auch Aufgabe des Szenariodesigners.

Ein Hilfsmittel zur Verifikation sind Diagramme, in denen die Werte der Parameter im Zeitverlauf dargestellt werden (s. Anhang D, Parameter Monitor bzw. Parameter Comparison). Mit Hilfe eines Diagrammes konnte unter anderem die im Folgenden dargestellte, nicht plausible Berechnungsvorschrift entdeckt werden.

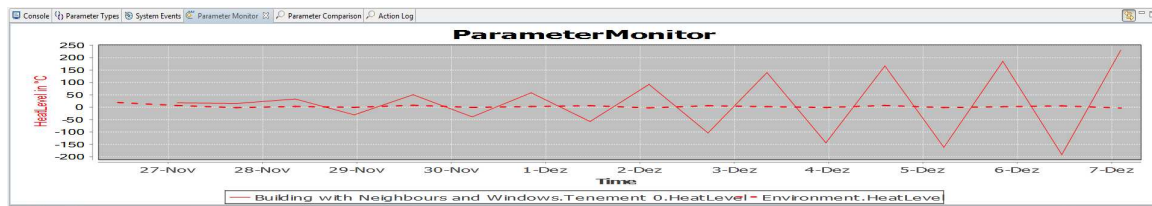


Abbildung 12: Parameter "HeatLevel": Verstärkende Schwingung

In Abbildung 12 sind die Werteverläufe von zwei Parametern eines Simulationsszenarios dargestellt. Die gestrichelte Linie entspricht der Außentemperatur, die durchgezogene Linie gibt die Werte für die Innentemperatur in einer Wohnung wieder. Der Graph für die Innentemperatur zeigt zwei Merkmale, die nicht plausibel sind: Zum einen schwingt die Innentemperatur um die Außentemperatur. Die Schwingungen haben dabei eine gleichmäßige Dauer. Zum anderen verstärken sich die Schwingungen auf sehr große Werte, auf der Skala für die Temperatur finden sich Werte von -200 °C bis 250 °C. Die Schrittweite zwischen zwei Simulationsschritten liegt bei 15 Stunden simulierter Zeit.

Die Schlussfolgerung ist, dass die Schrittweite der Simulationsschritte zu groß ist: Um eine vereinfachte Berechnungsvorschrift nutzen zu können, wird die Innentemperatur in Form einer Differenz ermittelt: Es wird die Wärmebilanz der Wohnung betrachtet. Zu einem bestimmten Zeitpunkt enthält die Wohnung bzw. genauer die darin enthaltene Luft eine definierte Menge an thermischer Energie. Diese Menge wird vermindert um die Transmissionswärmeverluste und vergrößert um die Menge der zugeführten Heizungswärmeenergie. Schließlich wird die nun enthaltene Wärmeenergie wieder in eine Temperatur umgerechnet. Die Wärmeverluste und Wärmegewinne werden über den Zeitraum der Berechnungsschrittweite als konstant betrachtet. Dieses entspricht aber nicht der Realität: Die Regelung der Heizung würde die Zufuhr von Wärmeenergie verändern, wenn eine bestimmte Temperaturschwelle überschritten bzw. unterschritten werden würde. Der Graph aus Abbildung 12 lässt sich dadurch erklären: Durch die große Schrittweite wird simuliert zu viel Wärmeenergie zugeführt, die zu einer ansteigenden Temperatur führt. Im nächsten Schritt wird die Heizung herunter geregelt, gleichzeitig hat sich der Wärmeverlust aber erhöht. Die Folge ist, dass die rechnerische Temperatur der Wohnung stark fällt. Eine tiefe Temperatur hat aber wieder ein Ansteigen der zugeführten Wärmeenergie zur Folge mit einer noch stärkeren Anhebung der Temperatur, der Zyklus beginnt wieder von vorne.

Eine auf 2/3 verringerte Schrittweite führt zu dem Verlauf der Kurve in Abbildung 13. Es findet kein Aufschaukeln der Werte der Raumtemperatur mehr statt, der resultierende Kurvenverlauf ist auch über einen längeren Zeitraum sehr gleichmäßig, wie im unteren Teil der Abbildung ersichtlich ist³⁰. Dennoch ist der Temperaturverlauf nicht plausibel, denn die Innenraumtemperatur fällt an einigen Stellen unter die Außentemperatur – das sollte dem zugrundegelegten Simulationsmodell ohne Kühlung theoretisch nicht möglich sein.

³⁰ Unterer und oberer Teil der Abbildung unterscheiden sich durch die aufgezeichnete Zeitdauer: Der untere Teil der Abbildung zeigt die Kurve über einen bedeutend längeren Zeitraum.

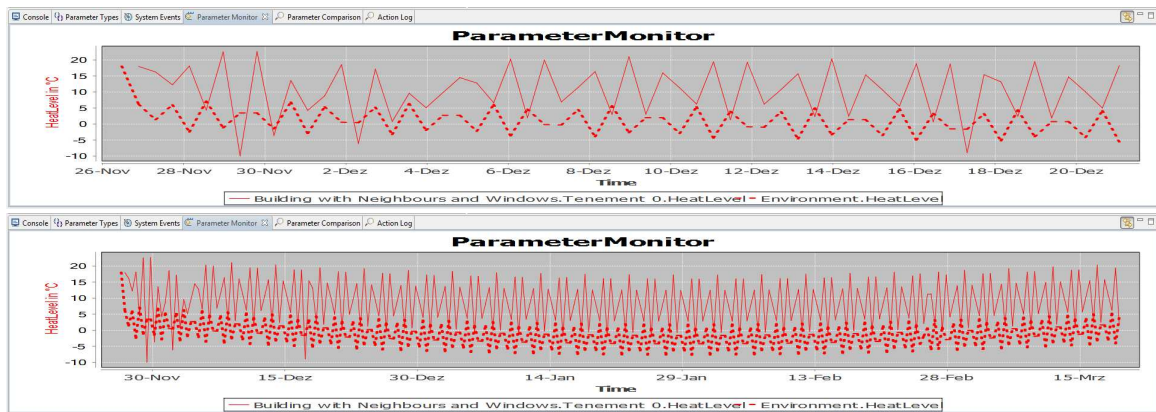


Abbildung 13: Parameter „HeatLevel“: Schrittweite um 1/3 verringert

Die weitere Senkung der Schrittweite auf 1/3 ergibt den Kurvenverlauf in Abbildung 14: Dieser ist wesentlich gleichmäßiger und oberflächlich plausibel – die Innentemperatur fällt nie unter die Außentemperatur – jedoch gibt es immer noch einige Ausreißer, die das Ergebnis als nicht zufriedenstellend erscheinen lassen.

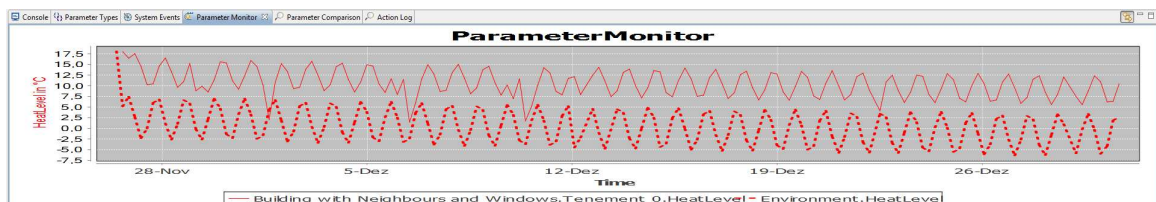


Abbildung 14: Parameter "HeatLevel": Schrittweite auf 1/3 verringert

Die Kurve der Innentemperatur in Abbildung 15 ist das Ergebnis einer Verminderung der Schrittweite auf 1/8 der ursprünglichen Schrittweite. Hier zeigt sich nach einer kurzen Einschwingphase ein relativ gleichmäßiger Verlauf während des Tages – so wie es die aus der Überlagerung zweier Sinusfunktionen erzeugten Temperaturen es vermuten lassen. Die zweite Schwingung (über das Jahr gestreckt) lässt sich angedeutet aus der unteren Kurve in Abbildung 15 erkennen. Die Raumtemperatur folgt der Außentemperatur zeitlich versetzt nach mit kleineren Amplituden. Hier zeigt sich der Effekt der Heizung, deren Kapazität von 500 W aber nicht ausreicht, um den Raum komplett zu heizen.

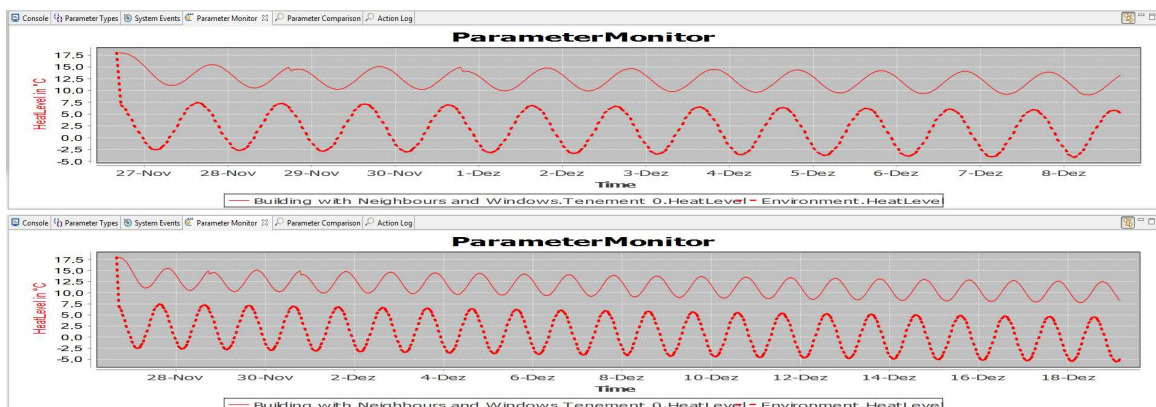


Abbildung 15: Parameter "HeatLevel": Schrittweite auf 1/8 verringert

Abbildung 16 zeigt den Verlauf der Innenraumtemperatur, wenn der Heizkörper mit einer maximalen Heizleistung ausreichend dimensioniert ist: Die Innenraumtemperatur wird abgesehen

von kleinen Schwankungen, die auf die Regulierung der Heizleistung in 500 W-Stufen zurückzuführen sind, konstant auf dem Niveau der Wunschtemperatur des Bewohners gehalten.

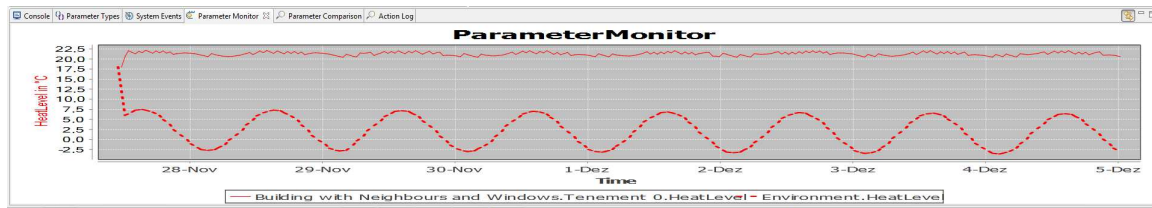


Abbildung 16: Parameter "HeatLevel": Schrittweite auf 1/8 verringert, max. Heizleistung 2000 W

2.5.2 Qualitätseigenschaften von numerischen Algorithmen

Das obige Beispiel zeigt eine generelles Problemquelle der Spielplattform: die Berechnungen unterliegen den Gesetzmäßigkeiten numerischer Lösungsverfahren. Zu diesen zählen Stabilität und Kondition: Mit der Kondition eines Problems wird der Grad der Auswirkung von Änderungen von Eingabedaten auf die Ausgabedaten bezeichnet. Sie wird gemessen mit der sogenannten Konditionszahl und ist eine Eigenschaft der Problemstellung (Martin Hermann, 2006, S. 7). Die Kondition wird auch als natürliche Stabilität bezeichnet. Daneben gibt es auch die numerische Stabilität. Sie ist eine Eigenschaft des Verfahrens: Hermann (2006, S. 14) nennt einen Algorithmus instabil, „wenn es Eingabedaten gibt, bei denen sich die Rundungsfehler während der Rechnung so akkumulieren, dass ein völlig falsches Ergebnis entsteht“.

2.5.3 Lösungsmöglichkeiten

Es gibt verschiedene Wege, den numerischen Problemstellungen zu begegnen. Sie werden im Folgenden diskutiert:

a) Fehlererkennung

Visuelle Verifikation

Mit Hilfe der in die Plattform integrierten Graphik-Werkzeuge (s. Anhang D) ist es möglich, den zeitlichen Verlauf von Parameterwerten darzustellen und eine graphische Verifikation durchzuführen. Dieses wurde auch im Eingangsbeispiel demonstriert.

Fehlerabschätzung

Methoden der Fehlerschätzung und Fehlerabschätzung gehören zur Entwicklung von numerischen Verfahren – ohne eine Angabe der Genauigkeit und damit der Güte eines Ergebnisses ist dieses quasi wertlos. Ziel dieser Plattform ist keine quantitative Simulation, sondern eine qualitative. Dennoch kann mit Fehlerabschätzungen nachvollzogen werden, ob die erhaltenen Werte einer qualitativen Simulation Genüge leisten.

Durch manuelle Berechnungen bzw. Analysen lassen sich ebenfalls Aussagen zu möglichen Fehlern tätigen. Generell ist eine Aussage zu treffen, welcher Zusammenhang größenmäßig zwischen zwei aufeinanderfolgenden Werten der zu simulierenden Größe besteht. Das obige Beispiel wird im Folgenden betrachtet: Wie wird die Innentemperatur zu einem bestimmten Zeitpunkt aus der Innentemperatur des davorliegenden Zeitpunktes berechnet?

Gemäß der im Prototyp verwendeten Formel ergibt sich die Temperatur T zum Zeitpunkt t_{n+1} aus der Temperatur zum Zeitpunkt t_n und einem Differenzwert ΔT (Formel 2: Temperatur T).

$$T_{n+1} = T_n + \Delta T$$

Formel 2: Temperatur T

$$\Delta T = \frac{(\Phi_{H,n} - \Phi_{V,n}) * (t_{n+1} - t_n)}{V * S * 3600}$$

Formel 3: Temperaturdifferenz ΔT

Der Nenner des Bruches wird als konstant angenommen. Der Heizwärmestrom $\Phi_{H,n}$ ist nach oben begrenzt durch die maximale Heizleistung $\Phi_{H,max}$. Der Verlustwärmestrom, $\Phi_{V,n}$ (Formel 5), kann auf lange Sicht nicht größer werden als der Heizwärmestrom. Somit ist in Formel 3 die Zeitdifferenz $t_{n+1} - t_n$ der Faktor, von der die Temperaturdifferenz linear abhängig ist. Mit sich vergrößernder Zeitdifferenz wird die Temperaturdifferenz ebenfalls größer.

$$\Phi_{H,n} = \text{Min}(\text{Max}([T_{opt} - T_n] * \Phi_{H,R}, 0), \Phi_{H,max})$$

Formel 4: Heizwärmestrom $\Phi_{H,n}$

$$\Phi_{V,n} = (T_n - T_{E,n}) * \sum_{i=1}^n (A_i * U_i)$$

Formel 5: Verlustwärmestrom $\Phi_{V,n}$

Symbol	Einheit	Erläuterung
T_n	K	Temperatur zum Zeitpunkt t_n
ΔT	K	Temperaturdifferenz
$\Phi_{H,n}$	W	Heizwärmestrom zum Zeitpunkt t_n
$\Phi_{V,n}$	W	Verlustwärmestrom zum Zeitpunkt t_n
$\Phi_{H,max}$	W	Maximaler Heizwärmestrom
$\Phi_{H,R}$	W	Minimale Einheit, in der der Heizwärmestrom geregelt werden kann.
T_{opt}	K	Gewünschte Innenraumtemperatur
$T_{E,n}$	K	Außentemperatur zum Zeitpunkt t_n
A_i	m ²	Oberfläche des i-ten Wandbauelements
U_i	W/(m ² *K)	U-Wert des i-ten Wandbauelements
V	m ³	Volumen des beheizten Raumes.
S	Wh/(m ³ *K)	Wärmespeicherzahl

Tabelle 6: Symbolverzeichnis: Beispiel zur Fehlerabschätzung

In Abbildung 17 wird der Einfluss der Schrittweite auf die Genauigkeit des Ergebnisses graphisch verdeutlicht: Gegeben ist die tatsächliche Kurve des Wärmestroms. Es gibt einen Zeitpunkt t_n , zu dem diese Kurve für den Wärmestrom einen bestimmten Wert definiert. Die zu vergleichenden Schrittweiten werden durch den zeitlichen Abstand von t_n zu den Zeitpunkten t_{n+1} bzw. t'_{n+1} definiert. Eine Fläche unter der Kurve bezeichnet eine Wärmemenge.

Die Grafik verdeutlicht den Unterschied zwischen berechneten und tatsächlichen Wärmemengen sowie die wachsende Ungenauigkeit bei größer werdender Schrittweite. Für die kleinere Schrittweite $\Delta t = t_{n+1} - t_n$ ist die berechnete Wärmemenge durch Q_1 sowie die tatsächliche Wärmemenge durch Q_1+Q_2 gegeben. ΔQ – die Abweichung des berechneten Wertes zum tatsächlichen Wert – wird somit durch Q_2 ausgedrückt. Für die doppelt so große Abweichung $\Delta t' = t'_{n+1} - t_n$ ist die berechnete Wärmemenge durch Q_1+Q_3 gegeben, die tatsächliche Wärmemenge durch $Q_1+Q_2+Q_3+Q_4$. Damit ist $\Delta Q' = Q_2+Q_4$. Durch die Größen der Flächen, die jeweils äquivalent zu den Wärmemengen sind, wird deutlich gemacht, dass bei wachsender Schrittweite auch die Abweichungen zum tatsächlichen Wert im schlechtesten Fall überproportional wachsen.

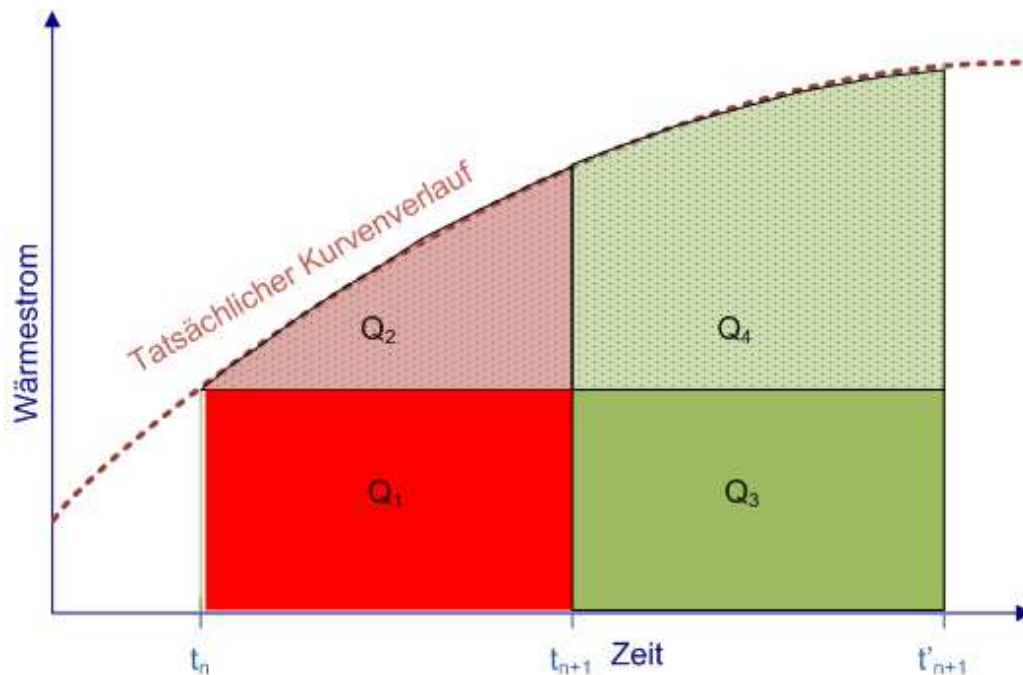


Abbildung 17: Berechnungsfehler bei unterschiedlicher Schrittweite

Überwachung durch die Plattform

Schaut man sich den Graphen der Abbildung 12 an, so ist der Verlauf der Innentemperatur untypisch für eine physikalische Größe und insbesondere für einen Temperaturverlauf. Ein solcher stark schwingender Werteverlauf kann auch automatisiert erkannt werden. Ähnlich der Funktionsweise eines verhaltensbasiert arbeitenden Virenschutzprogrammes könnte die Plattform eine Überwachungsfunktion bereitstellen, die regelbasiert Parameterverläufe überwacht und den Szenariodesigner auf solche mit großer Wahrscheinlichkeit fehlerhaften Parameterwertverläufe hinweist. Parameter könnten zur Überwachung kategorisiert werden mit verschiedenen Attributen. Beispielsweise können erwarteter Maximal- und Minimalwert bzw. bestimmte Verlaufskategorien zugeordnet werden. Nachteilig ist, dass eine solche automatisierte Überwachung Rechenzeit kostet.

b) Fehlerbehebung und konstruktive Fehlervermeidung

Anpassung der Schrittweite

Ein Gütekriterium eines numerischen Algorithmus ist der notwendige Aufwand, der zur Erreichung des vorgesehenen Ergebnisses anfällt. Weniger Aufwand ist besser und wird angestrebt. Reicht die Qualität des Ergebnisses nicht – wie dieses im obigen Beispiel der Fall ist – bringt eine Verringerung der Schrittweite in der Regel eine Qualitätsverbesserung auf Kosten des notwendigen Rechenaufwandes. Problem einer möglichen, freien Anpassung der Schrittweite kann sein, dass die Zeitmaßstäbe nicht mehr zueinander passen: Soll mit einem Szenario beispielsweise ein Zusammenhang demonstriert werden, der einen Monat dauert und ist gleichzeitig die Schrittweite so klein, dass das Szenario mehrere Stunden echter Zeit für den Ablauf braucht, so ist es nur noch erschwert spielbar.

Heuristiken und Regeln

Durch den geschickten Entwurf eines Algorithmus ist es möglich, die numerische Stabilität positiv zu beeinflussen. Aus diesem Grund bietet es sich an, in der Dokumentation der Plattform Heuristiken und Regeln darzustellen, die dem nicht notwendigerweise auf dem Gebiet der Nu-

merik erfahrenen Szenariodesigner Hilfestellung geben können. Beispiele für solche Informationen sind folgende Regeln und Hinweise:

- **Auslöschung:** Addition von betragsgleichen Zahlen mit umgekehrten Vorzeichen führt zum Phänomen der Auslöschung und ist zu vermeiden (Martin Hermann, 2006, S. 11).
- **Betragskleine Zahlen:** Die Division durch betragsskleine Zahlen ist ebenfalls zu vermeiden (Martin Hermann, 2006, S. 11).
- **Unterstützung der Programmiersprache:** Programmiersprachen stellen Datentypen bereit, die für numerische Berechnungen besonders geeignet sind. Für die Sprache Java sind das u.a. die Datentypen `BigInteger` und `BigDecimal`.
- **Minimierung der elementaren Rechenoperationen** Stehen mehrere unterschiedliche Algorithmen zur Auswahl, so kann es lohnend sein, die Algorithmen bezüglich Anzahl und Art der notwendigen Rechenoperationen zu vergleichen (Engeln-Müllges, Niederdröck & Wodicka, 2005, S. 26).

2.5.4 Schlussfolgerungen

Es ist mit dem Auftreten von numerischen Fehlern in den Berechnungen der Plattform zu rechnen, insbesondere bei vereinfachten Berechnungsmethoden. Szenarien sollen durch Fachexperten erstellt werden können. Dieses bedeutet, dass schlechtestenfalls wenig Wissen zur numerischen Stabilität von Algorithmen vorhanden ist. Begleitende systematische analytische Betrachtungen zur Stabilität eines Algorithmus bei der Erstellung von Berechnungsvorschriften können von den Fachexperten nicht gefordert werden. Anderenfalls ist mit sinkender Akzeptanz der Plattform zu rechnen. Daher wird ein pragmatisches Vorgehen mit den folgenden Punkten vorgeschlagen:

- **Kurzanleitung zur Vermeidung numerischer Probleme**
Eine kurze Einführung in die Thematik schärft das Bewusstsein. Gleichzeitig werden dem Szenariodesigner die wichtigsten Werkzeuge an die Hand gegeben, mit denen sich numerische Probleme schon beim Entwurf von Algorithmen vermeiden lassen.
- **Visuelle Verifikation mit Hilfe von Graphen**
Die Einbeziehung einer menschlichen Kontrollinstanz ist durch die graphische Darstellung von Parameterwertefolgen schnell und mit wenig Aufwand möglich.
- **Konfigurierbare automatisierte Überwachung**
Eine automatisierte Überwachung der Parameterwerte hilft, größere Abweichungen vom erwarteten Verlauf zu finden. Um den Laufzeitaufwand niedrig zu halten, kann sich die Überwachung auf eine Testphase beschränken. Mit der Konfiguration eines erwarteten Verlaufs der Parameterwerte wird der Szenariodesigner zur Reflektion angeregt. Die dabei gewonnenen Erkenntnisse können auch in der visuellen Verifikation angewendet werden.

3 Oberfläche des Szenario-Editors

In diesem Kapitel werden die Bestandteile der Oberfläche des Szenario-Editors beschrieben³¹. Zum Verständnis der Benutzeroberfläche unter Eclipse RCP sind verschiedene Begrifflichkeiten zu klären:

- **View** (dt. Sicht): Eine View ist eine Fenster-Komponente der Oberfläche in Eclipse RCP: Sie beinhaltet gewöhnlich mehrere Steuerelemente und kann eine Werkzeugleiste sowie ein Menü aufnehmen.
- **Editor**: Ein Editor³² ist – genauso wie eine View - eine Fenster-Komponente der Oberfläche in Eclipse RCP. Im Gegensatz zur View kann ein Editor nur in einer Instanz in der gesamten Oberfläche einer Anwendung vorhanden sein und hat einen festgelegten Platz (gewöhnlich in der Mitte) in der Oberfläche. Auch ist ein Editor mit einer Datenquelle (in der Regel eine Datei) verbunden und kann durch die Editieroperationen mit der Datenquelle nicht synchronisierte Daten enthalten³³. Hingegen können die in einer View angezeigten Daten nicht editiert werden (Arthorne & Laffra, 2004).
- **Perspective** (dt. Perspektive): In einer Perspective werden in der Eclipse RCP-Oberfläche verschiedene Fenster-Komponenten zu einer geeigneten Arbeitsumgebung zusammengefasst. Diese Anordnung von Fenster-Komponenten kann jederzeit durch Größenanpassungen, Ausblenden von Komponenten sowie Einblenden von zusätzlichen Komponenten angepasst bzw. auf die Ursprungsconfiguration zurückgesetzt werden.

3.1 Beschreibungsinhalte

Die Views werden schematisch beschrieben, die Beschreibungselemente sind weitgehend selbsterklärend. In diesem Kapitel werden einige Bestandteile des Beschreibungsschemas zusätzlich näher erläutert.

Attribute

Zu Beginn einer View-Beschreibung werden in einer Tabelle einige Attribute zur Beschreibung des Views dargestellt.

- **Kategorie**: Der Test-Client hat mehrere Funktionalitäten. Die Kategorie der View gibt an, zu welcher dieser Funktionalitäten die View beiträgt.
- **Sichtbar**: Der Eintrag zu diesem Attribut gibt an, ob die View in der Standardkonfiguration der SimuFrame-Perspektive enthalten ist, d.h. ob sie sichtbar ist, wenn der Test-Client (zum ersten Mal) gestartet wird.
- **Position**: Mit diesem Attribut wird die Standard-Position in der SimuFrame-Perspektive der View angegeben.

³¹ Anzumerken ist, dass im gegenwärtigen Entwicklungsstand die Test-Funktionalität und Monitoring-Funktionalität des Szenario-Editors die wesentliche Komponente ist. Es können Szenarien zur Ausführung gebracht und währenddessen manipuliert und beobachtet werden, es können aber keine neue Szenarien zusammengestellt und editiert werden.

³² In SimuFrame ist zur Zeit noch kein Editor enthalten, dieser Eintrag soll zur Abgrenzung des Begriffes der „View“ beitragen.

³³ Ein Editor besitzt gewöhnlich zum Anzeigen des Status von veränderten Daten ein Dirty-Flag. Das ist ein Schalter: Ist es gesetzt, so sind die Daten des Editors nicht synchron mit denen des Speichermediums.

Erweiterungsmöglichkeiten

Hier werden zusätzliche Funktionalitäten genannt, die bisher noch nicht realisiert wurden, aber einen großen Beitrag zu Funktionsumfang und Benutzerkomfort der Plattform liefern.

Dieser Beschreibungspunkt wird nur aufgeführt, wenn es solche Abhängigkeiten gibt.

Verknüpfung mit anderen Oberflächenkomponenten

Einige Oberflächenkomponenten sind in sich abgeschlossen, sie benötigen zur Erfüllung der Funktionalität keine weiteren Oberflächenkomponenten. Andere Oberflächenkomponenten jedoch interagieren mit weiteren Oberflächenkomponenten. Unter diesem Punkt werden derartige Interaktionen beschrieben.

Dieser Beschreibungspunkt wird nur aufgeführt, wenn es solche Abhängigkeiten gibt.

3.2 Perspective

Zur Ausführung und zum Monitoring der Szenarien wurde eine Perspektive `simuFrame` definiert (Abbildung 18). Sie enthält in der Grundkonfiguration die folgenden Elemente:

1. **Navigator:** Der Navigator erlaubt die Auswahl des gewünschten Szenarios. Die Szenarien werden zurzeit programmatisch definiert und stehen als Prototyp zur Verfügung.
2. **Scenario-Viewer:** Der Scenario-Viewer erlaubt es dem Spieler, ein komplettes Szenario mit allen Simulationselementen und Parametern³⁴ zu untersuchen.
3. **Parameter-Type View:** Dies ist eine View, in der alle in den Szenarien verwendeten Parameter-Typen dargestellt werden. Sie wird im für den View-Stack reservierten Bereich angezeigt.
4. **Anwendungs-Menü:** Im Menü der Anwendung sind alle globalen Operationen untergebracht, wie z.B. das Öffnen einer neuen View, der Verweis auf das Hilfesystem und das Einstellen von Systemoptionen. Unter dem Anwendungs-Menü ist die Werkzeugleiste zur Auswahl der Perspektive dargestellt. Es gibt im Moment nur eine Perspektive.
5. **View-Werkzeugleiste:** Jede View kann ihre eigene Werkzeugleiste besitzen. Hinter dem nach unten gerichteten Dreieck verbirgt sich das Menü der View.
6. **View-Stack:** In diesem Bereich der Benutzeroberfläche werden alle sonstigen Views eingeblendet. Der Benutzer kann eine View auswählen, die für ihn sichtbar ist. In Abbildung 18 ist das die *Parameter Types-View*.

³⁴ Genutzt wird hier ein sogenannter *TreeView*, der ein Szenario hierarchisch darstellt und dem Benutzer erlaubt, alle Verzweigungen und Blätter anzuschauen.

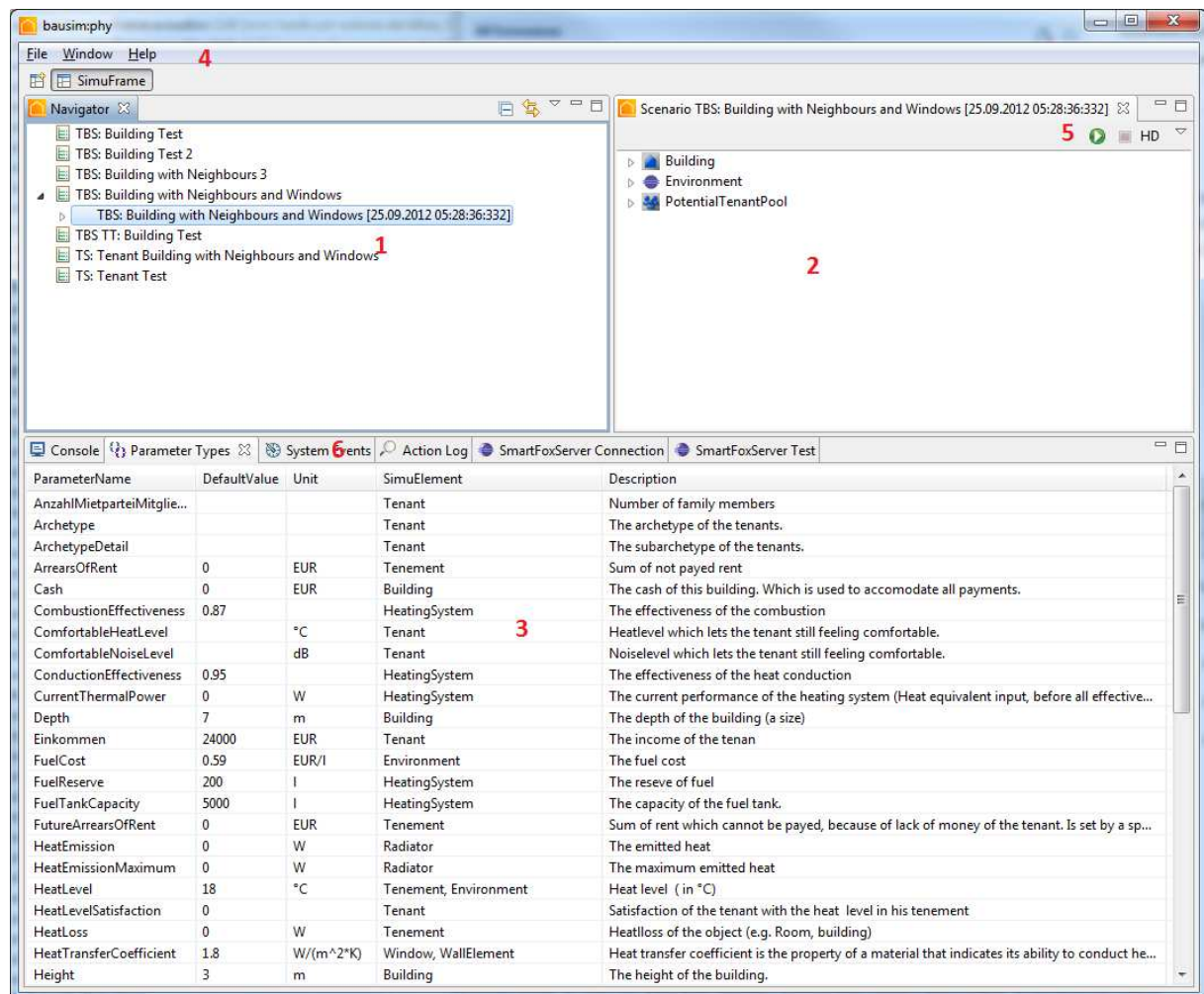


Abbildung 18: Default-Perspektive von SimuFrame

3.3 Views der Kern-Module

Die folgenden Views sind im Modul `edu.uni_weimar.simuframe.ui` definiert.

3.3.1 Scenario Outline

Kategorie	Test-Client
Sichtbar	Ja
Position	Rechts Oben

Tabelle 7: Attribute der View Scenario Outline

Die in Abbildung 19 dargestellte View *Scenario Outline* dient der Darstellung eines vollständigen Szenarios. Das Szenario wird in Form einer Hierarchie dargestellt, da die Struktur des Simulationsmodells diese Darstellung unterstützt. Jeder Knoten lässt sich öffnen, danach werden die untergeordneten Knoten dargestellt. Die Parameter eines Simulationselementes werden diesem untergeordnet gezeigt. Für weitere Elemente, wie dem Simulationskontext, wird ein künstliches Wurzelement eingefügt, so dass sie sich auch in einer Hierarchie zeigen lassen.

Zur Darstellung einer Hierarchie stellt Eclipse ein spezifisches Steuerelement bereit: den *TreeView*. Diesem wird ein Wurzelement übergeben. Für das weitere Funktionieren des *TreeViews* sind zwei Bedingungen zu erfüllen: Alle Elemente der Hierarchie implementieren eine Schnittstelle, mit deren Hilfe die untergeordneten Elemente ermittelt werden können. Zu-

sätzlich ist für jede in der Hierarchie vorhandene Klasse bekannt, wie diese dargestellt werden soll.

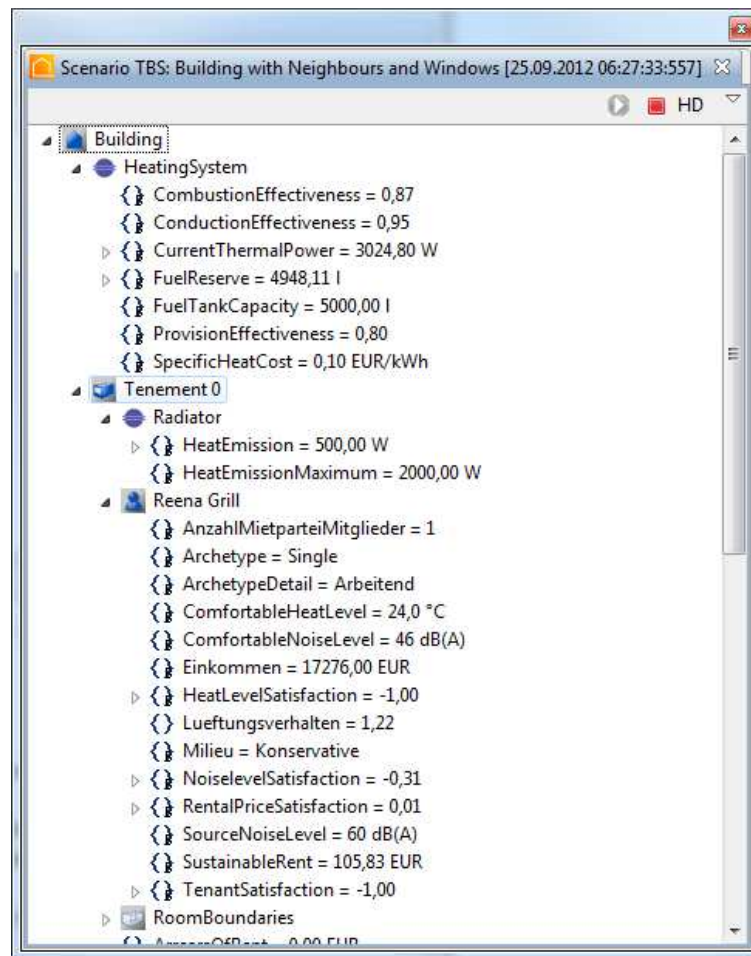


Abbildung 19: View Scenario Outline

a) Verknüpfungen mit anderen Oberflächenkomponenten

Diese View ist mit der View *Navigator* (s. Kap. 3.3.2) synchronisiert: Das in dieser View angezeigte Szenario richtet sich nach dem in der *Navigator*-View angezeigtem Szenario. Die Auswahl eines Szenarios in dieser View sorgt für die Anzeige der Details des ausgewählten Szenarios in der View *Scenario Outline*.

b) Funktionalitäten

Start/Stop

Wird in dieser View ein Szenario angezeigt, so kann es nur ausgeführt werden, wenn es sich nicht um ein Template-Szenario handelt. Um es auszuführen, gibt es in der Werkzeugleiste eine *Start*-Schaltfläche (▶). Der einmal gestartete Simulationslauf kann wieder unterbrochen werden mit der rechts daneben positionierten *Stop*-Schaltfläche (■).

Benutzeraktionen

Nach dem Starten des Szenarios werden die Benutzeraktionen aktiviert: Es ist möglich, aus dem Kontextmenü der einzelnen Elemente des TreeViewers die angebotenen Benutzeraktionen anzustoßen. Teilweise werden für das Durchführen der Benutzeraktionen Parameter benötigt. Sofern diese sich nicht automatisiert ableiten lassen, werden sie über einen zusätzlichen Dialog

abgefragt. Abbildung 20 zeigt den Dialog, der über die Aktion *Install New Windows* (im Kontextmenü der Wohnung oder des Gebäudes) aufgerufen wird: Es kann die Fensterart gewählt werden über das Kombinationsfeld *Type*. Im unteren Teil des Dialogs werden die Werte der Parameter des gewählten Fenstertyps angezeigt. Abgeschlossen werden kann die Auswahl mit der Schaltfläche *Finish*.

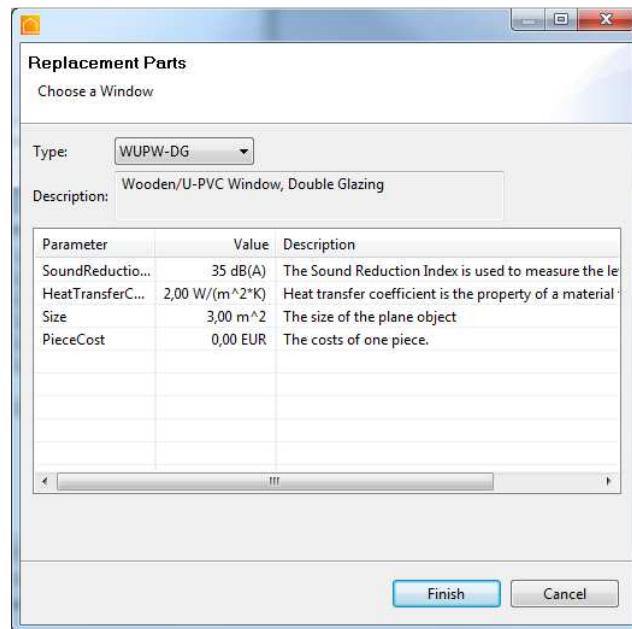


Abbildung 20: Dialog Fensteraustausch

In Abbildung 21 wird der Dialog gezeigt, mit dem der Spieler einem Mietinteressenten eine neue Wohnung anbieten kann. Dieser Dialog erscheint, wenn der Spieler einen Mieter aus der Menge der Mietinteressenten (Simulationselement `PotentialTenantPool`) auswählt und die Aktion *Offer a Tenement* des Kontextmenüs selektiert (Abbildung 22).

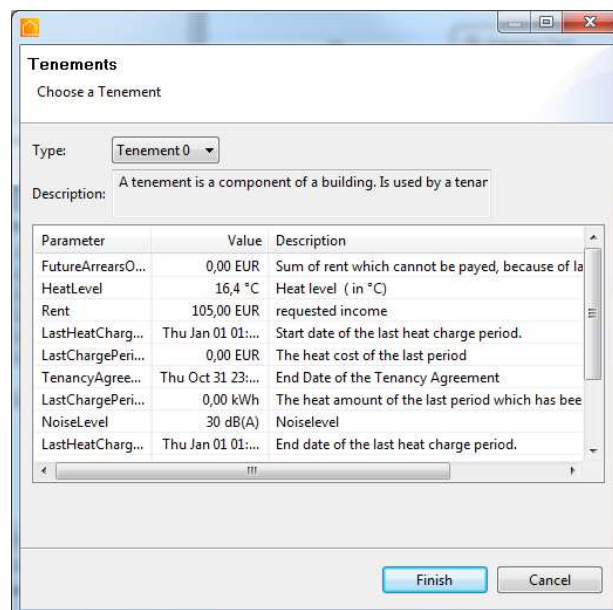


Abbildung 21: Dialog: Auswahl einer Wohnung für einen neuen Mieter

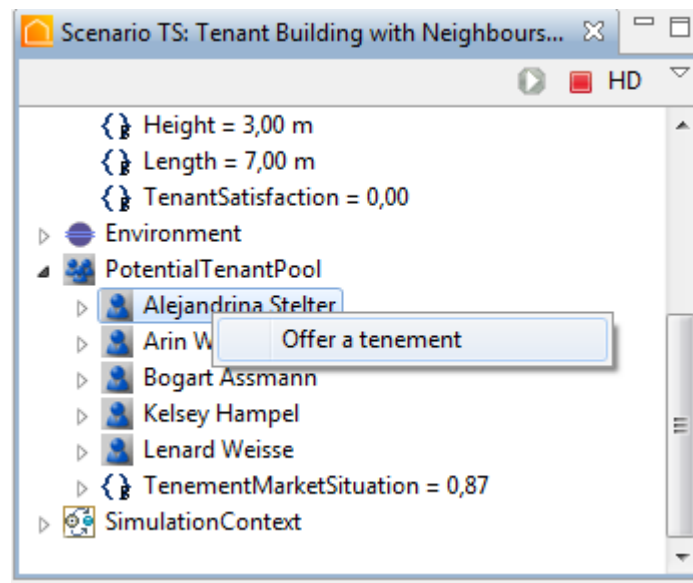


Abbildung 22: Benutzeraktion: Anbieten einer Wohnung

In Abbildung 23 wird der generische Änderungsdialog für einen Parameter gezeigt. Jeder Parameter, der nicht als schreibgeschützt gekennzeichnet ist, lässt sich ändern: ein Eintrag im Kontextmenü führt zu dem gezeigten Dialog.

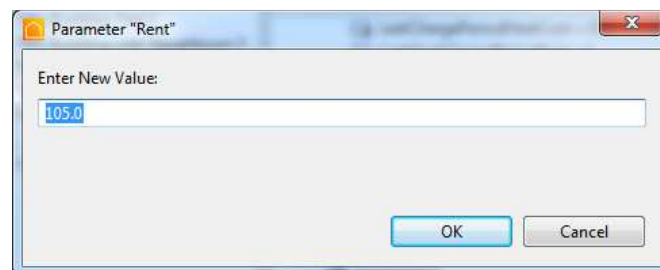


Abbildung 23: Änderung des Wertes eines Parameters

3.3.2 Navigator

Kategorie	Test-Client
Sichtbar	Ja
Position	Links Oben

Tabelle 8: Attribute der Navigator-View

Die *Navigator*-View (Abbildung 19) dient der Darstellung der im System vorhandenen Szenarios. Sie wird gefüllt durch die Inhalte, die die per Extension Point `scenarioFactory` deklarierten Klassen liefern.

Die aufgeführten Szenarios sind zunächst einmal prototypische Objektgeflechte, sogenannte Template-Szenarios. Sie müssen vor dem Simulationslauf kopiert werden, damit parallele Simulationsläufe nicht dieselben Objekte des Simulationsmodells verändern. Das Kopieren wird über das Kontextmenü angestoßen. Der entsprechende Menüeintrag heißt *Generate Scenario*.

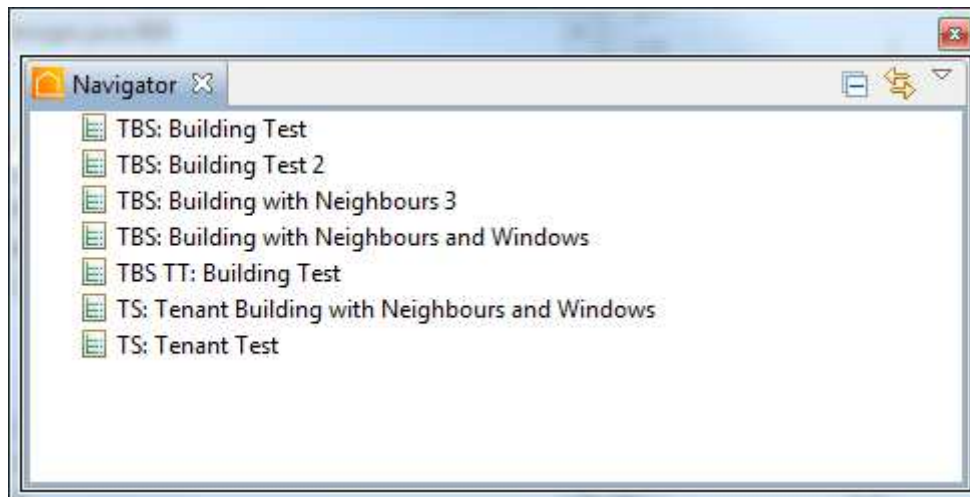


Abbildung 24: View Navigator

a) Verknüpfungen mit anderen Oberflächenkomponenten

Diese View ist mit der View *Scenario Outline* (siehe Kap. 3.3.1) synchronisiert: Die Auswahl eines Szenarios in dieser View sorgt für die Anzeige der Details des ausgewählten Szenarios in der View *Scenario Outline*.

b) Erweiterungsmöglichkeiten

Filter- und Suchfunktion

Zurzeit ist die Anzahl der angezeigten Szenarios noch übersichtlich. Mit zunehmender Anzahl an Szenarios ist es vorteilhaft, die Szenarios durchsuchen und filtern zu können. Zu den möglichen Kriterien für Suche und Filter gehören:

- Ersteller
- Simulationselemente
- Parameter

Gesucht werden könnte eine einzugebende Zeichenkette mit Platzhaltern in den Namen der oben genannten Objekte.

3.3.3 Parameter Types

Kategorie	Test-Client
Sichtbar	Nein
Position	View-Stack unten

Tabelle 9: Attribute der View Parameter Types

Die View *Parameter Types* gibt einen Überblick über die auf der Plattform bekannten Parameter Typen, unabhängig von einem konkreten Szenario. Für jeden Parameter werden die folgenden Informationen angezeigt:

- Name
- Initialwert
- Einheit
- Assoziierte Simulationselemente³⁵
- Beschreibung

³⁵ Ein Parameter kann mehreren Simulationselementen zugeordnet sein. Ein Beispiel ist die Temperatur: Sie ist Parameter sowohl der Umgebung (Environment) als auch der Wohnung (Tenement).

Ein Klick auf die Spaltenüberschriften sorgt für eine Sortierung der Tabelle bezüglich der Werte der Spalte. Ein weiterer Klick auf dieselbe Spaltenüberschrift kehrt die Sortierreihenfolge um³⁶.

ParameterName	DefaultValue	Unit	SimuElement	Description
AnzahlMietparteiMitglie...			Tenant	Number of family members
Archetype			Tenant	The archetype of the tenants.
ArchetypeDetail			Tenant	The subarchetype of the tenants.
ArrearsOfRent	0	EUR	Tenement	Sum of not payed rent
Cash	0	EUR	Building	The cash of this building. Which is used to accomodate all payments.
CombustionEffectiveness	0.87		HeatingSystem	The effectiveness of the combustion
ComfortableHeatLevel		°C	Tenant	Heatlevel which lets the tenant still feeling comfortable.
ComfortableNoiseLevel		dB(A)	Tenant	Noiselevel which lets the tenant still feeling comfortable.
ConductionEffectiveness	0.95		HeatingSystem	The effectiveness of the heat conduction
CurrentThermalPower	0	W	HeatingSystem	The current performance of the heating system (Heat equivalent input, before all effective...
Depth	7	m	Building	The depth of the building (a size)
Einkommen	24000	EUR	Tenant	The income of the tenant
FuelCost	0.59	EUR/l	Environment	The fuel cost
FuelReserve	200	l	HeatingSystem	The reseve of fuel
FuelTankCapacity	5000	l	HeatingSystem	The capacity of the fuel tank.
FutureArrearsOfRent	0	EUR	Tenement	Sum of rent which cannot be payed, because of lack of money of the tenant. Is set by a sp...
HeatEmission	0	W	Radiator	The emitted heat
HeatEmissionMaximum	0	W	Radiator	The maximum emitted heat
HeatLevel	18	°C	Tenement, Environment	Heat level (in °C)
HeatLevelSatisfaction	0		Tenant	Satisfaction of the tenant with the heat level in his tenement

Abbildung 25: View: Parameter Types

a) Erweiterungsmöglichkeiten

Filter- und Suchfunktion

Zur komfortablen Suche von Parametern wäre eine Filterfunktion hilfreich: So könnten beispielsweise die angezeigten Parameter ausschließlich auf Parameter mit einer bestimmten Einheit eingeschränkt werden. Auch könnte nach Parametern gesucht werden, deren Name eine vorgegebene Zeichenfolge enthält.

Weitere Felder

Es sind weitere nutzbringende Felder in der Tabelle denkbar:

- Name des beitragenden Moduls
- Szenarien, in denen der Parameter verwendet wird³⁷.

3.3.4 System Events

Kategorie	Monitoring
Sichtbar	Nein
Position	View-Stack unten

Tabelle 10: Attribute der View System Events

Die View System Events (Abbildung 26) stellt alle wartenden Systemereignisse der laufenden Szenarien dar. Sie dient dem Monitoring und Debugging und ist nicht für den Spieler, sondern für den Szenarioentwickler gedacht. Sie beruht auf der Implementationseigenschaft von Events, dass für jedes Event ein Ausführungszeitpunkt gemäß seiner Konfigurationsparameter ermittelt

³⁶ Dies ist eine Funktionalität des in Eclipse RCP enthaltenen JFace-Frameworks.

³⁷ In dieser Spalte würde eine Wiederholgruppe angezeigt werden, d.h. es gäbe mehrere Einträge. Das ist auch bei anderen Spalten der Fall (Beispiel ist die Spalte „Assoziierte Simulationselemente“ (SimuElement)). Die Darstellung mehrerer Werte in einer Zelle ist nicht sehr übersichtlich, daher sollte hier eine andere Darstellungsweise genutzt werden. Möglichkeiten sind ein ausblätterbarer Baum wie er bei Gruppierungen in Microsoft Excel genutzt wird oder ein Tooltip mit der übersichtlichen Darstellung der Elemente.

und das Event dann in eine Warteschlange eingereiht wird, um zum Ausführungszeitpunkt aktiviert zu werden.

Name	Type	Appearance	nextAdvisedCall (Realti...	nextAdvisedCall (Gameti...	Description
TenantIsUnsatisfied	parameterValueSy...	once	27.09.2012 09:44:53:793	29.09.2012 02:44:40:382	The tenant is unsatisfied and so he decides to cancel the tenancy agree...
TenantIsUnsatisfied	parameterValueSy...	once	27.09.2012 09:44:53:793	29.09.2012 02:44:40:382	The tenant is unsatisfied and so he decides to cancel the tenancy agree...
TenantIsUnsatisfied	parameterValueSy...	once	27.09.2012 09:44:53:793	29.09.2012 02:44:40:382	The tenant is unsatisfied and so he decides to cancel the tenancy agree...
TenantIsUnsatisfied	parameterValueSy...	once	27.09.2012 09:44:53:793	29.09.2012 02:44:40:382	The tenant is unsatisfied and so he decides to cancel the tenancy agree...
TenantIsUnsatisfied	parameterValueSy...	once	27.09.2012 09:44:53:793	29.09.2012 02:44:40:382	The tenant is unsatisfied and so he decides to cancel the tenancy agree...
TenantIsUnsatisfied	parameterValueSy...	once	27.09.2012 09:44:53:793	29.09.2012 02:44:40:382	The tenant is unsatisfied and so he decides to cancel the tenancy agree...
TenantIsUnsatisfied	parameterValueSy...	once	27.09.2012 09:44:53:793	29.09.2012 02:44:40:382	The tenant is unsatisfied and so he decides to cancel the tenancy agree...
RentPayment	timedSystemEvent	repeatedly	27.09.2012 09:45:27:048	30.09.2012 11:59:58:382	Regular payment of the rent.
TenantDoesNotPayRent	timedSystemEvent	repeatedly	27.09.2012 09:56:47:330	29.10.2012 07:16:53:582	The tenant does not pay the rent, because there is not enough money.
FamilyGrows	timedSystemEvent	repeatedly	27.09.2012 11:22:46:474	01.06.2013 07:25:31:982	A new child is born in this family, so they need more space and move o...
TenantChangesJob	timedSystemEvent	repeatedly	27.09.2012 11:31:16:358	22.06.2013 01:18:34:382	The Tenant changes the job and moves out of the building.

Abbildung 26: View System Events

Pro Event werden die folgenden Informationen dargestellt:

- **Name des Events**
- **Typ des Events:** Status-orientiert oder zeit-orientiert
- **Auftreten:** einmalig oder wiederholt
- **Nächstes geplantes Auftreten:** sowohl in simulierter Zeit als auch in Simulationszeit
- **Beschreibung**


a) Erweiterungsmöglichkeiten

Manipulationsmöglichkeiten: Um mit den Events experimentieren zu können, ist es hilfreich, wenn der Benutzer die Events verändern könnte. Dazu gehört, den Ausführungszeitpunkt zu verändern oder das Event komplett zu streichen.

3.3.5 Parameter Monitor

Kategorie	Monitoring
Sichtbar	Nein
Position	View-Stack unten

Tabelle 11: Attribute der View Parameter Monitor

Die View Parameter Monitor dient dazu, die Verläufe von Parametern darzustellen. Mit Hilfe der *Synchronize*-Schaltfläche () in der Werkzeugleiste lässt sich die Anzeige auf Echtzeit einstellen.

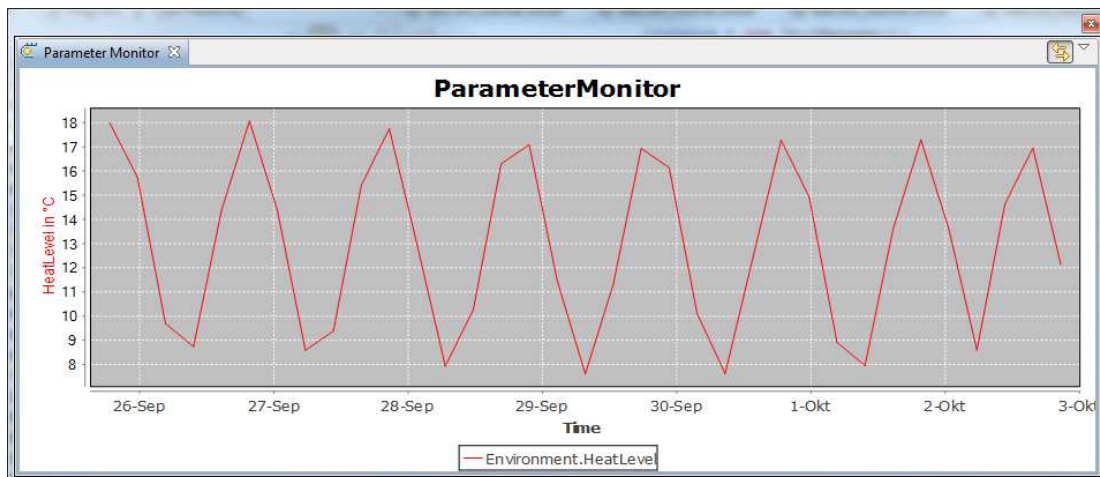


Abbildung 27: View Parameter Monitor

a) Verknüpfungen mit anderen Oberflächenkomponenten

Diese View kann auch mehrere Parameter anzeigen, hinzugefügt werden die anzuzeigenden Parameter über einen Kontext-Menüeintrag *Show History* des Parameters in der *Scenario Outline View* (siehe Kapitel 3.3.1).

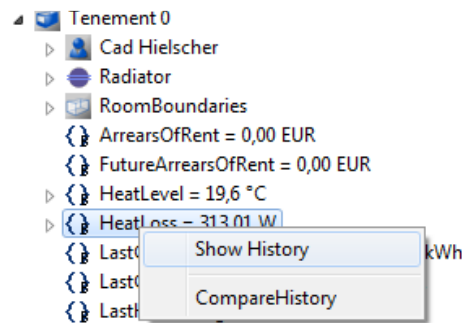


Abbildung 28: View Scenario Outline - Kontextmenü "Show History"

b) Grundlagen

Parameter-Historie

Grundlage für das Anzeigen von früheren Werten eines Parameters ist das Führen einer Historie des Parameters. Standardmäßig wird für die Parameter in SimuFrame eine Historie geführt. Dieses bedeutet, dass bei einer Änderung des Parameter-Wertes der bisherige Wert archiviert wird. Das erfolgt zum Zeitpunkt, bis zu dem der Wert Gültigkeit besaß. Die Werte der Historie lassen sich auch in der View *Scenario Outline* einsehen. Jedem Parameter, der historisierbar ist, sind die historischen Werte zeitlich absteigend untergeordnet (Abbildung 29).

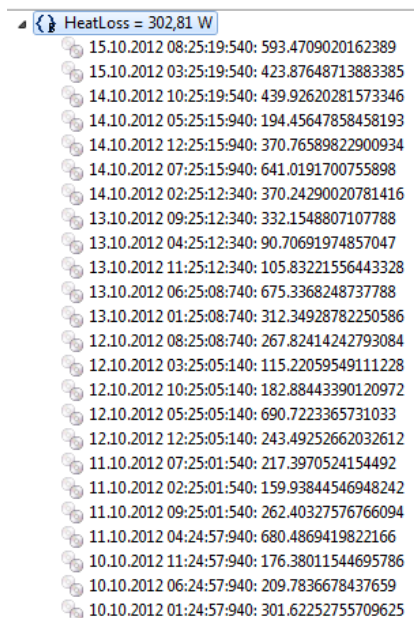


Abbildung 29: Beispiel Parameter Historie

3.3.6 Parameter Comparison

Kategorie	Monitoring
Sichtbar	Nein
Position	View-Stack unten

Tabelle 12: Attribute der View Parameter Comparison

Die View *Parameter Comparison* (Abbildung 30) basiert technisch auf der View *Parameter Monitor*. Mit Hilfe dieser View ist es möglich, Zeitreihen der Werte verschiedener Parameter zu vergleichen. Im Gegensatz zum *Parameter Monitor* lässt sich in der View *Parameter Comparison* der Zeitraum eingrenzen, in dem die Parameterwerte verglichen werden. Hierzu wird ein Dialog genutzt, der nach der Auswahl des Eintrags *Compare History* aus dem Kontext-Menü eines Parameters erscheint (siehe Abbildung 31).

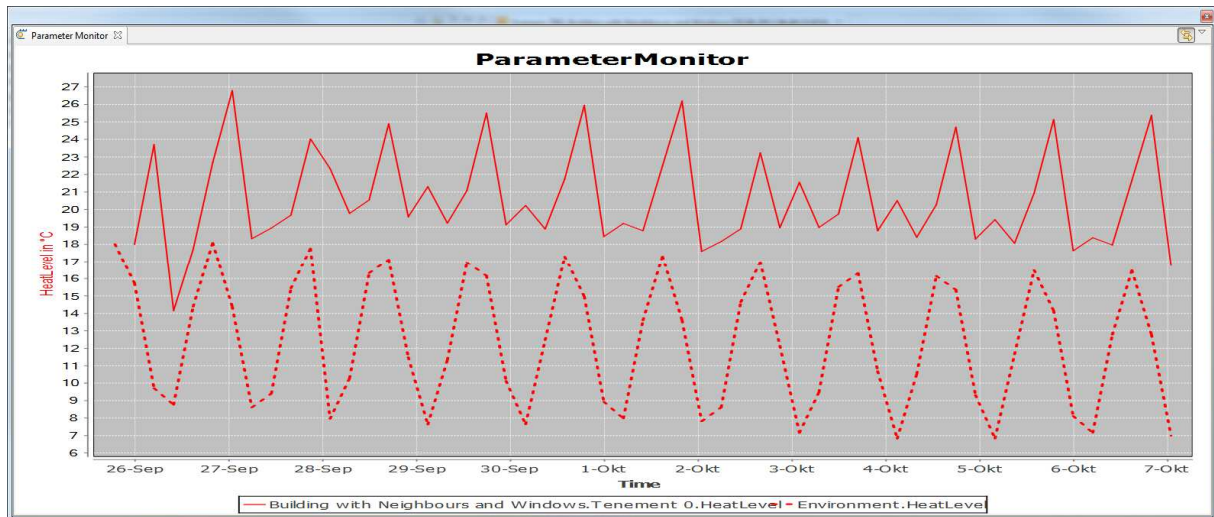


Abbildung 30: View Parameter Comparison

Parameter	Start	End
Environment.HeatLevel	2012-09-27	2012-09-28

Abbildung 31: Dialog Parameter-Auswahl

Der Dialog *Parameter Compare Options* (Abbildung 31) hat verschiedene Eingabeoptionen:

- Type of Comparison:** Der Typ des Vergleichs definiert, ob unterschiedliche Parameter in derselben Periode dargestellt werden sollen oder ein Parameter in verschiedenen Perioden. Mit dem Vergleich unterschiedlicher Parameter in derselben Periode lassen sich eventuelle Zusammenhänge zwischen den Parametern darstellen, beispielsweise der Zusammenhang zwischen Außentemperatur, Innentemperatur und genutztem Heizwärme-strom. Der Vergleich unterschiedlicher Perioden desselben Parameters macht es möglich, den Einfluss von Spieleraktionen darzustellen. Zum Beispiel können die Kurven des Heizwärmeverbrauches für mehrere Perioden übereinandergelegt werden - jeweils mit unterschiedlichen Arten von Fenstern und Wärmedämmungen – und verglichen werden.

- **Parameter:** Für den Fall des Vergleiches von mehreren Parametern über eine Zeitperiode müssen diese Parameter ausgewählt werden. Das geschieht mit Hilfe der Tabelle Parameter und der Schaltfläche *Add Item*.

3.3.7 Action Log

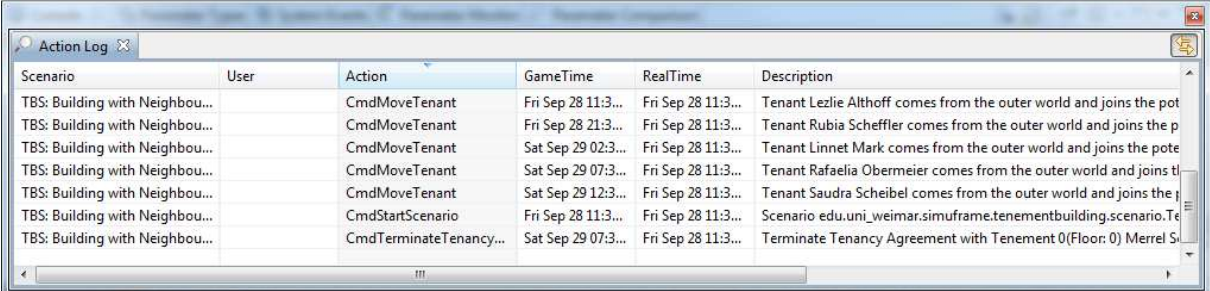
Kategorie	Monitoring
Sichtbar	Nein
Position	View-Stack unten

Tabelle 13: Attribute der View Action Log

In der View Action Log (Abbildung 32) werden die bei der Ausführung von Szenarien anfallenden Ereignisse dargestellt. Ein Eintrag im Action Log ist programmatisch zu erstellen, d.h. es finden sich nur Einträge im Action Log wieder, für die das bei der Entwicklung des Szenario so vorgesehen war, es werden keine Einträge automatisch erzeugt.

In der Tabelle werden pro Eintrag die folgenden Informationen angezeigt:

- **Erzeugendes Szenario**
- **Benutzer**³⁸
- **Auslösende Action:** Oft ist Benutzeraktion für den Eintrag verantwortlich. Alternative Auslöser einer Action können SystemEvents sein.
- **Zeitpunkte:** Der ActionLog-Eintrag wird zu einem bestimmten Zeitpunkt erzeugt, dieser wird hier sowohl als simulierter Zeitpunkt (*Game Time*) als auch als Simulationszeitpunkt (*Real Time*) vermerkt.
- **Beschreibung:** Das auftretende Ereignis wird mit Hilfe eines Beschreibungstextes näher erläutert. Beispielsweise werden bei einem Einzug eines Mieters der Name des Mieters sowie die Wohnung genannt.



Scenario	User	Action	GameTime	RealTime	Description
TBS: Building with Neighbou...		CmdMoveTenant	Fri Sep 28 11:3...	Fri Sep 28 11:3...	Tenant Lezie Althoff comes from the outer world and joins the pot
TBS: Building with Neighbou...		CmdMoveTenant	Fri Sep 28 21:3...	Fri Sep 28 11:3...	Tenant Rubia Scheffler comes from the outer world and joins the p
TBS: Building with Neighbou...		CmdMoveTenant	Sat Sep 29 02:3...	Fri Sep 28 11:3...	Tenant Linnet Mark comes from the outer world and joins the pote
TBS: Building with Neighbou...		CmdMoveTenant	Sat Sep 29 07:3...	Fri Sep 28 11:3...	Tenant Rafaelia Obermeier comes from the outer world and joins t
TBS: Building with Neighbou...		CmdMoveTenant	Sat Sep 29 12:3...	Fri Sep 28 11:3...	Tenant Saudra Scheibel comes from the outer world and joins the
TBS: Building with Neighbou...		CmdStartScenario	Fri Sep 28 11:3...	Fri Sep 28 11:3...	Scenario edu.uni_weimar.simuframe.tenementbuilding.scenario.Te
TBS: Building with Neighbou...		CmdTerminateTenancy...	Sat Sep 29 07:3...	Fri Sep 28 11:3...	Terminate Tenancy Agreement with Tenement 0(Floor: 0) Merrel S

Abbildung 32: View Action Log

a) Erweiterungsmöglichkeiten

Filter- und Suchfunktion

Auch im ActionLog ist es sinnvoll, nach verschiedenen Feldinhalten zu filtern und gezielt nach enthaltenen Zeichenfolgen suchen zu können. Die Sortierung nach Werten einer einzelnen Spalte ist bereits möglich.

3.4 Views des Skill-Manager-Moduls

Die folgenden Views sind im Modul `edu.uni_weimar.simuframe.skillmanager.ui` definiert³⁹.

³⁸ Bei einem Mehrspielerspiel ist es notwendig, einzelne Benutzer zu unterscheiden. Deswegen ist eine Anmeldung erforderlich. Das Konzept der Plattform sieht eine Benutzerverwaltung vor, jedoch ist in der gegenwärtigen Version noch keine Anmeldung notwendig, daher ist das Feld „User“ leer.

3.4.1 Skill Tree Admin

Kategorie	Administration
Sichtbar	Nein
Position	View-Stack unten

Tabelle 14: Attribute der View Skill Tree Administration

Die *View Skill Tree Admin* wird genutzt, um dem Szenario-Entwickler bzw. Administrator einen Überblick über die auf der Plattform vorhandenen Fähigkeiten zu geben. Durch das erfolgreiche Lösen von Missionen können die Fähigkeiten durch den Spieler erworben werden. Das Erlernen einer Fähigkeit ist dann beendet, wenn eine definierte Mindestanzahl an Punkten erreicht wurde. Die Fähigkeiten sind in Kategorien unterteilt, in der Darstellung (Abbildung 33) sind dies *acoustics* und *heat*, sie sind hellblau hinterlegt.

Fähigkeiten können voneinander abhängig sein, d.h. einige Fähigkeiten können erst erworben werden, wenn alle vorausgesetzten Fähigkeiten als vollständig erlernt gelten. Die Abhängigkeiten werden im Diagramm durch die Pfeile angezeigt, die Pfeilspitze zeigt in Richtung der abhängigen Fähigkeit. Die Zahlen auf den Pfeilen geben die Anzahl der notwendigen Punkte an.

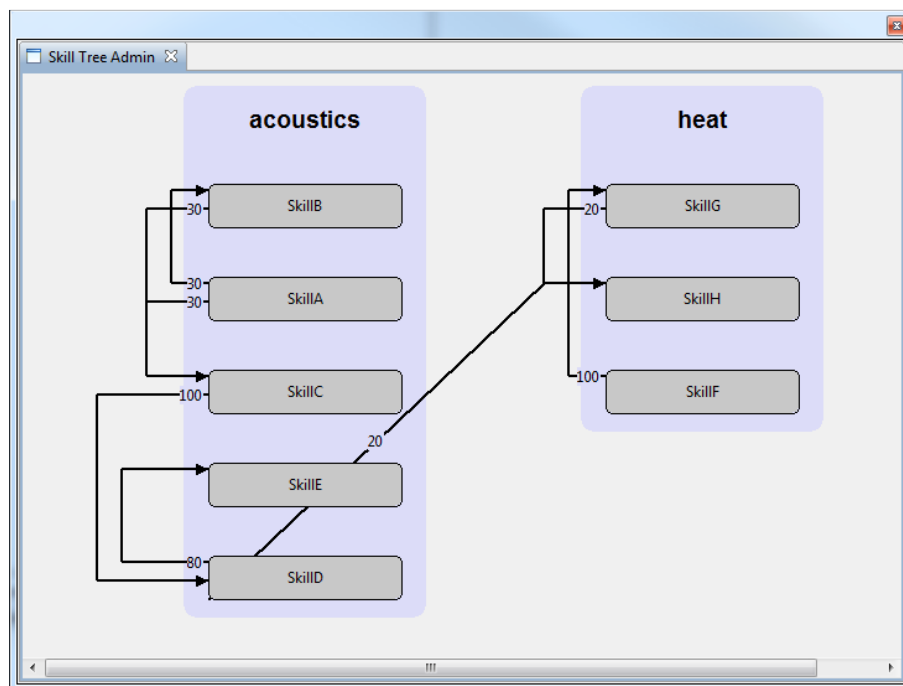


Abbildung 33: View Skill Tree Admin

a) Erweiterungsmöglichkeiten

Filter- und Suchfunktion

Um einen schnellen Zugriff auf einzelne Fähigkeiten zu erlauben, sollte es möglich sein, mit Hilfe einer Suchfunktion eine bestimmte Fähigkeit zu identifizieren und in den Mittelpunkt der Grafik zu stellen.

Editor

Im Moment sind keine Editieraktionen am Fähigkeitsgraphen möglich, die zugrundeliegenden Daten werden ausschließlich über *Extensions* definiert. Diese View könnte zum Editor ausgebaut

³⁹ Sie wurden maßgeblich durch die Arbeit von Janina Held entwickelt (Held, 2011).

werden, der es ermöglicht, neue Fähigkeiten im visuellen Kontext zu definieren und ebenfalls Abhängigkeiten zwischen den Fähigkeiten zu erzeugen.

Missionen

Zu jeder Fähigkeit können auch die Missionen (und die zugehörigen Szenarien, in denen die Missionen ausgelobt werden) gezeigt werden, deren Bewältigung Punkte für die jeweilige Fähigkeit ergibt. Damit lässt sich eine Übersicht erhalten, welche Möglichkeiten es gibt, um die notwendigen Punkte zu erzielen.

3.4.2 Skill Tree

Kategorie	Monitoring
Sichtbar	Nein
Position	View-Stack unten

Tabelle 15: Attribute der View Skill Tree

Die View *Skill Tree* (Abbildung 34) ist eine Abwandlung der View *Skill Tree Admin* aus dem vorigen Kapitel 3.4.1: Diese View ist benutzerorientiert, d.h. sie zeigt den Status des angemeldeten Spielers. In den roten Ellipsen finden sich die aktuell erreichten Punktzahlen jeden Spielers für die jeweilige Fähigkeit. Im unteren Bereich werden die Auszeichnungen des Spielers angezeigt. Im angezeigten Beispiel wurden bisher keine Auszeichnungen vergeben.

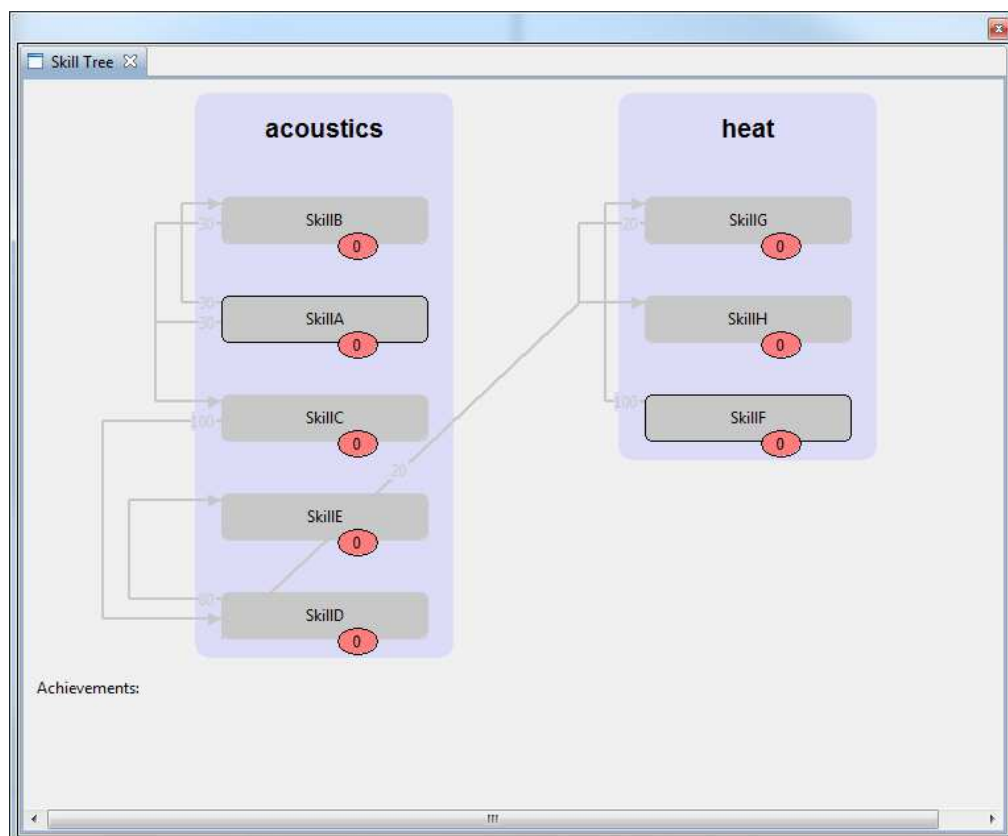


Abbildung 34: View Skill Tree

a) Erweiterungsmöglichkeiten

Missionsassistent

Zu jeder Fähigkeit, die schon freigeschaltet ist, d.h. deren vorausgesetzte Fähigkeiten ausreichend erworben wurden, können diejenigen Missionen (bzw. die assoziierten Szenarien) ange-

zeigt werden, die erfüllt werden müssen, um die notwendigen Punkte zum „Erlernen“ der Fähigkeit zu erhalten. Damit hat der Spieler eine Art Missionsassistent zur Verfügung, der ihm anzeigt, wie er auf schnellem Wege die einzelnen Fähigkeiten erwerben kann.

3.5 Views des Moduls Missionsmanager

Die folgenden Views sind im Modul `edu.uni_weimar.simuframe.missionmanager.ui` definiert⁴⁰.

3.5.1 Mission Administration

Kategorie	Administration
Sichtbar	Nein
Position	View-Stack unten

Tabelle 16: Attribute der View Mission Administration

Die View *Mission Administration* (Abbildung 36) bietet eine Übersicht über die auf der Plattform bekannten Missionen. Die Tabelle im oberen Teil der View zeigt die Felder mit dem Namen der Mission und ihrer Beschreibung. Die Auswahl einer Mission in der Tabelle löst eine Anzeige der Details dieser Mission im unteren Teil der View aus (*Mission Inspector*). Zu den Detailangaben einer Mission gehören zusätzlich ein Identifikator und die vorausgesetzten Fähigkeiten mit geforderten Punktzahlen. Weiterhin werden die möglichen Zugewinne an Fähigkeitspunkten durch diese Mission angezeigt.

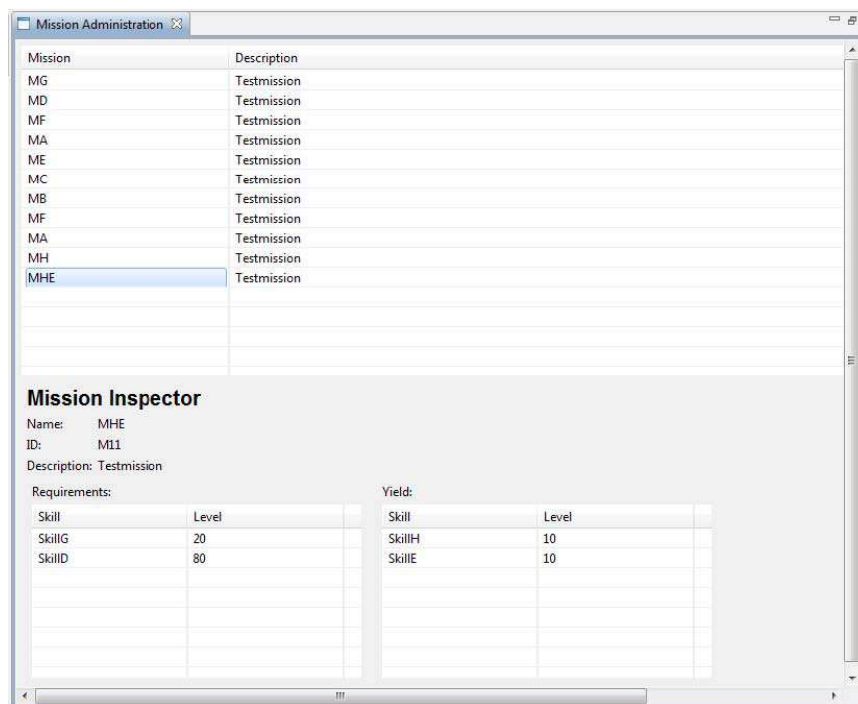


Abbildung 35: View Mission Administration

3.5.2 Mission

Kategorie	Test-Client
Sichtbar	Nein
Position	View-Stack unten

Tabelle 17: Attribute der View Mission

⁴⁰ Das Modul MissionManager wurde ebenso wie das Modul SkillManager maßgeblich durch die Arbeit von Janina Held entwickelt (Held, 2011).

Die View *Mission* kann zum Test der Missionen und der Verteilung der Fähigkeitepunkte genutzt werden. Angezeigt werden in einer Tabelle der Name der Mission, die Beschreibung und die verdienbaren Fähigkeitepunkte. Es werden nur diejenigen Missionen gezeigt, deren Voraussetzungen durch den aktuellen Spieler schon erfüllt werden.

[illegible]

Abbildung 36: View Mission

Unter der Tabelle ist eine *Play*-Schaltfläche platziert, die die Ausführung der Mission simuliert. Dabei werden die eigentlich zur Vollendung der Mission notwendigen Bedingungen nicht überprüft. Primäres Ziel der View ist der Test der korrekten Verarbeitung von ausgegebenen Fähigkeitspunkten.

4 Module

4.1 Namenskonventionen

Zur Benennung der Module werden die folgenden Namenskonventionen genutzt:

- **Präfix:** Die Namen starten mit `edu.uni_weimar.simuframe`.
- **UI-Modul**⁴¹: Zu einem Logikmodul gehört in der Regel ein UI-Modul: Diese Module werden benannt, indem die Endung `.ui` an den Namen des UI-Moduls angehängt wird.
- **Interface-Module:** Interface-Module enthalten Programmierschnittstellen. Sie sollen sicherstellen, dass auf Modulebene eine Trennung zwischen Schnittstelle und Implementierung gewährleistet ist. Für aufbauende Module wird eine Abhängigkeit zum Schnittstellenmodul eingeführt. Das implementierende Modul ist nicht bekannt und wird erst zur Laufzeit, nicht aber zur Übersetzungszeit benötigt.
- **Test-Module:** Testmodule enthalten Unit-Tests. Das sind Tests, die automatisiert ablaufen können und somit helfen, die korrekte Funktionalität der Module sicherzustellen. Diese Module erhalten die Endung `tests` – angehängt an den Namen des Moduls, dessen Elemente getestet werden⁴².

⁴¹ UI: user interface: Benutzerschnittstelle im Sinne von Benutzeroberfläche.

⁴² Es sollte Testmodule für jedes Modul geben. Aus Gründen des Zeitaufwandes ist dieses für den Prototypen jedoch nicht der Fall.

- **Dokumentationsmodule:** Es besteht die Möglichkeit, zum Eclipse Hilfe-Modul beizutragen. Beiträge zur Online-Hilfe werden in Module mit der Endung `.doc` ausgelagert.
- **Szenariomodul:** Diese Module stellen Szenarios bereit. Der vorletzte Namensbestandteil ist `.scenario.`, darauf folgt die Benennung des Szenarios.

4.2 Liste der Module

Tabelle 18 enthält eine Gesamtübersicht der im SimuFrame-Framework enthaltenen Module, die in den folgenden Kapiteln gruppenweise beschrieben werden.

Modul	Gruppe
edu.uni_weimar.simuframe	Kern
edu.uni_weimar.simuframe.acoustics	Akustik
edu.uni_weimar.simuframe.acoustics.tests	Akustik
edu.uni_weimar.simuframe.acoustics.ui	Akustik
edu.uni_weimar.simuframe.client	Client-Server: Test-Client
edu.uni_weimar.simuframe.client.tests	Client-Server: Test-Client
edu.uni_weimar.simuframe.client.ui	Client-Server: Test-Client
edu.uni_weimar.simuframe.doc	Kern
edu.uni_weimar.simuframe.interfaces	Kern: Interface
edu.uni_weimar.simuframe.missionmanager	Kern
edu.uni_weimar.simuframe.missionmanager.ui	Kern
edu.uni_weimar.simuframe.protocol	Client-Server: Kommunikation
edu.uni_weimar.simuframe.protocol.tests	Client-Server: Kommunikation
edu.uni_weimar.simuframe.scenario.tenantTaxonomy	Fachlich: Szenario
edu.uni_weimar.simuframe.scenario.tenementbuilding1	Fachlich: Szenario
edu.uni_weimar.simuframe.server	Client-Server: Server
edu.uni_weimar.simuframe.server.tests	Client-Server: Server
edu.uni_weimar.simuframe.skillmanager	Kern
edu.uni_weimar.simuframe.smartfoxserver.protocol	Client-Server: Kommunikation
edu.uni_weimar.simuframe.skillmanager.ui	Kern
edu.uni_weimar.simuframe.tenementbuilding	Fachlich: Gebäude
edu.uni_weimar.simuframe.tenementbuilding.heat	Fachlich: Wärme
edu.uni_weimar.simuframe.tenementbuilding.heat.tests	Fachlich: Wärme
edu.uni_weimar.simuframe.tenementbuilding.heat.ui	Fachlich: Wärme
edu.uni_weimar.simuframe.tenementbuilding.tenant	Fachlich: Mieter
edu.uni_weimar.simuframe.tenementbuilding.-tenementcontract_heat	Fachlich: Vertrag
edu.uni_weimar.simuframe.tenementbuilding.tests	Fachlich: Gebäude
edu.uni_weimar.simuframe.tenementbuilding.ui	Fachlich: Gebäude
edu.uni_weimar.simuframe.tenementcontract	Fachlich: Vertrag
edu.uni_weimar.simuframe.tenementcontract.ui	Fachlich: Vertrag
edu.uni_weimar.simuframe.tests	Kern
edu.uni_weimar.simuframe.ui	Kern
edu.uni_weimar.simuframe.ui.interfaces	Kern: Interface

Tabelle 18: Liste der Module (Plug-ins)

4.3 Statistik

Das Werkzeug „EMMA“⁴³ wird eingesetzt für die Ermittlung der Codeabdeckung durch automatisierte Tests, d.h. die Ergebnisse geben an, welche Codezeilen bisher nicht durch automatisierte Tests überprüft werden. Als Nebenprodukt wird auch eine Übersicht des Umfangs des erstellten Rechenkerns *SimuFrame* erzeugt. Ein in Tabelle 1 dargestellter Ausschnitt der Ergebnisse zeigt

⁴³ EMMA ist frei verfügbar über die Projekt-Website <http://emma.sourceforge.net/>, letzter Zugriff 12.10.2012.

u.a. die Anzahl der Klassen-Dateien (294) sowie die Anzahl der Codezeilen (10504), aus denen *SimuFrame* besteht.

total packages:	93
total executable files:	294
total classes:	371
total methods:	2280
total executable lines:	10504

Tabelle 1: EMMA Overall Stats Summary

4.4 Einordnung der Module

4.4.1 Kern-Module

Die Kern-Module stellen die Funktionalitäten bereit, die für den Simulationskern benötigt werden. Sie enthalten keine fachlichen Inhalte, sondern bieten übergeordnete Dienste, die von den fachlichen Modulen genutzt werden können.

- `edu.uni_weimar.simuframe` beinhaltet die Simulation-Engine, abstrakte Basisklassen und Manager-Klassen zur Verwaltung der wesentlichen Elemente der Simulationsmodelle.
- `edu.uni_weimar.simuframe.doc` enthält die kontextsensitive Online-Dokumentation.
- `edu.uni_weimar.simuframe.missionmanager` enthält die Logik zur Verwaltung der Missionen und stellt einen Extension Point zur Definition von Missionen bereit.
- `edu.uni_weimar.simuframe.missionmanager.ui` beinhaltet die Benutzeroberflächenanteile für den Missionsmanager (`edu.uni_weimar.simuframe.missionmanager`).
- `edu.uni_weimar.simuframe.skillmanager` stellt die Funktionalität zur Verwaltung der Fähigkeiten bereit und definiert Schnittstellen (Extension Points) zur Definition weiterer Fähigkeiten und Abhängigkeiten von Fähigkeiten.
- `edu.uni_weimar.simuframe.skillmanager.ui` enthält Erweiterungen der Benutzeroberfläche bezüglich des Moduls Skillmanager (`edu.uni_weimar.simuframe.skillmanager`).
- `edu.uni_weimar.simuframe.tests` enthält Unit-Tests für das zentrale Kern-Modul `edu.uni_weimar.simuframe`.
- `edu.uni_weimar.simuframe.ui` enthält die Anteile an der Benutzerschnittstelle für das Kern-Modul `edu.uni_weimar.simuframe`.

a) Interface-Module der Kernmodule

Interface-Module dienen der Entkopplung der Kern-Module von den fachlichen Modulen. Die fachlichen Module benutzen die Schnittstellen (Interfaces) der Kern-Module – unabhängig von der aktuellen tatsächlichen Implementierung der Kernmodule. Dies erlaubt den späteren Austausch der Kernmodule oder das Redesign ohne Einfluss auf die fachlichen Module, solange die durch die Interfaces fixierte Funktionalität beibehalten wird. Dieses Vorgehen wird durch die Erstellung von Unit-Tests unterstützt. Sie dokumentieren das Verhalten der zu testenden Klassensysteme und melden etwaige Verhaltensänderungen durch Anpassung der Implementierung.

- `edu.uni_weimar.simuframe.interfaces` enthält die zu implementierenden Schnittstellen für die Elemente des Simulationsmodells, wie Simulationselemente, Parameter, Systemereignisse und Benutzeraktionen.

- `edu.uni_weimar.simuframe.ui.interfaces` enthält die Schnittstellen für die SimuFrame-spezifischen Erweiterungen der Benutzeroberfläche.

4.4.2 Fachliche Module

Fachliche Module nutzen die Funktionalitäten, die durch die Kernmodule bereitgestellt werden, um fachliche Inhalte (vornehmlich der Bauphysik) darzustellen. Gleichzeitig bilden sie im Sinne des Modularitätsprinzips auch eine Hierarchie von aufeinander aufbauenden Modulen.

a) Gebäude

Die Module dieser Gruppe stellen die Funktionalitäten bereit zur Darstellung eines Gebäudes. Sie tragen die Simulationselemente bei, mit deren Hilfe ein Modell aufgespannt werden kann, in das durch weitere Modulgruppen beigetragene Parameter eingehängt werden können.

- `edu.uni_weimar.simuframe.tenementbuilding` stellt im Wesentlichen Simulationselemente zum Aufbau eines Gebäudes bereit, wie beispielsweise Wände und Fenster.
- `edu.uni_weimar.simuframe.tenementbuilding.ui` stellt die Erweiterungen zur Benutzeroberfläche für das Modul `edu.uni_weimar.simuframe.tenementbuilding` bereit.
- `edu.uni_weimar.simuframe.tenementbuilding.tests` beinhaltet Unit-Tests für das Modul `edu.uni_weimar.simuframe.tenementbuilding`.

b) Mieter

Die Mieter sind in Form von NPCs⁴⁴ realisiert. Das erfordert einen erhöhten Aufwand für Algorithmen zur Steuerung der Mieter. Daher wurde ein eigenes Modul zur Abbildung der Mieter bereitgestellt.

- `edu.uni_weimar.simuframe.tenementbuilding.tenant` enthält die Algorithmen zur Steuerung des Mieters. Wichtiger Parameter ist die Zufriedenheit. An diesen sind Algorithmen geknüpft, die die Entscheidung über Auszug oder Verbleib in der Wohnung treffen.

c) Akustik

Die Modulgruppe Akustik implementiert ausgewählte Aspekte der Akustik. Wie bei allen fachlichen Modulen ist das eine gewählte Darstellung der Aspekte, die weder den Anspruch auf vollständige und umfassende Darstellung des fachlichen Gebietes erhebt noch die einzig mögliche Variante einer Darstellung ist. Sie demonstriert lediglich, dass es möglich ist, mit Hilfe des dargestellten Frameworks fachliche Aspekte abzubilden.

- `edu.uni_weimar.simuframe.acoustics` trägt Parameter zur Darstellung des Wissensgebietes um das Schalldämmmaß bei.
- `edu.uni_weimar.simuframe.acoustics.ui` erweitert die Oberfläche zur Darstellung der Aspekte des Moduls `edu.uni_weimar.simuframe.acoustics`.
- `edu.uni_weimar.simuframe.acoustics.tests` enthält die Unit-Tests für das Modul `edu.uni_weimar.simuframe.acoustics`.

⁴⁴ Non-Player Character: Spielfiguren, die durch den Rechner gesteuert werden.

d) Wärmeschutz

Wärmeschutz ist neben Akustik der zweite Aspekt, der im Rahmen der prototypischen Realisierung abgebildet wurde. Auch das ist ebenfalls lediglich eine von vielen möglichen, partiellen Abbildungen.

- `edu.uni_weimar.simuframe.tenementbuilding.heat`: Durch dieses Modul werden Parameter wie U-Wert sowie Simulationselemente wie Heizung und Heizkörper zum Simulationsmodell beigetragen.
- `edu.uni_weimar.simuframe.tenementbuilding.heat.ui` fügt die modulspezifischen Ergänzungen zur Benutzeroberfläche hinzu.
- `edu.uni_weimar.simuframe.tenementbuilding.heat.tests` enthält die Unit-Tests für das Modul `edu.uni_weimar.simuframe.tenementbuilding.heat`.

e) Vertrag

Die Modulgruppe Vertrag unterstützt den Spielaspekt der Szenarien. Es werden keine weiteren fachlichen - im Sinne von bauphysikalischen - Zusammenhänge ergänzt, sondern dieses Modul dient zur Darstellung der wirtschaftlichen Folgen der bauphysikalischen Phänomene. Eine wesentliche Auswirkung ist die Kündigung durch den Mieter und sein Auszug, falls die Zufriedenheit des Mieters dauerhaft einen Schwellwert unterschreitet mit der Konsequenz einer leer stehenden Wohnung und fehlenden Mieteinnahmen für den Vermieter. Daher werden in dieser Modulgruppe Elemente wie Mietvertrag und Mietzahlungen abgebildet. Um den spielerischen Aspekt zu unterstützen, werden auch NPC-Aktionen, wie ausbleibende Mietzahlungen, modelliert.

- `edu.uni_weimar.simuframe.tenementcontract` trägt Parameter wie Miete und Mietrückstände sowie Ereignisse wie Ausfall der Mietzahlung zum Szenario bei.
- `edu.uni_weimar.simuframe.tenementcontract.ui` ist zuständig für die Ergänzungen der Benutzeroberfläche um Aspekte des Moduls `edu.uni_weimar.simuframe.tenementcontract`.
- `edu.uni_weimar.simuframe.tenementbuilding.tenementcontract_heat` ist ein sogenanntes Verbindungsmodul: es stellt eine Verbindung zwischen den Modulen `edu.uni_weimar.simuframe.tenementcontract` und `edu.uni_weimar.simuframe.tenementbuilding.heat` her. Konkret ermöglicht es die Verbuchung von Heizkosten.

f) Szenariomodule

Szenariomodule dienender Bereitstellung von Szenarien für den Spieler. Sie definieren eine Menge von zugrundeliegenden Modulen und stellen auf der Basis der durch diese Module beigetragenen Elemente Szenarien bereit, die vom Spieler gelöst werden können.

- `edu.uni_weimar.simuframe.scenario.tenementbuilding1` enthält verschiedene Szenarien, die Aspekte zum Wärmeschutz und zur Akustik demonstrieren.
- `edu.uni_weimar.simuframe.scenario.tenantTaxonomy` konzentriert sich in den bereitgestellten Szenarien auf einen alternativen Weg, die Mieterzufriedenheit festzustellen.

4.4.3 Client-Server-Unterstützung

Die Spielplattform unterstützt mehrere Spieler und sieht eine Client-Server-Architektur vor. Daher sind Module notwendig, die den notwendigen Rahmen für Client-Server-Verteilungen bereitstellen.

a) Server

Der Server wird mit Hilfe der Software SmartFoxServer("SmartFoxServer," 2012) realisiert. Ein Adaptermodul sorgt für die Verbindung des SmartFoxServers mit dem SimuFrame-Framework.

- `edu.uni_weimar.simuframe.server` stellt Adapterfunktionalitäten zum SmartFoxServer bereit für das SimuFrame-Framework bereit und sorgt für die Verteilung der Anfragen von den Clients.
- `edu.uni_weimar.simuframe.server.tests` trägt Unit-Tests für das Modul `edu.uni_weimar.simuframe.server` bei.

b) Test-Client

Ein Test-Client erlaubt es, die korrekte Verarbeitung von Client-Anfragen durch den Server zu testen. Gleichzeitig können durch den Test-Client auch gezielt Anfragen zusammengestellt werden, die im Gegensatz zum Test durch einen Spielclient mit geringerem Vorbereitungs-Aufwand und damit kürzeren Testzyklen genutzt werden können.

- `edu.uni_weimar.simuframe.client` erlaubt das Senden von Anfragen an den Server und die Auswertung der Antworten.
- `edu.uni_weimar.simuframe.client.ui` stellt die notwendigen Erweiterungen der Benutzeroberfläche bereit, die das Senden von Anfragen unterstützen.
- `edu.uni_weimar.simuframe.client.tests` trägt Unit-Tests für das Modul `edu.uni_weimar.simuframe.client` bei.

c) Client-Server Kommunikation

In einem Client-Server-System muss Kommunikation zwischen Client und Server stattfinden: Die Clients senden Anfragen und der Server sendet entsprechende Antworten. Im Rahmen der Spielplattform werden häufig Objekte und zugehörige Parameter-Werte ausgetauscht. Diese Funktionalität muss zum einen bereitgestellt werden und zum anderen gekapselt vorliegen, so dass ein Austausch des Kommunikationsprotokolls die fachlichen Module nicht berührt. Diese Aufgabe wird durch die Module dieser Gruppe übernommen.

- `edu.uni_weimar.simuframe.protocol` stellt grundlegende Mechanismen zur Übertragung von Objekten als Parameter über eine internetbasierte Kommunikation bereit.
- `edu.uni_weimar.simuframe.smartfoxserver.protocol` adaptiert die Funktionalität des Moduls `edu.uni_weimar.simuframe.protocol` an die Eigenheiten des zugrundeliegenden SmartFoxServers.
- `edu.uni_weimar.simuframe.protocol.tests` enthält Unit-Tests für die Funktionalitäten des Moduls `edu.uni_weimar.simuframe.protocol`.

5 Extension Points

5.1 Beschreibungsschema

Im Folgenden werden die Beschreibungen der Extension Points dargelegt, wie sie durch die Eclipse IDE generiert werden. Extension Points werden über eine XML-Schemadatei⁴⁵ beschrieben, die in der Dokumentation teilweise wiedergegeben wird.

⁴⁵ Für XML-Schemata gibt es vom *World Wide Web Consortium* (W3C) eine Norm, (<http://www.w3.org/XML/Schema>, Abruf am 03.09.2012), die auch für dieses Schema angewendet wird.

5.2 simulationElementParameter

Identifier:

edu.uni_weimar.simuframe.simulationElementParameter

Since:

Version 0.4

Description:

This extension point is used to create a link between Simulation Element and Parameter.

Configuration Markup:

```
<!ELEMENT extension (simuElementParameter+)>
```

```
<!ATTLIST extension
```

```
point CDATA #REQUIRED
```

```
id CDATA #IMPLIED
```

```
name CDATA #IMPLIED
```

```
>
```

```
<!ELEMENT sourceAlgorithm EMPTY>
```

```
<!ATTLIST sourceAlgorithm
```

```
class CDATA #IMPLIED
```

```
attribute CDATA #IMPLIED
```

```
>
```

A source algorithm describes a way how a parameter is calculated.

- **class** - The class which is instantiated.
- **attribute** - Attribute used for configuration

```
<!ELEMENT simuElementParameter (simuElement , initialValue? , sourceAlgorithm? ,  
initialValue2?)>
```

```
<!ATTLIST simuElementParameter
```

```
parameterId IDREF #REQUIRED
```

```
defaultValue CDATA #IMPLIED
```

```
unit CDATA #IMPLIED
```

```
>
```

- **parameterId** - The id of the parameter (declared by an extension)
- **defaultValue** - The default value for the parameter in this connection between simulation element and parameter.
- **unit** - Unit of the parameter, overrides the unit of the parameter.

```
<!ELEMENT simuElement EMPTY>
```

```
<!ATTLIST simuElement
```

```
simuElementId IDREF #IMPLIED
```

```
>
```

The simulationElement to which this parameter belongs. One of the attributes has to be filled.

- **simuElementId** - The Id of the simulationElement.

The **initialValue** element is deprecated

```
<!ELEMENT initialValue EMPTY>
```

```
<!ATTLIST initialValue
```

```
averageValue CDATA #IMPLIED
```

maxValue CDATA #IMPLIED
minValue CDATA #IMPLIED
distribution (Gaussian|Equal)

>

The element initialValue is used to generate an initial value for this parameter.

- **averageValue** - Average value
- **maxValue** - The max value (if applicable)
- **minValue** - The minimum value (if applicable)
- **distribution** - Distribution function of the initial value.

<!ELEMENT initialValue2 EMPTY>

<!ATTLIST initialValue2

initialValueProvider CDATA #IMPLIED

>

- **initialValueProvider** – A link to a class which generates an initial value for the attached parameter.

Examples:

```
<extension
  point="edu.uni_weimar.simuframe.simulationElementParameter">
  <simulationElementParameter
    parameterId="edu.uni_weimar.simuframe.acoustics.NNoiseLevel">
    <simulationElement
      class="edu.uni_weimar.simuframe.tenementbuilding.element.Tenement">
    </simulationElement>
  </simulationElementParameter>
  <simulationElementParameter
    defaultValue="50"
    parameterId="edu.uni_weimar.simuframe.acoustics.ComfortableNoiseLevel">
    <simulationElement
      class="edu.uni_weimar.simuframe.tenementbuilding.element.Tenant">
    </simulationElement>
  </simulationElementParameter>
  <simulationElementParameter
    defaultValue="0"
    parameterId="edu.uni_weimar.simuframe.acoustics.NoiselevelSatisfaction">
    <simulationElement
      class="edu.uni_weimar.simuframe.tenementbuilding.element.Tenant">
    </simulationElement>
  </simulationElementParameter>
</extension>
```

API Information:

Either initialValue or initialValue2 has to be supplied.-

Supplied Implementation:

-

5.3 parameterSource

Identifier:

edu.uni_weimar.simuframe.parameterSource

Since:

Version 0.3

Description:

This extension point allows the declaration of parameters. Parameters are defined by several attributes and the source algorithm, which describes how the parameter value is generated.

Configuration Markup:

```

<!ELEMENT extension (parameter+)>
<!ATTLIST extension
point CDATA #REQUIRED
id CDATA #IMPLIED
name CDATA #IMPLIED
>

<!ELEMENT parameter (sourceAlgorithm?)?>
<!ATTLIST parameter
class CDATA #IMPLIED
name CDATA #IMPLIED
description CDATA #IMPLIED
defaultValue CDATA #IMPLIED
readOnly (true | false)
unit CDATA #IMPLIED
>

```

- **class** - The class which will be instantiated.
- **name** - The name of the parameter. This is a key to retrieve a parameter from the parameterlist, which is connected to any simulationElement.
- **description** - The description for the parameter. It is shown in the user interface.
- **defaultValue** - Textual representation of the default value for this parameter.
- **readOnly** - True: This parameter cannot be set. False: This parameter can be set. Default is false. If the parameter can be set, there will be an action in the user interface.
- **unit** - Unit of the parameter. There is a list of supported units in order to do automated conversions of parameter values during calculations (Class UnitManager).

```

<!ELEMENT sourceAlgorithm EMPTY>
<!ATTLIST sourceAlgorithm
class CDATA #IMPLIED
attribute CDATA #IMPLIED
>

```

A source algorithm describes a way how a parameter is calculated.

- **class** - The class which is instantiated.
- **attribute** - Attribute used for configuration

Examples:

```

<extension
  point="edu.uni_weimar.simuframe.parameterSource">
  <parameter
    class="edu.uni_weimar.simuframe.parameter.ParameterDouble"
    defaultValue="70"
    description="Noiselevel of the tenement"
    name="NNoiseLevel">
  </parameter>
  <parameter
    class="edu.uni_weimar.simuframe.parameter.ParameterDouble"
    description="Noiselevel which lets the tenant still feeling comfortable."
    name="ComfortableNoiseLevel">
  </parameter>
  <parameter
    class="edu.uni_weimar.simuframe.parameter.ParameterDouble"
    defaultValue="0"
    description="Satisfaction of the tenant with the noise level in his tenement"
  </parameter>
</extension>

```

```

        name="NoiselevelSatisfaction">
    <sourceAlgorithm
        class="edu.uni_weimar.simuframe.acoustics.algorithm.NoiseLevelSatisfactionCalculator"↵
    </sourceAlgorithm>
</parameter>
</extension>

```

Supplied Implementation:

-

5.4 weightedAverageContributor

Identifier:

edu.uni_weimar.simuframe.weightedAverageContributor

Since:

Version 0.4

Description:

There are parameters (target) which are calculated by using the values of some other parameters (sources). This extension point creates the link between target and sources.

Configuration Markup:

```

<!ELEMENT extension (contributor+)>
<!-- extension
point CDATA #REQUIRED
id CDATA #IMPLIED
name CDATA #IMPLIED
-->
<!-- contributor EMPTY
-->
<!-- contributor
targetParameterId CDATA #IMPLIED
contributingParameterId CDATA #IMPLIED
weight CDATA #IMPLIED
-->

```

- **targetParameterId** - Name of the targetParameter (name is used as an identifier)
- **contributingParameterId** - Id of the contributing parameter.
- **weight** - Factor with which the value gets multiplied.

Examples:

```

<extension
    point="edu.uni_weimar.simuframe.weightedAverageContributor">
    <contributor
        contributingParameterId="NoiselevelSatisfaction"

targetParameterId="edu.uni_weimar.simuframe.tenementbuilding.TenantSatisfaction"
        weight="1">
    </contributor>
</extension>

```

API Information:

The source parameters need to be numerical.

5.5 systemEvent

Identifier:

edu.uni_weimar.simuframe.systemEvent

Since:

Version 0.3

Description:

A system event is a sporadically occurring event which changes the simulation model. As an alternative the event can be triggered by parameter changes.

Configuration Markup:

```
<!ELEMENT extension (timedSystemEvent | parameterValueSystemEvent)+>
```

```
<!ATTLIST extension
```

```
point CDATA #REQUIRED
```

```
id CDATA #IMPLIED
```

```
name CDATA #IMPLIED
```

```
>
```

```
<!ELEMENT parameterValueSystemEvent (parameterEventCondition , eventAction ,  
parameterSource)>
```

```
<!ATTLIST parameterValueSystemEvent
```

```
name CDATA #REQUIRED
```

```
description CDATA #IMPLIED
```

```
>
```

System event which is triggered by a parameter value change.

- **name** – The name of the event.
- **description** – A description why the event is occurring

```
<!ELEMENT eventAction EMPTY>
```

```
<!ATTLIST eventAction
```

```
class CDATA #REQUIRED
```

```
>
```

- **class** – the java class which implements the model changes caused by the event.

```
<!ELEMENT timedSystemEvent (timedEventCondition , eventAction , simuElement)>
```

```
<!ATTLIST timedSystemEvent
```

```
name CDATA #REQUIRED
```

```
description CDATA #IMPLIED
```

```
>
```

- **name** – The name of the event.
- **description** – The description of the event: The reason for its occurrence and its consequences.

```
<!ELEMENT timedEventCondition EMPTY>
```

```
<!ATTLIST timedEventCondition
```

```
class CDATA #IMPLIED
```

```
frequency CDATA #REQUIRED
```

```
random (true | false)
```

```
>
```

- **class** – The java class which checks if the condition of the event is fulfilled.
- **frequency** - Frequency of the event: Number of seconds game time (not real time) after the event occurs.
- **random** - Should the event occur at equidistant points of time or randomly?

<!ELEMENT parameterEventCondition EMPTY>

<!ATTLIST parameterEventCondition

class CDATA #IMPLIED

>

Checks if the condition for this event is fulfilled.

- **class** – The class which checks the event prerequisites

<!ELEMENT simuElement EMPTY>

<!ATTLIST simuElement

simuElementId CDATA #IMPLIED

>

- **simuElementId** - Id of the simulation element (the id is build as the plugin name plus ". " plus SimuElement name.

<!ELEMENT parameterSource EMPTY>

<!ATTLIST parameterSource

parameterId IDREF #IMPLIED

>

- **parameterId** - The id of the parameter (declared by an extension)

5.6 scenarioFactory

Identifier:

edu.uni_weimar.simuframe.scenarioFactory

Since:

Version 0.3

Description:

Adds Scenario Factories

Configuration Markup:

<!ELEMENT extension (scenarioFactory+)>

<!ATTLIST extension

point CDATA #REQUIRED

id CDATA #IMPLIED

name CDATA #IMPLIED

>

<!ELEMENT scenarioFactory EMPTY>

<!ATTLIST scenarioFactory

class CDATA #IMPLIED

>

- **class** -

Examples:

<extension

```

    point="edu.uni_weimar.simuframe.systemEvent">
<timedSystemEvent
    description="The Tenant changes the job and moves out of the building."
    name="TenantChangesJob">
    <timedEventCondition
        class="edu.uni_weimar.simuframe.sytemEvent.TimerCondition"
        frequency="36000000"
        random="true">
    </timedEventCondition>
    <eventAction
        class="edu.uni_weimar.simuframe.tenementbuilding.systemEvent.action. ↵
            TenantMovesOutAction">
    </eventAction>
    <simuElement
        simuElementId="edu.uni_weimar.simuframe.tenementbuilding.Tenant">
    </simuElement>
</timedSystemEvent>
<timedSystemEvent
    description="A new child is born in this family, so they need more space and↵
        move out."
    name="FamilyGrows">
    <timedEventCondition
        class="edu.uni_weimar.simuframe.sytemEvent.TimerCondition"
        frequency="72000000"
        random="true">
    </timedEventCondition>
    <eventAction
        class="edu.uni_weimar.simuframe.tenementbuilding.systemEvent.action. ↵
            TenantMovesOutAction">
    </eventAction>
    <simuElement
        simuElementId="edu.uni_weimar.simuframe.tenementbuilding.Tenant">
    </simuElement>
</timedSystemEvent>
<timedSystemEvent
    description="A fire wipes out the complete building - it&apos;s ↵
        not habitable any longer."
    name="BuildingBurnsDown">
    <timedEventCondition
        class="edu.uni_weimar.simuframe.sytemEvent.TimerCondition"
        frequency="100000000"
        random="true">
    </timedEventCondition>
    <eventAction
        class="edu.uni_weimar.simuframe.tenementbuilding.systemEvent.action. ↵
            DestroyBuilding">
    </eventAction>
    <simuElement
        simuElementId="edu.uni_weimar.simuframe.tenementbuilding.Building">
    </simuElement>
</timedSystemEvent>
<parameterValueSystemEvent
    description="The tenant is unsatisfied and so he decides to cancel the ↵
        tenancy agreement."
    name="TenantIsUnsatisfied">
    <parameterEventCondition
        class="edu.uni_weimar.simuframe.tenementbuilding.systemEvent.action. ↵
            MovesOutUnsatisfiedCondition">
    </parameterEventCondition>
    <eventAction
        class="edu.uni_weimar.simuframe.tenementbuilding.systemEvent.action. ↵
            TenantMovesOutAction">
    </eventAction>
    <parameterSource
        parameterId="edu.uni_weimar.simuframe.tenementbuilding.TenantSatisfaction">
    </parameterSource>

```

```
</parameterValueSystemEvent>
</extension>
```

5.7 simulationElement

Identifier:

edu.uni_weimar.simuframe.simulationElement

Since:

Version 0.4

Description:

This extension point is used for the declaration of simulation elements.

Configuration Markup:

```
<!ELEMENT extension (simulationElement+)>
<!-- extension
point CDATA #REQUIRED
id CDATA #IMPLIED
name CDATA #IMPLIED
-->
<!ELEMENT simulationElement EMPTY>
<!-- simulationElement
name CDATA #IMPLIED
description CDATA #IMPLIED
class CDATA #IMPLIED
-->
```

- **name** - Name of the simulation element.
- **description** - A short description of the purpose.
- **class** - Class which represents the simulation element.

Examples:

```
<extension
  point="edu.uni_weimar.simuframe.simuElement">
  <simulationElement
    class="edu.uni_weimar.simuframe.element.RootSimuElement"
    description="The root simulation element. A special simulation element, ↵
                just to have one root in a scenario."
    name="RootSimuElement">
  </simulationElement>
  <simulationElement
    class="edu.uni_weimar.simuframe.tenementbuilding.element.WallElement"
    description="A wall which is used as a boundary for rooms"
    name="WallElement">
  </simulationElement>
</extension>
```

5.8 listenerSupplier

Identifier:

edu.uni_weimar.simuframe.listenerSupplier

Since:

Version 0.3

Description:

This extension point allows the declaration of listeners. Listeners are used for decoupling purposes within the SimuFrame-Framework.

Configuration Markup:

```
<!ELEMENT extension (listenerSupplier+)>
<!--ATTLIST extension
point CDATA #REQUIRED
id CDATA #IMPLIED
name CDATA #IMPLIED
-->
<!ELEMENT listenerSupplier EMPTY>
<!--ATTLIST listenerSupplier
listenerSupplier CDATA #IMPLIED
publisherClass CDATA #IMPLIED
-->
```

- **listenerSupplier** - The listener supplier class which provides a listener.
- **publisherClass** - The publisher class

Examples:

```
<extension
  point="edu.uni_weimar.simuframe.listenerSupplier">
  <listenerSupplier
    listenerSupplier= "edu.uni_weimar.simuframe.tenementbuilding.↵
                      tenementcontract_heat. systemEvent.action.BookFuelPurchase"
    publisherClass="edu.uni_weimar.simuframe.tenementbuilding.↵
                  tenementcontract_heat.systemEvent.↵
                  action.HeatSystemFactory">
  </listenerSupplier>
</extension>
```

5.9 mission

Identifier:

edu.uni_weimar.simuframe.missionmanager.mission

Since:

Version 0.6

Description:

This extension point is used for the declaration of missions.

Configuration Markup:

```
<!ELEMENT extension (mission*)>
<!--ATTLIST extension
point CDATA #REQUIRED
id CDATA #IMPLIED
name CDATA #IMPLIED
-->
<!ELEMENT mission (requirement* , yield*)>
<!--ATTLIST mission
name CDATA #REQUIRED
id CDATA #REQUIRED
```

description CDATA #REQUIRED

>

- **name** – The name of the mission
- **id** – The mission's identifier
- **description** – A describing text: Targets and conditions of the mission.

<!ELEMENT requirement EMPTY>

<!ATTLIST requirement

id CDATA #REQUIRED

level CDATA #REQUIRED

>

- **id** – the id of the required skill.
- **level** – the needed level of the required skill.

<!ELEMENT yield EMPTY>

<!ATTLIST yield

id CDATA #REQUIRED

level CDATA #REQUIRED

>

- **id** – The id of the rewarded skill.
- **level** – The number of levels of the rewarded skill.

Examples:

```
<extension point="edu.uni_weimar.simuframe.missionmanager.mission">
  <mission
    name="MA"
    id="M1"
    description="Testmission">
    <yield
      id="A"
      level="70"></yield>
    </mission>
</extension>
```

5.10 badge

Identifier:

edu.uni_weimar.simuframe.skillmanager.badge

Since:

Version 0.6

Description:

This extension point is used for the declaration of badges. Badges are awarded for achieving certain skills.

Configuration Markup:

<!ELEMENT extension (badge*)>

<!ATTLIST extension

point CDATA #REQUIRED

id CDATA #IMPLIED

name CDATA #IMPLIED

>

<!ELEMENT badge (requirement*)>

```

<!ATTLIST badge
name CDATA #REQUIRED
description CDATA #REQUIRED
id CDATA #REQUIRED
>

```

- **name** – The name of the badge
- **description** – The meaning of this badge: Why is it awarded?
- **id** – An identifier of the badge.

```

<!ELEMENT requirement EMPTY>
<!ATTLIST requirement
id CDATA #REQUIRED
level CDATA #REQUIRED
>

```

- **id** – the Id of a required skill.
- **level** – The level of a required skill.

Examples:

```

<extension
  point="edu.uni_weimar.simuframe.skillmanager.badge">
  <badge
    description="Testbadge"
    id="edu.uni_weimar.simuframe.acoustics.badge"
    name="acoustic expertise">
    <requirement
      id="A"
      level="30"></requirement>
    <requirement
      id="B"
      level="30"></requirement>
    <requirement
      id="C"
      level="30"></requirement>
    <requirement
      id="D"
      level="30"></requirement>
    <requirement
      id="E"
      level="30"></requirement>
  </badge>
</extension>

```

5.11 skill

Identifier:

edu.uni_weimar.simuframe.skillmanager.skill

Since:**Description:**

This extension point is used for the declaration of skills.

Configuration Markup:

```

<!ELEMENT extension (skill*)>
<!ATTLIST extension
point CDATA #REQUIRED
id CDATA #IMPLIED

```

```
name CDATA #IMPLIED
>
<!ELEMENT skill (requirement*)>
<!-- skill
name CDATA #REQUIRED
id CDATA #REQUIRED
description CDATA #IMPLIED
>
  • name - Name of the skill.
  • id -
  • description -
<!-- requirement EMPTY
<!-- requirement
id CDATA #REQUIRED
level CDATA #REQUIRED
>
  • id -
  • level -
```

Examples:

```
<extension
  point="edu.uni_weimar.simuframe.skillmanager.skill">
  <skill
    name="SkillA"
    id="A"
    description="Testskill">
  </skill>
```

6 Quellcode und Anwendungen

Die Anwendung BuildVille bzw. Easy Gold steht zur Verfügung unter <http://apps.facebook.com/buildville/>. Um die Anwendung starten zu können, ist ein Facebook-Konto notwendig sowie die Installation des Unity-Webplayers. Der entsprechende Download-Link wird angezeigt.

Unter der URL

<https://www.dropbox.com/sh/kdmkec4uyhs24xc/FspC4Cr7lO/DissertationSoebkeDeliverables> stehen die im Rahmen dieser Arbeit erzeugten Artefakte zur Verfügung:

- Java-Code des Simulationskerns
- Unity3D-Projekt des BuildVille-Clients
- SmartFoxServer-Konfiguration
- XAMPP-Konfiguration
- EMMA Test Coverage Report

7 Abbildungsverzeichnis

Abbildung 1: Klassendiagramm des Publisher-Subscriber Patterns.....	8
Abbildung 2: Hierarchie von Simulationselementen mit angehängten Parametern	10
Abbildung 3: Lage der Wohnungen zueinander	17
Abbildung 4: Szenario-Editor: Kontext-Menü	23
Abbildung 5: Dialog zum Austausch der Fenster in einem Szenario	24
Abbildung 6: Mieter in Simulationsmodell des Szenario-Editors	28
Abbildung 7: Grundsätzliches UML-Klassendiagramm: Parameter	30
Abbildung 8: Vererbungshierarchie IInitialValueProvider.....	30
Abbildung 9: Vererbungshierarchie IDistribution	31
Abbildung 10: Modulabhängigkeiten für ein Szenario-Modul	35
Abbildung 11: Abhängigkeiten des Moduls tenementbuilding1.....	37
Abbildung 12: Parameter "HeatLevel": Verstärkende Schwingung	38
Abbildung 13: Parameter „HeatLevel“: Schrittweite um 1/3 verringert.....	39
Abbildung 14: Parameter "HeatLevel": Schrittweite auf 1/3 verringert	39
Abbildung 15: Parameter "HeatLevel": Schrittweite auf 1/8 verringert	39
Abbildung 16: Parameter "HeatLevel": Schrittweite auf 1/8 verringert, Heizleistung 2000 W	40
Abbildung 17: Berechnungsfehler bei unterschiedlicher Schrittweite	42
Abbildung 18: Default-Perspektive von SimuFrame	46
Abbildung 19: View Scenario Outline.....	47
Abbildung 20: Dialog Fensteraustausch	48
Abbildung 21: Dialog: Auswahl einer Wohnung für einen neuen Mieter	48
Abbildung 22: Benutzeraktion: Anbieten einer Wohnung.....	49
Abbildung 23: Änderung des Wertes eines Parameters	49
Abbildung 24: View Navigator.....	50
Abbildung 25: View: Parameter Types	51
Abbildung 26: View System Events	52
Abbildung 27: View Parameter Monitor	52
Abbildung 28: View Scenario Outline - Kontextmenü "Show History"	53
Abbildung 29: Beispiel Parameter Historie	53
Abbildung 30: View Parameter Comparison	54
Abbildung 31: Dialog Parameter-Auswahl	54
Abbildung 32: View Action Log.....	55
Abbildung 33: View Skill Tree Admin.....	56
Abbildung 34: View Skill Tree	57
Abbildung 35: View Mission Administration	58
Abbildung 36: View Mission	59

8 Tabellenverzeichnis

Tabelle 1: Manager-Klassen	9
Tabelle 2: In SimuFrame verwendete Entwurfsmuster	12
Tabelle 3: Formales Beschreibungsschema einer Designentscheidung	13
Tabelle 4: Parameter im Modul TenementContract_Heat	26
Tabelle 5: Simulationselement Tenant: Parameter	28
Tabelle 6: Symbolverzeichnis: Beispiel zur Fehlerabschätzung	41
Tabelle 7: Attribute der View Scenario Outline	46
Tabelle 8: Attribute der Navigator-View	49
Tabelle 9: Attribute der View Parameter Types	50
Tabelle 10: Attribute der View System Events	51
Tabelle 11: Attribute der View Parameter Monitor	52
Tabelle 12: Attribute der View Parameter Comparison	53
Tabelle 13: Attribute der View Action Log	55
Tabelle 14: Attribute der View Skill Tree Administration	56
Tabelle 15: Attribute der View Skill Tree	57
Tabelle 16: Attribute der View Mission Administration	58
Tabelle 17: Attribute der View Mission	58
Tabelle 18: Liste der Module (Plug-ins)	60