technische universität
dortmund

Abschlussbericht

Planspiele für Krisenmanagement in Rechenzentren

Alexander Schäferdiek, Benjamin Tokgöz
Fabian Kotschenreuter, Hang Yu
Jan Möller, Konstantin Tkachuk
Patrik Elfert, Ruikun Chang
Volkan Gümüs

30. Juni 2016

Betreuer:
Michael Neubauer
Johannes Neubauer
Stefan Naujokat
Michael Lybecait

Fakultät für Informatik
Programmiersysteme (LS5)
Technische Universität Dortmund
http://ls5-www.cs.tu-dortmund.de

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Aufbau der Arbeit 3

1.2 Ziele und Kooperationspartner 3

2 Das Rechenzentrum **5**

2.1 Dienste und Aufgaben 5

2.2 Die Büchse der Pandora 6

3 Das Planspiel **7**

3.1 Definition und Ausprägungen 7

3.2 Digitales Lernen 9

3.3 Motivation 10

3.4 Kritik 12

3.4.1 Wirtschaftlicher Faktor 12

3.4.2 Pädagogischer Einfluss 13

3.5 Planspiele und klassische Spiele 13

3.6 Ziele 13

4 Krisenmanagement **15**

4.1 Planspiele zur Bewältigung von Krisen in Rechenzentren 16

5 Grundlagen des Prototype-Driven Development 18

5.1 Modellierungsphase . . . 19

5.1.1 Cinco SCCE Meta Tooling Suite . . . 20

5.2 Implementierungsphase . . . 21

5.2.1 DyWA . . . 21

5.3 Testphase . . . 22

6 Planspieldefinition: Krisenplan 23

6.1 Kernelemente und Ziele des Spiels . . . 23

6.1.1 Spielmechanik . . . 24

6.1.2 Spiel-Metriken . . . 25

6.1.3 Metrik-Kategorien . . . 25

6.2 Mini-Rätsel . . . 28

6.3 Spielrollen . . . 29

6.3.1 Kunden . . . 29

6.3.2 Ankaufspersonal . . . 29

6.3.3 Technikpersonal . . . 29

6.3.4 Finanzpersonal . . . 30

6.4 Spielleitung . . . 30

6.5 Schwierigkeitsgrade . . . 31

6.6 Vermittlung von Lerninhalten . . . 32

7 Modellierungsphase 34

7.1 Einführung . . . 34

7.2 Grundlagen des Planspielmodells . . . 34

7.3 Komponentenmodellierung . . . 35

7.3.1 Ticks . . . 35

7.3.2 Balance . . . 36
7.3.3 Job . . . 36
7.3.4 Job Market . . . 36
7.3.5 Service Center . . . 37
7.3.6 Server . . . 37
7.4 Features . . . 37
7.4.1 Kunden . . . 37
7.4.2 Offer Market . . . 39
7.4.3 Finanzübersicht . . . 39
7.4.4 Abhängigkeiten - Stromerzeugung und Kühlung . . . 39
7.4.5 Markt und Status-Effekte . . . 41
7.4.6 Rollen und Kommunikation . . . 41
7.5 Zusammenfassung . . . 42
7.6 Rollenmodellierung . . . 43
7.6.1 Beratung . . . 44
7.6.2 Technik . . . 44
7.6.3 Finanz . . . 45
7.6.4 Grafische Oberfläche . . . 45
7.6.5 Kunde . . . 46
7.6.6 Beratung . . . 46
7.6.7 Finanzverwalter . . . 48
7.7 Leveleditor . . . 49
7.7.1 Anforderungen . . . 50
7.7.2 Aufbau und Struktur . . . 50
7.7.3 Funktionsweise . . . 52
7.7.4 Verwendete Techniken . . . 53

7.7.5 Übertragung in Web-Applikation 54

8 Implementierungsphase **55**

8.1 Arbeiten im Team 55

8.2 Erstellen der Datenstruktur 56

8.2.1 Pyro 56

8.2.2 Krisenplan Connector 56

8.3 Prozessmodellierung 56

8.3.1 Prozessüberblick 57

8.3.2 Spielwelt initialisieren 57

8.3.3 Server kaufen 58

8.3.4 Stromgenerator reparieren 59

8.3.5 DoTick-Prozesse 59

8.4 Erstellen der Oberflächen 64

8.4.1 Gestaltung der Spielkomponenten mit DyWA-GUI 64

8.4.2 Implementierte Oberflächen 65

8.5 Erweiterung der DyWA 70

8.5.1 Neue Komponenten 70

8.5.2 Styling mittels CSS 74

8.6 Umgesetzte Ziele 75

9 Auswertung einer Spieleinheit **78**

9.1 Spielablauf 78

9.1.1 Vorbereitungen auf die erste Simulations-Runde 78

9.1.2 Ablauf der weiteren Runden 79

9.1.3 Rollenansichten 79

9.1.4 Zusammenfassung 79

9.2 Kommunikation 80
9.2.1 Stress durch Kommunikation 80
9.2.2 Gewinne für das Team 81
9.3 Mini-Rätsel 81
9.4 Auswertung 82

10 Ausblick **83**
10.1 Zusammenfassung 83
10.2 Prototype-Driven Development 84
10.3 Planspiel der Projektgruppe 84
10.4 Toolstack des Lehrstuhl 5 85
10.5 Abschluss 85

Anhang **86**

Abbildungsverzeichnis **91**

Literaturverzeichnis **93**

Kapitel 1

Einleitung

Sprachen werden durch das Memorieren von Vokabeln erlernt. Dabei können diese abwechselnd verdeckt und sichtbar betrachtet werden, bis diese in das Gedächtnis übergehen. Mathematische Formeln können ebenfalls auswendig gelernt und ihre Anwendung in konkreten Beispielen trainiert werden. Soll das Fahrradfahren erlernt werden, kann sich auf ein Fahrrad gesetzt und ausprobiert werden. Ein Sturz vom Rad wäre ein mögliches Risiko, welches jedoch durch den anfänglichen Einsatz von Stützrädern reduziert werden kann. Der Fahrer kann nun aus der vorherigen Fahrt mögliche (Gefahren-)Situationen erkennen und für weitere Fahrten daraus lernen. Beispielsweise, dass bei nasser Fahrbahn die Vorderradbremse äußerst vorsichtig zu betätigen ist. Bei der nächsten Regenfahrt könnte er aus seiner Erfahrung profitieren und bewusst bremsen.

Doch wie kann ein richtiges Verhalten bei Ausnahmesituationen in einem komplexen Umfeld trainiert werden? Beispielsweise könnte in einem Atomkraftwerk die Kühlung ausfallen. Ein Kraftwerksarbeiter, der diesen Fehler bereits miterlebt hat, weiß, wie zu handeln ist.

Aber wie sieht es mit einem Arbeiter aus, der einen Ausfall der Kühlung bisher nicht erlebt hat? Wie kann sichergestellt werden, dass der Arbeiter angemessen vorgeht, um das Problem zu beheben? Offensichtlich kann nicht für jeden neu einzustellenden Mitarbeiter die Kühlung sabotiert werden. Einleuchtend ist auch, dass die entsprechenden Arbeiter auf eine derartige Situation vor einem realen Einsatz vorbereitet werden müssen.

Diese Vorbereitung kann auf verschiedene Art und Weisen erfolgen. Der Lernende könnte versuchen, alle Informationen über sein Tätigkeitsfeld zu verinnerlichen. Dabei müsste er nahezu alle Details abdecken, die Erfahrungen anderer mit einbeziehen und mögliche Szenarien versuchen herauszustellen. Obwohl der Lernende fundiertes Wissen erlangt haben könnte, wäre der Einsatz in einem realen Umfeld kritisch aufgrund der Gefahr eines Praxis-

schocks. Ungeachtet dessen, dass der Arbeiter das nötige Wissen hat, um die Kühlung des Kraftwerks zu reparieren, misslingt der Versuch. Er kennt die Auswirkungen des Versagens der Kühlanlage. Unter Umständen sind eine Vielzahl von Menschenleben, darunter sein eigenes, gefährdet. Dieser Umstand erzeugt bei dem Arbeiter eine ihm bisher unbekannte Stresssituation. Dabei kann er sein eigenes Wissen und seine eigenen Handlungen in Frage stellen, obwohl sie korrekt sind. Ferner kann er Informationen bzw. Teilvorgänge vergessen oder in einen Angstzustand verfallen. Aufgrund dieser Problematik ist das Lernen in einer dem ursprünglichen Geschehen nachempfundenen Situation vorzuziehen.

Das Szenario kann real nachgebildet werden. Folglich könnte in einem stillgelegten Kraftwerk eine Kühlanlage vom Ausbildungspersonal manipuliert werden, sodass der Arbeiter dieses Problem in einem realen Umfeld lösen muss. Nachteile dieser Variante sind zum einen die hohen Kosten, die für den Lernversuch eines Einzelnen entstehen. So muss unter Umständen ein ganzes Gebäude und eine hohe Anzahl von Maschinen funktionsfähig gehalten werden. Zum anderen lassen sich einige Szenarien aufgrund ihrer Folgen nicht nachbilden, wie beispielsweise das Szenario einer Kernschmelze, die unvorhersehbare und bzw. oder irreversible Folgen nach sich ziehen würde.

Eine Alternative zu dieser Lernmethodik bietet das Lernen durch Simulationen, beispielsweise in Form eines Planspiels. Dabei wird die Wirklichkeit als fiktive Welt, beispielsweise als Computeranwendung, nachgebildet. Es entsteht ein virtuelles Kraftwerk, welches die Funktionen des realen nachbildet. Der Arbeiter kann anhand des Planspiels üben ohne das reale System zu gefährden.

Die Herausforderung und zugleich das Abgrenzungsmerkmal eines reinen Simulationsspieles ist einen (möglichst großen) Lerneffekt zu erzielen. Ein interessanter Bestandteil eines Planspiels ist die Spielleitung. Diese ist in der Regel kein direkter Teil des Spiels, kann jedoch dynamisch das Spielgeschehen beeinflussen, indem sie Szenarien herbeiführt. Ein möglicher Eingriff im fiktiven Kraftwerk wäre die Auslösung einer Krise, beispielsweise des Ausfalls der Kühlung, welche die Spieler zu meistern haben.

Im Rahmen der Projektgruppe soll ein solches Planspiel im Kontext *Rechenzentrum* und *Krisensimulation im Rechenzentrum* entwickelt werden. Dabei existieren eine Vielzahl von Rollen, wie die des Technikers oder die der Auftragsbeschaffung, die im Rahmen des Spiels von den Spielern ausgeübt werden können. Die Probleme, welche im Laufe des Planspiels von den Spielern gelöst werden müssen, sind mannigfaltig: Größere Störungen wie der Ausfall des (Ersatz-)Strom- und Kühlungsnetzes, höhere Gewalt oder kleinere Störungen wie defekte Festplatten, Kabelbrüche, falsche Verkabelung oder unsachgemäße Modifikation durch Mitarbeiter. Leicht lässt sich feststellen, dass in diesem Kontext, genau wie in dem des Kraftwerks, es unmöglich ist jeden Mitarbeiter erfolgreich über jede Störungsart zu

belehren. Grund dafür ist, dass eine Störung durchaus uneindeutig respektive unkenntlich sein und eine komplexe Analyse zur Identifizierung benötigen kann.

Eine technische Störung wird durch eine Reihe von sozialen Aspekten zu einem Problem, welches über die eigentliche Ursache weit hinausgeht: Der ausgelöste Stress, Kommunikationsprobleme sowie fehlende Fach- oder Sozialkompetenz eskalieren die Situation. Primäres Ziel des Planspiels ist es deshalb, den Spielern eine (allgemeine) Vorgehensweise zur Krisen- und Störungsbewältigung, die Wichtigkeit von (ordentlicher) Kommunikation sowie zusätzliche fachliche Informationen zu vermitteln.

1.1 Aufbau der Arbeit

Diese Arbeit behandelt die Erarbeitung, Definition und Umsetzung einer Krisensimulation in Form eines Planspiels. Dabei werden zu Beginn der Arbeit die nötigen Hintergründe zum Themengebiet Planspiel sowie Vermittlung von Lerninhalten erläutert. Anschließend wird unter Betrachtung verschiedener Spielarten eine geeignete Definition zur Krisensimulation gebildet. Anhand dieser wird ein Anforderungskatalog erstellt, welchen die spätere Spielmechanik zu erfüllen hat. Nachdem die theoretischen Grundlagen erörtert wurden, wird eine Einführung in die methodische Vorgehensweise - das *Prototype Driven Development (PDD)* - gegeben. Dabei wird die relevante Software betrachtet und die Entscheidung für das Prototype Driven Development begründet. Anschließend werden Ansätze zur Realisierung der Anforderungen aus dem vorher genannten Anforderungskatalog beschrieben. Konkret wird die Modellierungsphase mit *Cinco* beschrieben, in der die einzelnen Spielkomponenten, wie Spielerrollen oder Schwierigkeitsstufen, in einer einfachen grafischen Darstellung realisiert wurden. Im abschließenden Kapitel wird der aktuelle Entwicklungsstand kritisch dem Anforderungskatalog gegenüber gestellt und evaluiert.

1.2 Ziele und Kooperationspartner

Die zugrundeliegende Thematik wird im Rahmen einer Projektgruppe des Lehrstuhl 5 bearbeitet, wobei der Schwerpunkt in der Erarbeitung der nachfolgenden Sachgebiete liegt:

- Konzepte des Prototype Driven Development im Kontext einer realen Anwendung bewerten sowie die entsprechenden Werkzeuge des Lehrstuhl 5 testen und erweitern,
- Erarbeitung von Schulungskonzepten zur Krisenbewältigung,

- Erstellen eines erweiterbaren Planspiels, das im Krisenmanagement eines Rechenzentrums eingesetzt werden kann, unter Benutzung der Prototype Driven Development Werkzeuge des Lehrstuhl 5 und der Projektgruppe 586: ProBio.

Die Projektgruppe arbeitet in Partnerschaft mit der Geschäftsführung von Citkomm, einem Rechenzentrumbetreiber für den Dienstleistungsmarkt einiger Kommunen und kommunaler Unternehmen. Die Kooperation mit Citkomm liefert der Projektgruppe wesentliche Erkenntnisse über verschiedene Aufgaben und den allgemeinen Betrieb eines Rechenzentrum:

- Aufbau und Organisation des Rechenzentrums
- Arbeitsabläufe der Mitarbeiter
- Aktuelle sowie vergangene Problemszenarien und Krisen

Insbesondere liefert die Kooperation mit Citkomm der Projektgruppe einen realitätsnahen Einblick in die vorhandene Problemstellung, Mitarbeiter des Rechenzentrums auf Ausnahmefälle, wie Krisen, schulen zu können. Deshalb geht es im Rahmen der Projektgruppe insbesondere darum, eine effiziente Schulungsmaßnahme zu erarbeiten. Die erarbeiteten Erkenntnisse können im Rahmen von experimentellen Schulungen an Citkomm-Mitarbeitern evaluiert werden.

Kapitel 2

Das Rechenzentrum

Ein Rechenzentrum ist ein Komplex, bestehend aus einer infrastrukturellen Anbindung eines verwaltenden und betreibenden Betriebs sowie einer (größeren) Räumlichkeit, in dem sich ein oder mehrere Computer befinden, die für interne oder externe Dienste genutzt werden. Es ist in der Regel eine Halle oder zumindest ein größerer Raum, der eine Reihe von Computern, Infrastruktur zum Zusammenschluss dieser Computer sowie zur Anbindung an die Außenwelt und eine zur Kühlung der Anlage genutzte Klimatisierungsanlage enthält. Die Computer sind in Form von *Racks* aufgebaut, wobei ein *Rack* verschiedene Recheneinheiten, Kühlungs- und Sicherungskomponenten sowie eine Anbindung zu einer Notfallstromquelle beinhaltet. Die Computer, üblicherweise als *Server* bezeichnet, werden in der Regel über das Internet in virtuellen oder dedizierten Teilen vertrieben. Nutzer können diese Teile mieten und ihre Software oder Dienste auf diesen ausführen. Es gibt jedoch auch Rechenzentren, die ausschließlich für Behörden und ihre Anliegen errichtet werden.

2.1 Dienste und Aufgaben

Das Angebot eines klassischen Rechenzentrums richtet sich üblicherweise an Großkunden. Diese Großkunden mieten eine Menge von Servern, welche im Rechenzentrum des Betreibers stehen. Folglich verwaltet der Großkunde lediglich die Software, die auf dem Computer verwendet wird. Der Betreiber ist für die lokalen Gegebenheiten wie Kühlung und Wartung der Hardware zuständig. Änderungen, die die Hardware betreffen, müssen beim Betreiber angefordert werden.

Der Großkunde vermietet die Server in der Regel als dedizierte Maschine, virtuelles Gastsystem oder in Form von Diensten weiter. Das Mieten von dedizierten Maschinen bietet

den Vorteil, dass der Endkunde alleiniger Nutzer auf der Maschine ist. Im Gegensatz dazu werden virtuelle Gäste parallel auf einem Wirtsystem betrieben, wobei sich verschiedene Virtualisierungen die Ressourcen des Wirts teilen. Je nach Nutzerzahl und laufenden Anwendungen kann die Ressourcenverteilung für Virtualisierungen ungerecht bis hin zum Ausfall verlaufen.

Die vom Großkunden angebotenen Dienste lassen sich grundsätzlich in zwei Kategorien einordnen. Einerseits wird Speicherplatz vermietet, der vom Endkunden beispielsweise für eine Homepage genutzt werden kann. Andererseits zahlt der Kunde dafür, dass eine von ihm gewünschte Anwendung auf dem System des Großkunden für einen bestimmten Zeitraum ausgeführt wird. Eine solche Anwendung kann beispielsweise ein Spiele- oder Kommunikationsserver sein. Es gibt jedoch auch von Großkunden angebotene Dienstleistungen, die sich nur schwer einordnen lassen. Dazu zählt beispielsweise das Anbieten von Domänen. [1]

2.2 Die Büchse der Pandora

Der Vergleich, ein Rechenzentrum als *Büchse der Pandora* zu bezeichnen, mag zwar etwas weit hergeholt sein, hat jedoch einen ernsthaften Kern. Rechenzentren können durchaus kritische Infrastruktur beeinflussen. Durch unseren Kooperationspartner wissen wir, dass viele öffentliche Dienste, stromerzeugende Kraftwerke, für Frischwasser sorgende Wasserwerke, Krankenhäuser und viele weitere Einrichtungen an Rechenzentren angebunden und von diesen abhängig sind. In nachfolgenden Kapiteln (vgl. 4.1) werden wir erfahren, dass ein Rechenzentrum höchst anfällig für Störungen aus menschlichen und technischen Quellen ist. Die Ausführung eines Löschbefehls auf einem falschen Rechner oder das Entfernen eines Kabels können bereits zur Unerreichbarkeit führen, angeschlossene Dienste damit abklemmen, was man durchaus als *das Öffnen der Büchse* sehen könnte. Die Auswirkungen eines Ausfalls sind in der Regel kaum abschätzbar und mit einer Vielzahl von Konsequenzen verbunden.

[1] Jeder Rechner, der im Netzwerk teilnimmt, hat eine grundsätzlich eine eindeutige Adresse, die sogenannte IP-Adresse. Weil sich aber Namen besser einprägen als IP-Adressen werden so genannte Domain-Namen verwendet. Über diese Namen können die Rechner ebenfalls angesprochen werden.

Kapitel 3

Das Planspiel

Planspiele werden als handlungsorientierte Methodik zur Vermittlung von Lehr- und Lerninhalten komplexer Zusammenhänge genutzt. Hintergrund bildet ein Szenario, das einem realen Geschehen nachempfunden ist. Die Teilnehmer des Planspiels übernehmen einen Akteur innerhalb des Planspiels, ähnlich einem Rollenspiel, und durchlaufen eine Reihe vorgegebener Entscheidungsprozesse. Dabei müssen sie die Interessen ihrer Rolle möglichst realitätsnah vertreten.

Genauso wie in einem Rollenspiel behandeln Planspiele mehrere Akteure, die sowohl einzeln als auch in Gruppen ein Rollenprofil, d.h. eine Charakterisierung und Handlungsabläufe, zugewiesen bekommen können. Das Planspiel beinhaltet für jedes Einzel- und Gruppenprofil Regeln, die von den Akteuren zu beachten sind. Ein weiteres wichtiges Merkmal ist die Rolle der Spielleitung. Diese bringt die Dynamik in das Spiel, da sie zu jeder Zeit innerhalb des Spiels Aktionen zur Beeinflussung der Spielumgebung durchführen kann. Die Akteure sind aufgefordert, auf die unerwarteten Änderungen entsprechend zu reagieren.

3.1 Definition und Ausprägungen

Planspiele, sogenannte *Serious Games*, haben nicht das Primärziel der Unterhaltung der Spieler, eher ist der Lerneffekt von Bedeutung:

> „A serious game is a game in which edutainment (in its various forms) is the primary goal, rather than entertainment." [18]

Ein Planspiel lässt sich in drei grobe Komponenten (vgl. 3.1) separieren: Die erste Komponente ist das Regelspiel. Es formuliert die Gesetzmäßigkeiten, die in der Spielwelt gelten. Das Rollenspiel ist eine weitere Komponente, welches die Interaktion der Spieler und die Simulation der Umwelt beschreibt. Die letzte Komponente bildet die simulierte Umwelt. Diese stellt die technische Spezifikation und Abbildung des Spielszenarios (vgl. Rechenzentrum) dar. Nachfolgend werden diese Komponenten genauer beschrieben. Die Schlüsselelemente von Planspielen können in folgende Gruppen separiert werden:

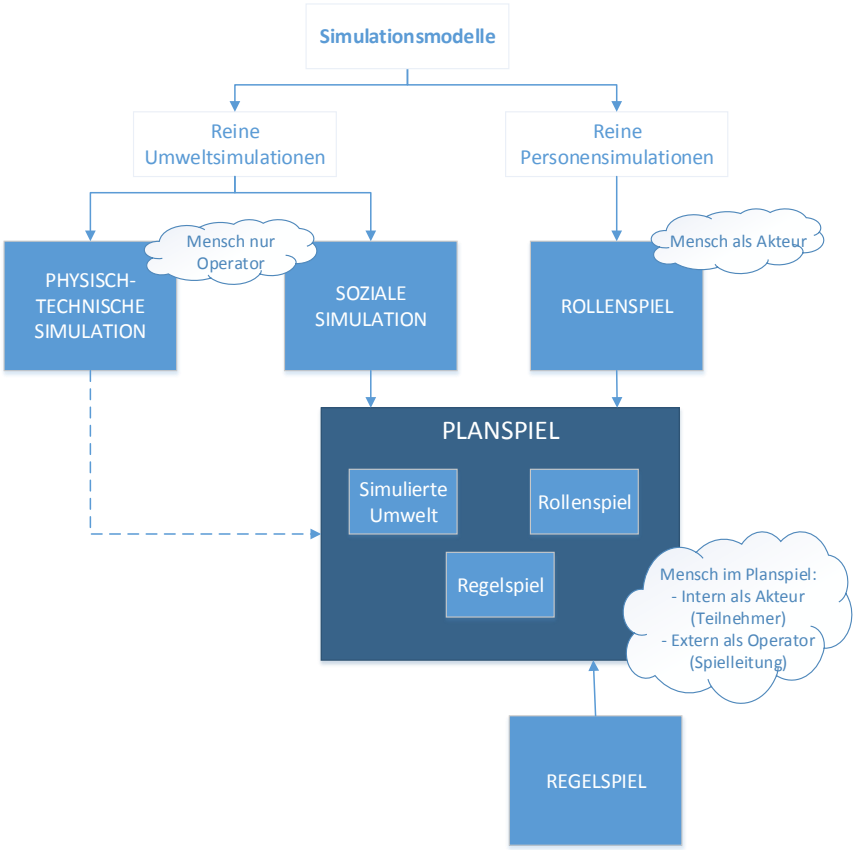

Abbildung 3.1: Das Planspiel als komplexe Mischform (angelehnt an [12])

Regelspiel

Die Kernkomponente - das Spiel - bildet den dynamischen Teil des Modells und lässt sich durch folgende Punkte charakterisieren:

Das Spiel

- ist auf eindeutigen Regeln aufgebaut, die auch das Verhalten der Spieler klassifizieren,

- entwickelt sich nachvollziehbar, beispielsweise durch handelnde Spieler oder eine fortschreitende Umwelt,
- bietet Möglichkeiten zur Bewertung und des Vergleichs der Spielerleistungen,
- kann auf eine Plattform abstrahiert und folglich durchgeführt werden, beispielsweise als Computerspiel.

Rollenspiel

Der Spieler bekommt, wie im klassischen Rollenspiel, eine Rolle zugewiesen und muss sich in einer ihm mehr oder weniger bekannten Situation zurecht finden. Ferner muss er sich für einen möglichen Lerneffekt mit seiner Rolle identifizieren können, sodass er die vom Spiel gegebene Aufgabe bearbeiten kann.

Simulierte Umwelt

Die simulierte Umwelt stellt ein System dar, in dem der Spieler sich befindet und agiert. Dieses ist einem realen System nachempfunden und soll es in seinen markanten Aspekten nachahmen, sodass die Immersion des Spielers verstärkt wird. Insbesondere deshalb ist ein Planspiel auch eine Simulation.

Durch Betrachtung der vorher genannten Komponenten lässt sich feststellen, dass das Planspiel grundsätzlich eine Mischform der ursprünglichen Spielformen ist. Dies visualisiert Geuting auch sehr stark in seinem Modell (Abb. 3.1), wobei die ursprünglichen Spielformen als *reine Simulation, reines Regelspiel* und *reines Rollenspiel* gekennzeichnet sind. Die Dreiteilung des Planspiels ist häufig vorzufinden, auch in [3] wird die o.g. Aufteilung als charakteristisch für ein Planspiel angesehen.

3.2 Digitales Lernen

Klassische Lehrmethoden finden sich im Schulunterricht. Dabei gibt zum Beispiel eine Lehreinheit, die die Lehrinhalte im Frontalunterricht vorträgt. Klassische Lernmethoden sind unter anderem das Auswendiglernen der entsprechenden zu lernenden Inhalte. Es gibt auch Bereiche, in denen vor einer theoretischen Ausbildung bzw. vor Schaffung einer theoretischen Grundlage eine praktische Probierphase vorgezogen wird. Dies ist häufig in beruflichen Ausbildungen der Fall, bei denen die Auszubildenden eine mehrwöchigen praktischen Einsatz in einem Betrieb durchlaufen, bevor sie die theoretischen Grundlagen an

einer Fachhochschule lernen. Obwohl sich diese Lehr- und Lernmethoden über Jahrhunderte hinweg bewährt haben, sind sie in vielen Bereichen ungeeignet. Das Problem ist nicht, dass die Menschen sich dem Lernen verweigern, denn diese sind grundsätzlich lernbereit. Die Schwierigkeit dahinter steckt darin, auf neue, möglicherweise noch nicht ausgiebig erforschte Lehr- und Lernmethoden umzusteigen sowie und insbesondere dadurch auf seltene Ereignisse wie Krisen vorzubereiten. Höchstwahrscheinlich wird kein Auszubildender in seiner (im Durchschnitt) 3 jährigen Ausbildungsphase eine Krise erleben und insbesondere deshalb keine Erfahrung im Umgang mit Krisenbewältigen erlangen. Wahrscheinlich ist jedoch, dass er in seinen nächsten (beispielsweise) 30 Arbeitsjahren mit einer Krise in Berührung kommt - und mit dieser sollte er angemessen umgehen können.

Zu Beginn dieser Ausarbeitung wurde herausgestellt, dass die wesentliche Herausforderung eines Planspiels ist, die Fähigkeit zu vermitteln, in einem komplexen Umfeld korrekte und rational nachvollziehbare Entscheidungen zu treffen. Hierbei besteht die Komplexität einerseits aus der Anzahl der Komponenten eines Sachverhalts. Andererseits kommt es ebenfalls auf die Vielfalt der Verknüpfungen zwischen den einzelnen Komponenten an. Diese Fähigkeit wird als Entscheidungskompetenz bezeichnet.

Durch einen Vergleich der klassischen Lehrmethoden mit der Entscheidungskompetenz lässt sich feststellen, dass diese Fähigkeit nicht durch die altbewährten Lehrmethoden erlangt bzw. ausgebaut werden kann. Statisches Wissen, wie es die klassischen Lehrmethoden vermitteln, ist ungeeignet um komplexe Situationen bewältigen zu können.

Im Zuge der Digitalisierung des Alltags ist es ein naheliegender Gedanke, auch auf digitale Lehr- und Lernmethoden umzusteigen. Ein klassisches Beispiel für solche Methoden sind *Massive Open Online Courses*, bei denen eine Vielzahl von Teilnehmern über das Internet an einem Kurs auf Universitätsniveau teilnehmen. Digitale Lehr- und Lernmethoden sind insbesondere deshalb immer beliebter, weil sie die vorher genannten Nachteile bzw. Probleme des klassischen Lernens zu lösen scheinen.

3.3 Motivation

In der Einleitung wurde erwähnt, dass der Spielspaß ein wichtiges Merkmal eines Planspiels darstellt, um einen besseren Lerneffekt zu erzielen. Spielspaß wird durch eine angeregte Motivation beim Spieler erzeugt. Ist der Spieler motiviert, so möchte er neue Inhalte des Spiels kennen lernen oder intensiviert sein Bemühen ein bisher ungelöstes Problem zu meistern. Das vom Spiel gegebene Wissen eignet er sich freiwillig an, da er durch seine Motivation eigenwillig weiter spielt.

Motivationsforscher *Barbuto* und *Scholl* [16] forschten über die grundsätzliche Entstehung von Motivation, genauer gesagt, wieso sich eine Person für eine Sache motiviert, und haben die Forschungsinhalte zu fünf Klassen von Motivationsquellen zusammengefasst. Dabei erfolgt grundsätzlich die Unterscheidung zwischen *intrinsischer* und *extrinsischer* Motivation.

Intrinsische Motivation

Intrinsische Motivation steht für das Lernen und Arbeiten aus eigenem, innerem Antrieb. Die Handlung, die aus dieser Motivation entsteht, dient der persönlichen Befriedigung. Dies kann der einfache Spaß, das Befriedigen der eigenen Interessen oder die Herausforderung sein. Eine Handlung, die einer intrinsischen Motivation hervorgeht, ist unabhängig von einer Belohnung ausgelöst worden.

- Interne Prozessmotivation: Diese Art der Motivation ist eine von zweien intrinsischen Motivationsquellen. Eine Aufgabe wird um ihrer selbst willen erledigt. Man erkennt den Sinn der Tätigkeit, hat Spaß daran, ist weder unter- noch überfordert und führt sie aus, ohne eine Belohnung zu erwarten.
- Internes Selbstverständnis: Das Handeln ist von der eigenen inneren Vorstellung, der eigenen Werte und Art, einer eigenen Idealvorstellung gesteuert. Dies ist für Außenstehende nicht unbedingt nachvollziehbar, prägt, führt und motiviert den Handelnden jedoch so stark, dass diese Art der Motivation ein besonders hohes Leistungsmotiv hervorruft.

Extrinsische Motivation

Extrinsische Motivation besteht aus Lern- und Arbeitsanreizen, die mit positiven Folgen versehen sind oder negative Folgen vermeiden. Der Handlungsvollzug selbst stellt eine eher untergeordnete Rolle dar. Der Anreiz kann eine (fiktiv-) materielle Belohnung sein oder in Form von sozialer Anerkennung durch das persönliche Umfeld gegeben werden. Die Wirkung extrinsischer Anreize nimmt einen anderen Verlauf, als die eines intrinsischen Impulses, denn extrinsische Motivation nimmt mit der Zeit deutlich ab. Ihre Wirkung muss durch Erneuerung oder Veränderung der Anreize aufrechterhalten werden.

- Instrumentelle Motivation: Das Verhalten wird dominiert vom Wunsch nach einer konkreten Belohnung oder einem anderen Vorteil für das Durchführen einer Tätigkeit.

- Externes Selbstverständnis: Das Handeln ist von der sozialen Rolle, die eine Person in einer Gemeinschaft einnimmt, abhängig. Eine soziale Rolle ist häufig mit Aufgaben und Pflichten verbunden, welche von der betreffenden Person als externe Motivationsquelle wahrgenommen werden. Einerseits um die Erwartungen zu erfüllen, andererseits aufgrund des Zugehörigkeitsgefühls, motiviert sich die Person zur Tätigkeit.
- Internalisierung von Zielen: Die Person identifiziert sich mit dem System, für das es tätig ist, in dem es die Ziele des Systems verinnerlicht. Das Handeln der Person ist folglich danach ausgerichtet, diese Ziele bestmöglich umzusetzen.

Die Leistung einer intrinsisch motivierten Person kann nicht immer durch extrinsische Belohnungen gesteigert werden. Wenn ein Verhalten fast nur durch extrinsische Reize gesteuert wird, sinkt die innere Beteiligung, da das Gefühl der Selbstbestimmung verringert wird. Die Selbstmotivierungsfunktion, die die Freude aus der eigentlichen Tätigkeit nutzt, wird außer Kraft gesetzt, wobei die Existenz dieses Phänomens umstritten ist. [21]

3.4 Kritik

Planspiele vereinen viele Eigenschaften von Lehr- und Lernmethoden, die andere aktive Lehr- und Lernmethoden nur eingeschränkt vorweisen können. Trotzdem gibt es für Planspiele wesentliche Kritikpunkte, die vor der Nutzungsentscheidung zu beachten sind. Diese werden in den folgenden Abschnitten behandelt.

3.4.1 Wirtschaftlicher Faktor

Die Erstellung eines Planspiels, dessen Durchführung und Auswertung sind zum Teil mit erheblichen Kosten verbunden. Einerseits muss das Spiel programmiert werden. Nach der Komplexität des zugrundeliegenden Szenarios steigt der Programmieraufwand für das Spielmodell, denn dieses ist im Idealfall eine detailgetreue Nachbildung der Wirklichkeit. Zudem kann viel Aufwand in die *Userexperience* eines Spiels investiert werden, wobei dieser nur durch das Budget des Auftraggebers limitiert ist. Weitere Kosten entstehen für Personal aus dem Entwicklerbetrieb, dass die Auftraggeber in das Spielsystem einweist. Diese Kosten unterscheiden sich je nach Spielkomplexität. Zusätzlich wird für ein optimales Ergebnis eine Räumlichkeit benötigt, die die Hardware für das Spiel bereitstellt, ein angenehmes Spielen für die Spieler ermöglicht und zudem mit Konferenzräumen (wobei virtuelle Räume durchaus denkbar sind) für den Spieleraustausch ausgestattet ist.

Den Aufwand zur Spielerstellung, zur Spieldurchführung und für die Betreuung beschreibt *Bronner* als Faktoren, die von Hochschulen häufig als Begründung zur Ablehnung von Planspielen genannt werden [4].

3.4.2 Pädagogischer Einfluss

Im Spiel erhält der Spieler entscheidungsrelevante Informationen ohne große Mühen. Er kann diverse Vorgehensweisen ausprobieren und über moralische oder wirtschaftliche Grenzen hinweg agieren. Er könnte den Eindruck bekommen, dass diese einfache Informationsbeschaffung, die Grenzen und die aufzuwendenden Kosten in der Wirklichkeit ähnlich oder gleich sind. Diese Eindrücke mit in die reale Welt zu nehmen wäre kontraproduktiv.

Andererseits könnten die Spieler das Spiel auch wirklich als solches Einstufen und unüberlegt bzw. übermäßig risikofreudig spielen. Solch ein Spieler kann zwar gute Ergebnisse im Spiel erzielen, der Lerneffekt ist jedoch durch die Diskrepanz zur Realität wesentlich geringer.

3.5 Planspiele und klassische Spiele

Spielen ist eine Tätigkeitsform, die in der Regel aus Freude oder auch zu Entspannungszwecken ausgeübt wird. Gespielt werden kann individuell oder auch in einer Gruppe. Planspiele verfolgen jedoch nicht das primäre Ziel des Vergnügens. Sie sollen spielerisch Inhalte vermitteln und den Spieler einer Stresssituation aussetzen (entgegen der Entspannung in klassischen Spielen), in der er rational handeln muss.

Klassische Spiele bieten genauso wie Planspiele eine Umwelt, in der der Spieler agieren muss. Jedoch ist diese oft eine fiktive, realitätsferne Welt, in welcher der Spieler in seiner Rolle durch Beachtung oder Missachtung der Spielregeln mehr oder weniger zum Spielfortschritt beitragen kann. Am Ende des Spiels spürt der Spieler den sozialen Erfolg des Gewinns und wird mit Freude belohnt. Im Planspiel kann der Spieler bei einem Sieg unter strikter Beachtung der Spielregeln und Einhaltung seiner Rolle ebenfalls diese Freude des Gewinns spüren, ferner jedoch einen Lerneffekt, den das Spiel zu vermitteln versucht, erzielen. Im Idealfall nimmt das Spiel dem Spieler den Praxisschock in einer zukünftigen realen Situation.

3.6 Ziele

Zusammengefasst sollen durch Planspiele folgende Ziele erreicht werden:

- Gestaltung einer realitätsgetreuen Lernumgebung mit minimalen Kosten und ohne Gefährdung der realen Umgebung
- Reflektion der eigenen Vorgehensweise
- Zugewinn an Faktenwissen
- Prävention des Praxisschocks
- Lerneffekt zur Behandlung von Stresssituationen
- Erproben verschiedener Rollen, eindenken in neue/ungewohnte Positionen
- Verständnis gesellschaftlicher Prozesse

Diese Ziele sind nicht als Komplement zu klassischen Spielen anzusehen. Sie ergänzen sie um die o. g. Lernkomponenten.

Kapitel 4

Krisenmanagement

Eine Krise ist ein komplexes Szenario, welches im Unternehmen in der Form noch nicht aufgetreten ist und zur Bewältigung überdurchschnittliche humane und finanzielle Ressourcen benötigt. Eine Krise ist der Höhepunkt einer Kette von Ereignissen, welche als einzelnes Problem eine geringere Auswirkung als die eigentliche Krise haben und in Relation zur Krise mit einfachen Mitteln gelöst werden kann. Ob eine Krise wirklich eine Krise ist, hängt vom subjektiven Empfinden ab. Insbesondere spielt zur Einstufung die persönliche Erfahrung der Krisenbewältiger sowie die auf ihn wirkenden äußeren Einflüsse eine Rolle.

Krisenmanagement umfasst eine Reihe von Maßnahmen, die auf unerwartete Ereignisse mit höchstwahrscheinlich negativen Auswirkungen reagieren. Auf der einen Seite sind dies Maßnahmen die getroffen werden, um Krisen zu vermeiden bzw. die Häufigkeit ihres Auftretens verringern. Auf der anderen Seite stehen Maßnahmen und Pläne, die bei der Bewältigung einer Krise eingesetzt werden können. Ein erfolgreiches Krisenmanagement deckt beide Aspekte ab. Ein Fokus allein auf die Prävention ist aufgrund der schier unbegrenzten Variation von Krisen nicht ausreichend.

Krisenmanagement kann demnach sowohl aktiv als auch reaktiv stattfinden. Aktives Krisenmanagement lässt sich untergliedern in antizipatives und präventives Krisenmanagement. Während es bei der erstgenannten Form um die gedankliche Vorwegnahme möglicher Krisen und die Vorbereitung von Alternativplänen geht, geht es im Falle des präventiven Krisenmanagements um die Einführung von Frühwarnsystemen, welche erste schwache Signale einer Krise wahrnehmen, sodass entsprechende Maßnahmen ergriffen werden können um eine Krise abzuwenden. Demgegenüber steht das reaktive Krisenmanagement, welches dann zum Einsatz kommt, wenn eine Krise bereits eingetreten ist. Hier liegt der Fokus auf der schnellstmöglichen Bewältigung der Krise, indem geeignete Maßnahmen und Pläne erarbeitet und umgesetzt werden [1].

Ein wichtiger Aspekt des Krisenmanagements ist zudem die Kommunikation. Insbesondere sind alle betroffenen und involvierten Personen über alle Ereignisse zu informieren. Die Kommunikation ist dabei gewissen Regeln unterworfen. So sollten die Informationen so einfach wie möglich gehalten werden, um Missverständnissen vorzubeugen und komplexe Sachverhalte zu vereinfachen. Darüber hinaus ist es von großer Bedeutung, dass Informationen schnell an die involvierten Personen weitergeleitet werden. Dabei darf jedoch nicht vernachlässigt werden, dass die Informationen stets auf ihre Korrektheit geprüft werden müssen. Um die Verständlichkeit der Informationen zu gewährleisten, ist eine einheitlicher, zuvor festgelegter Sprachgebrauch zu empfehlen.

4.1 Planspiele zur Bewältigung von Krisen in Rechenzentren

Ein Rechenzentrum besteht aus vielen einzelnen Computern. Computer können aufgrund ihrer Komplexität eine Vielzahl von Störungen erleiden. Dabei kann ein Erklärungsversuch die allgegenwärtige Fehlerquelle „Mensch“ sein. Einerseits wird fremde Software eingesetzt, für dessen fehlerfreie Funktionsweise die Betreiber des Rechenzentrums keine Verantwortung tragen. Andererseits hat ein Rechenzentrum viel eigens entwickelte Software zur Verwaltung interner Prozesse sowie zur Inanspruchnahme durch Kunden. Je nach Qualität der Software und gegenseitigen Abhängigkeiten können Störungen leicht entstehen.

Hardwareseitig gibt es eine Vielzahl von Komponenten, die zu einer Störung führen können. Dies sind die einzelnen Komponenten der Computer, die Verkabelung bzw. die Infrastruktur innerhalb des Rechenzentrums sowie die Anbindung nach Außen und die Anbindung an das öffentliche Stromnetz sowie die eigenen Stromgeneratoren. Aufgrund der hohen Komplexität der Einzelkomponenten und der nicht vorhersehbaren Schwierigkeiten, die durch den Zusammenschluss der einzelnen Komponenten entstehen können, wird deutlich, dass die Mitarbeiter eines Rechenzentrums niemals auf jede mögliche Störung vorbereitet sein können.

Um jedoch eine bestmögliche Vorbereitung auf Störungen und Krisen zu ermöglichen, können Planspiele eingesetzt werden. Wie im vorangegangenen Kapitel bereits angesprochen, kann hierdurch eine strukturierte Vorgehensweise bei der Krisenbewältigung gelernt werden. Bei der Umsetzung eines Planspiels zur Bewältigung von Krisen ist darauf zu achten, dass der wichtige Aspekt der Kommunikation umfangreich abgedeckt wird. Hierzu können beispielsweise Chats, Mails, Protokolle und Flipboards in das Spiel integriert werden. Das Planspiel sollte zudem vermitteln, dass die unterschiedlichen Mitarbeiter nur gemeinsam

eine Lösung für die Krise finden können und daher die unterschiedlichen Fähigkeiten der Mitarbeiter zum Einsatz kommen müssen.

Kapitel 5

Grundlagen des Prototype-Driven Development

Zur Entwicklung des Planspiels bzw. der Komponenten werden Werkzeuge des prototype-driven Verfahren [19] eingesetzt. Diese benutzen Teilaspekte des Rapid-Prototyping. Die Verfahren sollen es „ermöglichen, Anhand der Spezifikation schnell einen realen Prototypen zu entwickeln. Dafür sind sowohl die Simulation des Prototypen als auch die Möglichkeit Änderungen schnell vornehmen zu können, wichtig“[20, S. 16ff].

Im Laufe des Rapid-Prototypings wird zunächst ein Produkt erstellt, welches in einer abstrahierten Form (beispielsweise in Form eines Modells) implementiert wird. Diese Abstraktion beinhaltet grob die Komponenten bzw. kritische Aspekte des zu entwickelnden Systems. Dadurch können bereits zu sehr früher Entwicklungszeit Anforderungen verifiziert, kritische Aspekte erkannt und ausgebessert werden. Durch das Rapid-Prototyping werden die Schwächen des traditionellen Wasserfallmodells ausgebessert: Auftragnehmer- und Geber können die Spezifikation der Kernelemente frühzeitig und kollektiv abschließen, noch bevor das eigentliche Produkt erstellt und Entwicklungskosten verursacht werden [13, S. 2].

In der Literatur werden drei Arten des Rapid-Prototyping unterschieden, wobei sie sich teilweise gegenseitig als Basis dienen [23, S. 29-32]:

- *Konzeptorientiertes Rapid-Prototyping* wird verwendet, um einen völlig realisierungsunabhängigen Prototypen zu entwickeln. Häufig werden dazu grafische Modellierungstools benutzt, aus denen anschließend Code automatisch generiert werden kann.

- *Architekturorientiertes Rapid-Prototyping* folgt häufig auf das konzeptorientierte Rapid-Prototyping. Es berücksichtigt spezifische Faktoren des späteren Zielsystems, wie beispielsweise die Hardwarecharakteristika des zugrundeliegenden Systems.
- Beim *realisierungsorientierten Rapid-Prototyping* werden hochspezialisierte Prototypen, die nur für konkrete Anwendungen bestimmt sind, erstellt. Meist bestehen diese aus bereits existierenden Komponenten, deren Funktionalität erweitert oder verbessert wird.

Im Gegensatz zu den ersten beiden, eher weniger spezialisierten, Prototypingverfahren, weist das realisierungsorientierte Rapid-Prototyping für vordefinierte Ziele Vorteile auf. Es erlaubt „sehr genaue Leistungstests, wie z.B. formale Vorgaben an die Einhaltung von Projektzielen und enthält häufig handgeschriebenen Code, der die Probleme der speziellen Anforderungen des Projekts behandelt [23]. Die Modellierung nach diesem Verfahren ist aufgrund der Tiefgründigkeit aufwendiger.

Der Prozess des Rapid-Prototyping lässt sich in verschiedene inhaltliche bzw. zeitliche Abschnitte einteilen. Zu Beginn findet die Modellierungsphase statt, in der die einzelnen Komponenten des Planspiels modelliert und die Sinnhaftigkeit durch Erstellen von Bedingungen überprüft wird. Anschließend wird das Modell in eine zweite Schicht exportiert, in der Spielkomponenten bzw. spielbare Elemente in der Zielsprache gebaut werden können.

Die Realisierung des Spiels erfolgt mithilfe von Teilaspekten des Rapid-Prototyping (*prototype driven*). In der *Modellierungsphase* wird ein spielbares Modell passend zur Anforderungsdefinition erstellt und im weiteren Verlauf durch Überführung in ein Anwendungsgerüst weiter bis zum Endprodukt hin verfeinert. Anschließend werden in der *Implementierungsphase* die Spielkomponenten erstellt und explizite Anforderungen umgesetzt.

Die Phasen des prototype-driven Verfahren und die darin verwendeten Werkzeuge werden nachfolgend beschrieben.

5.1 Modellierungsphase

Die Modellierungsphase lässt sich als Iteration bezeichnen, die immer wieder ausgeführt wird. Zu Beginn jeder Iteration werden Inhalte, Anforderungen und Bedingungen zusammen getragen. Diese Komponenten werden abstrahierter Weise in eine Modellierungsumgebung und zu einem ausführbaren Zwischenprodukt überführt. Durch die Ausführung des Zwischenprodukts lässt sich die Sinnhaftigkeit der Komponenten validieren. Fehlerhafte Ansätze oder neue Ideen lassen sich mit vergleichbar geringem Aufwand im Modell an-

passen und das Zwischenprodukt erneut erzeugen, womit die nächste Iteration begonnen wäre. Solche Modellierungsumgebungen erlauben in der Regel das Setzen von Bedingungen an Komponenten von Diagrammen. Ein Beispiel für solch ein Werkzeug ist die von der Projektgruppe genutzte *Cinco SCCE Meta Tooling Suite*.

5.1.1 Cinco SCCE Meta Tooling Suite

Cinco [7, 19] ist eine Entwicklungsumgebung zum Erstellen von domänenspezifischen, graphischen Modellierungstools. Das Werkzeug wird vom Lehrstuhl für Programmiersysteme an der Technischen Universität Dortmund entwickelt und befindet sich aktuell in der Version 0.6.

Cinco kombiniert das Metamodell-Framework *Eclipse Modeling Framework* [9] mit dem *Graphiti Diagram Editor* [10]. Des Weiteren unterstützt Cinco die Integration von Prozessen, die mit *Java Application Building Center (jABC)* [24] erzeugt wurden. jABC ermöglicht die Prozessmodellierung bzw. Bearbeitung von Workflows. So können zu einem späteren Zeitpunkt Domänen Experten die Modellabläufe erweitern oder ändern.

Wesentliches Ziel von Cinco ist die Generierung von neuen *Eclipse* [8] Entwicklungsumgebungen, welche genau auf die vorher definierten Domänen zugeschnitten sind. Dadurch muss sich der Benutzer nicht mit internen Eclipse Strukturen und APIs befassen, welche zum Erstellen einer Eclipse Anwendung unter normalen Umständen nötig wären. Die daraus resultierenden Werkzeuge sind besser zugeschnitten als zum Beispiel mit *BPMN* [14] oder *UML* [15]. Zugleich ummantelt Cinco viele der technisch komplexen Implementierungsdetails.

Um die Werkzeuge zu generieren gehen Cinco Nutzer in zwei Schritten vor. Als erstes wird ein *Meta Graph Language Model* (MGL-Modell) erstellt. Dieses gibt an, welche Modellkomponenten und Attribute in dem generierten Produkt zur Verfügung stehen. Aus diesem Metamodell wird das Modellierungstool generiert.

Ferner wird ein Style-Modell entworfen, dass das Aussehen der einzelnen Komponenten, wie zum Beispiel Kanten und Knoten, beschreibt.

Aus diesen Komponenten kann ein Produkt generiert werden, das direkt lauffähig ist und die vom Nutzer erstellten Komponenten darstellt. Mit diesem Produkt können nun die Anforderungen zum zugrundeliegenden System validiert und überprüft werden.

5.2 Implementierungsphase

Die Implementierungsphase ist die Phase, in der die technisch funktionale Umsetzung des zugrundeliegenden Systems erfolgt. Das finale Modell wird entweder manuell oder durch ein zum Modellierungswerkzeug gehörendes Code-Generierungstool in das Modell der zu implementierenden Anwendung überführt.

5.2.1 DyWA

DyWA, die Kurzform für *Dynamic Web Application*, ist ein Framework welches eine prototypische, modellbasierte Entwicklung von Webapplikationen ermöglicht [11].

Das Framework wird vom Lehrstuhl für Programmiersysteme der technischen Universität Dortmund entwickelt. Die Basis von DyWA bildet JavaEE 7. Klassische APIs für Enterprise-Web-Anwendungen, wie Beispielsweise die *Java Persistence API* oder *Dependency Injection*, sind häufig genutzte Komponenten.

Da Webanwendungen heutzutage immer komplexer werden, hat sich das DyWA Framework die Trennung von Frameworkentwicklung und domänenspezifische Anwendungsentwicklung als Kernziel gesetzt. Da Entwickler in der Regel kein Expertenwissen haben und Anwendungsexperten in der Regel nur domänenunspezifisches Wissen, aber keine Programmierkenntnisse, sollen Entwickler das Framework unabhängig von domänenspezifischen Kontext weiterentwickeln und Anwendungsexperten sich um die Domäne kümmern. Dabei ist das Framework besonders für agile Entwicklungsansätze geeignet, da dieses für eine sich ständig ändernde Domäne ausgelegt ist. So kann sehr schnell ein neuer lauffähiger Prototyp erstellt werden.

Das Projekt wird mit Hilfe von DyWA realisiert, da DyWA ein Applikationsrahmen erstellt der zum Beispiel die Persistenz übernimmt und gleichzeitige eine GUI generiert, ohne das viel Code geschrieben werden muss.

Pyro

Pyro ist ein Werkzeug, das im Rahmen einer Masterarbeit entworfen und entwickelt wird. Es ermöglicht das automatisierte Überführen von Cinco Produkten in das DyWA-Framework. Dabei entsteht ein als Webanwendung ausführbarer graphischer Editor, welcher UI- und Persistenzfunktionalitäten erhält.

5.3 Testphase

In der Testphase wird die Anwendung getestet und auf Fehler überprüft. Dabei können jedoch nur System- und Abnahmaetests durchgeführt werden, da die generierten Komponenten nicht ohne weiteres getestet werden können. Ein für das Rapid-Prototyping interessanter Ansatz wäre es deshalb, einzelne Komponenten mit Testklassen ausstatten zu können.

Kapitel 6

Planspieldefinition: Krisenplan

Die zentrale Idee des Spiels lässt sich folgendermaßen veranschaulichen: Der Spieler betreibt ein Rechenzentrum, das eine bestimmte Menge an Rechenleistung besitzt. Seine Aufgabe ist es während des Spiels unterschiedliche Aufträge anzunehmen, diese zu bearbeiten und damit seine Vergütung zu erhöhen. Das laufende Rechenzentrum verursacht Kosten, weshalb der Spieler Einnahmen erzeugen muss, welche sich als Spielaufgabe charakterisieren lässt. Der Spieler muss nun eine Balance zwischen Dienstleistungen und Instandhaltung des Rechenzentrums finden, wobei sein primäres Ziel darin besteht, zum Spielende hin eine positive Balance respektive einen maximal großen positiven Betrag zu erreichen.

Die wesentliche Problematik eines jeden Rechenzentrums, nämlich die mögliche Komplexität von Störungen, soll in das Planspiel überführt werden. Dazu sollen Störungen in den laufenden Spielbetrieb eingebaut werden, die nicht offensichtlich erkennbar sind und sich beispielsweise bei fehlender Beseitigung zu größeren Störungen entwickeln. Spieler sollen ferner durch Aufgaben (s. Mini-Spiele) in Zeitstress versetzt werden. Das gesamte Spiel beinhaltet eine Vielzahl weiterer Komponenten und Gedanken, die nachfolgend in diesem Kapitel erläutert werden.

6.1 Kernelemente und Ziele des Spiels

Das wesentliche Ziel des Planspiels ist es, Lehrinhalte zu vermitteln. Diese sollen unter anderem die Fähigkeiten beinhalten, angemessen mit einer Krisensituation umzugehen. Ferner soll das Planspiel Fachwissen, dass Rechenzentrumsmitarbeiter innerhalb ihrer Arbeitsabläufe anwenden können, liefern.

6.1.1 Spielmechanik

Eine Einstufung in ein Genre orientiert sich in der Regel an der Spielmechanik. So ist es für Action-, Shooter- und Echtzeit-Spiele typisch, schnell mit dem Spiel interagieren zu können. Das Spiel konfrontiert den Spieler mit einer für ihn zur Entscheidung drängenden Umgebung. Ein Beispiel dafür ist die von *Valve* veröffentlichte Ego-Shooter-Serie *Counter-Strike*. Eine beispielhafte Entscheidung könnte sich in folgender Fragestellung repräsentieren: „Laufe ich nach links, weil der Gegner rechts hinter der Wand ein Geräusch von sich gegeben hat oder verweile ich an diesem Ort?“. Der Spieler wird dadurch vor eine Entscheidung gestellt, deren Ausgang das weitere Geschehen beeinflusst. Falls sich der in oben beschriebener Situation befindliche Spieler für das Warten entscheidet, könnte er durch andere Mitspieler gesehen werden. Andererseits könnte der Gegenspieler ungeduldig werden und einen Fehler machen. Anhand des Beispiels ist erkennbar, dass eine Interaktion und dessen Konsequenz selbst bei diesen wenig komplexen Spielen, die eine Entscheidung in Echtzeit erfordern, umfangreich sind. Die Konsequenzen erkennt der Spieler meist nach kürzester Zeit.

Im Gegensatz zu Echtzeitspielen müssen Spieler bei strategischen Spielen, wie beispielsweise (rundenbasierten) Simulationen oder Planspielen, Entscheidungen im Voraus bedenken und werden möglicherweise erst im späteren Spielverlauf für diese Entscheidung bestraft oder belohnt. Ein Beispiel aus Städtesimulationen wäre folgendes Szenario: „Investiere ich jetzt €10.000 für den Ausbau der öffentliche Ämter wie der Feuerwehr? Fehlt mir das Geld anschließend im weiteren Verlauf an anderer Stelle, beispielsweise Bildung? Was passiert, wenn ich statt in Feuerwehreinrichtungen in Bildung investiere? Lösen meine Bürger weniger Feuer aus, weil sie gebildeter sind?“. Kausale Zusammenhänge werden dabei genauso wie bei Echtzeitspielen direkt für den Spieler erkennbar. Die Konsequenzen sind jedoch schwieriger abzuschätzen, weshalb die Entscheidungsfindung häufig intensivere Überlegungen erfordert. Effekte werden später sichtbar und sind für vorausschauende und gut überlegende Spieler in der Regel ertragreicher.

Echtzeitsimulation

Die Projektgruppe entscheidet sich für eine Spielmechanik vom Typus Echtzeitsimulation. Nachfolgend werden die Anforderungen des Planspiels an die Spielmechanik erörtert, um die Entscheidung zu legitimieren.

Eine Simulation ist die Adaption eines Systems als Modell. Das System kann durchaus theoretisch, jedoch auch real existent sein. Die Simulation wird in einer künstlichen Umwelt durchgeführt, sodass die reale Umgebung unberührt bleibt. Solch eine Simulation wird mit

der Absicht durchgeführt, neue Erkenntnisse über das System gewinnen zu können, ohne es erst aufwendig entwickeln oder das existente System gefährden zu müssen.

Die Wahl der PG592, um die Anforderung, ein Planspiel mit wirtschaftlichen Simulationsaspekten im Bereich Rechenzentrum zu erstellen, zu erfüllen, erfordert den zweiten Typus Spielmechanik. Handlungen müssen früh bedacht und geplant werden. Strategisches, planmäßiges und musterhaftes Vorgehen wird belohnt, nicht rational nachvollziehbare Entscheidungen werden bestraft. Dabei geht es im Umgang mit dem Eintritt von unvorhersehbaren Umständen (Krisen) um die Vermittlung von Lerninhalten bei Planspielen.

6.1.2 Spiel-Metriken

Ein Ziel des Planspiels ist es, die Resultate eines Teams sichtbar und messbar zu machen. Aus diesem Grund werden in das Planspiel diverse Metriken integriert, anhand welcher der Erfolg eines Teams beurteilt werden kann. Dadurch sollen Lerneffekte im Verlaufe des Planspiels deutlich und eine Vergleichbarkeit mit anderen Teams gewährleistet werden. Darüber hinaus sollen die Metriken auch Einfluss auf das Spielgeschehen haben, indem sich beispielsweise eine hohe Kundenzufriedenheit positiv auf die Auftragslage und eine niedrige Kundenzufriedenheit negativ auswirkt.

6.1.3 Metrik-Kategorien

Zu diesem Zweck wurden die Metriken zunächst in drei verschiedene Kategorien unterteilt:

1. Ökonomische Metriken
2. Rechenzentrum-bezogene Metriken
3. Metriken zur Messung der Kundenzufriedenheit

Die Unterteilung der Metriken in Kategorien dient der besseren Visualisierung und soll für eine bessere Lesbarkeit und Verständlichkeit sorgen. Jede dieser Kategorien umfasst wiederum einzelne Metriken, die nun im Folgenden genauer vorgestellt werden sollen:

Ökonomische Metriken

Zu den ökonomischen Metriken zählen in erster Linie all die Metriken, die eindeutig messbar, also in absoluten Zahlen gemessen werden können, den Erfolg des Rechenzentrums direkt widerspiegeln und Aussagen über die Wirtschaftlichkeit des betriebenen Rechenzentrums ermöglichen. Dazu werden die folgenden Metriken in das Planspiel integriert:

- **Umsatz:**

  In die Umsatz-Metrik fließen alle umgesetzten Geldbeträge des Rechenzentrums ein. Darunter fallen in erster Linie die Erlöse aus erfolgreich abgeschlossenen Aufträgen.

- **Kosten:**

  Die Kosten-Metrik besteht aus vielen einzelnen Faktoren. Die Kosten setzen sich in erster Linie aus den Betriebskosten der Server und den Strafkosten durch die Überschreitung von Fristen zusammen. Die Betriebskosten lassen sich wiederum in Grundkosten, Auslastungskosten und Job-Zuteilungskosten aufschlüsseln. Während die Basiskosten nur von der Anzahl der im Rechenzentrum befindlichen Server abhängen, sind die Auslastungskosten davon abhängig, wie stark ein jeweiliger Server belastet wird. Die Job-Zuteilungskosten sind von der Zahl der einem Server zugewiesenen Jobs abhängig. Die einzelnen Kosten lassen sich vom Spielleiter zu Beginn eines Spiels festlegen. Unter den Strafkosten bei der Überschreitung von Fristen versteht man diejenigen Kosten, die entstehen, wenn ein Auftrag nicht in dem im Vertrag mit dem Kunden ausgehandelten Zeitraum erfüllt werden konnte. Die Strafkosten werden dabei nicht vom Spielleiter festgelegt, sondern werden durch den Kunden im Vertrag festgesetzt.

- **Absoluter Gewinn / Verlust:**

  Die Metrik *Gewinn* ist eine Kombination der Metriken *Umsatz* und *Kosten*, indem die Differenz der beiden Metriken gebildet wird. Die dadurch entstehende Metrik soll daher lediglich der besseren Übersicht dienen.

- **Gewinnverlauf / Zeit in Schulden:**

  Ziel eines jeden Unternehmens ist es, nach Möglichkeit immer in der Gewinnzone zu bleiben. Daher ist es wünschenswert, die Zeit, die sich das Rechenzentrum in der Verlustzone befindet, möglichst kurz zu halten. Diese Metrik dient dazu, die Zeit in Schulden visuell darzustellen und den Gewinn- bzw. Verlustverlauf auf einen Blick sichtbar zu machen. Die Zeit wird in dem Planspiel als *Ticks* bezeichnet und ist in Form eines diskreten Zeitverlaufs realisiert (weitere Hinweise siehe Abschnitt 7.3.1).

Rechenzentrum-bezogene Metriken

Die zweite Kategorie umfasst alle Metriken, die rein Rechenzentrum-bezogen sind. Darunter fallen:

- Zahl der ausgefallenen Server
- Zeitspanne bis zur Wiederherstellung der Funktionalität eines ausgefallenen Servers

  - Bis zu 2 Ticks
  - 3 bis 5 Ticks
  - Über 5 Ticks

Metriken zur Messung der Kundenzufriedenheit

Die dritte große Gruppe an Metriken sind diejenigen zur Messung der Kundenzufriedenheit. Im Gegensatz zu den Finanz-Metriken ist die Kundenzufriedenheit nicht derart einfach messbar. Stattdessen bedarf es verschiedener Parameter, die Einfluss auf die Kundenzufriedenheit nehmen können:

- **Zahl der fristgerecht bearbeiteten Aufträge (I):**

  Für die Zufriedenstellung von Kunden ist es für ein Rechenzentrum essenziell, die im Vertrag vereinbarten Fristen zur Erfüllung von Aufträgen einzuhalten. Aus diesem Grund misst eine Metrik, wie viele Aufträge innerhalb der vorgegebenen Frist bearbeitet wurden.

- **Zahl der schneller als vorgesehen bearbeiteten Aufträge (II):**

  Eine weitere Metrik beschreibt die Zahl der Aufträge, die schneller als ursprünglich vorgesehen bearbeitet werden konnte.

- **Zahl der überschrittenen Fristen (III):**

  Dem gegenüber steht die Zahl der Aufträge, deren Frist überschritten wurde. Auch diese Metrik spielt für die Kundenzufriedenheit eine große Rolle.

- **Längen der überschrittenen Fristen (IV):**

  Analog zu der Zahl der Aufträge soll ebenfalls gemessen werden, wie stark die Fristen überschritten wurden. Dazu werden die Überschreitungen in drei verschiedene Bereiche eingeteilt:

  - Überschreitung von bis zu 2 Ticks
  - Überschreitung von 2 bis 5 Ticks
  - Überschreitung von über 5 Ticks

- **Zahl erfolgreich abgeschlossener Aufträge (V):**

  Für die Kundenzufriedenheit spielt nicht nur die fristgerechte Bearbeitung eines Auftrages eine Rolle, vielmehr ist auch die Zahl der insgesamt erfolgreich abgeschlossenen Aufträge von Bedeutung.

- **Zahl nicht erfolgreich / abgebrochener Aufträge (VI):**

  Für die Kundenzufriedenheit spielen auch abgebrochene Aufträge eine große Rolle, also jene Aufträge, die nicht ordnungsgemäß bearbeitet und abgeschlossen werden konnten. Die Zahl der abgebrochenen Aufträge wird deshalb im Planspiel ebenfalls dargestellt.

- **Verhältnis von erfolgreich und nicht erfolgreich bearbeiteten Aufträgen (VII):**

  Diese Metrik stellt eine Kombination der beiden vorangegangenen Metriken dar. Das Verhältnis der erfolgreich und der nicht erfolgreich bearbeiteten Aufträgen wird im Planspiel prozentual dargestellt.

Die vorangegangenen Metriken bieten eine detaillierte Möglichkeit, den Erfolg verschiedener Spielteams miteinander zu vergleichen, einzuordnen und den Lernerfolg im Laufe eines Spiels zu dokumentieren.

6.2 Mini-Rätsel

Ein Kernelement des Krisenplan-Simulators stellt das Erzeugen von Stresssituationen seitens der Benutzer dar. Durch solche Situationen soll überprüft werden, wie die Spieler auf Stress reagieren und Lösungen gemeinsam ausarbeiten. Aus diesem Grund wird diesem Teilbereich eine große Bedeutung zuteil, damit diese Aspekte ausreichend und möglichst effizient angewendet werden können. Um Stresszustände zu generieren bieten sich unter anderem die Mini-Rätsel an.

In der Abschlusspräsentation durchgeführten Spielrunde wurde deutlich, dass die Mini-Rätsel ihren Zweck erfüllen. Der Spieler befand sich in einem Zustand, in dem mehrere Komponenten des Rechenzentrums ausgefallen waren und durch das Lösen eines Rätsels repariert werden können. Der Kontostand des Spielers wurde durch den Ausfall immer stärker belastet. Da mehrere Komponenten ausgefallen waren, musste der Spieler mehr als ein Rätsel lösen. Irgendwann wurde die Belastung so hoch, dass der Spieler im Rätsel eine fehlerhafte Entscheidung durchführte und in ein zeitliches Defizit geriet. Der Spieler versucht, die Problemsituationen möglichst schnell zu lösen, organisiert sich jedoch zu wenig. Die Entscheidungswahl in solch einer Situation lässt sich in die Kategorien *schnelles Probieren mit Risiken* bis *langsames Lösen mit Verlusten* differenzieren. Ziel ist es dabei, dem Spieler zu vermitteln, dass er mit Organisation und vernünftigen Überlegungen das Risiko minimieren und eine ökonomische Lösung anstreben kann, ohne die erstbeste Möglichkeit mit unbekanntem Risiko auszuprobieren.

6.3 Spielrollen

Die Aufgaben innerhalb eines Rechenzentrums werden für das Planspiel insgesamt auf vier Rollen abstrahiert. Diese sind Kunde, Finanzpersonal, Technikpersonal sowie Ankaufspersonal. Da die vier Rollen die relevanten Tätigkeiten aller Geschäftsfelder beinhalten [5] , können diese in vielerlei Hinsicht das Spielgeschehen beeinflussen. Im vorliegenden Kapitel werden die Tätigkeiten und Aufgaben der unterschiedlichen Rollen detailliert dargestellt.

6.3.1 Kunden

Für fast alle Firmen oder Betriebsorganisationen stellt die Pflege der Kundenbeziehungen eine der wichtigsten Geschäftstätigkeiten dar [17] . Die Kunden bieten ihre Aufträge (Jobs) mit einer enstprechendenen Vergütung und gewünschten Fristen in der Jobbörse (Jobmarkt) an. Die Aufträge in der Jobbörse sind für alle öffentlich, d.h. dass die Aufträge sowohl vom Rechenzentrum, als auch von anderen (fiktiven) Betriebsorganisationen akzeptiert werden können. Wenn die Deadlines der akzeptierten Aufträge nicht eingehalten werden, erhalten die Kunden vom Rechenzentrum entsprechende Ausgleichszahlungen. Außerdem können die Kunden regelmäßig den aktuellen Status ihrer Aufträge erfragen.

6.3.2 Ankaufspersonal

Im Rechenzentrumsmodell dient das Ankaufspersonal als Zuständiger für die Auftragsannahme sowie als Kundenberater. Diese Rolle beschäftigt sich damit, die Aufträge zu akzeptieren und die Kunden zu betreuen. Das Ankaufspersonal wählt unter mehreren Aufträgen in der Jobbörse aus und kann daraufhin mit den Kunden kommunizieren. Generell spielt die Kommunikation eine wichtige Rolle im Tätigkeitsbereich des Ankaufspersonals. Diese Rolle muss Meinungen und Informationen über die Aufträge mit den Kunden austauschen und rechtzeitig die Fragen von Kunden beantworten. Um diese Tätigkeit zu erledigen, muss das Ankaufspersonal jederzeit mit seinen Kollegen des Finanz- und Technikpersonals kommunizieren. Der Arbeitsbereich des Ankaufspersonals wird als Übersicht im Rechenzentrum dargestellt, durch das die Kunden nötige Informationen erhalten, gewünschte Aufträge annehmen und positive Rückmeldungen sammeln, um den Betrieb des Rechenzentrums zu verbessern.

6.3.3 Technikpersonal

Da sich das zu modellierende Rechenzentrum mit IT-Dienstleistungen beschäftigt, stellt der technische Bereich eine der Kernkomponenten dar. Der normale Betrieb wird von vielen

verschiedenen Geräten gestützt, wie z.B. Servern, Generatoren sowie Kühlgeräten. Eine wichtige Aufgabe des Technikpersonals ist es daher, diese Geräte regelmäßig zu warten, um den reibungslosen Ablauf des Rechenzentrums zu gewährleisten.

Im Planspiel dient das Technikpersonal außerdem dazu, die angenommenen Aufträge entsprechend zuzuweisen. Die von dem Ankaufspersonal akzeptierten Aufträge werden ins Servicezentrum verschoben, für welches das Technikpersonal zuständig ist. Das Technikpersonal muss daraufhin jeden Auftrag einem Server zuweisen. Hierbei müssen der Rechenaufwand, die Deadline sowie die tatsächlich zur Verfügung stehenden Leistungen sorgsam berücksichtigt werden. Dies ist wichtig, da eine unvernünftige Zuweisung von Aufträgen zusätzliche Kosten und Auslastungen verursacht, welche wiederum zur Nicht-Einhaltung der Deadline führen kann. Hierdurch kann ein erheblicher finanzieller Schaden entstehen und die Beziehung zum Kunden verschlechtert werden, wobei sich dies in Form von weniger häufigen oder lukrativeren Aufträgen sowie einer früheren Storno-Gefahr des Auftrags auswirkt.

6.3.4 Finanzpersonal

In Anlehnung an andere Unternehmen steht beim Rechenzentrum die Generierung von Gewinnen im Vordergrund. Das Einkommen des Rechenzentrums wird hauptsächlich durch die Abarbeitung von Aufträgen generiert. Aus diesem Grund ist eine Finzanzpersonal-Rolle für das zu modellierende Rechenzentrum unerlässlich. Diese ist für die Verwaltung der Finanzen zuständig. Die Aufgaben des Finanzpersonals lassen sich in die Zustellung von Aufträgen, die Wartung von Geräten, sowie dem Überblick der Grundkosten des gesamten Betriebs gliedern. Das Finanzpersonal sollte daher den Fluss der Finanzen effizient verfolgen, rechtzeitig den aktuellen Zustand den Managern mitteilen und Vorschläge weitergeben.

6.4 Spielleitung

Die natürliche Aufgabe eines jeden Spielleiters ist es, die Spieler in das Spielgeschehen hineinzuführen, Fragen während des Spiels zu beantworten und eine Hilfestellung zu geben. Das anfängliche Einführen der Spieler in das Planspiel hängt stark von der Art des Planspiels und von der Erfahrung des Spielers ab. Ist der Spieler bereits mit dem Spielkontext vertraut, wird er keine große Einführung in die einzelnen realen Komponenten benötigen, sonder eher in die Spielmechanik. Der Spielleiter kann zusätzlich Merkmale zur optimalen Spielgestaltung anmerken. Im Anschluss an eine Runde kann von der Spielleitung zusätz-

lich eine Feedback-Runde durchgeführt werden, in dem Spielgeschehen, Spielinhalte und das Spiel an sich von den Teilnehmern bewertet werden können.

Ferner kann der Spielleiter in die Ressourcen- und Aufgabenverteilung des Spiels eingreifen. Das Planspiel erzeugt zu Beginn einer jeden Runde eine Schwierigkeitskurve, anhand der Aufträge, Störungen und Belohnungen verteilt werden. Jedes dieser Ereignisse, welches ohne einen unmittelbaren Auslöser auskommt, kann zu Spielbeginn geplant und visualisiert werden. Ein Spielleiter könnte in einer entsprechenden Spielansicht diese einsehen und modifizieren (beispielsweise der nächste eingehende Auftrag gibt eine höhere Belohnung, hat jedoch schwieriger einzuhaltende Anforderungen und komplexere Kriterien).

Eine weitere mögliche Rolle für den Spielleiter liegt in der Auswahl und der Verteilung der Minispiele (siehe Sektion 9.3). Der Spielleiter kann die für ein Problemszenario gedachte Aufgabe während des Spiels (jedoch vor Beginn des Problemszenarios) durch eine andere ersetzen oder mit einem höheren Schwierigkeitsgrad versehen. Der Spielleiter könnte auch während des Spiels, ohne dass ein konkretes Problem auftritt, eine Aufgabe zur Bewältigung aufgeben. Ferner könnte das Auslösen von Störungen und Krisen sowie das dynamische Anpassen der Schwierigkeitskurve Aufgabe der Spielleitung sein.

Dies sind triviale Ansätze zur Besetzung der Rolle des Spielleiters. Eine für das Spielgeschehen interessantere Positionierung des Spielleiters ist die Übernahme eines Kunden. Dieser kann *besser* (oder menschlicher) als ein computergesteuerter Kunde reale Anfragen an die Spieler verschicken und auf die Antworten dieser reagieren. Die Spieler würden mit dem Wissen, dass sich hinter den Kunden eine reale Person befindet, eines viel natürlicheren Drucks ausgesetzt werden.

6.5 Schwierigkeitsgrade

Um den Spielablauf abwechslungsreich und lehrreich gestalten zu können wird der Schwierigkeitsgrad abhängig von der Leistung und Erfahrung der teilnehmenden Gruppen bestimmt.

Im ersten Durchgang werden die Teilnehmer mit der geringsten Schwierigkeitsstufe konfrontiert. Hierbei erhalten sie ein ausreichend großes Budget sowie eine große Anzahl an lukrativen Aufträgen, die sie aus dem Job-Markt auswählen können. Eine steigende Schwierigkeit macht sich beispielsweise durch häufiger auftretende Rätsel (vgl. 9.3) bemerkbar. Diese haben den Sinn, den Spieler in eine unbekannte bzw. unerwartete Situation zu drängen sowie ein kurzes Erfolgserlebnis zu vermitteln, wodurch der Spieler zum Weiterspielen motiviert werden soll. Dieser Ablauf basiert auf der Grundannahme, dass ein Teilnehmer,

der Freude an der Simulation empfindet, einen höheren Lernerfolg erzielt, als ein Spieler, der das Spiel erzwungenermaßen beendet. [2].

Der Erfolg eines Teams wird zum Ende einer Spielrunde am vorhandenen Budget gemessen. Sofern zum Rundenende ein ausreichend großer Betrag erreicht wurde, wird die Gruppe in der folgenden Spielrunde mit einem höheren Schwierigkeitsgrad konfrontiert. Höhere Schwierigkeitsgrade zeichnen sich durch ein geringeres Startbudget und einem prozentual geringeren Wert an lukrativen Aufträgen im Job-Markt aus. Zusätzlich treten Rätsel mit höherer Wahrscheinlichkeit auf. Die Erhöhung des Schwierigkeitsgrades soll den Lerneffekt der einzelnen Teilnehmer erhöhen. Der Druck auf die Spieler soll zunehmend erhöht werden um die Kommunikation und die Mühe zur Problembewältigung zu intensivieren. Die Schwierigkeit wird bei erfolgreichem Rundenbestehen bis zu einer finalen dritten Runde erhöht.

6.6 Vermittlung von Lerninhalten

Klassische Lehrmethoden finden sich im Schulunterricht. Dabei gibt zum Beispiel eine Lehreinheit, die die Lehrinhalte im Frontalunterricht vorträgt. Klassische Lernmethoden sind unter anderem das Memorieren von entsprechenden, zu lernenden Inhalten (vergleiche Abschnitt 3.2). Obwohl sich diese Lehr- und Lernmethoden in der Gesellschaft etabliert haben, sind sie in vielen Bereichen ungeeignet. Die Schwierigkeit dahinter steckt darin, auf neue, möglicherweise noch nicht ausgiebig erforschte Lehr- und Lernmethoden umzusteigen.

Die wesentliche Herausforderung eines Planspiels ist, die Fähigkeit zu vermitteln, in einem komplexen Umfeld korrekte und rational nachvollziehbare Entscheidungen zu treffen. Hierbei besteht die Komplexität einerseits aus der Anzahl der Komponenten eines Sachverhalts. Andererseits kommt es ebenfalls auf die Vielfalt der Verknüpfungen zwischen den einzelnen Komponenten an. Diese Fähigkeit wird als Entscheidungskompetenz bezeichnet. Durch einen Vergleich der klassischen Lehrmethoden mit der Entscheidungskompetenz lässt sich feststellen, dass diese Fähigkeit nicht durch die altbewährten Lehrmethoden erlangt bzw. ausgebaut werden kann. Statisches Wissen, wie es die klassischen Lehrmethoden vermitteln, ist ungeeignet um komplexe Situationen bewältigen zu können.

Die bei der PG592 verwendete Vorgehensweise zur Vermittlung von Lerninhalten basiert auf einem simplen Feedbacksystem und Anzeigen wichtiger Merkmale im Spiel selbst. Nach einer definierten Anzahl von Runden entscheidet sich der Spielleiter, Rückmeldung an die Teilnehmer zu geben. Was haben die Spieler gut und was schlecht gelöst? Wo gibt es mögliche Verbesserungen und wie können diese erreicht werden?

Neben dieser direkten Methodik, erhalten die Teilnehmer des Spiels aber auch indirekte Hinweise, wie effizient sie spielen. Zum einen manifestieren sich Belohnungen und Spielabzüge in der Anzeige des aktuell verfügbaren Kapitals und der Finanzansicht. Zum anderen wird den Spielern schnell bewusst werden, dass das Thema Kommunikation ein wesentlicher Bestandteil des Spiels ist. Ohne die gegenseitige Verständigung benötigen Prozesse länger, sind schwieriger lösbar aufgrund mangelnder Informationen von anderen Teilnehmern und können so den ganzen Spielfluss beeinträchtigen. Dies ist vor allem in Hinblick auf die auftretenden Probleme bzw. Krisen in Form von Mini-Spielen (siehe Sektion 9.3) erkennbar. Details zum konkreten Spielablauf, der bereits erwähnten Kommunikation und der Einbindung von Krisen (Mini-Spielen) werden in nachfolgenden Kapiteln thematisiert.

Kapitel 7

Modellierungsphase

Im Rahmen der Projektgruppe wurde unter Verwendung des vom Lehrstuhl 5 der Informatik, TU Dortmund, bereitgestellten Tools *Cinco* [7] ein Meta-Model entwickelt, das die Anforderungen an die Projektziele anschaulich in einem (ausführbaren, d.h. spielbaren) Modell demonstriert. Nachfolgend werden die einzelnen Elemente des Modells detailliert erörtert.

7.1 Einführung

Im Rahmen der Projektgruppe wird ein Meta-Modell unter Benutzung von Cinco entwickelt, d.h. es werden Syntax und Semantik, sowie die grafische Darstellung und die ausführbaren Aktionen eines Systems festgelegt. Zusätzlich wird unter Verwendung der durch das Meta-Modell spezifizierten Sprache ein konkretes Modell eines Krisenplan-Spiels erstellt.

Nachfolgend wird zunächst eine Grundlage zum Verständnis des Modells gegeben, wobei einzelne Elemente sowie ihre Interaktionsmöglichkeiten näher erläutert werden.

7.2 Grundlagen des Planspielmodells

Die zentrale Idee des Spiels lässt sich folgendermaßen veranschaulichen: Der Spieler betreibt ein Rechenzentrum, das eine bestimmte Menge an Rechenleistung besitzt. Seine Aufgabe ist es während des Spiels unterschiedliche Jobs anzunehmen, sie zu bearbeiten und zum Schluss das versprochene Geld einzunehmen. Das laufende Rechenzentrum verursacht Kosten, weshalb der Spieler Einnahmen erzeugen muss, welches sich als Spielaufgabe

charakterisieren lässt. Der Spieler muss nun eine Balance zwischen Dienstleistungen und Instandhaltung des Rechenzentrums finden, wobei sein primäres Ziel darin besteht, zum Spielende hin eine positive Balance respektive einen maximal großen positiven Betrag zu erreichen.

Das Modell lässt sich grob in zwei Blöcke unterteilen: den *Kern* und die *Features*. Der Kern des Modells spiegelt den zentralen Spielprozess wieder (vgl. Abb. 7.1). Nachfolgend werden die einzelnen Komponenten aus Abb. 7.1 im Detail erläutert.

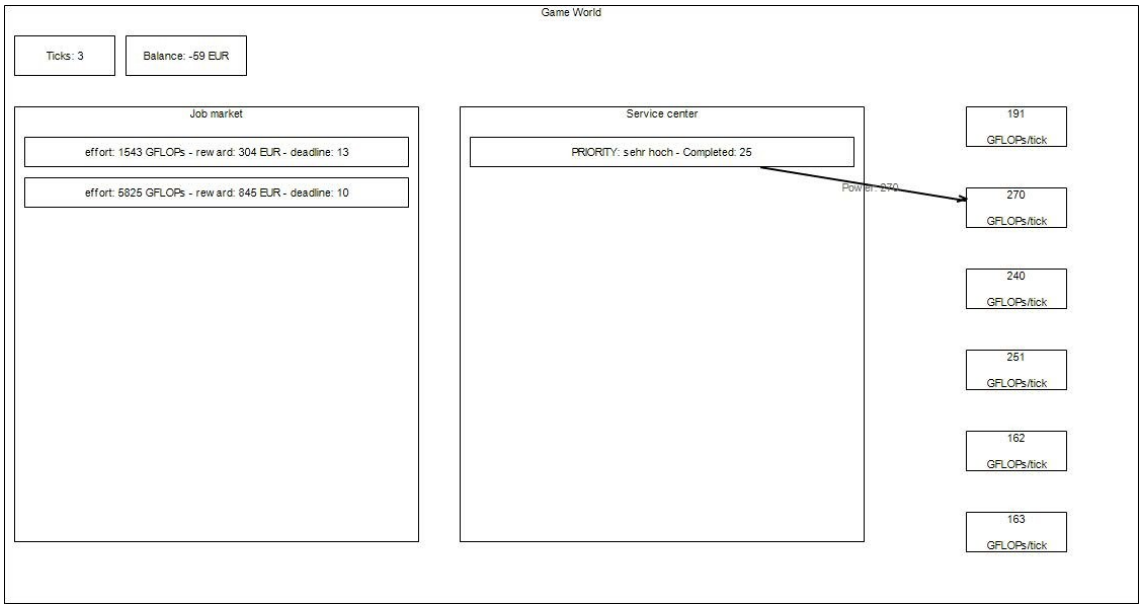

Abbildung 7.1: Kern des Modells

7.3 Komponentenmodellierung

In dieser Sektion werden die Elemente *Tick*, *Balance*, *Job Market*, *Job*, *Service Center* und *Server* detailliert erläutert.

7.3.1 Ticks

Wie in Sektion 6.1.1 erläutert, handelt es sich beim Krisenplan um ein Echtzeit Spiel mit diskretisierter Zeit. Gespielt wird in Ticks, welche einer Zeiteinheit innerhalb des Spiels entsprechen. Das Spiel beginnt immer im ersten Tick. Der Tick wird zum Ende einer laufenden Runde um 1 inkrementiert. Dies wird im Modell durch eine Doppelklick-Aktion auf die Tick-Kernkomponente erreicht, kann allerdings auch in einen automatischen Modus versetzt werden, in dem die Erhöhung des Ticks automatisch in von der Spielleitung bestimmten Intervallen stattfindet.

7.3.2 Balance

Krisenplan ist ein *Highscore-Spiel*, d.h. der Spieler oder die Gruppe mit dem höchsten Punktestand geht als Sieger aus dem Spiel hervor. Demzufolge ist das zentrale Ziel des Spiels bis zum Spiel- oder Rundenende eine höchstmöglichen Punkestand zu erreichen. Der konkrete Zeitpunkt wird vor Spielbeginn festgelegt und nicht im Modell definiert.

Der höchstmögliche Punktestand wird im Krisenplan als *Balance* definiert und symbolisiert die Beziehung zwischen Einnahmen und Ausgaben, wobei als Währung der Euro (€) gewählt wurde.

7.3.3 Job

Die Annahme und das Abarbeiten von *Jobs*, klassisch für Kundenauftrag, bilden den zentralen Kern des Krisenplans. Jobs sind stets unterschiedlich und bestehen aus einer Vielfalt von Parametern. Jeder Job hat einen charakterisierenden Rechenaufwand, der geleistet werden muss, bis er als erledigt angesehen wird. Der Lohn für erfolgreich fertiggestellte Jobs, der zur Balance des Spielers hinzugefügt wird, unterscheidet sich ebenfalls. Umgekehrt kann ein Job eine *Deadline* haben, einen Zeitpunkt, in dem er idealerweise bearbeitet wurde, die eingehalten werden sollte. Ein Verpassen dieser Deadline kann zu erhöhten Kosten oder geminderten Einnahmen führen.

Die vorher genannten Merkmale bilden eine allgemeine Charakterisierung eines jeden Jobs. Nachfolgend werden weitere Merkmale genannt, die bei jedem Job unterschiedlich ausfallen können, beispielsweise spezielle technische Anforderungen. Einige Jobs können höchstens auf einem Server abgearbeitet werden und erlauben keine Parallelität. Andere besitzen eine maximale Rechenleistung, die sie pro Tick akzeptieren können. Die Entscheidung, welche Jobs angenommen oder abgelehnt werden sollten, ist somit nicht trivial.

7.3.4 Job Market

Im *Job Market*, ein virtueller Marktplatz für Kundenaufträge, erscheinen (un-)regelmäßig Jobs, die vom Spieler frei akzeptiert oder abgelehnt werden können. Jobs verschwinden automatisch aus der Jobbörse, wenn ihre Deadline verstrichen ist. Zusätzlich können Jobs unerwartet früher verschwinden, was durch einen Rückruf seitens des Kunden oder durch Annahme durch ein anderes Rechenzentrum zu erklären ist. Zu langes Zögern wird auf diese Weise bestraft.

7.3.5 Service Center

Akzeptierte Jobs werden aus der Jobbörse in das Service Center, einen internen Auftragspool innerhalb des Rechenzentrums, verschoben. An dieser Stelle beginnt der Lebenszyklus eines Auftrags innerhalb des Rechenzentrums. In der Regel wird dieser Auftrag von einem entsprechenden zuständigen Arbeiter auf einem Server eingerichtet und ausgeführt. Angenommene Jobs können nicht mehr zurückgewiesen werden. Sie müssen folglich erledigt werden, was zur Einnahme der versprochenen Entlohnung führt oder abgebrochen werden, was zur Bestrafung des Spielers führt.

7.3.6 Server

Das Rechenzentrum verfügt über eine Reihe von Servern (Racks). Jeder Server bietet eine feste maximale Rechenleistung pro Tick, die für die zugewiesenen Jobs bereitgestellt wird. Bei Zuweisung von mehreren Jobs an einen Server bleibt es dem Spieler überlassen, wie die Rechenleistung unter den Jobs aufgeteilt wird. Zusätzlich generiert die Zuweisung von mehreren Jobs pro Server einen Overhead, etwa ein Leistungsverlust, da in diesem zwischen den einzelnen Jobs geschaltet werden muss.

Ferner bedarf jeder Server der Instandhaltung. Pro Tick verursacht jeder Server Kosten, die einerseits aus einem festen Basispreis sowie dynamischen Preisen zur Laufzeit geltender Eigenschaften, wie beispielsweise der Server-Auslastung, bestehen.

7.4 Features

Neben dem Kern des Modells, das den rudimentären Spielablauf widerspiegelt, besitzt das Krisenplan-Modell über eine zahlreiche Anzahl an Features, die zunächst trivial erscheinende Aufgaben des Spielers anspruchsvoll und interessant erscheinen lassen. Nachfolgend werden diese Features im Detail vorgestellt.

7.4.1 Kunden

Die Jobs im Spiel sind abhängige Erscheinungen. Jeder Job besitzt eine Zuordnung zu einem Kunden (vgl. Abb. 7.2).

Zur Spielzeit existieren Kunden, welche eine Menge der Spielwährung sowie Jobs besitzen. Zum Beginn eines jeden Ticks kann jeder Kunde mit einer gewissen Wahrscheinlichkeit neue Jobs anbieten und diese Jobs mit einem Teil seiner Spielwährung entlohnen. So-

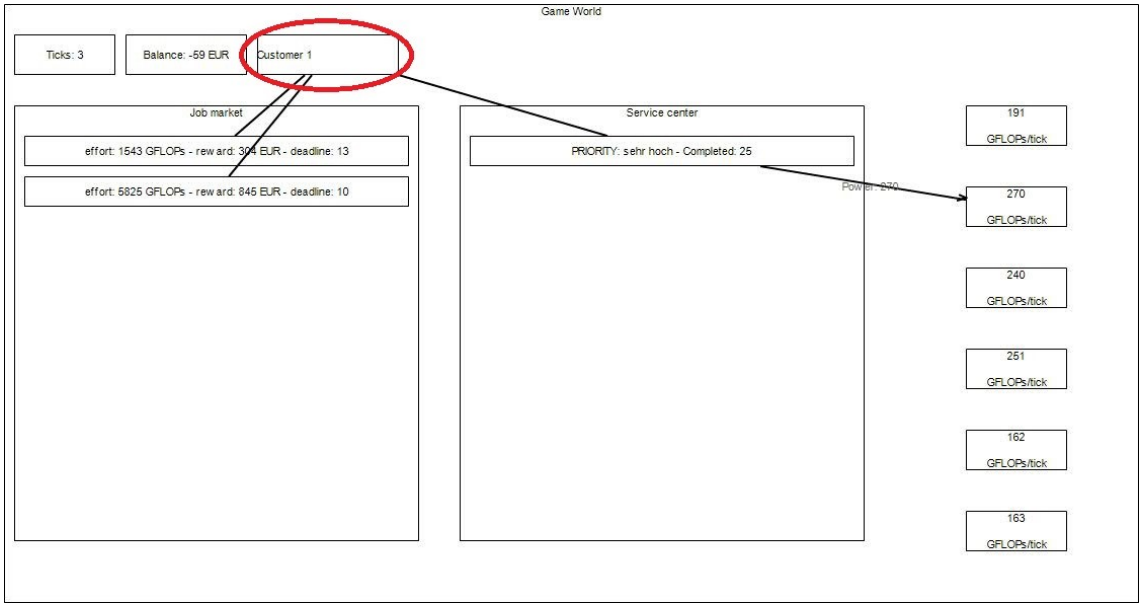

Abbildung 7.2: Zuordnung von Jobs zu Kunden

bald der Job erledigt ist, wird Geld vom Kunden zum Rechenzentrum transferiert, wobei der Betrag bei Mängeln verringert oder gänzlich ausfallen kann. Strafzahlungen an den Kunden sind ebenfalls möglich. Ein markantes Merkmal des Kunden ist, dass die Menge seines Geldes endlich ist. Es ist möglich, dass der Kunde im Laufe des Spieles kein Geld mehr hat und eine vom Rechenzentrum erbrachte Dienstleistung nicht entlohnen kann. Die pro-Tick-Schwankungen der Geldmenge der Kunden unterliegen gewissen vom Spiel bestimmenden Tendenzen, sodass eine weitere wichtige Aufgabe der Spieler ist, die Bonität eines Kunden vor Auftragsannahme zu prüfen.

Abseits der finanziellen Aspekte hat der Kunde zusätzliche, ihn charakterisierende, Merkmale. Ein Kunde kann jederzeit angebotene Jobs zurückziehen, auch wenn diese bereits vom Rechenzentrum abgearbeitet werden, wobei in letztem Fall eine Ausgleichszahlung an das Rechenzentrum gezahlt wird. Ferner erzeugt der Kunde Fragen über seine Jobs, etwa den aktuellen Abarbeitungsstand oder zusätzliche Wünsche. Die (Nicht-)Beantwortung dieser Fragen beeinflusst das Wohlbefinden des Kunden. Dies hat direkte Konsequenzen auf sein Verhalten: Er stellt bei korrekter Beantwortung weniger Fragen oder bei Nicht-/Falschbeantwortung viele weitere Fragen. Er kann bei schlechter Gesinnung, also wenn seine Fragen nicht stets korrekt zeitnah beantwortet werden, seltener Angebote des Rechenzentrums annehmen oder bereits laufende Jobs widerrufen. In der Abgabeversion des Spiels übernimmt die Rolle der Kunden die Spielleitung.

Abbildung 7.3: Der Offer-Market

7.4.2 Offer Market

Der Offer-Market (vgl. Abb. 7.3), ein Marktplatz für Dienstleistungsangebote des Rechenzentrums, ist der Ort, an dem der Spieler Dienstleistungen im Namen des Rechenzentrums veröffentlichen kann. Diese sind im Allgemeinen äquivalent zu einem Job. Sie verfügen über dieselben Eigenschaften. Der zentrale Unterschied ist, dass an dieser Stelle der Spieler selbst alle konkreten Werte festlegt, beispielsweise die Bereitschaft **n** Rechenleistung über einen Zeitraum **t** für den Preis **p** zur Verfügung zu stellen.

7.4.3 Finanzübersicht

Um den Finanzfluss erfolgreich verfolgen zu können, steht dem Spieler eine spezielle Finanzübersicht zur Verfügung (vgl. Abb. 7.10). Hier lassen sich die Einkommen und Ausgaben pro Tick im Detail betrachten. Es wird deutlich, wie viele Wartungskosten gezahlt werden müssen. Außerdem lassen sich die geplanten Kosten des aktuellen Ticks analysieren.

7.4.4 Abhängigkeiten - Stromerzeugung und Kühlung

Ein Rechenzentrum besteht nicht nur aus Servern, die eigenständig problemlos funktionieren. Viel mehr ist es ein komplexes System mit einer Menge von Komponenten, deren Interaktion die Arbeit der Server erst ermöglicht. Diese Tatsache ist auch im Planspiel durch Abhängigkeiten zwischen Komponenten repräsentiert (vgl. Abb. 7.4).

Jeder Server erzeugt im Betrieb Hitze. Diese Hitze muss durch ein Kühlsystem vom Rechenzentrum abgeführt werden. Eine mangelhafte Kühlleistung führt zu einem *Throtteling*, einer verminderten Arbeitsleistung durch runtertaktende Prozessoren, bis zu einem totalen Ausfall der Server. Auf diesem Server laufende Jobs werden folglich nicht oder nicht ausreichend bearbeitet.

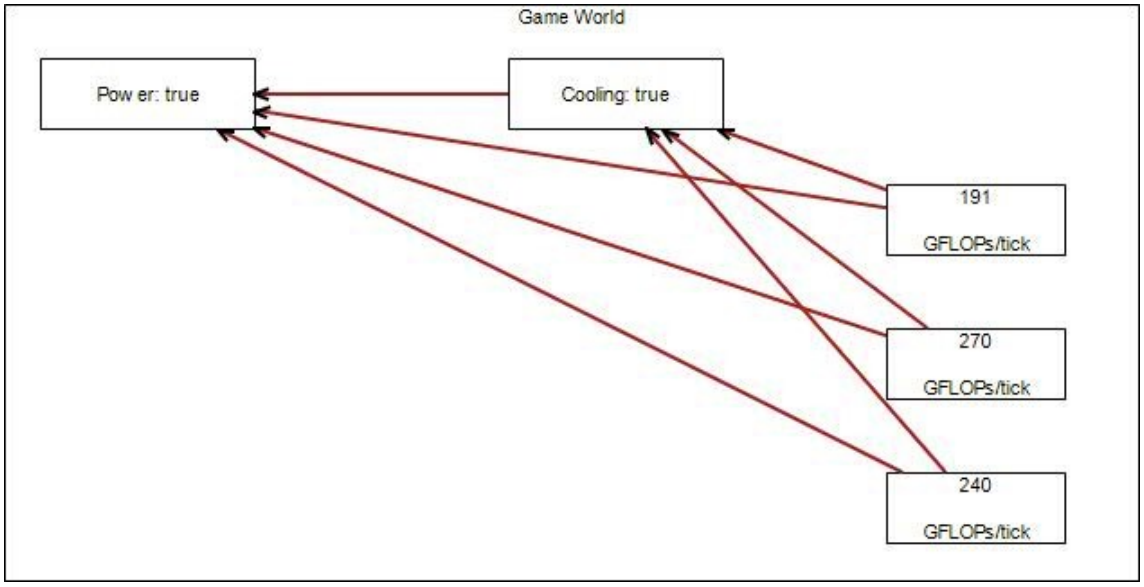

Abbildung 7.4: Abhängigkeiten zwischen einzelnen Modellelementen

Zusätzlich bestehen Abhängigkeiten vom Server sowie der Kühlung zum Stromnetz. Ein Ausfallen des Stromnetzes führt zu einem augenblicklichen Stillstand der Server und der Kühlung. Um den Totalausfall zu überbrücken oder zumindest zu verzögern wird jede dieser Komponenten an einen Power-Generator, ein Gerät zur Erzeugung von Strom durch beispielsweise Verbrennung von Diesel, angeschlossen.

Ein Power-Generator kann nur eine begrenzte Menge an Strom produzieren. Folglich kann er nur eine durch seine Leistung begrenzte Anzahl von abhängigen Komponente versorgen, die wiederum einen eigenen festen Bedarf an Strom besitzen. Dies bedeutet, dass eine korrekte Konfiguration der Abhängigkeitskomponenten von zentraler Bedeutung ist. Ein Überbeanspruchter Power-Generator führt zu einem Ausfall des Generators und vorher beschriebene Ereignisketten treten ein.

Jeder Generator bedarf eine feste Anzahl Treibstoff pro Tick um Strom zu produzieren. Treibstoff muss vom Rechenzentrum nachgekauft werden, wobei fehlender Treibstoff zum Ausfall des entsprechenden Generators führt.

Schließlich haben alle Elemente einen natürlichen Verschleiß, dem entgegengewirkt werden muss. Deswegen bedarf jede Komponente regelmäßiger Wartung ihrer markanten Merkmale. Ein Wartungsmangel kann zu einem Ausfall von Einzel- oder Gesamtkomponenten sowie verminderter Arbeitsleistung führen.

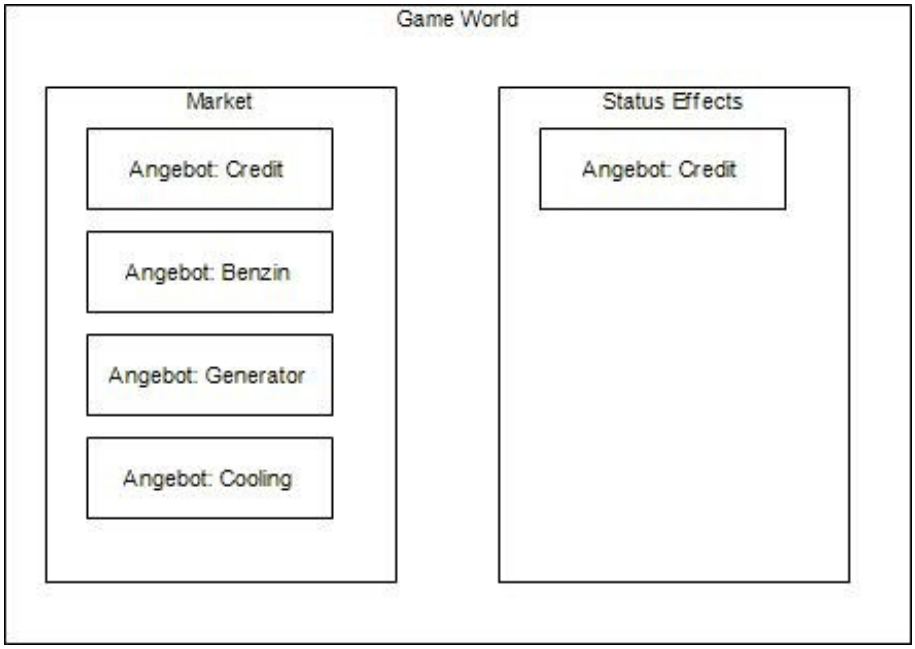

Abbildung 7.5: Market und Status-Effekte

7.4.5 Markt und Status-Effekte

Im Laufe des Spiels kann der Spieler Geld verdienen. Von diesem Geld werden Straf- und Wartungszahlungen abgeführt. Ferner kann und soll in den Ausbau des Rechenzentrums investiert werden, um finanziellen als auch strukturellen Wachstum zu erreichen.

Auf dem Markt können Rechenzentrum-Elemente gekauft werden, beispielsweise Generator, Kühlungseinheiten oder Server. Diese Instanz bietet dem Rechenzentrum zudem die Möglichkeit, ein Darlehen zu beanspruchen. Die Raten zum Darlehen müssen pro Tick in vorher genannter Höhe bezahlt werden.

Auf das Rechenzentrum können verschiedene Status-Effekte einwirken. Derzeit sind hier die Kredite aufgelistet, die der Spieler noch zurückzuzahlen hat (vgl. Abb. 7.5).

7.4.6 Rollen und Kommunikation

Obwohl das Krisenplan-Spiel theoretisch durch einen einzigen Spieler gespielt werden kann, ist es vorgesehen, dass die Herausforderungen durch ein Team gemeistert werden. Hierzu ist eine Verteilung der Spieler in verschiedene Rollen vorgesehen. Jede Rolle hat ihre eigene Verantwortung im Rechenzentrum und sieht entsprechend auch nur die für sie relevanten Informationen (vgl Abb. 7.6).

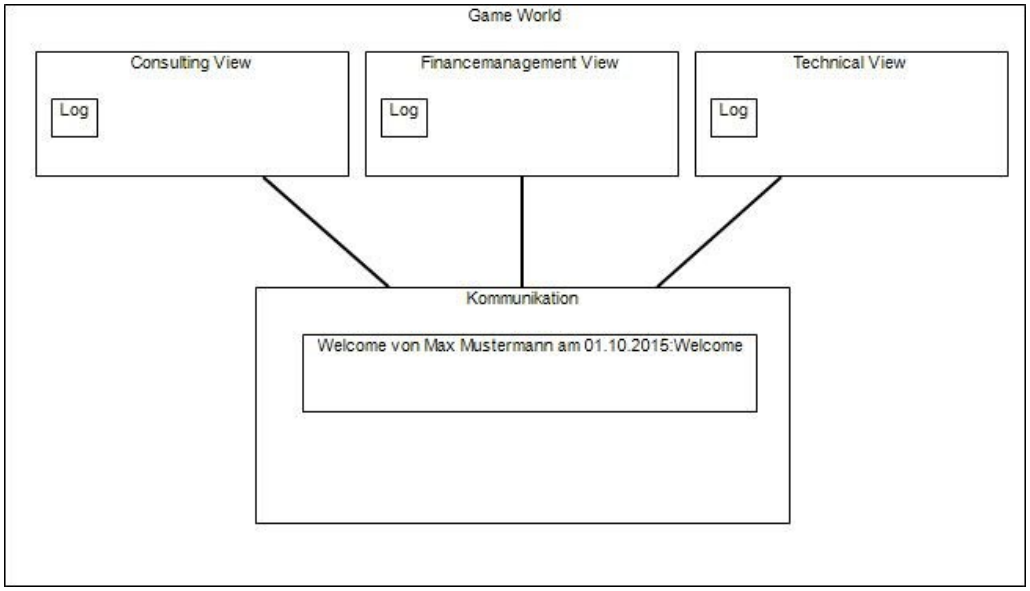

Abbildung 7.6: Rollen und Kommunikation

Jeder Rolle steht es frei, ihren eigenen Log zu führen. Diese kann sie durch eine Doppelklick-Aktion öffnen und für sie relevante Informationen für sich notieren. Dies kann wichtig sein, um z.B. Fragen von Kunden zu beantworten - eine schlechte, bzw. fehlende Dokumentation kann das Beantworten deutlich erschweren, wenn nicht gar unmöglich machen. Näheres zur Rollenverteilung im Modell ist in Sektion 7.6 aufgeführt.

Rollenübergreifende Aufgaben bedürfen erhöhter Kommunikation. Hierzu existiert ein gemeinsames *Flipboard*, eine Schreibtafel, das von allen Rollen eingesehen und editiert werden kann. Die dort stehenden Informationen sind völlig den Spielern überlasse. Es kann frei kommuniziert werden. Diese Entscheidung wird in 9.2 näher erläutert.

7.5 Zusammenfassung

Das Krisenplan-Modell besteht aus vielen Komponenten. Sie lassen sich jedoch im Allgemeinen in Kern und Features unterteilen. In Abbildung 7.7 ist das gesamte Krisenplan-Spiel mit allen Komponenten zu sehen.

Zum Kern gehören die Komponenten, die den zentralen, rudimentären Spielverlauf ermöglichen. Hierzu gehören vor allem Ticks, Balance, Job Market, Job, Service Center und Server. Sie ermöglichen dem Spieler das Analysieren, Annehmen und Abarbeiten von Jobs mit dem Ziel möglichst viel Geld zu verdienen.

Die Features sind Aspekte des Modells, die aus einer trivial erscheinenden Aufgabe eine echte Herausforderung für ein ganzes Team von Spielern gestalten. Hierzu gehören Kunden,

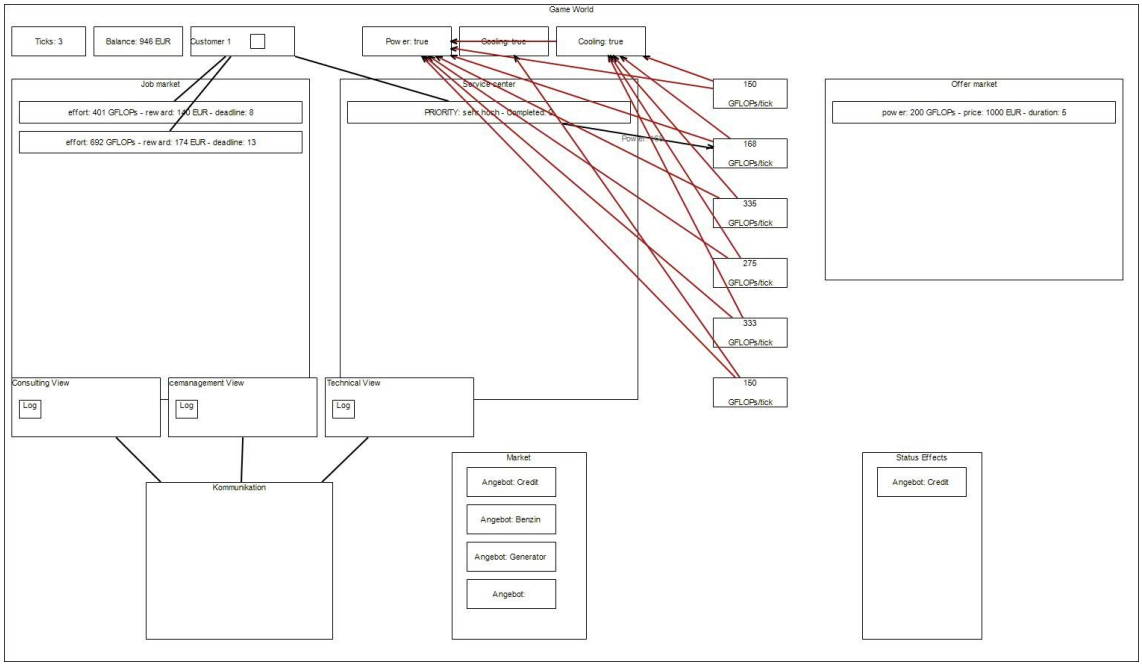

Abbildung 7.7: Gesamtes Modell

die für die Bereitstellung von Jobs verantwortlich sind und sich stets über den Fortschritt angenommener Jobs erkundigen. Der Offer-Market bietet dem Spieler die Möglichkeit eigene Vorschläge gegenüber Kunden zu machen. Durch die Abhängigkeiten zwischen verschiedenen Komponenten ist die Komplexität eines Rechenzentrums widergespiegelt.

Der Markt bietet dem Spieler zahlreiche Möglichkeiten sein Rechenzentrum auszubauen um auf Dauer noch höhere Einnahmen zu gewährleisten. Schließlich runden die verschiedenen Rollen mit den daraus entstehenden Kommunikationsproblemen die Komplexität des Spiels ab.

7.6 Rollenmodellierung

Das Betriebsmodell des Rechenzentrums besteht aus vielen verschiedenen Komponenten, wie etwa Jobmarkt oder Servicecenter. Jede Rolle kann jedoch nur auf einen begrenzten Abschnitt des gesamten Modells zugreifen, da nicht jede Rolle für alle Geschäftsbereiche zuständig ist. Diese Abgrenzung bedeutet jedoch nicht, dass alle Personen voneinander unabhängig sind. Besonders wichtig ist daher die Kommunikationskomponente, über die sich die einzelnen Spieler besprechen und Informationen weitergeben können.

7.6.1 Beratung

Zu den eigentlichen Aufgabenbereichen der einzelnen Rollen werden im zweiten Abschnitt der Projektgruppenphase entsprechende Views erstellt, welche zur Umsetzung im eigentlichen Planspiel dienen.

Zur Simulation von Rückfragen auf Seiten des Kundens werden so dessen Fragen angezeigt, welche idealerweise korrekt beantwortet werden müssen.

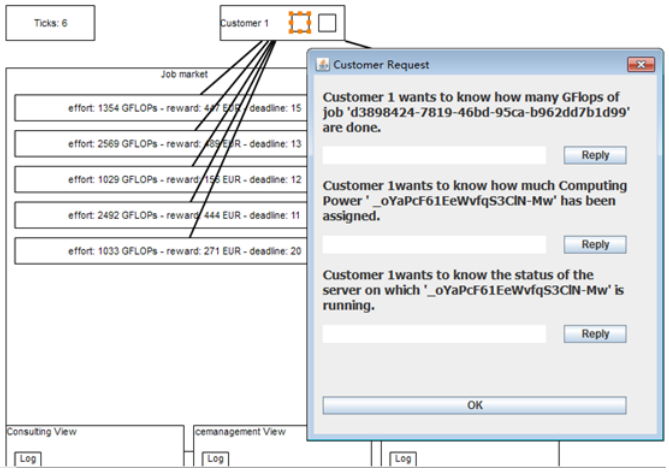

Abbildung 7.8: Berater

Außerdem steht es dem Berater und auch seinen Kollegen frei, eine Verlaufshistorie (auch Log genannt) mit Informationen zu führen. Darin können wichtige Informationen notiert werden, wie etwa die Anforderungen der Kunden sowie der aktuelle Status der Geräte.

7.6.2 Technik

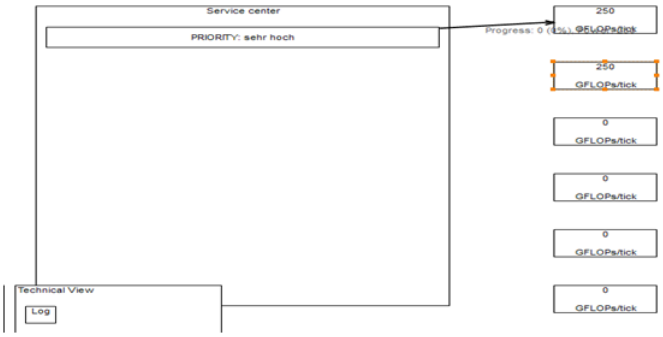

Abbildung 7.9: Fortschritt Techniker

Abbildung 7.9 zeigt die Ansicht des Technikers. Diese bietet eine Zustandübersicht der verschiedenen Server, sodass offensichtlich nicht funktionsfähige Maschinen direkt ersichtlich sind.

7.6.3 Finanz

Die Finanz-Rolle entspricht eher einem „Finanzberater“ für die anderen Mitspieler. Diese Rolle beobachtet den aktuellen Saldo des Rechenzentrums und benachrichtigt rechtzeitig alle anderen. Abbildung 7.10 illustriert die Finzanzübersicht, die dieser Rolle vorbehalten ist.

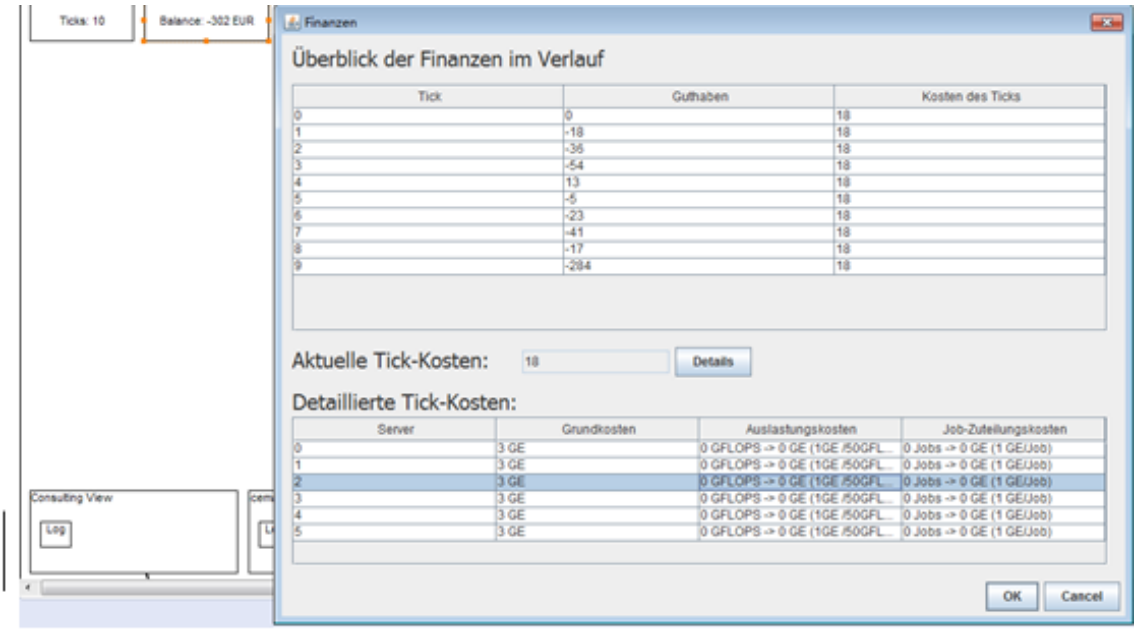

Abbildung 7.10: Finanzübersicht

Hierdurch lassen sich die Einzelheiten des Finanzflusses betrachten. Der Finanzverwalter kann seine weiteren Aktionen durch Analyse des Finanzflusses planen. Bei der Bewältigung von Krisensituation lässt sich eine vollständige Dokumentation von Finanzaktionen nicht umgehen und das Finanzpersonal ist demnach unverzichtbar.

7.6.4 Grafische Oberfläche

Um das Planspiel schließlich durch eine Web-Applikation zu realisieren, entwerfen wir für jede Rollenansicht eine grafische Oberfläche, die benutzerfreundlich und einfach anzuwenden ist, wodurch jedes Mitglied seine eigene Aufgabe erledigen kann.

7.6.5 Kunde

Das User-Interface variiert je nach Rolle der Benutzer. Wenn der Benutzer als Kunde angemeldet ist, steht ihm eine wie in Abbildung 7.11 illustrierte Ansicht zur Verfügung. In dem Fenster, welches oben links liegt, sieht der Kunde alle von ihm angebotenen Aufträge. Weitere Details können durch einen Klick eingesehen werden. Wenn ein Auftrag akzeptiert wird, wird er danach im rechts stehenden Fenster angezeigt. Außerdem wird ein Chat-Fenster für alle Rollen am unteren Rand der Seite zur Verfügung gestellt, da die Kommunikation von allen Mitgliedern eine Kerntätigkeit im Planspiel ist. Hier kann der Kunde rechtzeitig mit den Kundenberatern kommunizieren und Fragen, die er hat, stellen.

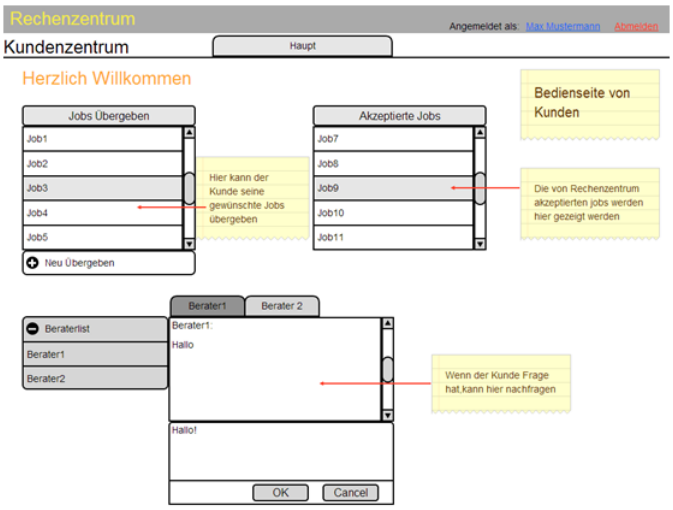

Abbildung 7.11: Kundenansicht

7.6.6 Beratung

Nachdem eine Anmeldung als Berater erfolgt ist, stehen grundsätzlich zwei Ansichten zur Verfügung. Die erste Ansicht stellt die Kundenaufträge dar. Im links oben liegenden Fenster werden alle Kundennamen sowie die zugehörigen Aufträge aufgelistet. Der Berater soll zuerst mit dem Kunden über die Kundenanforderungen diskutieren. Wenn sie einen Konsens erreichen, wählt der Berater den entsprechenden Auftrag aus und tätigt nötige Konfigurationen. Das Fenster in der Mitte (vgl. 7.12) zeigt alle relevanten Informationen des ausgewählten Auftrags. Der Berater kann die Priorität des Auftrags setzen sowie dem Techniker wichtige Nachrichten bei der Auftragsannahme hinterlassen. Der akzeptierte Auftrag wird dann ins Servicecenter an Techniker weitergeleitet. Die zweite Ansicht zeigt, analog zu den anderen Ansichten der anderen Rollen, die Kommunikations- und Logansicht.

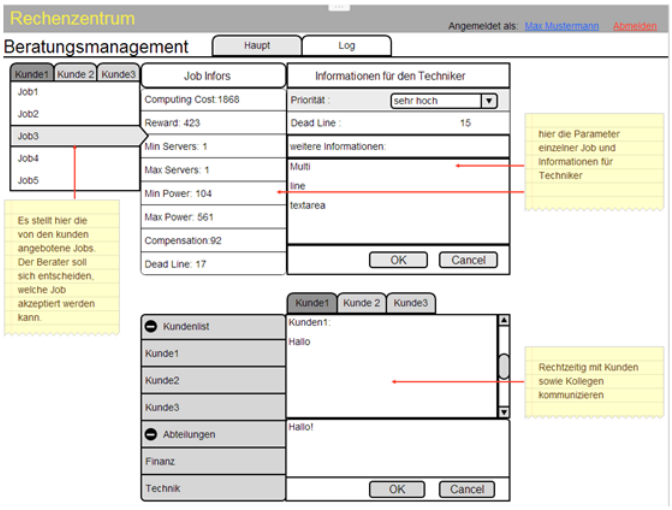

Abbildung 7.12: Auftragsdetailsansicht

Wie in Abbildung 7.13 gezeigt, haben der Berater und auch andere Mitglieder wie Techniker und Finanzverwalter, die Möglichkeit einen Historien-Eintrag (Log-Eintrag) zu erstellen. Diese sind chronologisch sortiert. Es steht jedem Mitglied frei, die alten Einträge zu bearbeiten oder neue Einträge zu verfassen.

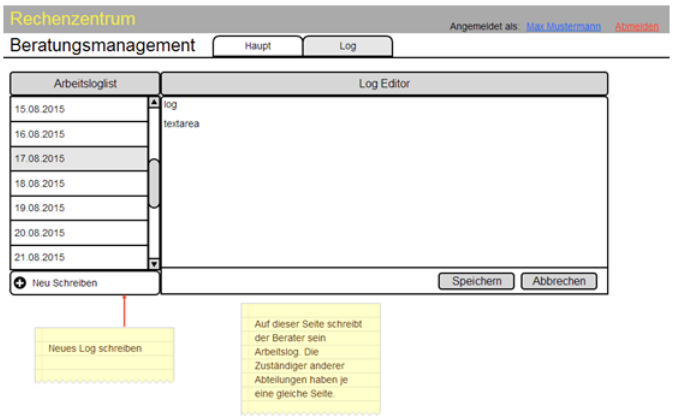

Abbildung 7.13: Logansicht

Techniker

Der Techniker beschäftigt sich mit der Wartung der Geräte und der Zuweisung der Aufträge an konkrete Server. Wie in Abbildung 7.14 zu sehen, werden alle Server aufgelistet. Wenn ein Server ausgewählt ist, lassen sich dann die relevanten Informationen des Servers

und die auf dem Server laufenden Aufträge betrachten. Außerdem wird ein Zuverlässigkeitsgraph des Servers generiert und rechts daneben angezeigt.

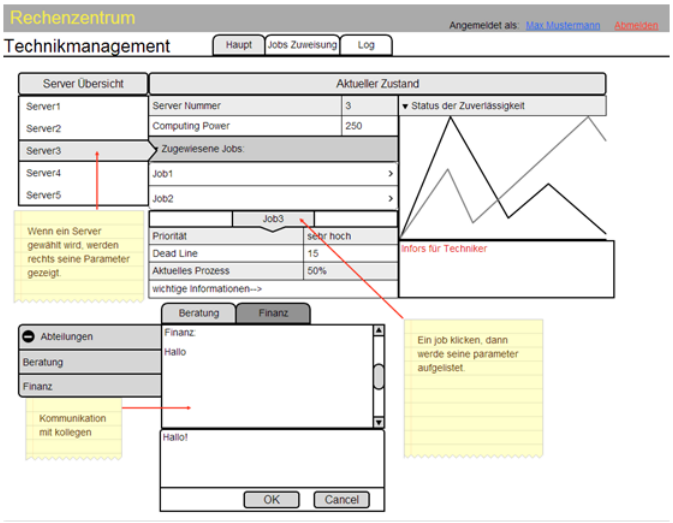

Abbildung 7.14: Serveransicht

Die angenommenen Aufträge werden an Techniker verschickt und schließlich auf der Seite „Jobs Zuweisung“ aufgelistet. Der Techniker weist hier den Servern die Aufträge zu und beobachtet rechtzeitig den Status bereits zugewiesener Aufträge.

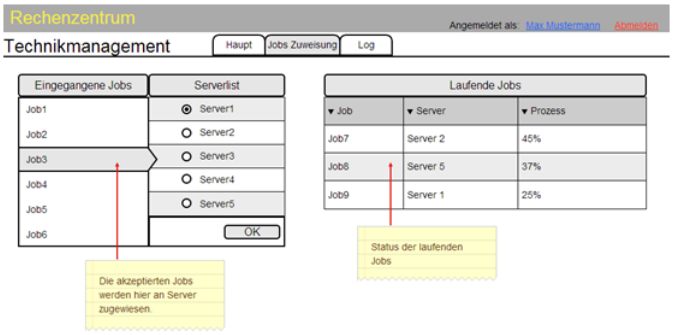

Abbildung 7.15: Zuweisungs- und Statusansicht

7.6.7 Finanzverwalter

Am oberen linken Rand der Finanzmanagementansicht lässt sich das aktuelle Guthaben des Rechenzentrums einsehen. In der Mitte der Seite werden chronologisch alle getätigten Ein- und Ausgaben sowie weitere Details aufgelistet. Dadurch ist z.B. erkennbar, wie sich das pro Tag verändert hat. Nach einem Klick auf eine der Einträge können detaillierte

Informationen eingesehen werden. Auf diese Weise lassen sich die Finanzen des Rechenzentrums systematisch nachverfolgen und einfach verwalten.

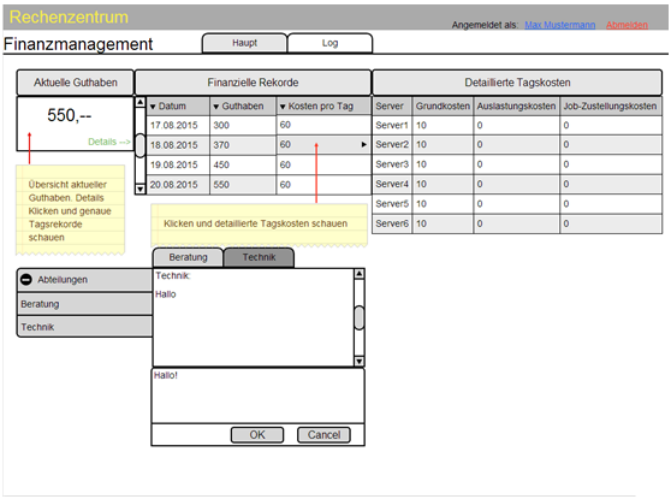

Abbildung 7.16: Finanzübersicht

7.7 Leveleditor

Der in Kapitel 9.1 erwähnte Spielablauf beschreibt den groben Verlauf des Spiels. Das Spiel umfasst unterschiedliche Phasen, die der Spieler während der Simulation durchlaufen muss. Die Verwaltung dieser Phasen obliegt dem Spielleiter. Dieser muss darüber entscheiden, in welchen Abschnitten der Simulation die Spieler welche Aufgaben (vgl. Kapitel 9.3), Rätsel oder andere Schwierigkeiten bewältigen müssen, um das Spiel erfolgreich abzuschließen. Das spontane Eingreifen in den Verlauf des Spiels und die in Kapitel 6.6 erläuterte Vermittlung von Lerninhalten dienen als Basis für dieses Kapitel, in welchem die Anforderungen an die durch den Spielleiter entworfenen Phasen (fortan auch Level genannt), die zugrunde liegende Logik und die Methodiken, mit welchen das Level und somit der Editor für den Leiter, im Spiel realisiert wird, erläutert werden. Dabei ist zu beachten, dass der Leveleditor in seiner Struktur und Umsetzung nur während der Arbeiten mit dem Software-Werkzeug Cinco benutzt wird. Die Web-Applikation verwendet ein Optionenmenü zur Verwaltung möglicher Spielkomponenten. Die Anforderungen an die Anpassbarkeit während des Verlaufs sind dennoch identisch und deshalb werden im Folgenden Anforderungen und die Struktur des Leveleditors während der Modellierungsphase mit Cinco beschrieben.

7.7.1 Anforderungen

Es existieren zwei wichtige Anforderungen an den Levelleditor, damit die Lernziele effizient verfolgt und erreicht werden können.

Das Level, als auch die unterschiedlichen Phasen, aus welchen ein Level besteht, sollen wiederholbar sein, um Erfolg und Misserfolg zwischen einzelnen Spielergruppen vergleichen zu können.
Eine weitere Anforderung stellen dynamische Veränderungen während des Spielverlaufs dar, um dem Spielleiter die Möglichkeit zu gewähren, die Spieler für fachlich sinnvolle Entscheidungen zu belohnen. Außerdem soll dadurch die Möglichkeit entstehen, die Spielschwierigkeit in unterschiedlichen Phasen anzuheben. Eine Anhebung der Schwierigkeit könnte sich z.B. auf den Stressfaktor, den die Spieler einer Gruppe erfahren, auswirken.

Ein Beispiel für eines der in der Schwierigkeit linear ansteigenden Szenarien wird durch folgenden grob skizzierten Ablauf illustriert:

- Der Beginn des Spiels ist einfach, viel Belohnung für wenig Arbeit, um den Spielern Vertrauen in ihre Handlungen zu geben und erste Abläufe zu trainieren.
- Anhebung des Schwierigkeitsgrades. Die Spieler müssen eventuell mehr Rätsel lösen, um an relevante Informationen zu gelangen.
- Stetiger Anstieg der Schwierigkeit, da die Spieler in dieser Phase möglicherweise Handlungsmuster entwickelt haben, um Probleme zu bewältigen.

Um die Anforderungen zu erfüllen und dem Spielleiter genug Freiheit bei der Gestaltung eines Levels zu gewähren, wird im nachfolgenden Abschnitt der Level-Builder der PG592 beschrieben.

7.7.2 Aufbau und Struktur

Die Modellierung des Levelleditors, bzw. eines Szenarios ist unabhängig von den in Cinco modellierten Entitäten. Diese Entscheidung wurde bewusst getroffen, um bei der Überführung in DyWA und die damit einhergehende Erstellung eines grafischen Levelleditors keine Abhängigkeiten im Kontrollfluss zu haben. Die auszuführenden Methoden des Grundleveleditors sind leicht erweiter- und anpassbar. Der nachfolgende Levelleditor stellt demnach nur das Modell dar, welches dem späteren grafischen Editor zugrunde liegen wird und aus welchem die Level generiert und einer Datenbank persistiert werden können. Aus diesem Grund der leichten Erweiterbarkeit besteht das Modell des aktuellen Levelleditors nur aus vier Kernklassen:

- LevelBuilder: mithilfe dieser Klasse, kann ein Level erzeugt werden. Details dazu befinden sich im Abschnitt 7.7.3
- Level: die Datenklasse, welche alle Informationen enthält. Dazu gehören unter anderem alle möglichen Attribute, die der Spielleiter während des Spielverlaufs verändern darf.
- LevelPhase: Eine Datenklasse, um eine Phase innerhalb des Spiels darzustellen.
- LevelAttributeModifier: eine Funktionsklasse, die das dynamische Verändern von Attributen im Spielverlauf erlaubt.

Das in Abbildung 7.17 illustrierte Klassendiagramm zeigt außerdem Operationen, die auf den Klassen ausgeführt werden können. Eine detaillierte Beschreibung der Methoden und der Funktionsweise befindet sich im Abschnitt 7.7.3.

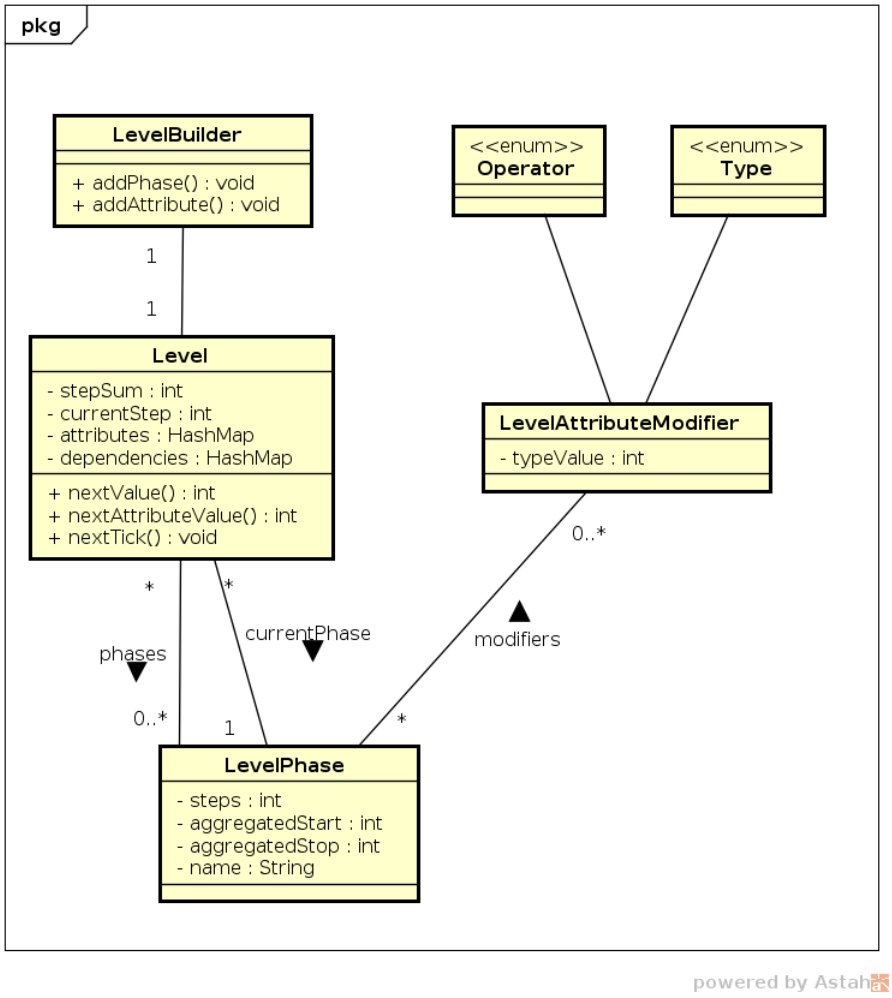

Abbildung 7.17: Klassendiagramm des Leveleditors

Zusammenfassend betrachtet ist der Leveleditor datenorientiert. Vor oder während des Spiels wird es dem Spielleiter so möglich sein im grafischen Editor der DyWA Änderungen einzufügen, die direkten Einfluss auf den weiteren Verlauf haben. Diese grafische Oberfläche wird die o.g. Entitäten benutzen, um das Level schrittweise aufzubauen oder möglicherweise zu verändern.

7.7.3 Funktionsweise

Die Funktionsweise des Level-Builders aus der Modellierungsphase mit Cinco lässt sich anhand eines Beispiels erklären:

```
1  public LevelBuilder() {
2    level = new Level();
3    addAttribute(Constants.LevelAttributes.JOB_REWARD, Constants.
     JOB_REWARD_MIN, Constants.JOB_REWARD_MAX);
4    addAttribute(Constants.LevelAttributes.SERVER_POWER, Constants.
     SERVER_POWER_MIN, Constants.SERVER_POWER_MAX);
5
6    Function<Integer, Integer> compensationReward = new Function<Integer,
     Integer>() {
7      @Override
8      public Integer apply(Integer foreign) {
9        return (int) (foreign/20);
10     }
11   };
12   addAttribute(Constants.LevelAttributes.JOB_COMPENSATION, Constants.
     LevelAttributes.JOB_REWARD, compensationReward);
13   // ...
14 }
15
```

Abbildung 7.18: Funktionsweise Leveleditor Schritt 1

Zunächst muss ein neues Level erstellt werden und alle später verfügbaren Attribute werden in diesem Level registriert. Zusätzlich ist es möglich, Abhängigkeiten von Attributen anzugeben. Die Methode zur Erstellung einer Abhängigkeit erwartet ein Function-Objekt, welches die Abhängigkeit abbildet. In dem oben gezeigten Bild hat `JOBCOMPENSATION` immer den den Wert von `JOB REWARD/20`.

```
1    // ...
2    LevelPhase first = new LevelPhase("erste Phase", 100);
3    LevelPhase second = new LevelPhase("zweite Phase", 100);
4    LevelPhase third = new LevelPhase("dritte Phase", 100);
5    // ...
6
```

Abbildung 7.19: Funktionsweise Leveleditor Schritt 2

Im nächsten Schritt müssen Phasen für das Level erstellt werden, indem ein Name und eine Anzahl der Schritte angegeben wird. Jeder dieser Schritte stellt einen Tick dar. Damit das Level mit den Ticks des Spiels synchron ist, stellt der Leveleditor die Funktion *nextTick* bereit, die nach Abschluss eines Ticks (vgl. Kapitel 9.1), aufgerufen werden muss.

Der letzte Schritt beinhaltet das Hinzufügen der sogenannten AttributeModifiern innerhalb von bereits angelegten Phasen. Diese verändern während des Spielverlaufs automatisch Attribute bzw. Abhängigkeiten nach festgelegten Mustern. In dem Beispiel (vgl. Abbildung 7.20) würden die Attribute `JOB REWARD` und `JOB COMPENSATION` in der dritten Phase zusätzlich einen Wert von `+1000%` erhalten.

```
1       // ...
2       LevelAttributeModifier add1000Percent = new LevelAttributeModifier();
3       add1000Percent.setOperator(Operator.PLUS);
4       add1000Percent.setType(Type.PERCENT);
5       add1000Percent.setTypeValue(1000);
6       third.addModifier(Constants.LevelAttributes.JOB_REWARD, add1000Percent
    );
7       third.addModifier(Constants.LevelAttributes.JOB_COMPENSATION,
    add1000Percent);
8
9       level.addPhase(first);
10      level.addPhase(second);
11      level.addPhase(third);
12    }
13
```

Abbildung 7.20: Funktionsweise Leveleditor Schritt 3

Abschließend müssen die Phasen dem Level hinzugefügt werden, bevor das Level dem Spiel hinzugefügt werden kann.

7.7.4 Verwendete Techniken

Der Level-Builder verwendet die Bibliothek uncommon-maths (`http://maths.uncommons.org`), um alle Attributwerte auch nach Abschluss eines Spiels für andere Gruppen wiederholbar zu machen, indem der Levelgestalter einen *seed* für das Erstellen der zufälligen Zahlen festlegen kann. Bei gleichem *seed* sind auch alle Werte reproduzierbar.

Neben der verwendeten Bibliothek benutzt der Level-Builder außerdem die seit Java 8 verfügbare Klasse Functions (`https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html`), um Funktionen bei z.B. Abhängigkeiten zwischen Attributen zu realisieren. Weitere spezielle Techniken und die Verwendung von spezifischen Metriken innerhalb des Spiels werden im nachfolgenden Kapitel thematisiert.

7.7.5 Übertragung in Web-Applikation

Die Übertragung des beschriebenen Leveleditors aus der Modellierungsphase mit Cinco in die Web-Applikation erfolgte nicht, da zum gegebenen Zeitpunkt das Migrieren von Logik in Form von reinem Java-Code nicht möglich ist. Zudem wurde im Rahmen der Projektgruppe entschieden, dass kein weiteres Cinco-Product für die Level-Modellierung erstellt wird. Stattdessen beinhaltet die Web-Anwendung das bereits erwähnte und mit nativen DyWA-Boardmitteln gebaute Optionenmenü. Dort ist es grafisch möglich, Veränderungen durch das Modifizieren von einzelnen Variablen während des Spiels vorzunehmen. So kann der Spielleiter im Spielverlauf beispielsweise die Belohnungen für abgeschlossene Aufträge erhöhen oder die Strafen bei Nichteinhaltung von Fristen verringern. Es existiert außerdem die Möglichkeit, das Spiel zu beschleunigen, zu verlangsamen, zu starten, zu stoppen oder zurückzusetzen (vgl. Ticksystem und `TimerButtonComponent`). Im Gegensatz zur ursprünglich geplanten Einteilung des Spielverlaufs in strikte Phasen durch den beschriebenen externen Leveleditor, kann der Spielleiter mithilfe des Optionenmenüs Änderungen agiler vornehmen und händisch Notizen erstellen, damit Spielverläufe reproduzierbar sind. Die Beziehung zu den in Abschnitt 7.7.3 beschriebenen Variablen wie beispielsweise `JOB COMPENSATION` ist offensichtlich. In der Web-Applikation ist dieses Konzept durch Vereinfachung der Phasen und der Handhabung während des Spiel nur dynamischer gestaltet.

Kapitel 8

Implementierungsphase

Der Modellierungsphase folgt die Implementierungsphase, in der aus dem Modell eine Datenstruktur abgeleitet wird, auf der die zukünftige Anwendung arbeiten soll. Als weiteres Prototyping-Tool zur Herstellung des finalen Produkts wird DyWA genutzt.

8.1 Arbeiten im Team

Damit die Projektgruppe die Implementierung der Webanwendung als Team durchführen kann, wird der Arbeitsprozess auf einen externen Server auslagert. Der Server ist eine virtuelle Maschine, auf der Ubuntu, WildFly, PostgreSQL, Jenkins, ein Apache und einige weitere in Abhängigkeit stehende Software installiert. Die grundlegende Einrichtung des WildFlys sowie der Datenbank folgt analog dem Schema der Installation auf einem regulären Client. Die von WildFly bereitgestellten Services sind normalerweise nur über eine lokale Adresse erreichbar, weshalb der auf dem Server laufende WildFly zur Erreichbarkeit von Außen konfiguriert und entsprechend geschützt wird. Zur allgemeinen Nachvollziehbarkeit, zur besseren Diagnosemöglichkeit und zur Datensicherheit werden *Audit Trigger* eingerichtet, die jede in der Datenbank ausgeführte Abfrage protokollieren und Schritt-für-Schritt nachvollziehbar machen. Da der Toolstack ein häufiges *Builden* erfordert, wird eine Instanz von Jenkins auf dem Server aufgesetzt und mit verschiedenen Build-Prozeduren ausgestattet. Der Jenkins sorgt dafür, dass vor jedem Build-Prozess alle Instanzen der DyWA nicht mehr erreichbar sind, führt ein Datenbankbackup durch, erstellt alle Komponenten und fügt sie erneut dem WildFly hinzu. Das Unerreichbarmachen hat u.A. den Vorteil, dass andere Projektgruppenmitglieder während eines Build-Vorgangs keinen Fehler erzeugen. Trotz der externen Webanwendung muss die gesamte Arbeitsprozedur in der PG jedoch koordiniert werden, damit nicht mit falschen Datensätzen gearbeitet wird. Um möglichst wenig Behinderungen auf dem Server zu erzeugen werden Aktionen, die das Da-

tenbankschema verändern, in jedem Fall auf einer lokalen Maschine mit Datenbankkopie getestet.

8.2 Erstellen der Datenstruktur

Ein zusätzliches Kernziel der Modellierungsphase ist es, die modellierten Datenstrukturen und Relationen in das finale Produkt überführen zu können, ohne dass diese erneut und ohne Mehraufwand angelegt werden müssen. Dabei werden die Datenstrukturen genau so in ein Datenbanksystem überführt, wie sie im Cinco-Produkt modelliert wurden. Insbesondere hat deshalb jedes Objekt die modellierten Attribute.

8.2.1 Pyro

Das Werkzeug Pyro importiert die Datenstrukturen eines Cinco-Produkts automatisch in eine für die DyWA verwertbare Form. Es ermöglicht zudem während des Imports Modifikationen am Modell vorzunehmen.

8.2.2 Krisenplan Connector

Da Pyro sich noch in einem frühen Entwicklungsstadium befindet, nutzt die Projektgruppe den *Krisenplan Connector* um die im Cinco-Produkt ermittelten Datenstrukturen zur DyWA zu überführen, wobei sich der Connector wie eine klassische ORM-Datenbankerstellung benutzen lässt.

8.3 Prozessmodellierung

Nachdem die Datenstrukturen importiert sind, kann mit der Prozessmodellierung fortgefahren werden. Die Prozessmodellierung erfolgt mit dem bereits vorgestellten Tool jABC4. Dabei werden Prozesse und Workflows modelliert, die in direkter Abhängigkeit zu den importierten Datenstrukturen stehen. Wie diese Prozesse nun konkret auszusehen haben, hängt von den Anforderungen des finalen Produkts ab. Jede Handlung bzw. jede Aktion, die im Produkt durchgeführt werden können soll, wird als Prozess modelliert.

Es besteht jedoch die Möglichkeit komplexes Prozessverhalten außerhalb von jABC4 auszulagern, indem man die Logik in pure Java-Methoden in POJOs auslagert und diese danach in das jABC4 importiert. Das kann im extrem zu sehr kleinen Prozessen füh-

ren. Daher gilt es den richtigen Abstraktionsgrad zu finden, der den richtigen Trade-Off zwischen Aussagekraft und Überschaubarkeit/Verständlichkeit der Modelle bietet.

Für das Krisenplanspiel wurde eine Vielzahl von Prozessen modelliert. Dazu gehören Initialisierungsprozesse, die das Spiel auf einen initialen Stand bringen, Spielhandlungen wie das Reparieren von Strom- und Kühleinheiten oder Prozesse, die für das korrekte Funktionieren der Spieloberfläche notwendig sind.

8.3.1 Prozessüberblick

Ein Spiel zeichnet sich durch seine Handlungsvielfalt aus. Dies macht sich insbesondere in der Quantität der Prozesse des Krisenplanspiels bemerkbar. Im Anhang befindet sich eine kurze Übersicht über die entwickelten Prozesse. Im Laufe der Ausarbeitung werden einige für das Planspiel besonders charakteristische Prozesse detailliert erläutert.

Die Prozesse im Anhang sind durch kurze Ein-Satz-Erklärungen beschrieben, da es zu viele sind, um auf alle im Detail einzugehen. Um nachfolgenden Projektgruppen, die das entstandene Krisenplan-Spiel aufgreifen und es auf ein neues Niveau bringen wollen, dennoch den Einstieg zu erleichtern, werden die Prozesse in diesem Format erläutert.

8.3.2 Spielwelt initialisieren

Der Prozess *initGameWorld* (vgl. Abb. 8.1) ist für die erste Initialisierung der Spielwelt sowie all der Spielkomponenten zuständig.

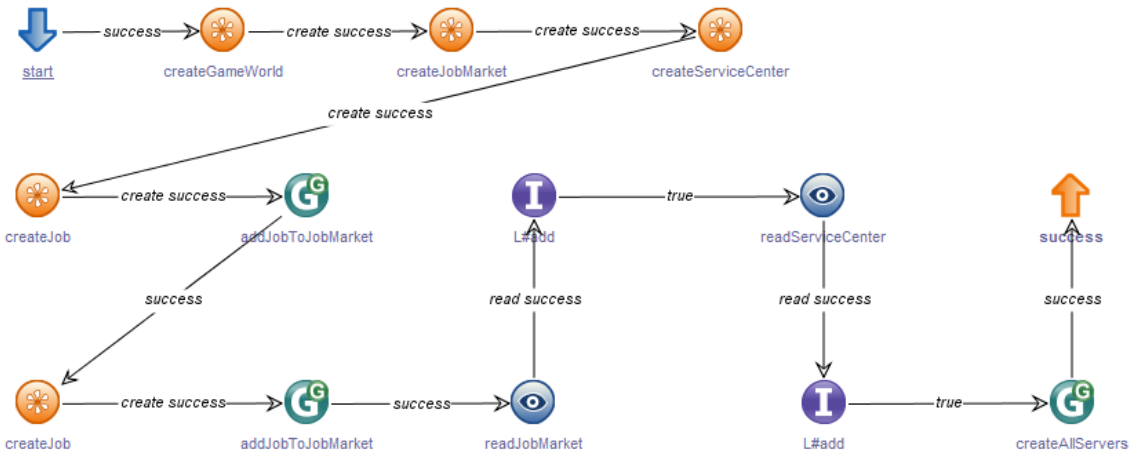

Abbildung 8.1: *initGameWorld*: Spielwelt initialisieren

Der Prozess erstellt die Kernkomponenten des Spiels: Die Spielumgebung, zu der alle weiteren Komponenten gehören, der JobMarket, in dem alle Jobs gelistet sind, das ServiceCen-

ter, dass Jobs aus dem JobMarket buchen kann. Bei Erfolg werden weitere Spielelemente in diese Kernkomponenten hinzugefügt: Jobs dem JobMarket, zum ServiceCenter gehörige Server.

8.3.3 Server kaufen

Der Spieler hat die Möglichkeit, das Spiel durch den Kauf von zusätzlichen Servern zu beeinflussen (vgl. Abb. 8.2). Ein Server ist eine Spielkomponente, die direkt mit Jobs interagiert. Der Server hat einige wesentliche Charakteristika:

- *computingPower*: Rechenkraft des Servers
- *name*: Name des Servers
- *neededPower*: Benötigter Stromwert, damit der Server funktioniert
- *reliability*: Wahrscheinlichkeit, dass der Server nicht ausfällt
- *working*: Aktueller Funktionszustand

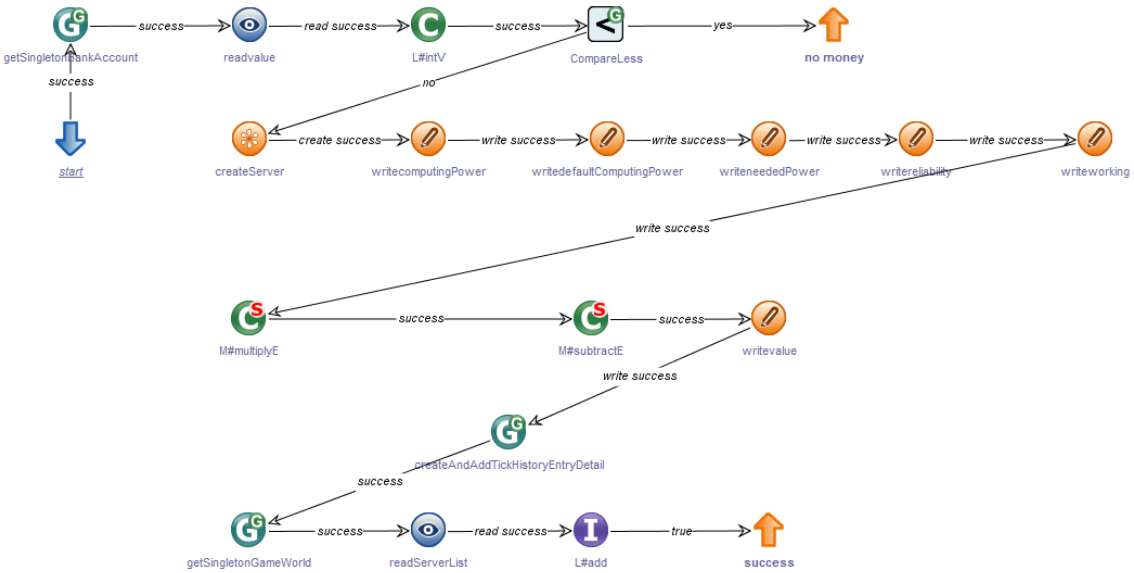

Abbildung 8.2: *buyServer*: Server kaufen

Der Kauf des Servers erfolgt durch Initiierung des Spielers. Dabei kann er die Rechenkraft seines zukünftigen Servers frei angeben, muss aber pro Rechenkraft-Einheit zusätzliches Geld zahlen. Je leistungsfähiger ein Server ist, desto teurer ist er auch. Der Prozess überprüft, ob das Spielgeldkonto ausreichend gedeckt ist. Bei ausreichender Deckung wird der gewählte Server gekauft, mit den Kennzahlen ausgestattet und dem ServiceCenter des

Spielers zugewiesen. Der Spieler kann von nun an frei über den Server verfügen und diesem Jobs zur Abarbeitung zuweisen.

8.3.4 Stromgenerator reparieren

Im Laufe des Spielgeschehens kann es vorkommen, dass der Generator nicht mehr funktionsfähig ist. In so einem Fall muss er repariert werden, was durch den in Abbildung 8.3 dargestellten Prozess realisiert wird.

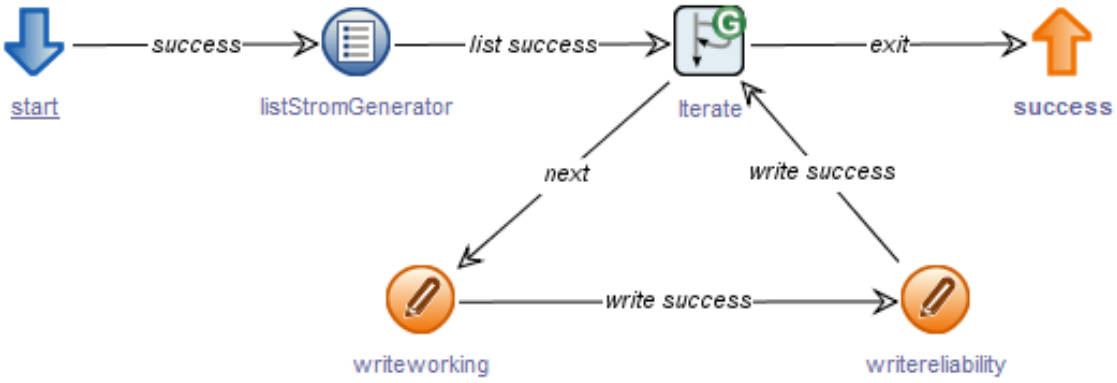

Abbildung 8.3: *repairGenerator*: Setzt alle Generatoren auf einen funktionsfähigen Zustand

Seitens des Prozesses wird dies simpel realisiert, indem die Reliabilität des Generators auf den maximalen Wert zurückgesetzt wird, sowie der Generator selbst als funktionsfähig markiert wird. Die Komplexität und die Herausforderung für den Spieler wird an dieser Stelle seitens der Oberfläche erzeugt, indem er gezwungen wird ein Mini-Rätsel(vgl. Sektion 9.3) zu lösen, bevor er den Prozess tatsächlich ausführen darf.

8.3.5 DoTick-Prozesse

Da die Zeit im Planspiel diskret ist, gibt es zu jedem zeitlichen Fortschritt einen Prozess, der eine Vielzahl von Aufgaben erledigt. Dieser Prozess ist der *DoTick*-Prozess (vgl. Abb. 8.4).

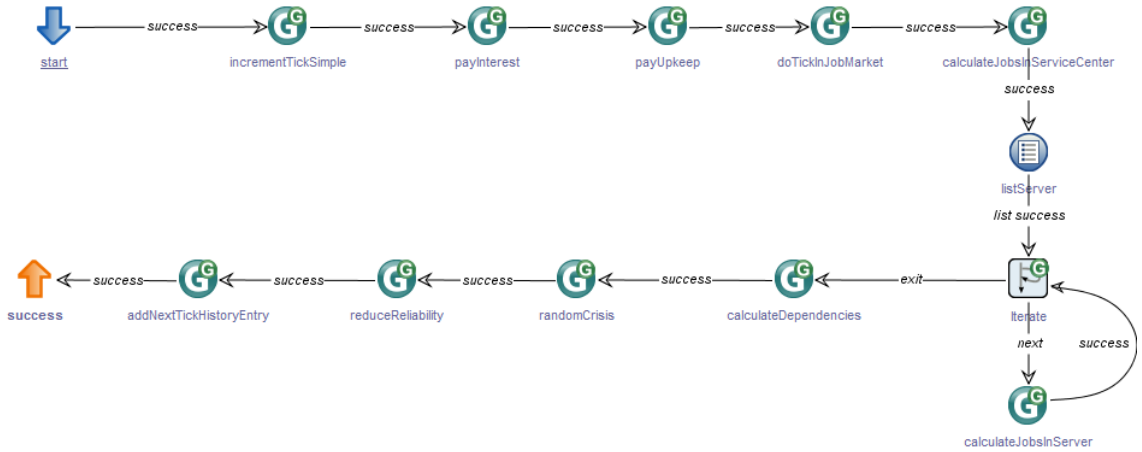

Abbildung 8.4: *doTick*: Tick ausführen

Dabei wird zunächst der aktuelle Tick inkrementiert. Danach werden Zinsen vom Bankkonto des Rechenzentrums abgezogen, sofern es sich im finanziellen Verzug befindet. Außerdem wird für die Aufrechterhaltung der Komponenten gezahlt. Änderungen des Bankkontos werden geloggt und dem Finanzverwalter im Spiel zur Verfügung gestellt.

In jedem Tick werden die Zustände der sich im ServiceCenter befindenden Jobs betrachtet. Erledigte Jobs bzw. Jobs mit versäumter Deadline werden entsprechend markiert, der Kontostand dahingehend angepasst. Mit festgelegten Wahrscheinlichkeiten können Komponenten des Rechenzentrums ausfallen. Es wird geprüft, ob vorher ausgefallene Komponenten wieder instand gesetzt wurden und welche Auswirkungen diese Reparaturen für Folgekomponenten haben.

Im folgenden werden einige Unterprozesse des DoTick-Prozesses detailliert vorgestellt:

Stromgenerator-Status berechnen

In jedem Tick wird geprüft, ob der Stromgenerator noch funktioniert. Ohne Strom funktionieren Server nicht und entsprechend können Jobs nicht bearbeitet werden. Der Stromgenerator bedarf jedoch Treibstoff pro Tick, da er ansonsten in einen nicht funktionsfähigen Zustand wechselt. Das Prüfen dieser Voraussetzungen findet im in der Abbildung 8.5 dargestellten Prozess statt.

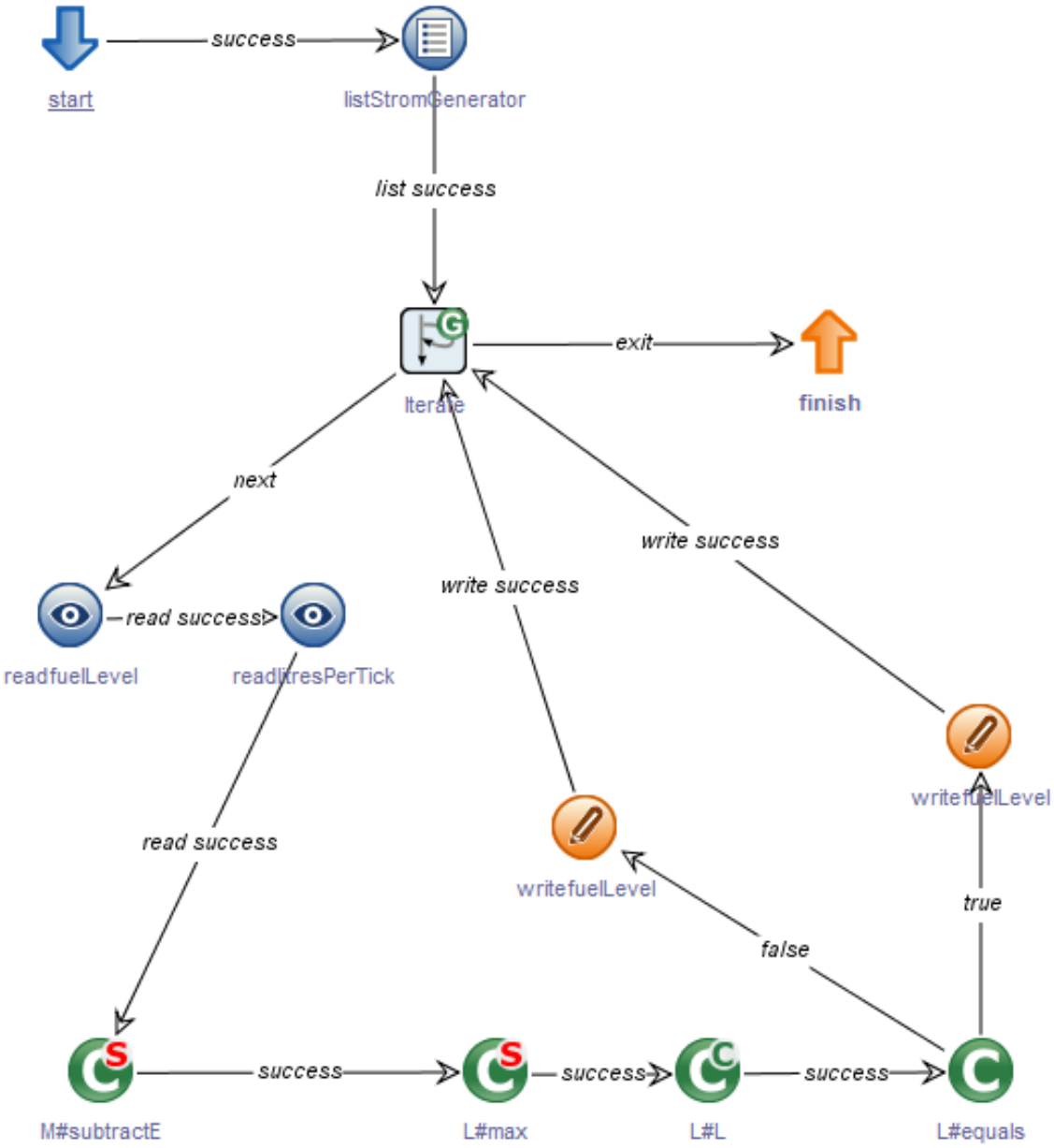

Abbildung 8.5: *calcGenerator*: Generator-Status berechnen

Job Status im Server berechnen

In jedem Tick werden an Server zugewiesene Jobs bearbeitet. Diese Modellierung ist in Abbildung 8.6 visualisiert.

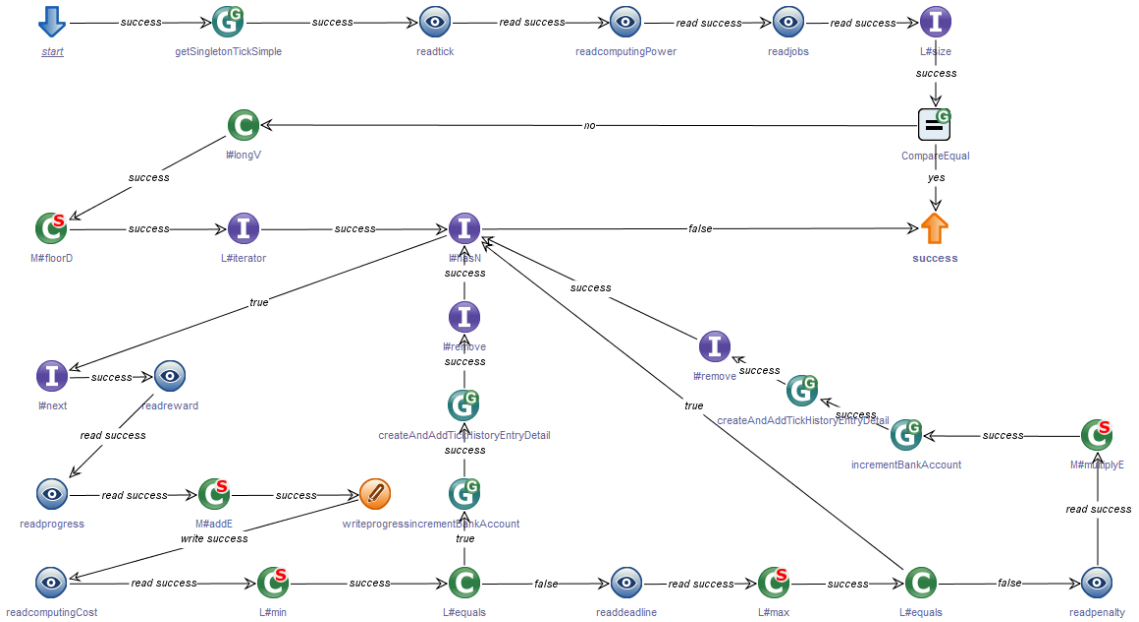

Abbildung 8.6: *calculateJobsInServer*: Berechne Job-Status im Server

Dabei wird die Rechenleistung des Servers auf die ihm zugewiesenen Jobs gleichmäßig aufgeteilt. Der Fortschritt jedes Jobs im laufenden Tick wird nacheinander berechnet. Anschließend wird geprüft, ob der Job zum Tickende erledigt sein wird. Falls ja, wird die Entlohnung dem Bankkonto des Rechenzentrums zugewiesen. Ansonsten wird geprüft, ob die Deadline des Jobs noch nicht überschritten ist. Falls die Deadline nicht eingehalten wurde, wird eine entsprechende Strafe vom Bankkonto abgebucht. In beiden Fällen wird der Job aus dem Server entfernt. Falls die Deadline noch in der Zukunft liegt, so wird mit der Bearbeitung des nächsten dem Server zugewiesenen Jobs fortgefahren.

Krise auslösen

Im Laufe des Spielgeschehens sollen die Spieler mit Krisen konfrontiert werden. Ein Krisenszenario ist das Ausfallen von kritischen Komponenten im Rechenzentrum. Dieses Verhalten ist im Prozess *randomCrisis* (Abb. 8.7) dargestellt.

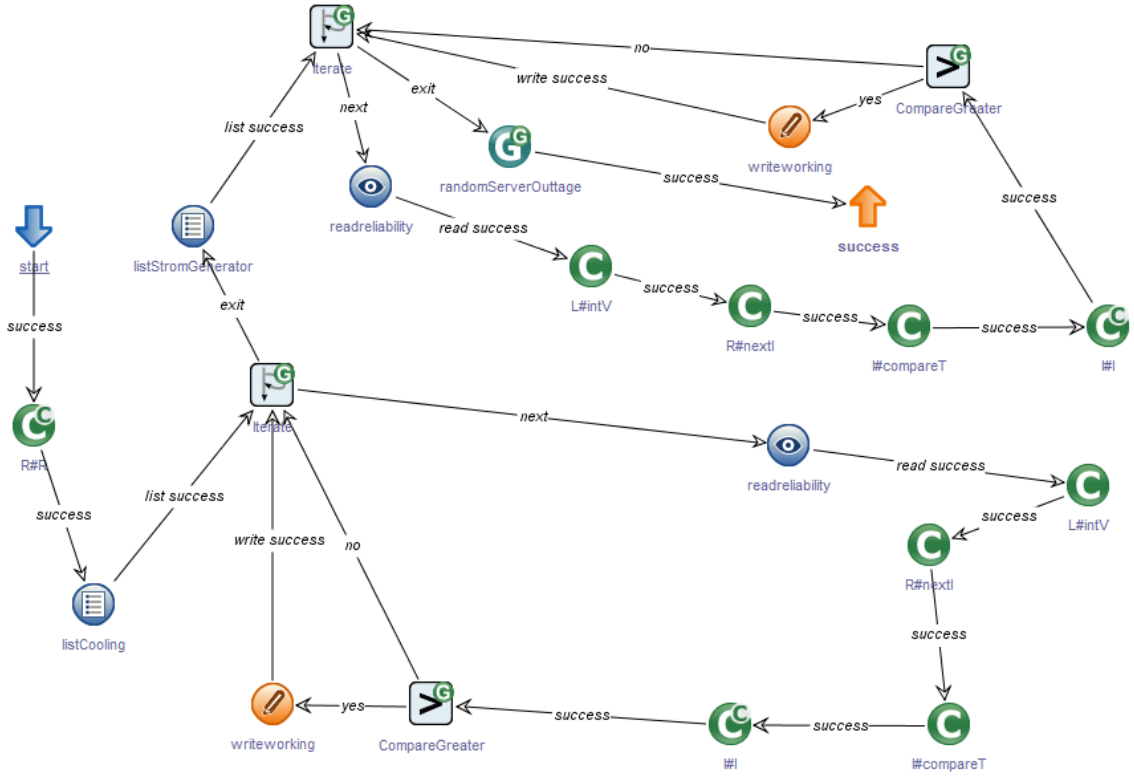

Abbildung 8.7: *randomCrisis*: Löst eine zufällige Krise aus

Dieser Prozess löst mit einer festgelegten Wahrscheinlichkeit einen Ausfall des Stromgenerators, der Kühlung oder eines Servers aus. Abhängig von der ausgefallenen Komponente werden ein oder mehrere Server nicht funktionsfähig, was zu finanziellen Verlusten des Rechenzentrums führt.

Jobs mit überschrittener Deadline entfernen

Der JobMarket ist so aufgebaut, dass die Gesamtheit der Jobs nicht von einem einzelnen Rechenzentrum erschöpfend abgearbeitet werden kann. Das bedeutet, dass im JobMarket Jobs verbleiben, die vom Berater nicht angenommen werden. Solche Jobs müssen im Laufe des Spielgeschehens entfernt werden, wenn ihre Deadline überschritten wird. Diese Funktionalität ist im Prozess *removeJobsPastDeadline* (Abb. 8.8) umgesetzt.

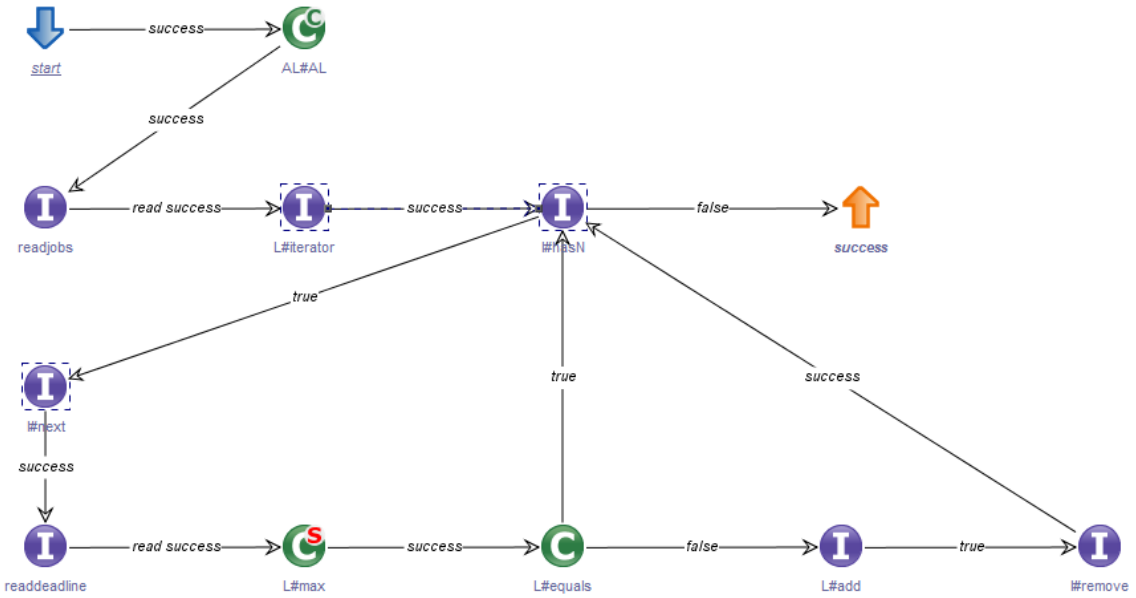

Abbildung 8.8: *removeJobsPastDeadline*: Entfernt nicht erledigte Jobs

Im Prozess wird geprüft, ob die Deadlines der sich im JobMarket befindenden Jobs unterhalb der spezifizierten Grenze befinden. Falls ja, werden diese Jobs aus dem JobMarket entfernt. Diese Jobs führen zu keiner Strafzahlung, da diese keine angenommenen Aufträge darstellen.

8.4 Erstellen der Oberflächen

Mit der Modellierung der Prozesse ist es nun möglich, die bereits in *Cinco* geplanten Spielelemente in DyWA abzubilden und eine reaktive Webanwendung mit interagierbaren Komponenten zu erstellen. Um dies zu erreichen kommen unterschiedliche Werkzeuge für die Gestaltung der Weboberfläche zum Einsatz, die von der Projektgruppe 586 entwickelt wurden.

8.4.1 Gestaltung der Spielkomponenten mit DyWA-GUI

Mit der Erweiterung *DyWA-GUI* können nach einem Baukastenverfahren mehrere Bausteine zu Webseiten kombiniert werden. Dazu stehen verschiedene Komponenten zur Verfügung, die je nach geplanten Einsatzzweck beliebig parametrisiert und platziert werden können. Neben simplen Elementen wie zum Beispiel ein Link oder der Einbettung einer Bilddatei können auch bereits hinterlegte Prozesse mit Eingaben versehen und ausführt werden. Die Abbildung 8.9 zeigt den einen Ausschnitt des GUI-Builders von einer zusammengebauten Seite. Über Schaltflächen lassen sich neue Komponenten hinzufügen, bearbei-

ten oder auch wieder entfernen. Außerdem können Komponenten innerhalb eines Rasters sowohl horizontal als auch vertikal beliebig anordnen. Ein weitere nützliche Funktion des GUI-Builders stellt das Absichern von Komponenten dar, um diese nur für bestimmte Nutzer-Rollen aufrufbar zu machen. Diese Funktion ist für das Planspiel sehr hilfreich, da hier mehrere unterschiedliche Rollen realisiert werden, die jeweils einen begrenzten Zugriff auf eine gemeinsame Datengrundlage haben.

8.4.2 Implementierte Oberflächen

Insgesamt gibt es im Krisenplanspiel mehrere Seiten, die alle unterschiedliche Spielelemente beinhalten. Im Vordergrund des Planspiels steht das Dashboard (vgl. Abbildung 8.10) , welches das „Herz“ einer jeden Rolle darstellt. In diesem Bereich wird ein grober Überblick über die einsehbaren Daten gegeben. Außerdem kann vom Dashboard aus kann jede – entsprechend der Rolle – zur Verfügung stehende Aktion aufgerufen oder genauere Daten eingesehen werden.

- **Tick Overview**: dieser Bereich zeigt den aktuellen Tick an, in dem sich das Planspiel momentan befindet. Der Button *Do Tick* erhöht nach Betätigung den Wert um eins und führt alle damit verbundenen Aktionen des *doTick*-Prozesses aus. Dieser Button ist nur für die Admin-Rolle sichtbar.
- **Job Markets**: in dem Bereich „Job Markets“ werden alle zur Verfügung stehenden Job-Anfragen aufgelistet. Dieser Bereich ist nur für die Berater-Rolle sichtbar. Mit dem Übergang in einen neuen Tick können sowohl Angebote verschwinden als auch neue hinzukommen. Durch Betätigung der Schaltfläche *Show* wird eine detaillierte Ansicht (vgl. 8.11) geöffnet und es werden genauere Informationen zu jedem Job angezeigt. Außerdem kann in dieser Ansicht ein Job zum Service Center übertragen und somit der Techniker-Rolle verfügbar gemacht werden.
- **Service Centers**: die Komponente des Service Centers wird nur Spielern der Techniker-Rolle angezeigt. In dieser Liste werden alle Jobs angezeigt, die zuvor von einem Berater angenommen wurden. Der Techniker ist nun für die Zuordnung zu einem Server zuständig.
- **Servers**: der gesamte Spielbereich *Servers* steht nur dem Techniker zur Verfügung und bildet sein Haupteinsatzgebiet ab (vgl. Abbildung 8.12). In der ersten Tabelle lassen sich alle Informationen zu den verwendeten Servern ablesen. Für den störungsfreien Einsatz wichtige Daten (*Reliability* für die Zuverlässigkeit und *Working* für den aktuellen Zustand) werden hierbei durch Ampeln visualisiert. Eine gelbe oder

gar rote Ampel signalisiert dem Techniker dabei, dass die entsprechenden Komponenten besondere Aufmerksamkeit benötigen. Neben den Servern werden in diesem Bereich auch der Stromgenerator und die Kühlung angezeigt. Fällt eines dieser Elemente aus, kann dies die Funktionsweise der Server negativ beeinflussen. Falls eine dieser Komponenten eine zu geringe Zuverlässigkeit aufweist oder bereits ausgefallen ist, muss diese repariert werden. Dies passiert in Form eines Minispiels, welches über die entsprechende Schaltfläche (Repair Server/Cooling/Generator) aufgerufen werden kann. Ziel dies Minispiels (vgl. Abbildung 8.13) ist es alle Checkboxen so anzuwählen, dass höchstens eine nicht aktiviert ist. Ist dieser Zustand erreicht, wird ein zusätzlicher Button eingeblendet, dessen Betätigung die ausgewählte Komponente repariert und die Zuverlässigkeit wieder herstellt.

- **Bankaccounts**: im Bankaccount (vgl. Abbildung 8.14) wird, wie der Name bereits vermuten lässt, das aktuelle Guthaben der Spielergruppe angezeigt. Da nur der Finanzverwalter eine genaue Einsicht in die Ausgaben und den aktuellen Guthabenstand hat, obliegt es ihm den finanziellen Überblick zu behalten.

- **Tick History**: der Bereich *Tick History* stellt einen Verlauf der bisherigen Geldbewegungen der vergangenen Ticks dar (vgl. Abbildung 8.15). Jede Zeile stellt innerhalb der Tabelle dabei genau einen Tick dar. In der Spalte *Costs* wird die Veränderung des Guthabens in diesem Tick festgehalten. Ein negativer Wert steht dabei für Verluste. In der Zelle *Current Balance* wird das Guthaben zum Ende dieses Ticks angezeigt (beispielsweise haben die Spieler zu Beginn ein Guthaben von 1000 Einheiten und verursachen bereits im ersten Tick Kosten in Höhe von 20. Das Guthaben beträgt folgerichtig zum Ende des Ticks nur noch 980 Einheiten). Mit der Schaltfläche *Show* lassen sich die Einnahmen und Ausgaben des jeweiligen Ticks detaillierter aufschlüsseln. Wie auch der Bankaccount, lässt sich dieser Bereich nur von Spieler mit der Rolle des Finanzverwalters einsehen.

- **Flipboard**: das Flipboard (vgl. Abbildung 8.16) stellt das Hauptkommunikationselement im Krisenplanspiel dar und kann (beziehungsweise soll) von jeder Spieler-Rolle genutzt werden. Mit dem Button *Write a Flipboard-Entry* lassen sich beliebige Nachrichten erzeugen, die bei allen anderen Mitspielern im Flipboard angezeigt werden. Da das Flipboard neben dem Dashboard eine Kernkomponente darstellt, kann es von jeder Stelle im Planspiel aus über die Navigationsleiste erreicht werden.

Neben den eigentlichen Spielelementen gibt es zusätzlich ein Optionsmenü für den Administrator. Dieses lässt sich direkt von der Hauptseite aus mit der Schaltfläche *Options* erreichen. In diesem Bereich können mehrere Faktoren gesetzt werden um Einfluss auf den Spielverlauf zu nehmen. Hierbei gibt es folgende Möglichkeiten:

- **Reward**: hier kann ein Wert eingetragen werden, mit dem der Gewinn (für das erfolgreiche Abschließen eines Jobs) multipliziert wird
- **Penalty**: analog zum Reward kann mit diesem Wert die Strafzahlung für das Nicht-erfüllen eines Jobs beeinflusst werden
- **Least deadline**: mit diesem Wert lässt sich eine untere Grenze für die Deadline eines Jobs festlegen. Alle Jobs die nach Festlegung des Wertes generiert werden, weisen keine Deadline mit tieferem Wert auf
- **InterestDivisor**: mit diesem Wert können die Zinsen festgelegt werden, die für das Überziehen des Guthabens (also sobald das Guthaben einen negativen Wert erreicht) anfallen. Der Wert 100 steht hierbei beispielsweise für 100%

Des Weiteren kann der Administrator das Planspiel in diesem Bereich über die entsprechende Schaltfläche zurücksetzen.

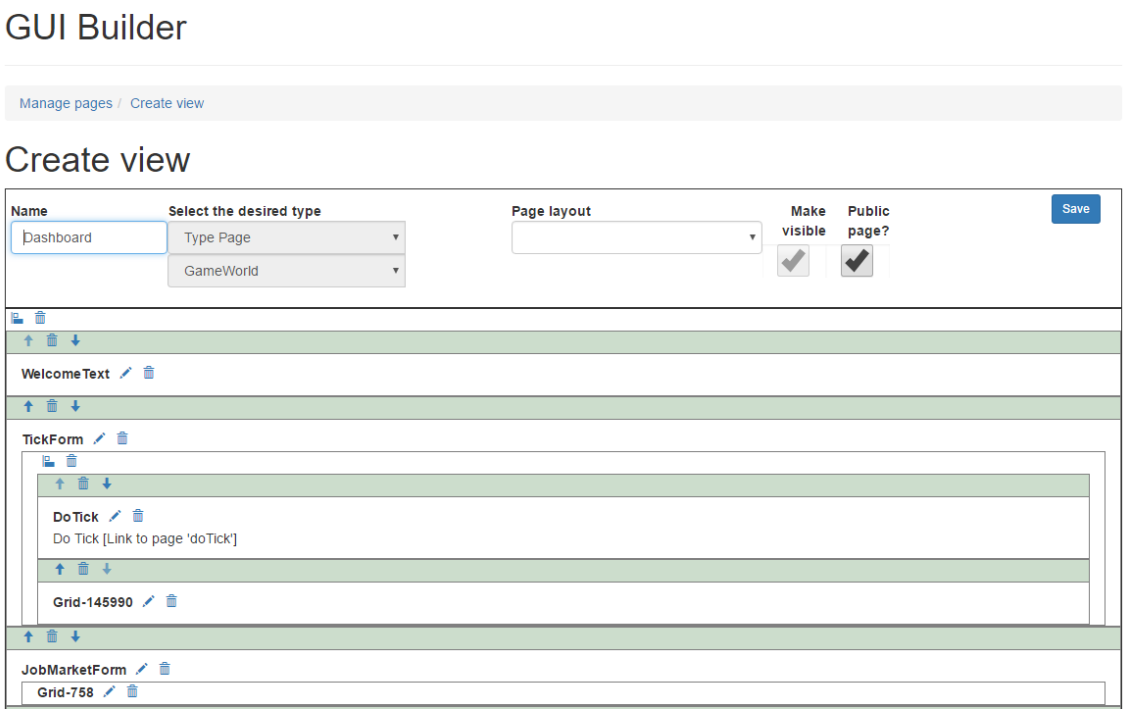

Abbildung 8.9: Ausschnitt des Dashboards im GUI-Builder

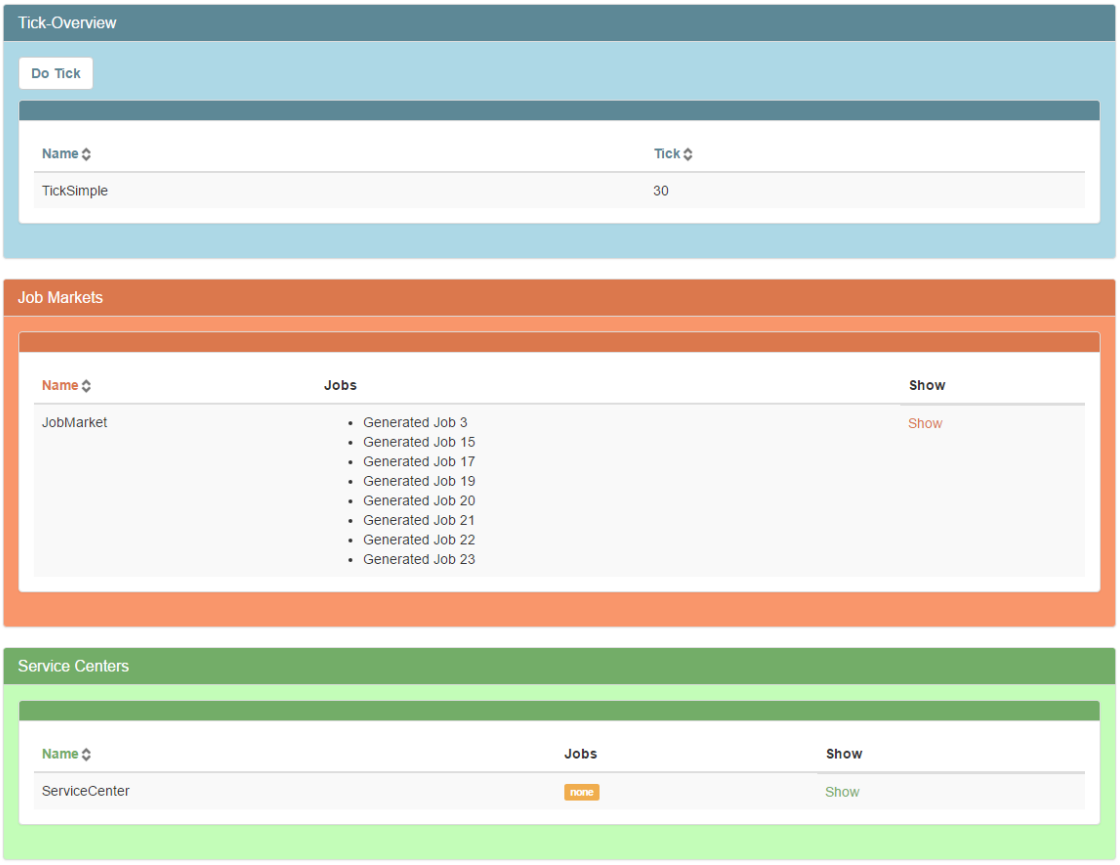

Abbildung 8.10: Teil des Dashboard aus Admin-Ansicht

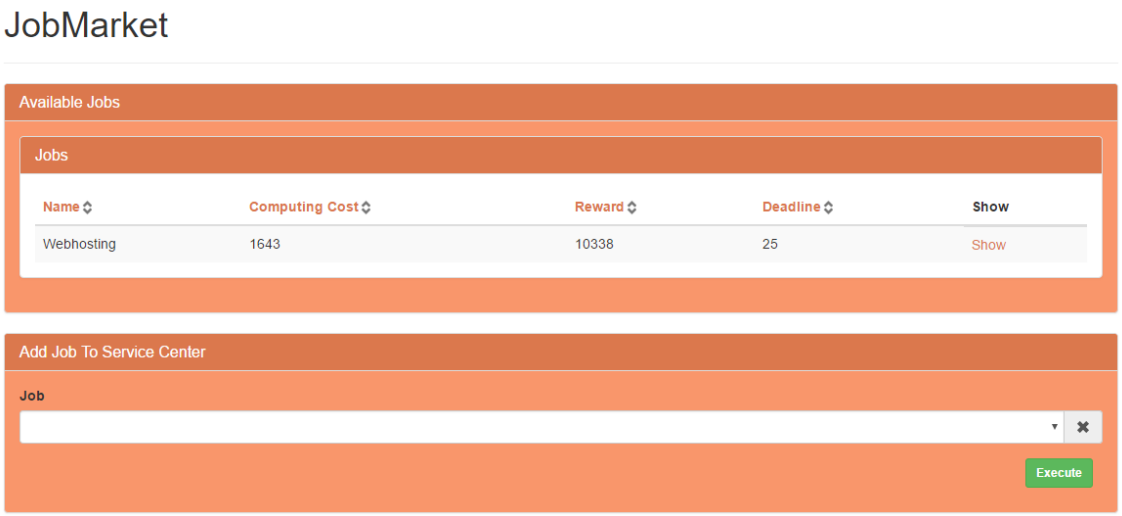

Abbildung 8.11: Detaillierte Ansicht des Job Market

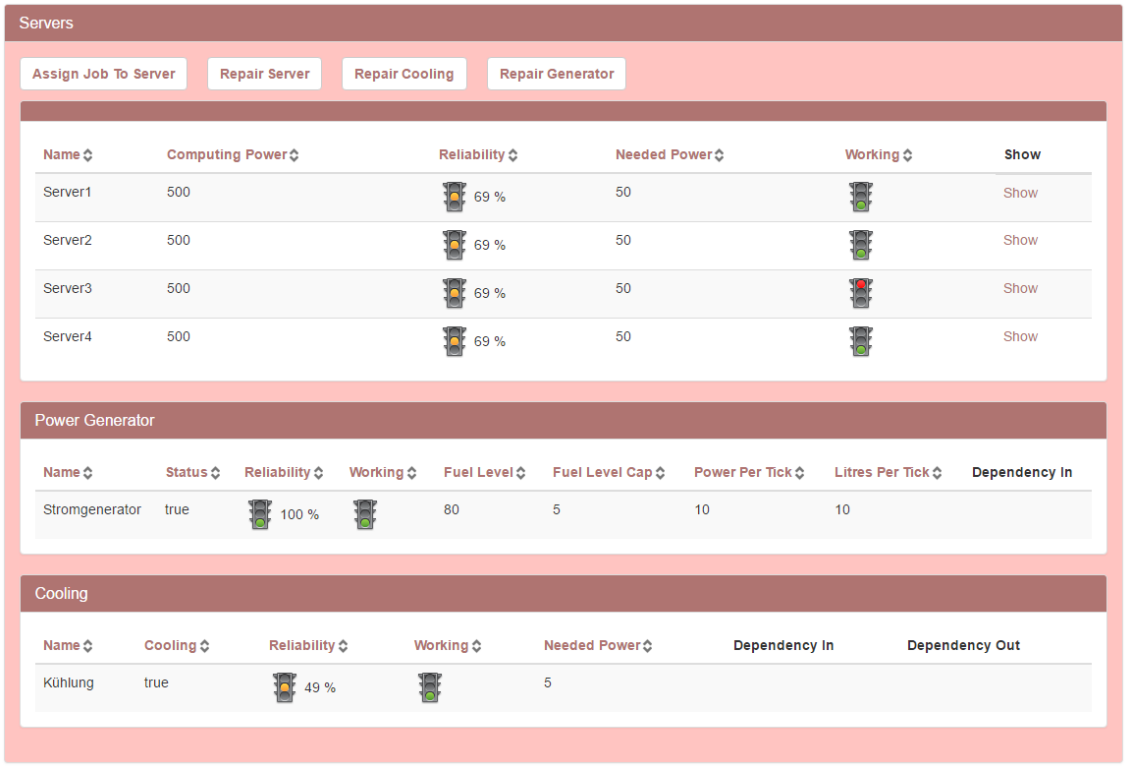

Abbildung 8.12: Serverbereich des Technikers

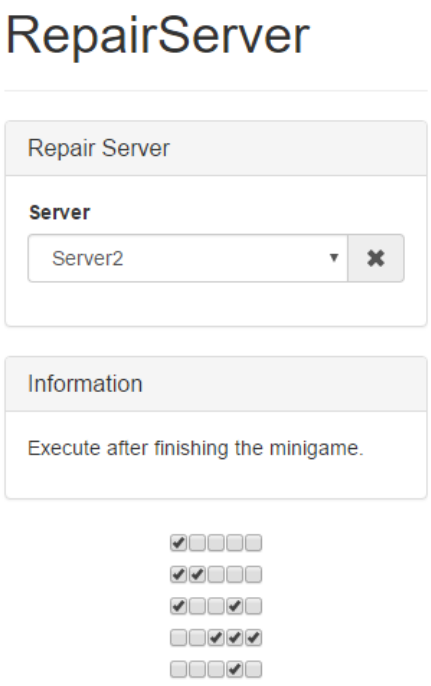

Abbildung 8.13: Minispiel für die Reparatur von Komponenten

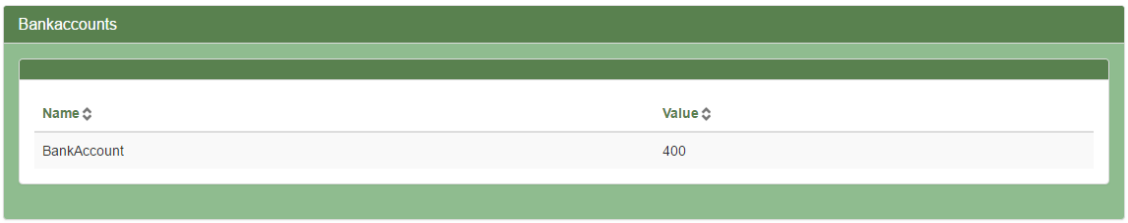

Abbildung 8.14: Aktuelles Guthaben im Bankaccount

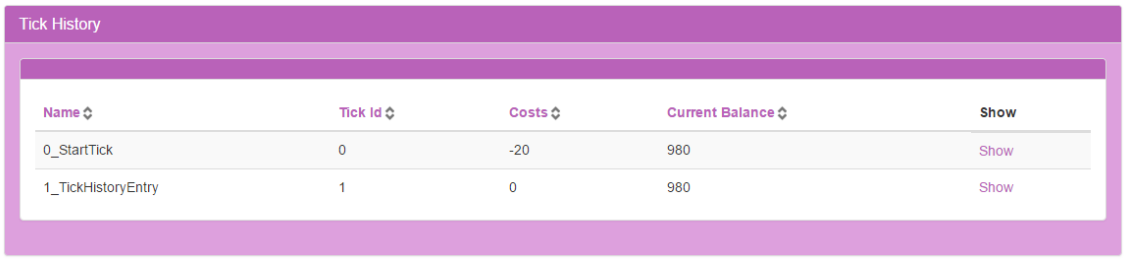

Abbildung 8.15: Verlauf der Einnahmen und Ausgaben

Abbildung 8.16: Kommunikation erfolgt über das Flipboard

8.5 Erweiterung der DyWA

Im Laufe der PG wurden einige DyWA-Plugins um verschiedene Funktionen erweitert, welche für die Realisierung des Krisen Planspiels mithilfe von DyWA benötigt werden. Dabei beziehen sich die Erweiterungen zum großen Teil auf die Benutzeroberfläche, also Änderungen im *DyWA GUI Plugin*.

8.5.1 Neue Komponenten

Um das Planspiel in DyWA zu realisieren, fehlten einige UI-Komponenten. Diese Komponenten wurden im DyWA GUI Plugin eingepflegt. Grob lassen sich die Funktionen in zwei Kategorien einordnen. Zum einen die Metaebene, welche allgemein die Usability

des GUI-Editors erhöht. Zum anderen Komponenten, welche für den Spielablauf benötigt werden.

Seiten kopieren

Eine neue Funktion ist das vollständige Kopieren von bereits bestehenden Seiten.

Durch diese Funktion können zuvor erstellte Seite dupliziert werden. Diese Funktion ist hilfreich um ähnliche Seiten effektiv anlegen zu können. Es muss nur eine Seite erstellt werden. Alle ähnlichen Seiten können von dieser kopiert werden, so muss nicht jede Seite komplett neu angelegt werden. Vielmehr können jetzt nur die relevanten Delta-Komponenten, also jene welche anders sind, auf der kopierten Seite geändert werde.

Name of the page	Title	Saved	Public Page	Renderable Type	Edit properties	Edit page	Copy page	Delete page
ServiceCenter	Service Center	true	true	Type Page	Edit properties	Edit	Copy	Delete
Dashboard	Dashboard	true	true	Type Page	Edit properties	Edit	Copy	Delete
addJobToJobMarket	Process AddJobToJobMarket	true	true	Process Page	Edit properties	Edit	Copy	Delete

Abbildung 8.17: Kopieren einer bestehenden Seite

Abbildung 8.17 zeigt den Menüpunkt, welcher das Kopieren einer Seite ermöglicht. Dieser befindet sich in der *Seitenverwaltungs-Ansicht*.

Wenn eine Seite kopiert wird ändert sich der Titel der kopierten Seite automatisch und bekommt das Präfix *Copy of*, darauf folgt anschließendem der original Seitentitel.

Um die Funktionalität der Kopierfunktion zu gewährleisten muss jede Komponenten, die von *Component* erbt, die Methode *clone()* implementieren und dafür Sorge tragen das ihre Inhalte dupliziert werden. Abbildung 8.18 zeigt den relevanten Codeausschnitt zum Klonen einer *MiniGameComponent*.

```
1    @Override
2    public MiniGameComponent clone() {
3        final MiniGameComponent clone = new MiniGameComponent();
4        super.clone(clone);
5        clone.target = target;
6        clone.context = context;
7        clone.targetUri = targetUri;
8        clone.targetType = targetType;
9        return clone;
10   }
```

Abbildung 8.18: `clone`-Methode der `MiniGameComponent`

Verschachteln von Layouts

Eine weitere effektive Möglichkeit Zeit bei der Erstellung von Seiten zu sparen, ist das einfügen von bereits bestehenden Komponenten in eine Seite.

Dabei wird in einer Seite, anstelle eines *Process Form* oder *Type Form*, eine *Complex Container* eingefügt (vgl. Abb. 8.19).

In diesen Container kann nun ein unterschiedlicher Container eingefügt werden, welcher sich auf einer anderen Seite befindet (Abbildung 8.20). Bearbeitet man nun einen dieser Container wird der Inhalt an allen Referenzstellen geändert.

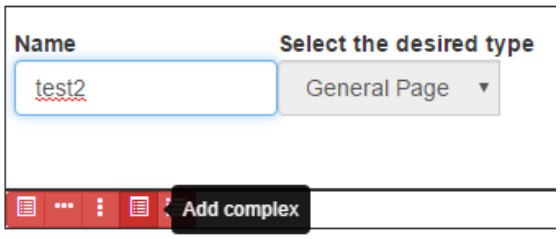

Abbildung 8.19: Complex-Container einfügen

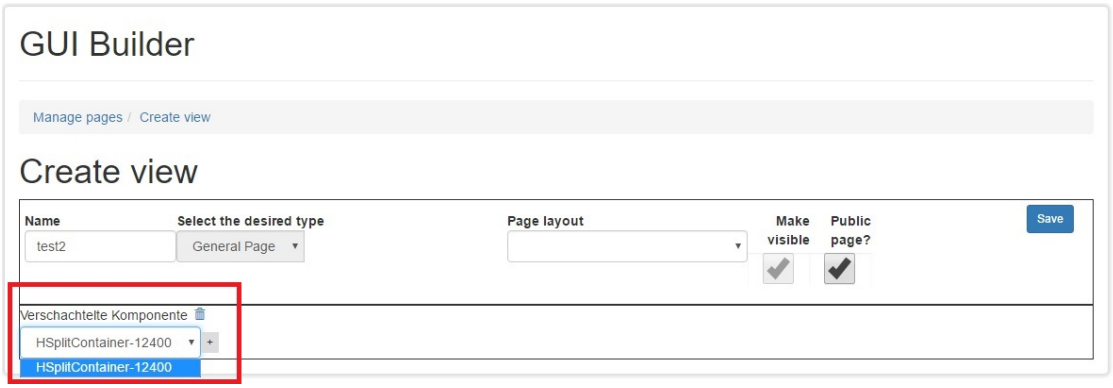

Abbildung 8.20: Einfügen eines anderen Container in die Seite

ProcessOutputGrid - Ausgabe von Prozessergebnisse

Das ProcessOutputGrid löst ein weiteres Problem, welches bei der Entwicklung des Planspiels aufgetreten ist. Es gab keine Möglichkeit die Ergebnisse eines Geschäftsprozesses direkt in einer Komponente darzustellen.

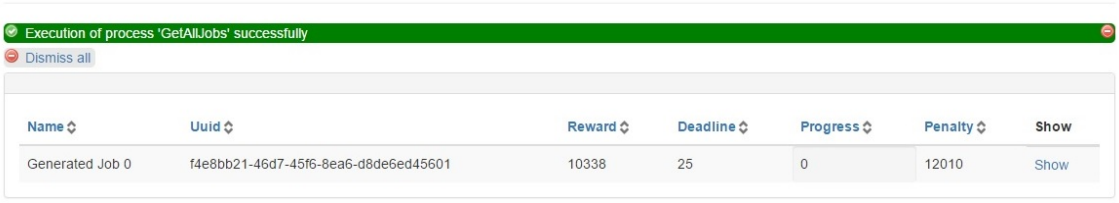

Abbildung 8.21: Ausgabe ProcessOutputGrid des getAllJobsInServiceCenter-Prozesses

Das ProcessOutputGrid löst dieses Problem. Dieses kann das Ergebnis eines Prozesses in Form einer Tabelle ausgeben (Abbildung 8.21). Dazu muss dieses in einem *ProcessForm-Container* eingebettet werden. Abbildung 8.22 stellt den Editor des Grids dar.

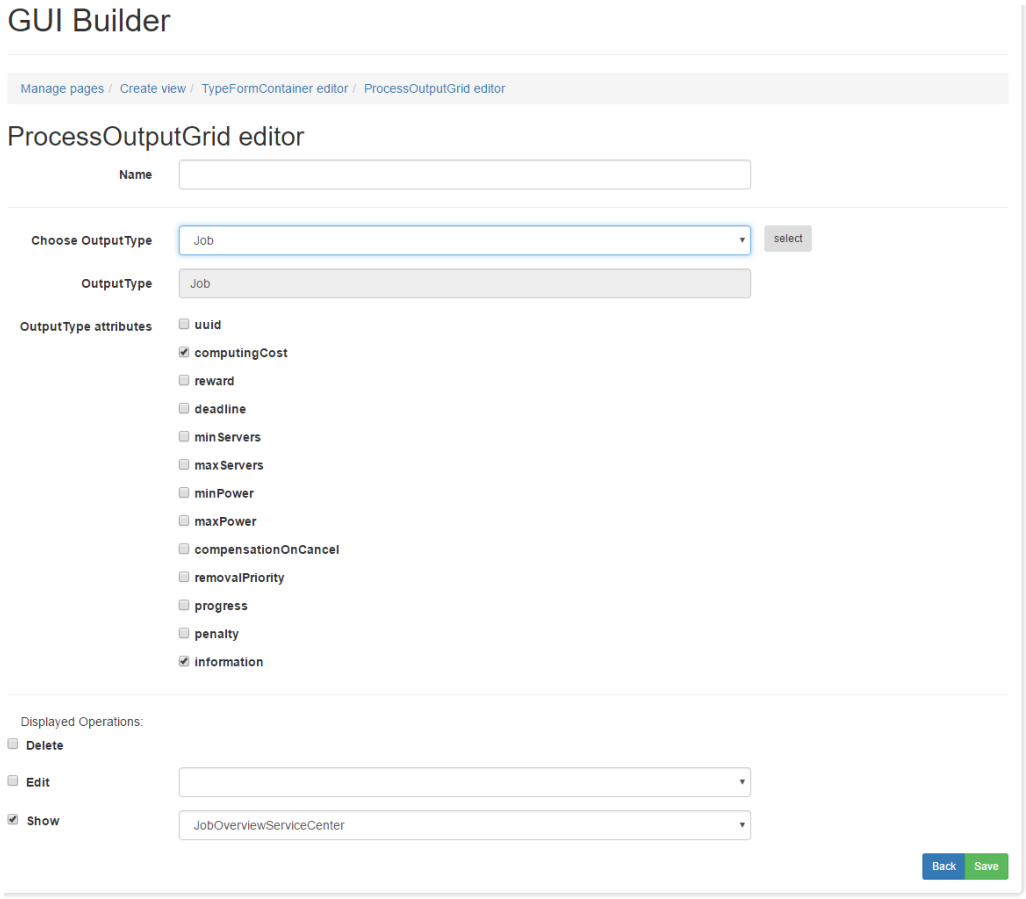

Abbildung 8.22: Editor des ProcessOutputGrid vom getAllJobsInServiceCenter-Prozess

MiniGameComponente

Zum Einfügen der JavaScript-Minispiele wird eine Komponente benötigt, welche JavaScript in der DyWA-Gui einbettet und das Ergebnis des Spiels an eine Komponente weitergeben kann.

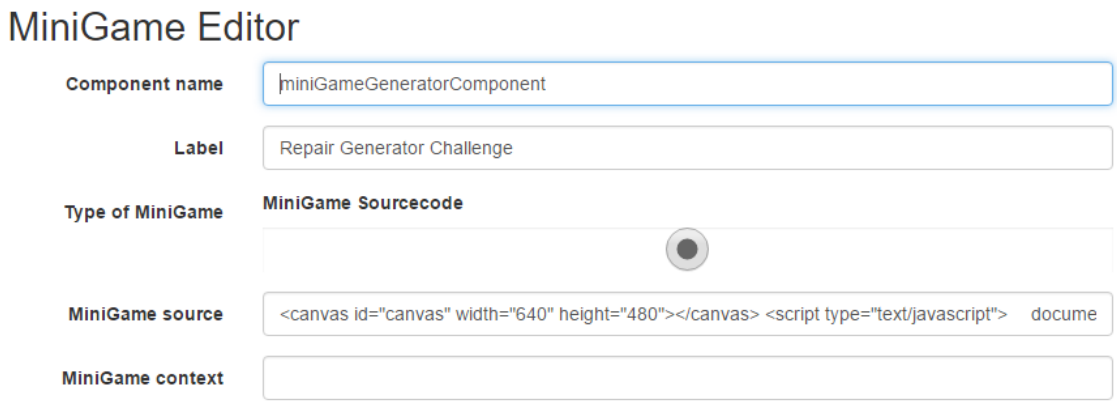

Abbildung 8.23: Minigame

Dies wird durch die *MiniGameComponent* umgesetzt. In diese können beliebige JavaScript-Minispiele eingefügt werden. Abbildung 8.23 zeigt die Integration eines Reversi in das Planspiel.

8.5.2 Styling mittels CSS

Damit das Spiel grafisch veränderbar und damit gestaltbar ist, muss es eine Komponente geben, die an einen Designexperten übergeben werden kann. Die Projektgruppe hat sich dafür entschieden, dies mittels externer CSS-Datei umzusetzen. Dies hat den großen Vorteil, dass die grafische Umsetzung von einem externen Designer durchgeführt und augenblicklich an der Anwendung getestet werden kann, ohne dass Personen aus dem eigentlichen Projekt hinzugezogen werden müssen.

Das DyWA-Gui-Plugin selbst wurde dafür so erweitert, das der *Unique-Identifier* einer jeden Komponente als Klasse an das Elternelement im DOM geschrieben wird. So kann das Element sowie alle Unterelemente per CSS angesprochen und im Aussehen individualisiert werden.

8.6 Umgesetzte Ziele

Im folgenden Abschnitt werden die angesetzten Ziele (vgl. Abschnitt 1.2) an die Arbeit in der Projektgruppe sowie deren Realisierung im Planspiel (vgl. Kapitel 6.1) reflektiert. Dabei sollen Antworten auf die Fragen „Was konnte die PG592 an Zielen umsetzen?“ und „Wie greifen die realisierten Spielkomponenten diese Ziele auf?“ gefunden werden. Dazu werden im Folgenden Teile aus den vorherigen Kapiteln stichpunktartig aufgegriffen.

1. **Anwendung von prototype-driven Verfahren mithilfe des Lehrstuhl 5 Toolstacks**
   Die PG592 hat sowohl in der Planungs- als auch in der Implementierungsphase die Tools des Lehrstuhl 5 benutzt. Die Modellierungsphase konnte mithilfe des Software-Werkzeugs CINCO erfolgen. Obwohl die Modellübertragung aufgrund von Versionsdifferenzen mit Pyro händisch erfolgte, konnten die in der ersten Phase erarbeiteten Ergebnisse übertragen werden, um danach das Planspiel mithilfe der DyWA und den durch die PG ProBio entwickelten Plugins zu realisieren. Während der unterschiedlichen Arbeitsphasen konnten viele der auftretende Probleme mit dem Toolstack behoben werden. Durch die Anwendung von prototype-driven Verfahren war es zudem möglich, diese Verbesserungen zeitnah zu integrieren und die Qualität der Arbeitsprozesse innerhalb der PG592 zu verbessern.

2. **Erarbeitung von Schulungskonzepten zur Krisenbewältigung**
   Ein weiteres Ziel der PG592 ist die Erarbeitung von Konzepten zur Krisenbewältigung. Dabei hat die PG592 den Fokus auf die Kommunikation zwischen den einzelnen Rollen des Planspiels und die präventiven Maßnahmen zur Krisenbewältigung gelegt. Dieser Fokus zeigt sich u.a. an den Aufgaben der Rolle des Technikers. Es werden gezielt die Kommunikationswege und die Präventivmaßnahmen (regelmäßige Wartung von Servern) gefördert.

   Domänenspezifisches Fachwissen bezüglich Rechenzentren kann durch die Variation der MiniGame Komponente erfolgen, indem beispielsweise Wissen für Techniker in diesen Rätseln gefestigt wird statt abstrahierte Minispiele wie z.B. Reversi oder Sudoku zu benutzen.

3. **Erstellung eines erweiterbaren Planspiels im Kontext von Rechenzentren**

   Die PG592 hat eine spielbare Simulation eines Rechenzentrums erstellt, die sowohl Aspekte der Krisenbehandlung als auch Aspekte eines Planspiels beinhaltet. Vor

allem liegt der Fokus auf der Rollenverteilung und der Interaktion zwischen den Spielern. Das Spiel besitzt einen Bezug zum Alltagsgeschäft eines Rechenzentrums, indem es den Arbeitsmarkt, interne Prozesse eines RZ und die strikte Trennung von Rollen abstrahiert nachbildet. Jede durch die PG592 erstellte Spielkomponente bezieht sich auf das ursprünglich angedachte Modell und das Ziel, wie in einer simulierten Umgebung Stress erzeugt werden kann, um die Teilnehmer des Planspiels auf das mögliche Auftreten einer Krise im Alltag vorzubereiten. Die nachfolgend beschriebenen verwendeten Methodiken und Spielkomponenten sollen dies näher erläutern.

- *Prozessmodellierung*: Das Annehmen von Aufträgen, die Weiterleitung an unterschiedliche Rollen sowie deren Kommunikation untereinander stellen einen zentralen Bestandteil des Planspiels dar. Diese abstrahierten internen Geschäftsprozesse eines RZs wurden mithilfe von jABC4 realisiert. Sie beziehen sich direkt auf den Alltagsaspekt einer simulierten Umgebung und sind strikt vom restlichen Einsatz des Lehrstuhl 5 Toolstacks getrennt. Dies ermöglicht das einfache Ändern sowie die weitere feinere Granulierung von Prozessen.
- *Rollenverteilung & Kommunikation*: Die Aufteilung von Spielteilnehmern in Rollen (vgl. Fort Fantastic) mithilfe von DyWA-Security und der Benutzung von CINCO-Modellkomponenten erzeugt beim Spieler direkten Stress: jede Rolle ist für sich selbst verantwortlich, aber ohne Zusammenspiel ist das erfolgreiche Bestehen des Planspiels nicht möglich.
- *Steuerung & Minispiele*: Mithilfe des von der PG592 realisierten Ticksystems, welches sich durch den Spielleiter jederzeit starten oder stoppen lässt (vgl. TimerButtonComponent) ist es möglich, die Spieler gezielt auf Schwächen oder Stärken ihres Zusammenspiels hinzuweisen. Des Weiteren stellt die durch die PG592 entwickelte Minispielkomponente einen zentralen Aspekt bei der Steuerung und dem Ablauf des Spiels dar. Diese Komponente kann vielseitig genutzt werden, indem z.B. Domänenexperten spezifisches Fachwissen abfragen, um nicht nur den Umgang mit Stress, sondern auch konkretes Wissen zu vermitteln.
- *Anpassungsmöglichkeiten*: Die realisierten Copy, ProcessOutputGrid und Complex-Container-Erweiterungen innerhalb des DyWA-GUI Toolstacks ermöglichen zudem die leichte Möglichkeit, das Spiel zu erweitern oder auf eine andere Domäne zu übertragen.

Zusammenfassend betrachtet unterstützen die realisierten Spielkomponenten und die benutzen Software-Werkzeuge das Ziel der Projektgruppe mittels Planspiel Lehrinhalte (vgl.

Kapitel 1.2) zu vermitteln. Die Rollenverteilung und Kommunikation setzt dabei den Fokus auf die Interaktion der Spieler, Minispiele erzeugen Zeitdruck, da möglich anfallende Tätigkeiten dieser Rolle nicht mehr zeitnah bearbeitet werden können. So entsteht ein Kommunikationsstau, der auf jede Rolle Einfluss hat und die Spieler zur Kommunikation zwingt.

Kapitel 9

Auswertung einer Spieleinheit

Um auf die Auswertung einer Spieleinheit eingehen zu können, werden zunächst die einzelnen Kernkomponenten während einer Simulation erläutert. Anschließend wird eine Auswertung erfüllter Ziele aufgezeigt und das Einhalten der angestrebten Ziele der Krisenplan-Simulation diskutiert.

9.1 Spielablauf

Damit ein Lerneffekt für dies Teilnehmer des Planspiels entstehen kann, müssen Ablauf und Schwierigkeitsgrad der einzelnen Krisenplan-Simulator-Stufen festgelegt werden. Hierbei ist es wichtig, dass die Teilnehmer trotz einer simulierten Stresssituation zu jeder Zeit einen „roten Faden“ erkennen können und sich motiviert fühlen, gegebene Probleme zu lösen. Die Notwendigkeit eines erkennbaren Ablaufes für den Lerneffekt der Teilnehmer stellt sich als unabdingbar dar, um gegebene Ziele der Spielleitung zu erreichen [22]. Um einen solchen Lerneffekt erreichen zu können, gestaltet sich der Spielablauf rundenbasiert. Dies wurde gewählt, da ein qualitatives Feedback des Spielleiters nach jeder Runde erfolgen kann, ohne gleichzeitig den Spielfluss zu unterbrechen [6]. Der folgende Abschnitt erläutert die Regeln, um einen möglichst lernintensiven Ablauf gewährleisten zu können.

9.1.1 Vorbereitungen auf die erste Simulations-Runde

Nachdem eine ausreichend große Gruppe vorhanden und die jeweiligen Rollen den Teilnehmern zugeordnet sind, wird der Gruppe vom Spielleiter das Spiel einführend erläutert. Dabei schildert er den Kontext der Simulation sowie wesentliche innerhalb der Simulation

zu bewältigende Aufgaben. Hierbei werden die einzelnen Punkte nur grob skizziert, um potenzielle Problemstellungen nicht unnötig vorweg zu nehmen.

Um die Frustrationstoleranz der Spieler beim Start des Spiels nicht zu erschöpfen, werden in Einzelfällen Hilfestellungen gegeben. Ein Beispiel für eine derartige Handlung wäre, dass die Teilnehmer früh mit der Kommunikation beginnen müssen um ein verfrühtes Erreichen eines negativen Budgets zu vermeiden.

Nach Abschluss der Einführung und einer kurzen Frage-Runde begeben sich die Teilnehmer zu den einzelnen Rechnern, woraufhin der Spielleiter die Simulation startet.

9.1.2 Ablauf der weiteren Runden

Nach Abschluss der ersten Simulations-Runde sowie der entsprechenden Diskussionsrunde folgen Simulations-Runden zwei und drei. Hierbei ist - abgesehen von der einführenden Erklärung - der Ablauf identisch. Unterschiede sind die jeweilige Anpassung des Schwierigkeitsgrades seitens des Spielleiters sowie, im Anschluss der jeweiligen Runde, Diskussionsrunden welche Abläufe besser oder auch schlechter abgelaufen sind.

9.1.3 Rollenansichten

Um diese Rollenansichten umsetzen zu können arbeiten alle Teilnehmer an separierten Computern, welche jeweils einer bestimmten Rolle zugeordnet sind und mit einem zentralen Web-Server kommuniziert. Dieses Vorgehen ist notwendig um eine Darstellung und einen Datenaustausch zu gewährleisten, der verhindert, dass die Teilnehmer untereinander die genauen Aktivitäten der anderen Teilnehmer erkennen können und so das Spielprinzip manipulieren.

9.1.4 Zusammenfassung

Durch die Entscheidung der Projektgruppe mehrere Spielrunden mit unterschiedlichen Schwierigkeitsgraden zu ermöglichen, soll insbesondere der Lernerfolg sowie der Spaß an der Krisenplan-Simulation erhöht werden. Durch das Erreichen von messbaren Zielen (beispielsweise hohe Einnahmen) kann außerdem ein Konkurrenzkampf zwischen verschiedenen Teilnehmergruppen in unabhängigen Spielrunden entwickelt werden. Bei den Teilnehmern soll im Idealfall nicht das Gefühl ausgelöst werden, dass sie einen Workshop absolvieren, sondern vielmehr mit Spaß spielen. Zudem wird durch die stetige Erhöhung des Schwierigkeitsgrades vermittelt, dass eine intensive Kommunikation, gerade in Krisensituationen, absolut notwendig ist, um eine Problembewältigung zu erreichen.

9.2 Kommunikation

Im Krisenplanspiel nimmt die Kommunikation eine bedeutende Rolle bei der Prävention und Bewältigung von Problemen ein. Da in dem Simulator die einzelnen Nutzer jeweils unterschiedliche Rollen belegen, muss ein Informationsaustausch ermöglicht werden. Dies ist notwendig, damit bestimmte Informationen die beispielsweise nur einer bestimmten Rolle zur Verfügung stehen an andere Teilnehmer weitergeleitet werden können.

Die gesamte Kommunikation, die für das Krisenplanspiel notwendig ist, soll sowohl nachvollziehbar als auch formlos sein. Um den Verlauf einer Spielrunde nachträglich vollständig analysieren zu können, werden daher sämtliche Informationsflüsse festgehalten. Dadurch können etwaige Schwächen erkannt werden, die in der Kommunikation begründet sind (beispielsweise Übermittlung von unwichtigen Informationen). Des Weiteren gibt es im Krisenplanspiel eine formlose Kommunikation in Form von Freitext, damit die Spieler sich selbst eine Struktur für den Informationsaustausch und die zu übermittelnden Daten überlegen. Im Krisenplanspiel wurde dies in Form eines simulierten Flipboard-Systems realisiert.

Diese Art der Kommunikation ermöglicht den Spielern viel Freiraum, stellt diese jedoch auch vor mögliche Hindernisse im Hinblick auf Risiken und Stress, worauf im Folgenden genauer eingegangen wird.

9.2.1 Stress durch Kommunikation

Erkennen die Spieler nicht, dass das Erstellen von festen Kommunikationsstrukturen in Form von einheitlichen Vokabular und Strukturen sowohl Zeit spart als auch mögliche Fehlinterpretationen reduzieren kann, wird in zeitkritischen Situationen automatisch Stress verursacht. Kommunizieren beispielsweise die Spieler des Ankaufspersonals und die des Technikpersonals ausschließlich über (lange) Freitexte, nimmt allein die Erstellung und das Lesen dieser Nachrichten viel Zeit in Anspruch. Das Technikpersonal muss zudem aus diesem Text anschließend nur die tatsächlich relevanten Daten extrahieren, die auch für die Abarbeitung des Auftrags notwendig sind. Wenn nun für jeden einzelnen Auftrag ein neuer Text folgt, besteht die Gefahr, dass die Texte nicht mehr sorgfältig gelesen werden. Dadurch können Informationen verloren gehen oder falsch interpretiert werden. Dies kann die Fehlerwahrscheinlichkeit erhöhen (falsche Zuordnungen), Zeitdruck und somit auch Stress auslösen (jeder Auftrag bedeutet neuen Leseaufwand) oder auch weitere Probleme verursachen (Strafkosten, Stress weitet sich auf andere Arbeitsbereiche aus). Diese Problematik kann neben den bereits festgelegten Strukturen von den Spielern auch durch Priorisierung eingegrenzt werden. Besonders wichtige Nachrichten (Störung, besonders wertvoller

Auftrag,...) könnten damit mit Vorrang bearbeitet werden. Wie dies realisiert wird, ist ebenfalls den Spielern selbst überlassen.

9.2.2 Gewinne für das Team

Durch das Erlernen einer strukturierten Kommunikation im Spiel können die Spieler viele positive Aspekte erlernen, welche für ihren Job und das Leben allgemein von großer Bedeutung und Wichtigkeit sind. Durch das erfolgreiche Überwinden der in den vorherigen Abschnitten geschilderten Kommunikationsprobleme sammelt das Team gemeinsame Erfolgserlebnisse, welche für das Team sehr wertvoll sein können. Dies stärkt den Teamgeist und das „Wir-Gefühl“. Das Team lernt durch das Erarbeiten von Kommunikationsstrategien auf einem ihnen unbekannten Gebiet, sich auf neue Gegebenheiten selbstständig einzustellen und über diese klar kommunizieren zu können. Auch lernen die Spieler, welche Bedeutung Absprachen und Regeln bei der Kommunikation zukommen. Auf der anderen Seite wird den Spielern vor Augen geführt, welche negativen Auswirkungen ein fehlendes Einhalten oder fehlendes Erstellen von Strukturen haben kann.

9.3 Mini-Rätsel

Ein Kernelement des Krisenplan-Simulators stellt das Erzeugen von Stresssituationen seitens der Benutzer dar. Durch solche Situationen soll überprüft werden, wie die Spieler auf Stress reagieren und Lösungen gemeinsam ausarbeiten. Aus diesem Grund wird diesem Teilbereich eine große Bedeutung zuteil, damit diese Aspekte ausreichend und möglichst effizient angewendet werden können. Um Stresszustände zu generieren bieten sich unter anderem die Mini-Rätsel an.

In der Abschlusspräsentation durchgeführten Spielrunde wurde deutlich, dass die Mini-Rätsel ihren Zweck erfüllen. Der Spieler befand sich in einem Zustand, in dem mehrere Komponenten des Rechenzentrums ausgefallen waren und durch das Lösen eines Rätsels repariert werden können. Der Kontostand des Spielers wurde durch den Ausfall immer stärker belastet. Da mehrere Komponenten ausgefallen waren, musste der Spieler mehr als ein Rätsel lösen. Irgendwann wurde die Belastung so hoch, dass der Spieler im Rätsel eine fehlerhafte Entscheidung durchführte und in ein zeitliches Defizit geriet. Der Spieler versucht, die Problemsituationen möglichst schnell zu lösen, organisiert sich jedoch zu wenig. Die Entscheidungswahl in solch einer Situation lässt sich in die Kategorien *schnelles Probieren mit Risiken* bis *langsames Lösen mit Verlusten* differenzieren. Ziel ist es dabei, dem Spieler zu vermitteln, dass er mit Organisation und vernünftigen Überlegungen das Risiko

minimieren und eine ökonomische Lösung anstreben kann, ohne die erstbeste Möglichkeit mit unbekanntem Risiko auszuprobieren.

9.4 Auswertung

Um im Anschluss einer Simulation festlegen zu können wie erfolgreich eine Gruppe gespielt hat, wird das Hauptaugenmerk auf das letztlich erreichte Budget gelegt. Dieses stellt neben der Ingame-Währung auch gleichzeitig einen Score dar, welcher sich dazu eignet nicht nur pro Spielrunde sondern auch im Anschluss des Spiels eine Bewertung festhalten zu können.

Neben dem erreichten Budget kann der Spielleiter ebenfalls die Herangehensweise der Kommunikation sowie das Spielverhalten der jeweiligen Rollen bewerten. Sollten sich hierbei zu starke Tendenzen nach oben oder unten ergeben wird dies festgehalten, um bei einer weiteren Runde den Schwierigkeitsgrad entsprechend anzupassen.

Sollten zwei Gruppen während einer Runde gegeneinander angetreten sein so wird auch hier zunächst das jeweilig erreichte Budget verglichen. Die informellen Bewertungen des Spielleiters bezüglich der Kommunikation und Herangehensweise werden im Anschluss in persönlichen Gesprächen mit den Teilnehmern diskutiert. Bei regelmäßig stattfindenen Simulations-Workshops kann der Spieler die jeweilig erreichten Highscores dokumentieren um so zum Einen feststellen zu können, ob das Leistungsniveau gestiegen ist und zum anderen einen höeheren Maßstab ansetzen zu können. Letzeres kann zur fortlaufenden Steigerung der Effizienz im Hinblick auf die Krisenbewältigung führen.

Kapitel 10

Ausblick

10.1 Zusammenfassung

Im Rahmen der Projektgruppe ist ein Planspiel zur Krisenbewältigung in Rechenzentren erstellt worden. Die Entwicklung eines ausführbaren Meta-Modells mit dem Cinco-Framework führte zu neuen Informationen über benötigte Anforderungen an das Planspiel, zeigte kritische Bereiche im Betrieb des Rechenzentrum und der Kommunikation der Mitarbeiter auf. Ebenso können durch das ausführbare Modell erste Ergebnisse in dem eigentlichen Spielablauf zusammengestellt werden, um anhand derer die Basis für das eigentliche webbasierte Planspiel stellen zu können. Die Ausarbeitung stellt basierend auf den Anforderungen und der anfänglichen Situation, der gezielten Vermittlung von Lehr- und Lerninhalten heraus, wie Krisen und Rollen modelliert werden können. Im Anschluss an die Modellierungsphase folgt die Implementierungsphase, in der die Projektgruppe das in der ersten Hälfte erarbeitete Modell in eine Webanwendung überführt. Die Webanwendung ermöglicht den Teilnehmern des Planspiels komfortabel, realitätsnah und spielerisch Lernerfolge zu erzielen. Zur Überführung des Meta-Modells aus Cinco in das Software-Werkzeug DyWA eingebunden, indem die Entität in eine DyWA geeignete Form überführt werden. Auf Basis dieser neu erzeugten Typen können Webformulare als spielerische Interaktionseinheiten gebaut werden. Dabei unterstützt unter anderem die DyWA-Erweiterung DyWA-GUI. Die Rollenverteilung kann beispielsweise mit DyWA Sec gelöst werden. Unter Ausnutzung aller Erweiterungen sowie eigener Entwicklung von DyWA wird ein Planspiel zum Krisenmanagement in Rechenzentren erzeugt, welches von mehreren Personen an verschiedenen Computern gespielt werden kann.

10.2 Prototype-Driven Development

Im Laufe der Projektgruppe wurde nach dem Konzept des Prototype-Driven Verfahren gearbeitet (vgl. Kapitel 5). Dabei geht es unter anderem darum, für den aktuellen Stand an Anforderungen einen ausführbaren Prototypen auf Modellbasis zu entwickeln und die Vorstellungen an das Produkt und die konkreten Anforderungen an diesem Modell auszuprobieren. Diese Art und Weise von Entwicklung ist im Rahmen der Projektgruppe als angenehm empfunden worden, da die konkrete Vorstellung von dem Endprodukt durch die iterativen Verfeinerungszyklen besser definiert wurde. Der vom Lehrstuhl 5 geführte Toolstack für Prototype-Driven Development bereitet auf diese Art von Entwicklung vor und liefert Werkzeuge, die das Modellieren und Ausführen von Anforderungen sowie das Überführen des Modells als Entitäten-Basis in eine weitere Applikation erlauben.

10.3 Planspiel der Projektgruppe

Das von der Projektgruppe erstellte Planspiel befindet sich zum Zeitpunkt der Abgabe in einem Stand, der ein Spielen erlaubt. Dabei können verschiedene Spieler unterschiedliche Aktionen im Spiel ausführen, in Krisen geraten und diese lösen, um das Spiel erfolgreich weiter zu führen oder zu beenden.

Für einen ersten produktiven Einsatz bedarf es allerdings weiteren Verbesserungen. Die einzige Metrik, die zum Abgabezeitpunkt eine Bewertung des Spiels für den Spielleiter als auch für die Spieler darstellt, ist der Kontostand. Denkbar sind hier weitere Kriterien wie beispielsweise die Ausfallzeiten von Servern. Auch könnte die Einführung von automatisierten Logs ein entscheidender Faktor bei der Spielbewertung und der Transparenz des Spielverlaufs sein. Dies würde sowohl den Verlauf als auch die Ergebnisse vergleichbarer und analysierbarer gestalten. Neben der Einführung weiterer Metriken kann das Planspiel verbessert werden, indem der Bezug zum realen Einsatzgebiet deutlicher gestaltet wird. Mögliche Erweiterungen können dabei

- eine verbesserte Darstellung der Spielwelt,
- eine detailreiche Wiedergabe von technischen Aspekten in einem Rechenzentrum (z.B. Vernetzung von Komponenten wie im Planspiel Fort Fantastic) sowie
- eine komplexere Abbildung von Fachwissen (z.B. in Minirätseln)

sein. Des Weiteren sollten zukünftige Iteration des Spiels Verbesserungen bezüglich des Spielflusses enthalten. Eine wichtige Nachbesserung stellt beispielsweise die Echtzeitkom-

munikation zwischen den Spielern dar. Diese könnte statt des verwendeten Flipboards mithilfe des XMPP-Protokolls realisiert werden.

10.4 Toolstack des Lehrstuhl 5

Abseits der neuen Implementierungen der Projektgruppe am Toolstack des PG ProBio (PG 586) gibt es weitere Dinge, die von der Projektgruppe als Nützlich empfunden wurden:

- Kürzere Wartezeiten im Build-Prozess bis Änderungen sichtbar werden
- Die Möglichkeit, ohne langwieriges Deployen der Anwendung Ergebnisse zu bekommen
- Echtzeitkomponenten
- Versionskontrolle für Programmcode und Datenbank
- Globales Refactoring (Änderung im Modell führt zu Änderung im Prozess)

10.5 Abschluss

Die Projektgruppe bedankt sich für das viele nützliche Wissen, den aufregenden Besuch im Rechenzentrum des Kooperationspartners Citkomm und die angenehme Projektgruppenzeit.

Anhang

Übersicht der modellierten Prozesse

1. **addEntryToFlipboard**: fügt einen Eintrag zu einem Flipboard hinzu, beide Parameter müssen spezifiziert werden.
2. **addEntryToSingletonFlipboard**: fügt einen Eintrag zum Singleton-Flipboard hinzu.
3. **addJobToJobMarket**: weist einen Job einem JobMarket hinzu.
4. **addJobToServiceCenter**: weist einen Job einem ServiceCenter zu, beide Parameter müssen spezifiziert werden.
5. **addJobToSingletonServiceCenter**: fügt einen Job zum Singleton-ServiceCenter hinzu.
6. **addNextTickHistoryEntry**: fügt eine zunächst leere TickHistoryEntry für den aktuellen Tick hinzu.
7. **addTickSimpleToGameWorld**: fügt ein TickSimple zu einer GameWorld hinzu, beide Parameter müssen spezifiziert werden.
8. **assignJobToServer**: verschiebt einen Job aus dem ServiceCenter in einen Server.
9. **buyServer**: fügt einen neuen Server der GameWorld hinzu, zieht den Preis von dem Bankkonto ab.
10. **calcGenerator**: Berechnet die Änderungen des Treibstoffs im Generator in einem Tick.
11. **calculateDependencies**: Prüft, ob die Kühlung und der Generator funktionieren. Falls nicht, werden die Server deaktiviert.
12. **calculateJobsInServer**: Berechnet den Progress der Jobs in einem Server nach einem Tick. Fertige Jobs werden gelöscht, die Belohnung zum Bankkonto hinzugefügt, verpasste Deadlines werden entsprechend bestraft.

13. **calculateJobsInServiceCenter**: Prüft, ob die Deadline der Jobs im ServiceCenter nicht eingehalten wurde. Falls ja, werden die Jobs gelöscht und die Strafe vom Bankkonto abgezogen.
14. **checkComputingPower**: Aktualisiert die aktuelle Rechenkraft eines Servers. Resultat hängt davon ab, ob der Server gerade funktioniert.
15. **coolingWorking**: prüft, ob mindestens eine Kühlung nicht funktioniert.
16. **createAllServers**: erstellt 4 Server in der spezifizierten GameWorld.
17. **createAndAddFlipboardEntry**: erstellt einen Eintrag und fügt ihn dem Singleton-Flipboard hinzu.
18. **createAndAddTickHistoryEntryDetail**: erstellt eine TickHistoryEntryDetail und fügt sie der aktuellen TickHistoryEntry hinzu.
19. **createFlipboard**: erstellt ein Flipboard mit einem spezifizierten Namen.
20. **createFlipboardEntry**: erstellt einen Flipboard-Eintrag.
21. **createJob**: erstellt einen Job mit den spezifizierten Attributen.
22. **createJobMarket**: erstellt einen JobMarket mit dem spezifizierten Namen.
23. **createRandomJobInJobMarket**: erstellt einen Job mit zufälligen Attributen und fügt ihn dem Singleton-JobMarket hinzu.
24. **createRoles**: erstellt die vier Rollen, die im Spiel teilnehmen.
25. **createServer**: erstellt einen default-Server mit dem spezifizierten Namen.
26. **createServerInSingletonGameWorld**: erstellt einen default-Server mit dem spezifizierten Namen und fügt ihn der Singleton-GameWorld hinzu.
27. **createServiceCenter**: erstellt ein ServiceCenter mit dem spezifizierten Namen.
28. **createSingletonBankkonto**: erstellt ein Singleton-Bankkonto, falls noch keins existiert.
29. **createSingletonFlipboard**: erstellt ein Singlton-Flipboard, falls noch keins existiert.
30. **createSubjectProcess**: erstellt einen User mit den spezifizierten Account-Daten. Der erstellt User kann sich danach im Spiel einloggen.
31. **createTickHistoryEntry**: erstellt eine TickHistoryEntry mit den spezifizierten Parametern und fügt sie der Singleton-Gameworld hinzu.
32. **createTickSimple**: erstellt ein TickSimple-Objekt mit den spezifizierten Parametern.
33. **deleteIterator**: ein generischer Hilfsprozess um einen Iterator zu erzeugen, mit dem man beliebige Objekte löschen kann. Wird verwendet, wenn das Spiel neugestartet wird.

34. **deleteJob**: entfernt den spezifizierten Job aus dem Spiel.
35. **doTick**: Kernprozess, der jeden Tick ausgeführt wird und zahlreiche Unterprozesse aufruft.
36. **doTickInJobMarket**: wird jeden Tick ausgeführt. Generiert zufällig neue und entfernt zufällig alte Jobs im/aus dem JobMarket.
37. **getAllJobs**: gibt alle im System existierenden Jobs zurück.
38. **getAllJobsInJobMarket**: gibt alle sich im spezifizierten JobMarket befindenden Jobs zurück.
39. **getAllJobsInServiceCenter**: gibt alle sich im Singleton-ServiceCenter befindenden Jobs zurück.
40. **getAllServers**: gibt alle sich in der spezifizierten GameWorld befindenden Server zurück.
41. **getCurrentBankAccountValue**: gibt den aktuellen Kontostand zurück.
42. **getCurrentTick**: gibt den aktuellen Tick im Singleton-TickSimple zurück.
43. **getOptions**: gibt das Singleton-Objekt mit den Spieleinstellungen zurück. Falls keins existiert, wird es erstellt.
44. **getRandomJobName**: gibt einen zufälligen Namen zurück.
45. **getSingletonActualTickHistoryEntry**: gibt die Singleton-TickHistoryEntry für den aktuellen Tick zurück.
46. **getSingletonBankAccount**: gibt das Singleton-Bankkonto zurück.
47. **getSingletonFlipboard**: gibt das Singleton-Flipboard zurück.
48. **getSingletonGameWorld**: gibt die Singleton-GameWorld zurück. Falls noch keine existiert, wird eine erstellt.
49. **getSingletonJobMarket**: gibt den Singleton-JobMarket zurück.
50. **getSingletonServiceCenter**: gibt das Singleton-ServiceCenter zurück.
51. **getSingletonTickSimple**: gibt das Singleton-TickSimple-Objekt zurück.
52. **incrementBankAccount**: Inkrementiert das Bankkonto um die spezifizierte (positive oder negative) Summe.
53. **incrementTickSimple**: Inkrementiert den aktuellen Tick um die spezifizierte (positive oder negative) Zahl.
54. **initGameWorld**: erstellt eine neue GameWorld und füllt sie mit den wesentlichen Objekten (JobMarket, ServiceCenter, etc.).
55. **isRolleAdmin**: überprüft, ob der aktuell eingeloggte User die Rolle des Administrators besitzt.

56. **isRolleBerater**: überprüft, ob der aktuell eingeloggte User die Rolle des Beratrers oder des Administrators besitzt.
57. **isRolleFinanzverwalter**: überprüft, ob der aktuell eingeloggte User die Rolle des Finanzverwalters oder des Administrators besitzt.
58. **isRolleTechniker**: überprüft, ob der aktuell eingeloggte User die Rolle des Technikers oder des Administrators besitzt.
59. **payInterest**: falls der aktuelle Kontostand negativ ist, wird ein von den Options spezifizierter Zinssatz abgezogen.
60. **payUpkeep**: berechnet abhängig von der Leistung der Server die Upkeep-Summe und zieht sie vom Bankkonto ab.
61. **powerWorking**: prüft, ob mindestens ein Stromgenerator ausgefallen ist.
62. **randomCrisis**: lässt mit einer gewissen Wahrscheinlichkeit einen zufälligen Server, Kühlung oder Generator ausfallen.
63. **randomServerOuttage**: erzeugt mit einer von der Reliabilität des Servers abhängigen Wahrscheinlichkeit seinen Ausfall.
64. **reduceReliability**: Reduziert die Reliabilität der Server, der Kühlungen und der Generatoren.
65. **removeJobsPastDeadline**: entfernt alle Jobs, deren Deadline unterhalb der spezifizierten Deadline ist, aus dem spezifizierten JobMarket.
66. **removeRandomJobFromJobMarket**: entfernt einen zufälligen Job aus dem spezifizierten JobMarket.
67. **repairCooling**: setzt alle Kühlung auf einen funktionsfähigen Zustand mit maximaler Reliabilität.
68. **repairGenerator**: setzt die Stromgeneratoren auf einen funktionsfähigen Zustand mit maximaler Reliabilität.
69. **repairServer**: setzt den spezifizierten Server auf einen funktionsfähigen Zustand mit maximaler Reliabilität falls die Kühlung und der Stromgenerator funktionsfähig sind.
70. **resetAll**: löscht alle Objekte außer der Singleton-GameWorld und den Options.
71. **resetAndInitGameWorld**: löscht alle Objekte außer der Singleton-GameWorld und den Options und erstellt die wesentlichen Objekte erneut. Wird beim Spiel-Neustart ausgeführt.
72. **setServerWorkingStatus**: setzt die Funktionsfähigkeit aller Server auf den spezifizierten Status.

73. **unassignJobFromServer**: verschiebt den spezifizierten Job vom spezifizierten Server zurück in das ServiceCenter.
74. **updateSingletonOptions**: aktualisiert die Spiel-Einstellungen mit den spezifizierten Werten.

Abbildungsverzeichnis

3.1 Das Planspiel als komplexe Mischform (angelehnt an [12]) 8

7.1 Kern des Modells 35

7.2 Zuordnung von Jobs zu Kunden 38

7.3 Der Offer-Market 39

7.4 Abhängigkeiten zwischen einzelnen Modellelementen 40

7.5 Market und Status-Effekte 41

7.6 Rollen und Kommunikation 42

7.7 Gesamtes Modell 43

7.8 Berater 44

7.9 Fortschritt Techniker 44

7.10 Finanzübersicht 45

7.11 Kundenansicht 46

7.12 Auftragsdetailsansicht 47

7.13 Logansicht 47

7.14 Serveransicht 48

7.15 Zuweisungs- und Statusansicht 48

7.16 Finanzübersicht 49

7.17 Klassendiagramm des Leveleditors 51

7.18 Leveleditor 1 52

7.19 Leveleditor 2 52

7.20 Leveleditor 3 53

8.1 *initGameWorld*: Spielwelt initialisieren 57

8.2 *buyServer*: Server kaufen 58

8.3 *repairGenerator*: Setzt alle Generatoren auf einen funktionsfähigen Zustand 59

8.4 *doTick*: Tick ausführen 60

8.5 *calcGenerator*: Generator-Status berechnen 61

8.6 *calculateJobsInServer*: Berechne Job-Status im Server 62

8.7 *randomCrisis*: Löst eine zufällige Krise aus 63

8.8 *removeJobsPastDeadline*: Entfernt nicht erledigte Jobs 64

8.9 Ausschnitt des Dashboards im GUI-Builder 67

8.10 Teil des Dashboard aus Admin-Ansicht 68

8.11 Detaillierte Ansicht des Job Market 68

8.12 Serverbereich des Technikers 69

8.13 Minispiel für die Reparatur von Komponenten 69

8.14 Aktuelles Guthaben im Bankaccount 70

8.15 Verlauf der Einnahmen und Ausgaben 70

8.16 Kommunikation erfolgt über das Flipboard 70

8.17 Kopieren einer bestehenden Seite 71

8.18 `clone`-Methode der `MiniGameComponent` 71

8.19 Complex-Container einfügen 72

8.20 Einfügen eines anderen Container in die Seite 72

8.21 Ausgabe ProcessOutputGrid des getAllJobsInServiceCenter-Prozesses . . . 73

8.22 Editor des ProcessOutputGrid vom getAllJobsInServiceCenter-Prozess . . . 73

8.23 Minigame 74

Literaturverzeichnis

[1] Ajjane, N.: *Krisenmanagement und Krisenkommunikation in Unternehmen.* Diplomarbeit, Universität Konstanz, 2004.

[2] al., K. et: *Lerntechniken für Erwachsene.* 1978.

[3] Boris;, M. H. K. C. R. M. G.: *Die Planspielmethode als Lernumgebung*, 2001.

[4] Bronner, R.: *Planspiele in der beruflichen Ausbildung*, 2008.

[5] Burr, W.: *IV. Branchen-und Unternehmensstrukturen bei Facility Management-Dienstleistungen in Deutschland.* In: *Markt-und Unternehmensstrukturen bei technischen Dienstleistungen*, S. 43–133. Springer, 2014.

[6] Conor Linehan, Shaun Lawson, M. D. B. K.: *Developing a serious game to evaluate and train group decision making skills.* 2009.

[7] Dortmund, T. U.: *Cinco SCCE Meta Tooling Suite*, Zugegriffen am 25. September 2015.

[8] Foundation, T. E.: *Eclipse*, Zugegriffen am 25. September 2015.

[9] Foundation, T. E.: *Eclipse Modeling Framework (EMF)*, Zugegriffen am 25. September 2015.

[10] Foundation, T. E.: *Graphiti - a Graphical Tooling Infrastructure*, Zugegriffen am 25. September 2015.

[11] Frohme, M.: *Agile Domänenmodellierung für prozessgesteuerte Webanwendungen.* Bachelorarbeit, TU Dortmund, 2013.

[12] Geuting, M.: *Planspiel und soziale Simulation im Bildungsbereich*, 1992.

[13] Gordon, S. und J. Bieman: *Rapid Prototyping: Lessions Learned* `http://www.cs.colostate.edu/~bieman/Pubs/GordonBiemanSW95.pdf`, Zugegriffen am 05. Oktober 2015.

[14] Group, O. M.: *Business Process Model and Notation*, Zugegriffen am 08. Oktober 2015.

[15] Group, O. M.: *Unified Modeling Language^{TM} (UML®) Resource Page*, Zugegriffen am 08. Oktober 2015.

[16] John Barbuto, R. S.: *Motivation sources inventory: development and validation of new scales to measure an integrative taxonomy of motivation. In: Psychological Reports, S. 1011–1022*, 1998.

[17] Lehmann, A.: *Dienstleistungsbeziehungen zwischen Kunde und Unternehmen*. Handbuch Dienstleistungsmanagement, Wiesbaden, S. 827–842, 1998.

[18] Micahel, David; Chen, S.: *Serious Games. Games That Educate, Train, And Inform*, 2006.

[19] Naujokat, S., L.-M. Traonouez, M. Isberner, B. Steffen und A. Legay: *Domain-Specific Code Generator Modeling: A Case Study for Multi-faceted Concurrent Systems*. In: *Part I of the Proceedings of the 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change - Volume 8802*, S. 481–498, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[20] Niedermayr, C.: *Hauptseminar Automotive Software Engineering Testen, Rapid Prototyping und X - in the loop* `http://www4.in.tum.de/lehre/seminare/hs/WS0405/automotive/Ausarbeitung_Christoph_Niedermayr.pdf`, Zugegriffen am 05. Oktober 2015.

[21] Pierce, J. C. K. B. D.: *Pervasive negative effects of rewards on intrinsic motivation. The myth continues. In: The Behavior Analyst. 24, Nr. 1, S. 1-44, 2001 http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2731358/pdf/behavan00009-0003.pdf, Zugriffszeitpunkt 21.02.2015,.*

[22] Psychology, University of Graz, D. A. Michael D. Kickmeier-RustAffiliated with Lancaster UniversityDepartment of: *Emergent Design: Serendipity in Digital Educational Games*. 2009.

[23] Spitzer, B.: *Modellbasierter Hardware-in-the-Loop Test von eingebetteten elektronischen Systemen*. Dissertation, Karlsruhe, Universität Fredericiana, 2001.

[24] Steffen, B., T. Margaria, R. Nagel, S. Jörges und C. Kubczak: *Model-driven Development with the jABC*. In: *Proceedings of the 2Nd International Haifa Verification Conference on Hardware and Software, Verification and Testing*, HVC'06, S. 92–108, Berlin, Heidelberg, 2007. Springer-Verlag.