Hochschule München
University of Applied Sciences – München

Fakultät für Informatik und Mathematik

Studiengang Informatik

Bachelor-Arbeit

Design und Implementierung einer Software zur Verwaltung von Flipperturnieren

Design and implementation of a pinball tournament administration software

Prüfer:	Prof. Dr. Axel Böttcher
Verfasser:	Christoph Ruiss
Matrikel-Nr.:	01955499
Abgabedatum:	04. März 2009

Christoph Ruiss

Design und Implementierung einer Software zur Verwaltung von Flipperturnieren

Kurzfassung

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung einer Software zur Verwaltung von Flipperturnieren.
Über eine kurze Erklärung des Flipperspiels und der Gegenüberstellung von drei verschiedenen Turniersystemen, wird der Leser mit der Thematik vertraut gemacht.
Die entwickelte Software setzt das so genannte Schweizer System um, welches exakt beschrieben und anhand eines Beispiels anschaulich gemacht wird.
Die technischen Vorraussetzungen für die Software, sowie deren Aufbau, Architektur und Funktionen werden detailliert erläutert.
Die Software wurde als Web-Anwendung auf Basis von J2EE und Tomcat realisiert.
Der mathematische Kern des umgesetzten Turniersystems wird ausführlich beschrieben.
Aufgetretene Probleme sind aufgeführt und erläutert.
Am Ende der Arbeit wird das diskutierte Turniersystem beispielhaft in der Deutschen-Fußball-Bundesliga eingesetzt.

Christoph Ruiss

Design und Implementierung einer Software zur Verwaltung von Flipperturnieren

1. Motivation

Im Raum München existiert eine große Fangemeinde der Flipper-Spielautomaten. Auch fast zwanzig Jahre nach der Hochzeit dieses Geschicklichkeitsspieles gibt es viele begeisterte Spieler, die ihre Spielstärke in regelmäßigen Abständen auf Turnieren messen.
Einzige Voraussetzung für die Durchführung eines Flipperturniers ist eine geeignete Räumlichkeit, in der sich genügend Flipper befinden um ein davor ausgewähltes Turniersystem zu spielen.

Abbildung 1: Die "Pinball-Lounge"

Durch den Zusammenschluss einiger engagierter Liebhaber gibt es im Großraum München mittlerweile drei Örtlichkeiten, die über eine ausreichende Anzahl an Flippern verfügen. Die je über fünfzig Spielautomaten sind genug um auch Turniere mit mehr als zwanzig Teilnehmern perfekt durchführen zu können. Diese äußeren Umstände haben zur Folge, dass derartige Turniere regelmäßig stattfinden.

Um den reibungslosen Ablauf von Flipperturnieren zu gewährleisten, setzt die Turnierleitung Software ein. Damit wird die Leitung bei der Verwaltung der Spieler, der eingesetzten Flippergeräte und dem gesamten Ablauf der Veranstaltung unterstützt.

Jedes Turniersystem hat einen gemeinsamen Punkt: Das Spiel am Flipperautomaten.
Von diesen Geräten gibt es schier unzählige verschiedene Typen. Angefangen bei rein mechanischen Geräten aus den sechziger Jahren bis hin zu den modernen, elektronisch gesteuerten Maschinen, welche bis heute produziert werden.

Ziel des Spieles ist, eine Stahlkugel mit Hilfe von kleinen, beweglichen Armen auf bestimmte Ziele unterschiedlicher Wertigkeit zu schießen und dabei möglichst viele

Abbildung 2: Addams Family Flipper

Punkte zu sammeln. Fällt die Kugel am Rand oder in der Mitte zwischen den beiden Flipperarmen ins Aus, ist sie verloren. Um das Spiel fortzusetzen, muss eine neue Kugel in das Spielfeld eingeschossen werden.

Ein einzelnes Spiel am Flipper kann von einem bis zu vier Spielern bestritten werden. Im Normalfall erhält jeder Spieler drei Kugeln, im Flipper auch Bälle genannt[1], gespielt wird abwechselnd, das heißt zuerst spielen alle Spieler Kugel Eins, dann Zwei, dann Drei. Das Erreichen möglichst vieler Punkte ist keineswegs reine Glückssache: Viel Übung und Fingerspitzengefühl machen es sehr wohl möglich die kleine Stahlkugel gezielt auf die kleinsten Trefferfelder zu schießen. Ebenso ist die genaue Kenntnis über das Gerät hilfreich, um in jeder Spielsituation zu wissen, durch welche Trefferflächen sich im jeweiligen Moment die meisten Punkte erreichen lassen.

Der Spieler, welcher nach der letzten Kugel die meisten Punkte gesammelt hat, ist in den meisten Turniervarianten der Sieger dieser Begegnung. In anderen Varianten geht es primär um die erreichte Anzahl von Punkten, der direkte Vergleich hat in dem Fall keine Auswirkung. Auf einige Varianten von Turnieren werde ich im Kapitel 2 eingehen.

In dieser Arbeit soll eine Software für die Abwicklung von Flipperturnieren im so genannten Schweizer System erstellt werden, mit deren Hilfe der komplette Turnierablauf abgebildet werden kann. Die Verwaltung von Turnieren soll möglichst einfach zu handhaben sein und alle im Turnierverlauf auftretenden Situationen abgebildet werden.

[1] Flipper können auch auf fünf oder mehr Kugeln pro Spiel gestellt werden. Ebenso ist es möglich über entsprechende Spielleistungen Extrabälle zu erlangen, was aber für Turnierspiele meist deaktiviert ist um die Spielzeit besser abschätzen zu können.

2. Turniere

Es gibt unzählige verschiedene Arten von Turniertypen. Alle haben das Ziel, dass der beste Spieler zum Schluss auf Platz Eins steht. Um dies zu gewährleisten wäre nahe liegend, jeden Teilnehmer gegen jeden antreten zu lassen. Ein Rundenmodus, wie er zum Beispiel in der deutschen Fußball-Bundesliga praktiziert wird, ist für ein zeitlich begrenztes Turnier nicht praktikabel. Auch ohne Hin- und Rückspiel wären enorm viele Runden (im Fußball Spieltage) nötig, bis jede Mannschaft gegen jede angetreten ist. Zwar dauert ein Match am Flipper keine 90 Minuten, dafür sind die Teilnehmerfelder häufig deutlich größer als 18. Ebenso ist die Dauer eines Turniers meist nur auf einige Stunden und nicht auf ein halbes Jahr angesetzt.

Das „perfekte" Turniersystem gibt es nicht. Alle Varianten haben Vor- und Nachteile – die meisten davon sind subjektiver Art. Das vorzeitige Ausscheiden spielschwächerer Spieler mag für den Einen ein Vorteil sein, andere empfinden dies als negativ. Auch der Einfluss, der tatsächlich erspielten Punkte der jeweiligen Wertungsgeräte auf die Platzierung, ist Geschmacksache jedes einzelnen Turnierteilnehmers.
Um einen kurzen Einblick in diese Thematik zu geben, werde ich drei dieser Turniertypen beschreiben.

2.1. Das K.O.-System

Ein Flipperturnier im reinen K.O.[2]-System habe ich persönlich noch nie gespielt. Theoretisch möglich wäre dies, allerdings ist der Faktor Glück in einem solchen System extrem hoch.
In einem K.O.-Turnier treten in jeder Runde die Spieler in Paare aufgeteilt gegeneinander an. Der Verlierer einer Begegnung scheidet aus, der Gewinner bekommt in der nächsten Runde einen neuen Gegner zugewiesen. Eine einzige Niederlage bedeutet das Aus, im Finale stehen sich zwei Spieler gegenüber, die bis zu diesem Zeitpunkt alle Begegnungen gewonnen haben.

[2] K.O. steht für Knock-Out, also „herausschlagen" des Gegners aus dem Turnier

"*Losing a single game, for which an unaccountable stroke of ill fortune may be responsible, is enough to cause elimination from a knock-out tournament if each match consists of only one game.*"[3]
Bei exakt einem Wertungsspiel pro Runde ist die Wahrscheinlichkeit, dass viele Top Spieler durch ein einziges schlechtes Spiel schon früh aus dem Turnier ausscheiden sehr hoch. Ein gewisses Maß von Glück und Zufall hat auch beim besten Spieler Anteil am Ausgang eines Flipperspiels. Dem Ziel, am Ende einer solchen Veranstaltung noch die hochklassigsten Teilnehmer im Feld zu haben, kommt man nur durch Modifizierungen dieses Systems nahe.

Zum einen besteht die Möglichkeit statt einem einzelnen Wertungsspiel mehrere auszutragen. Wer zuerst zwei von drei Spielen gewinnt („best-of-three"), geht als Sieger aus der Runde. Ein einzelnes verlorenes Spiel hat somit nicht mehr den extremen Einfluss. Ein überlegener Gegner hat nach einem verlorenen Einzelspiel noch immer eine Chance die Runde für sich zu entscheiden und im Turnier zu verbleiben. Die Spieldauer jeder einzelnen Runde erhöht sich allerdings deutlich. Statt eines Spieles müssen mindestens zwei, wenn nicht drei Partien pro Runde bestritten werden.

Ein weiterer Modus um die spielstärksten Teilnehmer möglichst lange im Turnier zu halten ist, die Begegnungen so anzusetzen, dass die beiden stärksten Teilnehmer erst im Finale aufeinander treffen.
Ein solches System setzt allerdings voraus, dass sämtliche Teilnehmer in Leistungskategorien eingeteilt werden. Dies könnte im Flippersport über die Position in der Weltrangliste[4] geschehen.

Der Nachteil des K.O. Systems ist das schnelle Ausscheiden der unterlegenen Spieler. Nach jeder Runde verlässt die Hälfte der Spieler das Turnier. Der Spaß am gegeneinander Spielen am Flipper, ist so für einen Großteil der Teilnehmer von sehr kurzer Dauer.

---

[3] [7] Kapitel 2.2

[4] Unter bestimmten Voraussetzungen können auf Flipperturnieren (wie in vielen anderen Sportarten) Weltranglistenpunkte für die bestplatzierten Spieler vergeben werden [11]

Vorteilhaft ist aber, dass die Dauer des Turniers relativ genau abgeschätzt werden kann, außerdem ist die insgesamt benötigte Zeit für den kompletten Turnierablauf extrem kurz. In diesem Modus kann ein sehr großes Teilnehmerfeld in wenigen Runden bis zu einem Finalspiel fortschreiten.

2.2. Der Family Mode

Turniere im so genannten „Family Mode“ werden meist in zwei Teile geteilt: eine Vorrunde und eine Finalrunde.
Die Vorrunde bestreitet jeder Spieler in einer festen Gruppe von drei oder vier Spielern, welche zusammen vorgegebene Flipper in frei wählbarer Reihenfolge bespielen. Ausschlaggebend für die Punktevergabe ist nicht der direkte Vergleich mit den anderen Gruppenmitgliedern, sondern die erspielten Scores am jeweiligen Gerät im Vergleich zum gesamten Turnierteilnehmerfeld. Der beste Spieler des gesamten Teilnehmerfeldes wird also mit den meisten Punkten für dieses eine Wertungsgerät belohnt. Im Normalfall erhält der beste Spieler pro Gerät die Anzahl der Teilnehmer als Punkte gutgeschrieben, der zweitbeste einen Punkt weniger, bis zum schlechtesten Spieler, welcher dann noch einen Punkt erhält.
Sind alle vorgegebenen Maschinen gespielt, rückt das Topfeld in die Finalrunde vor. Diese zweite Runde wird meist im K.O.-System bestritten, sodass sich das Feld immer weiter reduziert, bis am Ende ein Finalspiel um den Gesamtsieg ansteht.
Im eigentlichen „Family Mode“ wird nur die Vorrunde gespielt, die Kombination mit einer zweiten Runde in einem anderen System ist allerdings üblich und deshalb mit aufgeführt.

Nachteile des Family Mode:
Der Großteil der Teilnehmer scheidet nach der Vorrunde aus, nach jeder K.O. Runde müssen weitere Spieler das Turnier verlassen. Außerdem kommt es oft zu langen Wartezeiten, da alle Spieler die gleichen Geräte zu spielen haben und an Flippern mit langer Spielzeit sich zum Ende viele wartende Gruppen sammeln.

Vorteile des Family Mode:
Ein Vorteil ist, dass der direkte Vergleich in der Vorrunde nicht zählt. Ein enorm spielstarker Mitspieler in der Gruppe mag jedes Spiel gewinnen, eine gute Platzierung seiner Gruppenpartner ist damit aber nicht ausgeschlossen, wenn diese im Vergleich zum gesamten Teilnehmerfeld ebenfalls überdurchschnittliche Ergebnisse erzielen.
Dass alle Teilnehmer die gleichen Flipper bespielen – extrem schwer eingestellte oder unbeliebte Geräte treffen alle Teilnehmer und belasten nicht nur eine kleine Anzahl von Spielern –dient der Ausgeglichenheit und der Fairness.

2.3. Das Schweizer System

Ein weiterer Turniertyp ist das Schweizer System. Christoph Gerlach schreibt in seiner Arbeit über die Ursprünge dieses Systems folgendes: „*Dr. J. Muller führte 1895 ein modifiziertes K.O.-System auf einem Züricher Schach-Turnier ein, das die Grundlage für das Schweizer System, wie wir es heute kennen, bildete.*“ [5]
Auch wenn die Wurzeln dieses Turniertyps im Schach zu finden sind, lässt es sich hervorragend auf Flipperturniere anwenden.

Ein Turnier wird auf eine zuvor bestimmte Anzahl von Runden ausgelegt. Zu Beginn jeder Runde werden alle Teilnehmer in Paare eingeteilt, welche dann jeweils gegeneinander antreten. So bekommt jeder Spieler pro Runde einen neuen Gegner zugewiesen. Der Gewinner einer Partie bekommt Punkte gutgeschrieben, der Verlierer erhält keine Punkte.
Das Schweizer System befolgt für die Bilder von Spielerpaaren im Wesentlichen drei Regeln:

1. Spielerpaarungen werden nicht wiederholt
2. Es spielen immer punktgleiche Spieler gegeneinander (falls wegen Regel 1 nicht möglich, Spieler mit möglichst geringer Punktedifferenz)
3. Nach einer definierten Anzahl von Runden gewinnt der punktstärkste Spieler

[5] [4], Kapitel 1.4

Bei einer ungeraden Anzahl von Teilnehmern wird das Feld durch einen „Dummy“ Spieler aufgefüllt. Partien gegen diesen fiktiven Gegner werden immer als gewonnen gewertet, jeder Spieler kann nur einmal diesen Punkt bekommen.
Die in Punkt 2 angesprochene, durchaus komplizierte Bildung von Gegnerpaaren in diesem System wurde anfangs in Münchner Flipperkreisen durch eine Software bewältigt, die im Rahmen der Diplomarbeit von Christoph Gerlach 1994 entwickelt wurde. [6]
Sein Programm dient nicht zur Durchführung von Flipper Turnieren, sondern ist als System für Go-Turniere[7] gedacht. Da die Software ursprünglich nicht für Flipperturniere optimiert ist, gilt es für die Turnierleitung und auch die Teilnehmer viele Umständlichkeiten in Kauf zu nehmen, und es ist viel Handarbeit bei der Verwaltung des Turniers erforderlich. So muss die Verwaltung von Flippergeräten und deren Zulosung zu den einzelnen Spielen komplett Software unabhängig bewerkstelligt werden.

Vorteile des Schweizer Systems:
Die Vorteile dieses Turniertyps sind unter anderem, dass sämtliche Teilnehmer bis zum Ende am Turnier teilnehmen und nicht, wie zum Beispiel beim K.O.-System, nach einer verlorenen Runde ausscheiden. Ein weiterer positiver Aspekt ist, dass für jede Runde neue Gegnerpaare gebildet werden, so kann sich jeder Mitspieler pro Runde mit einem anderen Kontrahenten messen.

Nachteile des Schweizer Systems:
Nachteilig ist das außer Acht lassen der am Flipper eigentlich erspielten Punkte. Ob man mit dem tausendfachen an Punkten den Gegner deklassiert, oder gerade mal einen Punkt mehr erspielt hat, macht keinen Unterschied, da jeder Sieg mit der gleichen Anzahl von Siegpunkten prämiert wird, egal wie hoch dieser ausgefallen ist.

---

[6] Christoph Gerlach, Ein MacMahon-Losungsprogramm für Go-Turniere unter Benutzung von Maximum Weight Perfect Matching

[7] Go ist ein altes chinesisches Brettspiel

An dieser Stelle darf eine Person nicht unerwähnt bleiben: Marco Wopkes. Er hatte als Organisator diverser Turniere und der Münchner Flipper Liga Mopl[8] als Erster versucht, das Schweizer System auf Flipper-Turniere anzuwenden. Auch von ihm existiert eine angepasste Software für die Durchführung von Turnieren, seinen modifizierten Modus bezeichnet er als „McWopkes“ Modus[9].

Nachdem sich das Schweizer System großer Beliebtheit erfreut, kam es zur Entscheidung ebenfalls eine für Flipper-Turniere optimierte Anwendung zu entwickeln, welche bei der Paarungsbildung exakt arbeitet und leicht erweiterbar ist.

[8] Munich Open Pinball League, 2003 gegründete, offene Münchner Flippermeisterschaft

[9] [10] Kapitel 2

3. Das Schweizer System im Detail

Gegenstand dieser Arbeit ist, eine Software für die Verwaltung von Turnieren im Schweizer System zu entwickeln. Aus diesem Grund werde ich diesen Turniertyp in Ablauf, Punktevergabe und der zu bildenden Gegnerpaare detailliert beschreiben.

3.1. Turnierablauf

Schweizer System Turniere laufen nach dem im Folgenden beschriebenen Schema ab.
Ein Turnier wird in einzelne Runden gegliedert. Die Anzahl der Runden darf die Anzahl der Mitspieler nicht übersteigen. Es muss gewährleistet werden, dass jeder Spieler im Verlauf des Turniers nur maximal einmal gegen denselben Spieler antritt.
Die Anzahl der Runden hängt von der Größe des Teilnehmerfeldes sowie der verfügbaren Zeit für die Veranstaltung ab. Mehr gespielte Runden ergeben ein exakteres Endergebnis, aber durch das Spielsystem wird forciert, dass die Tabelle bereits nach wenigen Runden große Aussagekraft über das Kräfteverhältnis der Kontrahenten hat.
Vor jeder Runde werden die Begegnungen berechnet. Eine Begegnung besteht aus zwei Teilnehmern, die in der betreffenden Runde auf einem zugelosten Flipper gegeneinander antreten. Der Spieler, welcher nach Beendigung des Spieles mehr Punkte erspielt hat, ist der Gewinner der Begegnung.
Sind sämtliche Partien einer Runde beendet, kann die Tabelle neu berechnet werden. Die siegreichen Spieler bekommen Punkte gutgeschrieben, die unterlegenen Teilnehmer behalten ihren Punktestand.
Die Ergebnisse der Vorrunden fließen in die Berechnung der Paarungen der Folgerunden ein.

3.2. Punktevergabe und Rangfolge

Zur Berechnung der Tabelle, bzw. der Platzierung der einzelnen Spieler, kommen drei Kriterien zum Einsatz:

- die Anzahl der bisher erspielten Punkte
- die Anzahl der bisher erspielten Punkte der Gegner (SOS)
- die Anzahl der bisher erspielten Punkte der Gegner der Gegner (SOSOS)

Die Punkte jedes Spielers werden nach jeder beendeten Runde summiert. Spieler mit mehr erspielten Punkten, stehen an einem höheren Platz in der Tabelle.
Der nach der letzten Runde punktstärkste Spieler steht in der Tabelle auf Platz eins und ist Gewinner des Turniers.

3.2.1. Erstkriterium: Punkte

Das Hauptkriterium sind die Punkte. Diese berechnen sich aus den Startpunkten und den erspielten Punkten durch Siege und Unentschieden.

Die Punkte der einzelnen Spieler berechnen sich nach dieser Formel:

$$Punkte = Starterpunkte + \sum Siege * Siegpunkte + \sum Unentschieden * Unentschiedenpunkte$$

Startpunkte können jedem Turnierteilnehmer vor Turnierbeginn zugewiesen werden. Wird ein Turnier zum Beispiel als Finale einer Saison gespielt, ist es sinnvoll den Spitzenreitern bereits zu Anfang ein bestimmtes Punkteguthaben zu gewähren. Bereits erbrachte Leistungen aus Saison-Spielen haben somit Einfluss auf das Finalturnier.
Ansonsten sind die Startpunkte sämtlicher Spieler null.

Zu den Startpunkten werden die erspielten Punkte des laufenden Turniers addiert.
Die erspielten Punkte ergeben sich aus der Anzahl der Siege und der Anzahl der Unentschieden, welche jeweils mit einer definierten Menge von Punkten multipliziert werden. Im Normalfall wird für einen Sieg ein Punkt und für ein Unentschieden ein halber Punkt vergeben. Diese Werte können in den Einstellungen frei angepasst werden,

so dass zum Beispiel auch ein Sieg mit drei Punkten und ein Unentschieden mit einem Punkt gewichtet werden können; dies hätte eine Aufwertung der Siege zu Folge.

3.2.2. Zweitkriterium: SOS

Bei Punktgleichheit kommt das zweite Kriterium zum Einsatz: die so genannten SOS Punkte. SOS steht hier für „Sum of Opponent's Scores" und beschreibt die Anzahl der Punkte, welche die Gegner des Spielers im laufenden Turnier gesammelt haben.
Die Punkte sämtlicher Gegner, gegen welche ein Spieler im Verlauf eines Turniers angetreten ist, werden für den SOS Wert summiert. Ob die Partien gewonnen oder verloren wurden, hat keine Bedeutung.

$$SOS = \sum_{Gegner} Punkte$$

Dieses Kriterium kommt auch bei Schachturnieren unter der Bezeichnung „Buchholzwertung" zum Einsatz.

3.2.3. Drittkriterium: SOSOS

Als drittes Kriterium kommt bei Gleichheit von Punkten und SOS die SOSOS Wertung zum Tragen. Die Abkürzung SOSOS steht für „Sum of Opponent's SOS", dies sind die aufsummierten SOS Punkte der Gegner.

$$SOSOS = \sum_{Gegner} SOS$$

Das Pendant im Schach ist die „verfeinerte Buchholzwertung".

3.2.4. Bedeutung der Kriterien

Die SOS und SOSOS Wertungen bilden die „Stärke“ der Gegner ab, mit denen sich die einzelnen Spieler im Turnierverlauf messen mussten. Durch das Summieren der Gegner Punkte wird das „Niveau“ der Kontrahenten abgebildet, unabhängig davon, welcher Spieler die Begegnung gewinnt. Das bedeutet unter anderem, dass Niederlagen gegen punktstarke Gegner weniger negative Auswirkung haben, als das Verlieren gegen schwache Spieler.
Bei eindeutigen Platzierungen, haben beide Nebenkriterien keine Auswirkung, lediglich bei Punktgleichheit wird nach SOS gewertet. SOSOS entscheidet gar nur bei Gleichheit von Punkten und SOS über die Platzierung.

3.3. Beispiel für die Punktevergabe im Schweizer System

Die Punktevergabe im Schweizer System will ich anhand eines kleinen Beispiel-Turniers mit acht Teilnehmern veranschaulichen.
Die Begegnungen der Runde eins sind zufällig gewählt. Nachdem zu diesem Zeitpunkt noch keine Aussage über die Spielstärke der Teilnehmer getroffen werden kann, wird die Zuordnung der Paare willkürlich gewählt.
An dieser Stelle würden vergebene Startpunkte Einfluss nehmen. Um auch in der ersten Runde möglichst punktgleiche Teilnehmer gegeneinander antreten zu lassen, würden zuerst die durch Startpunkte hochgewerteten Spieler gepaart, so dass Begegnungen hochklassiger Spieler gegen schwächere Gegner zu Beginn vermieden werden. Im kommenden Beispiel wurden keine Startpunkte vergeben, die Begegnungen in Runde Eins sind rein zufällig.

Im folgenden Beispiel sind zuerst die Begegnungen einer ersten Runde in tabellarischer Form (Tabelle 1) dargestellt. Die Gewinner der jeweiligen Partien sind unterstrichen, die daraus resultierende Tabelle werde ich nun kurz erläutern.
In der Tabelle nach Runde Eins (Tabelle 2) sind nun die Sieger der ersten Runde zu erkennen: Diese haben einen Siegpunkt gutgeschrieben bekommen, die Punktekonten der unterlegenen Gegner sind nach wie vor leer.

Genau anders herum verhält es sich mit den Gegnerpunkten (SOS): die Gegner der siegreichen Spieler hatten ihre einzige Begegnung verloren, daher null Punkte welche ihre jeweiligen Konkurrenten als SOS Punkte erhalten.
Ebenso zu erkennen sind nach einer gespielten Runde die SOSOS Punkte. Diese entsprechen nach Runde eins den Siegpunkten – der Gegner eines jeden Gegners ist der Spieler selbst. Nach einer gespielten Runde gibt es noch keine weiteren Gegner deren Punkte aufzusummieren wären.
Da nach einer Runde noch keine exakten Positionen für alle Spieler ermittelt werden können, werden nach allen Kriterien gleichstarke Spieler mit der gleichen Position gewertet. Das ist der Grund, warum nach der ersten Runde vier Spieler auf Position Eins gesetzt sind.

Spieler 1		**Spieler 2**
Alexander	vs.	Thomas
Axel	vs.	Christoph
Sabine	vs.	Conny
Claudia	vs.	Michael

Tabelle 1: Begegnungen Runde 1

Pos	**Spieler**	**Siege**	**SOS**	**SOSOS**	**R 1**	**R 2**
1	Alexander	1	0.0	1.0	5+	-
1	Axel	1	0.0	1.0	5+	-
1	Claudia	1	0.0	1.0	5+	-
1	Conny	1	0.0	1.0	5+	-
5	Sabine	0	1.0	0.0	1-	-
5	Thomas	0	1.0	0.0	1-	-
5	Christoph	0	1.0	0.0	1-	-
5	Michael	0	1.0	0.0	1-	-

Tabelle 2: Tabelle nach Runde 1

An den Begegnungen der Runde zwei ist deutlich zu sehen, dass nun die jeweils siegreichen und unterlegenen Spieler der ersten Runde aufeinander treffen. Die Problematik, der Vermeidung gleicher Begegnungen, ist in Runde zwei noch gering, da definitiv davor keine Begegnungen punktgleicher Spieler stattgefunden haben.

Spieler 1		Spieler 2
Alexander	vs.	Conny
Axel	vs.	Claudia
Sabine	vs.	Christoph
Thomas	vs.	Michael

Tabelle 3: Begegnungen Runde 2

Pos	Spieler	Siege	SOS	SOSOS	R 1	R 2
1	Axel	2	2.0	4.0	4+	5+
2	Conny	2	1.0	6.0	7+	3+
3	Alexander	1	3.0	2.0	6+	2-
4	Christoph	1	2.0	5.0	1-	7+
5	Claudia	1	2.0	4.0	8+	1-
6	Thomas	1	1.0	5.0	3-	8+
7	Sabine	0	3.0	3.0	2-	4-
8	Michael	0	2.0	3.0	5-	6-

Tabelle 4: Tabelle nach Runde 2

Wie in Runde eins werden auch hier die Sieger in der Begegnungsübersicht (Tabelle 3) unterstrichen dargestellt. Die berechnete Tabelle nach Runde zwei (Tabelle 4) ist schon differenzierter als die vorherige – allen Teilnehmern kann bereits ein konkreter Tabellenplatz zugeordnet werden.
Um die Reihenfolge nachvollziehen zu können, sind die beiden rechten Spalten R1 und R2 zu betrachten. Hier werden die jeweiligen Gegner der Spieler angezeigt. Der Spaltentitel bezeichnet jeweils die Runde, die eingetragene Zahl die aktuelle Position des Gegners und das Plus oder Minus Zeichen sagt aus, ob die Begegnung gewonnen oder verloren wurde. Ein „4+“ in der Spalte R1 bedeutet, dass der Spieler dieser Zeile in Runde eins gegen den aktuell viertplatzierten Spieler gewonnen hat.

Die Berechnung von Punkten, SOS und SOSOS wird nun schon ein wenig aufwändiger. Der viertplatzierte Spieler „Christoph“ dient als Beispiel:
In Runde eins unterlag dieser gegen den erstplatzierten Spieler, Punkte gibt es dafür keine, allerdings werden die Punkte des Gegners *(+2)* dem SOS Wert gutgeschrieben und die SOS Punkte *(+2)* dem SOSOS Wert. Diese Werte mit der Tabelle der ersten Runde zu vergleichen macht keinen Sinn, da nur die Punkte selbst „stabil“ sind, die der Gegner sich von Runde zu Runde ändern.

Der Spieler „Christoph“ errang in Runde zwei einen Sieg und damit einen Punkt gegen die Spielerin auf Position sieben. Wieder werden deren Punkte *(+0)* auf den SOS Wert und deren SOS Punkte *(+3)* auf den SOSOS Wert addiert.

Somit sind alle Werte ausgelesen und es können die Summen der drei Kriterien für den Beispielspieler berechnet werden:

$Punkte = 0 + 1 = 1$

$SOS = 2 + 0 = 2$

$SOSOS = 2 + 3 = 5$

Diese Werte finden sich in der Tabelle nach Runde zwei wieder.

Nach Beendigung der letzten Runde hat derjenige Spieler das Turnier gewonnen, welcher auf Platz Eins der Tabelle steht – also die meisten Punkte gesammelt hat.
Bei Punktgleichheit entscheiden als Zweit- und Drittkriterium SOS und SOSOS.
Zwei Spieler teilen sich eine Position bei Gleichheit von Punkten, SOS und SOSOS. Dieser Fall ist im Beispiel nach Runde eins aufgetreten, wird aber mit zunehmender Rundenanzahl immer unwahrscheinlicher.

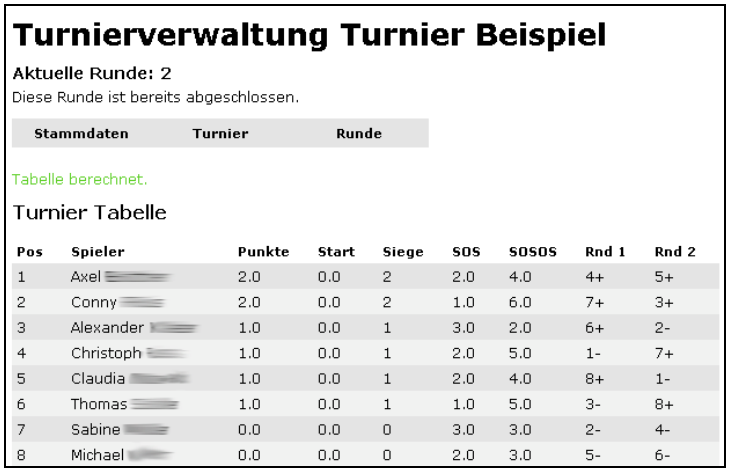
Das Herauslesen von Begegnungen der nächsten Runde aus der Tabelle wird von Runde zu Runde schwieriger – ist ab der dritten Runde kaum noch ohne Software-Unterstützung zu bewältigen.

4. Funktion der Software

Die Software soll den Verlauf eines Turniers im Schweizer System durchgängig abbilden.

Sie enthält die Verwaltung von Spielern, Flippergeräten und Turnieren, sowie die Zuweisung von Spielern und Geräten zu Turnieren und dessen Runden. Weiter bildet es den Turnierverlauf ab. Das Wechseln von Runde zu Runde, die automatische Bildung von Gegnerpaaren, das Zulosen von Flippern, die Eingabe der Ergebnisse und das Berechnen der Tabelle werden von der Software unterstützt.

Auch Sonderfälle wie eine ungerade Anzahl von Spielern, das Definieren von Starterpunkten oder das Verlassen eines Spielers des laufenden Turniers sollen korrekt gehandhabt werden.



Turnierverwaltung Turnier Beispiel

Aktuelle Runde: 2

Diese Runde ist bereits abgeschlossen.

Stammdaten Turnier Runde

Tabelle berechnet.

Turnier Tabelle

Pos	Spieler	Punkte	Start	Siege	SOS	SOSOS	Rnd 1	Rnd 2
1	Axel	2.0	0.0	2	2.0	4.0	4+	5+
2	Conny	2.0	0.0	2	1.0	6.0	7+	3+
3	Alexander	1.0	0.0	1	3.0	2.0	6+	2-
4	Christoph	1.0	0.0	1	2.0	5.0	1-	7+
5	Claudia	1.0	0.0	1	2.0	4.0	8+	1-
6	Thomas	1.0	0.0	1	1.0	5.0	3-	8+
7	Sabine	0.0	0.0	0	3.0	3.0	2-	4-
8	Michael	0.0	0.0	0	2.0	3.0	5-	6-

Abbildung 3: Tabellenanzeige der Software

4.1. Technische Umsetzung

Besonders bei größeren Turnieren kommt es immer wieder zu Wartezeiten, da die zentrale Eingabe von Spielergebnissen einiges an Zeit benötigt.

Deshalb soll die Software problemlos von mehreren Plätzen aus bedient werden können. Aus diesem Grund bietet es sich an eine Web-Applikation zu erstellen. Die Client/Server-Architektur muss somit nicht extra entwickelt werden, ebenso steht den

einzelnen Clients die Wahl des Betriebssystems frei – zumindest solange dieses netzwerkfähig ist und über eine funktionstüchtige Web-Browser-Applikation verfügt.

Die persistente Speicherung der Turnierdaten soll in einer Datenbank erfolgen. Hier wurde das weit verbreitete, freie Datenbanksystem „MySQL“[10] gewählt.
Vorteile der Datenhaltung in der Datenbank sind, dass problemlos viele Clients parallel zugreifen können und der Client außer der Datenstruktur keinerlei Informationen über die Art der Datenspeicherung benötigt.
So ist es auch denkbar, die Ergebnistabellen von Turnieren zur Veröffentlichung auf Webseiten live aus der Datenbank zu generieren, statt diese als statischen Text auf eine Homepage zu kopieren. Hierfür erforderlich sind natürlich Zugriffsrechte auf die Datenbank, die Serverumgebung der Webseite allerdings ist frei wählbar, ob dies eine PHP[11], ASP[12] oder ebenfalls eine Java-Anwendung ist, bleibt dem Entwickler überlassen.

Die Applikation selbst wird in Java[13] als Web-Anwendung nach der Servlet Spezifikation entwickelt. Als Entwicklungsumgebung kommt die freie Software „Eclipse“[14] zum Einsatz, als Servlet-Container „Tomcat“[15]. Der Zugriff auf die Datenbank erfolgt über JDBC – den „Java Database Connector“.
Die Anwendung wurde mit der Java Version 1.6 entwickelt, Tomcat kam in der Version 5.5 zum Einsatz. Für die Implementierung verwendete ich die Entwicklungsumgebung „Eclipse Platform 3.4.1 - Ganymede“, welche sämtliche Plugins und Funktionen für die Entwicklung von Java basierten Web-Anwendungen enthält.

---

[10] Datenbankmanagement System unter General Public License (GPL), 2008 von Sun gekauft

[11] PHP: Hypertext Preprocessor, Skriptsprache der PHP Group für Webapplikationen, freie Verwendung (PHP Licence)

[12] Active Server Pages, serverseitige Sprache zur Webseitenerstellung von Microsoft

[13] Programmiersprache und Laufzeitumgebung von Sun unter General Public License (GPL)

[14] Open Source Framework unter der Eclipse Public License (EPL), vor allem als Java Entwicklungsumgebung genutzt

[15] Servlet Container, zur Ausführung von Java Code auf Webservern der Apache Software Foundation, freie Verwendung (Apache-Lizenz)

Die Anwendung ist auf beliebigen Webservern mit passendem Servlet Manager und installierter Java Umgebung ausführbar. Für die Darstellung der Seiten wird ein Web Browser benötigt.

4.2. Architektur

Die Softwarearchitektur orientiert sich am Model-View-Controller Konzept, welches in Kai Büssaus Eclipse WTP Buch in Kapitel 3 beschrieben wird.

„*In diesem Zusammenhang ist es üblich, Servlets für die Steuerung (Controller) der Anwendung, JSPs*[16] *für die Darstellung (View) und Java Beans für die Datenhaltung und Geschäftslogik (Model) einzusetzen [...]. Eine solche Architektur wird als Model View Controller-Architektur bezeichnet.*“ [17]

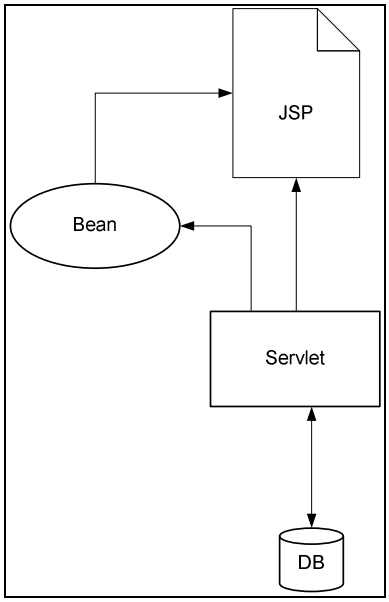


Abbildung 4: Architektur

Im Fall der Turniersoftware ist die Datenhaltung „doppelt“, die Objekte werden nicht nur in den Java Beans gehalten, sondern persistent in die Datenbank gespeichert. Die

[16] Java Server Pages, Technik um auf dem Webserver ausgeführten Javacode in HTML Seiten einzubetten

[17] [1], Kapitel 3.5.4

Speicherung erfolgt manuell, auf die Benutzung eines Frameworks, welches die Synchronisation der Java-Objekte mit der Datenbank automatisiert, wurde verzichtet.
So sind in den Controller Servlets nicht nur die Aktualisierung der Objekte, sondern parallel dazu auch die SQL-Abfragen für die MySQL Datenbank implementiert.

Die Abbildung 4 verdeutlicht, dass ausschließlich die Servlets mit der Datenbank kommunizieren. Die JSP-Seiten greifen über die von den Servlets erzeugten Bean-Objekte auf die Daten zu.

Da diese Arbeit mein erster Kontakt zu Java Web-Anwendungen war, ist bewusst auf bestehende Frameworks verzichtet.
Für die erste Anwendung wollte ich möglichst viel Verständnis für die Technik erlangen, ohne, dass diese im Datenbank Bereich zum Beispiel von Hibernate und in der Präsentationsschicht von zum Beispiel Java-Server-Faces oder Struts „verdeckt“ wird.

4.2.1. Datenbankmodell

Für die dauerhafte Speicherung von Turnieren schreibt die Anwendung alle relevanten Daten in eine MySQL-Datenbank. Beim Laden eines Turniers werden diese wieder ausgelesen.
Das folgende Entitäten-Modell bildet sämtliche Anforderungen ab:

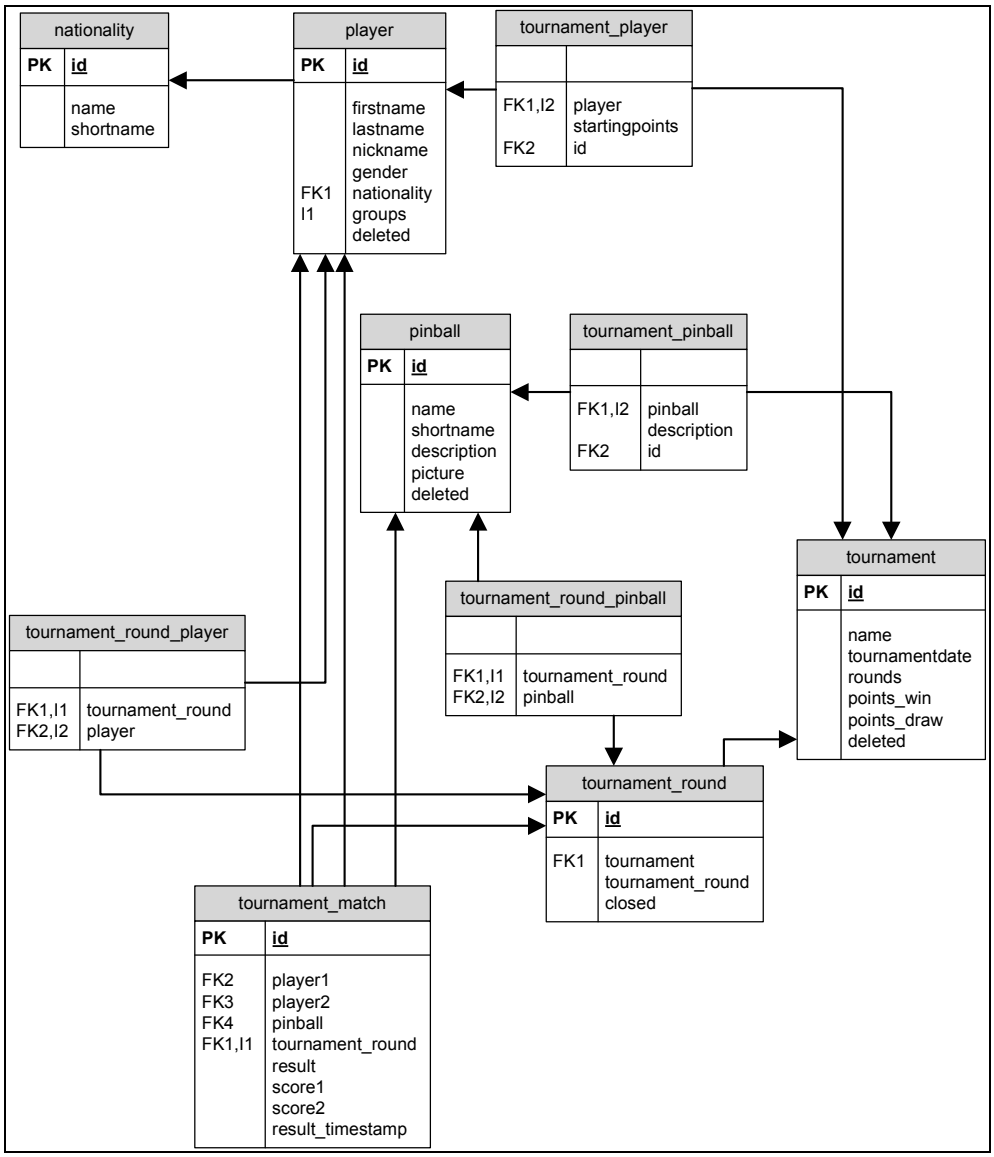


Abbildung 5: Datenbank Modell

Die Zuordnung von Spielern „*player*“ und Flippern „*pinball*“ zu Turnieren „*tournament*“ und Runden „*tournament_round*“ scheint im ersten Moment ein wenig unübersichtlich. Dafür verantwortlich ist eine Art doppelte Zuordnung. Spieler und

Flipper welche einem Turnier zugewiesen sind, finden sich in den Tabellen „*tournament_player*“ und „*tournament_pinball*“ wieder. Die Möglichkeit Spieler oder Flipper während eines laufenden Turniers aus einzelnen Runden herauszunehmen, macht es nötig auch die Zuweisung zu einzelnen Turnierrunden zu speichern. Die Zuweisung von Spielern und Flippern zu einzelnen Turnierrunden ist in den Tabellen „*tournament_round_player*“ und „*tournament_round_pinball*“ gespeichert.
Die Begegnungen bzw. Spiele sind in „tournament_match“ abgebildet. Eine Begegnung von zwei Spielern an einem Flippergerät ist allerdings nicht mit dem Turnier sondern mit einer Turnier Runde verknüpft, so dass auch diese Zuordnung nicht verloren geht.

4.2.2. Funktion der Servlets

Die Servlets haben die Funktion des „Controllers“, das heißt sie steuern die Anwendung. Sämtliche Benutzer-Eingaben werden von Servlets verarbeitet, der Klick auf einen Speichern Button veranlasst das betreffende Servlet die veränderten Werte zu sichern und die Ansicht zu aktualisieren.
Die Registrierung des Servlets bei der Web-Anwendung über einen Eintrag im Deployment Deskriptor wird dem Benutzer der Eclipse Entwicklungsumgebung abgenommen, diese trägt beim Erstellen eines neuen Servlets alle erforderlichen Tags in die Datei web.xml ein.

4.2.3. Zugriff auf die Datenbank

Da der Zugriff auf die Datenbank an mehreren Stellen der Anwendung nötig ist, wird dieser über ein Singleton-Objekt dargestellt. Dieses wird einmalig instanziert, jedes Servlet greift bei Datenbankoperationen auf die gleiche Instanz des Zugriffobjektes zu[18]. Der Verbindungsaufbau über die JDBC-Schnittstelle findet in diesem Objekt statt.
Um das Laden eines Turnieren zu vereinfachen und die daraus resultierenden Java-Objekte direkt zu generieren, gibt es ein zweites Objekt, welches diese Aufgaben

[18] [1], Kapitel 3.7

erledigt – der direkte Zugriff auf die Datenbank erfolgt auch hierbei über das oben beschriebene Singleton-Objekt.

4.2.4. Basisklassen

Die Basisklassen sind keine speziellen Objekte für Web-Anwendungen. Diese Klassen können ohne Modifikation in jede anderen Java-Applikation eingebunden werden.
Die Klassen im Turniersystem bilden im Wesentlichen das Datenmodell ab, die Bezeichnungen und Datenfelder sind denen in der Datenbank sehr ähnlich.
Folgende Basisklassen wurden implementiert:
Match, Pinball, Player, Tournament, TournamentRound
Sämtliche 1-n Beziehungen der Datenbank wurden im Objektmodell über ArrayLists des jeweiligen Objekttyps abgebildet.
Da die persistente Speicherung in die Datenbank erfolgt, ist die Haltung der Daten in den Java-Objekten eine Art Zwischenmodell. Bei Datenmodifikation des Controllers werden immer beide Modelle aktualisiert: das sich im Speicher befindliche Objektmodell und das auf Festplatte gesicherte Abbild in der Datenbank.

4.2.5. Tabelle und Paarungsbildung

Die Berechnung der aktuellen Tabelle und die Bildung von Spielerpaarungen habe ich getrennt vom Rest der Anwendung abgebildet.
Die Funktionalität der Klasse PlayerCalc hätte auch in der eigentlichen Spielerklasse implementiert werden können, ich habe mich aber aus Gründen der Übersichtlichkeit für eine zweite Klasse entschieden.

Tabelle
Um die aktuelle Tabelle zu berechnen, wird zunächst ein Array mit den dem Turnier zugewiesenen Spielern angelegt.
Im zweiten Schritt werden in allen geschlossenen, das heißt für die Berechnung freigegebenen, Runden die einzelnen Partien den Spielern zugewiesen.
Für jede Begegnung werden den beiden betroffenen Spielern Punkte für Sieg oder Unentschieden gutgeschrieben, sowie die beiden in ihre jeweils vorhandenen

Gegnerlisten eingetragen. Das Verwalten der Gegner eines jeden Spielers ist für Berechnung der SOS(OS) Werte, sowie für die spätere grafische Darstellung der Tabelle von Bedeutung.
Sind die Punkte aller Runden gutschrieben wird im nächsten Schritt die Gegnerliste jedes Spielers durchlaufen, um die Punkte der dort gespeicherten Spieler für den SOS Wert aufzusummieren.
Sind sämtliche SOS-Werte berechnet, kann ein weiterer Durchlauf erfolgen, welcher dann anstatt der Punkte die SOS-Scores addiert um den SOSOS-Wert jedes Spielers zu errechnen.

Sind diese Werte berechnet stellt das sortierte Spieler-Array die aktuelle Tabelle dar.
Das PlayerCalc Objekt ist vom Typ Comparable, der Vergleich erfolgt nach den drei Kriterien Punkte, SOS und SOSOS, somit kann die Tabelle in einem einfachen Array gehalten werden, welches sich mit jedem Sortieralgorithmus ordnen lässt.

Paarungsbildung

Um die Gegnerpaare für eine Runde zu berechnen, werden zunächst alle theoretisch möglichen Partien gebildet; Jeder gegen jeden, unabhängig vom Tabellenplatz oder ob diese Begegnung bereits statt gefunden hat.
Für jedes dieser „Possible Matches“ wird eine Gewichtung berechnet[19] und jeder Paarung ein Variable zugewiesen.
Die Gewichtungen und Variablen werden dann zu einem zu optimierenden Gleichungssystem zusammengefügt und mit Hilfe des Simplex-Algorithmus gelöst.
Die Lösung des Gleichungssystem, in diesem Fall sämtliche Variablen welche in der Lösung gleich 1 sind, bilden die passenden Paarungen für die nächste Runde ab.
Es muss nun der entsprechenden Runde nur noch eine Liste der gefundenen Begegnungen zugewiesen werden.

[19] siehe Kapitel 5.2 Kostenfunktion

4.2.6. Aufbau JSP-Seiten

Die JSP-Seiten für die Darstellung der Anwendung in einem Web-Browser, sind alle nach dem gleichen Schema aufgebaut.
Um den Code übersichtlich zu halten und diesen möglichst auf die Inhalte der eigentlichen Seiten zu begrenzen habe ich mit Java-Custom-Tags (JCT) gearbeitet. Diese Tags sind eigene kleine JSP-Codefragmente, welche an beliebigen Stellen mehrfach eingebunden werden können.

Jede JSP-Seite inkludiert mehrere Tags:

- htmlincludes: Hier werden die Stylesheets für die Formatierung des HTML Codes eingebunden
- header: Kopfbereich der Seiten in welchem unter anderem der Turniername und die aktuelle Runde dargestellt werden
- mainmenu: Stellt das Dropdown Hauptmenü dar
- statusMessage: Darstellung von Hinweisen und Fehlermeldungen

Für den Fall, dass zum Beispiel der Kopfbereich der Seiten neu gestaltet werden soll, hat man diese Änderung nur an einer Stelle vorzunehmen um alle Ansichten zu aktualisieren.
Die Formular Seite zum Editieren eines Flippers zeigt exemplarisch die den Quellcode einer JSP-Seite, durch das Einbinden der Tags ist dieser sehr kompakt.

```
<?xml version="1.0" encoding="UTF-8" ?>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

Einbinden der Tag Library:
<%@ taglib tagdir="/WEB-INF/tags" prefix="jct" %>

Einbinden der für die JSP benötiten Java Beans:
<jsp:useBean id="pinball" scope="session" class="pinball.Pinball" />

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Flipper Formular</title>

Tag mit Stylesheet Angaben:
<jct:htmlincludes />
</head>
<body>

Anzeige des Seitenkopfes:
<jct:header />

Einbinden des Dropdown Hauptmenüs:
<jct:mainmenu />

Einbinden der Statuszeile für Anzeige von Meldungen:
<jct:statusMessage />


Hauptteil der JSP Seite (im Beispiel das Formular zum editieren von Flippern):
<h2>Flipper Formular</h2>

<form method="get" action="savePinball">
<input type="hidden" name="id" value="<jsp:getProperty name="pinball" property="id" />"
/>

<label for="name">Name</label>
<input type="text" name="name" id="name" value="<jsp:getProperty name="pinball"
property="name" />"  onkeypress="messageSaveChanges();" /> <br />

<label for="shortname">Kurzbezeichnung</label>
<input type="text" name="shortname" id="shortname" value="<jsp:getProperty
name="pinball" property="shortname" />"  onkeypress="messageSaveChanges();" /> <br />

<label for="description">Beschreibung</label>
<input type="text" name="description" id="description" value="<jsp:getProperty
name="pinball" property="description" />"  onkeypress="messageSaveChanges();" /> <br />

<input type="submit" value="Speichern" />
</form>

</body>
</html>
```

Abbildung 6: Codebeispiel JSP

4.3. Aufbau der grafischen Benutzeroberfläche

An oberster Stelle wird der Benutzer immer über das aktuell geladene Turnier und die aktuelle Runde informiert.
Unter diesem Info-Bereich befindet sich das Hauptmenü. Dieses Dropdown-Menü ist in drei Hauptpunkte unterteilt: Stammdaten, Turnier und Runde. Sämtliche Aktionen lassen sich über das Menü auswählen.
Darunter befindet sich eine Statusanzeige. In dieser Zeile werden Informationen über Laden und Speichern, sowie Programmhinweise und Fehlermeldungen angezeigt.
Der Hauptbereich der Seite zeigt den jeweiligen Dialog an. Sämtliche Formulare, Listen und Tabellen sind in diesem Bereich abgebildet.

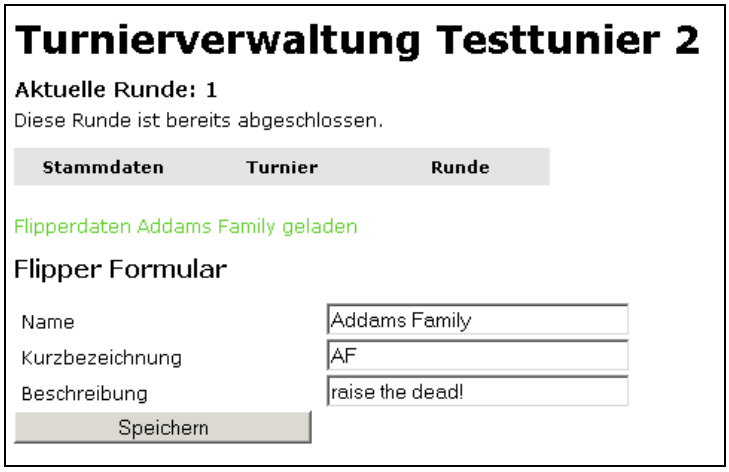


Abbildung 7: Benutzeroberfläche (Flipper editieren)

4.4. Verwaltung der Daten

Über den Hauptmenüpunkt „Stammdaten“ können Spieler, Flippergeräte und Turniere verwaltet werden. Es ist je eine Übersicht bereits angelegter Einträge und ein Formular zum neu anlegen und bearbeiten der Einträge vorhanden. Die Aktionen der Stammdatenverwaltung sind in Abbildung 8 skizziert.
In der Übersicht der Turniere können diese geladen werden.
Der Titel eines geladenen Turniers wird immer im oberen Bereich der Seite angezeigt, ebenso die aktuell gewählte Runde.

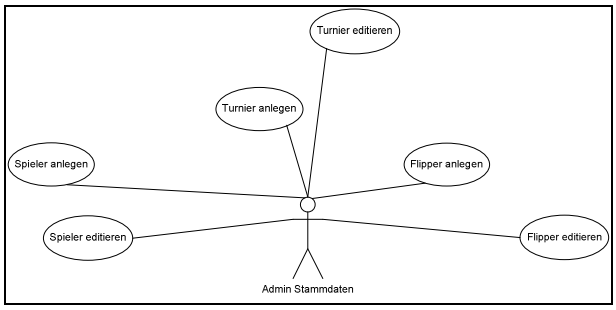


Abbildung 8: Usecase Stammdatenverwaltung

Ist ein Turnier geladen, kann man diesem über den Menüpunkt „Turnier“ Spieler und Flipper zuweisen. Nur die hier ausgewählten Einträge lassen sich in den einzelnen Runden benutzen.
Ebenso findet sich hier ein Eintrag „Tabelle“. Der aktuelle Turnierstand kann hier berechnet und angezeigt werden.

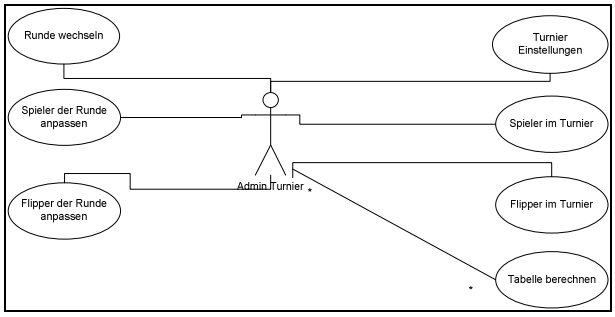


Abbildung 9: Usecase Turnierverwaltung

Die Auswahl von Spielern und Flippern kann in jeder einzelnen Runde bearbeitet werden. So ist es möglich im Laufe eines Turniers zum Beispiel defekte Geräte nicht weiter zuzuweisen. Spieler, welche das Turnier vorzeitig verlassen, können aus den späteren Runden heraus genommen werden. Diese Aktionen sind im Usecase Diagramm Turnierverwaltung in Abbildung 9 dargestellt.

Die Zuweisung zu den einzelnen Runden erfolg über das Menü „Runde“.
Im Menü „Runde“ sind weiter die Funktionen zum Wechseln von Runden, dem Abschließen der Runden und dem Generieren von Paarungen der Runde verlinkt.

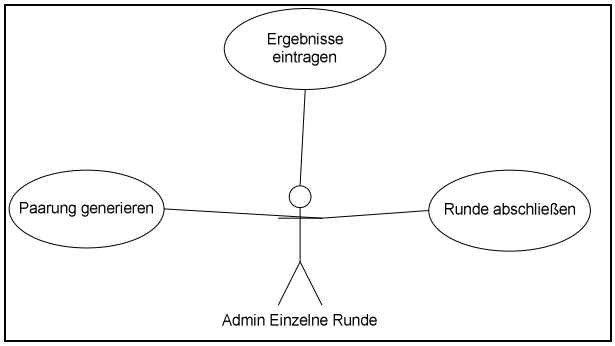


Abbildung 10: Usecase Rundenverwaltung

Die möglichen Benutzerinteraktionen innerhalb einer Runde sind im Usecase Diagramm der Abbildung 10 zusammengefasst.

4.5. Ablauf eines Turniers

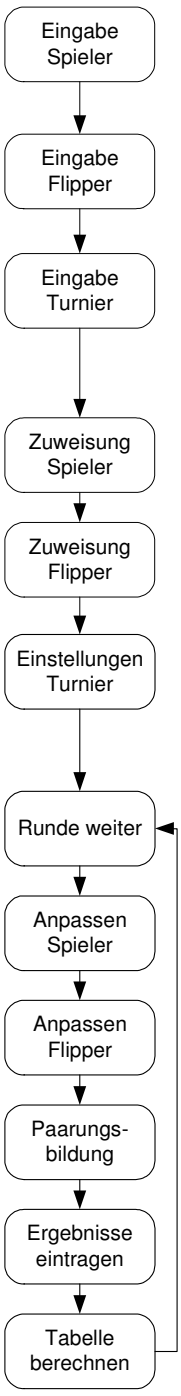


Abbildung 11: Turnierverlauf

Vor Beginn eines Turniers werden alle noch nicht erfassten, teilnehmenden Spieler und verfügbaren Flipper in den Stammdaten hinzugefügt. Ebenso wird das Turnier angelegt.

Das neu angelegte Turnier wird automatisch geladen, somit kann begonnen werden Spieler und Flipper diesem zuzuordnen. Alternativ ist es natürlich möglich jedes bereits zuvor angelegte Turnier zu laden. Die Eigenschaften, wie Name und Anzahl Runden, eines geladenen Turniers können jederzeit unter Einstellungen angepasst werden.

Zu Beginn des Turniers ist die fiktive Runde 0 ausgewählt, dieser können keine Spielpaarungen zugewiesen werden. Sind die Turniereinstellungen und Zuweisungen komplett, kann zur ersten Runde gewechselt werden.

Nun ist es möglich die Zuweisung von Spielern und Flippern zur aktuellen Runde nochmals zu überarbeiten. Die erste Runde erhält standardmäßig die gleichen Zuweisungen wie das Turnier selbst. Alle später angelegten Runden orientieren sich an der Vorrunde (somit ist gewährleistet, dass gelöschte Einträge nicht bei jeder neuen Runde wieder gewählt sind).

Eine neu angelegte Runde enthält noch keine Spielerpaarungen, diese werden über das Menü „Runde“ erzeugt. Falls sich Änderungen am Teilnehmerfeld der laufenden Runde ergeben, können angelegte Spielerpaarungen in der Rundenübersicht wieder gelöscht werden. So ist es möglich die Paarungen für das aktualisierte Teilnehmerfeld neu zu generieren.

Sind die Ergebnisse der erzeugten Begegnungen bekannt können diese ebenfalls im Menü Runde eingetragen werden. Das voreingestellte Ergebnis einer jeden Partie ist

unentschieden, per Mausklick kann der Siegpunkt einem der beiden Spieler zugewiesen werden.

Ist die Runde komplett gespielt und alle Resultate im System eingetragen kann die betreffende Runde geschlossen werden. Erst durch das Schließen der Runde wird diese in die Berechnung der Platzierungen einbezogen.

Nach dem Generieren der Tabelle wird diese angezeigt, die einzelnen Spalten wie Punkte, SOS und SOSOS sind im Kapitel 3.2 Punktevergabe und Rangfolge detailliert beschrieben. In den Runden Spalten wird die aktuelle Platzierung des Gegner dieser Runde angezeigt, das Vorzeichen sagt aus ob der Spieler diese Runde gewonnen („+“) oder verloren („-„) hat. Wird ein Spiel unentschieden gewertet, wird ein Punkt („.“) angezeigt.
Begegnungen mit dem, bei einer ungeraden Anzahl von Spielern, eingefügten „Dummy“ Spieler, sind in der Runden Spalte mit „DMY“ gekennzeichnet.

5. Umsetzung der Paarungsbildung

Die Erzeugung der einzelnen Paarungen ist das Kernstück des Turniersystems. Es wird in jeder Runde versucht möglichst gleichstarke Spieler gegeneinander antreten zu lassen. Allerdings darf im Verlauf eines Turniers eine Begegnung nicht wiederholt werden.
Als Spielstärke der einzelnen Spieler wird die aktuelle Platzierung in der Tabelle herangezogen.
Zur ersten Runde (ohne Starterpunkte) ist dies natürlich noch nicht möglich, da alle Spieler null Punkte auf ihrem Konto haben, aber schon in der zweiten Runde wird man feststellen, dass nun jeweils die Sieger bzw. Verlierer der ersten Runde gegeneinander antreten.
Unter der Berücksichtigung, dass eine Paarung nicht wiederholt wird, ist es in weiteren Runden natürlich oftmals nicht mehr möglich, exakt punktgleiche Spieler einander zuzuweisen.
Dieses Prinzip ist anhand eines Beispiels im Kapitel 3.3. erklärt.

5.1. Mathematische Lösung

Um dieses Problem mathematisch zu lösen, wird eine zu optimierende Gleichung mit mehreren Nebenbedingungen aufgestellt.
Die einzelnen Variablen entsprechen hierbei allen möglichen Spielerkombinationen. Über die Nebenbedingungen wird gewährleistet, dass jede Begegnung maximal einmal stattfinden darf.
Die einzelnen Paarungen werden über eine Kostenfunktion gewichtet. Um die optimalen Begegnungen zu erhalten, muss die Summe über alle gewichteten Unbekannten unter Berücksichtigung der Nebenbedingungen maximiert werden.

Zielfunktion:

Die von zwei Spielern *i* und *j* abhängige Funktion *g* ist die Kostenfunktion, über welche berechnet wird wie passend die entsprechende Begegnung ist. Ist die Begegnung *ij*

bereits gespielt, hat *g* einen sehr geringen Wert. Ist die Begegnung noch offen und sind die beiden Spieler punktgleich ist der Wert der Kostenfunktion extrem hoch.

$$\sum_{\forall i,j;i\neq j} g(i,j) * x_{ij} = \max$$

Nebenbedingungen:

$$\forall i : \sum_{\forall j;j\neq i} x_{ij} + \sum_{\forall j;j\neq i} x_{ji} = 1$$

Die Nebenbedingungen sagen aus, dass jeder Spieler nur an einer Begegnung teilnehmen kein. Die Anzahl der Nebenbedingungen ist gleich der Anzahl der teilnehmenden Spieler.

Ganzzahligkeit:

$x_{ij} \in N_0$

Die Ergebnisse x_{ij} geben Auskunft darüber, ob eine Begegnung gespielt wird oder nicht. Durch die Nebenbedingungen ist Folgendes sichergestellt: $0 \leq x \leq 1$.
Die Ganzzahligkeit der Ergebnisse muss allerdings explizit sichergestellt werden. Gültige Ergebnisse für x_{ij} sind 0 und 1.

Die so erzeugten Gleichungssysteme bestehen aus $\frac{n^2 - n}{2}$ Variablen und *n* Nebenbedingungen, wobei *n* die Anzahl der Teilnehmer bezeichnet.

Dieses Gleichungssystem wird mit Hilfe des Simplex-Algorithmus gelöst.
Im Fall, dass im Ergebnis nicht ganzzahlige Werte auftreten, wird dieses mit dem Branch-and-Bound-Verfahren zu einer korrekten Lösung umgerechnet.

5.1.1. Beispiel eines Gleichungssystems

Den Zusammenhang zwischen den Variablen des Gleichungssystems und den möglichen Begegnungen der einzelnen Spieler will ich an einem Beispiel mit sechs Turnierteilnehmern verdeutlichen.

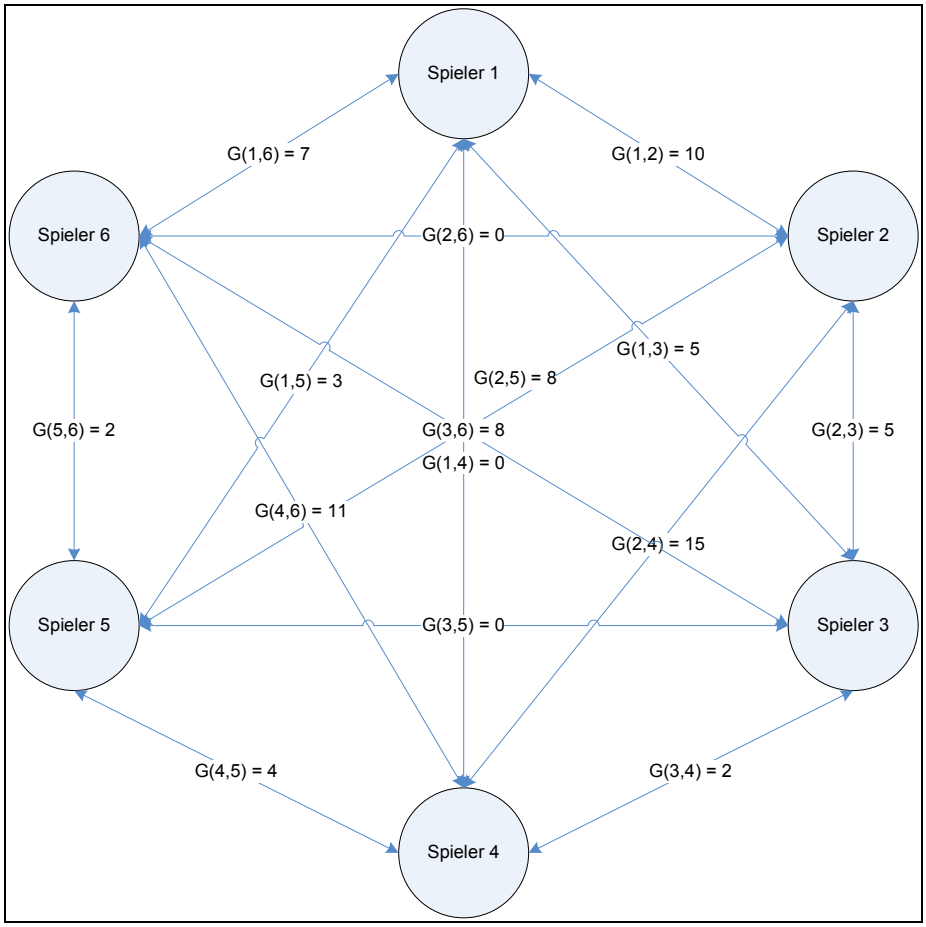


Abbildung 12: Beispiel Gleichungssystem

Die Knoten des Graphen symbolisieren die Spieler, die Kanten alle möglichen Begegnungen.
Der Graph ist vollständig, dass heißt auch die Kanten von bereits gespielte Begegnungen sind eingetragen.

Die Kosten der Begegnungen sind auf den jeweiligen Kanten vermerkt, diese Werte sind im Beispiel nicht von der Kostenfunktion berechnet, sondern frei gewählte Werte[20]. Anhand der Werte erkennt man bereits gespielte Begegnungen, da diese mit 0 bewertet sind.
Da bei sechs Spielern bereits drei Begegnungen stattgefunden haben, ist aus dem Beispielgraphen zu erkennen, dass es sich um die Paarungsberechnung für eine zweite Runde handelt.
Die bereits gespielten Begegnungen sind Spieler 1 gegen 4, Spieler 2 gegen 6 und Spieler 3 gegen 5.

Nun gilt er das zu optimierende Gleichungssystem aufzustellen.
Die Kanten des Graphen entsprechen hierbei den Variablen, in der Zielfunktion deren Gewichtung ihrem Koeffizienten.

Zielfunktion F:

$$F = 10x_{12} + 5x_{13} + 0x_{14} + 3x_{15} + 7x_{16} + 5x_{23} + 15x_{24} + 8x_{25} + 0x_{26} + 2x_{34} + 0x_{35} + 8x_{36} + 4x_{45} + 11x_{46} + 2x_{56}$$

Nebenbedingungen:

$$x_{12} + x_{13} + x_{14} + x_{15} + x_{16} = 1$$

$$x_{12} + x_{23} + x_{24} + x_{25} + x_{26} = 1$$

$$x_{13} + x_{23} + x_{34} + x_{35} + x_{36} = 1$$

$$x_{14} + x_{24} + x_{34} + x_{45} + x_{46} = 1$$

$$x_{15} + x_{25} + x_{35} + x_{45} + x_{56} = 1$$

$$x_{16} + x_{26} + x_{36} + x_{46} + x_{56} = 1$$

Dieses Gleichungssystem muss nun zu einem maximalen Zielfunktionswert optimiert werden.

---

[20] Die von der Kostenfunktion berechneten Werte sind deutlich größer (im dreistelligen Bereich) und würden das Beispiel unnötig unübersichtlich und kompliziert machen. Zur Kostenfunktion vgl. Kapitel 5.2

5.1.2. Lösung mit dem Simplex-Algorithmus

Das erzeugte Gleichungssystem wird mit Hilfe des Zwei-Phasen-Simplex-Algorithmus gelöst. Sämtliche Nebenbedingungen sind Gleichungen. Aufgrund ihrer Konstellation ist keine gültige Basislösung vorhanden, wodurch es nötig wird in der ersten Phase die fiktiven Variablen y_1-y_6 einzuführen.

Das initiale Simplex-Tableau für den ersten Schritt von Phase Eins, des in Beispiel 5.1.1 aufgestellten Gleichungssystems, ist in Tabelle 5 dargestellt. Die Nebenbedingungen lassen sich in den Spalten x_{12} bis x_{56} erkennen, sowie deren rechte Seiten in der Spalte *b* ablesen. Die Koeffizienten der Zielfunktion sind in der Zeile *F* eingetragen.
Die für die Zwei-Phasen-Methode benötigen fiktiven Variablen sind mit y_1-y_6 bezeichnet, die für die erste Phase künstlich erzeugte Zielfunktion mit *F~*.

BV	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{23}	x_{24}	x_{25}	x_{26}	x_{34}	x_{35}	x_{36}	x_{45}	x_{46}	x_{56}	y_1	y_2	y_3	y_4	y_5	y_6	**b**
y_1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	*1*	*0*	*0*	*0*	*0*	*0*	1
y_2	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	*0*	*1*	*0*	*0*	*0*	*0*	1
y_3	0	1	0	0	0	1	0	0	0	1	1	1	0	0	0	*0*	*0*	*1*	*0*	*0*	*0*	1
y_4	0	0	1	0	0	0	1	0	0	1	0	0	1	1	0	*0*	*0*	*0*	*1*	*0*	*0*	1
y_5	0	0	0	1	0	0	0	1	0	0	1	0	1	0	1	*0*	*0*	*0*	*0*	*1*	*0*	1
y_6	0	0	0	0	1	0	0	0	1	0	0	1	0	1	1	*0*	*0*	*0*	*0*	*0*	*1*	1
F	10	5	0	3	7	5	15	8	0	2	0	8	4	11	2	*0*	*0*	*0*	*0*	*0*	*0*	0
F~	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	*0*	*0*	*0*	*0*	*0*	*0*	6

Tabelle 5: Initiales Simplex-Tableau zu Beispiel 5.1.1

Die Optimierung dieses Systems erfolgt nach den Regeln der Zwei-Phasen-Methode [21].

Die Lösung des im Beispiel aufgestellten Gleichungssystems ist folgende:
$x_{15} = 1; x_{24} = 1; x_{36} = 1$ alle anderen Unbekannten x_{ij} sind 0.
Die Lösung ist ganzzahlig und damit final.

[21] [5], Kapitel 2.2.4.1

Ihre Plausibilität kann in einigen Punkten schnell und einfach überprüft werden:

1. Es sind drei Begegnungen in der Lösung (bei sechs Spielern)
2. Die Begegnungen sind in der Vorrunde nicht gespielt worden
3. Jeder Spieler nimmt nur an einer Partie teil

Diese Lösung bezeichnet nun die optimalen Begegnungen für die kommende Spielrunde, die Variablenindizes bezeichnen die jeweiligen Spieler. Somit lauten die berechneten Spielerpaarungen Spieler 1 gegen 5, Spieler 2 gegen 4 und Spieler 3 gegen 6.

Bei der Lösung mit der Zwei-Phasen-Methode ist problematisch, dass nach der ersten Phase degenerierte Basislösungen auftreten können. Das bedeutet, dass eine der fiktiven Variablen nach dem letzten Pivotschritt Nicht-Basisvariable ist.
Um dieses Problem zu beheben und die zweite Phase korrekt zu berechnen, war es nötig eine Strategie zu entwerfen, welche derartige Fälle durch weitere Umpivotisierung korrekt löst.

Von einer degenerierten Basislösung spricht man in Fällen, in denen eine Basisvariable gleich null ist. Bei solchen Problemen ändert ein weiterer Pivotschritt lediglich die Menge der Basisvariablen, die Basislösung bleibt unverändert[22].

Das folgende Beispiel zeigt die letzten beiden Simplexschritte der ersten Phase eines degenerierten Falles (Schritt n-1 und n). Die mit *y* bezeichneten Unbekannten, sind die für die erste Phase hinzugefügten fiktiven Variablen.

[22] [12]

BV	x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3	y_4	y_5	b
x_1	1	1	1	0	0	0	*1*	*0*	*0*	*0*	*0*	1
x_2	0	-1	-1	1	1	0	*-1*	*1*	*0*	*0*	*0*	0
x_6	0	1	0	1	0	1	*0*	*0*	*1*	*0*	*0*	1
y_4	0	0	2	-2	0	0	*1*	*-1*	*-1*	*1*	*0*	0
y_5	0	-1	-1	0	0	0	*-1*	*0*	*0*	*0*	*1*	0
F	0	2	11	-10	0	0	*1*	*-8*	*-4*	*0*	*0*	-11
F~	0	-1	1	-2	0	0	*-1*	*-2*	*-2*	*0*	*0*	0

Tabelle 6: Degeneriertes Problem, Simplex-Tableau, Schritt n-1

Das in Tabelle 6 umrahmte Element ist Pivotelement für den nächsten Schritt. Durch Ausführen des Pivotschritts erhält man das in Tabelle 7 dargestellt Ergebnis. Der Simplex-Algorithmus würde an diese Stelle abbrechen, da in der „F~ Zeile“[23] keine weiteren positiven Werte vorhanden sind.

BV	x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3	y_4	y_5	b
x_1	1	1	0	1	0	0	*0,5*	*0,5*	*0,5*	*-0,5*	*0*	1
x_5	0	-1	0	0	1	0	*-0,5*	*0,5*	*-0,5*	*0,5*	*0*	0
x_6	0	1	0	1	0	1	*0*	*0*	*1*	*0*	*0*	1
x_3	0	0	1	-1	0	0	*0,5*	*-0,5*	*-0,5*	*0,5*	*0*	0
y_5	0	-1	0	-1	0	0	*-0,5*	*-0,5*	*-0,5*	*0,5*	*1*	0
F	0	2	0	1	0	0	*-4,5*	*-2,5*	*1,5*	*-5,5*	*0*	-11
F~	0	-1	0	-1	0	0	*-1,5*	*-1,5*	*-1,5*	*-0,5*	*0*	0

Tabelle 7: Degeneriertes Problem 2, Simplex-Tableau, Schritt n

Die Basislösungen in den Tableaus Tabelle 6 und Tabelle 7 sind gleich:

$$x_1 = 1; x_6 = 1; x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4 = 0$$

Um die zweite Phase korrekt zu starten, dürfen keine der fiktiven Variablen Teil der Basislösung sein.

[23] In der ersten Phase der Zwei-Phasen Methode wurde an dieser Stelle die Zielfunktion mit F~ bezeichnet um die ursprüngliche Zielfunktion unter der Bezeichnung F für die zweite Phase mitzuführen

Um dies zu erreichen, muss ein weiterer Pivotschritt ausgeführt werden. Da es sich um ein degeneriertes Problem handelt, verändert sich die Basislösung dadurch nicht. Für diesen Schritt wird der in Tabelle 7 umrandete Eintrag als Pivotelement gewählt. Mit dem in Tabelle 8 dargestellten Ergebnis kann nun problemlos die zweite Phase gestartet werden, da sämtliche fiktiven Variablen Basisvariablen sind.

BV	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	$\mathbf{x_5}$	$\mathbf{x_6}$	$\boldsymbol{y_1}$	$\boldsymbol{y_2}$	$\boldsymbol{y_3}$	$\boldsymbol{y_4}$	$\boldsymbol{y_5}$	**b**
x_1	1	0	0	0	0	0	*0*	*0*	*0*	*0*	*1*	1
x_5	0	0	0	1	1	0	*0*	*1*	*0*	*0*	*-1*	0
x_6	0	0	0	0	0	1	*-0,5*	*-0,5*	*0,5*	*0,5*	*1*	1
x_3	0	0	1	-1	0	0	*0,5*	*-0,5*	*-0,5*	*0,5*	*0*	0
x_2	0	1	0	1	0	0	*0,5*	*0,5*	*0,5*	*-0,5*	*-1*	0
F	0	0	0	-1	0	0	*-5,5*	*-3,5*	*0,5*	*-4,5*	*2*	-11
F~	0	0	0	0	0	0	*-1*	*-1*	*-1*	*-1*	*-1*	0

Tabelle 8: Degeneriertes Problem 3, Simplex-Tableau, Schritt n+1

Im hier dargestellten Beispiel braucht die zweite Phase eigentlich nicht berechnet zu werden. Der Simplex-Algorithmus würde sofort terminieren und die Lösung ausgeben, da die „F Zeile“ keine positiven Werte beinhaltet.
Die optimale Basislösung ist folgende:

$x_1 = 1; x_6 = 1; x_2, x_3, x_4, x_5 = 0$

5.1.3. Ganzzahlige Lösungen

Der Simplex-Algorithmus liefert von sich aus nicht notwendig ganzzahlige Ergebnisse. Für die hier beschriebene Anwendung ist es nötig, dies zu gewährleisten.

In einer optimierten Lösung können im Ergebnis statt einer *1* (z.B. $x_{12} = 1$) beispielsweise auch zwei Unbekannte mit *0,5* (z.B. $x_{13} = 0{,}5$ und $x_{14} = 0{,}5$) sämtliche Nebenbedingungen erfüllen.

Vorgehensweise

Für einen nicht-ganzzahligen Wert werden zwei neue Gleichungssysteme aufgestellt. Diese sind bis auf jeweils eine zusätzliche Nebenbedingung identisch mit dem Ausgangssystem.

Die neue Nebenbedingung setzt eine der nicht-ganzzahligen Variablen im einen Fall auf 1, im anderen auf 0. Andere Werte können die Lösungen nicht annehmen, die Nebenbedingungen des Gleichungssystems schränken diese ein[24].

Beispiel für zusätzliche Nebenbedingungen

Findet sich im Ergebnis die Lösung $x_{13} = 0{,}5$, so werden folgende Nebenbedingungen je einem der beiden neu erstellen Gleichungssystem angefügt:

a) $x_{13} = 0$

b) $x_{13} = 1$

So wird gewährleistet, dass diese eine Variable bei Neuberechnung der beiden Systeme einen korrekten Wert annimmt.

Die neu erstellten Gleichungssysteme werden wieder mit dem Simplex-Algorithmus gelöst und im Falle von weiteren nicht-ganzzahligen Ergebnissen rekursiv um weitere derartige Nebenbedingungen erweitert.

Die Suche nach dem optimalen ganzzahligen Ergebnis wird mit dem Branch-and-Bound-Algorithmus [25] durchgeführt. Dieser beschreibt eine Strategie um möglichst schnell ein korrektes und optimales Ergebnis zu errechnen.

Die softwaregestützte Berechnung des Branch-and-Bound birgt zwei Probleme in sich: Speicherplatz und Rechenzeit.

---

[24] Normalerweise muss in einem solchen Fall der nicht-ganzzahlige Wert betrachtet werden. Abhängig von diesem werden dann die neuen Nebenbedingungen aufgestellt.

[25] [5], Kapitel 6.7

Im meinem Fall der Umsetzung des Algorithmus in Java, ist die Größe des maximal verfügbaren Java-Heap-Space[26] problematisch. Dessen Standardeinstellung reichte bei großen zu berechnenden Problemen nicht mehr aus.
Jede nicht ganzzahlige Einzellösung erzeugt zwei neue Probleme, für jeden Teilungsvorgang muss also ein Simplex-Tableau mehr im Speicher gehalten werden[27]. In Fällen mit extrem hohem Anteil von nicht ganzzahligen Einzellösungen entstehen auf diese Art in kurzer Zeit zu speichernde Zwischenlösungen im dreistelligen Bereich.
Durch eine Erhöhung des für den Java-Heap zur Verfügung stehenden Speichers konnten alle von diesem Problem verursachten Programmabbrüche vermieden werden.

Die Laufzeiten von real auftretenden Problemen betragen bei auf Ganzzahligkeit hin zu optimierenden Fällen knapp zweieinhalb Sekunden. Ein einzelnes Simplexproblem mit der Zwei-Phasen-Methode zu lösen benötigt dagegen lediglich 40 Millisekunden[28]. Der Quotient der beiden Laufzeiten entspricht in etwa der Anzahl der zu lösenden Teilprobleme im Branch-and-Bound. Der Rechenaufwand dieses Algorithmus, ist im Vergleich zum Lösen des Simplex, also so gut wie vernachlässigbar.
Die hier aufgeführten Rechenzeiten wurden bei der Paarungsbildung eines Turniers mit 36 Teilnehmern gemessen, was einer Zielfunktion mit 630 Variablen entspricht. Die Anzahl der Nebenbedingungen ist zu Beginn der Berechnung 36. Allerdings ist diese nicht konstant, sie wächst mit jeder Stufe des Branch-and-Bound.

5.2. Kostenfunktion

Die Kostenfunktion hat die Aufgabe jeder möglichen Spielerpaarung eine Gewichtung zuzuweisen. Paarungen, die vermieden werden sollen (z.B. bereits gespielte Begegnungen) haben einen geringen Wert, Paarungen die passend erscheinen (z.B. punktgleiche Spieler) erhalten einen hohen Wert.

---

[26] Dynamischer Speicher welchen Programme zur Laufzeit reservieren können

[27] Nur ein Tableau mehr, da das aufgeteilte Vorproblem für die beiden neuen Systeme aus der Liste gestrichen wird

[28] Die Laufzeitmessungen sind auf einem Windows XP Notebook mit Intel Centrino 1,7Ghz CPU und 2 Gigabyte RAM durchgeführt worden.

Während der Aufstellung der Zielfunktion des zu optimierenden Gleichungssystems wird für jede mögliche Paarung über die Kostenfunktion ein Wert berechnet.

Der Hauptfaktor für diese Funktion ist, ob die beiden Spieler bereits gegeneinander gespielt haben. Ist dies der Fall wird die Begegnung immer mit null bewertet, falls nicht, wird der Partie eine bestimmte Anzahl von Punkten gutgeschrieben.
Von ebenso von großer Bedeutung ist der Abstand in der Tabelle. Es soll versucht werden, möglichst punktgleiche Spieler gegeneinander antreten zu lassen, so dass der Beitrag zur Kostenfunktion mit dem Abstand in der Tabelle sinkt. Da die Optimierung des Systems alle Paarungen berücksichtigt, allerdings ein möglichst geringer Abstand beider Spieler einen enorm hohen Einfluss haben soll, wird ein kleiner Unterschied der Platzierung deutlicher gewichtet.
Der Anteil der Platzierung in der Kostenfunktion berechnet sich wie folgt:

$$G_{Platzierung} = (Teilnehmeranzahl - | PlatzierungSpieler1 - PlatzierungSpieler2 |)^2$$

Durch den Betrag der Differenz der Platzierungen der beiden Spieler errechnet sich der Abstand in der Tabelle. Wird dieser Wert von der Gesamtanzahl der Teilnehmer subtrahiert, erhält man als Ergebnis einen hohen Wert für eng beieinander liegende Spieler und einen geringen Wert für extrem unterschiedlich platzierte Teilnehmer. Die Quadratur dieses Ergebnisses bewirkt die bereits angesprochene Aufwertung von sehr nah oder gar gleich platzierten Spielern.

Nachdem in der ersten Runde noch keiner der beiden beschriebenen Faktoren großen Einfluss hat, habe ich drei sehr gering gewichtete Faktoren mit in die Kostenfunktion aufgenommen.

1. Der erste Buchstabe des Vornamens soll möglichst unterschiedlich zu dem des Gegners sein. Damit soll vermieden werden, dass in der ersten Runde alphabetisch gegeneinander angetreten wird.
2. Die Nationalität der beiden Spieler soll sich unterscheiden. Paarungen von Spielern unterschiedlicher Nationen sollen forciert werden, so dass in der ersten Runde ein gemischtes Teilnehmerfeld entsteht.

3. Ein kleiner Faktor Zufall soll bei Gleichheit aller anderen Faktoren den Ausschlag geben. Mit diesem Beitrag wird erreicht, dass die Startbegegnungen der ersten Runde bei einer Neuberechnung nicht zwangsläufig die Gleichen bleiben.

Die Gewichtung der hier beschriebenen Faktoren können verändert werden, so dass sich auch marginal veränderte Kriterien für die Begegnungen beschreiben lassen. Es wäre somit möglich die Gewichtung der Länderwertung höher zu setzten als die Gewichtung der Platzierung. Dies hätte zur Folge, dass im gesamten Turnierverlauf primär Spieler verschiedener Nationen aufeinander treffen und die Platzierung in der Tabelle eine untergeordnete Rolle spielte.

Ebenso ist es möglich weitere Faktoren in die Kostenfunktion einzubringen, sowie diese unter Umständen durch eine gänzlich andere auszutauschen.

6. Fazit

Mit dieser Arbeit wollte ich mehrere Dinge vereinen.
Zum ersten die Erstellung ein Web-Applikation, ein Bereich der Programmierung für den ich mich schon sehr lange interessiere, der allerdings im Rahmen des Studiums nicht behandelt wurde.
Zum zweiten das Realisieren einer solchen Anwendung in der Sprache Java. Für mich im Bezug auf Webanwendungen Neuland, als Programmiersprache begleitete mich Java wiederum viele Jahre durchs Studium.
Der dritte Punkt den ich mit einfließen lassen wollte ist die Mathematik. Durch all die Semester hinweg begleitete mich diese und ich denke, ich bin nicht der erste Informatik Student, der über weniger Mathematik im Studium froh gewesen wäre. Nichts desto trotz ist Mathematik faszinierend, deshalb fand ich es interessant, ein Thema zu haben, bei welchem ich dies sonst eher theoretische Themengebiet praktisch anwenden kann.
Der letzte Punkt ist der eigentliche Hintergrund der Anwendung: das Flipperspiel. Seit nunmehr zehn Jahren ist es eines meiner Hobbies, vor allem der Wettstreit auf Turnieren hat mich immer wieder begeistert.

Diese vier Faktoren trugen dazu bei, dass ich dieses Thema gewählt habe. Vor allem im Bezug auf die Implementierung mathematischer Algorithmen war die Arbeit sehr lehrreich. Leider kam es bei der Umsetzung zu einigen Schwierigkeiten die im Vorfeld nicht berücksichtigt wurden. Die ursprünglich geplante Lösung über einen primalen Simplex war nicht möglich und auch der Weg über die M-Methode und die 2-Phasen-Methode war sehr steinig.

Besonderen Dank möchte ich Frau Prof. Dr. Eich-Soellner aussprechen, die mir bei diesen mathematischen Problemen weitergeholfen hat.
Dadurch, dass der Paarungsalgorithmus soviel Zeit in Anspruch nahm, ist die Software nun bereit für einen Einsatz im Turnierleben, leider war am Ende kein Zeit mehr, um ein paar kleine Zusatzfunktionen einzubauen.

Ebenso zeitaufwendig, aber auch äußert interessant, war die Einarbeitung in die Erstellung von Java basierten Web-Applikationen. Meine Erfahrungen auf diesem

Gebiet waren davor auf PHP, Perl und ASP beschränkt. Die Java Technologie in diesem Bereich als Vergleich zu kennen, war für mich sehr lehrreich. Ich bin mir sicher, dass dies nicht mein letztes Java-Web Projekt war und ich in meiner beruflichen Laufbahn des Öfteren in diesem und auch im J2EE Bereich zu tun haben werde.

6.1. Aussichten

Um langfristig Turniere mit diesem System zu verwalten, fehlt es sicherlich noch ein wenig am Feinschliff der Benutzeroberfläche.
Diese mit Benutzerrechten zu versehen, so dass es zum Beispiel einen Turnier Administrator und Standard Benutzer zum Eintragen von Ergebnissen gibt, würde den Einsatz auf größeren Veranstaltungen extrem vereinfachen.
Ebenso sind auch grafisch noch bei weitem nicht alle Möglichkeiten ausgeschöpft, die Oberfläche optisch ansprechender zu gestalten sollte in der nächsten Version dieses Verwaltungssystems auf jeden Fall in Angriff genommen werden.

All dies war nicht der Hauptfokus dieser Arbeit, die einfache Verwaltung von Turnieren, die Berechnung der Tabellen und vor allem die Generierung von Gegnerpaaren standen im Vordergrund und diese Komponenten sind funktionstüchtig.
Ich denke, dass für eine erste Version eines solchen Systems der Leistungsumfang ausreichend ist. Nötige Veränderungen und fehlende Funktionen werden wohl erst die Erfahrungen des Einsatzes bei der Verwaltung einiger tatsächlicher Turniere aufzeigen.

6.2. Das Schweizer System in der Fußball-Bundesliga

Im Rahmen meiner Arbeit beschäftigte ich mich mit verschiedenen Formen von Turniersystemen. Dabei stellte sich mir die Frage, wie weit ein voller Rundenmodus durch das Schweizer System ersetzbar wäre.
Die deutsche Fußball-Bundsliga spielt ein doppeltes Rundensystem; Hin- und Rückrunde mit jeweils vertauschten Heim- und Auswärtsspielen. Mit der im Rahmen dieser Arbeit entwickelten Software sollten sich die beiden Systeme relativ schnell vergleichen lassen. Reale Spielergebnisse lassen sich problemlos recherchieren.

Wie ähnlich sind die Endtabellen, wenn man anstatt zwei Mal 17 Runden pro Halbserie nur noch zwei Turniere mit je 9 Runden im Schweizer System spielt? Um dies herauszufinden habe ich für Spielernamen die Bundesliga Vereine eingetragen und diese 18 Mannschaften einem Hin- und einem Rückrundenturnier zugewiesen.

Als Beispiel dient die Bundesliga-Saison 2007/2008. Um eine ganze Saison „realistisch“ im Schweizer System abzubilden, wird diese in zwei Turniere, eine Hin- und Rückrunde, geteilt. Im Beispiel werden den Teams vor der Hinrunde keine Startpunkte zugewiesen. Zum Start der Rückrunde bekommen sie ihre in der Hinrunde erspielten Punkte gutgeschrieben. Das Splitten in zwei Turniere ist notwendig, da gewährleistet sein muss, dass gleiche Begegnungen in der Rückrunde nochmals auftreten können.
Im Beispiel habe ich für die Ergebnisse der Hin- bzw. Rückrunde die real erspielten Resultate der jeweiligen Halbserie verwendet. Die zuerst genannte Mannschaft kann allerdings nicht als Indikator für die Heimmannschaft übernommen werden, da die Partie im echten Spielbetrieb evtl. auswärts ausgetragen wurde. Wird eine Begegnung im Hin- und Rückrundenturnier gelost, gleicht sich dies allerdings aus, da in den realen Begegnungen das Heimrecht gewechselt hat.

In den folgenden Tabellen sind die Ergebnisse der realen Fußball-Bundesliga-Hinrunde 07/08 (Tabelle 9) und dem dazu erzeugten Turnier im Schweizer System (Tabelle 10) abgebildet. Ferner werden die Fußball-Bundesliga-Endtabelle 07/08 (Tabelle 11) und das dazugehörigen Rückrundenturnier im Schweizer System (Tabelle 12) dargestellt.

Die Tabellen der Schweizer System Turniere (Tabelle 10 und 12) sind am Ende des Kapitels 4.5. sowie in Kapitel 3.3. erläutert.

Die Spalten der Bundesliga-Tabellen (Tabelle 9 und 11) sind wie folgt zu interpretieren:

- Spiele: absolvierte Spiele zum Zeitpunkt der Tabellenberechnung
- Punkte: Es werden 3 Punkte pro Sieg und 1 Punkt für jedes Unentschieden addiert
- S/U/N : Anzahl der Siege / Unentschieden/ Niederlage
- TorDiff: Differenz der Tore und Gegentore

	Team	Spiele	Punkte	S	U	N	TorDiff
1	FC Bayern München	17	36	10	6	1	23
2	Werder Bremen	17	36	11	3	3	18
3	Hamburger SV	17	32	9	5	3	11
4	Bayer Leverkusen	17	30	9	3	5	16
5	FC Schalke 04	17	29	7	8	2	9
6	Karlsruher SC	17	28	8	4	5	-2
7	Hannover 96	17	27	8	3	6	-1
8	VfB Stuttgart	17	25	8	1	8	-1
9	Eintracht Frankfurt	17	23	5	8	4	-4
10	Borussia Dortmund	17	21	6	3	8	-4
11	VfL Wolfsburg	17	20	5	5	7	0
12	Hertha BSC Berlin	17	20	6	2	9	-5
13	VfL Bochum	17	19	5	4	8	-2
14	Arminia Bielefeld	17	18	5	3	9	-19
15	Hansa Rostock	17	17	5	2	10	-10
16	1. FC Nürnberg	17	15	4	3	10	-7
17	FC Energie Cottbus	17	15	3	6	8	-10
18	MSV Duisburg	17	13	4	1	12	-12

Tabelle 9: Bundesliga Saison 07/08, Hinrunde

	Team	Punkte	Start	Siege	SOS	SOSOS	Rd1	Rd2	Rd3	Rd4	Rd5	Rd6	Rd7	Rd8	Rd9
1	FC Bayern München	21	0	6	111	1076	17.	18+	3+	4+	15+	10+	2+	9.	7.
2	Werder Bremen	17	0	5	125	1027	4-	13+	5.	16+	11+	7.	1-	3+	9+
3	Bayer Leverkusen	16	0	5	126	1056	9+	10.	1-	17+	8+	4+	11+	2-	6-
4	Hannover 96	16	0	5	122	1061	2+	15+	11+	1-	10-	3-	8+	17+	5.
5	VfL Wolfburg	15	0	4	104	1057	14+	11-	2.	9-	17+	16.	15+	10+	4.
6	Hamburger SV	15	0	4	102	999	16-	7.	18-	14+	13+	12+	10.	8.	3+
7	FC Schalke 04	14	0	3	115	1032	10-	6.	12.	18+	9+	2.	16.	11+	1.
8	Karlsruher SC	14	0	4	109	1013	13+	16+	10-	11.	3-	14+	4-	6.	12+
9	Hertha BSC Berlin	13	0	4	116	1025	3-	12+	15-	5+	7-	18+	17+	1.	2-
10	FC Energie Cottbus	12	0	3	131	1027	7+	3.	8+	15-	4+	1-	6.	5-	11.
11	Arminia Bielefeld	11	0	3	123	1050	12+	5+	4-	8.	2-	15+	3-	7-	10.
12	VfB Stuttgart	10	0	3	103	980	11-	9-	7.	13+	16+	6-	14-	18+	8-
13	1. FC Nürnberg	10	0	3	98	988	8-	2-	14+	12-	6-	17-	18.	15+	16+
14	Borussia Dortmund	10	0	3	96	969	5-	17-	13-	6-	18+	8-	12+	16.	15+
15	Hansa Rostock	9	0	3	115	1010	18+	4-	9+	10+	1-	11-	5-	13-	14-
16	Eintracht Frankfurt	9	0	2	112	969	6+	8-	17+	2-	12-	5.	7.	14.	13-
17	MSV Duisburg	7	0	2	117	994	1.	14+	16-	3-	5-	13+	9-	4-	18-
18	Vfl Bochum	7	0	2	109	973	15-	1-	6+	7-	14-	9-	13.	12-	17+

Tabelle 10: Bundesliga im Schweizer System, hypothetische Hinrunde nach 9 Runden

	Team	Spiele	Punkte	S	U	N	TorDiff
1	FC Bayern München	34	76	22	10	2	47
2	Werder Bremen	34	66	20	6	8	30
3	FC Schalke 04	34	64	18	10	6	23
4	Hamburger SV	34	54	14	12	8	21
5	VfL Wolfsburg	34	54	15	9	10	12
6	VfB Stuttgart	34	52	16	4	14	0
7	Bayer Leverkusen	34	51	15	6	13	17
8	Hannover 96	34	49	13	10	11	-2
9	Eintracht Frankfurt	34	46	12	10	12	-7
10	Hertha BSC	34	44	12	8	14	-5
11	Karlsruher SC	34	43	11	10	13	-15
12	VfL Bochum	34	41	10	11	13	-6
13	Borussia Dortmund	34	40	10	10	14	-12
14	FC Energie Cottbus	34	36	9	9	16	-21
15	Arminia Bielefeld	34	34	8	10	16	-25
16	1. FC Nürnberg	34	31	7	10	17	-16
17	Hansa Rostock	34	30	8	6	20	-22
18	MSV Duisburg	34	29	8	5	21	-19

Tabelle 11: Bundesliga Saison 07/08, Endstand

	Team	Punkte	Start	Siege	SOS	SOSOS	Rd1	Rd2	Rd3	Rd4	Rd5	Rd6	Rd7	Rd8	Rd9
1	FC Bayern München	39	21	5	248	2171	10+	2.	3.	4+	8.	5+	9+	7-	11+
2	VfL Wolfburg	34	15	5	255	2157	6+	1.	4+	8.	3+	11+	5.	10.	7+
3	Werder Bremen	34	17	5	252	2139	8+	10+	1.	5+	2-	6+	4+	9-	13.
4	FC Schalke 04	30	14	5	249	2091	14.	12+	2-	1-	11+	8+	3-	5+	9+
5	Hertha BSC Berlin	27	13	4	241	2060	13+	14+	6.	3-	10+	1-	2.	4-	18+
6	Hannover 96	27	16	3	224	2041	2-	8.	5.	10-	7+	3-	17-	15+	16+
7	FC Energie Cottbus	26	12	4	225	1976	15.	9-	16+	17.	6-	13+	8+	1+	2-
8	Hamburger SV	25	15	2	249	2031	3-	6.	12+	2.	1.	4-	7-	17+	15.
9	Eintracht Frankfurt	25	9	5	231	2027	18+	7+	11-	12.	14+	10+	1-	3+	4-
10	Bayer Leverkusen	24	16	2	246	2032	1-	3-	14.	6+	5-	9-	16+	2.	12-
11	VfB Stuttgart	23	10	4	223	1956	12-	18+	9+	14+	4-	2-	15+	13.	1-
12	Borussia Dortmund	23	10	3	194	1963	11+	4-	8-	9.	17.	15.	18.	16+	10+
13	Arminia Bielefeld	21	11	2	196	1904	5-	16.	17+	15.	18-	7-	14+	11.	3.
14	Karlsruher SC	19	14	1	198	1958	4.	5-	10.	11-	9-	16-	13-	18+	17-
15	1. FC Nürnberg	19	10	1	193	1883	7.	17.	18+	13.	16.	12.	11-	6-	8.
16	Hansa Rostock	18	9	2	189	1859	17+	13.	7-	18.	15.	14+	10-	12-	6-
17	Vfl Bochum	17	7	2	191	1860	16-	15.	13-	7.	12.	18.	6+	8-	14+
18	MSV Duisburg	13	7	1	192	1856	9-	11-	15-	16.	13+	17.	12.	14-	5-

Tabelle 12: Bundesliga im Schweizer System, hypothetische Rückrunde nach 9 Runden

Nach Hin- und Rückrunde nähern sich die Schweizer System Tabellen schon recht gut an die echten Bundesligaendstände an. Das Ergebnis überrascht, zumal Schweizer System-bedingt relativ selten eine spielschwache Mannschaft auf einen haushohen Favoriten trifft. Die so genannten „einfachen Punkte“ der Top Teams werden deutlich reduziert. Die Annäherung wäre bei 12 oder gar 15 Runden noch größer, die Anzahl der Spiele würde sich damit erhöhen, was aber in diesem Gedanken-Experiment nicht der Fall sein sollte.
Ein Schweizer System Turnier mit 17 Runden würde bewirken, dass jede Mannschaft einmal gegen jede andere angetreten ist. Die Reihenfolge der Partien wäre allerdings eine andere, als im realen Spielbetrieb. Die Tabellen würden sich allerdings sehr gleichen, lediglich bei punktgleichen Mannschaften gäbe es evtl. Unterschiede, da Tordifferenz, geschossene Tore und der direkte Vergleich im Schweizer System keinen Einfluss haben.

Im dargestellten Beispiel mit neun Runden würden insgesamt 162 Partien gespielt, eine volle Bundesliga-Saison beläuft sich auf 306 Begegnungen. Mit ca. der Hälfte der Partien, ergibt das Schweizer System bereits ein Ergebnis, welches relativ nah am „Jeder gegen Jeden“ liegt.

7. Literatur

[1] K. Brüssau (2008): Eclipse Web Tool Platform; entwickler.press

[2] V. Turau, K. Saleck, C. Lenz (2004): Web basierte Anwendungen entwickeln mit JSP2; dpunkt.verlag

[3] M. Seeboerger-Weichselbaum (2004): Java Server Pages; Markt+Technik

[4] C. Gerlach (1994): Diplomarbeit: Ein MacMahon-Losungsprogramm für Go-Turniere unter Benutzung von Maximum Weight Perfect Matching; Universität Hildesheim

[5] E. Eich-Soellner (2007): Skriptum Vorlesung Operations Research SS07; Hochschule München

[6] W. Domschke, A. Drexel (2005): Einführung in Operations Research; Springer Verlag

[7] D. R. Appleton (1995): May the Best Man Win?, The Statistican 44, No. 4, Seite 529-538; online unter `http://www.oxfordcroquet.com/tech/appleton/index.asp`

Internet Quellen:

[8] Spielsysteme (`http://www.mattzug.de/dokumen/system.htm`, 17.02.2009)

[9] G. Ortmann (1999): Kurzbeschreibung der Verfahren Gomory und Branch and Bound (`http://www.hs-augsburg.de/informatik/projekte/mebib/emiel/entw_inf/or_verf/ganzopt.html`, 19.02.2009), MeiLe-Projekt der Fachhochschule Augsburg

[10] M. Wopkes (2008) McWopkes Turnier System (`http://www.pinball-heaven.de/spieleabende.htm`, 19.02.2009)

[11] World Pinball Player Rankings (`http://www.pinballrankings.com/`, 04.03.2009)

[12] J. Leydold (1997): Degenerierte Basislösung (`http://statmath.wu-wien.ac.at/~leydold/MOK/HTML/node167.html` , 23.02.2009)

Die Internet Quellen finden sich im PDF Format auf der beiliegenden CD. Das jeweils angegebene Datum bezeichnet den Tag der PDF-Erzeugung.