UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Heinz Nixdorf Institut
Fachgruppe Softwaretechnik
Zukunftsmeile 1
33102 Paderborn

Abschlussdokumentation

im Rahmen des Softwaretechnikpraktikums 2017

Team 10

smarten
professional software development

Betreuer: Christin Löer

Paderborn, den 24. Juli 2017

Autoren:

David Bock
Niklas Doppelstein
Max Krömker
Anke Küstner
Björn Luchterhandt
Sebastian Pranger
Jost Rossel
Wolfgang Schaperdot
René Scherf
Hanna Siek
Moritz Thiele
Robin Wulfes

Inhaltsverzeichnis

1 Komponenten

Im Folgenden wird auf die entwickelten Software-Komponenten eingegangen. Dabei wird die implementierte Version der jeweiligen Komponente auch der Vergleich zum *Pflichtenheft* und zu *Analyse & Entwurf*-Dokument gezogen.

1.1 Konfigurator

1.1.1 Implementierte Version

Die graphische Benutzeroberfläche des Konfigurators ermöglicht dem Benutzer die Parameter für das Spiel einzustellen. Das Einstellen der Parameter beinhalten die Spielfeldgröße, nichtbespielbare Felder, das Auswählen der *Tiles* für die Decks der Spieler und die Möglichkeit einzelne *Tiles* zu bearbeiten. Weiterhin werden die eingegebenen Parameter direkt überprüft auf ihre Zulässigkeit für das Spiel. Wenn ein Benutzer keine eigene Konfiguration erstellen möchte, kann er die voreingestellte Standardkonfiguration verwenden.
Die implementierte Version des Konfigurators beinhaltet die Möglichkeit durch den Benutzer erstellte Konfigurationen als Konfigurationsdatei abzuspeichern. Die Spiel-Engine kann somit auch auf abgespeicherte Konfigurationsdateien zugreifen und diese nutzen um ein Spiel mit den abgespeicherten Parametern zu starten. Ebenfalls lassen sich abgespeicherte Spielkonfigurationen laden um sie erneut zu verwenden oder zu bearbeiten.

1.1.2 Vergleich zum Pflichtenheft und zu Analyse & Entwurf

Es wurden alle gewünschten Funktionen aus dem Pflichtenheft umgesetzt. Zu Beginn öffnet sich jedoch kein Dialog, welcher abfragt ob man eine neue Konfiguration erstellen oder eine abgespeicherte Konfiguration laden möchte. Beim Öffnen des Konfigurators erscheint automatisch die Standardeinstellung. Dennoch ist die Funktion zum Öffnen einer abgespeicherten Konfiguration implizit realisiert worden über die *MenuBar* und das *MenuItem* „Open File”.
Die im Analyse- & Entwurf-Dokument gestellten Anforderungen sind ebenfalls alle umgesetzt worden. Wir mussten die Klassen *OpenFileDialog* und *SaveFileDialog* jedoch nicht schreiben, da diese durch die Verwendung von *JavaFX* nur aufgerufen werden mussten.

1.2 Game Engine

1.2.1 Implementierte Version

Unsere Spiel-Engine ist der Mittelpunkt unser kompletten Produktkette. Sämtliche andere von uns implementierte Software interagiert ausschließlich mit der Spiel-Engine. Es kann eine vom Konfigurator erstellte Datei geladen werden. Der Desktop- und der Smartphone-Client sowie die KI können sich mit der Spiel-Engine verbinden und dort von dieser beim Teilnehmen am Spiel verwaltet werden. Die Spiel-Engine legt die Teilnehmer fest, bestimmt wer anfängt und richtet das Spiel aus. Neben Einzelspielen ist die Engine in der Lage, Turniere mit mehr als vier Spielern auszurichten.

1.2.2 Vergleich zum Pflichtenheft und zu Analyse & Entwurf

Es wurden bis auf einige Ausnahmen alle gewünschten Funktionen aus dem Pflichtenheft und dem Analyse und Entwurf-Dokument umgesetzt. Eine dieser Ausnahmenist, dass der Ausrichter beim Erstellen nicht auswählen kann, ob er Beobachter sein möchte. Wir haben uns dazu entschieden dieses optionale Feature nicht zu implementieren. Außerdem wird die Anzahl der Ebenen eines Turniers nicht mehr statisch bestimmt, sondern abhängig von der Anzahl der Teilnehmer und der gewählten Konfiguration dynamisch generiert. Um die Engine nicht zu blockieren, wird zudem nicht der gesamte Turnierbaum angezeigt, sondern der Gesamtgewinner, wenn das Turnier zuende ist.

1.3 Künstliche Intelligenz

1.3.1 Implementierte Version

Der implementierte *AIController* wählt aus verschiedenen Strategien die Jenige aus, welche die beste Reaktion auf die zuletzt getätigten Züge aufweist. Zusätzlich wurde ein Konsolen-Interface implementiert. Über dieses kann man die Aktionen der KI verfolgen und die Verbindung zum Server koordinieren.

1.3.2 Vergleich zum Pflichtenheft und zu Analyse & Entwurf

Alle gewünschten Funktionalitäten wurden implementiert. Zusätzlich entstand das Konsolen-Interface um die Aktivitäten der KI verfolgen zu können. Im Folgenden wird die Logik der Hauptstrategy erklärt:

Annahme: Die ersten beide Züge (erstes Placement und Token gesetzt) sind bereits von der

Fallbackstrategy durchgeführt worden.

Es gibt zwei grundlegende Prinzipien die diese Strategie verfolgt: Zum einen möchte Sie einen möglichst langen Pfad aufbauen und zum anderen will sie möglichst nicht mit anderen Spielern in Konfilkt treten. Die Strategie ist also nicht aggressiv und versucht andere Spieler aktiv aus dem Spiel zu werfen, sondern geht passiver vor und nutzt die Vorteile der Vorausberechnung und Simulierung von mehreren zukünftigen Zügen.
Zu diesem Zweck und zur Berechnung von diesen Zügen nutzen wir zwei Datenstrukturen: Einen Baum und ein Integer-Array, welches wir als HeatMap benutzen.
Jedes Blatt einer Ebene des Baums repräsentiert einen weiteren simulierten Zug. Auf der ersten Ebene, wird also in jedem Blatt ein möglicher erster Zug abgespeichert und die daraus folgende (theoretische) Spielsituation abgespeichert.
Diese Berechnung wird solange Ebene für Ebene weitergeführt, bis der nächste Spieler an der Reihe ist. Dann werden für diesen Spieler die nächsten Züge simuliert und so weiter.
Wenn die KI dann selber an der Reihe ist, werden auch hier wieder die künftigen möglichen Züge berechnet. Allerdings gibt es nach der Berechnung der möglichen Züge einen weiteren Schritt, der bei der Simulierung der anderen Player-Aktionen nicht vorhanden ist:
Während der Berechnung der Züge der anderen Spieler, inkrementiert die Strategie jeweils die Positionen in der HeatMap, über die die anderen Spieler potenziell laufen könnten. Auf diese Art und Weise bekommen wir eine Karte auf der man ablesen kann, in welchem Bereich die Wahrscheinlichkeit für einen Zug von einem gegnerischen Spieler möglichst gering ist.
Anhand dieser HeatMap können wir dann entscheiden, welcher der längsten bisher berechneten Pfade der Beste ist. Auf Basis dieser Entscheidung setzen wir dann das nächste Tsuro-Plättchen und die Simulation für die nächste Runde beginnt von vorne.

1.4 Client

1.4.1 Implementierte Version

Es wurden mehrere Clients implementiert. Es gibt einen PC-Observer, mit diesem können laufende Spiele beobachtet werden, sofern der Server eine Verbindung zulässt. Außerdem gibt es einen Smartphone-Competitor, dieser ermöglicht die Teilnahme an vom Server gehosteten Spielen. Sowohl beim Observer als auch beim Competitor muss man sich zunächst mithilfe einer IP am Server registrieren. Hat das funktioniert, so kann man aus der Liste der vorhandenen Spiele eines zum Beobachten/ Teilnehmen wählen, oder sich dem GamePool hinzufügen lassen, um auf neu erstellte Spiele zu warten. Wird das Spiel gestartet, dem man beigetreten ist, so betritt man den Beobachtungs- bzw. Teilnahmebereich. In diesem kann der Observer das Spielgeschehen verfolgen. Der Competitor kann zusätzlich, sofern er am Zug ist, Aktionen ausführen.

1.4.2 Vergleich zum Pflichtenheft und zu Analyse & Entwurf

Es wurden mehr oder weniger alle Anforderungen und Funktionen aus dem Pflichtenheft und dem Analyse- & Entwurf-Dokument umgesetzt. Es wurden teilweise kleine Änderungen in den Anzeigen vorgenommen, welche aber die Funktionen nicht grundsätzlich ändern.

2 Wesentliche Ideen des Teams

2.1 Besonderheiten im Vorgehen der Implementierung

Bei der Implementierung wurde das Team in kleinere Gruppen aufgeteilt, welche an jeweils einer Komponente gearbeitet haben. Die einzelnen Gruppen haben sich jeweils in aufgeteilt in die einzelnen Schichten des MVC Modells, so konnte gewährleistet werden, dass die Trennung möglichst strickt ist.
Neben den Komponeneten wurde von einigen des Teams zusätzlich an der KI und dem `Shared` Package gearbeitet. `Shared` soll an dieser Stelle besonders herausgehoben werden. Der Gedanke hinter dem Package ist ein essenzieller in unserem Vorgehen. In `Shared` liegen alle Klassen die von nahezu allen Komponenten genutzt werden. Nachdem die erste Implementierung stand wurde `Shared` stetig vom gesamten Team weiterentwickelt, wobei dies steht abwärtskompatibel geblieben ist. Eine Reihe von `JUnit` Tests haben dies stehts gewährleistet. Das heißt die einzelnen Komponenten wurden strickt getrennt das `Shared` Package aber strickt gemeinsam entwickelt.

Die Verwaltung unseres Projekt ist ein ebenfalls essenzieller Punkt. Die Verwaltung wurde mit Maven geregelt, dies hat eine intelligente Projetstruktur, automatisch verwaltete Dependencies, automatische Testausführungen, eine einfache JavaDoc Generierung und das einbinden externer Projekte zum Vorteil. Des weiteren können über Maven Tycho unsere Projekte sowohl als Standalone Java Application als auch als Eclipse Plugin gebaut werden.

2.2 Besonderheiten im Testvorgehen

Um eine hohe Qualität unseres Codes zu gewährleisten, muss dieser die folgenden fünf Phasen durchlaufen:

Review Jede unserer Klassen (ob Test- oder Produktivcode) wurde von einer zweiten Person gereviewed. Dadurch sind schnell logische Fehler oder konzeptionelle Probleme aufgefallen.

Story Test Jede geschriebene Klasse wurde dann von einem dritten Teammitglied explorativ getestet. Unter einem explorativen Test versteht man, dass der Tester den Code frei bedient und die Anwendung beliebig nutzt. Wichtig ist dabei, dass der Tester jeden seiner Schritte aufschreibt um gegebenenfalls anfallende Probleme reproduzieren zu können.

Unittest Parallel zum Story Test wurden, wenn sich die zu testenden Klassen dafür anboten, automatisierte JUnit Tests geschrieben. Hierbei wurde besonders darauf geachtet, zu jedem

Positiv-Testfall auch einen Negativ-Fall ab zu testen. Diese passen jedoch nicht immer 1:1 zueinander, da zum Beispiel eine Konfiguration nur auf eine Art richtig sein kann, aber auf viele verschiedene Arten falsch. Diese Tests beschäftigen sich mit einzelnen implementierten Klassen.

Komponententest Nach dem Fertigstellen der GUI wurden dann manuelle Tests geschrieben. Da wir nicht die Ressourcen aufbringen konnten (finanziell und perso- nell) automatisierte GUI Tests zu schreiben, stellen wir mithilfe von manuellen Tests die Qualität unserer Produkte sicher. Hier wird, anders als beim explorativen Testen, ein spezielles Vorgehen in Dokumenten definiert. Anhand dieser Dokumente klickt sich unser Tester dann durch das Produkt und notiert, sollten sich Unregelmäßigkeiten auftuen. Der Komponententest überprüft, ob eine einzelne Komponente alleinstehend funktioniert. Zum manuellen Testen der künstlichen Intelligenz, wurde extra eine Klasse AIWindow.java geschrieben. Diese dient als Interface zur Kommunikation mit der KI.

Systemtest Der Systemtest ist am aufwendigsten durchzuführende Test. Hier werden alle unsere Softwarekomponenten miteinander getestet. So wird z.B. anders als beim Komponententest, eine Konfiguration nicht nur im Konfigurator erstellt, sondern zusätz- lich wird sie von unserem Server geladen und die unterschiedlichen Clients spielen ein Spiel mit dieser Konfiguration. Der Systemtest umfasst also die komplette von smarten tsuro angebotene Produktkette und legt seinen Fokus daruf, ob die einzelnen Elemente miteinander harmonieren.

3 Evaluation

In diesem Teil wird die Zusammenarbeit unseres Teams aufgegriffen. Dabei wird eine Retrospektive über den Verlauf des Semesters erstellt. Des Weiteren geben wir ein Feedback zur Veranstaltung SWTPra.

3.1 Retrospektive

Unser Team *smarten* besteht aus 12 Personen. Während des ersten Treffen wurden die vom *SWTPra* vorgegebenen Rollen untereinander verteilt, dabei wurde stark auf die Fähigkeiten und Interessen der einzelnen Personen geachtet.
Im Verlauf des Semester haben wir zunehmend gelernt, wie man die in der Veranstaltung *Softwareentwurf* erlernten Vorgehen umsetzen kann und effektiv nutzen kann.
Nach jeder Abgabe wurden Reviews abgehalten sowie wöchentliche *Scrums* um darüber zu informieren, wie weit die jeweiligen Aufgaben fortgeschritten sind. Die Aufgaben wurden über die web-basierte Projektmanagementsoftware Trello im wöchentlichen Meeting verteilt, so hatte man die Möglichkeit auch außerhalb der Meetings die Fortschritte verfolgen zu können.
Ab dem *Analyse&Entwurf*-Dokument wurden Teams gebildet, um die einzelnen Software-Komponenten zu bearbeiten. Diese wurden zu Beginn der Implementierungsphase weiter beibehalten. Später teilten sich diese jedoch auf, um den Teams, die mit ihrer Komponente in Verzug gerieten, weiter zu helfen.
Die Zeitplanung gestaltete sich vorerst schwierig. Trotzdem entstanden einige Termine, an denen man sich traf um die jeweils nächste Abgabe vorzubereiten. Diese stärkten primär den Teamzusammenhalt, da der meist unterschätzte Arbeitsaufwand oftmals möglichst schnell aufgeholt werden musste.
Das Team zeigte eine hohe Leistungsbereitschaft, auch in stressigen Situationen.
Insgesamt empfanden wir die Zusammenarbeit als sehr angenehm.

3.2 Kritik und Anregungen bezüglich des SWTPras

In den Tagen vor der Endabgabe entstanden bei einer weiteren *Review*-Runde Vorschläge, wie man das Softwaretechnikpraktikum anders gestalten könnte. Das SWTPra versucht möglichst realitätsnah zu sein. Dies ist einerseits hilfreich, um zu erlernen wie man in einem realen Softwareprojekt arbeitet, andererseits gelingt dies nicht unbedingt, da durch andere Veranstaltungen der Fokus nicht bei jedem auf dem SWTPra lag. Es kam die Überlegung auf das Softwaretechnikpraktikum auf zwei Semester aufzuteilen. Somit könnte man in einem Semester die Inhalte aus der Veranstaltung *Softwareentwurf* vertiefen und im nächsten den Fokus auf die Implementierung einer Software legen. Dabei sollte das Projekt auch größer angelegt werden.

Dessen ungeachtet könnte man die Bewertungen der Abgaben transparenter machen und Mängel genauer herausstellen. Das SWTPra vermittelte uns keine zusätzlichen Lerninhalte, sondern bat uns nur Vertiefungen an.
Um genauer zu wissen wie die Dokumente aussehen müssen, wären Beispiele die inhaltlich genau das vermitteln, was sie sollen, hilfreich gewesen. Bei einigen Komponenten der jeweiligen Beispieldokumente hatte man nur Negativ-Beispiele zur Verfügung.

4 Aufwandsschätzung

Im Nachfolgenden werden die geschätzten Stunden, wie sie im Angebot zu finden sind, und die - in der PDF ”Stundenzettel” detailiert aufgelisteten - tatsächlich geleisteten Stunden des Teams verglichen.

Vergleich zwischen der Aufwandsschätzung und den tatsächlich benötigten Stunden

	Stunden geschätzt	Stunden tatsächlich
Projektmanagement	320	368,95
Anforderungsdefinition	550	167
Analyse Lastenheft	190	x
Abstraktion Lastenheft	230	x
Pflichtenheft schreiben	130	x
Analyse & Entwurf	390	325,5
Analyse	230	250,5
Entwurf	160	75
Präsentation und Dokumentation	435	301,25
Messe und Vorbereitungen	60	x
Abschlusspräsentation	40	x
Turnier	45	x
Projektdokumentation	290	x
Implementierung	1600	1101,75
Entwicklung Messeversion	760	x
Integration	100	x
Entwicklung Smartphone-Teilnehmer	640	x
Entwicklung KI	100	x
Tests	750	112,5
Spezifikation	270	x
Aufsetzen Testumgebung	90	x
Testimplementierung	230	x
Durchführung	90	x
Testdokumentation	70	x
Auslieferung & Installation	165	72
Website	80	x
Abschlussdokumentation	70	x
Endabgabe	15	x
Gesamt	**4210**	**2448,95**

Abbildung 1: Gegenüberstellung der Stunden

Wie man in Abbildung 1 sieht konnten bei den tatsächlichen Stunden nicht alle Werte bestimmt werden (an diesen Stellen wurde ein ”x” eingetragen), dies ist der Tatsache geschuldet, dass alle geleisteten Stunden unter den Punkten ”Projektmanagement”, ”Anforderungsdefinition”, ”Analyse”, ”Entwurf”, ”Präsentation+Dokumentation”, ”Implementierung”, ”Test” und ”Auslieferung+Installation” eingeordnet wurden und die genauere Aufteilung rückwirkend nicht mehr nachvollziehbar ist. Des weiteren kam es zu Uneindeutigkeiten, da zum Beispiel Testdokumentationen teilweise unter ”Präsentation+Dokumentation” eingeordnet wurden.
Wenn man die Summe der Stunden betrachtet fällt auf, dass die Abschätzung zu Beginn des Projektes sehr viel höher ausfällt, als die tatsächliche Stundenzahl. Dieses Bild sieht man auch, wenn man sich die einzelnen Punkte ansieht, da bis auf bei ”Projektmanagement” stets die geschätzten Stunden größer sind als die tatsächlichen. Allerdings kann man in den Abbildung 2 und 3 gut sehen, dass viele der Punkte Anteilig gut eingeschätzt wurden. Hier fällt vor allem auf, dass sich bei ”Test” nicht nur absolut, sondern auch anteilig stark verkalkuliert wurde.
Abschließend lässt sich sagen, dass der erwartete Arbeitsaufwand deutlich über dem tatsächlichen lag, sich also die Schätzung insgesamt nicht bewarheitet, allerdings die Anteile in den meisten Fällen gut geschätzt wurden.

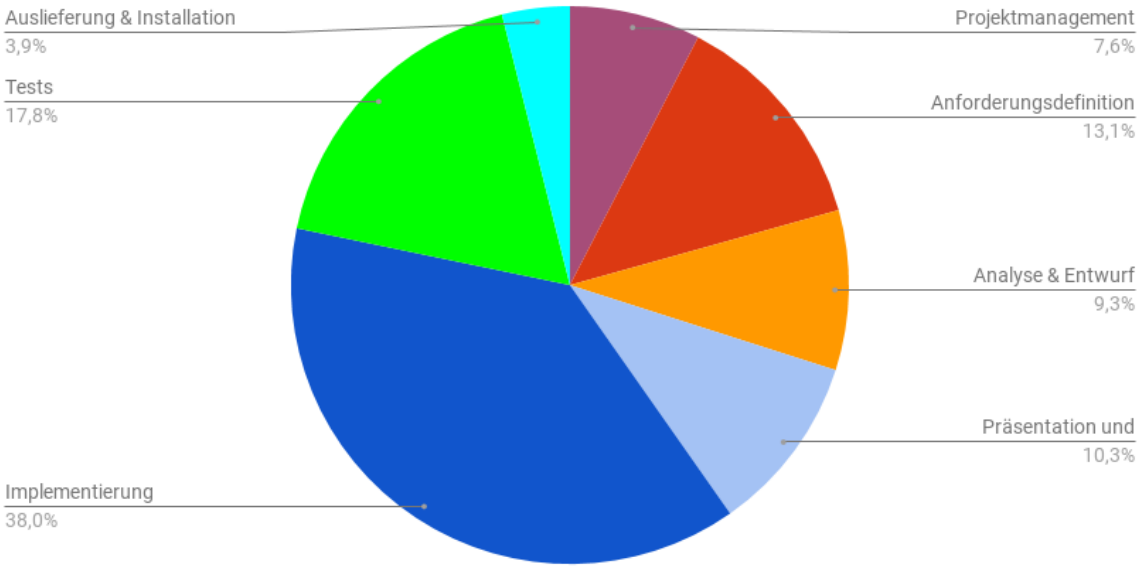

Abbildung 2: Prozentueler Anteil der geschätzten Stunden

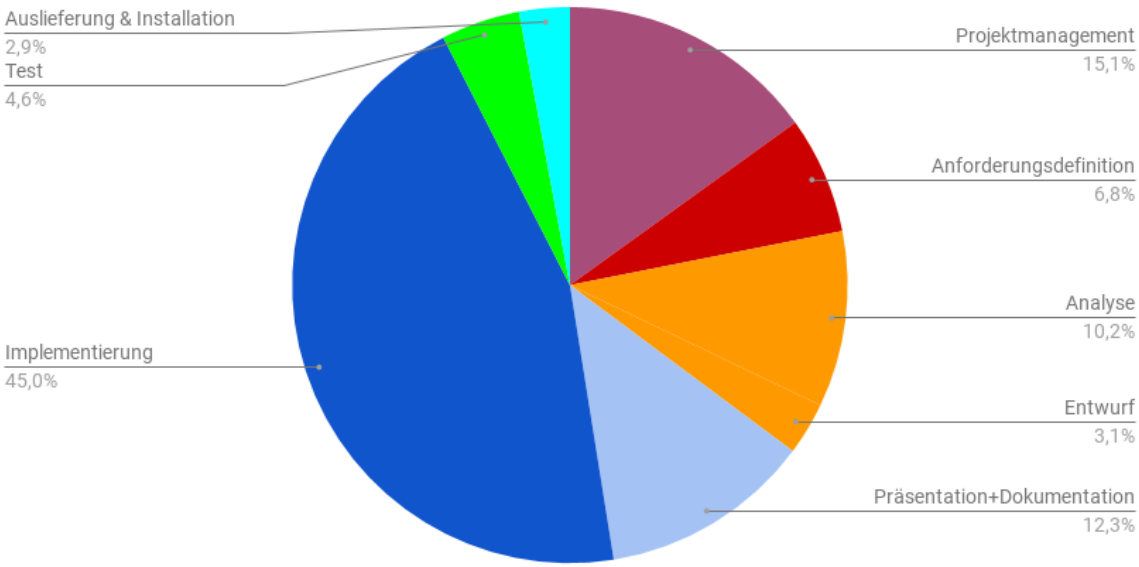

Abbildung 3:
Prozentueler Anteil der tatsächlichen Stunden

Abbildungsverzeichnis