Softwaretechnikpraktikum WiSe 2020/21

Umsetzung eines Amazonen-Spiels

Product Vision & Vorgaben

Auftraggeber

Prof. Dr. Axel-Cyrille Ngonga Ngomo, Dr. Stefan Heindorf, Adrian Wilke

DICE Research Group, Institut für Informatik,
Fakultät für Elektrotechnik, Informatik und Mathematik,
Universität Paderborn, Technologiepark 6, 33100 Paderborn

Version 1.0

Paderborn, den 15.10.2020

1 Product Vision

1.1 Einleitung

Wir, die DICE Research Group, beauftragen Sie mit der Entwicklung eines Amazonen-Spiels, welches als Software unter Linux, Mac OS, und Windows lauffähig ist. Dieses Dokument beschreibt unsere Wünsche an das zu entwickelnde Produkt und den Projektablauf.

1.2 Nutzergruppen und Funktionalität

In diesem Kapitel wird die gewünschte Funktionalität aus Benutzersicht beschrieben. Unterschieden wird zwischen KI-Spielern, Spielservern, Turnierverwaltungen und Beobachtern.

KI-Spieler (ohne GUI)

Es ist ein KI-Spieler, d. h. ein Nichtmenschlicher-Spieler, zu entwerfen, der das Amazonen-Spiel spielt. Der KI-Spieler stellt einen Webservice zur Verfügung, der vom Spielserver aufgerufen wird. KI-Spieler können auf dem PC via Kommandozeile gestartet werden (z. B. durch Aufruf einer Jar-Datei) und funktionieren ohne grafische Oberfläche. Der KI-Spieler enthält einen Algorithmus, der den nächsten Spielzug berechnet und dem Spielserver mitteilt.

Genau dann, wenn ein Spieler in einem Spiel am Zug ist, kann dieser einen gültigen Zug ausführen. Der Spielserver prüft, ob der Zug des Spielers den Spielregeln entspricht. Vor jedem Zug teilt der Spielserver dem Spieler den aktuellen Zustand des Spiels mit. Der Spielserver erkennt auch das Spielende, legt das Ergebnis fest und teilt die Beendigung des Spiels inklusive des Ergebnisses den betreffenden Spielern mit.

Spieler haben eine festgelegte Bedenkzeit zur Ausführung eines Spielzugs. Die Bedenkzeit eines Spielers beginnt abzulaufen sobald der Spieler am Zug ist. Die Bedenkzeit wird pro Spiel am Server festgelegt und ist für alle Teilnehmer gleich. Überschreitet ein Spieler seine Bedenkzeit, wird der Zug als ungültig gewertet und der Spieler hat verloren. Die Kontrolle der Bedenkzeit obliegt dem Server und wir empfehlen den KI-Spielern dringend einen Sicherheitspuffer einzuplanen, damit dies nicht passiert.

Spielserver (ohne GUI)

Ausrichter starten, konfigurieren und bedienen den *Spielserver*. Folgende Parameter können Ausrichter am Spielserver mindestens konfigurieren (unabhängig für jedes Spiel):

- Die Anzahl und Position der initialen Amazonen.
- Die Bedenkzeit pro Spielzug.

Der Spielserver muss alle Nachrichten, die ausgetauscht werden, protokollieren und validieren. Ungültige Nachrichten dürfen nicht zu einem Absturz oder einer Fehlfunktion des Spielservers führen. Spieler, die ungültige Nachrichten senden, verlieren das Spiel.

Turnier-Verwaltung (ohne GUI)

Zur Organisation von Turnieren kann der Ausrichter eine Menge von auszuführenden Spielen planen. Dazu kann der Ausrichter eine Liste von Spielern angeben (z. B. bestehend aus dem Spielernamen und den URLs der Webservices). Die Turnier-Verwaltung kommuniziert dann mit dem Spielserver: Zu Beginn werden alle vorhandenen Spiele auf dem Spielserver gelöscht. Danach überwacht die Turnierverwaltung den aktuellen Status der Spiele und erzeugt neue Spiele sobald vorherige Spiele der gleichen Spieler abgeschlossen sind. Vor dem Start des Turniers testet die Turnierverwaltung, ob alle Webservices der Spieler erreicht werden können. Während des Turniers werden alle Spielergebnisse sofort in einer Datei protokolliert (so dass sie selbst bei einem Absturz der Turnierverwaltung oder des Spielservers erhalten bleiben). Bei der Entwicklung der Turnier-Verwaltung soll ein besonderes Augenmerk auf ihre Robustheit gelegt werden, so dass es z. B. möglich ist, ein Turnier nach technischen Problemen fortzusetzen.

Beobachter (mit GUI)

Spielübersicht

Beobachter fragen beim Spielserver regelmäßig die Spiele ab (z. B. alle 5 Sekunden, um den Server nicht zu überlasten) und stellen sie grafisch dar (z. B. in Form einer Tabelle).

Spielansicht

Nach Auswahl eines konkreten Spiels, stellt ein Beobachter das Spiel grafisch dar und der Nutzer kann, z. B. durch “vor”, “zurück”-Buttons auch die Historie einsehen.

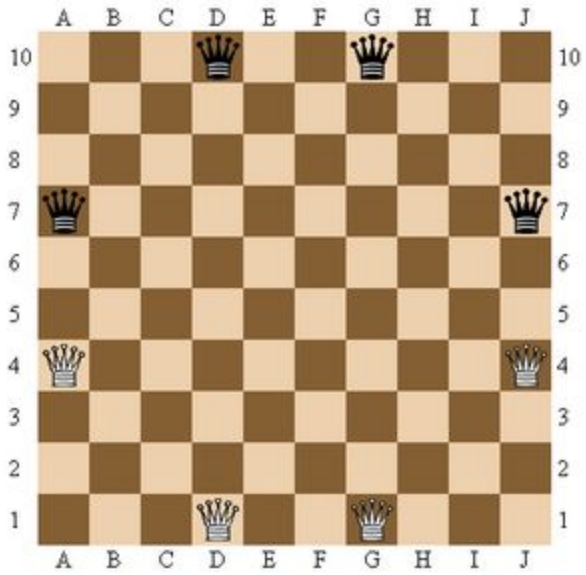
Turnieransicht

Um auf einem Spielserver ein Turnier austragen zu können zeigt der Beobachter für jeden Spieler die Anzahl der gewonnen, verlorenen und laufenden Spiele an, wobei die Liste der Spieler entweder alphabetisch oder nach der Anzahl der gewonnen Spiele sortiert werden können.

1.3 Spielregeln

Adaptiert nach: https://de.wikipedia.org/wiki/Amazonen_(Spiel)

- Das Spiel der Amazonen wird auf einem 10×10-Schachbrett gespielt.
- Die beiden Spieler spielen mit Weiß und Schwarz; jeder Spieler hat (typischerweise) vier Amazonen, die zu Beginn wie in der Abbildung gezeigt auf das Brett gesetzt werden. Außerdem werden Blockadesteine benötigt, die „giftige“ Pfeile symbolisieren.



- Die Spieler ziehen abwechselnd, Weiß beginnt.
- Jeder Zug besteht aus zwei Teilen: Zuerst wird eine eigene Amazone auf ein leeres benachbartes Feld oder über mehrere leere Felder in orthogonaler oder diagonaler Richtung gezogen, genau wie eine Dame beim Schach. Sie darf dabei kein Feld überqueren oder betreten, das bereits von einer eigenen oder gegnerischen Amazone oder einem Pfeil (Blockadestein) besetzt ist. Anschließend verschießt die gezogene Amazone vom Endpunkt ihres Zuges aus einen „giftigen“ Pfeil (Blockadestein) auf ein anderes Feld. Dieser Pfeil kann in jede orthogonale oder diagonale Richtung beliebig weit fliegen, also wiederum wie eine Dame beim Schach. Er darf auch rückwärts in Richtung des Feldes geschossen werden, von dem aus die Amazone gerade gezogen hat. Ein Pfeil darf jedoch kein Feld überqueren oder auf einem Feld landen, wo sich bereits ein anderer Pfeil oder eine Amazone befindet.
- Es besteht Zugpflicht: der Spieler am Zug muss stets eine Amazone ziehen und einen Pfeil verschießen. Verloren hat der Spieler, der als Erster nicht mehr ziehen kann.

1.4 Schnittstellen (API)

Siehe: https://github.com/dice-group/Amazons

2 Organisatorische & Technische Vorgaben

In diesem Kapitel wird beschrieben, welche organisatorischen und technischen Vorgaben wir als Auftraggeber an Sie während des Praktikums stellen.

2.1 Deliverables

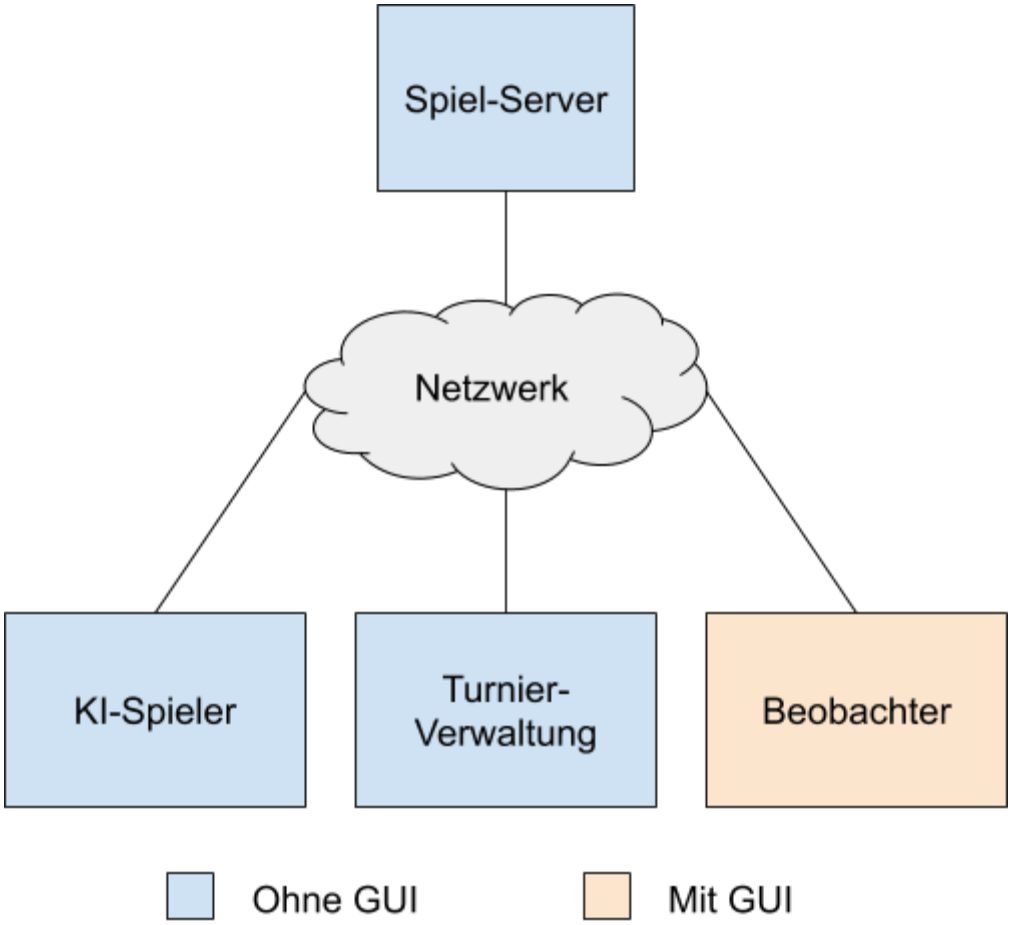
Im Rahmen des Softwaretechnikpraktikums (SWTPra) entwickeln Sie, entsprechen der Product Vision (Kapitel 1), ein netzwerkfähiges Amazonen-Spiel. Während des Praktikums wird in PANDA ein Abgabeplan veröffentlicht. Entsprechend dem veröffentlichten Abgabeplan sind zu den gegebenen Deadlines folgendes Dokumente bzw. Implementierungen abzugeben:

- Angebot
- Product Backlog
- Quality-Assurance-Dokument
- Interface-Dokument
- DevOps-Dokument
- Endabgabe: Präsentation, Implementierung, Test-Report, Website

Verspätete Abgaben werden sanktioniert. Sämtliche abzugebenden Dokumente müssen auf Deutsch (in Ausnahmefällen auch Englisch) verfasst werden. Details zum Inhalt der Abgaben finden Sie in PANDA. Der Code selbst und Codekommentare werden auf Englisch verfasst.

Sollten Sie Wirtschaftsinformatik nach alter Prüfungsordnung (Softwarepraktikum mit 5 ECTS) studieren, müssen Sie *keinen Beobachter* abgeben und keine *Turnierverwaltung*.

Eine zusammenfassende Übersicht über die Bestandteile des zu liefernden Produkts finden Sie in folgender Abbildung. Um die Kompatibilität zwischen den Komponenten verschiedener Teams zu gewährleisten, muss das vorgegebene Kommunikationsprotokoll eingehalten werden.



2.2 Ablauf des Praktikums

Nach Abgabe Ihres Angebots, Product Backlogs und des Quality Assurance Dokuments, starten Sie mit der Umsetzung des Produkts (Implementierung, Testen, Dokumentation, Sprint, Planung, etc.). In der Rolle des Kunden behalten wir es uns vor ggf. eine Überarbeitung des Angebots, Product Backlogs, oder Quality-Assurance-Dokuments vor dem Start der Umsetzung zu verlangen.

Während der Umsetzungsphase wird eine (virtuelle) Messe stattfinden, die zur Präsentation sowie zum Erwerb bzw. Veräußerung der bisher erstellten Komponenten aller Teams dient.

- Auf der Messe bieten Sie den anderen Teams einen KI-Spieler zum kostenlosen Erwerb an. Das Team dessen Spieler am häufigsten gekauft wird, erhält einen Bonus auf die *bestandene* Klausur (0,3 Notenschritte)

Zusätzlich zur Implementierung stellen Sie dem erwerbenden Team Ihr DevOps-Dokument bereit, das die Konfiguration und Inbetriebnahme der betreffenden Komponenten beschreibt. Auch nach der Messe liegt es in Ihrer Pflicht Support für die vertriebenen Komponenten zu leisten, indem Sie Fehler in Ihrer Software beheben und den Erwerbern als neue Versionen zugänglich machen.

Zeigen Sie Ihrem Betreuer, dass Sie den eingekauften KI-Spieler selber ausführen können indem Sie ihn gegen ihren eigenen Spieler auf ihrem eigenen Server spielen lassen.

Wir empfehlen allen Teams die Weiterentwicklung ihres eigenen KI-Spielers, indem Sie ihn mit dem eingekauften KI-Spieler kombinieren, um die Spielstärke zu steigern.

Nach Fertigstellung Ihres Produkts und der dazugehörigen Dokumente, präsentieren Sie Ihr Produkt in einer Abschlusspräsentation inklusive einer Live-Demonstration. Die Abschlusspräsentation schließt die Entwicklung einer Webseite ein, auf der Sie Ihr Produkt und Ihre Dokumente präsentieren.

2.3 Praktikums-Turnier

Am Ende des Praktikum findet ein Turnier statt, in dem die von den Teams entwickelten KI-Spieler gegeneinander antreten. Der *Spielserver* und die *Turnierverwaltung* wird entweder von den Auftraggebern gestellt oder eine Entwicklung der Teams verwendet. Bei der Auswahl des *Beobachters* wird analog verfahren.

Das Turnier wird nach dem Modus double round robin ausgetragen, d. h., jeder Spieler spielt gegen jeden anderen Spieler einmal mit der Farbe Weiß und einmal mit der Farbe Schwarz. Die Anzahl der Turnierrunden ist durch die Anzahl der Teams bestimmt (ca. 13 bis 15) und die maximale Zeit pro Zug wird ca. 10 Sekunden betragen.

2.4 Technologien und Entwicklungswerkzeuge

Zur Anfertigung des Product Backlogs ist die Online-Platform Kanboard[1] einzusetzen.

Die Implementierung erfolgt unter Benutzung der Programmiersprache Java. Das fertige Produkt soll unter der aktuellsten Version von Java SE 11 (LTS) lauffähig sein.

[1] https://kanboard.cs.uni-paderborn.de/

Das Kommunikationsprotokoll zwischen Spielern, Spielserver und Turnierverwaltung werden auf der Grundlage von JSON umgesetzt. Für die Konvertierung zwischen Java Objekten und JSON, empfehlen wir Ihnen, Gson[2] zu benutzen.

Zur Entwicklung und Versionsverwaltung im Team werden innerhalb des Praktikums die GitLab[3] Server der Universität genutzt. Hierzu werden Git-Repositories für die einzelnen Gruppen von uns angelegt. Um Zugriff auf das Git-Repository einer Gruppe zu erhalten, wird ein Git-Client benötigt. Git-Clients mit GUI sind zum Beispiel GitKraken[4], TortoiseGit[5] oder SourceTree[6].

Um verlässliche und reproduzierbare Builds Ihres entwickelten Produkts und der einzelnen Komponenten zu realisieren, werden Sie die Build-Systeme Maven bzw. Gradle einsetzen. Einführungen zu Maven[7] und Gradle[8] finden Sie unter den angegebenen Adressen. Zusätzlich können Sie sich bei Fragen und Problemen an die Programmierbetreuung wenden.

Für die Kommunikation im Team empfehlen wir den Zulip-Server der Uni Paderborn.[9]

Zusätzlich zu den Git-Repositories bietet die Universität Paderborn mit GitLab CI ein System zur kontinuierlichen Integration Ihrer Implementierung. Mithilfe von GitLab CI kann bspw. bei Pushes in den master-branch automatisiert der Build-Prozess und die Testausführung angestoßen werden. Somit bietet es Ihnen die Möglichkeit bei jedem Push automatisiert Ihren Code zu bauen, zu testen, und bei fehlerhaften Commits sofort Benachrichtigungen per E-Mail zu erhalten. Dies hilft Ihnen bei der kontinuierlichen Qualitätskontrolle und Koordination. Da Sie in dieser Veranstaltung entsprechend Scrum nach jedem Sprint ein inkrementelles Produkt Artefakt entwickeln, können Sie diese mit GitLab CI kontinuierlich bauen und testen. Hilfe zu GitLab CI finden Sie unter https://git.cs.upb.de/help/ci/README.md. Zusätzlich können Sie sich bei Fragen und Problemen an den Programmierberater wenden.

3. Disclaimer

Dieses Dokument sowie die API ist “Work in Progress” und wird während des SWTPras möglicherweise aktualisiert und präzisiert.

[2] https://github.com/google/gson
[3] https://git.cs.upb.de/
[4] https://www.gitkraken.com/
[5] https://tortoisegit.org/
[6] https://www.sourcetreeapp.com/
[7] https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html
[8] http://www.vogella.com/tutorials/Gradle/article.html
[9] https://chat.cs.upb.de/