Hashi

Projektentwicklung mit Scrum

# Überblick

Das Logikrätsel Hashi soll in einer Gruppenarbeit implementiert werden. Der Ablauf des Projekts erfolgt nach dem Prozessmodell Scrum.  
Beim Scrum werden zuerst die Projektrollen (Product Owner, Scrum Team, Scrum Master) definiert.  
Der Product Owner definiert die Produktanforderungen in Form von User Stories, priorisiert diese und legt sie im Product Backlog ab. In einer Sitzung, dem Sprint Planning Meeting, werden die nächsten in Angriff genommenen User Stories in den Sprint Backlog übernommen und während dem Sprint vom Team abgearbeitet. Der Product Backlog wird ständig aktuell gehalten.  
Am Ende wird ein Sprint Review durchgeführt, d.h. die inkrementierten Softwarebestandteile werden getestet und abgenommen. Dabei werden nur vollständige und fehlerfreie Arbeitsergebnisse akzeptiert. Nicht erfüllte User Stories gehen zurück in den Backlog.  
In der Sprint-Retrospektive wird über den Ablauf des Sprints nachgedacht.

## Scrum Rollen

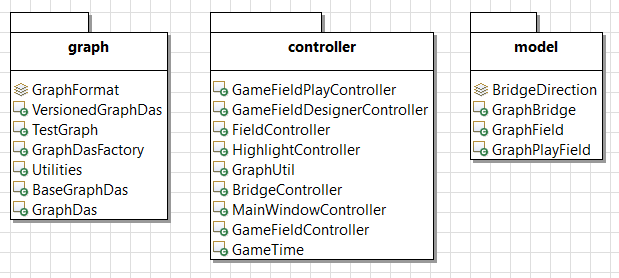
|  |  |
| --- | --- |
| Rolle | Name |
| Product Owner | Chris Müller |
| Scrum Master | Chris Müller |
| Team (Entwickler) | Joel Hüppi  Gabriel Ghenzi  Philippe Mazenauer  Martin Messmer  Christian Schmid |

# Programm

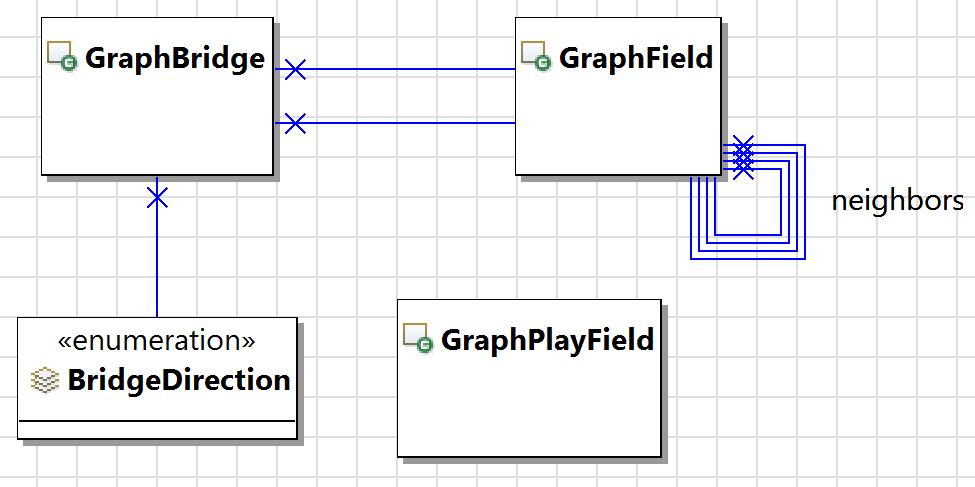
## Klassendiagramme

### Klassenstruktur

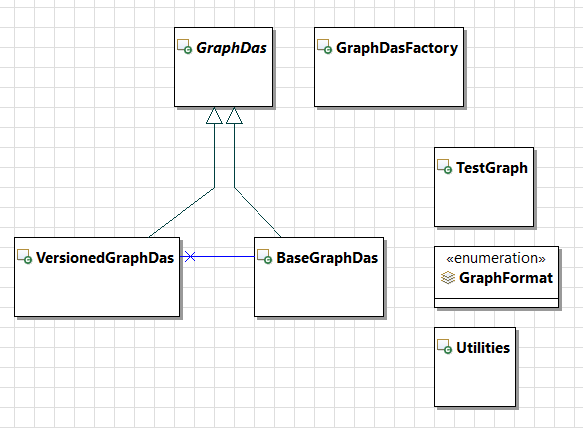
Die ganze Software wurde unterteilt in drei Packages. Im Package «graph» wird



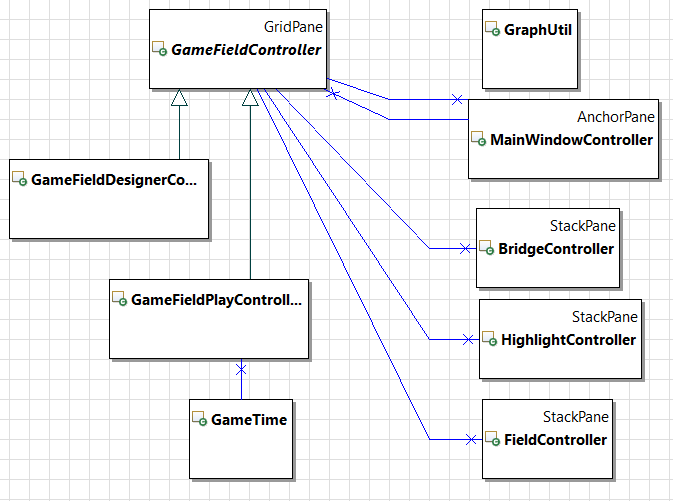
### Klassendiagramm Model



### Klassendiagramm Graph



### Klassendiagramm Controller



## Verwendete Patterns

### MVC (Model-View-Controller)

Das MVC-Pattern trennt das Game in die Komponenten Model (Verarbeitung/Datenhaltung), View (Anzeige Spielfeld) und Controller (Eingaben im Spielfeld).

### Decorator

Das Decorator-Pattern wird für die Undo/Redo Funktion verwendet. Die Klasse «VersionedGraphDas» erweitert die Klasse «BaseGraphDas» um die Undo/Redo Funktionalität.

### Command

Die Klassen «AddBridgeOperation», «RemoveBridgeOperation», «AddSolutionBridgeOperation» und «RemoveSolutionBridgeOperation» sind Kommandos und implementieren jeweils eine undo() und redo() Methode. Die ausgeführten Kommandos werden dann in der Klasse «VersionedGraphDas» in zwei separaten Stacks gespeichert. Ein Stack beinhaltet alle Undo-Kommandos, der Andere alle Redo-Kommandos. Eine Lösung mit nur einem Stack war nicht möglich, weil alle Redu-Kommandos gelöscht werden müssen sobald man nach Undo-Kommandos einen anderen Weg eingeschlagen hat.

### Factory

Für die Klasse «GraphDas» benötigten wir eine Factory «GraphDasFactory». Diese gibt beim Aufruf der Methode «getGraphDas» eine Instanz von GraphDas, sofern eine existiert, zurück. Gibt es noch keine Instanz, wird eine erstellt. Dies stellt sicher, dass nur eine GraphDas-Instanz existiert, die das Datensystem des aktuellen Spielfelds repräsentiert. Zudem kann über die Factory auch die aktuelle Instanz beendet, und eine Neue erzeugt werden. Diese Funktion wird beim Erstellen einer neuen Spielvorlage gebraucht.

### Template

## Spiel speichern/laden

Funktion gegeben durch GraphDas-Bibliothek

# Erfahrungen

Aller Anfang ist schwer. Das haben wir auch bei der Abwicklung des Hashi-Projektes mit dem Scrum-Prozessmodell erfahren. Was sich in der Theorie simple anhörte, war bei der Umsetzung zum Teil schwierig. Trotzdem konnten wir einige Eindrücke und Erfahrungen sammeln, welche uns im späteren Berufsleben vielleicht wieder weiterhelfen können.

Obwohl man bei der Realisierung eines Projektes mit Scrum relativ frei ist, war eine grobe Grundplanung unverzichtbar. Nachdem die User Stories definiert wurden, sassen wir zusammen um die Grundstruktur der Software zu planen. Wir erstellten ein Domänenmodell und leiteten daraus ein Klassendiagramm ab. Dadurch war der Aufbau der Software von Anfang an klar. Gewisse Unklarheiten und Meinungsverschiedenheiten wurden somit aus dem Weg geräumt. Nach mehreren Diskussionen, wie wir die ganze Spiellogik programmieren möchten, ging es flott zur Sache. Das Entwicklerteam verstand sich super und arbeitete effizient zusammen. Jeder konnte seine Stärken und Fähigkeiten gezielt einsetzen, was auch den Lerneffekt unter den Teammitgliedern förderte. Hier sehen wir einen klaren Vorteil von Scrum.  
Etwas unklar war die Erstellung des Product und Sprint Backlogs. Wir wussten nicht genau, wie exakt die verschiedenen Items ausgearbeitet werden müssen, somit waren sie anfangs recht schwammig definiert. Dadurch konnten die ersten Sprints nicht ganz fertig abgearbeitet werden. Diese Items haben wir dann feiner unterteilt. Der Product Backlog hat sich dadurch stetig vergrössert.  
Wenn man Scrum konsequent anwenden würde, hätte man jeden Tag das sogenannte «Daily Scrum Meeting», in dem das Entwicklerteam seine Aktivitäten synchronisiert und an der Planung für die nächsten 24 Stunden arbeitet. Da wir nicht täglich an unserem Projekt arbeiteten und unser Team aus Vollzeit und Teilzeit Studenten bestand, führten wir nicht täglich ein Meeting durch.

Rückblickend war die Projektabwicklung mit Scrum sehr interessant. Wir denken, dass die Projektentwicklung mit Scrum noch verbessert werden kann, wenn man intensiv geschult wird und das erlernte auch tagtäglich im Berufsleben anwenden kann.

# Vergleich der Prozesse

Scrum bietet viele Vorteile wie einfache Regeln, wenige Rollen, Selbstorganisation und Eigenverantwortung in Teams und ist speziell geeignet für hochkomplexe Projekte mit unklaren Anforderungen. Die Projekte werden in mehreren Inkrementen entwickelt. Das macht Scrum sehr flexibel.  
Im Vergleich zu anderen Prozessmodellen, wird bei Scrum auf einen Grossteil der Planung verzichtet. Es wurden keine Use-Cases, Systemsequenzdiagramme, Sequenzdiagramme, usw. erstellt. Das minimiert den Planungsaufwand enorm. Das könnte aber in grösseren Projekten auch zum Verhängnis werden.   
Da unser Projekt relativ klein war, sahen wir es als ausreichend an, nur ein Klassendiagramm zu erstellen. Viele Funktionalitäten und Ideen zur Umsetzung der Software, folgten aus Diskussionen unter Teammitgliedern.  
Abschliessend können wir sagen, dass die Projektabwicklung mit Scrum, bis auf einige Kleinigkeiten, erfolgreich war.

# Entwicklungsschritte

## Product Backlog

Anforderungen und Funktionalitäten welche in einem Produkt enthalten sein können, werden in einer geordneten Liste, dem Product Backlog abgelegt. Die Einträge werden Anfangs aus den User Stories abgeleitet und hinsichtlich ihrer Priorität bewertet. Der Product Backlog ist niemals vollständig. Er entwickelt sich ständig weiter.

Screenshot ProductBacklog einfügen

## Sprint Backlog

Vom Entwicklungsteam werden Einträge aus dem Product Backlog ausgewählt, welche im nächsten Sprint umgesetzt werden sollen. Diese werden im Sprint Backlog abgelegt.  
Die Sprintdauer haben wir am Projektanfang auf eine Woche festgelegt, da wir zur Umsetzung des Projektes nur ca. 2 Monate zur Verfügung hatten. Dabei kam es aber einige Male vor, dass gewisse Einträge/Funktionalitäten im Sprint Backlog nach einer Woche nicht fertig implementiert werden konnten, weil sie einfach zu aufwendig waren. Diese Einträge wurden in den nächsten Sprint übertragen. Die kurze Sprintdauer hatte allerdings auch einen positiven Aspekt, wir hatten jederzeit einen guten Überblick, wie der Entwicklungsstand des Projektes war.

Screenshots der Sprints einfügen

# Literaturverzeichnis

<http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-DE.pdf#zoom=100>

<http://scrum-master.de/Scrum-Einfuehrung>