

# Hashi

Projektentwicklung mit Scrum – Team SG

# Ablauf

- Prozessmodell Scrum
- Entwicklungsschritte
  - User Stories
  - Product und Sprint Backlog
- Software
  - Klassendiagramme
  - Verwendete Patterns
  - Spiel speichern/laden
- Metriken und Tests
- Erfahrungen
- Vergleich der Prozesse
- Vorführung Hashi

# Scrum

- User Stories

StoryID	User Story	Priority (1-5)	Story Points (1-10)
1	<u>GUI</u>  Alle Funktionen des Spiels sind mit der Maus ausführbar. Graphische Darstellung des Spiels.	5	10
2	<u>Spiel laden</u>  Neues Spiel aus Datei (XML) laden. Datei beinhaltet das Spiel sowie die Lösung des Spiels.	5	7

- Product Backlog

StoryID	ItemID	Backlog Item	Category	Priority (1-5) 1: low, 5: high	Sprint	Status	Datum erledigt	Verantwortlicher
12	1	Scrum-Rollen verteilen	Planung	5	1	erledigt	19.04.2016	Team
12	2	Userstories definieren	Planung	5	1	erledigt	19.04.2016	Team
12	3	GitHub einrichten	Programmierung	4	1	erledigt	19.04.2016	Mazenauer
12	4	Domänenmodell erstellen	Planung	4	2	erledigt	26.04.2016	Team
12	5	Aus Domänenmodell grobes Klassendiagramm erstell	Planung	4	2	erledigt	26.04.2016	Team

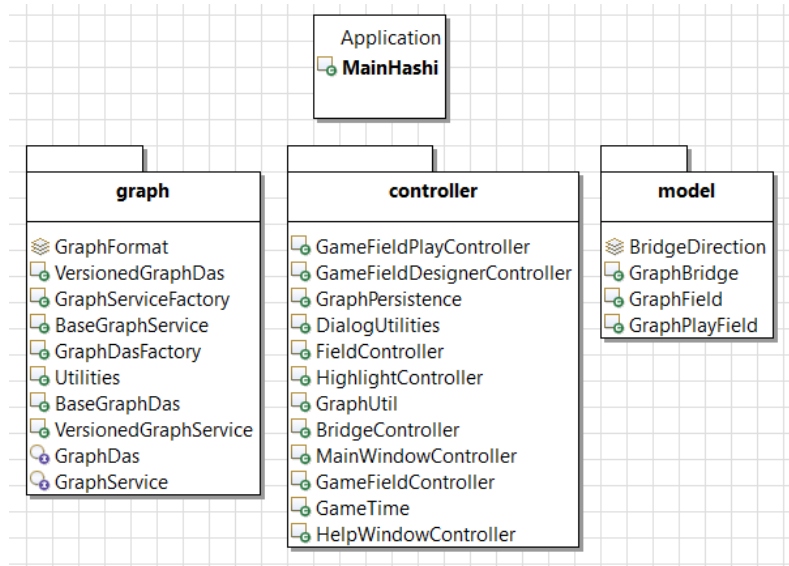
# Scrum

- Sprint Backlog

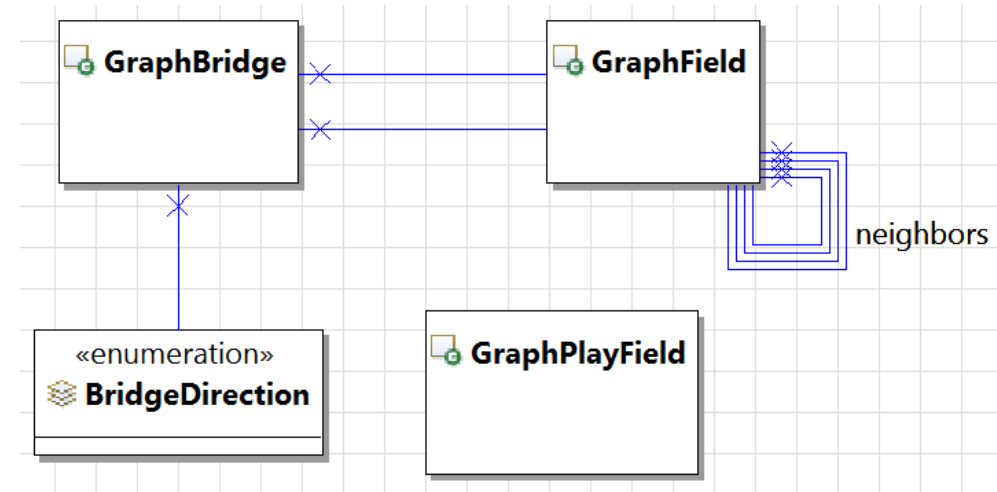
Datum: 19. - 25. April 2016				
StoryID	ItemID	Backlog Item	Category	Verantwortlicher
11	1	Scrum-Rollen verteilen	Planung	Team
11	2	Userstories definieren	Planung	Team
11	3	GitHub einrichten.	Programmierung	Mazenauer

# Klassendiagramm

- Software unterteilt in 3 Packages

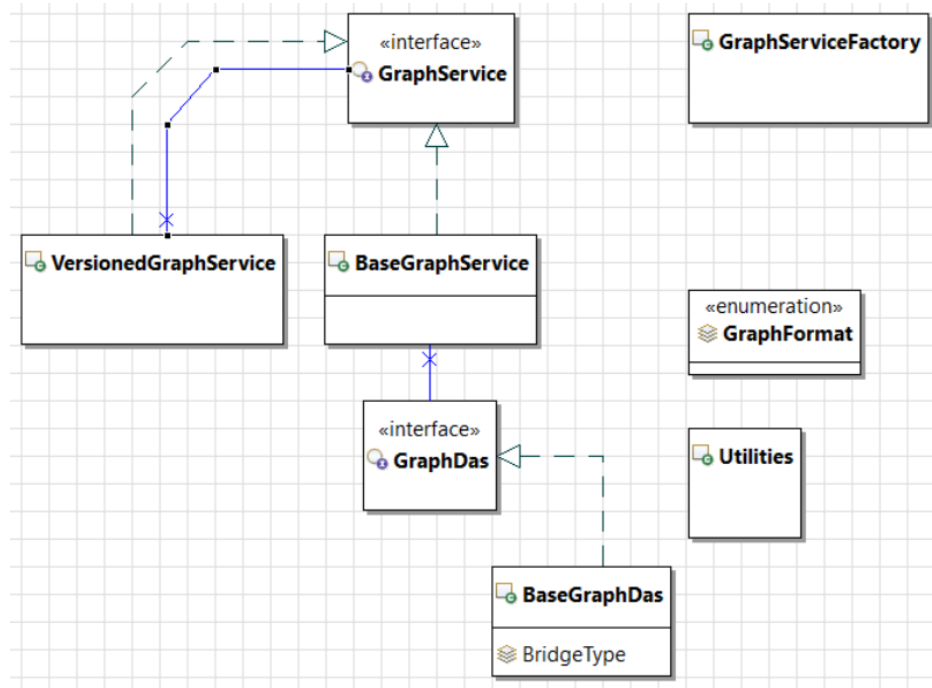


- Model

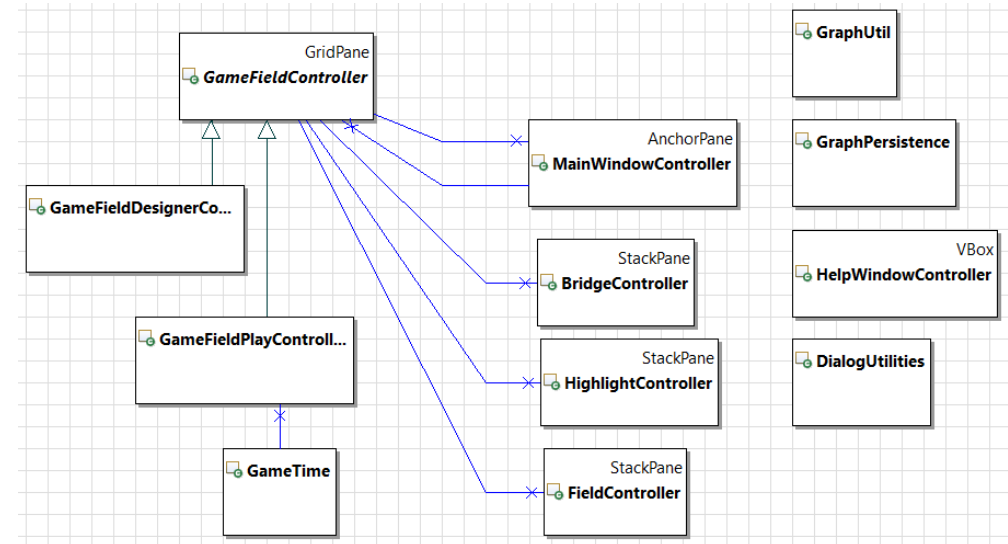


# Klassendiagramm

- Graph



- Controller



# Verwendete Patterns

- MVC (Model-View-Controller)
  - Model (Verarbeitung/Datenhaltung)
  - View (GUI)
  - Controller (Eingaben im Spielfeld)
- Decorator
  - Undo/Redo
- Command
  - Undo/Redo
- Factory
  - Erstellen neuer Spielvorlage
- Template
  - Abstrakte Klasse «GameFieldController»

# Spiel speichern/laden


- Basierend auf Tinkerpop/neo4j (Graphedatenbank)
- Funktionen speichern und laden integriert
- Gespeichert als GraphML (XML) oder JSON



# Metriken und Tests

- Tests
  - Junit
  - Code-Coverage
  - Keine gravierende Fehler, jedoch Code besser strukturieren
- Metrik
  - Software-Tool «Borland Together»
  - Schlechte CBO (Coupling Between Objects) und DOIH (Depth of Inheritance Hierarchy) Werte

## Metrics Test on 4th July 2016

Resource	<a href="#">Attribute Inheritance Factor (AIF)</a>	<a href="#">Coupling Between Objects (CBO)</a>	<a href="#">Cyclomatic Complexity (CC)</a>	<a href="#">Comment Ratio (CR)</a>	<a href="#">Depth Of Inheritance Hierarchy (DOIH)</a>	<a href="#">Lack Of Cohesion Of Methods 3 (LCOM3)</a>	<a href="#">Lines Of Code (LOC)</a>	<a href="#">Method Inheritance Factor (MIF)</a>	<a href="#">Number Of Child Classes (NOCC)</a>
 hashi	33	33	12	0	7	100	1824	95	2

# Erfahrungen

- Scrum
  - Grundplanung
  - Entwicklung parallel → gute Kommunikation zwingend notwendig
  - Gezieltes Einsetzen der Stärken und Fähigkeiten der Teammitglieder
  - Schwierigkeiten beim Erstellen des Product & Sprint Backlogs
  - «Daily Scrum Meeting» nicht «daily» durchgeführt
  - Projektabwicklung interessant

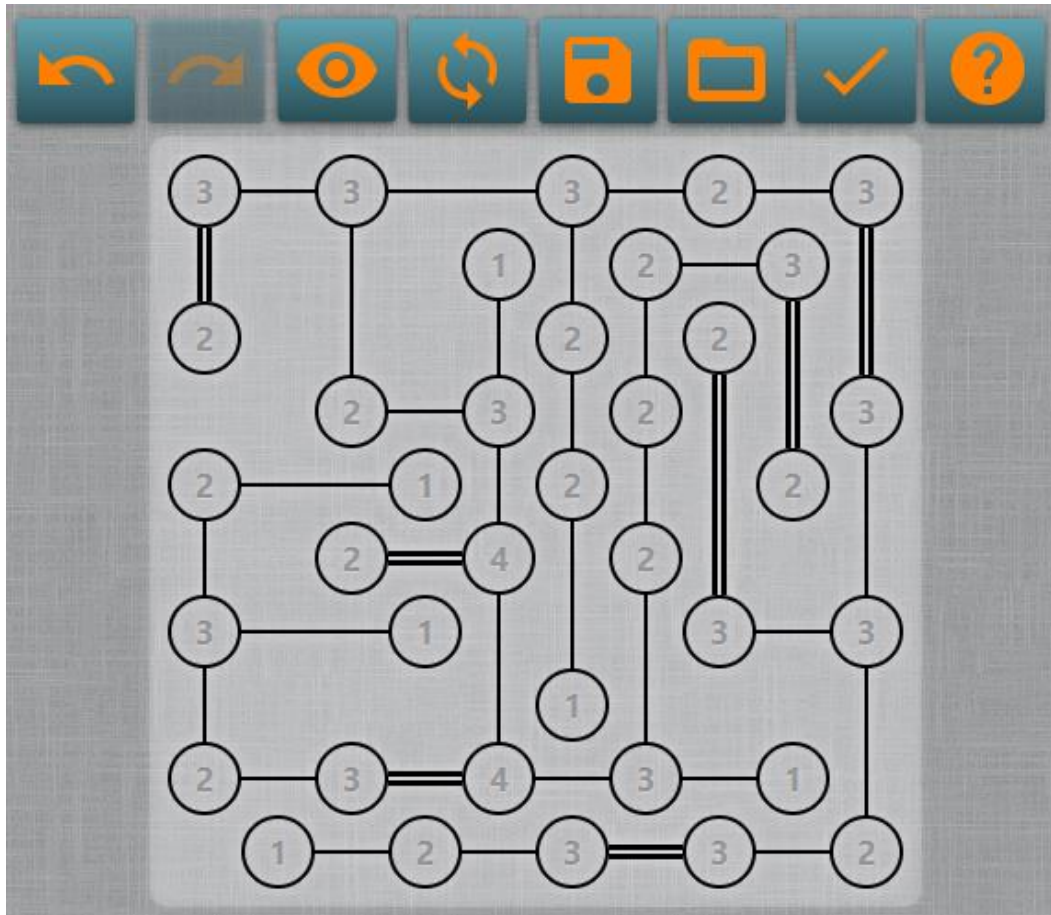
# Erfahrungen

- Verwendete Tools
  - [Git](#)
  - Tinkerpop
  - FXML
  - Junit, EclEmma, Borland Together

# Vergleich der Prozesse

- Wenig Planung erforderlich
- Flexibilität
- Entwickler viel Freiheit aber auch Eigenverantwortung

# Vorführung Hashi



Fragen?