Softwarepraktikum

Vincent Langenfeld, Daniel Dietsch

24. Oktober 2019

Agenda

- Organisation
- Thema und Anforderungen
- Ablauf
- Scrum als Vorgehensmodell
- Scrum im Softwarepraktikum

Lernziele

Entwickeln Sie eine komplexe Software in einem Team.

- Selbständiges Einarbeiten in ein Gebiet
- Arbeiten im Team
- Umgang mit Komplexität
- Anwendung softwaretechnischer Prinzipien

Lernziele

Entwickeln Sie eine komplexe Software in einem Team.

- Selbständiges Einarbeiten in ein Gebiet
- Arbeiten im Team
- Umgang mit Komplexität
- Anwendung softwaretechnischer Prinzipien

```
Wir setzen voraus,
dass Sie bereits
programmieren
können!
```

Organisation

Team

- **Tutoren**
  Jonas Werner, Alexander Petrov, Antonia Hinkerohe, Michael Fleig, Samuel Becker, Yannick Vogt, Victor Maier
- **Dozenten**
  Vincent Langenfeld, Daniel Dietsch
- **Verantwortlich**
  Prof. Dr. A. Podelski

Organisation

- Gruppen mit 5-8 Studenten
- 6 ECTS (180h) über 17 Wochen
  - 14 Arbeitsabschnitte (Sprints)
- 4 Vorlesungen (Do. 14-18 Uhr)
- Termine
  - 6 Abgaben (Samstags bis 23:59)
  - 3 Präsentationen (Do. 14-18 Uhr)
  - Wöchentliches Gruppentreffen

Dienste und Werkzeuge

- Wiki
- Fragen und Informationen
  Wiki, Gruppenmitglieder, Tutoren, Poolbetreuung
- Primärdienste
  Gitea, Git, Jenkins, Mailinglisten (`sopra-crew@...`, `sopra-XX@...`)
- Sekundärdienste
  Poolrechner, Sonar, (Doxygen)
- Werkzeuge
  C#, F#, .NET, Monogame 3.7, Visual Studio Community 19, Resharper

Mitarbeit

- Kontinuierliche Mitarbeit während der Sprints:
  - Commits im Git-Repository und
  - Aktivität (Tickets, Kommentare, etc.) in Gitea.
  - Es darf max. 2x nicht mitgearbeitet werden
- Im Durchschnitt Aufgaben mit min. 7h geschätzter Arbeitszeit pro Sprint erledigt.

Mitarbeit

- Kontinuierliche Mitarbeit während der Sprints:
  - Commits im Git-Repository und
  - Aktivität (Tickets, Kommentare, etc.) in Gitea.
  - Es darf max. 2x nicht mitgearbeitet werden
- Im Durchschnitt Aufgaben mit min. 7h geschätzter Arbeitszeit pro Sprint erledigt.

`ECTS:`

$$\frac{6 * 30h}{180h}$$

`Termine:`

$$\begin{aligned} 4 * 2h &= 8h \\ 3 * 4h &= 12h \\ 14 * 2h &= 26h \\ \hline & 48h \end{aligned}$$

`Arbeit:`

$$\frac{(180h - 48)/14}{\textbf{9,43h}}$$

Mitarbeit

- Kontinuierliche Mitarbeit während der Sprints:
  - Commits im Git-Repository und
  - Aktivität (Tickets, Kommentare, etc.) in Gitea.
  - Es darf max. 2x nicht mitgearbeitet werden
- Im Durchschnitt Aufgaben mit min. 7h geschätzter Arbeitszeit pro Sprint erledigt.

Gruppentreffen

- Anwesenheitspflicht
  - Es darf max. 1x gefehlt werden

`ECTS:`

$$\frac{6 * 30h}{180h}$$

`Termine:`

$$\begin{aligned} 4 * 2h &= 8h \\ 3 * 4h &= 12h \\ 14 * 2h &= 26h \\ \hline & 48h \end{aligned}$$

`Arbeit:`

$$\frac{(180h - 48)/14}{\textbf{9,43h}}$$

Mitarbeit

- Kontinuierliche Mitarbeit während der Sprints:
  - Commits im Git-Repository und
  - Aktivität (Tickets, Kommentare, etc.) in Gitea.
  - Es darf max. 2x nicht mitgearbeitet werden
- Im Durchschnitt Aufgaben mit min. 7h geschätzter Arbeitszeit pro Sprint erledigt.

Gruppentreffen

- Anwesenheitspflicht
  - Es darf max. 1x gefehlt werden

Präsentationen

- Anwesenheitspflicht

`ECTS:`

$$\frac{6 * 30h}{180h}$$

`Termine:`

$$\begin{aligned} 4 * 2h &= 8h \\ 3 * 4h &= 12h \\ 14 * 2h &= 26h \\ \hline & 48h \end{aligned}$$

`Arbeit:`

$$\frac{(180h - 48)/14}{\textbf{9,43h}}$$

Note

Berechnet sich aus:

- 50% Endprodukt
- 50% Einzelnote

Endprodukt bewertet nach

- F eatures
- A rtefakte
- U sability
- S pass
- T echdemo

Einzelnote

- Pro Sprint bekommt jeder Studierende 5 Punkte
  - Zugeteilte Arbeit erledigt?
  - Note ergibt sich aus Punkten

Thema

Entwickeln Sie ein

Computerspiel

Initiale Anforderungen

Anforderungen[1]

Anforderungen legen fest, was man von einem Softwaresystem als Eigenschaften erwartet.

[1] Helmut Balzert, Lehrbuch der Softwaretechnik - Basiskonzepte und Requirementsengineering, 2009, ISBN 978-3-8274-1705-3

Anforderungen[1]

Anforderungen legen fest, was man von einem Softwaresystem als Eigenschaften erwartet.

Gewöhnlich werden 3 Arten von Anforderungen unterschieden:

[1] Helmut Balzert, Lehrbuch der Softwaretechnik - Basiskonzepte und Requirementsengineering, 2009, ISBN 978-3-8274-1705-3

Anforderungen[1]

Anforderungen legen fest, was man von einem Softwaresystem als Eigenschaften erwartet.

Gewöhnlich werden 3 Arten von Anforderungen unterschieden:

- Funktionale Anforderungen legen eine vom Softwaresystem oder einer seiner Komponenten bereitzustellende Funktionen [...] fest.

[1] Helmut Balzert, Lehrbuch der Softwaretechnik - Basiskonzepte und Requirementsengineering, 2009, ISBN 978-3-8274-1705-3

Anforderungen[1]

Anforderungen legen fest, was man von einem Softwaresystem als Eigenschaften erwartet.

Gewöhnlich werden 3 Arten von Anforderungen unterschieden:

- Funktionale Anforderungen legen eine vom Softwaresystem oder einer seiner Komponenten bereitzustellende Funktionen [...] fest.
- Qualitätsanforderungen beschreiben zusätzliche Eigenschaften, die diese Funktionen haben sollen.

[1] Helmut Balzert, Lehrbuch der Softwaretechnik - Basiskonzepte und Requirementsengineering, 2009, ISBN 978-3-8274-1705-3

Anforderungen[1]

Anforderungen legen fest, was man von einem Softwaresystem als Eigenschaften erwartet.

Gewöhnlich werden 3 Arten von Anforderungen unterschieden:

- Funktionale Anforderungen legen eine vom Softwaresystem oder einer seiner Komponenten bereitzustellende Funktionen [...] fest.
- Qualitätsanforderungen beschreiben zusätzliche Eigenschaften, die diese Funktionen haben sollen.
- Randbedingungen legen organisatorische und/oder technische Restriktionen für das Softwaresystem und/oder den Entwicklungsprozess fest.

[1] Helmut Balzert, Lehrbuch der Softwaretechnik - Basiskonzepte und Requirementsengineering, 2009, ISBN 978-3-8274-1705-3

- 2D oder 3D Grafik (kein ASCII)
- Soundeffekte und Musik
- Mindestens 2 Spieler, min. einer menschlich
- Echtzeit
- Indirekte Steuerung (Point & Click)
- Pausefunktion
- Eigenes Menü (komplett mit der Maus steuerbar, außer Tastatureingaben)

Initiale Anforderungen!

- 2D oder 3D Grafik (kein ASCII)
- Soundeffekte und Musik
- Mindestens 2 Spieler, min. einer menschlich
- Echtzeit
- Indirekte Steuerung (Point & Click)
- Pausefunktion
- Eigenes Menü (komplett mit der Maus steuerbar, außer Tastatureingaben)
- Spielobjekte
  a. Min. 5 Kontrollierbare
  b. Min. 5 Auswählbare
  c. Min. 5 Nichtkontrollierbare, davon min. 3 Kollidierende
  d. Min. 3 Kontrollierbare, Kollidierende und Bewegliche.

Initiale Anforderungen!

- 2D oder 3D Grafik (kein ASCII)
- Soundeffekte und Musik
- Mindestens 2 Spieler, min. einer menschlich
- Echtzeit
- Indirekte Steuerung (Point & Click)
- Pausefunktion
- Eigenes Menü (komplett mit der Maus steuerbar, außer Tastatureingaben)
- Spielobjekte
  a. Min. 5 Kontrollierbare
  b. Min. 5 Auswählbare
  c. Min. 5 Nichtkontrollierbare, davon min. 3 Kollidierende
  d. Min. 3 Kontrollierbare, Kollidierende und Bewegliche.

Initiale Anforderungen!

- Min. 5 Statistiken
- Achievements
- Min. 1000 gleichzeitig aktive Spielobjekte der Art (d) möglich (Tech-Demo).
- Speichern und Laden muss potenziell zu jedem Zeitpunkt möglich sein, jedoch nicht zwingend vom Spieler gesteuert werden.

Anforderungen • Funktionale Anforderungen

- 2D oder 3D Grafik (kein ASCII)
- Soundeffekte und Musik
- Mindestens 2 Spieler, min. einer menschlich
- Echtzeit
- Indirekte Steuerung (Point & Click)
- Pausefunktion
- Eigenes Menü (komplett mit der Maus steuerbar, außer Tastatureingaben)
- Spielobjekte
  a. Min. 5 Kontrollierbare
  b. Min. 5 Auswählbare
  c. Min. 5 Nichtkontrollierbare, davon min. 3 Kollidierende
  d. Min. 3 Kontrollierbare, Kollidierende und Bewegliche.

Initiale Anforderungen!

- Min. 5 Statistiken
- Achievements
- Min. 1000 gleichzeitig aktive Spielobjekte der Art (d) möglich (Tech-Demo).
- Speichern und Laden muss potenziell zu jedem Zeitpunkt möglich sein, jedoch nicht zwingend vom Spieler gesteuert werden.
- Min. 10 verschiedene Aktionen
  - z.B.: Laufen o. Fähigkeiten
  - Allen Spielobjekten der Art (d) muss es möglich sein, von jedem beliebigen Punkt in der Welt zu jedem anderen begehbaren Punkt zu gelangen, ohne sich gegenseitig übermäßig zu behindern, festzustecken, usw. (”Pathfinding”).

Anforderungen

Qualitätsanforderungen

- Entwickeln Sie ein gutes Produkt
- Qualität der Grafik ist nicht relevant muss aber in sich stimmig sein
- Akustische Effekte sollen in sich stimmig sein

Randbedingungen

- C# und/oder F# mit .NET
- MonoGame 3.7
- Lauffähig auf Windows 10 (x64)
- Visual Studio Community 2019
- Keine Warnings oder Errors vom Compiler oder ReSharper (wöchentlich), keine Buildfehler.

Initiale Anforderungen!

Anforderungen • Beispiele

Anforderungen • Gegenbeispiele

Ablauf

Ablauf • Übersicht

Woche	Organisation	Entwurf	MS 01	MS 02	MS 03	MS 04	MS 05	Was?	Wann und Wo?
0	✓	✗	✗	✗	✗	✗	✗	• Vorlesung "Organisation und Prozess" (Einführungsveranstaltung) • Gruppeneinteilung abwarten	• Einführungsveranstaltung: 24.10., 14:00 - 16:00, 082-00-006 • Fragebogen: Bis 24.010. 23:59 • Gruppen ab 27.10. in Gitea
1	✓	✓	✗	✗	✗	✗	✗	• Vorlesung "Game Design Document (GDD)" • Abgabe Hausaufgabe	• Vorlesung: 31.10., 14:00 - 16:00, 082-00-006 • Abgabe: 02.11. bis 23:59
2	✗	✓	✓	✗	✗	✗	✗	• Vorlesung "Grundlagen Softwarearchitektur" • Wiederkehrende Aufgaben: Product Owner • Abgabe GDD (beta)	• Vorlesung: 07.11., 14:00 - 16:00, 082-00-006 • Abgabe: 09.11. bis 23:59
3	✗	✓	✓	✗	✗	✗	✗	• Vorlesung "Architektur von Videospielen" • Wiederkehrende Aufgaben: Architektur und Coaching • Wiederkehrende Aufgaben: Qualitätssicherung und Clean-Code	• Vorlesung: 14.11., 14:00 - 16:00, 082-00-006
4	✗	✓	✓	✓	✗	✗	✗	• MS01 erreicht (Spielobjekt in der Welt bewegbar, bewegliche Ansichten, Level laden/speichern, Soundausgabe) • Präsentation der Spielidee • Abgabe Architektur (beta)	• Präsentation: 21.11. 14:00 - 16:00, 082-00-006 • Abgabe: 23.11. bis 23:59
5	✗	✓	✗	✓	✗	✗	✗		
6	✗	✓	✗	✓	✓	✗	✗		
7	✗	✓	✗	✗	✓	✗	✗	• MS02 erreicht (Mehrere Spielobjekte bewegen, Interaktionen zwischen Spielobjekten, Screen-Management, Menü, HUD, Musik)	
8	✗	✗	✗	✗	✓	✗	✗	• Präsentation Programm (beta) • Abgabe Programm (beta)	• Präsentation: 19.12. 14:00 - 18:00, 082-00-006 • Abgabe: 21.12. bis 23:59
9	**Ferien (24.12.2012 - 06.01.2013)**								
10									
11	✗	✗	✗	✗	✓	✓	✗		
12	✗	✗	✗	✗	✗	✓	✓	• Abgabe GDD (final) • MS03 erreicht (KI bzw. Gegnerverhalten, primäre Interaktionen vorhanden, Sieg-/Niederlagebedingungen, Inhalte, Grafik/Soundeffekte)	• Abgabe 16.01. bis 23:59

sopra.informatik.uni-freiburg.de

Ziel

Funktionierende und kompetente Gruppen zusammenstellen.

- Wir teilen Gruppen ein
- Fragebogen
  - Namen und Emails für Dienste abfragen

Ziel

Funktionierende und kompetente Gruppen zusammenstellen.

- Wir teilen Gruppen ein
- Fragebogen
  - Namen und Emails für Dienste abfragen

Ausfüllen bis heute Abend, 23:59

Ziel

Werkzeuge und Vorgehen kennenlernen und Probleme frühzeitig aufdecken.

- Beschreibung auf dem Wiki
- Ungefähr:
  - Werkzeuge installieren und testen
  - Dienste testen
  - Git und Gitea kennenlernen
  - Texte zu Clean Code lesen
  - Ein MonoGame Programm schreiben
- Die Hausaufgabe ist der erste Sprint d.h. es können 5 Punkte erreicht werden

Game Design Document (GDD)

Beschreibung der wesentlichen Merkmale des Spiels.

- Lastenheft
  - Legt die Anforderungen und Rahmenbedingungen eines Produktes (aus Kunden/Anwendersicht) fest
  - Beinhaltet keine Details über die technische Umsetzung
- Hilft bei Aufwandsabschätzung und Organisation
- Details: GDD-Vorlesung, Wiki

Meilenstein

Ein Meilenstein ist ein überprüfbares Ziel, dass zu einem Termin erreicht werden soll.

- Meilensteine teilen den Projektverlauf in Etappen mit überprüfbaren Zwischenzielen ein und erleichtern damit sowohl die Projektplanung als auch die Kontrolle des Projektfortschritts
- Entwicklung gegliedert in 5 Meilensteine
  - Referenzablauf ermöglicht das Erkennen von Problemen
  - Müssen entsprechend des Spielkonzepts modifiziert werden

Meilenstein

Ein Meilenstein ist ein überprüfbares Ziel, dass zu einem Termin erreicht werden soll.

- Meilensteine teilen den Projektverlauf in Etappen mit überprüfbaren Zwischenzielen ein und erleichtern damit sowohl die Projektplanung als auch die Kontrolle des Projektfortschritts
- Entwicklung gegliedert in 5 Meilensteine
  - Referenzablauf ermöglicht das Erkennen von Problemen
  - Müssen entsprechend des Spielkonzepts modifiziert werden

Meilenstein #1

- Spielobjekt in der Welt bewegbar
- Interaktive Kamera
- Karte laden/speichern
- Soundausgabe

Ablauf

Meilenstein

Ein Meilenstein ist ein überprüfbares Ziel, dass zu einem Termin erreicht werden soll.

- Meilensteine teilen den Projektverlauf in Etappen mit überprüfbaren Zwischenzielen ein und erleichtern damit sowohl die Projektplanung als auch die Kontrolle des Projektfortschritts
- Entwicklung gegliedert in 5 Meilensteine
  - Referenzablauf ermöglicht das Erkennen von Problemen
  - Müssen entsprechend des Spielkonzepts modifiziert werden

Meilenstein #1
- Spielobjekt in der Welt bewegbar
- Interaktive Kamera
- Karte laden/speichern

Meilenstein #5
- Spiel ist fehlerfrei
- Spielmechanik ist ausbalanciert

SCRUM als Vorgehensmodell

Scrum

- ist ein iteratives Vorgehensmodell
- gehört zu den agilen Methoden
- sieht sich als Vertreter empirischer Theorien

- Entwicklung wird in Sprints aufgeteilt (1 bis 4 Wochen)
- Es existiert eine fertige Software am Ende jedes Sprints
- Scrum definiert sich über Rollen, Events und Artefakte.

Developer

- Aufgabe: Programmieren, Design, technische Lösung, Projektablauf
- Verantwortung: Produktqualität, Zeit

Product Owner

- Aufgabe: Kundengespräche, Vermarktung, Zielvorgabe, Anforderungserhebung
- Verantwortung: Produktmerkmale, Budget, Markterfolg

Scrum Master

- Aufgabe: Organisation, Mediation, Prozesskontrolle
- Verantwortung: Compliance, Prozess

Product Backlog

- Liste von Anforderungen mit abgeleiteten User Stories
- Nach Wichtigkeit für den Projekterfolg geordnet

User Stories

- Kompakte Beschreibung einer Funktion aus Nutzersicht
- Aufwandsabschätzung mit Story Points
- `Als <Rolle> möchte ich <Funktion>, um <Nutzen>.`
  - Wer oder was benötigt die Funktion?
  - Was ist die Funktion?
  - Welchen Nutzen hat diese Funktion?
- Manchmal Akzeptanzkriterium (wie wird es getestet)

```
#47: Speichern
Als Spieler möchte ich den aktuellen Spielstand zu einem beliebigen Zeitpunkt auf der Festplatte speichern, um später weiterzuspielen.
                                                    Points:8
```

Sprint Backlog

- Liste an Aufgaben für den aktuellen Sprint
- Liste von User Stories mit abgeleiteten Tasks
- Wird für jeden Sprint neu erstellt

Task

- Teilaufgabe einer User Story
- Innerhalb eines Sprints erfüllbar
- Aufwandsabschätzung in Personenstunden

```
#47: Speichern
Als Spieler möchte ich den
aktuellen Spielstand zu
einem beliebigen Zeitpunkt
auf der Festplatte
speichern, um später
weiterzuspielen.
                    Points:8
```

Sprint Backlog

- Liste an Aufgaben für den aktuellen Sprint
- Liste von User Stories mit abgeleiteten Tasks
- Wird für jeden Sprint neu erstellt

#47: *Speichern*
Als Spieler möchte ich den aktuellen Spielstand zu einem beliebigen Zeitpunkt auf der Festplatte speichern, um später

Points:8

Task

- Teilaufgabe einer User Story
- Innerhalb eines Sprints erfüllbar
- Aufwandsabschätzung in Personenstunden

#47-1
Datei auf Festplatte schreiben.

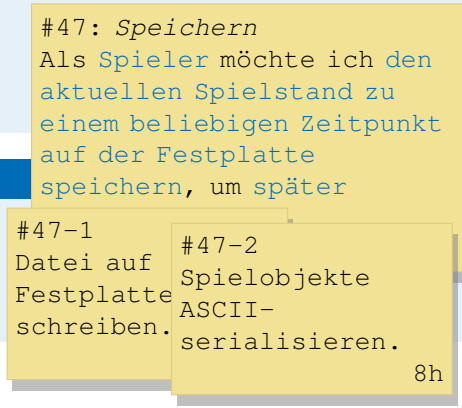
2h

Sprint Backlog

- Liste an Aufgaben für den aktuellen Sprint
- Liste von User Stories mit abgeleiteten Tasks
- Wird für jeden Sprint neu erstellt

Task

- Teilaufgabe einer User Story
- Innerhalb eines Sprints erfüllbar
- Aufwandsabschätzung in Personenstunden

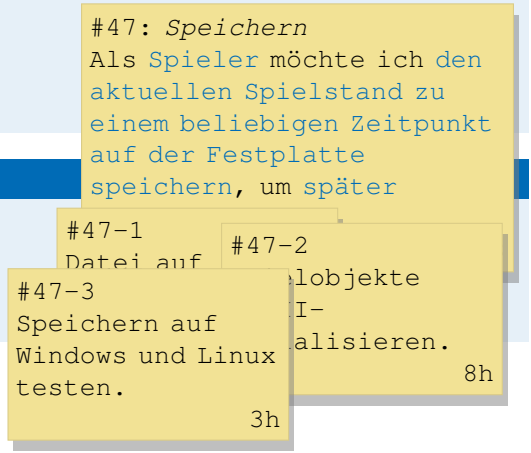


Sprint Backlog

- Liste an Aufgaben für den aktuellen Sprint
- Liste von User Stories mit abgeleiteten Tasks
- Wird für jeden Sprint neu erstellt

Task

- Teilaufgabe einer User Story
- Innerhalb eines Sprints erfüllbar
- Aufwandsabschätzung in Personenstunden



Product Increment

- Neue Produktversion am Ende des Sprints
- Direkt auslieferbar

Definition of Done

- Bestimmt wann eine Aufgabe fertig ist
- Wird vom Team erstellt
- Darf sich im Laufe des Projekts ändern

Product Increment

- Neue Produktversion am Ende des Sprints
- Direkt auslieferbar

Definition of Done

- Bestimmt wann eine Aufgabe fertig ist
- Wird vom Team erstellt
- Darf sich im Laufe des Projekts ändern

Definition of Done Beispiele

- Team einverstanden (incl. PO)
- Auf *master* eingecheckt
- Keine neuen Bugs
- API dokumentiert
- Tests erfolgreich
- Codestyle eingehalten
- Keine *//TODOs*

Product Increment

- Neue Produktversion am Ende des Sprints
- Direkt auslieferbar

Definition of Done

- Bestimmt wann eine Aufgabe fertig ist
- Wird vom Team erstellt
- Darf sich im Laufe des Projekts ändern

Definition of Done Beispiele

- Team einverstanden (incl. PO)
- Auf *master* eingecheckt
- Keine neuen Bugs
- API dokumentiert
- Tests erfolgreich
- Codestyle eingehalten

Sopra DoD

- Item in Gitea ist geschlossen.
- Schätzung und tatsächliche Arbeitszeit ist eingetragen.
- Alle für das Item relevanten Daten sind im Git-Repository (release).
- Tutor und Team haben das Item im Produktinkrement abgenommen.

Sprint Planning

- Vor jedem Sprint
- Länge abhängig von Sprintlänge (ca. 2h je Woche)
- Was wird im nächsten Sprint getan?
  - Product Owner präsentiert die wichtigsten Items aus dem Product Backlog
- Wie wird es umgesetzt?
  - Developer wählen ihre Aufgaben beginnend beim wichtigsten Item
  - Developer zerlegen ausgewählte User Stories in Tasks
  - Developer erstellen neuen Softwareentwurf

Daily Scrum

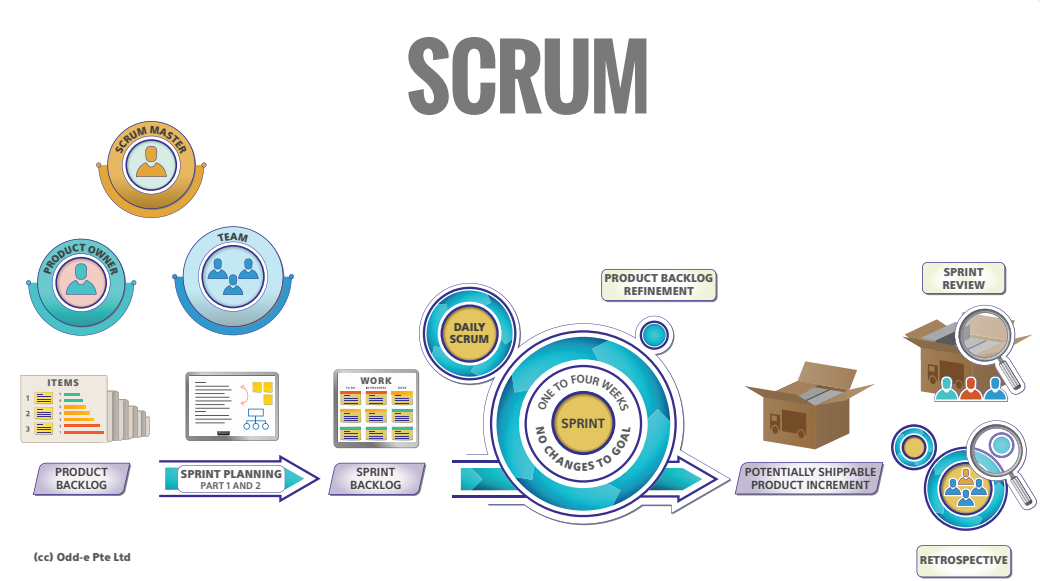
- Tägliches Treffen
- Maximal 15 Minuten
- Developer beantworten nacheinander
  - Was habe ich seit dem letzten Treffen getan?
  - Was plane ich bis zum nächsten Treffen zu tun?
  - Welche Probleme hatte ich und wo benötige ich Hilfe?
- Fragen werden im Daily Scrum nicht beantwortet.

Sprint Review

- Treffen am Ende des Sprints
- Länge abhängig von Sprintlänge (ca. 1h je Woche)
- Developer präsentieren neuen Product Increment
- Product Owner bestimmt welche Tasks und User Stories fertig sind
- Product Owner erklärt, wie gut die Aufwandsabschätzung war

Sprint Retrospective

- Treffen des Teams nach dem Sprint Review
- Länge abhängig von Sprintlänge (ca. 45min je Woche)
- Scrum Master hilft, den Prozess zu verbessern
  - Wie lief es im letzten Sprint hinsichtlich Personen, Beziehungen, Prozessen, Werkzeugen?
  - Muss die Definition of Done angepasst werden?
  - Wie kann die Arbeitsweise des Teams verbessert werden?


SCRUM
SCRUM MASTER
PRODUCT OWNER
TEAM
ITEMS
1
2
3
PRODUCT BACKLOG
SPRINT PLANNING
PART 1 AND 2
WORK
SPRINT BACKLOG
DAILY SCRUM
PRODUCT BACKLOG REFINEMENT
ONE TO FOUR WEEKS
SPRINT
NO CHANGES TO GOAL
POTENTIALLY SHIPPABLE PRODUCT INCREMENT
SPRINT REVIEW
RETROSPECTIVE
(cc) Odd-e Pte Ltd

Scrum im Sopra

- Dozenten sind Kunden
- Tutoren sind Scrum Master
- Sie sind Developer
- Sie können in Ihrer Gruppe Product Owner werden

- Dozenten sind Kunden
- Tutoren sind Scrum Master
- Sie sind Developer
- Sie können in Ihrer Gruppe Product Owner werden

Dozenten sind aber auch Dozenten, stellen Sie Fragen! (z.B.: Poolsprechstunde)

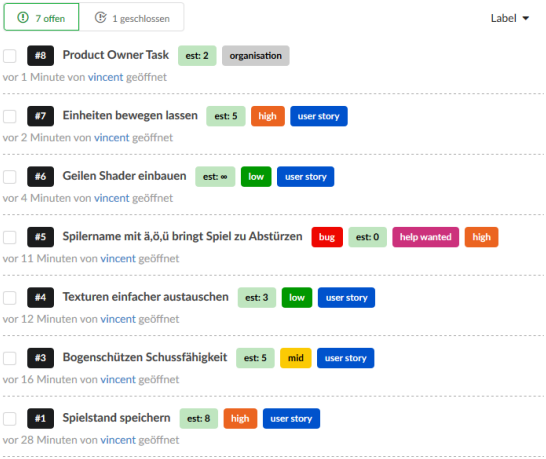
- Dozenten sind Kunden
- Tutoren sind Scrum Master
- Sie sind Developer
- Sie können in Ihrer Gruppe Product Owner werden

Dozenten sind aber auch Dozenten, stellen Sie Fragen! (z.B.: Poolsprechstunde)

Tutoren sind auch Kundenvertreter und Berater.

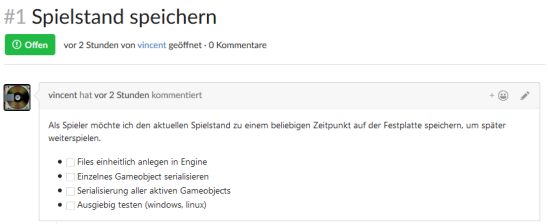
Product Backlog

- User Stories
  - Beschreibender Titel, Story im Text
  - Label `user story`
  - Wichtigkeit `low, mid, high`
  - Zeitschätzung (Gesamtarbeitszeit)
    `est: 0h,1h,2h,3h,5h,...,∞`
- Sonstige Items
  - Label
    `bug, organisation, task, question, ...`
  - Wichtigkeit `low, mid, high`
  - Zeitschätzung (Gesamtarbeitszeit)
    `est: 0h,1h,2h,3h,5h,...,∞`



User Story

- Beschreibender Titel
- Vollständige User Story in der Beschreibung

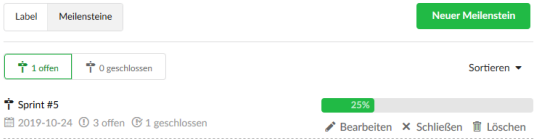


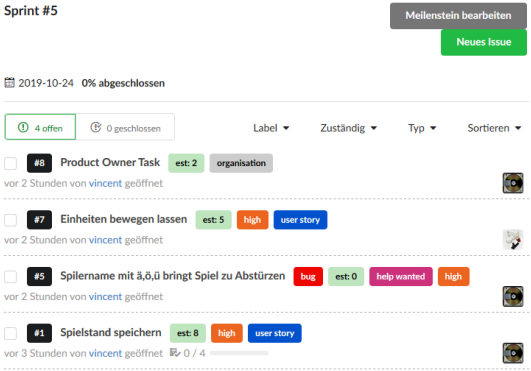
Task

- Ticket mit Tag task
  - Developer dem Task zuweisen (nicht der User Story)
  - Zeitschätzung
  - Abhängigkeit zur Userstory dokumentieren
    z.B.: mit Kommentar oder
    Gitea-Abhängigkeiten
- Checkboxen unter der User Story
  - `- [ ] ein Task`
  - Für User Stories die von einem Developer erledigt werden

Sprint Backlog

- Liste der im Sprint zu erledigenden Items
- Jedes Item ist dem Sprint zugeordnet (ein Meilenstein in Gitea)
- Jedem Item ist ein Developer zugeordnet





Gruppentreffen

Zweistündiges Treffen mit dem Tutor als Moderator und Scrum Master

1. Sprint Review (ca. 30min)
2. Sprint Retrospective (ca. 15min)
3. Sprint Planning (ca. 45min)

Sprint Review

- Developer führen Product Increment vor
- Gruppe entscheidet was gemäß DoD (nicht) erledigt ist?
- Wie gut war die Zeitschätzung?
- Wird der nächste Meilenstein erreicht?

Gruppentreffen

Zweistündiges Treffen mit dem Tutor als Moderator und Scrum Master

1. Sprint Review (ca. 30min)
2. Sprint Retrospective (ca. 15min)
3. Sprint Planning (ca. 45min)

Sprint Retrospective

- Was lief in diesem Sprint gut oder schlecht?
  - Probleme mit oder Lob für Gruppenmitglieder, Prozess, Zeiteinteilung oder Tools
- Bei Bedarf DoD anpassen
- Entschlüsse schriftlich festhalten (z.B. in einer Datei im Git)

Gruppentreffen

Zweistündiges Treffen mit dem Tutor als Moderator und Scrum Master

1. Sprint Review (ca. 30min)
2. Sprint Retrospective (ca. 15min)
3. Sprint Planning (ca. 45min)

Sprint Planning

- Verfügbare Arbeitszeit für diesen Sprint festlegen
- Product Owner stellt vor wie das nächste Increment aussehen soll (was soll dann funktionieren)
- Product Backlog angefangen bei der wichtigsten User Story durchgehen:
  - Aufwand schätzen (falls nötig)
  - Ins Sprint Backlog verschieben
  - Wiederholen bis verfügbare Arbeitszeit aufgebraucht
- Items im Sprint Backlog an die Gruppenmitglieder verteilen

Product Owner (Woche 2)

- Pflege des Product Backlog
  - User Stories an Entwicklungsstand anpassen, und nach Wichtigkeit ordnen
  - Weitere User Stories aus GDD extrahieren
- Gruppentreffen vorbereiten:
  - Was ist nach DoD fertig?
  - Wie waren die Aufwandsabschätzungen?
  - Product Increment vorbereiten

Architektur (Woche 3)

- Pflege der Architektur
- Schnittstellen und Grenzen definieren und deren Einhaltung sicherstellen

Qualitätssicherung (Woche 3)

- Code auf Clean Code Richtlinien prüfen
- Code Reviews machen und Ergebnisse im Gruppentreffen präsentieren
- ReSharper Konformität herstellen

- Alle Recurring Tasks müssen verteilt werden
- Ticket mit Zeitschätzung (max. 2h) je Aufgabe und Sprint

Was Nun?

Was Nun?

Ab sofort

- Fragebogen bis spätestens heute Abend 23:59 ausfüllen.
- Auf Gruppeneinteilung warten (bis Sonntag)

Danach

- Hausaufgaben machen
- Termin für Gruppentreffen ausmachen (Mo-Do)
- Spielidee entwickeln

Was Nun?

Ab sofort

- Fragebogen bis spätestens heute Abend 23:59 ausfüllen.
- Auf Gruppeneinteilung warten (bis Sonntag)

Überlegen Sie jetzt ob Sie dieses Semester genug Zeit haben und zu allen Pflichtterminen anwesend sein können.

Danach

- Hausaufgaben machen
- Termin für Gruppentreffen ausmachen (Mo-Do)
- Spielidee entwickeln

Was Nun?

Ab sofort

- Fragebogen bis spätestens heute Abend 23:59 ausfüllen.
- Auf Gruppeneinteilung warten (bis Sonntag)

Überlegen Sie jetzt ob Sie dieses Semester genug Zeit haben und zu allen Pflichtterminen anwesend sein können.

Danach

- Hausaufgaben machen
- Termin für Gruppentreffen ausmachen (Mo-Do)
- Spielidee entwickeln

Stellen Sie sicher, dass Ihr Postfach nicht voll ist und Emails von der Uni ankommen (Spamfilter)!