Softwarepraktikum

Vincent Langenfeld, Daniel Dietsch

November 5, 2020

Kontext

[Donnerstag Nachmittag, Büro von Mitarbeiter S.]

Chef:
S., die WiKe AG (Wichtiger Kunde) möchte, dass wir ein Computerspiel für sie produzieren.

S.:
Ein Computerspiel? Das ist etwas Neues, haben wir da Leute für?

Chef:
Nein, aber man soll sich ja nie etwas Neuem gegenüber verschließen. S., Sie und Ihr Team, Sie bekommen das hin!

Kontext

[Donnerstag Nachmittag, Büro von Mitarbeiter S.]

Chef:
S., die WiKe AG (Wichtiger Kunde) möchte, dass wir ein Computerspiel für sie produzieren.

S.:
Ein Computerspiel? Das ist etwas Neues, haben wir da Leute für?

Chef:
Nein, aber man soll sich ja nie etwas Neuem gegenüber verschließen. S., Sie und Ihr Team, Sie bekommen das hin!

- Im Softwarepraktikum sind Sie Mitarbeiter *S.*
- Im Softwarepraktikum sind Sie Dozenten die Vertreter der WiKe AG (Wichtiger Kunde AG)
- Im Softwarepraktikum simulieren wir Teile der Dynamik zwischen Auftraggeber und Auftragnehmer Softwareentwicklung

Vorgehen

```
Mitarbeiter S. beginnt mit der Arbeit am Projekt.
Sie beginnen damit ...
```

Vorgehen

```
Mitarbeiter S. beginnt mit der Arbeit am Projekt.
Sie beginnen damit ...
```

- die *Domäne* analysieren und verstehen

Vorgehen

```
Mitarbeiter S. beginnt mit der Arbeit am Projekt.
Sie beginnen damit ...
```

- die *Domäne* analysieren und verstehen
- die *Anforderungen* erheben und verstehen

Vorgehen

```
Mitarbeiter S. beginnt mit der Arbeit am Projekt.
Sie beginnen damit ...
```

- die *Domäne* analysieren und verstehen
- die *Anforderungen* erheben und verstehen
- das *Umsetzung* konzipieren

Agenda

- Organisatorisches
- Vorgehen
- Gesamtablauf
- Vorgehensmodell: Scrum
- Scrum im Sopra

Organisatorisches

Lernziele

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

Lernziele

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

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

Team

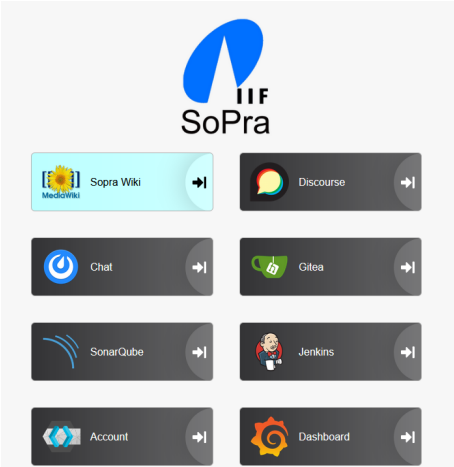
- *Tutoren*
  Samuel Becker, Antonia Hinkerohe, Leon Holtmeier, Ming Hu, Victor Maier, Yannick Vogt, Jonas Werner
- *Infrastruktur*
  Julian Löffler
- *Dozenten*
  Daniel Dietsch, Vincent Langenfeld
- *Verantwortlich*
  Prof. Dr. A. Podelski

Organisation

- Gruppen mit 5-7 Studenten
- 6 ECTS (180h) über 15 Wochen
  - 14 Arbeitsabschnitte (Sprints), je eine Woche
- 4 Vorlesungen
- Termine
  - 3 Abgaben
  - 1 Vortrag vor den Dozenten
  - 3 Präsentationen vor dem gesamten Kurs (Do. 14-18 Uhr)
  - Wöchentliches Gruppentreffen

Dienste und Werkzeuge

- *Wiki*
- *Fragen und Informationen*
  Gruppenmitglieder, Tutoren, Wiki, Discourse, Sprechstunde und Mattermost
- *Primärdienste*
  Gitea, Git, Jenkins, Sonar, Dashboard, Mailinglisten (`sopra-crew@...`, `sopra-XX@...`)
- *Sekundärdienste*
  Mattermost, Discourse
- *Werkzeuge*
  C#, .NET Core 3.1, Monogame 3.8, Visual Studio Community 19, Resharper



services.sopranium.de

Dienste und Werkzeuge

Pandemie

- *Gruppentreffen*
  - *Big Blue Button* in Ilias (je Gruppe ein Raum)
- *Vorlesungen*
  - *Vorlesungen* je Montags als Aufzeichnung
  - *Fragesession* am entsprechenden Slot
  - Links immer auf dem Wiki
- *Sprechstunden*
  - Nach Absprache (Mention, Email, Mattermost)

Mitarbeit

- Kontinuierliche Mitarbeit während der Sprints:
  - *Commits* im Git-Repository und
  - *Aktivität* (Tickets, Kommentare, etc.) in Gitea
  - Es darf max. 2 Sprints 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. 2 Sprints 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. 2 Sprints 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. 2 Sprints 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 Kriterien:

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

Vorgehen

```
Mitarbeiter S. beginnen mit der Arbeit am Projekt.
Sie beginnen damit ...
```

- die *Domäne* analysieren und verstehen
- die *Anforderungen* erheben und verstehen
- das *Produkt* entwerfen

Domäne

Unter einer (Problem)-Domäne versteht man [...] in der Softwaretechnik ein abgrenzbares Problemfeld oder einen bestimmten Einsatzbereich für Computersysteme oder Software.

Domäne

Unter einer (Problem)-Domäne versteht man [...] in der Softwaretechnik ein abgrenzbares Problemfeld oder einen bestimmten Einsatzbereich für Computersysteme oder Software.

- *Entität* (z.B.: Tastatur, Spiel, Spieler),
- *Funktionen* (z.B.: gedrückte Buttons, Spielermotivation)
- *Ereignisse* (z.B.: Einschlafen, Einschalten)
- *Interaktionen* (z.B.: Spieler drückt Taste)

Domäne

Unter einer (Problem)-Domäne versteht man [...] in der Softwaretechnik ein abgrenzbares Problemfeld oder einen bestimmten Einsatzbereich für Computersysteme oder Software.

- *Entität* (z.B.: Tastatur, Spiel, Spieler),
- *Funktionen* (z.B.: gedrückte Buttons, Spielermotivation)
- *Ereignisse* (z.B.: Einschlafen, Einschalten)
- *Interaktionen* (z.B.: Spieler drückt Taste)
- Aus Sicht verschiedener *Stakeholder*
  - Stakeholder sind in irgendeiner Weise von der Domäne betroffen
  - z.B.: Programmierer, Spieledesigner, Freiwillige Selbstkontrolle (FSK)

Domäne

Vier grundlegende Elemente [eines Spiels][1]

- *Ästhetik*
- *Handlung*
- *Mechanik*
- *Technologie*

[1] Jesse Schell, The Art of Game Design, 2008, ISBN 978-0-12-369496-6

Domäne

Vier grundlegende Elemente [eines Spiels][1]

- *Ästhetik*
- *Handlung*
- *Mechanik*
- *Technologie*

- Jede Sicht lässt sich weiter zerlegen
  - Mechanik z.B.: Spielobjekte und deren Eigenschaften und Interaktionen
  - Handlung z.B.: Akte, Spannungskurve und überleitende Ereignisse
- Die Anforderungen (nächster Schritt) sollten jede Sicht berücksichtigen

[1] Jesse Schell, The Art of Game Design, 2008, ISBN 978-0-12-369496-6

Anforderungen[2]

Anforderungen ist eine Aussage darüber, was man von einem Softwaresystem als Eigenschaften erwartet.

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

Anforderungen[2]

Anforderungen ist eine Aussage darüber, was man von einem Softwaresystem als Eigenschaften erwartet.

Gewöhnlich werden *3 Arten* von Anforderungen unterschieden:

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

Anforderungen[2]

Anforderungen ist eine Aussage darüber, 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.

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

Anforderungen[2]

Anforderungen ist eine Aussage darüber, 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.

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

Anforderungen[2]

Anforderungen ist eine Aussage darüber, 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.

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

Anforderungen

```
Chef: Die WiKe AG hat die
folgenden Anforderungen
geschickt.
```

- 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)

Anforderungen

```
Chef: Die WiKe AG hat die
folgenden Anforderungen
geschickt.
```

- 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

Anforderungen

```
Chef: Die WiKe AG hat die
folgenden Anforderungen
geschickt.
```

- 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
- 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

```
Chef: Die WiKe AG hat die
folgenden Anforderungen
geschickt.
```

- 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
- 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 Core 3.1
- MonoGame 3.8
- Lauffähig auf Windows 10 (x64)
- Visual Studio Community 2019
- Keine Warnings oder Errors vom Compiler oder ReSharper (wöchentlich), keine Buildfehler.

```
Chef: Die WiKe AG hat die
folgenden Anforderungen
geschickt.
```

Anforderungen · Beispiele

Anforderungen · Gegenbeispiele

Anforderungen

Game Design Document (GDD)

Beschreibung der *wesentlichen Merkmale* des Spiels.

- Lastenheft
  - Dokumentiert die Anforderungen und Rahmenbedingungen eines Produktes
  - Aus Sicht des *Anwenders* oder *Kunden*
  - Beinhaltet keine Details über die technische Umsetzung
- Hilft bei Aufwandsabschätzung und Organisation
- Details: GDD-Vorlesung, Wiki

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: 5.11., 14:00 - 16:00, TBA • Fragebogen: Bis 5.11. 23:59 • Gruppen ab 7.11. in Gitea
1	✓	✓	✗	✗	✗	✗	✗	• Vorlesung "Game Design Document (GDD)" • Abgabe Hausaufgabe	• Vorlesung: 12.11., 14:00 - 16:00, TBA • Abgabe: 14.11. bis 23:59
2	✗	✓	✓	✗	✗	✗	✗	• Vorlesung "Grundlagen Softwarearchitektur" • Wiederkehrende Aufgaben: Product Owner • Abgabe GDD (beta)	• Vorlesung: 19.11., 14:00 - 16:00, TBA • Abgabe: 21.11. bis 23:59
3	✗	✓	✓	✗	✗	✗	✗	• Vorlesung "Architektur von Videospielen" • Wiederkehrende Aufgaben: Architektur und Coaching • Wiederkehrende Aufgaben: Qualitätssicherung und Clean-Code	• Vorlesung: 26.11., 14:00 - 16:00, TBA
4	✗	✓	✓	✓	✗	✗	✗	• MS01 erreicht (Spielobjekt in der Welt bewegbar, bewegliche Ansichten, Level laden/speichern, Soundausgabe) • Präsentation der Spielidee • Abgabe Architektur (beta)	• Präsentation: 03.12. 14:00 - 16:00, TBA • Abgabe: 05.12. bis 23:59
5	✗	✓	✗	✓	✗	✗	✗		
6	✗	✓	✗	✓	✓	✗	✗		
7	✗	✓	✗	✗	✓	✗	✗	• MS02 erreicht (Mehrere Spielobjekte bewegen, Interaktionen zwischen Spielobjekten, Screen-Management, Menü, HUD, Musik)	
8	**Ferien (23.12.2020 - 06.01.2021)**								
9	✗	✗	✗	✗	✓	✗	✗	• Präsentation Programm (beta) • Abgabe Programm (beta)	TBA
10	✗	✗	✗	✗	✓	✓	✗		
11	✗	✗	✗	✗	✓	✓	✓	• Abgabe GDD (final) • MS03 erreicht (KI bzw. Gegnerverhalten, primäre Interaktionen vorhanden, Sieg-/Niederlagebedingungen, Inhalte, Grafik/Soundeffekte, Techdemo-Beta)	• Abgabe 23.01. bis 23:59
12	✗	✗	✗	✗	✗	✓	✓		
13	✗	✗	✗	✗	✗	✓	✓	• MS04 erreicht (finale Version vorhanden) • Abgabe Architektur (final)	• Abgabe: 06.02. bis 23:59

Ablauf

Gruppeneinteilung

- Ziel: *Funktionierende* und *kompetente* Gruppen zusammenstellen
- Fragebogen
  - Namen und Emails für Dienste abfragen
  - Bis heute Abend, 23:59
- Wir teilen Gruppen ein
- Einteilung bis Sonntag

Ablauf

Hausaufgabe

- 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

Ablauf

Gliederung

Meilenstein

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

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

Gliederung

Meilenstein

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

- Meilensteine teilen den Projektverlauf in ein und erleichtern damit sowohl die
  - Je eine *Etappen mit überprüfbaren Zwischenzielen*.
  - Erleichtert damit die *Projektplanung* und
  - 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

Gliederung

Meilenstein

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

- Meilensteine teilen den Projektverlauf in ein und erleichtern damit sowohl die
  - Je eine *Etappen mit überprüfbaren Zwischenzielen*.
  - Erleichtert damit die *Projektplanung* und
  - 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 *auslieferbare* 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

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

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

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

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

Task

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

#47-1
Datei auf Festplatte schreiben.

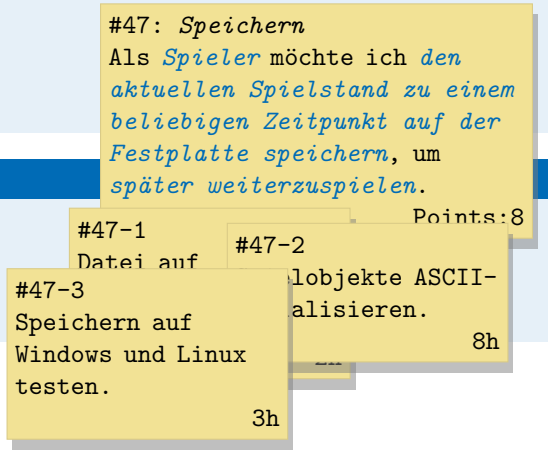
#47-2
Spielobjekte ASCII-serialisieren.
8h

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
- Das Item ist in Gitea geschlossen.
- Im Item sind die geschätzte und die tatsächliche Arbeitszeit eingetragen.
- Alle für das Item relevanten Dateien sind im aktuellen Stand des remote release Branch integriert.
- Der Tutor hat die Fertigstellung des Items im Sprint Review anhand des aktuellen Standes des remote release Branch bestätigt.

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

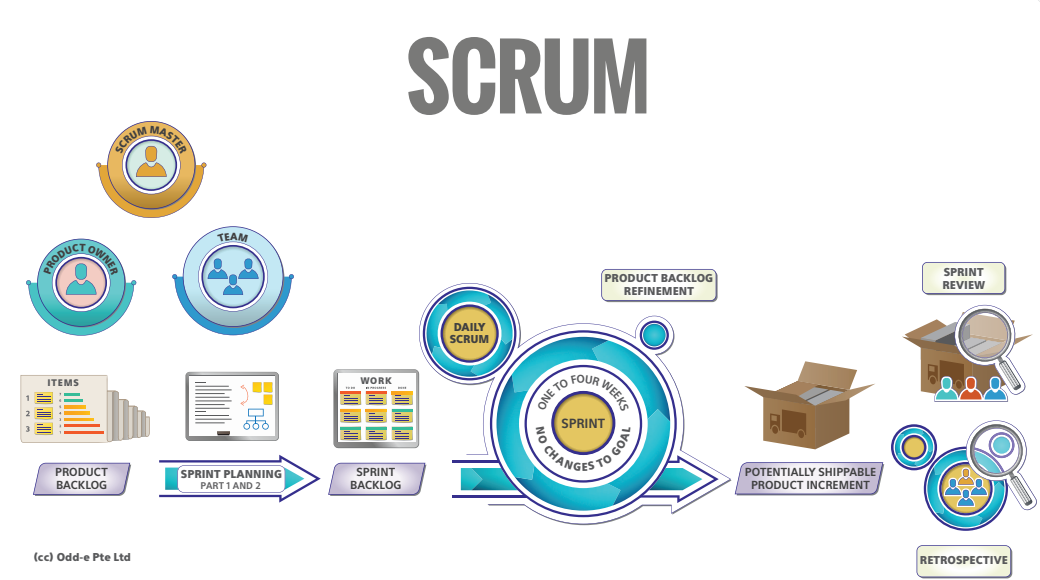
- *Tägliches* Treffen
- Maximal 15 Minuten
- *Developer* beantworten nacheinander
  - Was habe ich seit dem letzten Treffen getan?
  - Was plane ich bis zum nächsten Treffen zu tun?
  - Welche Probleme hatte ich und wo benötige ich Hilfe?
- Fragen werden im Daily Scrum nicht beantwortet

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, denen sie *Fragen stellen* können und die Ihnen helfen (z.B.: Bei Treffen, in Mattermost, oder als Mention in einem Gitea-Ticket)

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

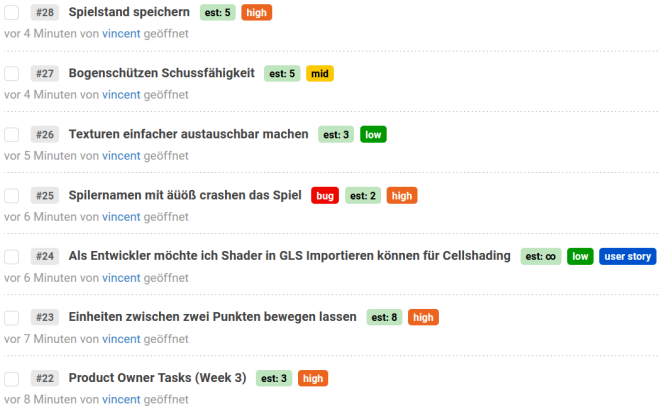
Dozenten sind aber auch Dozenten, denen sie *Fragen stellen* können und die Ihnen helfen (z.B.: Bei Treffen, in Mattermost, oder als Mention in einem Gitea-Ticket)

Tutoren sind auch *Kundenversteher* und *Berater*.

Product Backlog

Menge von *Items*

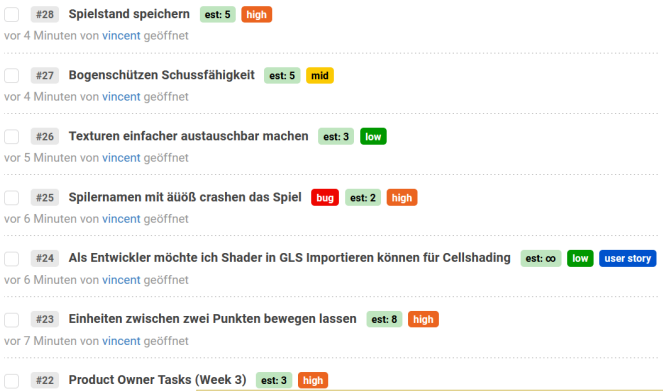
- Beschreibender Titel
- Ausführliche Beschreibung
- Labels nach Funktion
  `bug, question, ...`
- Wichtigkeit
  `low, mid, high`
- Zeitschätzung (Gesamtarbeitszeit)
  `est: 0h,1h,2h,3h,5h,...,∞`



Product Backlog

Menge von *Items*

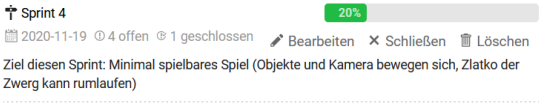
- Beschreibender Titel
- Ausführliche Beschreibung
- Labels nach Funktion
  `bug, question, ...`
- Wichtigkeit
  `low, mid, high`
- Zeitschätzung (Gesamtarbeitszeit)
  `est: 0h,1h,2h,3h,5h,...,∞`

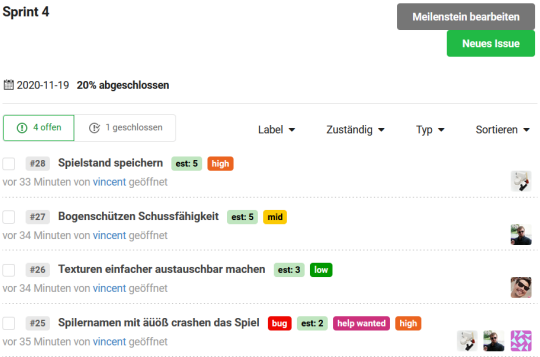


Annahme: Das *Game Design Document* entspricht einer Sammlung von *User Stories*, jedes abgeleitete *Item* einem *Task*.

Sprint Backlog

- Liste der im Sprint zu erledigenden *Items*
- Jedes *Items* ist dem *Sprint* zugeordnet (ein Meilenstein in Gitea)
- Jedem *Items* 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:
  - Gemeinsam konkrete *Funktion* besprechen und *Aufwand* schätzen.
  - Ins Sprint Backlog verschieben
  - Wiederholen bis verfügbare Arbeitszeit aufgebraucht
- Items im Sprint Backlog an die Gruppenmitglieder verteilen

Product Owner (ab Woche 2)

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

Architektur (ab Woche 3)

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

Qualitätssicherung (ab 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)
- Mit der Hausaufgabe anfangen

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)
- Mit der Hausaufgabe anfangen

Danach

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

Überlegen Sie *jetzt* ob Sie:

- dieses Semester genug Zeit haben und zu allen Pflichtterminen anwesend sein können
- Zugriff auf einen zum Entwickeln geeigneten Rechner haben

Was nun?

Ab sofort

- Fragebogen bis spätestens *heute Abend 23:59* ausfüllen.
- Auf Gruppeneinteilung warten (bis Sonntag)
- Mit der Hausaufgabe anfangen

Danach

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

Überlegen Sie *jetzt* ob Sie:

- dieses Semester genug Zeit haben und zu allen Pflichtterminen anwesend sein können
- Zugriff auf einen zum Entwickeln geeigneten Rechner haben

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