



Softwarepraktikum

WS 2018/2019



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



ORGANISATION



Team

- Tutor
Samuel Roth
- Dozenten
Vincent Langenfeld
- Verantwortung
Prof. Dr. A. Podelski

Organisation

- 6 ECTS (180h) in 14 Wochen.
- Teams mit ca. 5 Studenten (nicht B.Sc. Informatik).
- 6 Abgaben, 3 Präsentationen.
- Wöchentliches Gruppentreffen mit dem Tutor.
 - Dabei Feedback zum Projektfortschritt.
- Keine regelmäßige Vorlesung.



Organisation

- Termine
 - Betreuung
Jederzeit in Büro (052-00-005).
 - Präsentationen
Mi. 14:00 – max. 16:00 Uhr hier.
 - Abgaben
Samstags bis 23:59 Uhr.

Dienste, Werkzeuge, Informationen

- **Wiki**
- **Informationen**
Wiki, Gruppenmitglieder, Tutoren, Poolbetreuung
- **Primäre Dienste**
Git, Gitea, Mailinglisten (sopra-crew@..., sopraXX@...), Poolrechner
- **Sekundäre Dienste**
Sonar, Doxygen
- **Werkzeuge**
C#, F#, .NET 4.7, MonoGame 3.6, Visual Studio Enterprise 2015, ReSharper 2017.1

Zulassung

- **Kontinuierliche Mitarbeit**
 - Belegt durch hinreichend viel **messbare** Aktivität (**Git**, **Gitea**).
 - **Verbrauchte Zeit** und **Aufwandsabschätzung** muss im Gitea angegeben werden.
 - Max. 2 Wochen nicht kontinuierlich mitarbeiten.
- **Gruppentreffen**
 - Anwesenheitspflicht.
 - 1x pro Woche 2h.
 - Max. 1x fehlen.
- **Präsentationen**
 - Anwesenheitspflicht.

Benotung

- 50% **Endprodukt**
 - Entspricht das Produkt den Anforderungen?
 - Ist das Produkt fehlerfrei (d.h. wir finden keine Fehler)?
 - Sind die Softwarequalitätsanforderungen erfüllt?
 - Wie ist die Qualität der finalen Artefakte?
- 50% **Einzelleistung**
 - Wurde die zugeteilte Arbeit *erfolgreich* erledigt?
 - Ab nächster Woche pro Woche max. 5 Punkte.
- Wenn Endprodukt oder Einzelleistung 5.0, dann Endnote 5.0.

Lernziele

- Selbstständiges Einarbeiten in unbekanntes Gebiet.
- Arbeiten im Team.
- Umgang mit Komplexität.
- Praktische Anwendung softwaretechnischer Prinzipien.



Simulation eines Softwareentwicklungsprozesses anhand eines Computerspiels.

THEMA

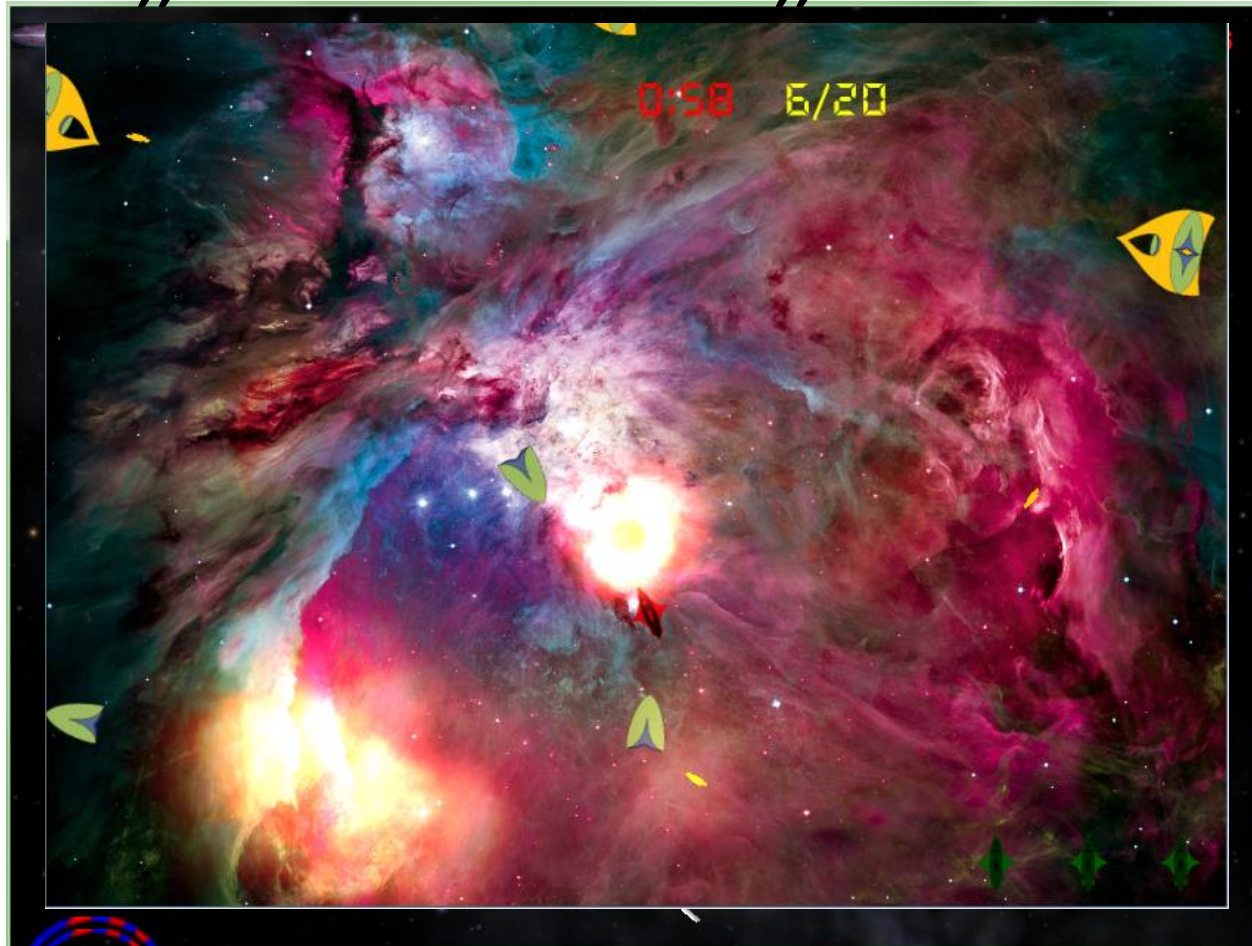
Sie haben die Wahl

- Brettspiele
 - WS 10/11: „Khet“
- Space Shoot ‘em Up
 - WS 11/12: „Xenon“ und „Triton“
- Platformer
 - WS 13/14, WS 14/15, WS 15/16, WS 16/17, WS17/18

Beispiel Brettspiel: „Khet“



Beispiel Space Shoot 'em Up: „Xenon“ und „Triton“





Beispiel Platformer: „Paper“ + „The man ran“





ANFORDERUNGEN



Was sind Anforderungen?

„Anforderungen legen die qualitativen und quantitativen Eigenschaften eines Produkts aus der Sicht des Auftraggebers fest.“

Helmut Balzert. Lehrbuch der Softwaretechnik.
2. Auflage. 2000. ISBN 3-8274-0480-0

Was sind Anforderungen?

- **3 Arten** von Anforderungen
 - **Funktionale Anforderungen** definieren die funktionalen Effekte, die eine Software auf ihre Umgebung ausüben soll.
 - **Qualitätsanforderungen** beschreiben zusätzliche Eigenschaften, die diese funktionalen Effekte haben sollen.
 - **Randbedingungen** beschränken die Art, auf die die Software funktionale Anforderungen erfüllt, oder auf die die Software entwickelt wird.

Funktionale Anforderungen

- 2D oder 3D Grafik (kein ASCII).
- Potenziell zu jedem Zeitpunkt Speichern / Laden, muss aber nicht zwangsläufig vom Spieler gesteuert sein.
- Pausefunktion.
- Eigenes Menü.
- Sound.
- Alleinstellungsmerkmal.

Funktionale Anforderungen: Brettspiele

- Min. 2 Spieler, min. einer davon „menschlich“, min. ein KI-Spieler.
- Rundenbasiert oder Echtzeit.
- Komplexer als „Mensch ärgere dich nicht“.
- Exotisch.

Funktionale Anforderungen: Space Shoot 'em Up

- Min. 1 Spielobjekt wird zu jeder Zeit gesteuert.
- Echtzeit.
- Zusätzliche Mechaniken, die über einfache Bewegung und Schießen hinaus gehen.
- Mehrere Level.
- Bossgegner.

Funktionale Anforderungen: Platformer

- Min. 1 Spielfigur wird zu jeder Zeit gesteuert.
- Echtzeit.
- Direkt gesteuerte Spielfiguren müssen „springen“ können.
- Korrektes Springen muss für die Bewältigung von Leveln notwendig sein.
- Mehrere Level oder ein sehr langes Level.

Qualitätsanforderungen

- Entwickeln Sie ein **gutes** Produkt.
- Qualität der Grafik ist nicht relevant.
- Grafiken sollen in sich stimmig sein.
- Akustische Effekte sollen in sich stimmig sein.
- Richtlinien zur Bedienbarkeit von Computerspielen beachten (Wiki-Artikel „Usability beim Spieldesign“).

Randbedingungen

- Programmiersprache C# und/oder F# mit .NET 4.5.
- MonoGame.
- Auf Windows 7 x86/x64 lauffähig.
- Visual Studio.
- Keine Warnings oder Errors vom Compiler oder ReSharper (wöchentlich), keine Buildfehler.



ABLAUF



Woche	Organi- sation	Ent- wurf	MS 01	MS 02	MS 03	MS 04	MS 05	Was?	Wann und Wo?
0	✓	✗	✗	✗	✗	✗	✗	- Einführungsveranstaltung besuchen - Gruppeneinteilung abwarten	- Einführungsveranstaltung: 17.10., 14:00 - max. 18:00, 082-00-006 - Gruppeneinteilung: Im Anschluss an die Einführungsveranstaltung
1	✓	✓	✗	✗	✗	✗	✗	- Ausarbeitung der "Zusammenfassung des Spiels" - Abgabe Hausaufgabe	Abgabe: 27.10. bis 23:59
2	✗	✓	✓	✗	✗	✗	✗	Abgabe GDD (beta)	Abgabe: 03.11 bis 23:59
3	✗	✓	✓	✗	✗	✗	✗	- Besprechung Klassendiagramm mit Tutor - Präsentation des aktuellen Stands (Spielidee)	Präsentation: 07.11. 14:00 - 15:00, 082-00-006
4	✗	✓	✓	✓	✗	✗	✗	- MS01 erreicht (Spielobjekt in der Welt bewegbar, bewegliche Ansichten, Level laden/speichern, Soundausgabe) - Abgabe Klassendiagramm (beta)	Abgabe: 17.11. bis 23:59
5	✗	✓	✗	✓	✗	✗	✗		
6	✗	✓	✗	✓	✓	✗	✗	MS02 erreicht (Mehrere Spielobjekte bewegen, Interaktionen zwischen Spielobjekten, Screen-Management, Menü, HUD, Musik)	
7	✗	✓	✗	✗	✓	✗	✗	Abgabe GDD (final)	Abgabe: 08.12. bis 23:59
8	✗	✗	✗	✗	✓	✗	✗	- Präsentation Programm (beta) - Abgabe Programm (beta)	- Präsentation: 12.12. 14:00 - 15:00, 082-00-006 - Abgabe: 15.12. bis 23:59
9	✗	✗	✗	✗	✓	✓	✗	MS03 erreicht (KI bzw. Gegnerverhalten, primäre Interaktionen vorhanden, Sieg-/Niederlagebedingungen, Inhalte, Grafik/Soundeffekte)	
10	Ferien (23.12.2018 - 06.01.2019)								
11									
12	✗	✗	✗	✗	✗	✓	✓		
13	✗	✗	✗	✗	✗	✓	✓		
14	✗	✗	✗	✗	✗	✓	✓	MS04 erreicht (finale Version vorhanden)	
15	✗	✗	✗	✗	✗	✗	✓	Abgabe Klassendiagramm (final)	Abgabe: 02.02. bis 23:59
16	✗	✗	✗	✗	✗	✗	✓	- MS05 erreicht (Fehlerbehebung & Balancing) - Präsentation Programm (final) - Abgabe Programm (final)	- Präsentation: 06.02. 14:00 - TBA, 082-00-006 - Abgabe: 09.02. bis 23:59

Hausaufgabe

- Genaue Beschreibung auf dem Wiki
- Ungefähr:
 - Werkzeuge installieren, Dienste testen.
 - Git kennenlernen.
 - Texte auf Wiki lesen
 - Clean Code, Dokumentation, Usability, Git und Gitea
 - Monogame.
- Demo

Game Design Document

- GDD beschreibt die wesentlichen Merkmale des Spiels für den Auftraggeber.
 - Ähnlich zu **Lastenheft**.
- Details im Anschluss und im Wiki.
- „Hall of Fame“.

Erster Entwurf der SW-Architektur

- Software-Architektur beschreibt **die Strukturen** eines Software-Systems durch **Bausteine** und deren **Beziehungen** und **Interaktionen** untereinander.
- Bei uns reduziert auf **Klassendiagramm** in UML.

Umsetzung

- Grob gegliedert in **5 Milestones** (MS)
 - Unsere MS beschreiben einen **Referenzablauf**.
 - Behalten Sie den Termin bei, definieren Sie sich jedoch **passende Inhalte** selbst.
 - Ob MS erreicht ist wird im Gruppentreffen mit dem Tutor entschieden.



SCRUM ALS VORGEHENSMODELL

Scrum

- Scrum...
 - ist ein **iteratives Vorgehensmodell**.
 - gehört zu den **agilen** Methoden.
 - sieht sich als Vertreter **empirischer** Theorien.
- Scrum besteht aus **Rollen, Events, Artefakten, Regeln**.
- Entwicklungszeit wird in **Sprints** aufgeteilt.
 - **Länge** von einer Woche bis zu einem Monat.

Rollen

- Ein Scrum-Team besteht aus
 - **Developers**
Organisieren sich selbst, verantwortlich für Qualität und Entwicklung des Produkts.
 - **Product Owner**
Sammelt und priorisiert Anforderungen, verwaltet Budget, ist verantwortlich für kommerziellen Erfolg.
 - **Scrum Master**
Löst organisatorische Probleme, ist verantwortlich für den Prozess, berät das Team.

Artefakte

- **Product Backlog**
 - Liste von **User Stories** mit abgeleiteten **Tasks**.
 - Ist geordnet nach Priorität der Items relativ zum nächsten Ziel.

Artefakte

- User Stories

- Beschreibung aus Sicht des Benutzers.
- Möglichst kurz (nur ein Satz).
- Maximal in einem Sprint zu erledigen.
- Oft nach **Vorlage**, z.B.
 - „Als <Rolle> möchte ich <Ziel/Wunsch>, um <Nutzen>“
 - „Um <Nutzen> als <Rolle> zu erhalten, möchte ich <Ziel/Wunsch>“

Artefakte

- Sprint Backlog
 - Liste von **User Stories** mit abgeleiteten **Tasks** und **Aufwandsabschätzungen**.
 - Für jeden Sprint wird ein neues Sprint Backlog aus dem Product Backlog abgeleitet.

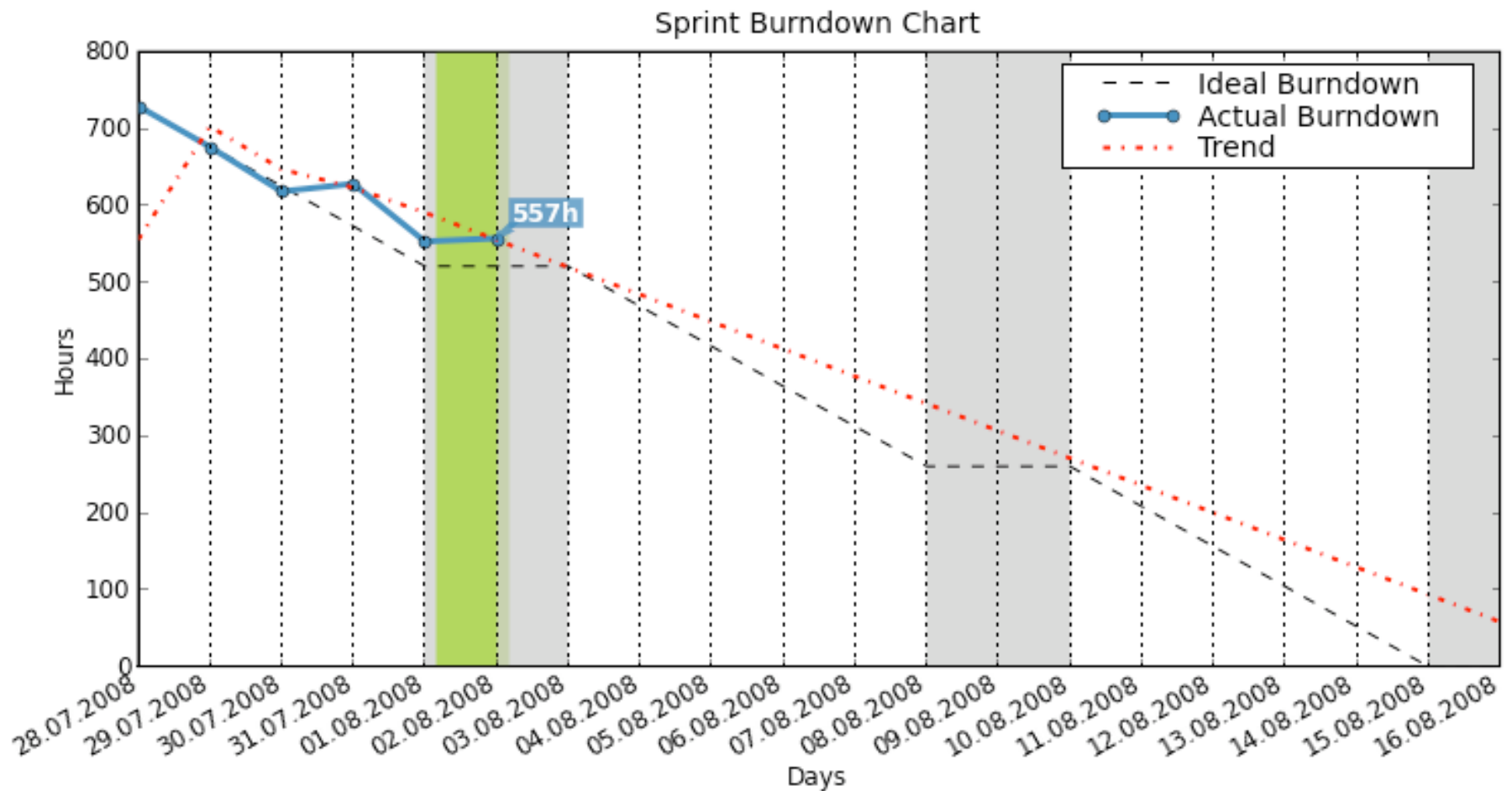
Artefakte

- Task
 - Tasks sind Arbeitspakete
 - vollständig
 - von jedem Developer bearbeitbar
 - innerhalb eines Bruchteils eines Sprints zu erfüllen
 - Der Aufwand eines Tasks wird in **Personenstunden** geschätzt.

Artefakte

- **Product Increment**
 - Am Ende eines Sprints erzeugte neue Version des Produkts.
 - Sollte direkt auslieferbar sein.
 - Enthält das Ergebnis aller im Sprint beendeter Tasks
- **Definition of Done**
 - Definition, die bestimmt, wann ein Task bzw. eine User-Story „fertig“ ist.
 - Wird vom Team erstellt.
 - Kann sich im Laufe des Projekts ändern.

Artefakte



Event: Sprint Planning Meeting

- Treffen des Teams **vor jedem Sprint**.
- Länge abhängig von Sprint-Länge (**2h pro Woche**).
- 1. **Was** ist das Ziel des nächsten Sprints?
 - **Product Owner** präsentiert wichtigste Backlog Einträge.
 - **Developer** diskutieren wie User Story zu erfüllen ist.
 - **Developer** entscheiden Arbeitsmenge für Sprint.
- 2. **Wie** wird das im Sprint zu erledigende umgesetzt?
 - **Developer** zerlegen ausgewählte Einträge in Tasks.
 - **Tasks** werden an die einzelnen Developer verteilt.

Event: Daily Scrum

- **Tägliches** Treffen.
- Begrenzt auf **15 Minuten**.
- **Developer** beantworten der Reihe nach folgende Fragen:
 - 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 **nicht** im Daily Scrum geklärt.

Event: Sprint Review

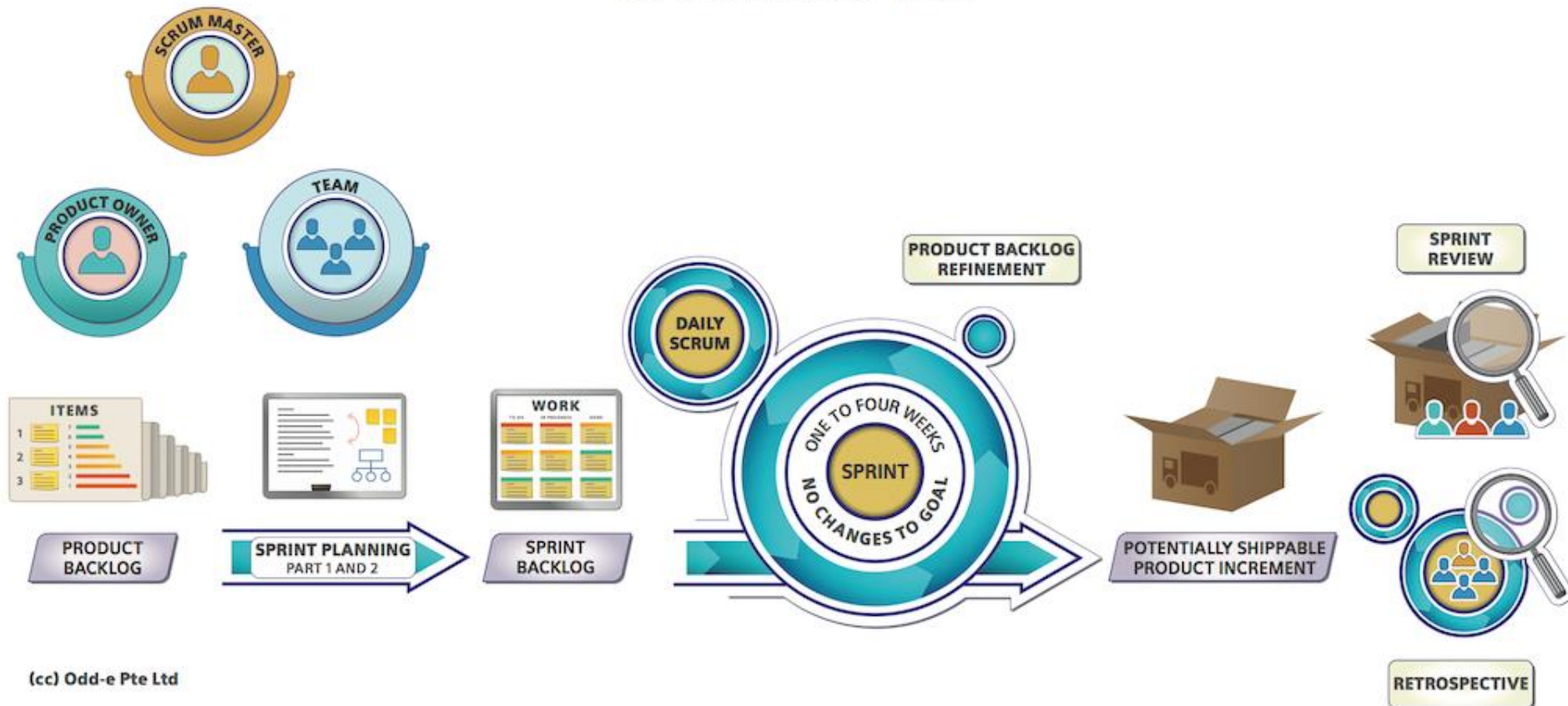
- Treffen des Teams **nach jedem Sprint**.
- Länge abhängig von Sprint-Länge (**1h pro Woche**).
- **Developer** demonstrieren neuen Product Increment.
 - **Product Owner** bestimmt, welche Tasks und User Stories fertig sind.
 - Unfertige User Stories und Tasks kehren in das Product Backlog zurück.
- **Product Owner** erklärt, wie gut die Aufwandsabschätzung war.

Event: Sprint Retrospective

- Treffen des Teams **nach dem Sprint Review**.
- Länge abhängig von Sprint-Länge (**45min/Woche**).
- **Scrum-Master** hilft, den Prozess zu verbessern:
 - Wie lief es im letzten Sprint hinsichtlich **Personen, Beziehungen, Prozessen, Werkzeugen**?
 - Ist die „**Definition of Done**“ ok?
 - Wie kann die **Arbeitsweise** des Teams verbessert werden?



SCRUM





SCRUM IM SOFTWAREPRAKTIKUM

Rollen

- **Tutor** ist Scrum-Master.
- **Tutor** ist Vertreter der Kunden.
- **Dozenten** sind Kunden.
- **Sie** sind Developer und Product Owner.



Artefakte in Gitea



Übersicht

Issues

Pull-Requests

Erkunden



sopra-WS18 / sopra-01

Beobachten beenden

1

Favorisieren

0

Fork

0

Code

Issues 8

Pull-Requests 0

Releases 0

Wiki

Aktivität

Einstellungen

Label

Meilensteine

Suche...

Suche

Neues Issue

4 offen

8 geschlossen

Label

Meilenstein

Zuständig

Typ

Sortieren

#16 Recherche zu 3D Objekten. help wanted priority: high question

vor 11 Minuten von vincent geöffnet

#13 Game Object Verwaltung planen. estimate: 2 priority: high

vor 14 Minuten von vincent geöffnet Sprint-01

#12 Drawable game object implementieren. estimate: 1 priority: high

vor 17 Minuten von vincent geöffnet Sprint-01

#11 Als Spieler möchte ich Spielobjekte auf der Karte sehen. priority: high user story

vor 18 Minuten von vincent geöffnet Sprint-01

Event: Gruppentreffen

- Gemeinsam mit Tutor.
- Länge max. 2h.
- **Aufteilung**
 1. Sprint Review (30min)
 2. Sprint Retrospective (15min)
 3. Sprint Planning (1h)
- Verbleibende Zeit wird mitgenommen.

Sprint Review

- **Studenten** demonstrieren neuen Product Increment.
 - Welche Aufgaben sind gemäß der **DoD** (nicht) erfüllt worden?
 - **Aufwandsabschätzung** diskutieren.
 - Nicht erreichte Aufgaben wandern ins Product Backlog zurück.
- Wird der nächste **Milestone** erreicht?

Sprint Retrospective

- Was lief im letzten Sprint **gut** oder **schlecht**?
 - Probleme mit **Teammitgliedern**.
 - Probleme mit dem **Prozess**.
 - Probleme mit der **Zeiteinteilung**.
 - Probleme mit **Tools**.
- **Schriftlichen Plan** machen, was verändert werden soll.
- Bei Bedarf die **Definition of Done** anpassen.

Sprint Planning

- Wie viel **Zeit** hat die Gruppe im nächsten Sprint?
- Product Backlog durchgehen:
 - **Wichtigste User Stories** für den nächsten Sprint **suchen**.
 - User Stories in Tasks **aufteilen**.
 - Benötigter Aufwand der entstandenen Tasks **abschätzen**.
 - **User Stories** und **Tasks** in Sprint Backlog **verschieben**.
 - **Wiederholen** bis verfügbare Zeit aufgebraucht ist.
- **Items** im Sprint Backlog an Teammitglieder verteilen.

Hinweise

- **Abhängigkeiten** zwischen den Aufgaben berücksichtigen.
- Wenn Aufgaben nicht geschafft werden:
 - **Rechtzeitig** an die **Gruppenliste** kommunizieren und im Task vermerken was schwerer/unerwartet war (Punkte).
- DoD bestimmt Ihre Einzelnote.
 - **Minimalanforderung**: Ticket geschlossen und durch Tutor „abgenommen“.
 - DoD steuert Qualität.

Hinweise

- **Gemeinsam arbeiten**
 - Termine finden, an denen man gemeinsam arbeiten kann (auch von zu Hause aus via Skype, IM, Gitea Kommentaren, ...).
 - Kurzes „Daily Scrum“ am Anfang solcher Treffen.
- **Aufwandsabschätzung** ernst nehmen.
- Organisation in **wiederkehrenden Aufgaben** strukturieren.
- Alle Probleme **früh** ansprechen.
 - In der Gruppe.
 - Direkt mit dem Tutor.
 - Direkt mit den Dozenten.

Was nun?

1. Gruppeneinteilung.
2. Pool Account besorgen?
3. E-Mail Adressen für Gruppenliste austauschen.
4. Regelmäßigen Termin für das Gruppentreffen ausmachen.

Ab morgen:

- Machen Sie die Hausaufgabe.
- Machen Sie sich mit den Werkzeugen und Techniken vertraut.
- Treffen Sie sich mit Ihrer Gruppe und dem Tutor.
- Entwickeln Sie eine Spielidee.
- Legen Sie Ihre erste Definition of Done im Gruppentreffen fest.



FRAGEN?