Softwarepraktikum

Sommersemester 2019

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

ORGANISATION

Team

- Tutoren
  Samuel Roth, Karsten Fix, Daniel Lux, Felix Karg, Thomas Leyh

- Dozenten
  Vincent Langenfeld, Daniel Dietsch

- Verantwortung
  Prof. Dr. A. Podelski

Organisation

- Teams mit ca. 7-8 Studenten,
  - Einteilung durch Fragebogen.
- 6 ECTS (180h) in 14 Wochen.
- 6 Abgaben, 4 Vorlesungen, 3 Präsentationen.
- Wöchentliches Gruppentreffen mit dem Tutor.

Organisation

- Termine
  - Betreuung
    Do. 14-18 Uhr im Pool.
  - Präsentationen & Vorlesungen
    Do. 14:00 – max. 18:00 hier.
  - Abgaben
    Samstags bis 23:59.

Dienste, Werkzeuge, Informationen

- Wiki
- Informationen

  Wiki, Gruppenmitglieder, Tutoren, Poolbetreuung
- Primäre Dienste

  Gitea, Git, Mailinglisten (sopra-crew@..., sopraXX@...), Poolrechner
- Sekundäre Dienste

  Sonar, (Doxygen)
- Werkzeuge

  C#, F#, .NET, MonoGame 3.7, Visual Studio Community 19, ReSharper

Zulassung

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

Benotung

- Es gibt eine Abschlussnote.
- Die Abschlussnote setzt sich aus zwei Teilen zusammen:
  - 50% Endprodukt
  - 50% Aufgabenorientierte Leistung
- Ist Endprodukt oder aufgabenorientierte Leistung 5.0 (nicht bestanden), dann ist die Endnote 5.0 (nicht bestanden).
- Details im Wiki.

Endprodukt

- Entspricht das Endprodukt den Anforderungen (d.h. sind alle Inhalte vorhanden, ist die Usability gewährleistet, ist alles konsistent)?
- Ist das Produkt fehlerfrei (d.h. finden wir bei der Abnahme keine Fehler oder Abstürze)?
- Sind die Softwarequalitätsanforderungen erfüllt (d.h. gibt es keine Compiler/ReSharper Warnungen oder Fehler)?

Aufgabenorientierte Leistung

- Wöchentliche Einzelleistung pro Teilnehmer:
  - Wurde die zugeteilte Arbeit *erfolgreich* erledigt?
  - Max. 5 Punkte pro Woche ab nächster Woche.
- 70 Punkte für Einzelleistung

Lernziele

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

Simulation eines Softwareentwicklungsprozesses

THEMA: COMPUTERSPIEL

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 SW funktionale Anforderungen erfüllt oder auf die die SW entwickelt wird.

Funktionale Anforderungen

- 2D oder 3D Grafik (kein ASCII).
- Min. 2 Spieler, min. einer davon „menschlich“.
- Indirekte Steuerung (Point & Click).
- Potenziell zu jedem Zeitpunkt speichern/laden, muss aber nicht zwangsläufig vom Spieler gesteuert sein.
- Pausefunktion.
- Eigenes Menü (komplett mit der Maus steuerbar, außer Texteingaben).

Funktionale Anforderungen

- Arten von Spielobjekten:
  a) Min. 5 Kontrollierbare.
  b) Min. 5 Auswählbare.
  c) Min. 5 Nicht-Kontrollierbare, davon min. 3 Kollidierende.
  d) Min. 3 Kontrollierbare, Kollidierende und Bewegliche.
- Min. 1000 gleichzeitig aktive Spielobjekte der Art (d) möglich (Tech-Demo).

Funktionale Anforderungen

- Arten von Aktionen:
  - Min. 10 verschiedene Aktionen (inkl. Laufen, Fähigkeiten, usw.).
  - 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“).

Funktionale Anforderungen

- Soundeffekte und Musik.
- Min. 5 verschiedene Statistiken.
- Achievements.
- Echtzeit:
  - Spieler müssen Aktionen durchführen, während ihre Gegner ebenfalls gleichzeitig Aktionen durchführen und zu jedem Zeitpunkt reagieren können.

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.
- Die im Spiel enthaltenen Texte müssen frei von Rechtschreib-, Grammatik- und Umlautfehlern sein.
- Richtlinen zur Bedienbarkeit von Computerspielen beachten (Wiki-Artikel „Usability beim Spieldesign“).

Randbedingungen

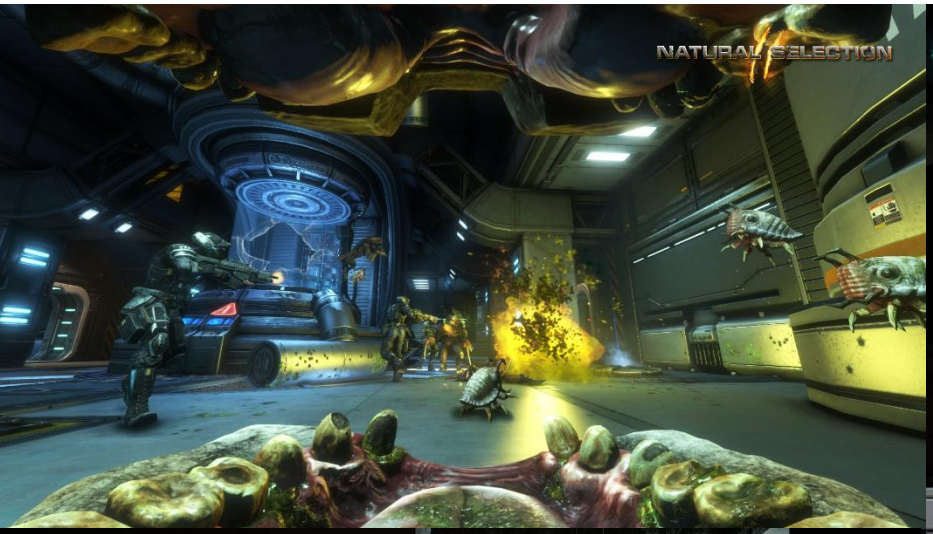
- Programmiersprache C# und/oder F# mit .NET.
- MonoGame 3.7.
- Auf Windows 7 x86/x64 lauffähig.
- Visual Studio Community 2019.
- Keine Warnings oder Errors vom Compiler oder ReSharper (wöchentlich), keine Buildfehler.

Beispiele

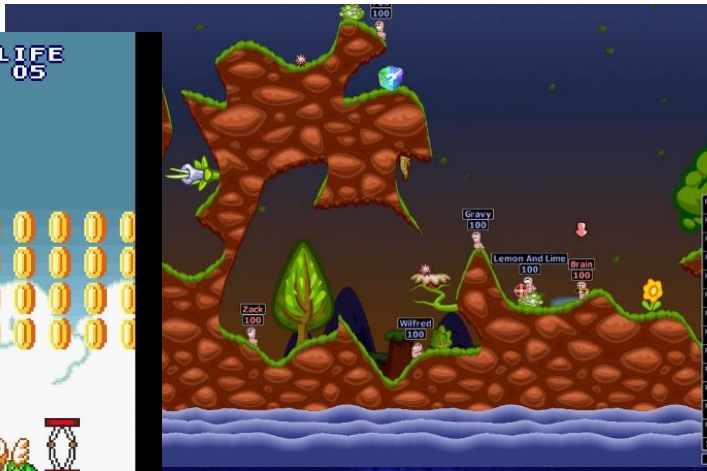

JOBS
CONSUMABLES
VITALS
REPORTS
RESEARCH
OVERLAYS
Nisbet
Meep
Sweep
Mop
Deconstruct
Cancel


Dwarf Fortress

Wackelkandidaten



Gegenbeispiele



ABLAUF

Woche	Organisation	Entwurf	MS 01	MS 02	MS 03	MS 04	MS 05	Was?	Wann und Wo?
0	✓	✗	✗	✗	✗	✗	✗	• Zur Vorlesung anmelden. • Vorlesung "Organisation und Prozess" (Einführungsveranstaltung) • Gruppeneinteilug abwarten	• Vorlesung: 25.04., 14:00 - max. 18:00, 082-00-006 • Fragebogen: Bis 25.04. 23:59 • Gruppen ab 28.04. in Gitea
1	✓	✓	✗	✗	✗	✗	✗	• Vorlesung "Game Design Document (GDD)" • Abgabe Hausaufgabe	• Vorlesung: 02.05., 14:00 - max. 18:00, 082-00-006 • Abgabe: 04.05. bis 23:59
2	✗	✓	✓	✗	✗	✗	✗	• Vorlesung "Grundlagen Softwarearchitektur" • Wiederkehrende Aufgaben: Product Owner • Abgabe GDD (beta)	• Vorlesung: 09.05., 14:00 - max. 18:00, 082-00-006 • Abgabe: 11.05. bis 23:59
3	✗	✓	✓	✗	✗	✗	✗	• Vorlesung "Architektur von Videospielen" • Wiederkehrende Aufgaben: Architektur und Coaching • Wiederkehrende Aufgaben: Qualitätssicherung und Clean-Code	• Vorlesung: 16.05., 14:00 - max. 18:00, 082-00-006
4	✗	✓	✓	✓	✗	✗	✗	• MS01 erreicht (Spielobjekt in der Welt bewegbar, interaktive Kamera, Karte laden/speichern, Soundausgabe) • Präsentation des Spielekonzepts • Abgabe Architektur (beta)	• Präsentation: 23.05., 14:00 - max. 18:00, 082-00-006 • Abgabe: 25.05. bis 23:59
5	✗	✓	✗	✓	✗	✗	✗		
6	✗	✓	✗	✓	✓	✗	✗		
7	✗	✓	✗	✓	✓	✗	✗	• MS02 erreicht (Mehrere Spielobjekte bewegen, Interaktionen zwischen Spielobjekten, Pathfinding, Screen-Management, Menü, HUD, Musik)	
8	✗	✓	✗	✗	✓	✗	✗	• Präsentation Programm (beta) **TERMIN WEGEN FEIERTAG VERSCHOBEN** • Abgabe Programm (beta)	• Präsentation: 21.06., 14:00 - max. 18:00, 082-00-006 • Abgabe: 22.06. bis 23:59
9	✗	✓	✗	✗	✓	✓	✗	• Abgabe GDD (final) • MS03 erreicht (KI, primäre Interaktionen vorhanden, Sieg-/Niederlagebedingungen, vollständiges Pathfinding, Inhalte, Grafik/Soundeffekte)	• Abgabe: 29.06. bis 23:59
10	✗	✗	✗	✗	✗	✓	✓		
11	✗	✗	✗	✗	✗	✓	✓		
12	✗	✗	✗	✗	✗	✓	✓	• MS04 erreicht (finale Version vorhanden) • Abgabe Architektur (final)	• Abgabe: 20.07. bis 23:59
13	✗	✗	✗	✗	✗	✗	✓	• MS05 erreicht (Fehlerbehebung & Balancing) • Präsentation Programm (final) • Abgabe Programm (final)	• Präsentation: 25.07., 14:00 - max. 18:00, 082-00-006 • Abgabe: 27.07. bis 23:59

Fragebogen

- Link auf dem Wiki.
- Zweck:
  - Gruppen so gerecht und sinnvoll wie möglich einteilen.
  - Dienste vorbereiten.
- Ausfüllen bis heute Abend, 23:59 Uhr!
- Wird nach der Vorlesung freigeschaltet.

Hausaufgabe

- Genaue Beschreibung auf dem Wiki.
- Ungefähr:
  - Werkzeuge installieren, Dienste testen.
  - Gitea kennenlernen.
  - Artikel im Wiki lesen.
    - Clean Code, Dokumentation, Usability, Git und Gitea
  - MonoGame Programm schreiben.

Game Design Document

- GDD beschreibt die wesentlichen Merkmale des Spiels für den Auftraggeber.
  - Ähnlich zu Lastenheft.
- Details in GDD-Vorlesung und im Wiki.

Erster Entwurf der SW-Architektur

- SW-Architektur beschreibt die Strukturen eines SW-Systems durch Bausteine und deren Beziehungen und Interaktionen untereinander.
- Bei uns reduziert auf Komponentendiagramm und Klassendiagramm in UML.
- Details in Vorlesungen ab Woche 3.

Umsetzung

- Grob gegliedert in 5 Milestones (MS)
  - Unsere MS beschreiben einen Referenzablauf.
  - Behalten Sie den Termin, definieren Sie sich passende Inhalte.

SCRUM ALS VORGEHENSMODELL

Scrum

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

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

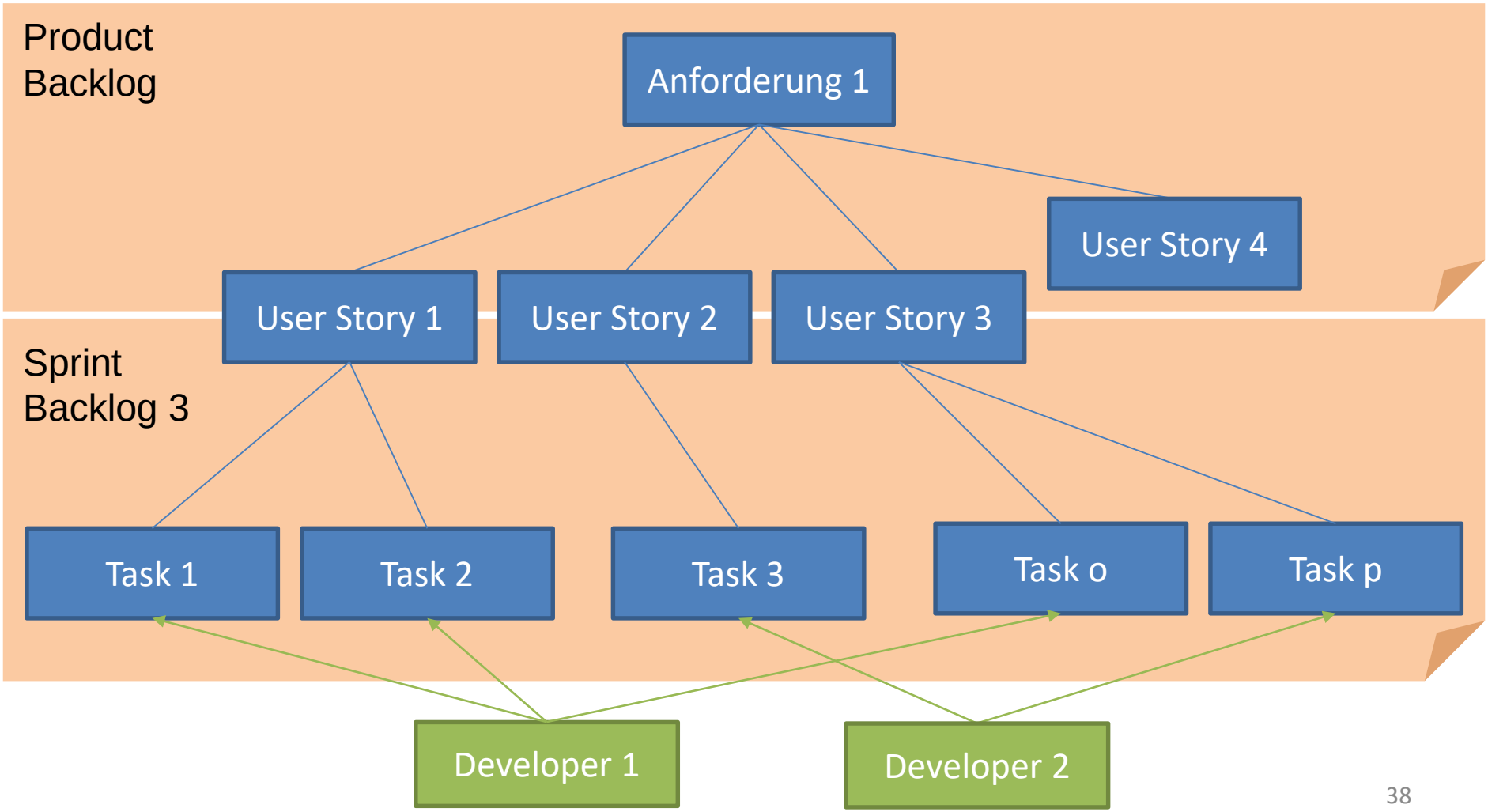
- Product Backlog
  - Liste von Anforderungen mit abgeleiteten User Stories.
  - Items nach Wichtigkeit geordnet.
- User Stories
  - Beschreibung aus Sicht des Benutzers.
  - Möglichst kurz.
  - Der Aufwand einer User Story wird in Story Points geschätzt.
  - Oft nach Vorlage, z.B.
    - „Als <Rolle> möchte ich <Ziel/Wunsch>, um <Nutzen>“

Artefakte

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

- Task
  - Teilaufgabe einer User Story.
  - Sind innerhalb eines Sprints erfüllbar.
  - Der Aufwand eines Tasks wird in Personenstunden geschätzt.

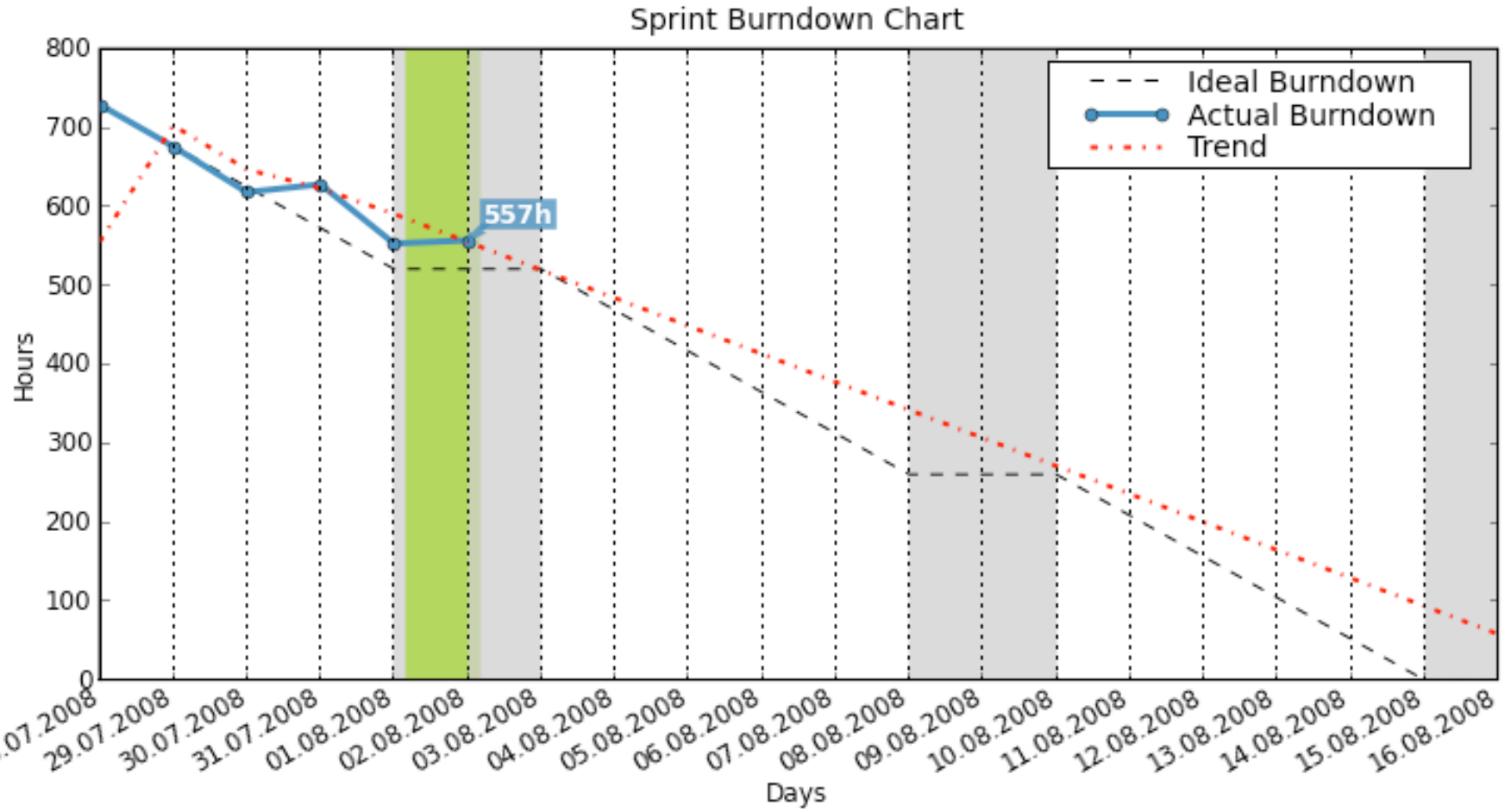
Artefakte


Product
Backlog
Anforderung 1
User Story 1
User Story 2
User Story 3
User Story 4
Sprint
Backlog 3
Task 1
Task 2
Task 3
Task o
Task p
Developer 1
Developer 2

Artefakte

- Product Increment
  - Neue Produktversion am Ende des Sprints.
  - Direkt auslieferbar.
- Definition of Done (DoD)
  - Definition, die bestimmt, wann ein Task bzw. eine User-Story „fertig“ ist.
  - Wird vom Team erstellt.
  - Kann sich im Laufe des Projekts ändern.

Artefakte


Sprint Burndown Chart
Ideal Burndown
Actual Burndown
Trend
557h
Hours
Days
800
700
600
500
400
300
200
100
0
28.07.2008
29.07.2008
30.07.2008
31.07.2008
01.08.2008
02.08.2008
03.08.2008
04.08.2008
05.08.2008
06.08.2008
07.08.2008
08.08.2008
09.08.2008
10.08.2008
11.08.2008
12.08.2008
13.08.2008
14.08.2008
15.08.2008
16.08.2008

Event: Sprint Planning Meeting

- Treffen des Teams vor jedem Sprint.
- Länge abhängig von Sprint-Länge (2h pro Woche).

1. Was wird im nächsten Sprint getan?
   - Product Owner präsentiert Product Backlog Einträge.
   - Developer wählen aus, was sie im nächsten Sprint umsetzen können.
2. Wie wird das im Sprint zu erledigende umgesetzt?
   - Developer zerlegen ausgewählte Einträge in Tasks.
   - Developer erstellen neuen Software-Entwurf.

Event: Daily Scrum

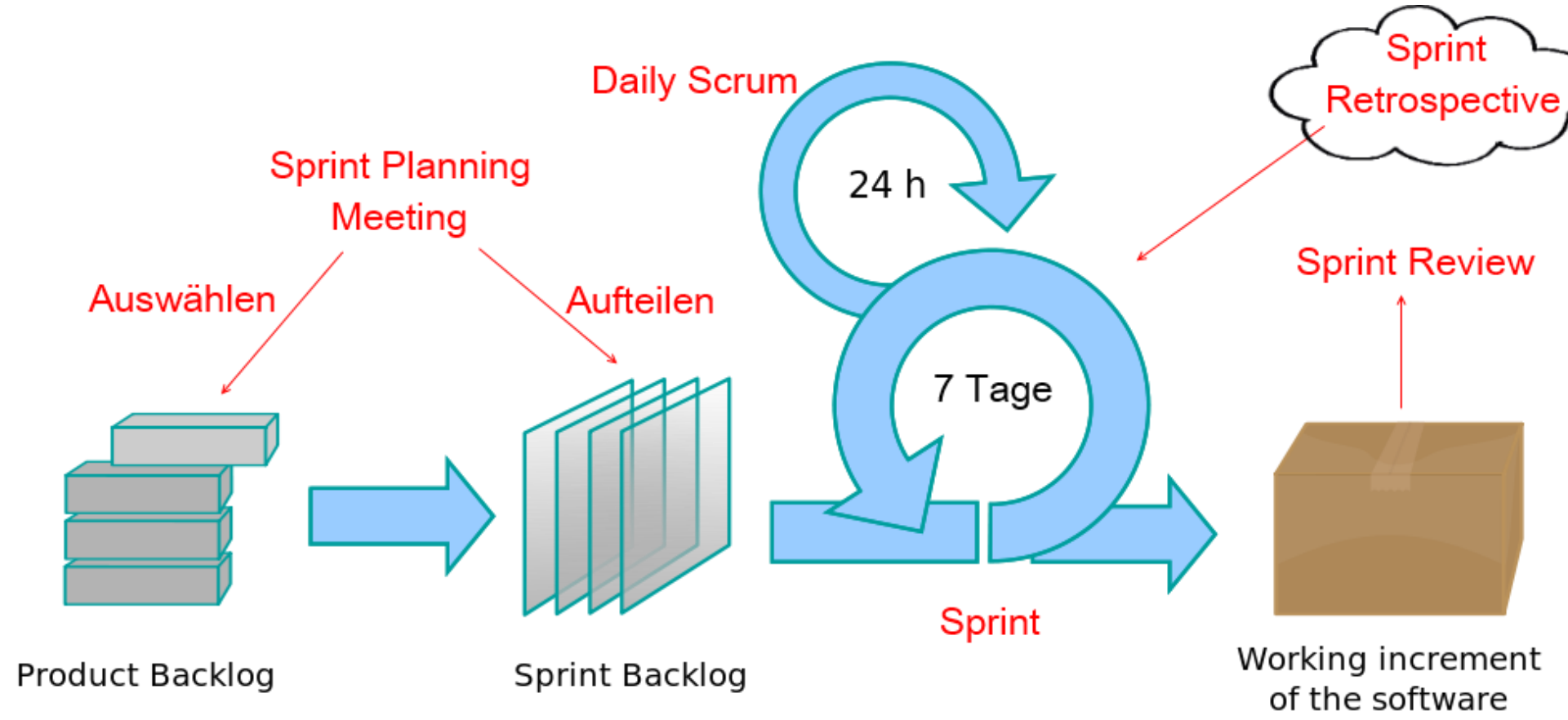
- Tägliches Treffen.
- Begrenzt auf 15 Minuten.
- Developer beantworten der Reihe nach folgende Fragen:
  - Was habe ich seit dem letzten Treffen getan?
  - Was plane ich bis zum nächsten Treffen zu tun?
  - Welche Probleme hatte ich und wo benötige ich Hilfe?
- Fragen werden nicht im Daily Scrum geklärt.

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 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?


Daily Scrum
Sprint Retrospective
Sprint Planning Meeting
24 h
Sprint Review
Auswählen
Aufteilen
7 Tage
Sprint
Product Backlog
Sprint Backlog
Working increment of the software

SCRUM IM SOFTWAREPRAKTIKUM

Rollen

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

Artefakte im Sopra

Code | Issues 5 | Pull-Requests 0 | Releases 0 | Wiki | Aktivität | Einstellungen

Label | Meilensteine | Suche... | Suche | Neues Issue

5 offen | 5 geschlossen | Label | Meilenstein | Zuständig | Typ | Sortieren

#10 **Jeder Student soll das in Aufgabe 5 beschriebene Programm schreiben, um seine Entwicklungswerkzeuge zu testen.** estimate: 5 | high | user story
vor 6 Minuten von vincent geöffnet | Hausaufgabe

#9 **Student Vincent soll die Dokumentation Texte lesen, um seinen Code sinvoll dokumentieren zu können.** estimate: 0.5 | high | user story
vor 7 Minuten von vincent geöffnet | Hausaufgabe

#8 **Student Vincent soll den Usability Artikel lesen, um von vornherein Usabilityprobleme zu vermeiden.** estimate: 2 | high | user story
vor 7 Minuten von vincent geöffnet | Hausaufgabe

#7 **Student Vincent soll die Clean Code Development Texte lesen, um besseren Code schreiben zu können.** estimate: 2 | high | user story
vor 7 Minuten von vincent geöffnet | Hausaufgabe

#6 **Student Vincent soll Scrum, Gitea und Git verstehen um effizient arbeiten zu können.** estimate: 1 | high | user story
vor 8 Minuten von vincent geöffnet | Hausaufgabe

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

- Neues Product Increment vorführen.
  - Bereiten Sie vor, was sie zeigen wollen.
- Welche Aufgaben sind gemäß DoD (nicht) erledigt worden?
- Aufwandsabschätzung diskutieren.
- Wird der nächste Milestone erreicht?

Sprint Retrospective

- Was lief im letzten Sprint gut oder schlecht?
  - Probleme mit Teammitgliedern.
  - Probleme mit dem Prozess.
  - Probleme in der Zeiteinteilung.
  - Probleme mit Tools.
- Schriftlichen festhalten, was verändert werden soll.
- Bei Bedarf DoD anpassen.

Sprint Planning

- Wie viel Zeit hat die Gruppe im nächsten Sprint?
- Product Backlog durchgehen:
  - Das wichtigste User Story für den nächsten Sprint suchen.
  - Benötigter Aufwand der User Story abschätzen.
  - User Story ins Sprint Backlog verschieben.
  - Wiederholen bis verfügbare Zeit aufgebraucht ist.
- Items im Sprint Backlog an Teammitglieder verteilen.

Recurring Tasks

- Ab Woche 2: Product Owner
  - Pflegen des Product Backlog.
  - User Stories nach Wichtigkeit für die aktuelle Entwicklung ordnen.
  - Gruppentreffen vorbereiten
    - Was ist nach DoD fertig geworden?
    - Wie waren die Aufwandsabschätzungen?
    - Product Increment präsentationsfertig machen.

Recurring Tasks

- Ab Woche 3: Architektur
  - Schnittstellen definieren.
  - Architekturbeschreibungen pflegen.
  - Einhaltung der Architektur sicherstellen.

- Ab Woche 3: Qualitätssicherung
  - Code auf Clean-Code Richtlinien prüfen.
  - Code Reviews vorbereiten.
  - ReSharper Konformität herstellen.

Hinweise

- Abhängigkeiten zwischen den Aufgaben berücksichtigen.
- Wenn Aufgaben nicht geschafft werden:
  - Rechtzeitig an Gruppenliste kommunizieren (Punkte).
- DoD bestimmt die Punkte für Ihre Einzelleistung.
  - Minimalanforderung: Alles im Git, Ticket geschlossen und vom Tutor „abgenommen“.
  - DoD steuert Qualität.

Hinweise

- Gemeinsam arbeiten
  - Termine finden, an dem man gemeinsam arbeiten kann (auch von zu Hause aus via Skype, IM, Remote-Desktop, …).
  - Kurzes „Daily Scrum“ am Anfang solcher Treffen.
- Aufwandsabschätzung ernst nehmen.
- Organisation in wiederkehrenden Aufgaben organisieren.
  - Tickets dafür anlegen und Arbeitszeit schätzen und aufschreiben.
- Alle Probleme früh ansprechen.
  - In der Gruppe.
  - Direkt mit dem Tutor.
  - Direkt mit den Dozenten.
- Rechtzeitig zur Prüfung anmelden, jetzt zur Veranstaltung anmelden.

Was nun?

1. Fragebogen ausfüllen bis heute Abend 23:59 Uhr!
2. Auf Gruppeneinteilung warten (bis 27.04.).
3. Alleine Hausaufgabe machen bis Sa., 05.05., 23:59 Uhr.

Nach der Gruppeneinteilung:

- Regelmäßigen Termin für Gruppentreffen ausmachen.
- Machen Sie sich mit den Werkzeugen und Techniken vertraut.
- Treffen Sie sich mit Ihrer Gruppe und dem Tutor.
- Entwickeln Sie eine gute Spielidee.
- Legen Sie Ihre erste Definition of Done fest.

FRAGEN?