Softwarepraktikum

Sommersemester 2016

- Organisation
- Thema und Anforderungen
- Ablauf
- Scrum als Vorgehensmodell
- Scrum im Softwarepraktikum
- Vorlesung: „GDD“

ORGANISATION

Team

- Tutoren
  Ivo Enke, Malte Höck, Maximilian Rohland
- Dozenten
  Marius Greitschus, Vincent Langenfeld
- Verantwortung
  Prof. Dr. A. Podelski

Organisation

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

Organisation

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

Dienste, Werkzeuge, Informationen

- Wiki
- Informationen

  Wiki, Gruppenmitglieder, IRC, Tutoren, Poolbetreuung
- Primäre Dienste

  SVN, Trac, Mailinglisten (sopra-crew@..., sopraXX@...), Poolrechner
- Sekundäre Dienste

  Sonar, StatSVN, Doxygen
- Werkzeuge

  C#, F#, .NET 4.6.1, MonoGame 3.5 / XNA 4.0, Visual Studio 2015, ReSharper 2016.1

Zulassung

- Kontinuierliche Mitarbeit
  - Belegt durch hinreichend viel messbare Aktivität (SVN, Trac).
  - Verbrauchte Zeit und geschätzte Restzeit müssen im Trac 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. finden wir bei der Abnahme 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

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

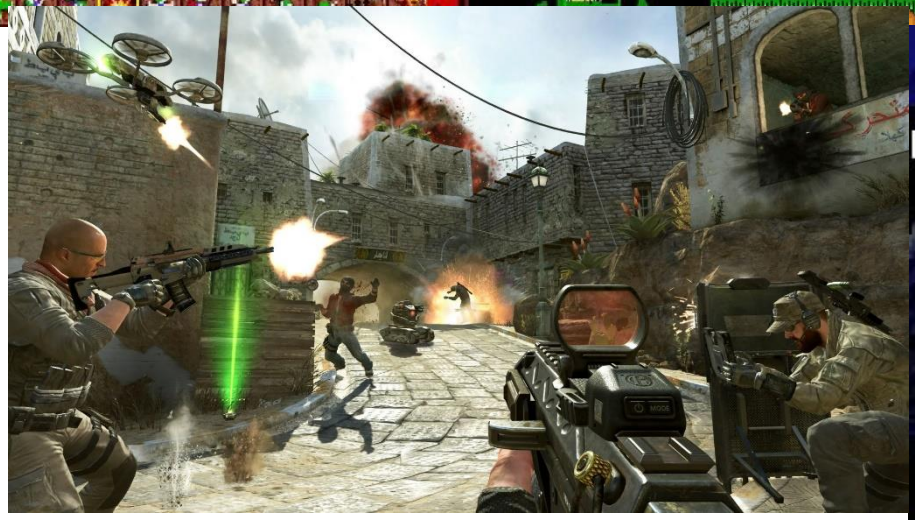
- Entwickeln Sie ein gutes Produkt.
- Qualität der Grafik ist nicht relevant.
- Grafiken sollen in sich stimmig sein.
- Akustische Effekte sollen in sich stimmig sein.
- Die im Spiel enthaltenen Texte müssen frei von Rechtschreib-, Grammatik- und Umlautfehlern sein.
- Richtlinen zur Bedienbarkeit von Computerspielen beachten (Wiki-Artikel „Usability beim Spieldesign“).

Randbedingungen

- Programmiersprache C# und/oder F# mit .NET 4.6.1.
- MonoGame 3.5.
- Alternativ: XNA 4.0 (XNA Game Studio 4.0 (Windows Phone SDK 7.1): 28. September 2011).
- Auf Windows 7 x86/x64 lauffähig.
- Visual Studio 2015.
- Keine Warnings oder Errors vom Compiler oder ReSharper (wöchentlich), keine Buildfehler.

Beispiele

Gegenbeispiele



ABLAUF

Woche	Organisation	Entwurf	MS 01	MS 02	MS 03	MS 04	MS 05	Was?	Wann und Wo?
0	✓	✗	✗	✗	✗	✗	✗	• Vorlesung "Organisation und Prozess" (Einführungsveranstaltung) • Vorlesung "Game Design Document (GDD)" • Gruppeneinteilung abwarten	• Vorlesung: 20.04., 14:00 - max. 18:00, 082-00-006 • Fragebogen: Bis 20.04. 23:59 • Gruppeneinteilung: Am 23.04. online
1	✓	✓	✗	✗	✗	✗	✗	• Vorlesung "Grundlagen Softwarearchitektur" • Abgabe Hausaufgabe	• Vorlesung: 27.04., 14:00 - max. 18:00, 082-00-006 • Abgabe: 30.04. bis 23:59
2	✗	✓	✓	✗	✗	✗	✗	• Vorlesung "Architektur von Videospielen" • Abgabe GDD (beta)	• Vorlesung: 04.05., 14:00 - max. 18:00, 082-00-006 • Abgabe: 07.05. bis 23:59
3	✗	✓	✓	✗	✗	✗	✗		
4	✗	✓	✓	✓	✗	✗	✗	• MS01 erreicht (Spielobjekt in der Welt bewegbar, interaktive Kamera, Karte laden/speichern, Soundausgabe)	
5	✗	✓	✗	✓	✗	✗	✗	• Präsentation des aktuellen Stands • Abgabe Architektur (beta)	• Präsentation: 25.05., 14:00 - max. 18:00, 082-00-010/14 • Abgabe: 28.05. bis 23:59
6	✗	✓	✗	✓	✓	✗	✗		
7	✗	✓	✗	✓	✓	✗	✗	• MS02 erreicht (Mehrere Spielobjekte bewegen, Interaktionen zwischen Spielobjekten, Pathfinding, Screen-Management, Menü, HUD, Musik)	
8	✗	✓	✗	✗	✓	✗	✗	• Präsentation Programm (beta) • Abgabe Programm (beta)	• Präsentation: 15.06., 14:00 - max. 18:00, 082-00-006 • Abgabe: 18.06. bis 23:59
9	✗	✓	✗	✗	✓	✓	✗	• Abgabe GDD (final) • MS03 erreicht (KI, primäre Interaktionen vorhanden, Sieg-/Niederlagebedingungen, vollständiges Pathfinding, Inhalte, Grafik/Soundeffekte)	• Abgabe: 25.06. bis 23:59
10	✗	✗	✗	✗	✗	✓	✓		
11	✗	✗	✗	✗	✗	✓	✓		
12	✗	✗	✗	✗	✗	✓	✓	• MS04 erreicht (finale Version vorhanden) • Abgabe Architektur (final)	• Abgabe: 16.07. bis 23:59
13	✗	✗	✗	✗	✗	✗	✓	• MS05 erreicht (Fehlerbehebung & Balancing) • Präsentation Programm (final) • Abgabe Programm (final)	• Präsentation: 20.07., 14:00 - max. 18:00, 082-00-006 • Abgabe: 23.07. 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.
  - Trac kennenlernen.
  - Artikel im Wiki lesen.
    - Clean Code, Dokumentation, Usability, Trac und SVN
  - MonoGame/XNA Programm schreiben.

Game Design Document

- GDD beschreibt die wesentlichen Merkmale des Spiels für den Auftraggeber.
  - Ähnlich zu Lastenheft.
- Details gleich 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 1.

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 Rollen, Events, Artefakten, Regeln.

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 Requirements mit abgeleiteten User Stories.
  - Ist geordnet nach Entwicklungsreife.

- Requirements
  - In Scrum sehr grob beschrieben, z.B:
    1. „Der Spieler soll Einheiten gruppieren können.“
    2. „Das Spiel soll ein Hauptmenü haben.“

Artefakte

- User Stories
  - Beschreibung aus Sicht des Benutzers.
  - Möglichst kurz.
  - Nur ein Satz.
  - 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.
  - Der Aufwand einer User Story wird in Story Points geschätzt.
  - Für jeden Sprint wird ein neues Sprint Backlog aus dem Product Backlog abgeleitet.

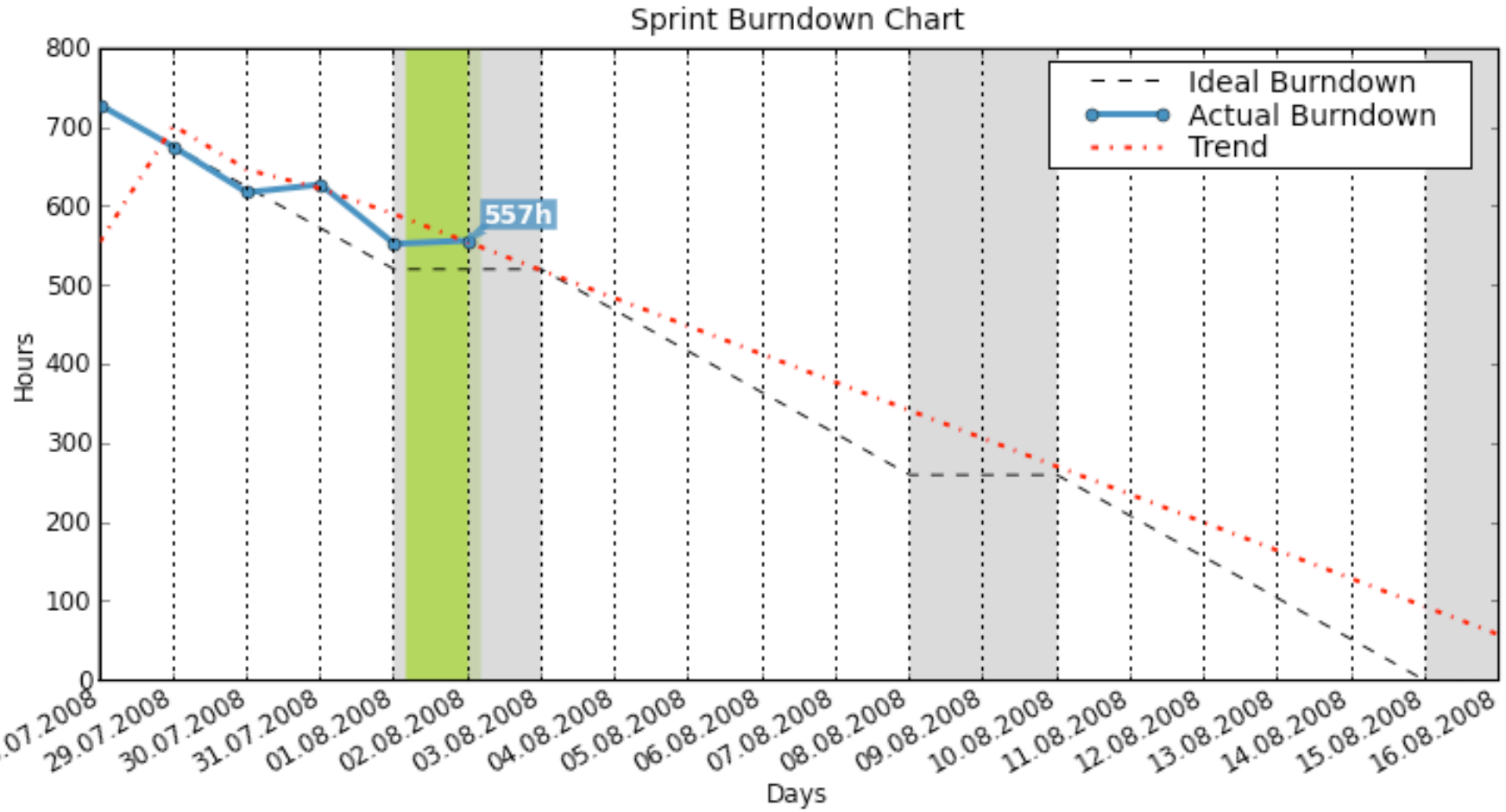
Artefakte

- Task
  - Tasks sind Arbeitspakete, die
    - vollständig,
    - von jedem Developer bearbeitbar und
    - innerhalb eines Sprints erfüllbar sind.
  - Der Aufwand eines Tasks wird in Personenstunden geschätzt.

Artefakte

- Product Increment
  - Am Ende eines Sprints erzeugte neue Version des Produkts.
  - Sollte direkt auslieferbar sein.
- 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 kleinere Teile.
   - Developer erstellen neuen Software-Entwurf.

Event: Daily Scrum

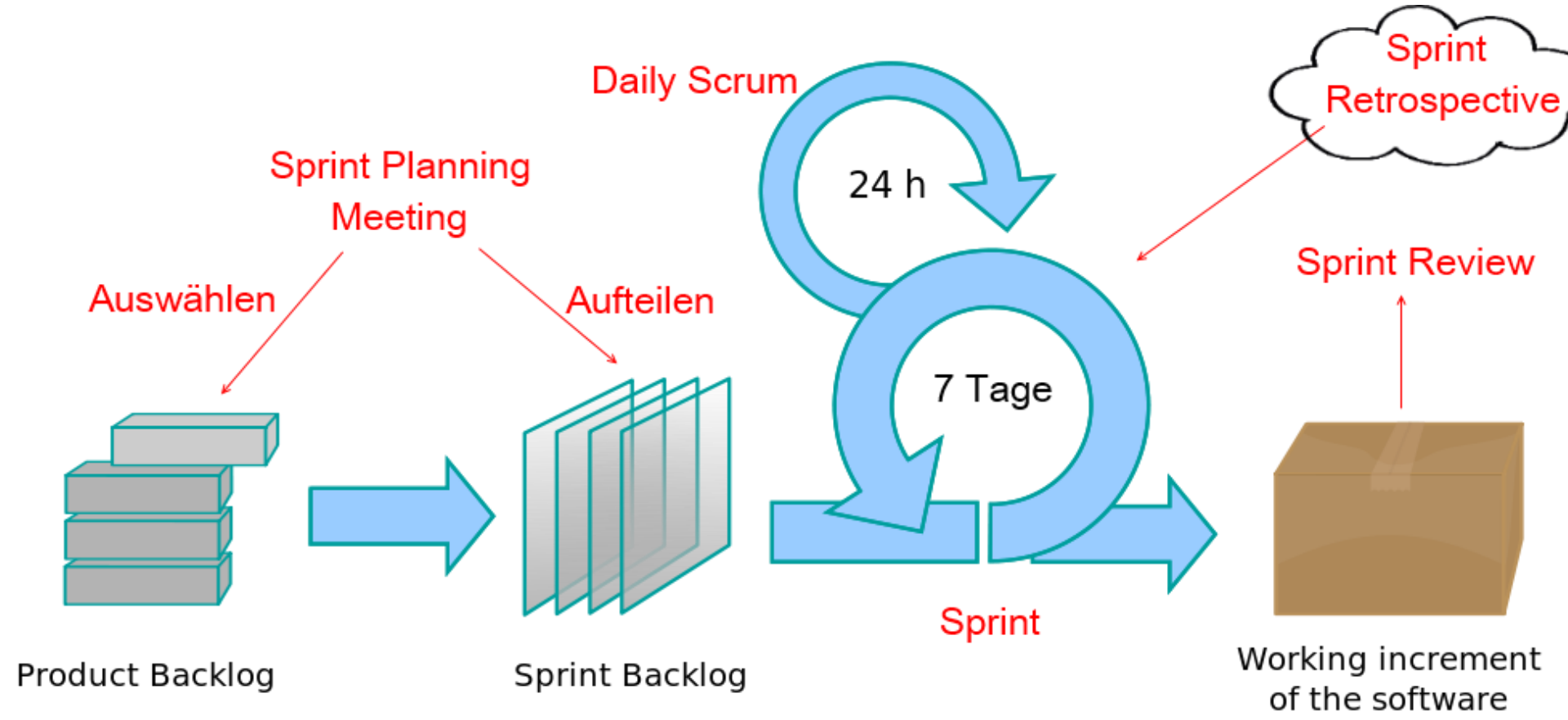
- Tägliches Treffen.
- Begrenzt auf 15 Minuten.
- Developer beantworten der Reihe nach folgende Fragen:
  - Was habe ich seit dem letzten Treffen getan?
  - Was plane ich bis zum nächsten Treffen zu tun?
  - Welche Probleme hatte ich und wo benötige ich Hilfe?
- Fragen werden nicht im Daily Scrum geklärt.

Event: Sprint Review

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

- Was ist mit dem Product Owner?

Artefakte im Trac

Sprint Backlog for Sprint Hausaufgabe

ID	Summary	Remaining Time	Owner	Resources	Spent Time
24	Hausaufgaben machen				
30	Student greitsch soll das Trac bedienen können, damit er in Zukunft produktiver arbeitet	7	greitsch		1
33	Scrum verstehen	5	greitsch		
34	Trac verstehen	2	greitsch		1
31	Student greitsch soll die Texte verstehen, damit er besser Spiele entwickeln kann	8	greitsch		
35	"Clean Code Development" Artikel lesen	1	greitsch		
36	"Dokumentation" Artikel lesen	2	greitsch		
37	"Usability-Prinzipien beim Spieldesign" Artikel lesen	3	greitsch		
38	"Trac und SVN" Artikel lesen	2	greitsch		
32	Student greitsch soll ein XNA Programm schreiben, damit er früh merkt, ob irgendetwas nicht funkti...	2	greitsch		
28	Programm schreiben	2	greitsch		
39	Student dzienian soll das Trac bedienen können, damit er in Zukunft produktiver arbeitet.	0	dzienian		
43	Trac verstehen	0	dzienian		
49	Scrum verstehen	0	dzienian		
40	Student dzienian soll die Texte verstehen, damit er besser Spiele entwickeln kann.	1.5	dzienian		
44	Clean Code Development' Artikel lesen	0.5	dzienian		
45	Dokumentation' Artikel lesen	0.5	dzienian		
46	Usability-Prinzipien beim Spieldesign' Artikel lesen	0	dzienian		
47	'Trac und SVN' Artikel lesen	0.5	dzienian		
41	Student dzienian soll ein XNA Programm schreiben, damit er früh merkt, ob irgendetwas nicht funkti...	0.5	dzienian		
48	Programm schreiben	0.5	dzienian		
21	Totals	19			1

Event: Gruppentreffen

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

Sprint Review

- Neues Product Increment vorführen.
- Welche Aufgaben sind gemäß DoD (nicht) erledigt worden?
- Aufwandsabschätzung diskutieren.
- Wird der nächste Milestone erreicht?

Sprint Planning

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

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 Plan machen, was verändert werden soll.
- Bei Bedarf DoD anpassen.

Hinweise

- Abhängigkeiten zwischen den Aufgaben berücksichtigen.
- Wenn Aufgaben nicht geschafft werden:
  - Rechtzeitig an Gruppenliste kommunizieren (Punkte).
- DoD bestimmt Ihre Einzelnote.
  - Minimalanforderung: 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, IRC,…).
  - Kurzes „Daily Scrum“ am Anfang solcher Treffen.
- Aufwandsabschätzung ernst nehmen.
- Organisation in wiederkehrenden Aufgaben organisieren.
- Alle Probleme früh ansprechen.
  - In der Gruppe.
  - Direkt mit dem Tutor.
  - Direkt mit den Dozenten.

Was nun?

1. Fragebogen ausfüllen bis heute Abend 23:59 Uhr!
2. Auf Gruppeneinteilung warten (bis 24.4.).
3. Alleine Hausaufgabe machen bis Sa., 30.4., 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 Spielidee.
- Legen Sie Ihre erste Definition of Done fest.

FRAGEN?