



# Softwarepraktikum

SS 2013



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



# ORGANISATION



# Team

- **Tutoren**  
Jeremi Dzienian, Heinke Hihn, Jonas Schlagenhauf
- **Dozenten**  
Daniel Dietsch, Marius Greitschus
- **Verantwortung**  
Prof. Dr. A. Podelski



# Organisation

- 6 ECTS (180h) in 14 Wochen.
- Teams mit ca. 8 Studenten, Einteilung durch Fragebogen.
- 6 Abgaben, 4 Vorlesungen, 3 Präsentationen.
- Wöchentliches Gruppentreffen mit dem Tutor.
  - Dabei Feedback zum Projektfortschritt (rot, gelb, grün).



# Organisation

- Termine
  - Betreuung  
Mi. 14-18 Uhr im Pool.
  - Präsentationen & Vorlesungen  
Mi. 15:30 – max. 19:00 hier.
  - Abgaben  
Samstags bis 23:59.



# Dienste, Werkzeuge, Informationen

- **Wiki**
- **Informationen**  
Wiki, Forum, Gruppenmitglieder, IRC, Tutoren, Poolbetreuung
- **Primäre Dienste**  
SVN, Trac, Mailinglisten (sopra-crew@..., sopraXX@...), Poolrechner
- **Sekundäre Dienste**  
Sonar, StatSVN, Doxygen
- **Werkzeuge**  
C#, F#, .NET 4.5, XNA 4.0, Visual Studio 2010, ReSharper 7



# Zulassung

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





# Benotung

- 50% **Endprodukt**
  - Entspricht das Produkt den Anforderungen?
  - Ist das Produkt fehlerfrei (d.h. finden wir bei der Abnahme keine Fehler)?
  - Ist die Softwarequalität „gut“ (wöchentlich)?
- 50% **Einzelleistung**
  - Wurde die zugewiesene 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:
  - Min. 5 Kontrollierbare, davon min. 3 Kollidierende.
  - Min. 5 Auswählbare.
  - Min. 5 Nicht-Kontrollierbare, davon min. 3 Kollidierende.
- Min. 1000 gleichzeitig aktive Spielobjekte möglich.





# Funktionale Anforderungen

- Arten von Aktionen:
  - Min. 10 verschiedene Aktionen (inkl. Laufen, Fähigkeiten, usw.).
  - Allen Spielobjekten muss es möglich sein, von jedem beliebigen Punkt in der Welt zu jedem anderen begehbaren Punkt zu gelangen, ohne zu kollidieren, 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.
- Richtlinien zur Bedienbarkeit von Computerspielen beachten (Wiki-Artikel „Usability beim Spieldesign“).



# Randbedingungen

- Programmiersprache C# und/oder F# mit .NET 4.5.
- XNA 4.0 (XNA Game Studio 4.0 (Windows Phone SDK 7.1): 28. September 2011).
- Alternativ: MonoGame oder ANX.
- Auf Windows 7 x86/x64 lauffähig.
- Visual Studio 2010.
- Keine Warnings oder Errors vom Compiler oder ReSharper (wöchentlich), keine Buildfehler.



# Beispiel Echtzeitstrategiespiel





# ABLAUF



Woche	Organi- sation	Ent- wurf	MS 01	MS 02	MS 03	MS 04	MS 05	Was?	Wann und Wo?
0	✓	✗	✗	✗	✗	✗	✗	<ul style="list-style-type: none"> <li>Einführungsveranstaltung besuchen</li> <li>Fragebogen ausfüllen</li> <li>Gruppeneinteilung abwarten</li> </ul>	<ul style="list-style-type: none"> <li>Einführungsveranstaltung: 17.04., 15:30 - max. 19:00, 082-00-006</li> <li>Fragebogen: Bis 17.04. 23:59</li> <li>Gruppeneinteilung: Am 19.04. online</li> </ul>
1	✓	✓	✗	✗	✗	✗	✗	<ul style="list-style-type: none"> <li>Vorlesung/Diskussion "How-to: GDD"</li> <li>Abgabe Hausaufgabe</li> </ul>	<ul style="list-style-type: none"> <li>Vorlesung, 24.04., 15:30 - max. 19:00, 082-00-006</li> <li>Abgabe: 27.04. bis 23:59</li> </ul>
2	✗	✓	✓	✗	✗	✗	✗	<ul style="list-style-type: none"> <li>Abgabe GDD (beta)</li> </ul>	<ul style="list-style-type: none"> <li>Abgabe: 04.05. bis 23:59</li> </ul>
3	✗	✓	✓	✗	✗	✗	✗	<ul style="list-style-type: none"> <li>Vorlesung/Diskussion: "How-to: Architektur"</li> </ul>	<ul style="list-style-type: none"> <li>Vorlesung: 08.05., 15:30 - max. 19:00, 082-00-006</li> </ul>
4	✗	✓	✓	✓	✗	✗	✗	<ul style="list-style-type: none"> <li>MS01 erreicht (Spielobjekt in der Welt bewegbar, interaktive Kamera, Karte laden/speichern, Soundausgabe)</li> <li>Besprechung Architektur durch Tutoren</li> <li>Präsentation des aktuellen Stands</li> <li>Abgabe Architektur (beta)</li> </ul>	<ul style="list-style-type: none"> <li>Präsentation: 15.05. 15:30 - TBA, 082-00-006</li> <li>Abgabe: 18.05. bis 23:59</li> </ul>
5	✗	✓	✗	✓	✗	✗	✗		
6	✗	✓	✗	✓	✓	✗	✗	<ul style="list-style-type: none"> <li>MS02 erreicht (Mehrere Spielobjekte bewegen, Interaktionen zwischen Spielobjekten, Pathfinding, Screen-Management, Menü, HUD, Musik)</li> <li>Vorlesung/Diskussion Clean Code und Code Review</li> </ul>	<ul style="list-style-type: none"> <li>Vorlesung: 29.05., 15:30 - max. 19:00, 082-00-006</li> </ul>
7	✗	✓	✗	✗	✓	✗	✗	<ul style="list-style-type: none"> <li>Abgabe GDD (final)</li> </ul>	<ul style="list-style-type: none"> <li>Abgabe: 08.06. bis 23:59</li> </ul>
8	✗	✗	✗	✗	✓	✗	✗	<ul style="list-style-type: none"> <li>Präsentation Programm (beta)</li> <li>Abgabe Programm (beta)</li> </ul>	<ul style="list-style-type: none"> <li>Präsentation: 12.06. 15:30 - 18:30, 082-00-006</li> <li>Abgabe: 15.06. bis 23:59</li> </ul>
9	✗	✗	✗	✗	✓	✓	✗	<ul style="list-style-type: none"> <li>MS03 erreicht (KI, primäre Interaktionen vorhanden, Sieg-/Niederlagebedingungen, vollständiges Pathfinding, Inhalte, Grafik/Soundeffekte)</li> </ul>	
10	✗	✗	✗	✗	✗	✓	✓		
11	✗	✗	✗	✗	✗	✓	✓		
12	✗	✗	✗	✗	✗	✓	✓	<ul style="list-style-type: none"> <li>MS04 erreicht (finale Version vorhanden)</li> </ul>	
13	✗	✗	✗	✗	✗	✗	✓	<ul style="list-style-type: none"> <li>MS05 erreicht (Fehlerbehebung &amp; Balancing)</li> <li>Präsentation Programm (final)</li> <li>Abgabe Architektur (final)</li> <li>Abgabe Programm (final)</li> </ul>	<ul style="list-style-type: none"> <li>Präsentation: 17.07. 15:30 - TBA, 082-00-006</li> <li>Abgabe: 20.07. bis 23:59</li> </ul>



# 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, grob:
  - Werkzeuge installieren, Dienste testen.
  - Trac kennenlernen.
  - Texte auf Wiki lesen
    - Clean Code, Dokumentation, Usability, Trac und SVN
  - XNA Programm schreiben.



# Game Design Document

- GDD beschreibt die wesentlichen Merkmale des Spiels für den Auftraggeber.
  - Ähnlich zu **Lastenheft**.
- Details in Vorlesung nächste Woche 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 Vorlesung in Woche 3.



# Umsetzung

- Grob gegliedert in **5 Milestones (MS)**
  - Unsere MS beschreiben einen **Referenzablauf**.
  - Behalten Sie den Termin, definieren Sie sich passende Inhalte.
  - 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 **Requirements** mit abgeleiteten **User Stories**.
  - Anforderungen besitzen einen „**Business Value**“.
  - Ist geordnet nach Entwicklungsreife.
- **Requirements** sind (hier!) 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.

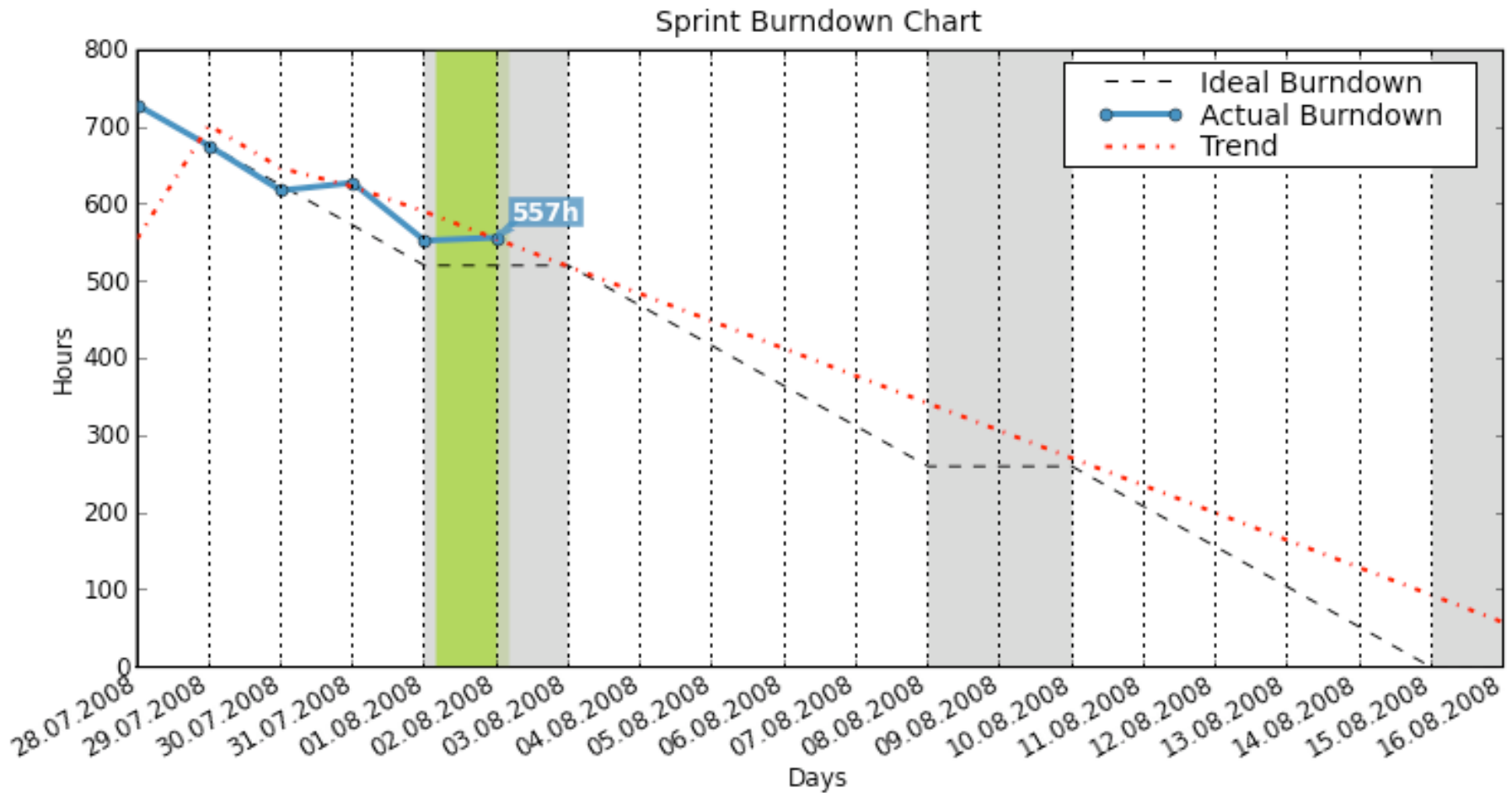


# Artefakte

- **Product Increment**
  - Am Ende eines Sprints erzeugte neue Version des Produkts.
  - Sollte direkt auslieferbar sein.
- **Definition of Done**
  - 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** 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

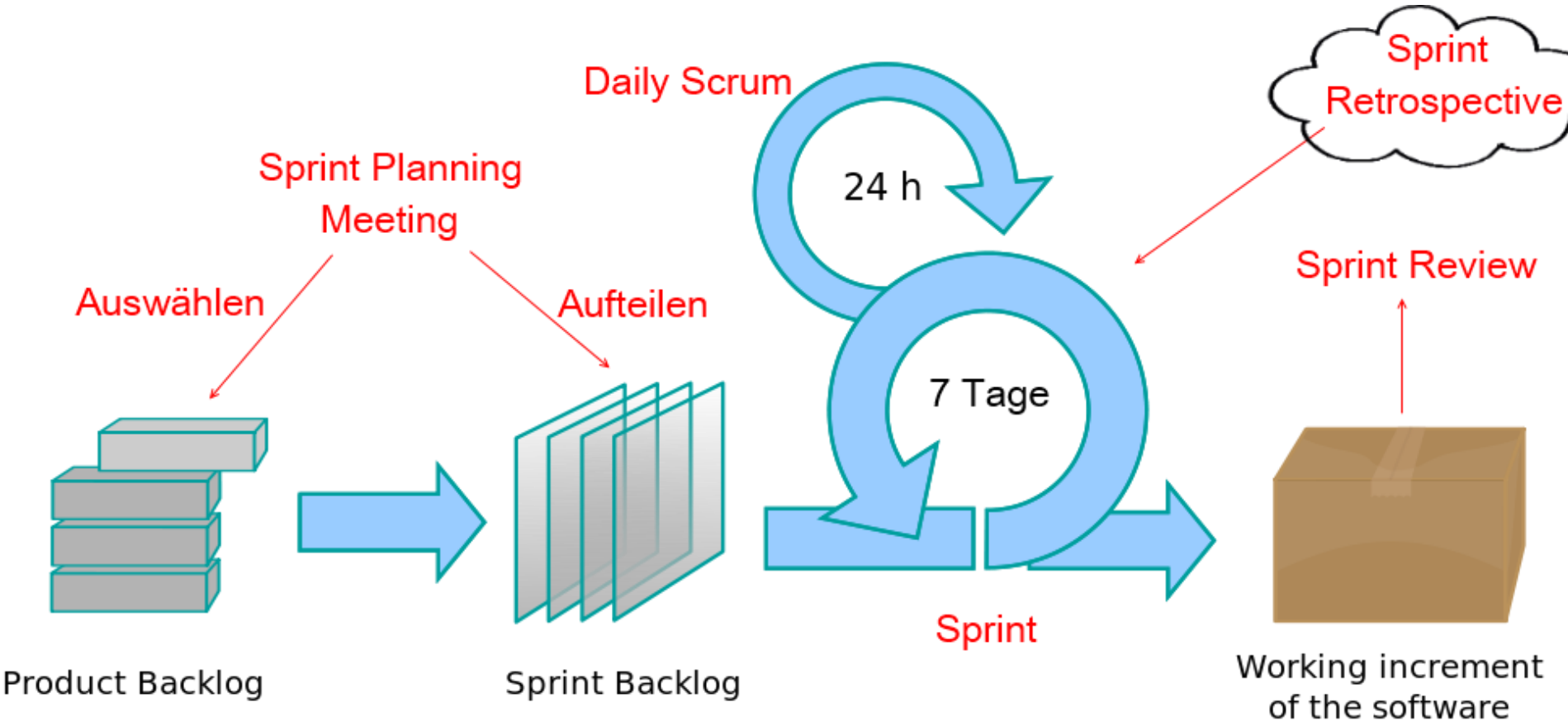
- Treffen des Teams **nach jedem Sprint**.
- Länge abhängig von Sprint-Länge (**1h pro Woche**).
- **Product Owner** bestimmt, welche Tasks und User Stories fertig sind.
  - Unfertige User Stories kehren in das Product Backlog zurück.
- **Developer** demonstrieren neuen Product Increment.
- **Product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





# SCRUM IM SOFTWAREPRAKTIKUM



# Product und Sprint Backlog 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



# Gruppentreffen

- Länge genau **2h**.
- **Aufteilung**
  1. Daily Scrum (15min)
  2. Sprint Review (30min)
  3. Sprint Planning (1h)
    - **Was** wird im nächsten Sprint von **wem** erledigt?
    - Nicht so detailliert: **Wie** wird es erledigt.
  4. Sprint Retrospective (15min)
- Verbleibende Zeit wird mitgenommen.



# Rollen

- **Tutoren** sind Scrum-Master.
- **Tutoren** sind Vertreter der Kunden.
- **Dozenten** sind Kunden.
- **Sie** sind Developer.
  
- Was ist mit dem Product Owner?



# Recurring Tasks

- Ab Woche 2: **Product Owner**
  - Pflegen und Anpassen von Requirements und User Stories im Product Backlog.
  - Verfeinern von Requirements zu User Stories.
  - Requirements nach Entwicklungsreife ordnen.
  - Gruppentreffen vorbereiten (was ist fertig, wie war die Aufwandsabschätzung).



# Recurring Tasks

- Ab Woche 3: **Architektur**
  - Schnittstellen definieren.
  - Architekturbeschreibungen pflegen.
  - Einhaltung der Architektur sicherstellen.
- Ab Woche 6: **Qualitätssicherung**
  - Code auf Clean-Code Richtlinien prüfen.
  - Code Reviews vorbereiten.
  - ReSharper Konformität herstellen.





# Was nun?

1. Fragebogen ausfüllen bis heute Abend 23:59 Uhr!
2. Auf Gruppeneinteilung warten (bis 19.4.).
3. Hausaufgabe machen bis Sa., 27.4., 23:59 Uhr.

Nach der Gruppeneinteilung:

- Regelmäßigen Termin für Gruppentreffen ausmachen.
- Treffen Sie sich mit Ihrer Gruppe und dem Tutor.
- Entwickeln Sie eine Spielidee.
- Machen Sie sich mit den Werkzeugen und Techniken vertraut.



**FRAGEN?**