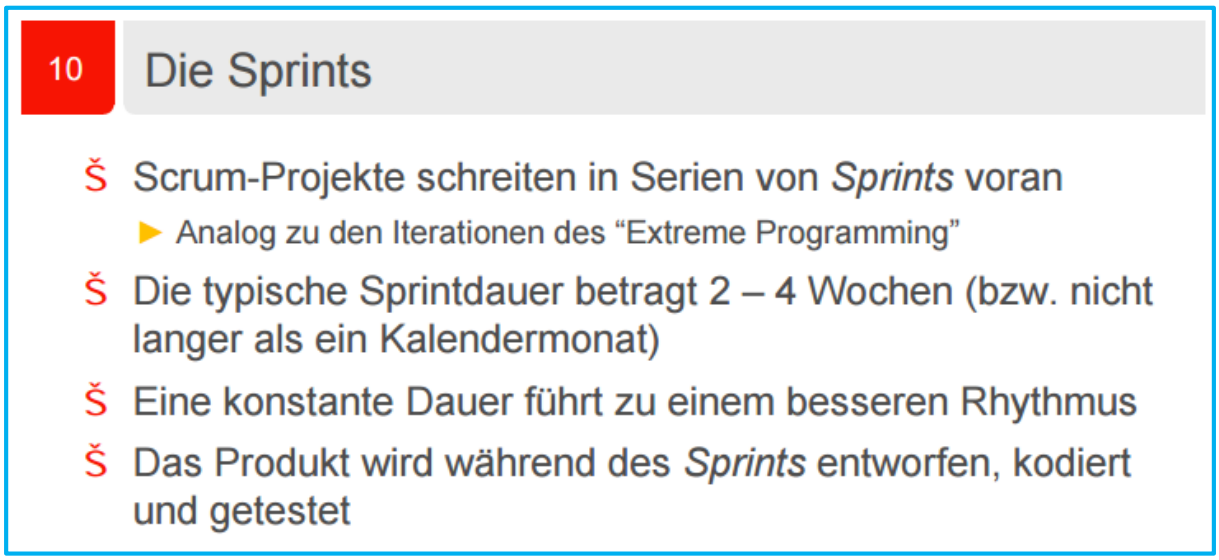
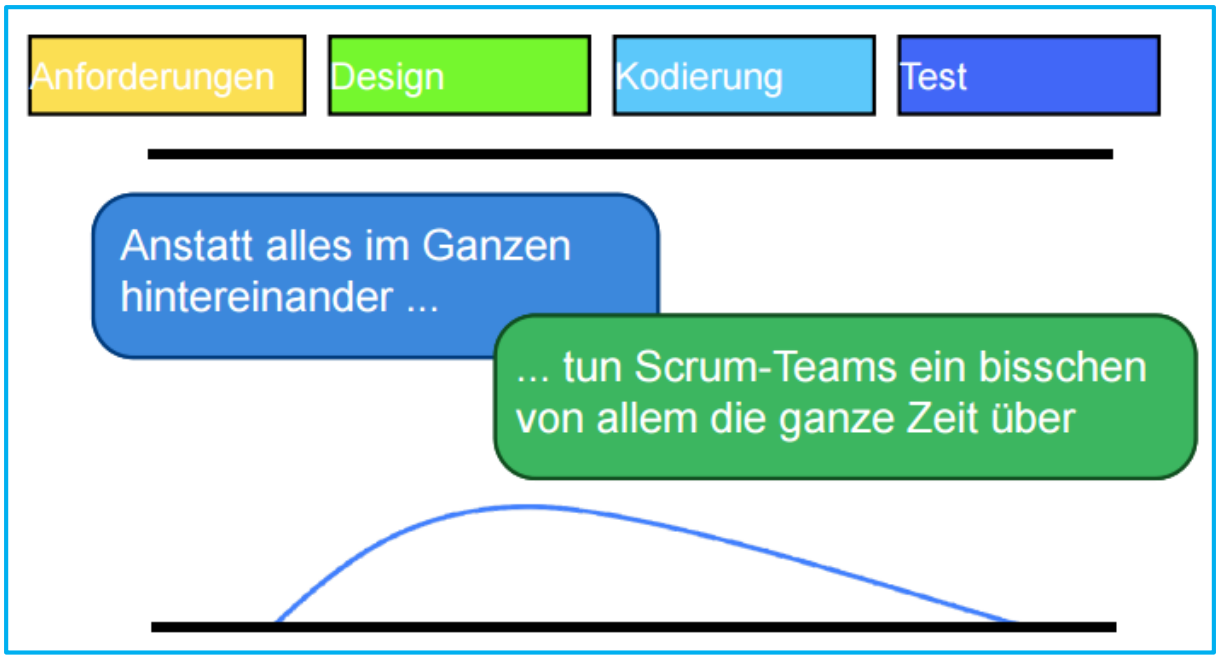
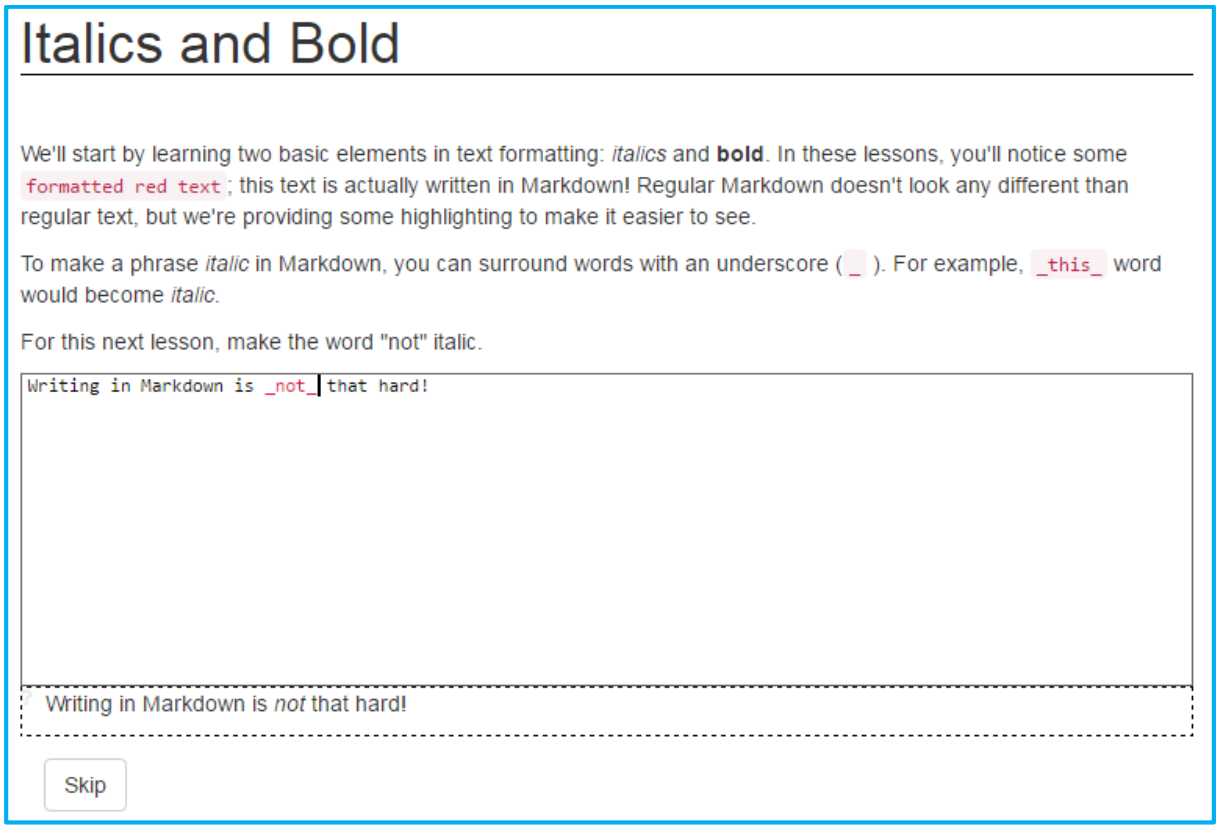
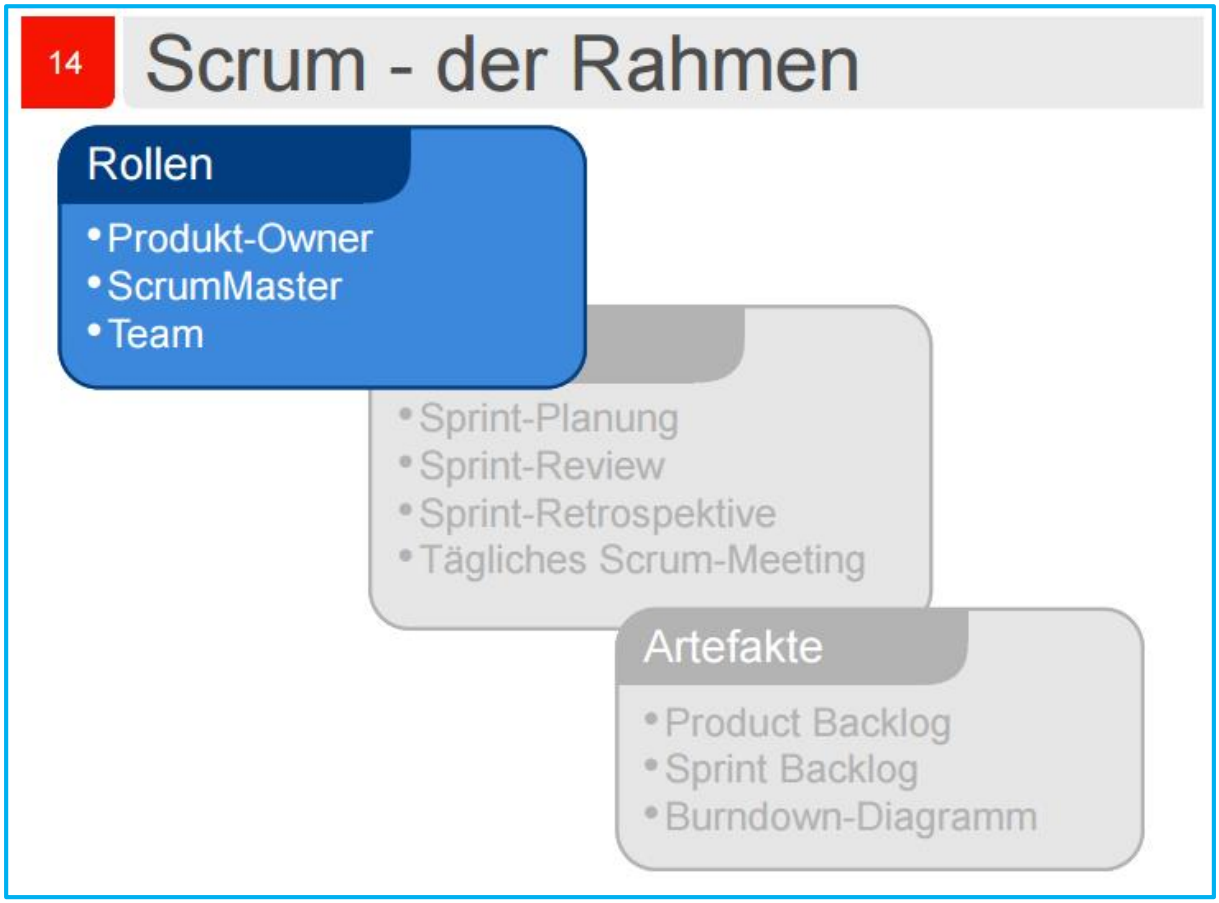
Hilfestellungen Software Engineering





Erstellen Sie auf GitHub im Markdown-Format ein Dokument




14
Scrum - der Rahmen
Rollen
• Produkt-Owner
• ScrumMaster
• Team
• Sprint-Planung
• Sprint-Review
• Sprint-Retrospektive
• Tägliches Scrum-Meeting
Artefakte
• Product Backlog
• Sprint Backlog
• Burndown-Diagramm

1. Einführung

1.1 Zweck

Dieses Anforderungsdokument ist im Zusammenhang mit dem Projekt "Börsen Dateien automatisiert herunterladen" erstellt worden. Das Dokument beinhaltet eine Übersicht, sowie die einzelnen Funktionen des geforderten Systems. Ebenfalls werden alle Anforderungen in diesem Dokument niedergeschrieben.

1.2 Konventionen

Es wird durchgehend die Schriftart "Arial" verwendet. Begriffe welche im Glossar erläutert werden, sind beim ersten Vorkommen kursiv geschrieben.

1.3 Zielgruppe und Lesehinweise

Dieses Dokument adressiert alle Stakeholder (siehe nächster Abschnitt). Da es ein kleineres Projekt ist, gibt es keine verschiedenen Lesepfade für die einzelnen Stakeholder-Gruppen. Das Dokument kann von der Einführung über die Übersicht, Funktionen, Anforderungen bis zum Anhang der Reihe nach gelesen werden.

Rolle	Beschreibung	Vertreter	Verfügbarkeit	Relevanz	Wissen
Anwender (Broker)	Braucht die Daten der verschiedenen Börsen um damit arbeiten zu können.	Herr Y. Frau X.	Bürozeiten (5 x 8h)	Gross	Tiefes Wissen über den Inhalt der Daten.
Management	Segnen das Projekt ab und sind an Qualitäts- und Effizienzsteigerung interessiert.	CIO Herr Z.	Mittwochs (14:00 – 14:30) Während dem Projekt	Mittel	Überblick über der Kosten und Infrastruktur der IT Abteilung
Börsen	Stellen die benötigten Daten zur Verfügung.	Auf Anfrage	Auf Anfrage	Gering	Wissen über den Inhalt der Daten, sowie die Abholung derjenigen.
IT-Supporter	Supporten das System und führen Änderungen daran aus.	Herr A.	Bürozeiten (5 x 8h)	Gross	Tiefes Wissen der IT Infrastruktur und der Datenübertragungen.
Bereichsleiter Broker	Auftraggeber und Vorgesetzter der Broker.	Herr B.	Bürozeiten (5 x 8h)	Gross	Übersicht der Arbeiten der Broker.

Entwickler	Implementieren die definierten Anforderungen.	Herr C.	Bürozeiten (5 x 8h)	Gross	Implementierung und Anforderungen sind im Detail bekannt.

1.4 Projekt Umfang / Vision

Eine automatische Dateisynchronisation, welche von beliebig definierbaren Börsen die Daten des letzten Handelstages holt und lokal abspeichert.
Der Zustand des Services, sowie der einzelnen Datentransfers sollen im Intranet publiziert werden.
Dieses Automation erhöht die Produktivität und lindert die Fehleranfälligkeit gegenüber dem heutigen vorgehen.

1.5 Referenzen

Referenz	Beschreibung
http://www.uml.org/	Beschreibung zur Unified Modeling Language
Projektantrag.doc	Projektantrag an das Management

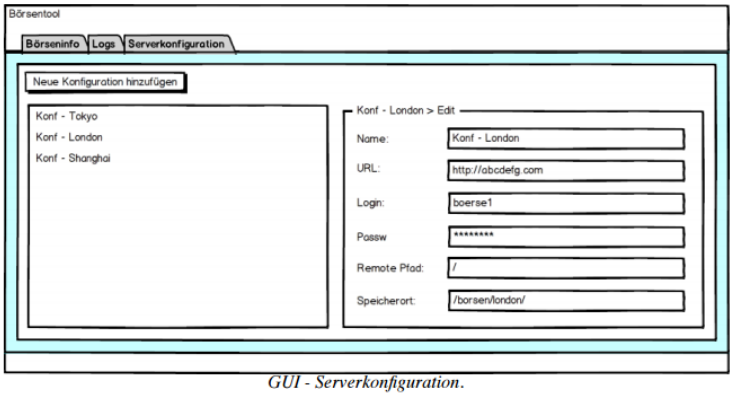
2.5 Betriebsumgebung

Das System läuft auf einem virtuellen Server auf der bereits existierenden Serverfarm. Somit sind Updates, Backups und die Wartung bereits im allgemeinen Prozess der virtuellen Server enthalten. Als Betriebssystem kommt Windows Server 2008 in den Einsatz. Der bereits bestehende Fileserver wird weiter genutzt. Die Berechtigungen für die User sind da bereits richtig gesetzt. Auf dem virtuellen Windows Server läuft die Überwachung sowie auch die Komponente, welche die Dateien automatisiert herunterlädt.

2.7 Benutzer Dokumentation

Die erstellten Dokumentationen werden im firmeninternen Wiki abgelegt und verlinkt. Es wird Prozessablauf geschrieben, wie ein User im Fehlerfall vorgehen muss. Des weiteren wird die Software selber dokumentiert. Ein Wartungsdokument für das System wird erstellt und entsprechend umgesetzt. Zuletzt wird noch ein Prozessablauf für die Administratoren und die User erstellt, welcher zeigt, wie eine neue Verbindung hinzugefügt wird.

GUI - Übersicht.



GUI - Serverkonfiguration.

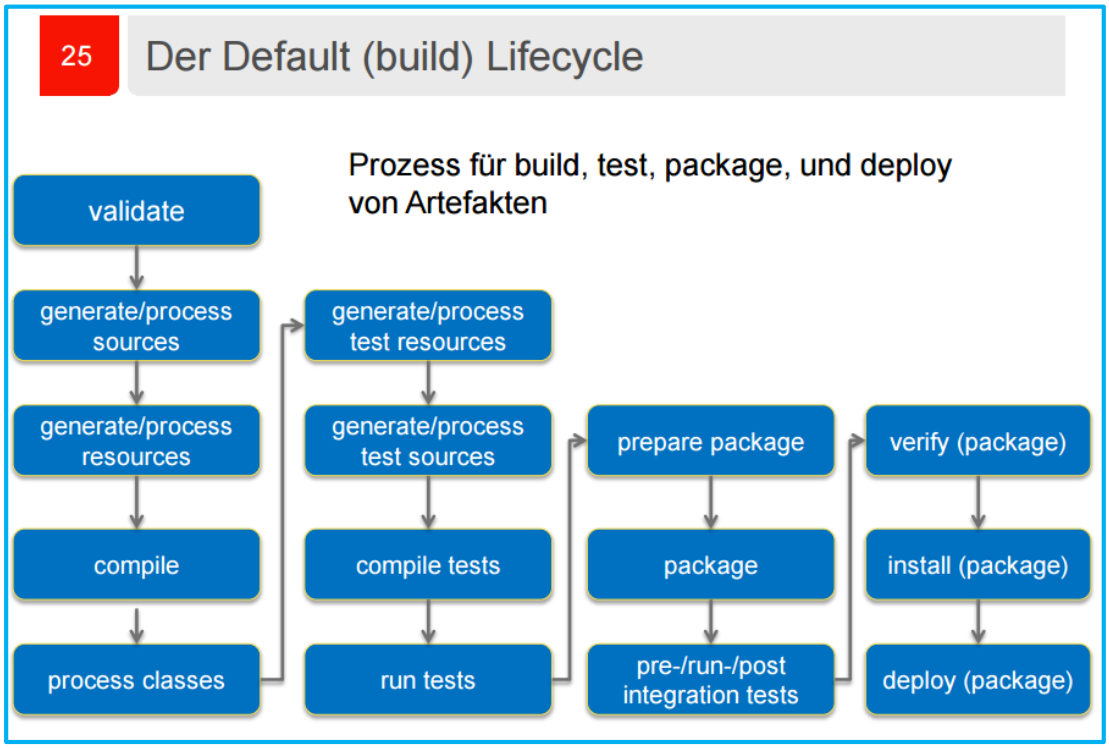
C. Projektbeschreibung	
Ausgangssituation / Projektbegründung:	Unsere Brokerfirma benötigt Börseninformationen nach einem Handelstag von den verschiedenen wichtigen Börsen rund um den Globus wie Tokio, Hong Kong, Singapur, Frankfurt, Zürich, Paris, London und New York. Aktuell werden die Daten auf den jeweiligen Servern der Börse nach Handelsschluss zur Verfügung gestellt. Jeden Morgen bzw. nach Handelsschluss der einzelnen Börsen muss ein Administrations-Mitarbeiter der Abteilung diese Dateien (10...1000 Stück) manuell auf den lokalen Dateiserver kopieren. Die Dateien befinden sich auf den Remote Servern der Börsen teilweise in einem oder mehreren Verzeichnissen. Für den Download wird aktuell das Programm WinSCP eingesetzt. In diesem Programm werden die Verbindungen zu den einzelnen Servern hinterlegt, so dass der Verbindungsaufbau schnell durchgeführt werden kann. Vor dem Download werden die Daten auf dem lokalen Fileserver der jeweiligen Börse in ein Archivverzeichnis verschoben. Pro Tag wird pro Börse ein neues Archivverzeichnis erstellt. Die Broker greifen aktuell auf freigegebenes FTP-Verzeichnis auf die Daten zu. Die interne IT ist verantwortlich für den Betrieb der Infrastruktur
Projektgesamtziel:	Unsere Brokerfirma muss jederzeit auf aktuelle Börsendaten Zugriff haben. Mit dem Börsen Datei Downloader (BDD) soll dieser Prozess effizient automatisiert und zeitnah zum Börsenschluss der Datenlieferanten im Hintergrund erfolgen. BDD soll jederzeit durch den Kunden einfach um weitere Quellserver erweitert werden können. Selbstverständlich besteht jederzeit der Zugriff auf aktuelle und alte Börsendaten, so dass diese für Auswertungen oder Vergleiche genutzt werden können. Das System überwacht sich selbständig und informiert im Störungsfall die gewünschten Mitarbeiter per Email.
Projektteilziele ····→	**Messbare Ergebnisse**
Konfiguration	• → IT-Abteilung kann selbständig innerhalb 1h neue Serverkonfigurationen erfassen und testen
Datenaktualität	• → Börsendaten sind (im fehlerfreien Betrieb) max. so alt wie in der Konfiguration definiert
Aufwand	• → Kein manueller Aufwand durch IT um Daten abzuholen
Datenarchiv	• → Rückwirkend auf alle veröffentlichten Daten Zugriff

Nicht-Ziele:	Der optimierte Zugriff via CMS wird in einem separaten Projekt geführt
Wirkung / Nutzen:	Das BDD-System soll die IT vom täglichen Abgleich-Aufwand befreien, gleichzeitig aber jederzeit aktuelle und vollständige Daten für die Broker garantieren.
Projektphasen / Hauptaufgaben:	1. → Abstimmung über Projekt und Zielsetzung 2. → Ist-/Soll-Zustand 3. → Fachliche/technische Detailierung 4. → Pflichtenheft 5. → System + Systemkontext abgrenzen 6. → Anforderungskatalog
Projektrisiken:	• → Das Qualitätsrisiko wird als Mittel eingestuft (Fehlerüberwachung des Systems) • → Sofern keine Änderungen seitens der Börsen stattfinden, sind keine technischen Risiken zu erwarten • → Für die Projektphase müssen die Stakeholder entsprechend der Planung verfügbar sein. Die Systemumstellung hat keinen Arbeitsunterbruch oder Verzögerung zur Folge • → Es sind keine Terminrisiken zu erwarten • → Das Akzeptanz-Risiko wird als sehr klein betrachtet, da die Broker nur indirekt mit dem System in Kontakt stehen (Datenabruf via bestehendem CMS). Für die IT-Abteilung stellt dies eine Vereinfachung dar.

Build-Anleitung

(checkout, mvn …., java –jar uhr.jar

a) Verständlichkeit Build-Anleitung



27 Beispiele von Maven Goals

- Ein Lifecycle “Goal” muss definiert werden
- mvn install
  - Invokes generate* and compile, test, package, integration-test, install
  - Installs artifact to local Maven repository (see later)
- mvn clean
  - Invokes just clean
- mvn clean compile
  - Clean old builds and execute generate*, compile
- mvn compile install → same as mvn install
  - Invokes generate*, compile, test, package, integration-test, install
- mvn clean install
  - Clean old builds and invokes generate*, compile, test, package, integration-test, install
- mvn jetty:run
  - Runs jetty web server

- Sehr gutes Help System
  - Übersicht der Hauptbefehle : git help
  - List von allen Befehlen: git help -a
  - Dateils eines spezifischen Befehls: git help <command>
  - Liste der Basics: git help -g

28 Git – Übersicht der „porclain“ Befehle

local commands

- git config
- git init
- git add
- git status
- git commit
- git log
- git diff
- git tag
- git revert

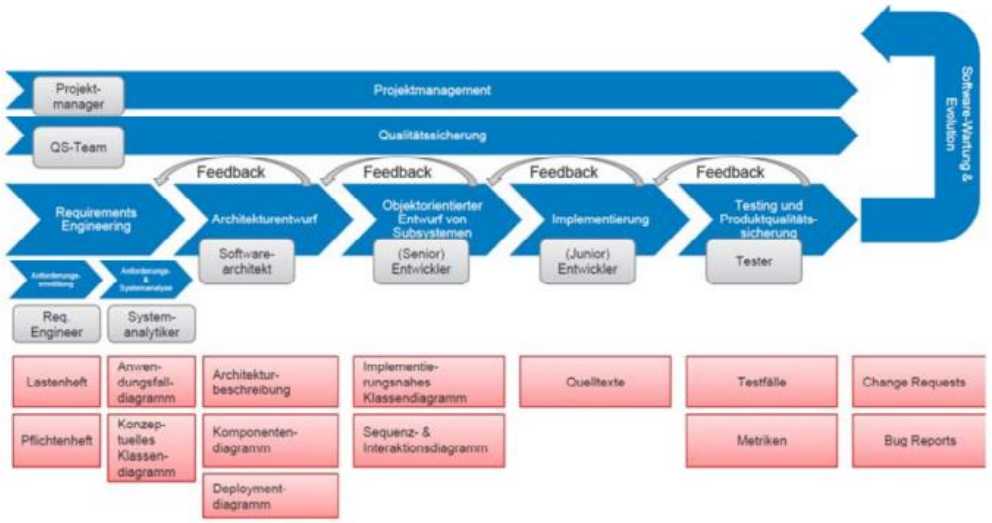
branch / merge

- git branch
- git merge
- git checkout
- git rebase
- git stash

remote commands

- git remote
- git clone
- git fetch
- git pull
- git push

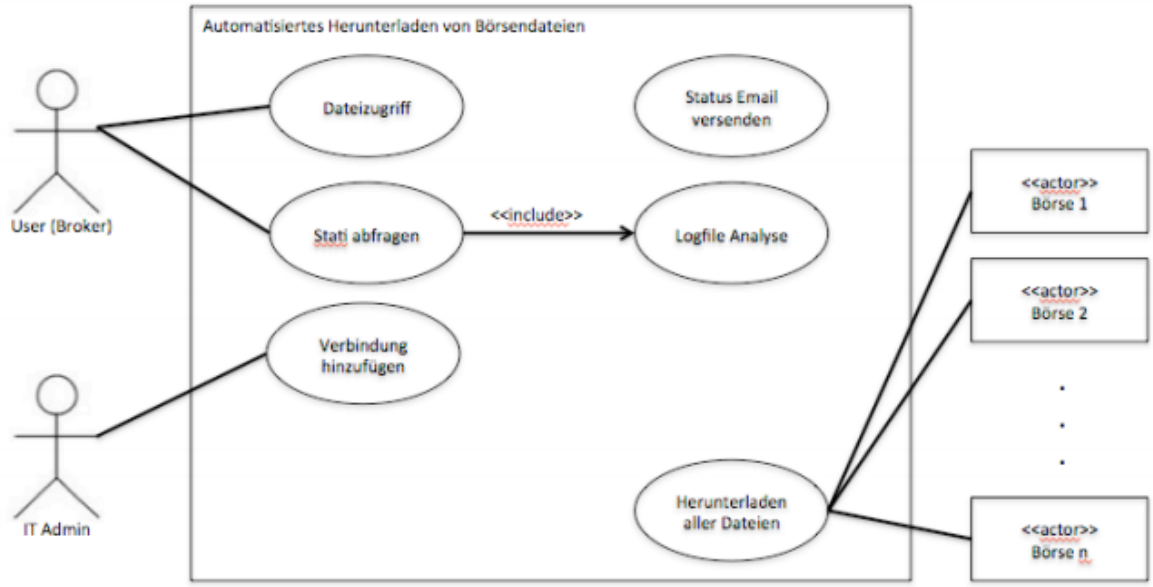
23 Software Engineering Aktivitäten, Rollen und Artefakte



Kurze Bedienungsanleitung

2.1 Produkt Perspektive

Dieses System löst das manuelle Herunterladen der Börsendateien ab. Die manuelle Tätigkeit wurde täglich ausgeführt. Die Hauptkomponenten, sowie Tätigkeiten des automatisierten Systems sind im nachfolgenden *Use Case* Diagramm nach dem *UML* Standard dokumentiert.



2.3 Produkt Funktionen

Die Hauptfunktionen des Systems sind nachfolgend aufgelistet und auf der nächsten Seite in einem Flussdiagramm dargestellt.

- Der User kann Dateien anfordern.
- Der User kann den Zustand der Übertragungen einsehen.
- Das System lädt alle benötigten Dateien automatisch bei den verschiedenen Börsen herunter.
- Alte Dateien werden archiviert.
- Der User bekommt ein Email wenn eine Datei nicht verfügbar ist.
- Der Administrator kann neue Verbindungen einfach hinzufügen.
- Die heruntergeladenen Dateien werden auf dem lokalen File Server gespeichert.
- Die Überwachung nimmt neue Verbindungen automatisch hinzu.
- Die einzelnen Arbeitsschritte des Systems werden in ein Logfile geschrieb

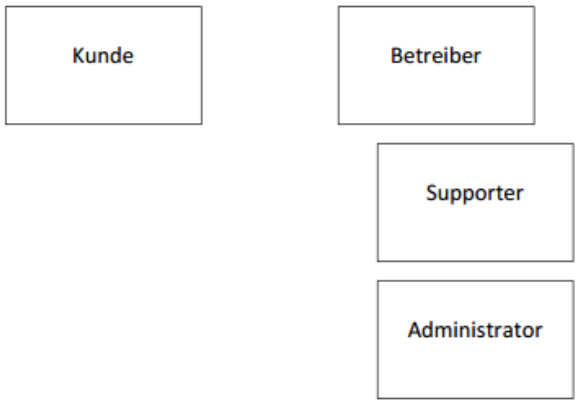
User Stories inkl. Akzeptanzkriterium, Aufwandschätzung in Story-Points und Priorisierung

b) Verständlich?

b) Vollständig formuliert?

User Rolle	Administrator
Die Software wird im täglichen Arbeitsablauf angewendet. Es ist ein detailliertes Wissen über die Software und die Grundthematik vorhanden. Das System wird überwacht und im Falle einer Störung wird ein Supporter aufgeboten. Das wichtigste Ziel ist, dass die Software möglichst fehlerfrei funktioniert.	

Beispiel der grafischen Anordnung der User Rollen:



Musterlösung Priorisierung

Im folgenden Dokument sind alle User Stories definiert, die mit der Fallstudie Parking Meter & Parking Spot Monitor zusammenhängen.

Aufgabe

User Story 1 – Parking Meter

Priorisierung: 1	Parkplatznummer eingeben	Story Points: 1
	Als Kunde möchte ich eine Parkplatznummer auswählen, um die Münzen einzuwerfen	
<Risiko>		<Story Points> (Post-Schätzung)

Angenommen die Parkuhr ist bereit eine Nummer entgegen zu nehmen,
wenn ein Parkplatz gewählt wird,
dann erscheint die Aufforderung Münzen einzuwerfen

Releaseplan mit den Ausbaustufen

c) Releaseplan realistisch?

Musterlösung Release Plan

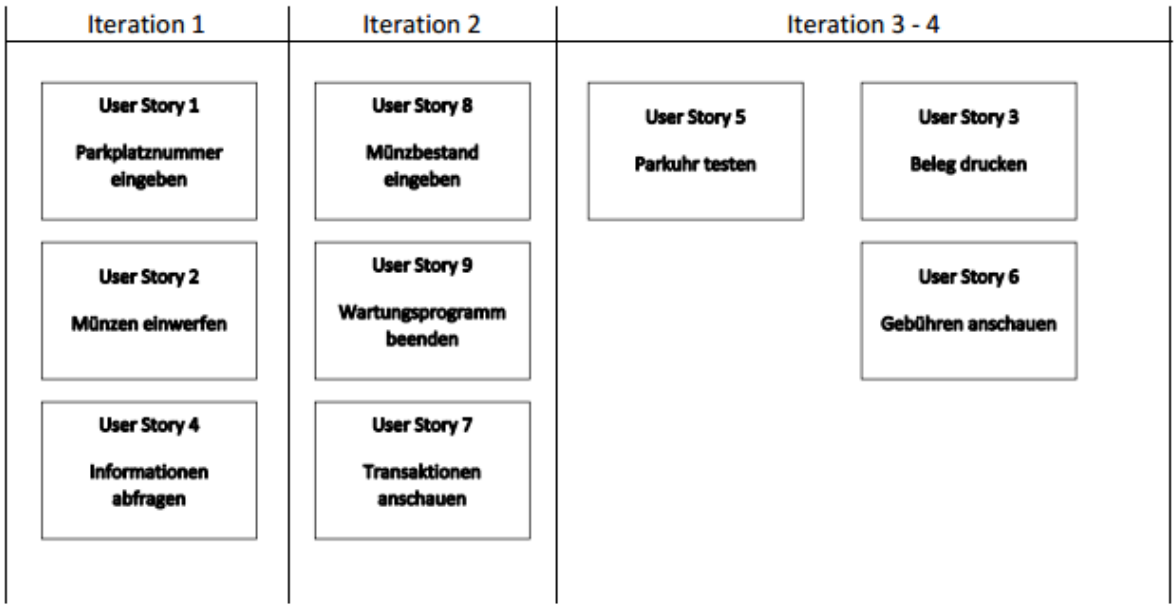
In dem folgenden Dokument sind alle User Stories in einem Release Plan zusammengefasst.

Aufgabe

Aufteilung der User Stories in Tasks.

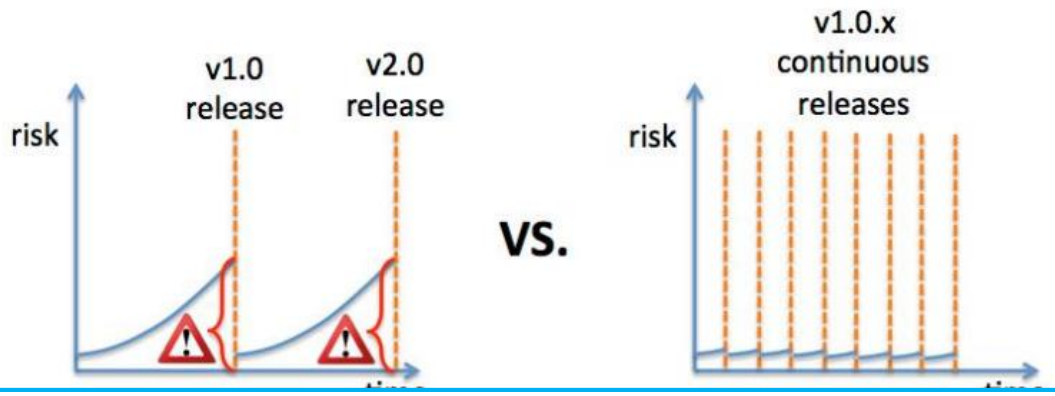
User Story 1	Als Kunde möchte ich eine Parkplatznummer auswählen, um die Münzen einzuwerfen	
Task 1	Design des User Interface	1 Stunde
Task 2	Implementierung des User Interface	2 Stunden
Task 3	Testen der Umsetzung	3 Stunden

Release Plan:

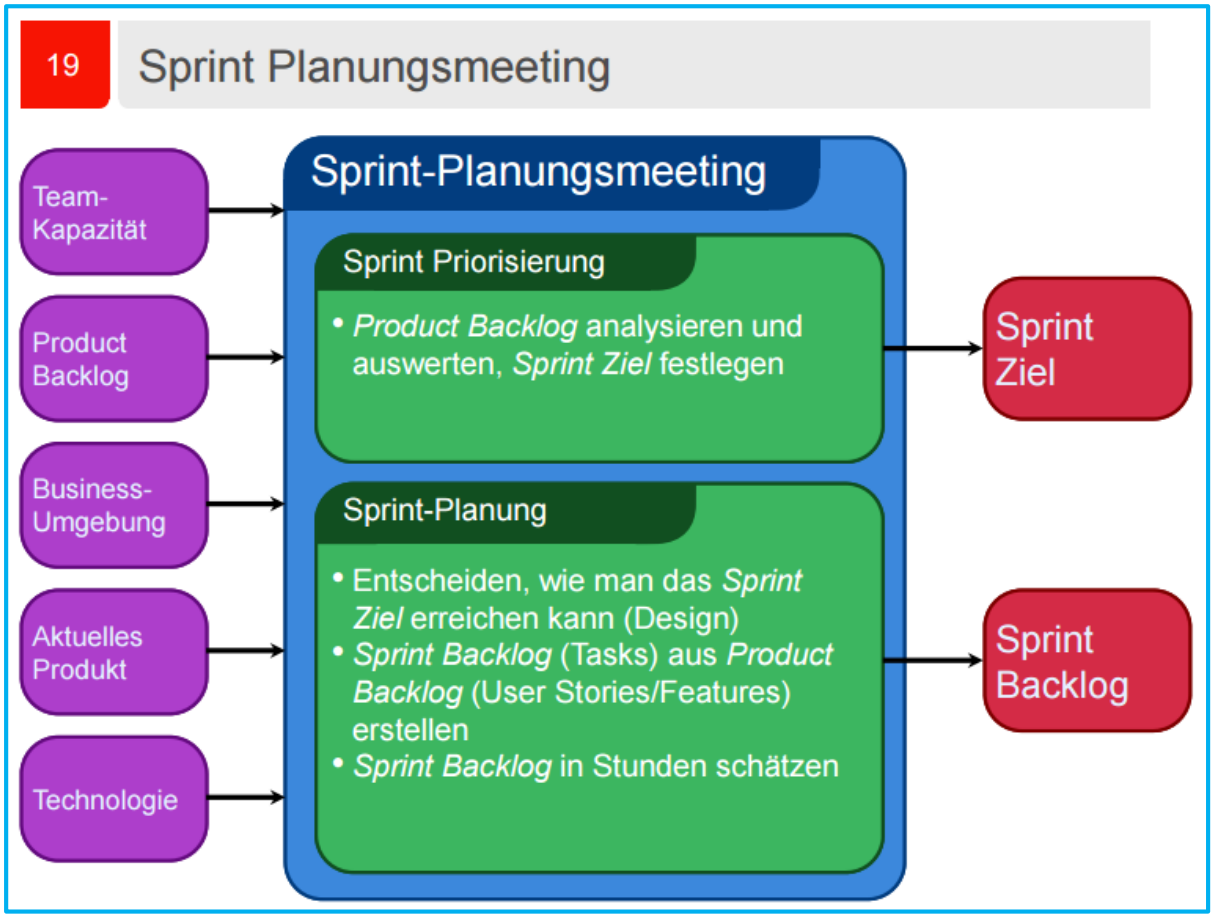


9 Risikominimierung durch regelmässigen Build

- Neue Releases und Verteilung durch viele kleine Aktualisierung in einem kurzen Zyklus reduziert das Risiko
  - Nur kleine Änderungen → Anzahl Code für Review und Test ist kleiner
  - Viele kleine Verbesserungen, schneller Code-Fix
  - Gewisse Online Systeme (wie. Netflix, Google docs) aktualisieren die Software täglich

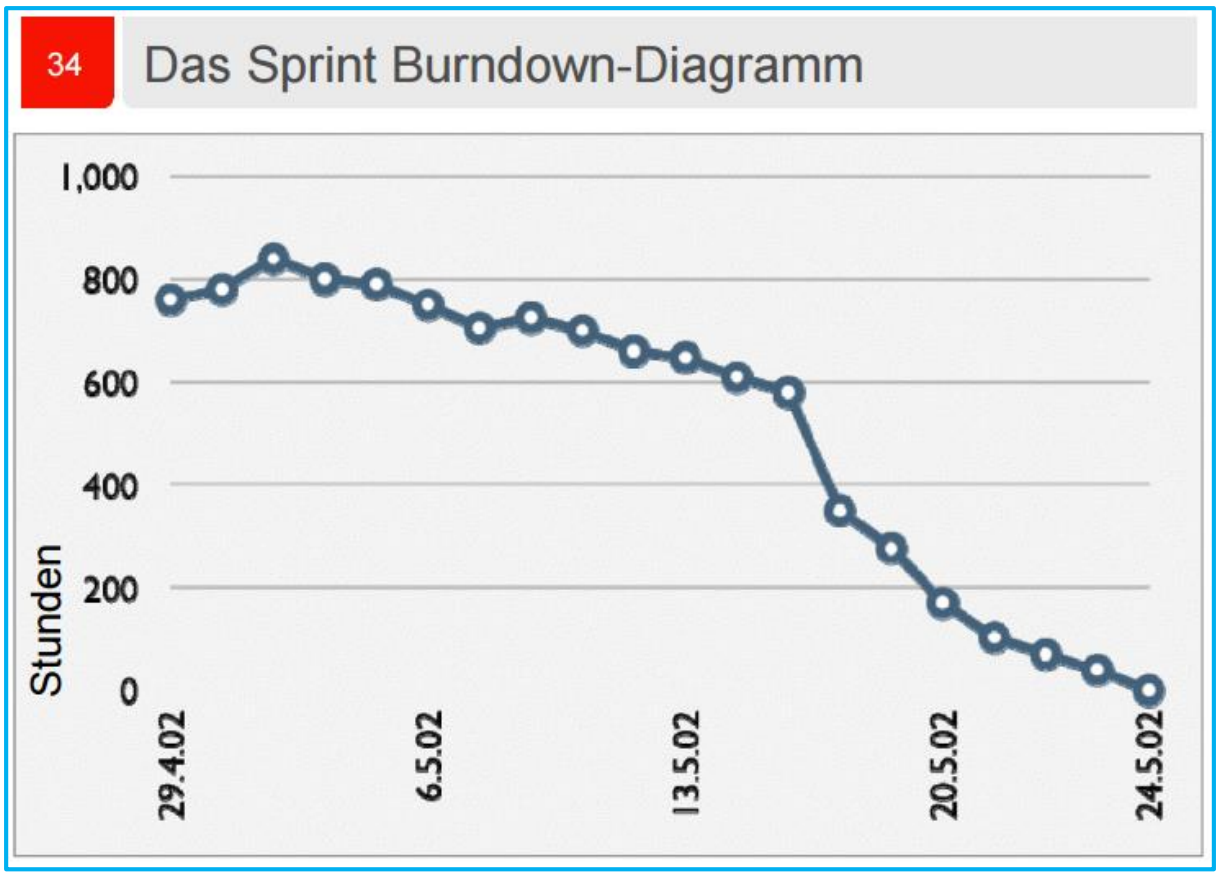


g) Wie viele Sprints wurden geplant?



33 Sprint Backlog: Beispiel

Tasks	Mo	Di	Mi	Do	Fr
Code the user interface	8	4	8		
Code the middle tier	16	12	10	4	
Test the middle tier	8	16	16	11	8
Write online help	12				
Write the foo class	8	8	8	8	8
Add error logging			8	4	


34
Das Sprint Burndown-Diagramm
1,000
800
600
400
200
0
Stunden
29.4.02
6.5.02
13.5.02
20.5.02
24.5.02

Dokumentation Sprint 1

Taskliste für die Umsetzung der

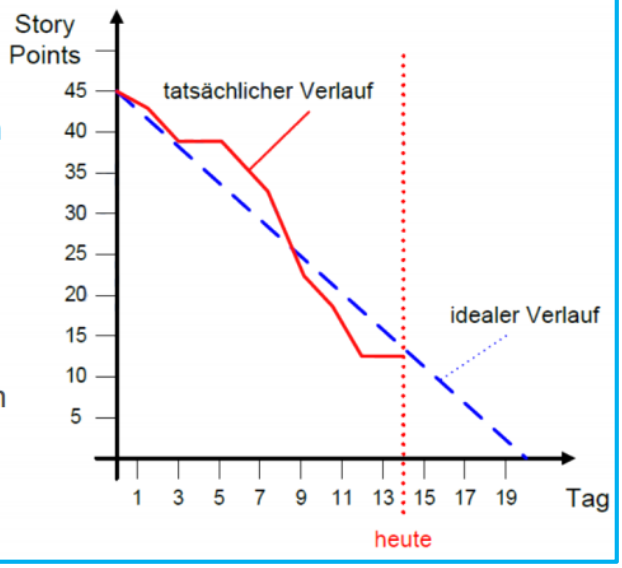
User Story (Schätzung in Stunden)

64 Task Board (1/2)

- In diesem Beispiel ist ein (typisches) Task Board dargestellt, bei dem in einem Sprint (der bereits begonnen hat) ein vierstufiger Bearbeitungsablauf gewählt wurde.

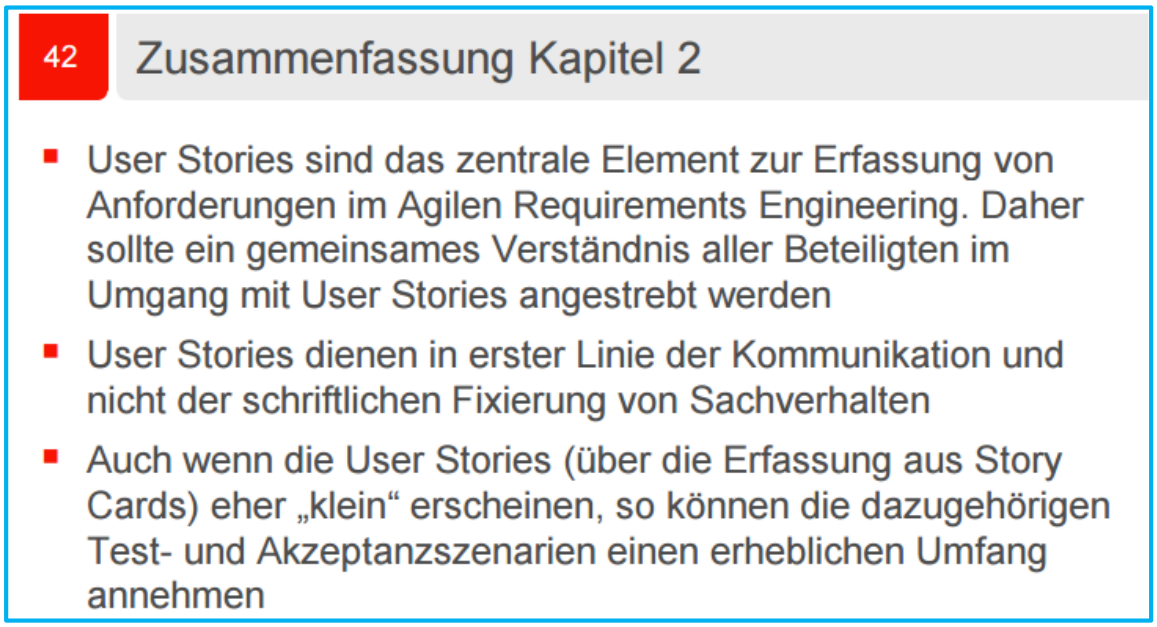
66 Burndown Chart

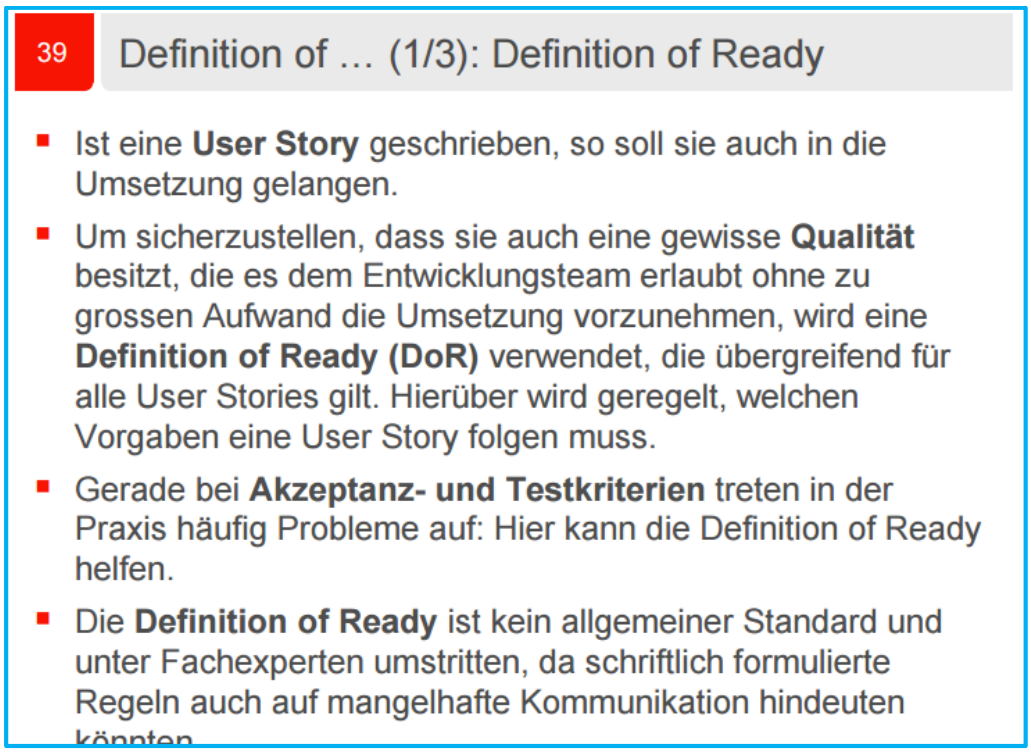
- Das hier dargestellte Sprint Burndown Chart zeigt den Verlauf eines 20-Tage-Sprints bis zum 14ten Tag. Die Geschwindigkeit des Teams beträgt 45 Story Points pro Sprint.
- Zum Stichtag („heute“) entsprechen die tatsächlich umgesetzten Story Points dem idealen Verlauf.



Anreicherung der User Stories für die Umsetzung

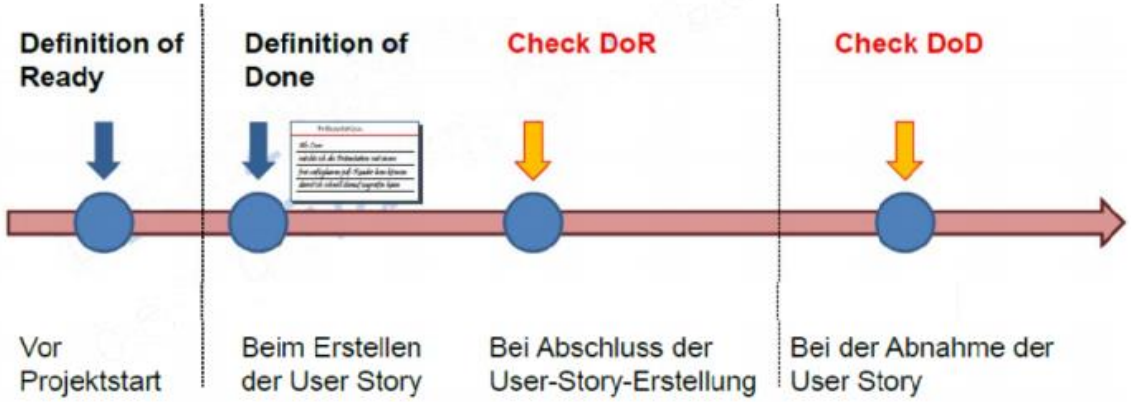
d) Liefert die Planung genügend Information für eine Umsetzung?





40 Definition of … (2/3): Definition of Done

- Die **Definition of Done (DoD)** beschreibt (im Vorhinein), wann eine **User Story** als fertig gilt, das heisst, vom Anwender abgenommen werden kann. Die (universelle) Überprüfung der DoD-Kriterien findet nach der vollständigen Erstellung der User Story statt. Üblicherweise wird die DoD über eine (Check-)Liste realisiert.



41 Definition of … (3/3): Vergleich

	Ready	Done
Kurzcharakterisierung	Qualitätskriterien für Anforderungen	Abnahmekriterien für User Stories
Bezug	Bezieht sich auf die Anforderungen	Bezieht sich auf die entwickelte Software
Wesentlich	Vollständigkeit der User Story / Story Card inkl. Priorisierung (und Schätzung)	Qualitätskriterien: Die entwickelte Software muss getestet sein und funktionieren
Ziel	„Abnahme“ einer User Story vor dem Sprint	„Abnahme“ der Umsetzung einer User Story
Wird wann erstellt?	Vor dem Projekt	Vor dem Projekt
Wird wann überprüft?	Nach der Erstellung der User Story vor Sprintbeginn	Während und nach der Umsetzung (Sprint Review) der User Story im Sprint
Wer ist verantwortlich?	Product Owner	Entwicklungsteam

UML Package, Klassen- und

Sequenzdiagramm

e) guter Entwurf erstellt?

Verständlichkeit

Kopplung

Kohäsion

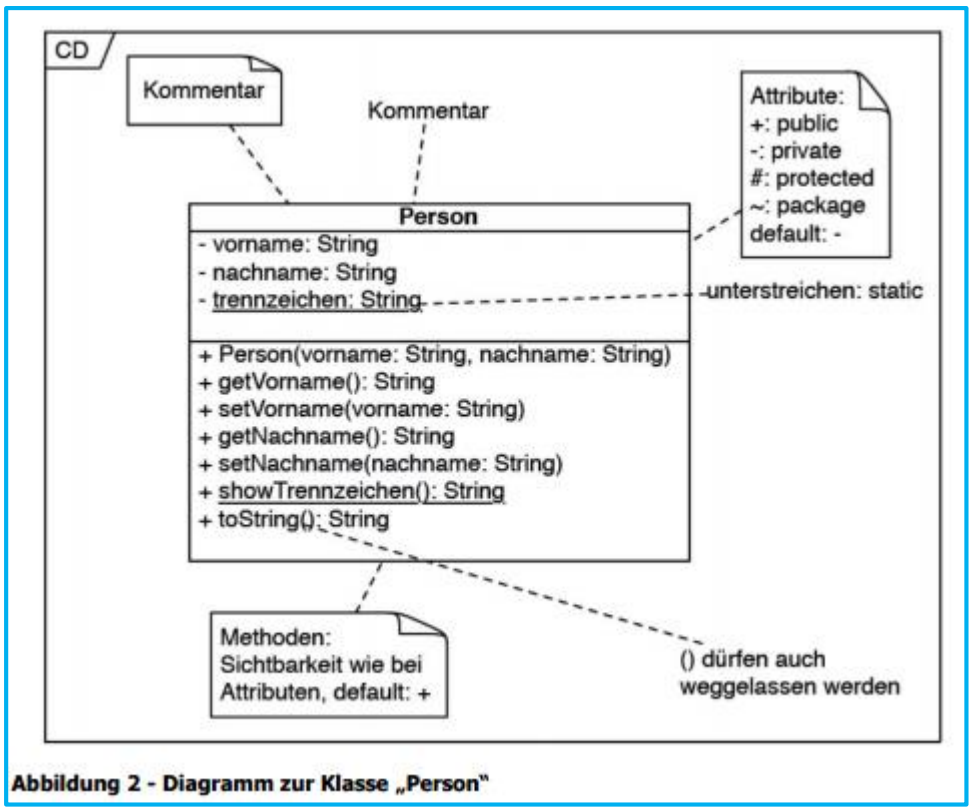


Abbildung 2 - Diagramm zur Klasse „Person"

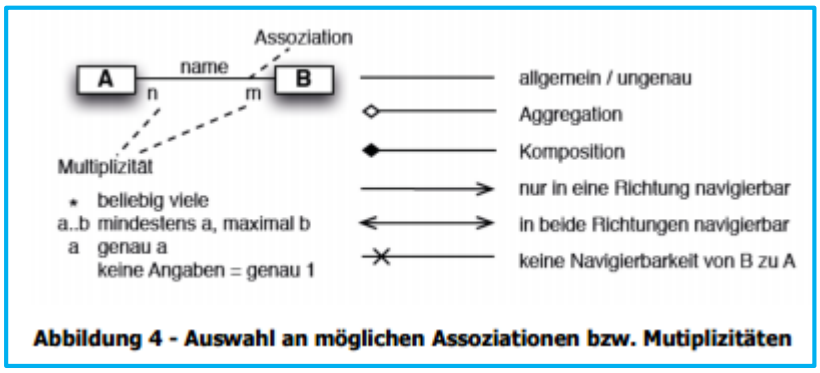


Abbildung 4 - Auswahl an möglichen Assoziationen bzw. Mutiplizitäten

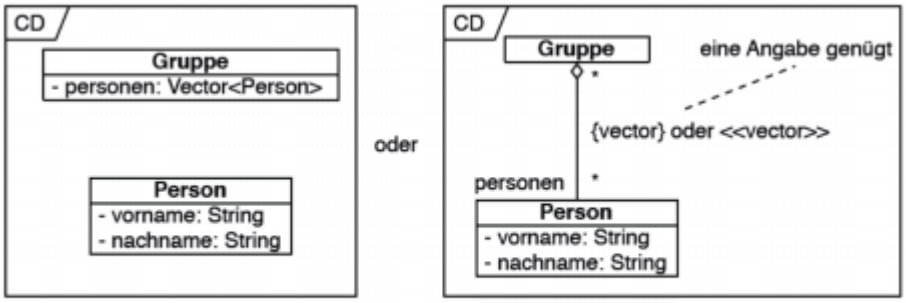


Abbildung 6 - Klassendiagramm zur Klasse Gruppe

Beispiel 2.5

Die Vererbungshierarchie zwischen der Klasse `GeometrischeFigur` und den Subklassen Kreis, Dreieck und Rechteck ist in Abbildung 7 zu sehen.

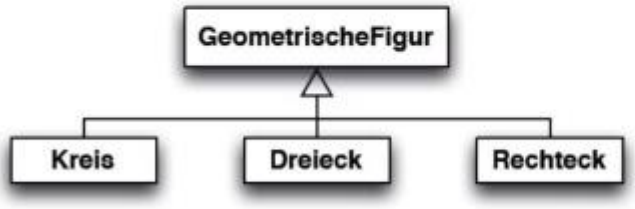


Abbildung 7 - Klassendiagramm mit Vererbungsbeziehungen

Beispiel 2.6

Abbildung 8 auf der nächsten Seite zeigt das Klassendiagramm einer Zoo-Implementierung. Ein Zoo besteht aus mehreren Käfigen. Käfige enthalten Tiere. Tiere werden in Unterklassen genauer spezifiziert. Die Klasse Tier ist abstract, von ihr dürfen keine Objekte erzeugt werden.

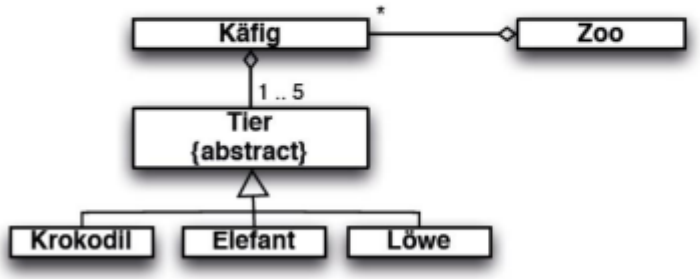


Abbildung 8 - Klassendiagramm mit Vererbungsbeziehungen und Assoziationen

Beispiel 2.7

Abbildung 9 zeigt das Klassendiagramm einer Klassenstruktur, welche die gemeinschaftliche Eigenschaft singend besitzt. Es ist leicht ersichtlich, dass die drei abgebildeten Klassen ansonsten keinerlei Gemeinsamkeiten besitzen und nicht durch eine Vererbungsbeziehung verbunden sein sollten. Das Interface Singend ist eine Eigenschaft und keine Variante einer Klasse von der Objekte erzeugt werden sollten.

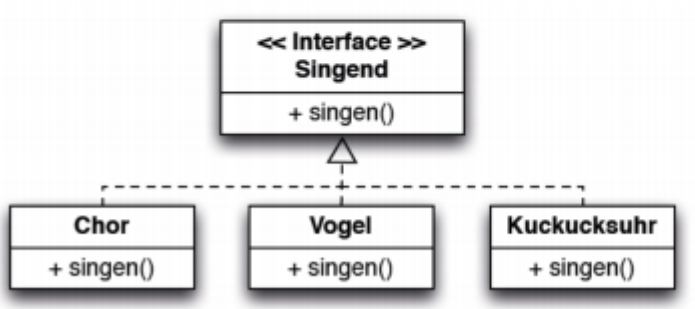


Abbildung 9 - Klassendiagramm mit Interface-Implementierung

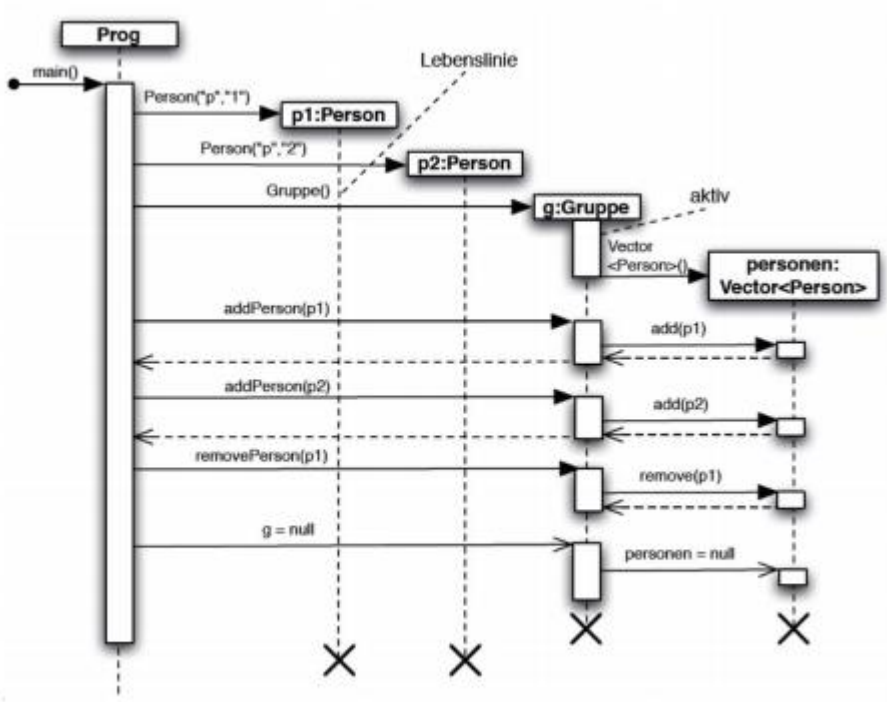


Abbildung 13 - Sequenzdiagramm zur Verwendung der Klasse Gruppe

Dokumentation wichtiger

Code Snippets

Eclipse and Javadoc with PDF Generation ~ l3oc
www.l3oc.com/.../**eclipse**-and-**javadoc**-with-pdf-**gen**... ▾ Diese Seite übersetzen
14.05.2015 - **Eclipse** and **Javadoc** PDF source-code documentation **generation**.

Eclipse and Javadoc with PDF Generation

Thursday, May 14, 2015 No comments

Beispiel-Quelltext [Bearbeiten | Quelltext bearbeiten]

```
1 /**
2  * Ein Hello-World-Programm in Java.
3  * Dies ist ein Javadoc-Kommentar.
4  *
5  * @author John Doe
6  * @version 1.0
7  */
8 public class Hello {
9     /**
10     * Hauptprogramm.
11     *
12     * @param args Kommandozeilenparameter
13     */
14    public static void main(String[] args) {
15        System.out.println("Hallo Welt!");
16    }
17 }
```

Writing Javadoc Comments in Eclipse - YouTube

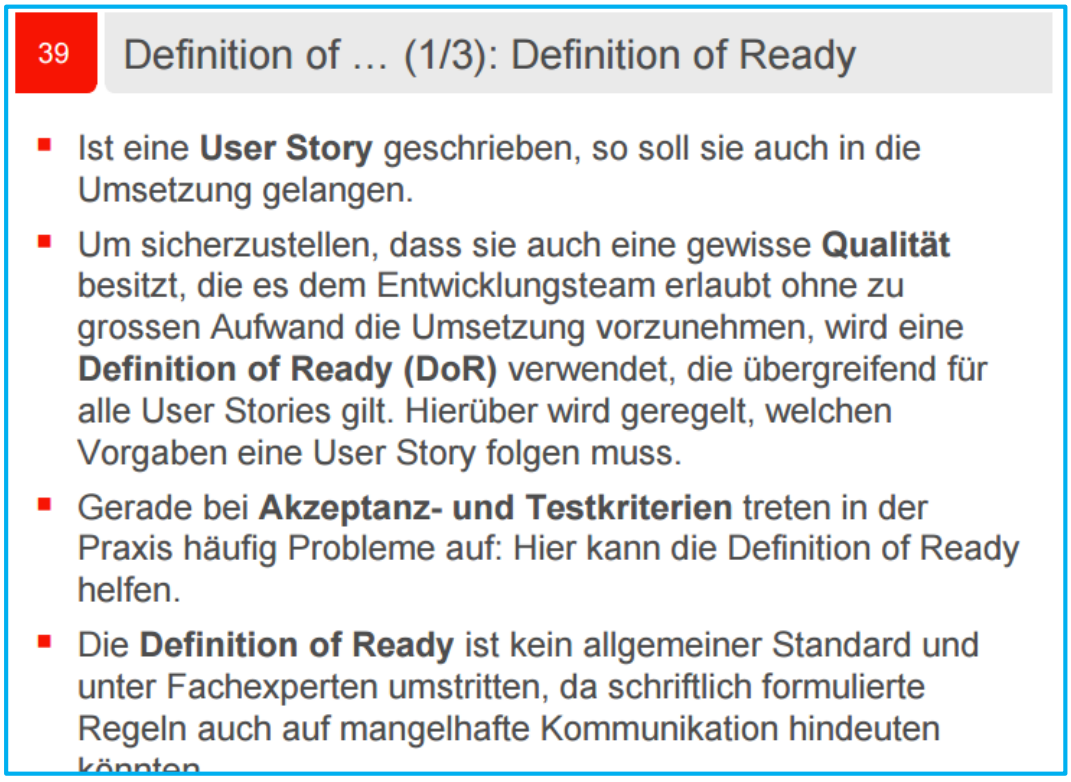
https://www.youtube.com/watch?v=6XoVf4x-tag ▾
12.01.2012 - Hochgeladen von Norm Krumpe
Covers the basics of writing **javadoc**-style comments, including some useful Eclipse features.

Herleitung der Testfälle aus den

Akzeptanzkriterien der User Stories

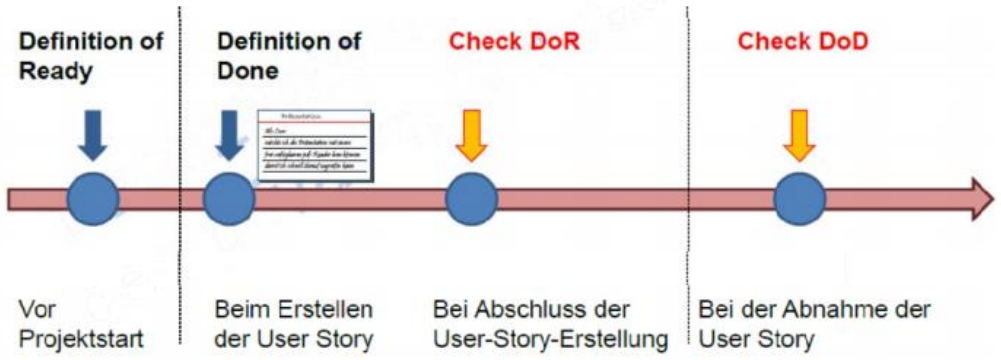
f) Testfälle vollständig?

f) Testfälle verständlich?



40 Definition of … (2/3): Definition of Done

- Die **Definition of Done (DoD)** beschreibt (im Vorhinein), wann eine **User Story** als fertig gilt, das heisst, vom Anwender abgenommen werden kann. Die (universelle) Überprüfung der DoD-Kriterien findet nach der vollständigen Erstellung der User Story statt. Üblicherweise wird die DoD über eine (Check-)Liste realisiert.



41 Definition of … (3/3): Vergleich

	Ready	Done
Kurzcharakterisierung	Qualitätskriterien für Anforderungen	Abnahmekriterien für User Stories
Bezug	Bezieht sich auf die Anforderungen	Bezieht sich auf die entwickelte Software
Wesentlich	Vollständigkeit der User Story / Story Card inkl. Priorisierung (und Schätzung)	Qualitätskriterien: Die entwickelte Software muss getestet sein und funktionieren
Ziel	„Abnahme“ einer User Story vor dem Sprint	„Abnahme“ der Umsetzung einer User Story
Wird wann erstellt?	Vor dem Projekt	Vor dem Projekt
Wird wann überprüft?	Nach der Erstellung der User Story vor Sprintbeginn	Während und nach der Umsetzung (Sprint Review) der User Story im Sprint
Wer ist verantwortlich?	Product Owner	Entwicklungsteam