Software-Qualitätssicherung VU
Block 6

Peter Frühwirt

Software Testen im agilen Umfeld

AGILE SOFTWAREENTWICKLUNG

Grundlagen der agilen SW-Entwicklung

Ein Tester in einem agilen Projekt arbeitet anders als ein Tester in einem traditionellen Projekt. Tester müssen die Werte und Prinzipien verstehen, die agile Projekte stützen. Sie müssen verstehen, dass Tester genauso wie Entwickler gleichberechtigte Mitglieder des Teams sind (Whole-Team Approach). Sie kommunizieren von Beginn an regelmäßig miteinander, was der frühzeitigen Fehlerreduzierung dient und hilft ein qualitativ hochwertiges Produkt zu liefern.

Agile Manifest (2001)

- Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge
- Funktionierende Software ist wichtiger als umfassende Dokumentation
- Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlungen
- Reagieren auf Veränderungen ist wichtiger als das Befolgen eines Plans

Prinzipien

- Es hat höchste Priorität, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufriedenzustellen.
- Anforderungsänderungen, selbst spät in der Entwicklung, werden begrüßt. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
- Funktionierende Software wird regelmäßig innerhalb weniger Wochen oder Monate geliefert, mit Präferenz für die kürzere Zeitspanne.
- Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
- Projekte werden rund um motivierte Individuen aufgebaut - ihnen wird das notwendige Umfeld und die Unterstützung gegeben, die sie benötigen und darauf vertraut, dass sie die Aufgabe erledigen.

Prinzipien

- Die effizienteste und effektivste Methode Informationen an und innerhalb eines Entwicklungsteams zu übermitteln ist im Gespräch von Angesicht zu Angesicht.
- Funktionierende Software ist das wichtigste Fortschrittsmaß.
- Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Anwender sollten ein gleichmäßiges Tempo dauerhaft halten können.
- Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
- Einfachheit -- die Kunst, die Menge nicht getaner Arbeit zu maximieren - ist essenziell.
- Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
- In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

Scrum

- Scrum gliedert ein Projekt in kurze Iterationen fester Länge. Eine solche Iteration heißt in Scrum „Sprint“. Sie dauert in der Regel zwei bis vier Wochen.
- Produkterweiterung (Inkrement): Jeder Sprint soll ein potenziell auslieferbares Produkt erzeugen, dessen Leistungsumfang mit jeder Iteration wächst.
- Product Backlog: Der Product Owner führt ein sog. Product Backlog. Es enthält eine priorisierte Auflistung der geplanten Produkt-Features. Das Product Backlog entwickelt und verändert sich über die Sprints hinweg. Dieses Arbeiten am Backlog wird auch „Backlog Refinement“ genannt.
- Sprint Backlog: Zu Beginn eines jeden Sprints zieht das Team diejenigen Anforderungen, die im priorisierten Product Backlog an der Spitze stehen und die es in diesem Sprint umsetzen will, aus dem Product Backlog in ein kleineres Sprint Backlog.

Scrum

- Definition of Done: Um abzusichern, dass am Sprint-Ende tatsächlich ein fertiges Produktinkrement vorliegen wird, formuliert das Team zu Beginn eines Inkrementes gemeinsam Kriterien, anhand derer es überprüfen und entscheiden kann, ob die Arbeit an dem Inkrement abgeschlossen ist.
- Timeboxing: Nur solche Aufgaben, Anforderungen oder Features, die das Team erwartungsgemäß innerhalb des Sprints fertigstellen kann, werden in das Sprint Backlog aufgenommen.
- Transparenz: Der Sprint-Status wird täglich (im Daily Scrum, der täglichen Statusrunde des Teams, „Stand-Up“) aktualisiert und abgebildet.

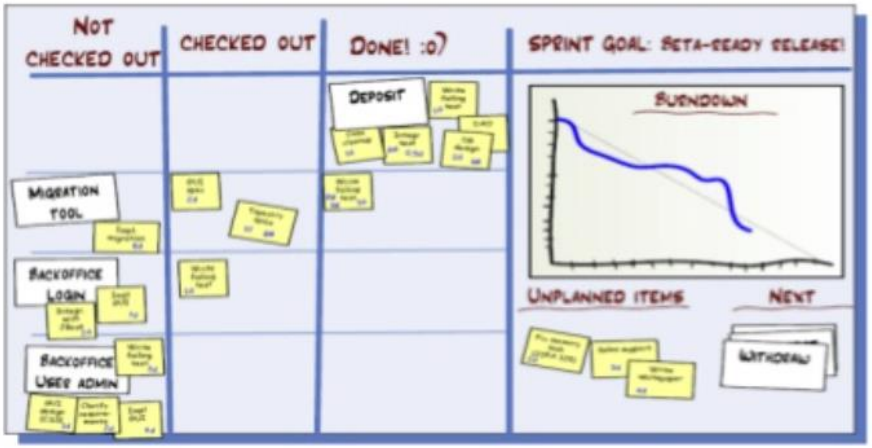
Kanban

Das allgemeine Ziel ist es, den Arbeitsfluss innerhalb einer Wertschöpfungskette abzubilden und zu optimieren. Kanban verwendet dazu drei Instrumente:

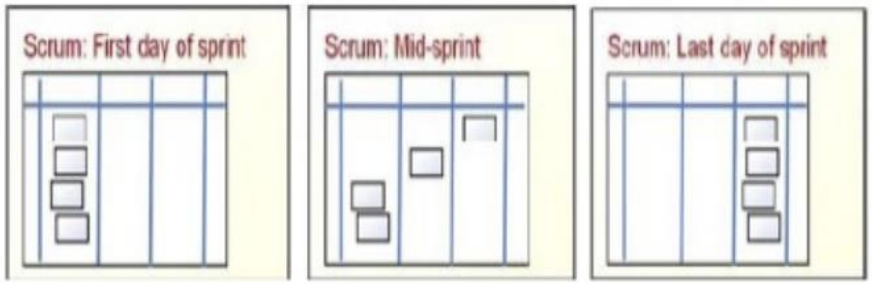
- Kanban Board: Die zu steuernde Wertschöpfungskette wird auf einem sogenannten Kanban- Board visualisiert.
- Work-In-Progress-Limit: Die Menge der gleichzeitig zu erledigenden Aufgaben (Work-in- Progress, WIP) wird limitiert. Dies geschieht durch Limits für die Anzahl der Tickets, die je Bearbeitungsstation und/oder im gesamten Board erlaubt sind.
- Lead Time: Kanban wird genutzt, um durch die Senkung der durchschnittlichen Bearbeitungszeit den kontinuierlichen Fluss von Aufgaben durch die gesamte Wertschöpfungskette zu optimieren.

Scrum vs. Kanban

Scrum Board

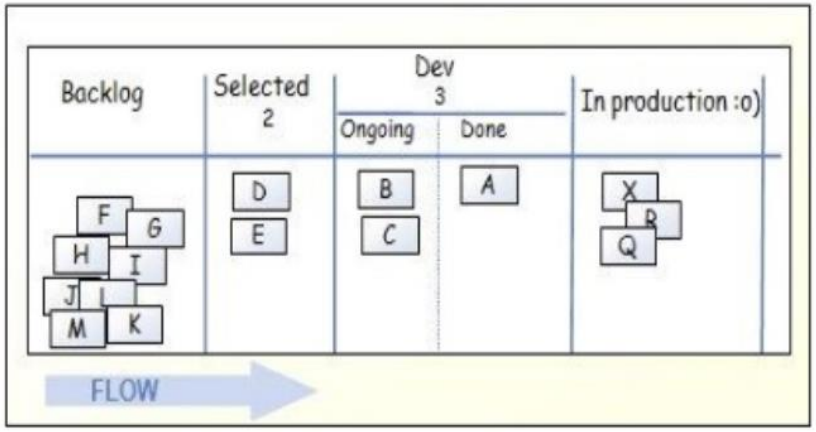


Scrum Board im Verlauf des Sprints

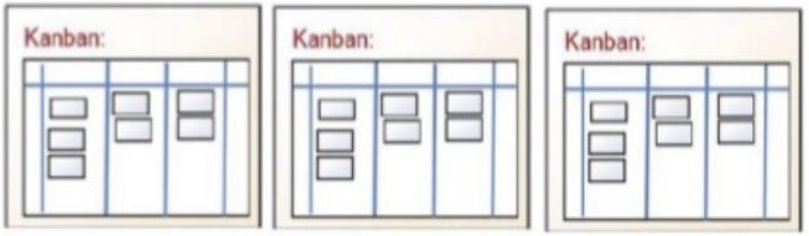


Scrum realisiert Stories in Sprints

Kanban Board



Kanban Board im Verlauf der Zeit

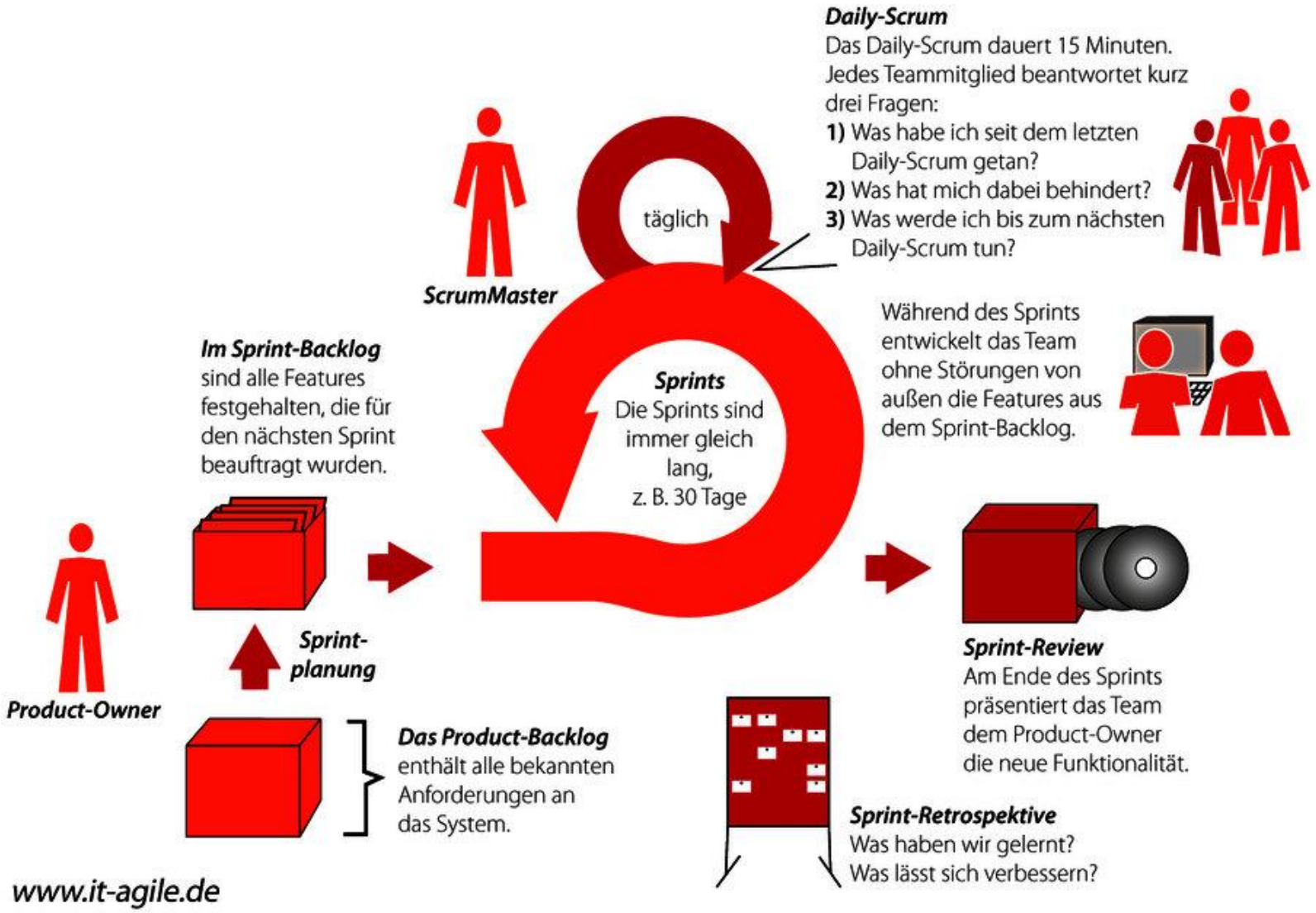


Kanban limitiert die Anzahl der Tasks pro Zustand

Rollen im Scrum Team

- Scrum Master: Er ist verantwortlich dafür, dass die Scrum Praktiken umgesetzt und verfolgt werden. Wenn sie verletzt werden, Personalfragen auftreten oder andere praktische Hindernisse (Impediments) auftreten, ist es Aufgabe des Scrum Master, dies abzustellen bzw. eine Lösung herbeizuführen. Der Scrum Master hat allerdings keine Teamleitungsfunktion, sondern er agiert als Coach.
- Product Owner: Der Product Owner ist die Person, die das Product Backlog verantwortet, führt und priorisiert. Er agiert gegenüber dem Team als Vertreter des oder der Kunden. Diese Person hat keine Teamleitungsfunktion, er verantwortet die Produkteigenschaften!
- Entwicklungsteam: Das Entwicklungsteam entwickelt und testet das Produkt. Es ist selbstorganisiert: Es gibt keine Teamleitung, das Team als Ganzes trifft alle Entscheidungen. Das Team arbeitet funktionsübergreifend zusammen.

Scrum Prozess



Tester im Scrum Team?

- Der Scrum Prozess sieht keine Tester vor.
- Testen ist Teil der Entwicklung
  - Unit Tests (!)
  - Test Driven Development
- Testen geschieht durch den Product Owner / Kunden
  - Jeden Sprint
  - Akzeptanzkriterien

- Braucht man überhaupt Tester noch?

Tester im Scrum Team

- Führen Entwickler auch Akzeptanztests durch?
- Werden Entwickler ihre Arbeit auch gut testen?
- Ist es genug nur Unit-Tests zu haben?
- Wie werden nicht-funktionale Anforderungen getestet?

- Vorteile von eigenen Testern
  - Fokus auf Kundensicht anstelle der technischen Implementierung
  - Fokus auf das Aufdecken von Fehlern anstelle der Verifikation der Vollständigkeit

Aufgaben eines Testers

- Statische Tests der Anforderungen / Dokumentation
- Dynamisches Testen (Ausführung von Akzeptanztests)
- Erstellung von Akzeptanztests
- Sicherstellung der Verwaltung von Testfällen im SCM
- Unterstützung bei Unit Tests
- Unterstützung bei test-getriebenem Design
- Test Ansatz
- Test Planung
- Regressionstests (manuell & automatisiert)
- Input während des Meetings (Planung, Sprint Review, etc.)
- Risikoanalyse und –abschätzung
- Schätzung der User Stories
- Demonstration und Unterstützung während des Sprint Reviews
- Tracability von Tests zu User Stories
- Erstellung und Wartung von automatisierten Tests
- Unterstützung bei Akzeptanztests

Spezialisierte Testgruppen

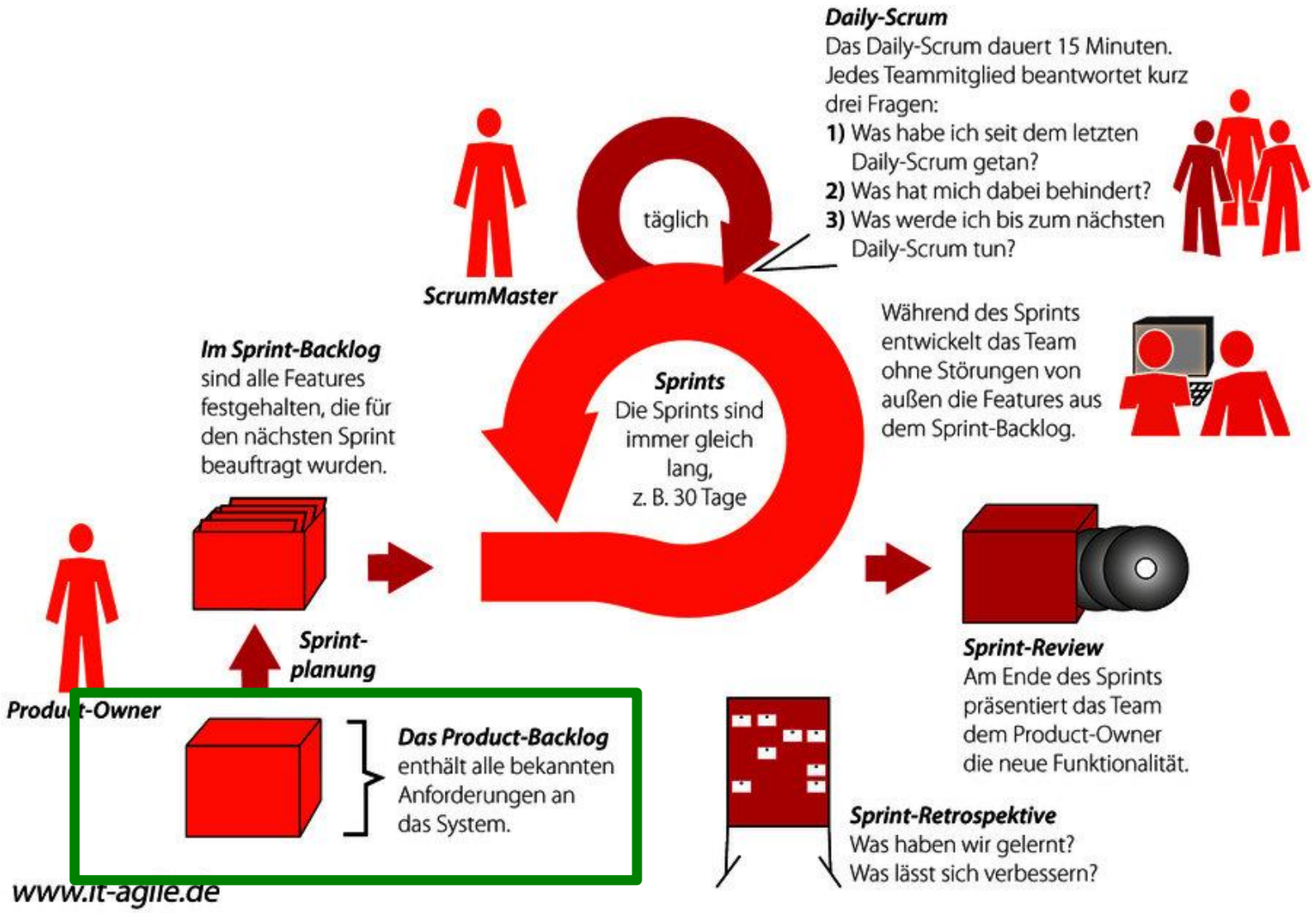
- Es gibt eine Vielzahl an spezifischen Testrollen, welche unter Umständen außerhalb des Teams angesiedelt sind, aber von der Entwicklung abhängig sind. Diese sind möglicherweise Vollzeit im Team integriert oder werden bei kritischen Punkten zugezogen.

- Folgende Spezialgebiete fallen z.B. darunter:
  - Security
  - Performance und Stresstests
  - Usability
  - Portability
  - Wartbarkeit

Aufgaben eines Test Manager

- Mentoring
- Personalmanagement
- Aufteilung der Testressourcen über Teams
- Wissenstransfer
- Training und Entwicklungsplanung
- Test Strategie / Ansatz
- Test Planung
- Identifikation, Analyse und Verhinderung von Risiken
- Bindeglied zwischen Business und Stakeholdern
- ...

Scrum Prozess



User Stories

- Die schlechte Qualität von Spezifikationen ist oft ein Grund für das Fehlschlagen eines Projektes. Ursachen können der fehlende überblick des Kunden über seine tatsächlichen Bedürfnisse, das Fehlen einer globalen Vision für das System, redundante oder sich widersprechende Anforderungen oder andere Fehler in der Kommunikation sein.
- Um solche Fehlerquellen zu vermeiden, werden in agilen Entwicklungen User-Stories eingesetzt. Diese beschreiben die Anforderungen aus Sicht der Fachbereichsvertreter, aber auch der Entwickler und Tester und werden von diesen gemeinsam verfasst.
- Die User-Stories müssen sowohl funktionale als auch nicht-funktionale Eigenschaften behandeln. Jede Story soll die Abnahmekriterien für diese Eigenschaften enthalten.

3 Elemente einer User Story Card

- Card: Die Karte (Card) ist das physische Medium, das die User-Story beschreibt. Hier werden die Anforderung, ihre Dringlichkeit, die erwartete Dauer für Entwicklung und Test sowie die Abnahmekriterien für diese Story identifiziert. Die Beschreibung muss genau sein, da sie im Product Backlog verwendet wird.
- Conversation: Die Diskussion (Conversation) erklärt, wie die Software genutzt werden wird.
- Confirmation: Die Abnahmekriterien, die in der Diskussion festgelegt werden, werden verwendet, um den Abschluss einer User-Story bestätigen (confirm) zu können.

User Story Konstrukt

- **As a [user role], I want to [goal], so that I can [reason].**

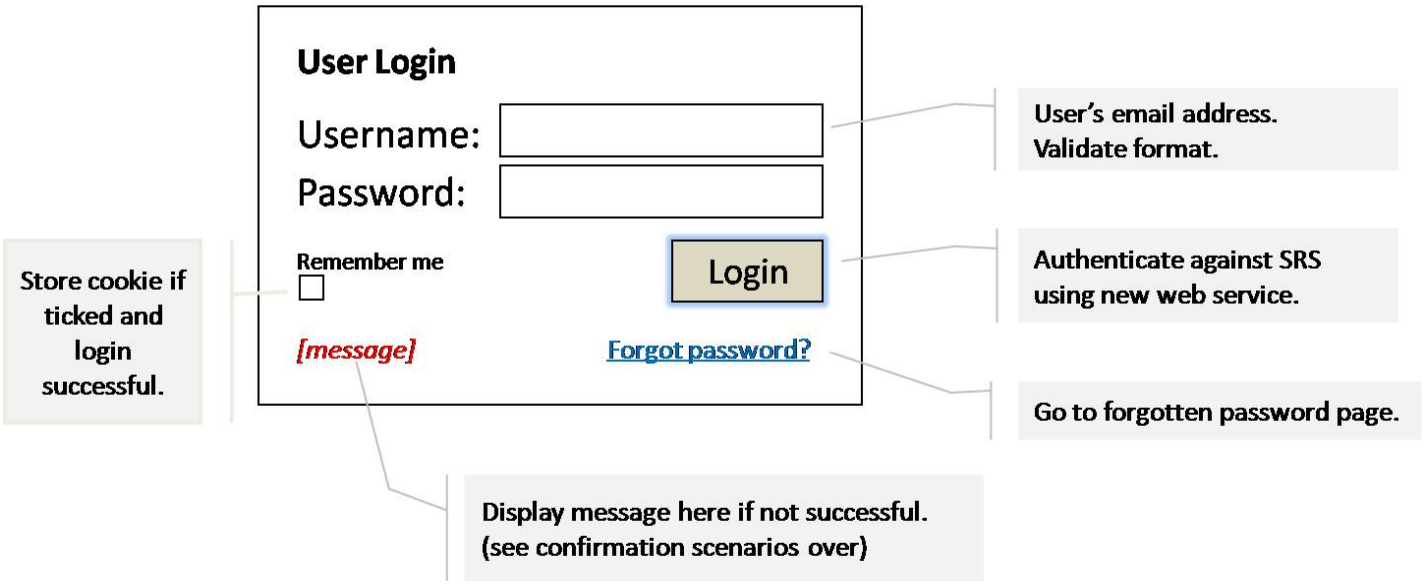
- Wer? (user role)
- Was? (goal)
- Warum? (reason)

- Gibt Aufschluss warum das Feature benötigt wird
- Kann beinflussen, wie ein Feature funktionieren soll
- Kann eine Ideen für andere nützliche Funktionen geben

Story Card | Beispiel

#0001	USER LOGIN	Fibonacci Size # 3
As a **[registered user]**, I want to **[log in]**, so I can **[access subscriber content]**.		

For new features, annotated wireframe. For bugs, steps to reproduce with screenshot. For non-functional stories, explain scope/standards.



Further information is attached to this story on VSTS Product Backlog.

http://www.allaboutagile.com/user-story-example/

Story Card | Beispiel

Confirmation

1. Success – valid user logged in and referred to home page.
   a. 'Remember me' ticked – store cookie / automatic login next time.
   b. 'Remember me' not ticked – force login next time.

2. Failure – display message:
   a) "Email address in wrong format"
   b) "Unrecognised user name, please try again"
   c) "Incorrect password, please try again"
   d) "Service unavailable, please try again"
   e) Account has expired – refer to account renewal sales page.

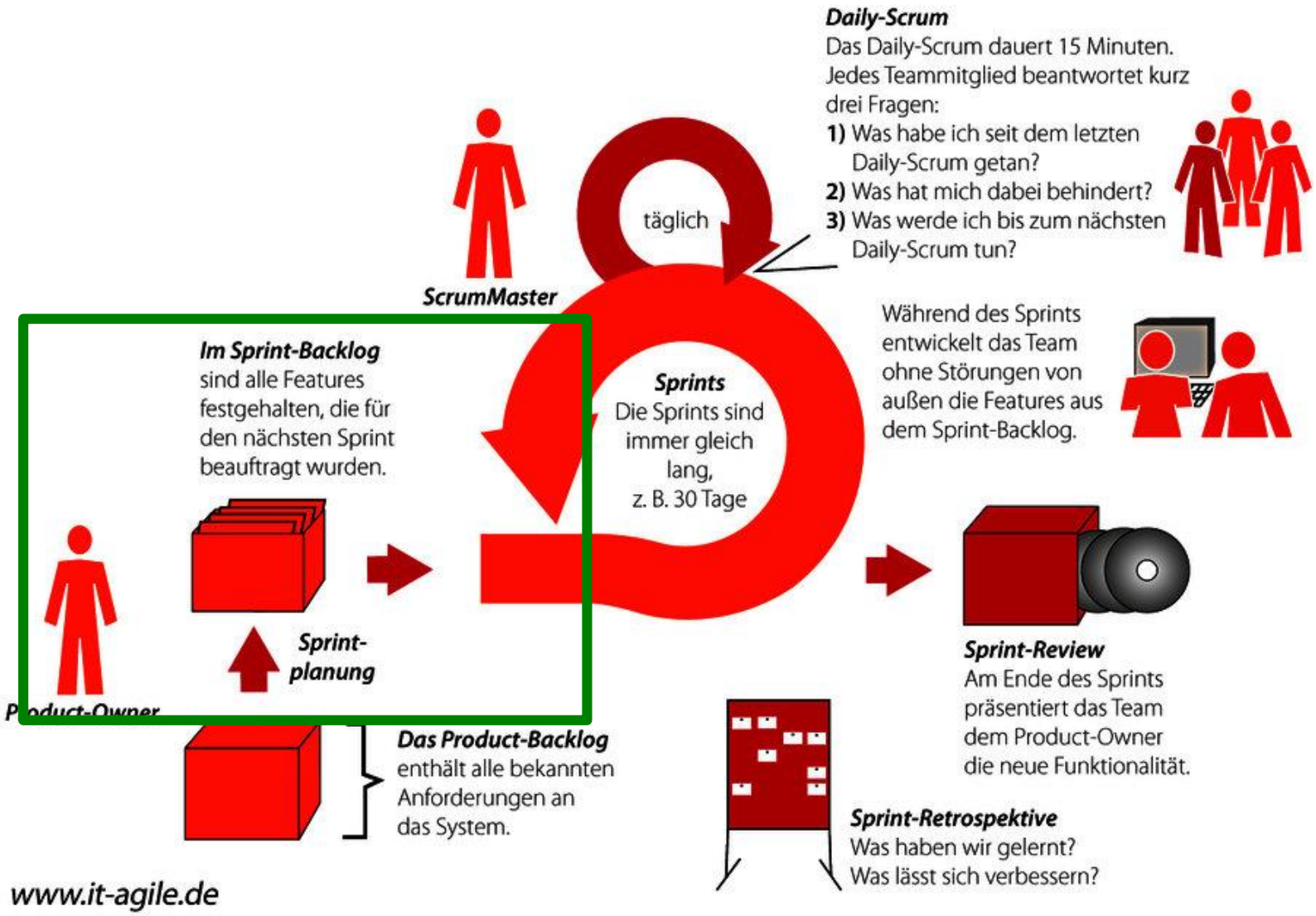
http://www.allaboutagile.com/user-story-example/

Kriterien zum Review von Anforderungen

- Vollständigkeit
- Konsistenz
- Unmissverständlichkeit
- Realisierbarkeit
- Testbarkeit
- Tracability
- Spezifisch
- Messbar
- Durchführbarkeit innerhalb eines Sprints
- Unabhängigkeit

„Definition of Ready“

Scrum Prozess

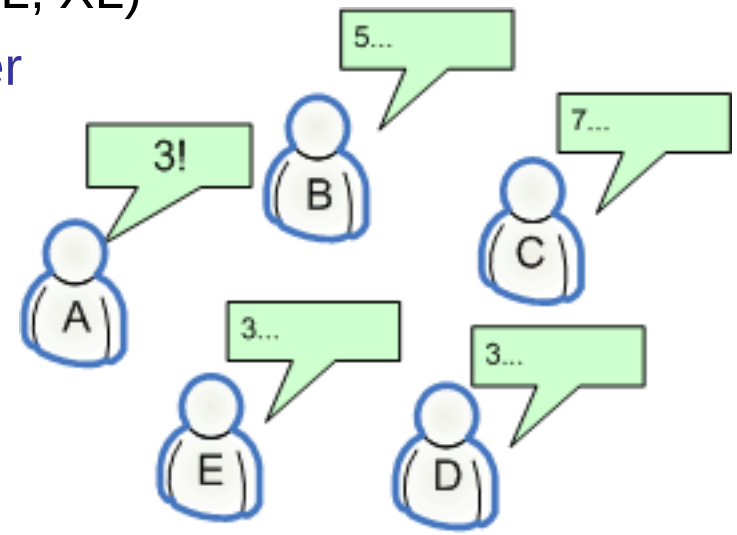


Iteration Zero

- Identifikation des Projektumfangs (d.h. Anfertigung des Product Backlogs)
- Erstellen einer initialen Systemarchitektur und ggf. erster Prototypen
- Planung, Erwerb und Installation der notwendigen Werkzeuge (z.B. für das Testmanagement, Fehlermanagement, die Testautomatisierung und die continuous Integration)
- Erstellen einer initialen Teststrategie für alle Teststufen, die (unter anderem) auf folgendes abzielen: Testumfang, technische Risiken, Testarten und überdeckungsziele
- Durchführung einer initialen Qualitätsrisikoanalyse
- Definition von Metriken zur Messung des Testfortschritts und zur Produktqualität
- Spezifikation der Definition of Done
- Festlegen der Struktur des Task Boards und initiales "befüllen" des Boards
- Definition des Zeitpunktes zu dem die Tests beendet werden sollen, bevor das System an den Kunden geliefert wird

Estimation / Aufwandsschätzung

- Abschätzung des Aufwands (Komplexität / Aufwand / Schwierigkeitsgrad)
- Möglichst keine Trennung zwischen Entwicklungs- und Testaufwänden (whole-team approach!)
- Normalweise wird der Aufwand / Größe nicht in Stunden/Tagen sondern in Story-Points geschätzt (1, 2, 3, 5, 8, 13 ...)
- Manchmal auch T-Shirt Größen (S, M, L, XL)
- Schätzung erfolgt durch Planning Poker



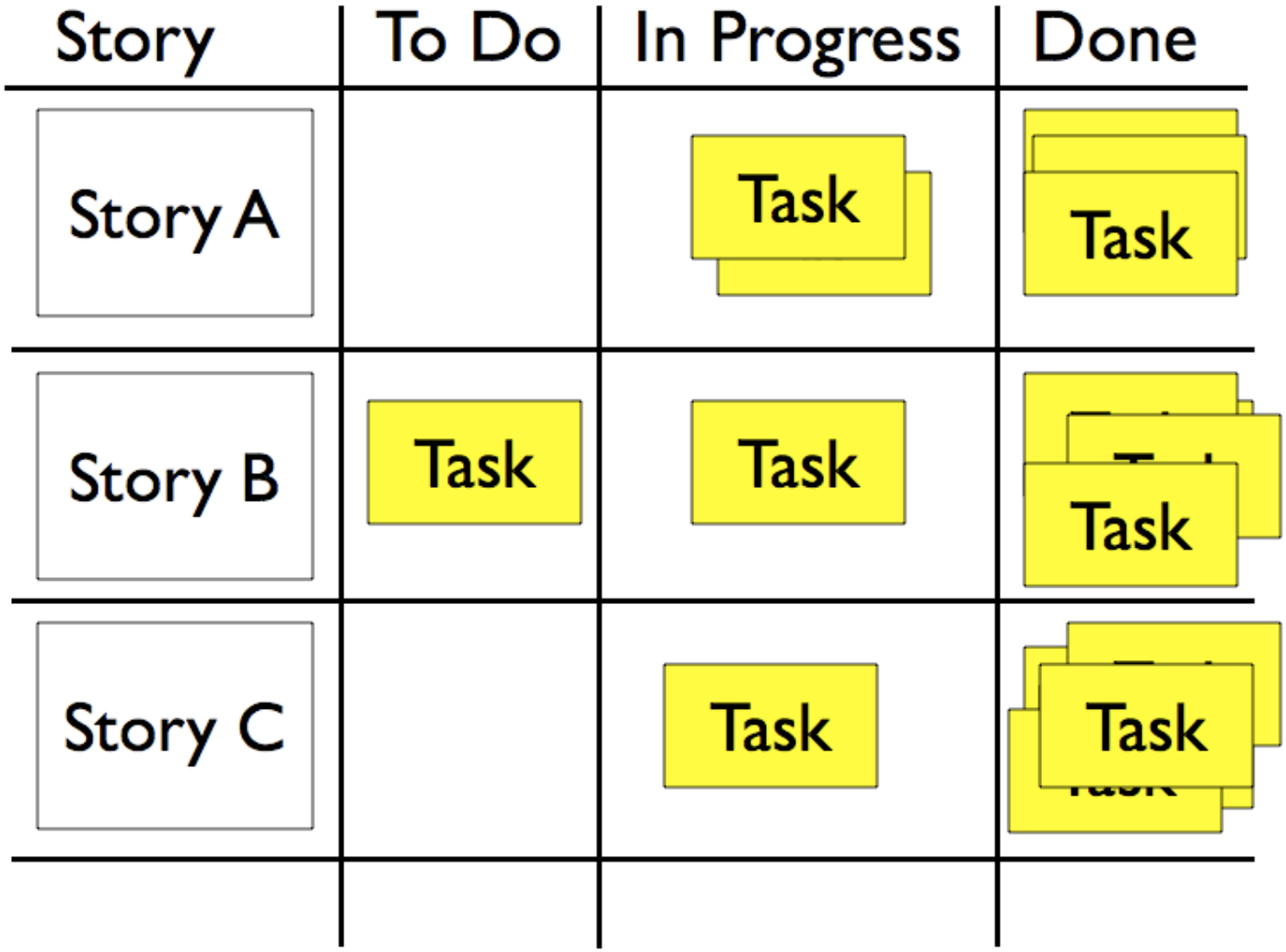
Iterationsplanung

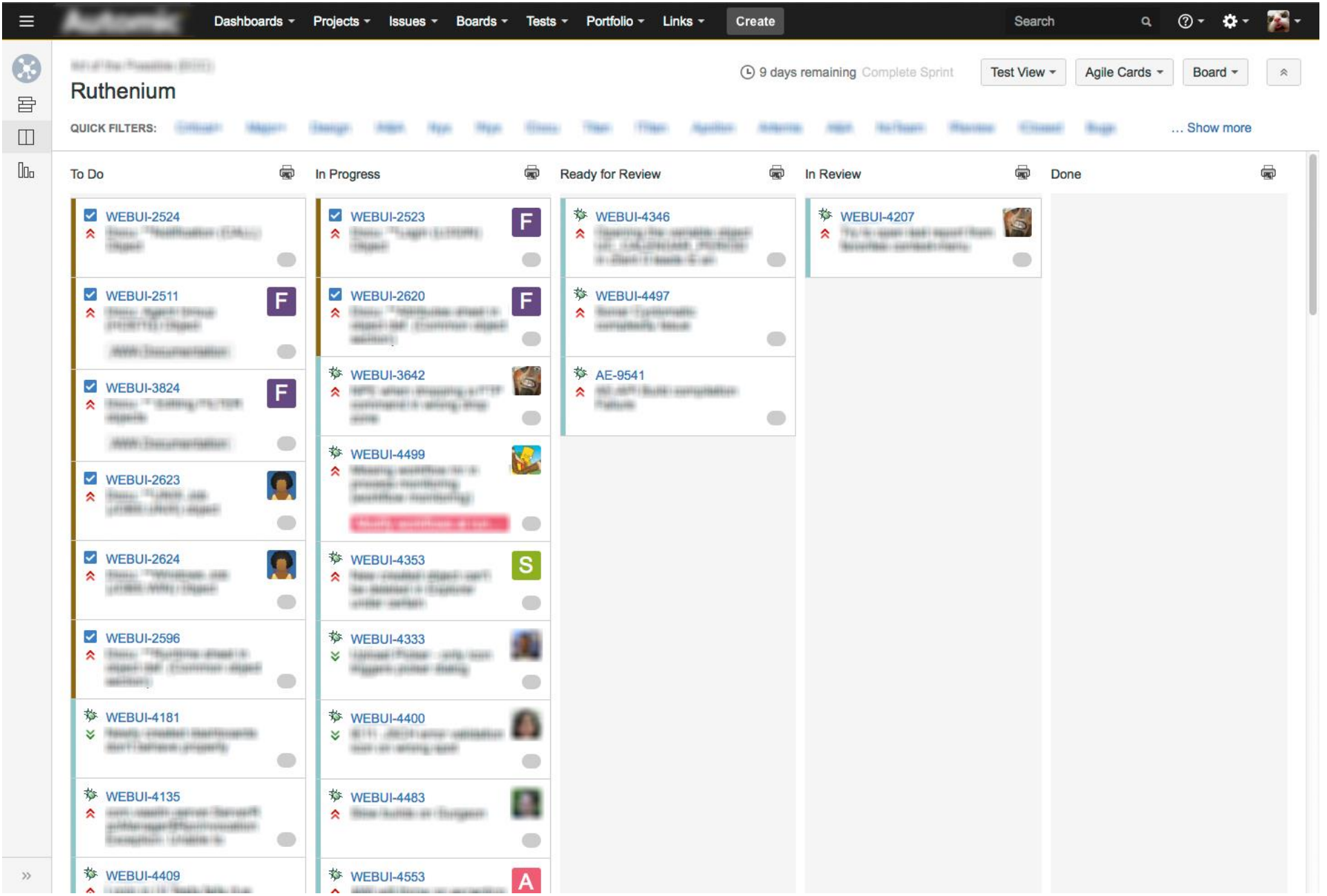
- Meeting des kompletten Team
- Priorisierung der offenen User Stories im Product Backlog
- Aufwandschätzungen (sofern nicht bereits erfolgt)
- Diskussion über Scope des Sprints
- Team-Commitment zum Scope

Mehrwert von Testern während der Iterationsplanung

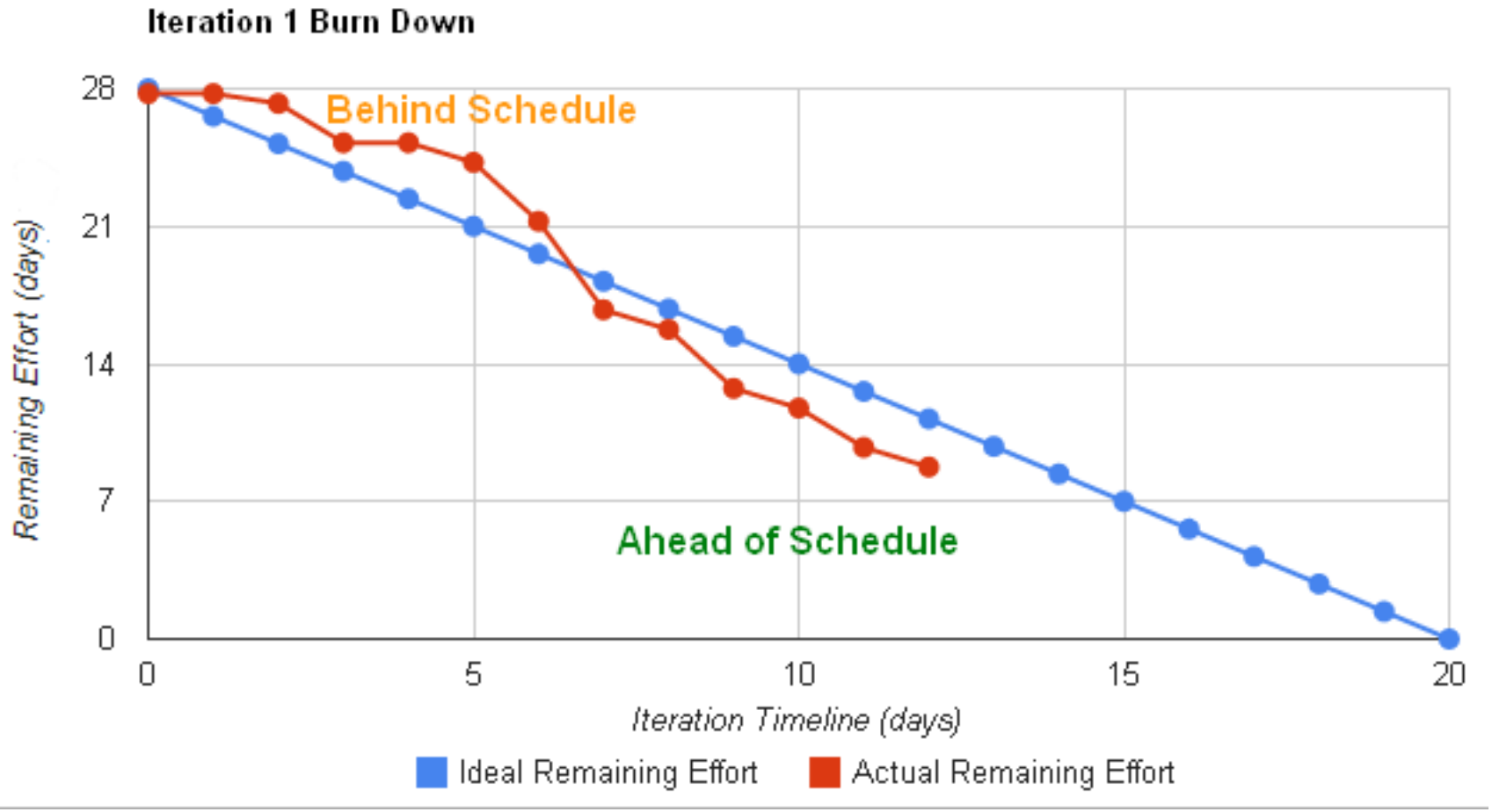
- Teilnehmen an der detaillierten Risikoanalyse der User-Stories
- Festlegen der Testbarkeit der User-Stories
- Erstellen der Abnahmetests für die User-Stories
- Herunterbrechen der User-Stories in Aufgaben (insbesondere Testaufgaben)
- Schätzen des Testaufwands für alle Testaufgaben
- Identifizieren funktionaler und nicht-funktionaler Eigenschaften des zu testenden Systems
- Unterstützen von und Mitarbeit an der Testautomatisierung

Task Board


Story
To Do
In Progress
Done
Story A
Task
Task
Story B
Task
Task
Task
Story C
Task
Task


Dashboards
Projects
Issues
Boards
Tests
Portfolio
Links
Create
Search
Ruthenium
9 days remaining
Complete Sprint
Test View
Agile Cards
Board
QUICK FILTERS:
... Show more
To Do
In Progress
Ready for Review
In Review
Done
WEBUI-2524
WEBUI-2511
WEBUI-3824
WEBUI-2623
WEBUI-2624
WEBUI-2596
WEBUI-4181
WEBUI-4135
WEBUI-4409
WEBUI-2523
WEBUI-2620
WEBUI-3642
WEBUI-4499
WEBUI-4353
WEBUI-4333
WEBUI-4400
WEBUI-4483
WEBUI-4553
WEBUI-4346
WEBUI-4497
AE-9541
WEBUI-4207

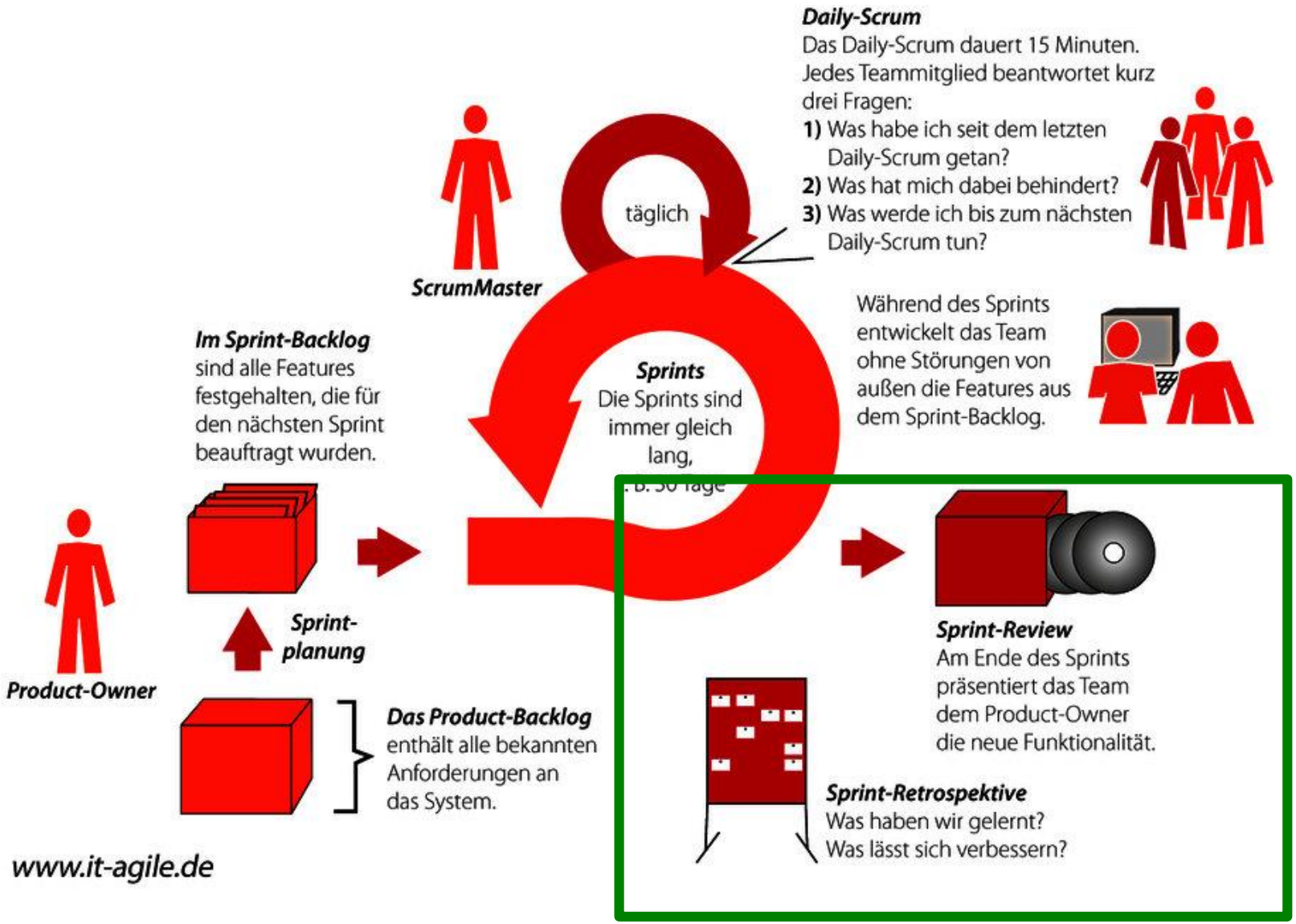
Burndown Chart



Burndown Chart Analyse

- Burndown Chart eine Advanced Software Engineering Gruppe

Scrum Prozess



Sprint Ende

- Sprint Review
  - Demonstration der vollständigen Stories gegenüber dem Stakeholders und Product Owner
  - Formale Abnahme der User Story
  - Unvollständige User Stories werden nicht präsentiert und wandern in den nächsten Sprint

- Retrospektive
  - In der agilen Entwicklung bezeichnet eine Retrospektive eine Team-Sitzung am Ende jeder Iteration, in der besprochen wird, was erfolgreich war, was verbessert werden könnte und wie die Verbesserungen in künftigen Iterationen umgesetzt werden können. Dabei wird auch ein Augenmerk auf die Erhaltung gut funktionierender Praktiken gesetzt. Retrospektiven decken Themen wie den Prozess, die beteiligten Personen, Organisationen und Beziehungen sowie die Werkzeuge ab.

Continuous Integration

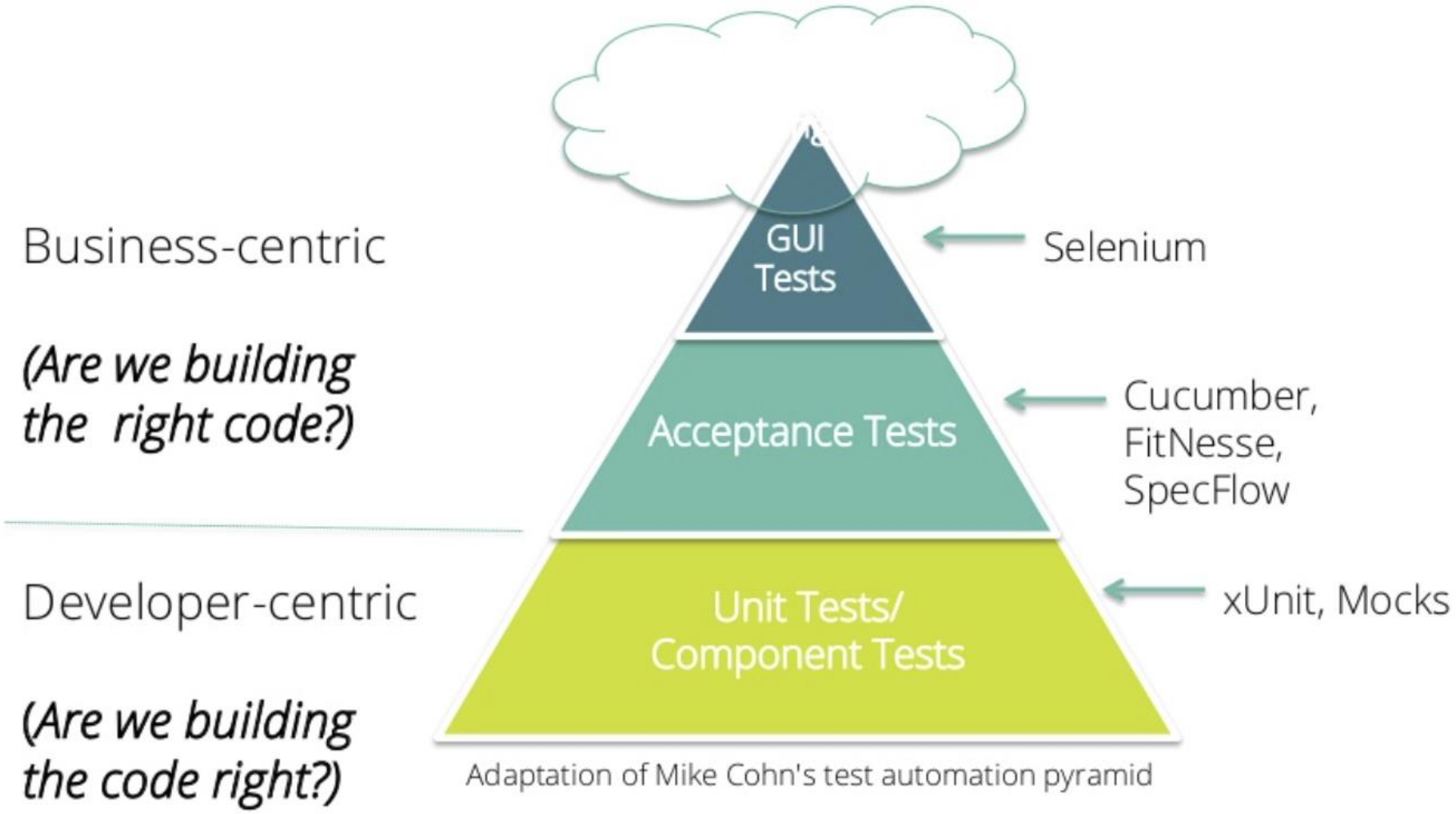
- Die Auslieferung einer Produkterweiterung (Inkrement) setzt voraus, dass am Ende einer jeden Iteration bzw. eines jeden Sprints eine funktionierende Software vorliegt.
- Lösung: Alle geänderten Softwarekomponenten werden regelmäßig (mind. 1x pro Tag) zusammengeführt: Konfigurationsmanagement, Kompilierung, Buildprozess, Verteilung in die Zielumgebung und die Ausführung der Tests sind in einem automatisierten, wiederholbaren Prozess zusammengefasst.

Continuous Integration

- Der Prozess der CI besteht aus folgenden automatisierten Aktivitäten:
  - Statische Codeanalyse: Durchführung einer statischen Codeanalyse und Aufzeichnung der Ergebnisse,
  - Kompilieren: Kompilieren und Linken des Codes, Erstellung der ausführbaren Dateien,
  - Unittest: Durchführen der Unittests, Prüfung der Codeabdeckung und Aufzeichnung der Testergebnisse,
  - Bereitstellung (Deployment): Installieren der Software in eine Testumgebung,
  - Integrations- und Systemtests: Durchführung der entsprechenden Tests und Aufzeichnung der Ergebnisse,
  - Bericht (Dashboard): Veröffentlichung des Status all dieser Aktivitäten an einem öffentlich sichtbaren Ort.

METHODEN, TECHNIKEN UND WERKZEUGE

Test Automatisierungspyramide



http://www.slideshare.net/RajIndugula/agile-testing-practices-38015016

Agile Testmethoden

- Testgetriebene Entwicklung (TDD)
  - Vorgehensweise: Testfall erstellen und automatisieren, Test ausführen (schlägt fehl), Erstellen des eigentlichen Codes, Code solange verbessern bis Testfälle erfolgreich sind, ggf. Refactoring
  - Überwiegend auf Unit-Test Ebene
  - Die Entwickler konzentrieren sich auf klar definierte, erwartete Ergebnisse. Die Tests sind automatisiert und werden in der continuous Integration verwendet.

Agile Testmethoden

- Abnahmetestgetriebene Entwicklung
  - Abnahmetestgetriebene Entwicklung definiert Abnahmekriterien und Tests während der Entwicklung von User-Stories.
  - Abnahmetestgetriebene Entwicklung ist ein kollaborativer Ansatz, der es jedem Stakeholder ermöglicht, zu verstehen, wie die Softwarekomponente sich verhalten soll und was Entwickler, Tester und Fachbereichsvertreter tun müssen, um dieses Verhalten zu erreichen.
  - In der abnahmetestgetriebenen Entwicklung werden wiederverwendbare Tests für die Regressionstests erstellt.

Agile Testmethoden

- Verhaltensgetriebene Entwicklung
  - Verhaltensgetriebene Entwicklung (Behaviour Driven Development) ermöglicht es dem Entwickler, sich darauf zu konzentrieren, den Code auf das erwartete Verhalten der Software hin zu testen. Da die Tests auf dem erwarteten Verhalten der Software basieren, sind sie in der Regel für andere Teammitglieder und Stakeholder leichter zu verstehen.
  - Spezifische Frameworks für BDD erlauben die Definition der Abnahmekriterien auf Basis des „Gegeben/Wenn/Dann-Formats“ (sog. Gherkin-Format):
    Gegeben ein Einstiegskontext,
    Wenn ein Ereignis auftritt,
    Dann werden bestimmte Wirkungen sichergestellt.
  - Aus diesen Definitionen erstellt das Framework ausführbaren Testcode.

Agile Testpraktiken

- Pairing: Zwei Teammitglieder (z. B. ein Tester und ein Entwickler, zwei Tester oder ein Tester und ein Product Owner) setzen sich zusammen, um gemeinsam eine Test- oder eine andere Sprintaufgabe zu bearbeiten.
- Inkrementelles Test Design: Testfälle und Chartas werden schrittweise aus User- Stories und anderen Testgrundlagen aufgebaut. Dabei wird mit einfachen Tests begonnen und man geht dann über zu komplexeren Tests.
- Mind Mapping: Mind Mapping ist ein nützliches Werkzeug für das Testen. Beispielsweise können Tester Mind Mapping nutzen, um zu bestimmen, welche Testsitzungen durchzuführen sind, um Teststrategien zu demonstrieren und um Testdaten zu beschreiben.

Abnahmekriterien

- In agilen Projekten werden die Anfangsanforderungen zu Beginn des Projekts als User-Stories in einem priorisierten Backlog niedergeschrieben.
- In jeder Iteration erstellen Entwickler Code, mit dem die in der User-Story beschriebenen Funktionen und Features mit den relevanten Qualitätsmerkmalen implementiert werden. Dieser Code wird durch Abnahmetests verifiziert und validiert.
- Abnahmekriterien / Themen bei der Abnahme:
  - Funktionales Verhalten
  - Qualitätsmerkmale
  - Szenarios (Use Cases)
  - Geschäftsprozessregeln
  - Externe Schnittstellen
  - Einschränkungen
  - Datendefinitionen

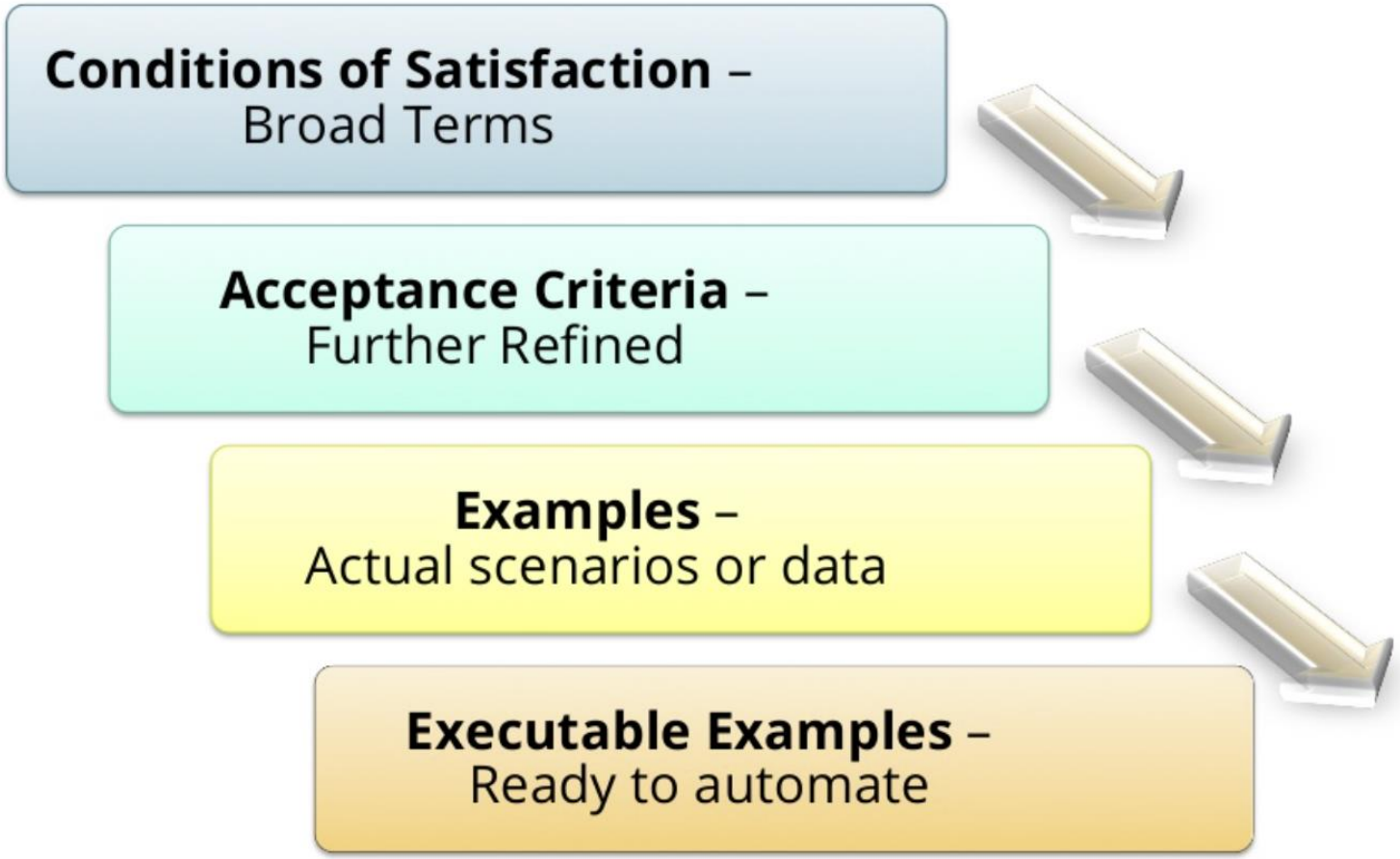
Abnahme / Definition of Done

- User Story
  - Die für die Iteration ausgewählten User-Stories sind vollständig, vom Team verstanden und haben detaillierte, testbare Abnahmekriterien.
  - Alle Elemente der User-Story sind spezifiziert und einem Review unterzogen worden.
  - Die Abnahmetests der User-Stories sind (als Testbeschreibung und/oder Testscript)
  - bereitgestellt.
  - Die notwendigen Aufgaben für die Implementierung und das Testen der ausgewählten User-
  - Stories sind identifiziert und vom Team geschätzt worden.

Abnahme / Definition of Done

- Iteration
  - Alle Features der Iteration sind fertig entwickelt und gemäß der Feature-Level Kriterien individuell getestet.
  - Jegliche nicht-kritischen Fehler, die während einer Iteration nicht behoben wurden, sind dem Product Backlog hinzugefügt und priorisiert worden.
  - Die Integration aller Features der Iteration ist erfolgt und getestet.
  - Die Dokumentation ist geschrieben und nach dem Review abgenommen.

Testen der Akzeptanzkriterien



http://www.slideshare.net/RajIndugula/agile-testing-practices-38015016

Akzeptanzkriterien | Beispiel

- Test einer *Registrieren Funktion*

- Welche Bedingungen / Anforderungen gibt es? (Conditions of Statisfaction)
  - Product Owner und Stakeholder haben sich geeinigt auf:
  - „Passwörter dürfen nicht einfach knackbar sein.“

Akzeptanzkriterien | Beispiel

- Product Owner und Tester erarbeiten zusammen Akzeptanzkriterien (Acceptance Criteria)
- Definition für „nicht einfach knackbar“
  - Mind. 8 Zeichen, aber nicht mehr als 12
  - Darf nur alphanumerische Zeichen beinhalten
  - Muss mind. eine Zahl beinhalten
  - Muss mind. einen Buchstaben beinhalten

Akzeptanzkriterien | Beispiel

- Anschließend werden Beispiel für Testdaten erarbeitet
- Bieten optimale Diskussionsgrundlage

Password	Expected	Expected Message	Comment
abc123	Invalid	You password must be at least 8 characters long, and no more than 12 characters long.	
abcdefghi	Invalid	Your password must contain at least one character and one number.	
1aaaaaaaaa	Valid		**Why valid?**
ajx972dab	Valid		

Akzeptanzkriterien | Beispiel

- Erstellung von ausführbaren Beispielen / Testdaten (Wiederholbarkeit!) z.B. durch verhaltensgetriebene Entwicklung

Given the "Unregistered User" user has navigated to the "register" page

When entering "newuser" in the "Username" field
And entering "abc123" in the "Password" field
And entering "abc123" in the "Confirm Password" field
And pressing the "Register" button

Then the text "Thank you for Registering" should appear on page
And the URL should end with "use/accountPage"

Black-Box Test Design

In agilen Projekten werden viele Tests von den Testern entwickelt während die Entwickler zeitgleich programmieren. So wie die Entwickler auf Grundlage der User-Stories und Abnahmekriterien entwickeln, so erstellen die Tester auch die Tests auf Grundlage der User- Stories und ihrer Abnahmekriterien. Tester können für die Erstellung dieser Tests traditionelle Black-Box Testentwurfsverfahren anwenden, wie Äquivalenzklassenbildung, Grenzwertanalyse, Entscheidungstabellen und zustandsbasierte Tests. Beispielsweise könnte die Grenzwertanalyse verwendet werden, um Testwerte auszuwählen.

Exploratives Testen

- Exploratives Testen ist in agilen Projekten wichtig wegen der begrenzten Zeit, die für die Testanalyse zur Verfügung steht und der begrenzten Detailgenauigkeit der User-Stories.
- Vorgehensweisen:
  - Erfahrungsbasiertes Verfahren
  - Riskobasiertes Testen
  - Anforderungsbasiertes Testen
  - Modellbasiertes Testen
  - Regessionsvermeidendes Testen
- Testsitzungen können folgendes beinhalten:
  - Überblickssitzung (um zu lernen, wie es funktioniert)
  - Analysesitzung (Bewertung der Funktionalität oder Eigenschaften)
  - Genaue Überdeckung (Ausnahmefälle, Szenarien, Interaktionen)

Werkzeuge

- Aufgabenmanagement- und Nachverfolgungswerkzeuge
  - Aufzeichnung der Stories
  - Erfassen von Aufwandsschätzungen
  - Verbindung von Entwicklungs- und Testaufgaben
  - Erfassen von Aufgabenstatus
- Kommunikations- und Informationsweitergabe-Werkzeuge
  - Email, Dokumenten, verbale Kommunikation
  - Wikis, Instant Messangers, Desktop Sharing
- Build und Distribution
- Konfigurationsmanagement
  - Speicherung von Quellcode, automatisierten Testskripten, manueller Tests, Arbeitsergebnisse, etc.
  - Versionsverwaltung (VCS)
- Werkzeuge für Testentwurf, Implementierung und Durchführung

Werkzeuge für Testentwurf, Implementierung und Durchführung

- Werkzeuge für den Testentwurf, z.B. Mind Maps
- Testfallmanagement-Werkzeuge
- Werkzeuge zur Testdatenvorbereitung und –Erstellung, z.B.: Werkzeuge, die Daten generieren, um die Datenbank einer Anwendung zu füllen
- Testdatenladewerkzeuge: Manuelle Dateneingabe ist oft zeitaufwändig und fehleranfällig, aber es gibt Datenladewerkzeuge, die den Prozess verlässlich und effizient machen.
- Automatisierte Testdurchführungswerkzeuge
- Werkzeuge für exploratives Testen, z.B. Aufzeichnungen von explorativen Testsitzungen

ZUSAMMENFASSUNG

Wichtige Begriffe

- Sprint / Iteration
- User Story
- Akzeptanzkriterien
- Estimation / Story Points
- Velocity
- Burndown Chart
- Definition of Done
- Task Board
- Daily Standup
- Iteration Planning
- Retrospektive
- Continuous Integration (CI)

Tester Aktivitäten

- Statische Tests der Anforderungen / Dokumentation
- Erstellung von Akzeptanztests
- Dynamisches Testen (Ausführung von Akzeptanztests)
- Unterstützung bei Unit Tests
- Unterstützung bei test-getriebenem Design
- Regressionstests (manuell & automatisiert)
- Risikoanalyse und –abschätzung
- Schätzung der User Stories
- Erstellung und Wartung von automatisierten Tests

Agile Testmethoden

- Testgetriebene Entwicklung
- Abnahmetestgetriebene Entwicklung
- Verhaltensgetriebene Entwicklung
- Abnahmekriterien
- Funktionales und nicht-funktionales Black-Box Test Design
- Exploratives Testen