XCEPTANCE
Software Technologies

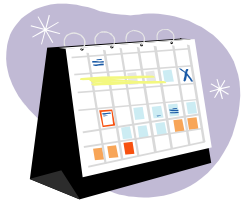
Software-Qualitätssicherung, Teil II: Test in Agilen Prozessen

Friedrich-Schiller-Universität Jena, Wintersemester 2017/2018
Ronny Vogel, Xceptance GmbH

Agenda

- Agile Vorgehensmodelle und Scrum
- Agile QS – Erster Ansatz
- Agile QS – Heute bewährt: Tester im Team
- QS in Scrum

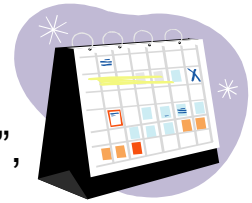
Agile Vorgehensmodelle und Scrum (1)



Historie

- viele Vorgänger, kein einzelner Ursprung
- 1957: erste inkrementelle SW-Entwicklungsmethoden bei IBM
- 1970er: Veröffentlichungen zu Konzepten inkrementeller SW-Entwicklung
- 1986: Begriff “Scrum” erstmals im Artikel “New New Product Development Game” von H. Takeuchi und I. Nonaka in einer Analogie benutzt
- 1988: Tom Gilb beschreibt “Evolutionary Development (EVO)” in seinem Buch “Principles of Software Engineering Management”
- 1990er: diverse Ansätze zu “leichtgewichtigen SW-Entwicklungsmethoden”

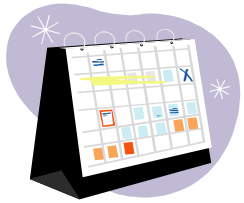
Agile Vorgehensmodelle und Scrum (2)



Historie (fortgesetzt)

- 1993: Ken Schwaber nutzt “Advanced Development Methods (ADM)”, die sich später zu “Scrum” weiterentwickeln
- 1993: erstes “Scrum”-Team um Jeff Sutherland bei der Easel Corporation
- 1995: erste öffentliche Präsentation von “Scrum” durch Ken Schwaber und Jeff Sutherland auf der OOPSLA '95 in Austin, Texas
- 1995: “Adaptive Software Development (ASD)”, “Feature Driven Development (FDD)”, “Dynamic Systems Dev. Method (DSDM)”
- 1996: “Crystal Clear”, “Extreme Programming (XP)”
- 1999: erste wirkliche Popularität Agiler Methoden durch das Buch “Extreme Programming Explained” von Kent Beck

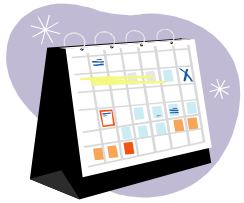
Agile Vorgehensmodelle und Scrum (3)



Historie (fortgesetzt)

- 2001: auf einem Treffen diskutieren 17 Softwareentwickler leichtgewichtige Vorgehensmodelle
  - die erstellen das "*Manifest für Agile Softwareentwicklung*" basierend auf *"Zwölf Prinzipien Agiler Softwareentwicklung"*
  - sie ersetzen den Begriff "*leichtgewichtig*" durch "*agil*"
  - sie prägen den Begriff "*Agile Vorgehensmodelle*"
- 2001: erstes Buch "Agile Software Development with Scrum" von Ken Schwaber und Mike Beedle
- 2003: erste zertifizierte Scrum Master
- November 2017: Aktualisierung von "The Scrum Guide" von Ken Schwaber und Jeff Sutherland, "Scrum Values" zugefügt, siehe www.scrumguides.org

Agile Vorgehensmodelle und Scrum (4)



Heute

- andauernde Evolution agiler Vorgehensmodelle
- noch immer gewisse Vielfalt von Methoden, Werkzeugen, Techniken und sich ändernden Meinungen - aber alle basieren auf ähnlichen Ideen
- Scrum mit Abstand am häufigsten eingesetzt
- agile Vorgehensmodelle weithin akzeptiert und genutzt
- VersionOne,”10th Annual State of Agile Development Survey” 2016: 95% aller Unternehmen nutzen agile Prozesse
- meist keine lehrbuchhafte Anwendung - agile Projekte wählen ein agiles Vorgehensmodell aus und passen es für ihren Zweck an

Agiles Manifest – Agile Werte

"Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:"

Individuen und Interaktionen	*mehr als*	*Prozesse und Werkzeuge.*
Funktionierende Software	*mehr als*	*umfassende Dokumentation.*
Zusammenarbeit mit dem Kunden	*mehr als*	*Vertragsverhandlung.*
Reagieren auf Veränderung	*mehr als*	*das Befolgen eines Plans.*

"Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein."

- in 2001 von 17 Softwareentwicklern erstellt
- agilemanifesto.org (en), agilemanifesto.org/iso/de/manifesto.html (de)

Agiles Manifest – Agile Prinzipien

Zwölf Prinzipien agiler Softwareentwicklung

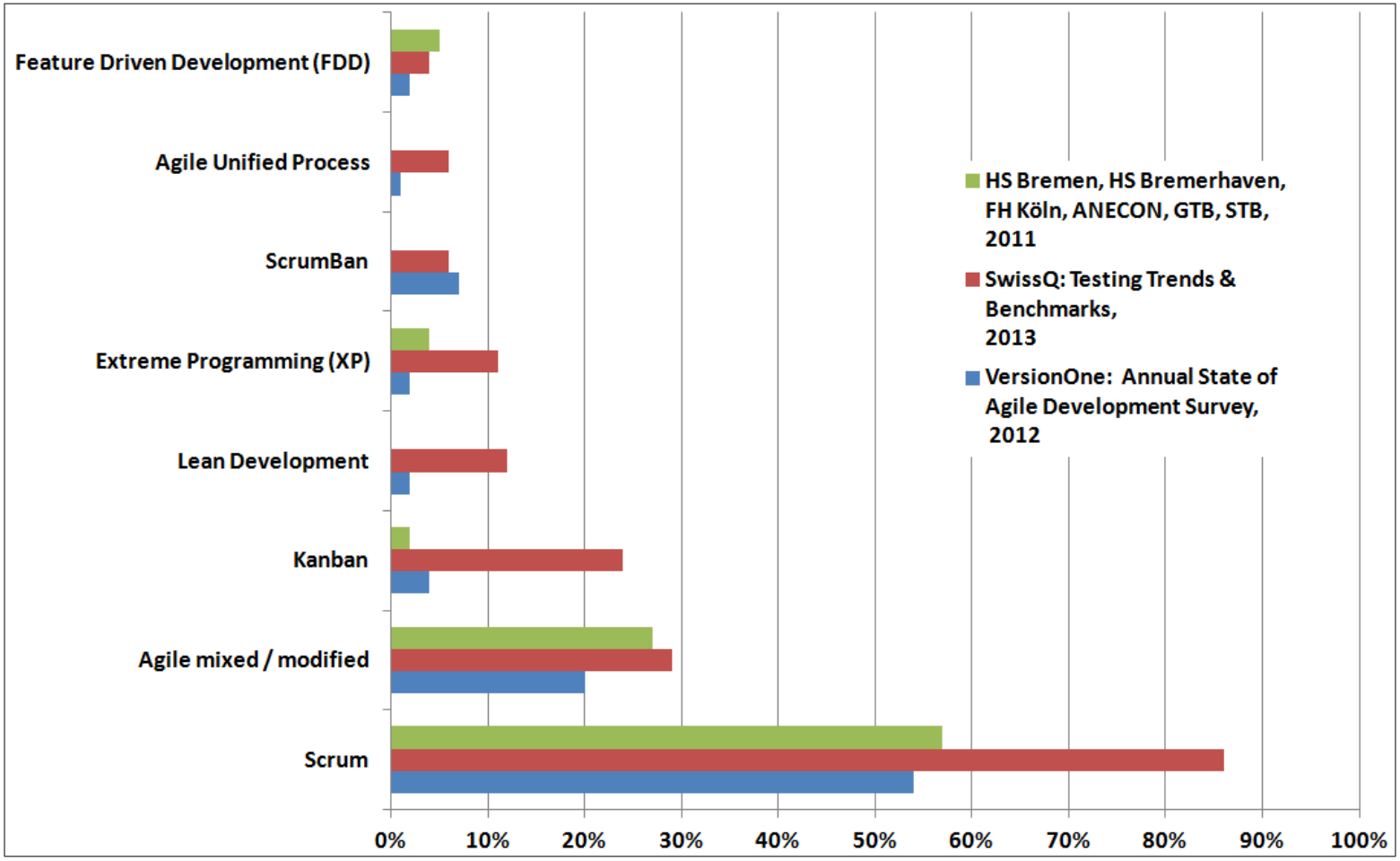
- Kundenzufriedenheit durch kontinuierliche Auslieferung nutzbarer Software
- heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen; agile Prozesse nutzen Veränderungen zum Vorteil des Kunden
- liefere funktionierende Software regelmäßig aus (in Wochen statt Monaten), bevorzuge dabei die kürzere Zeitspanne
- funktionierende Software ist das wichtigste Fortschrittsmaß
- nachhaltige Entwicklung mit gleichmäßigem Tempo auf unbegrenzte Zeit
- enge, tägliche Zusammenarbeit von Fachexperten und Entwicklern

Agiles Manifest – Agile Prinzipien

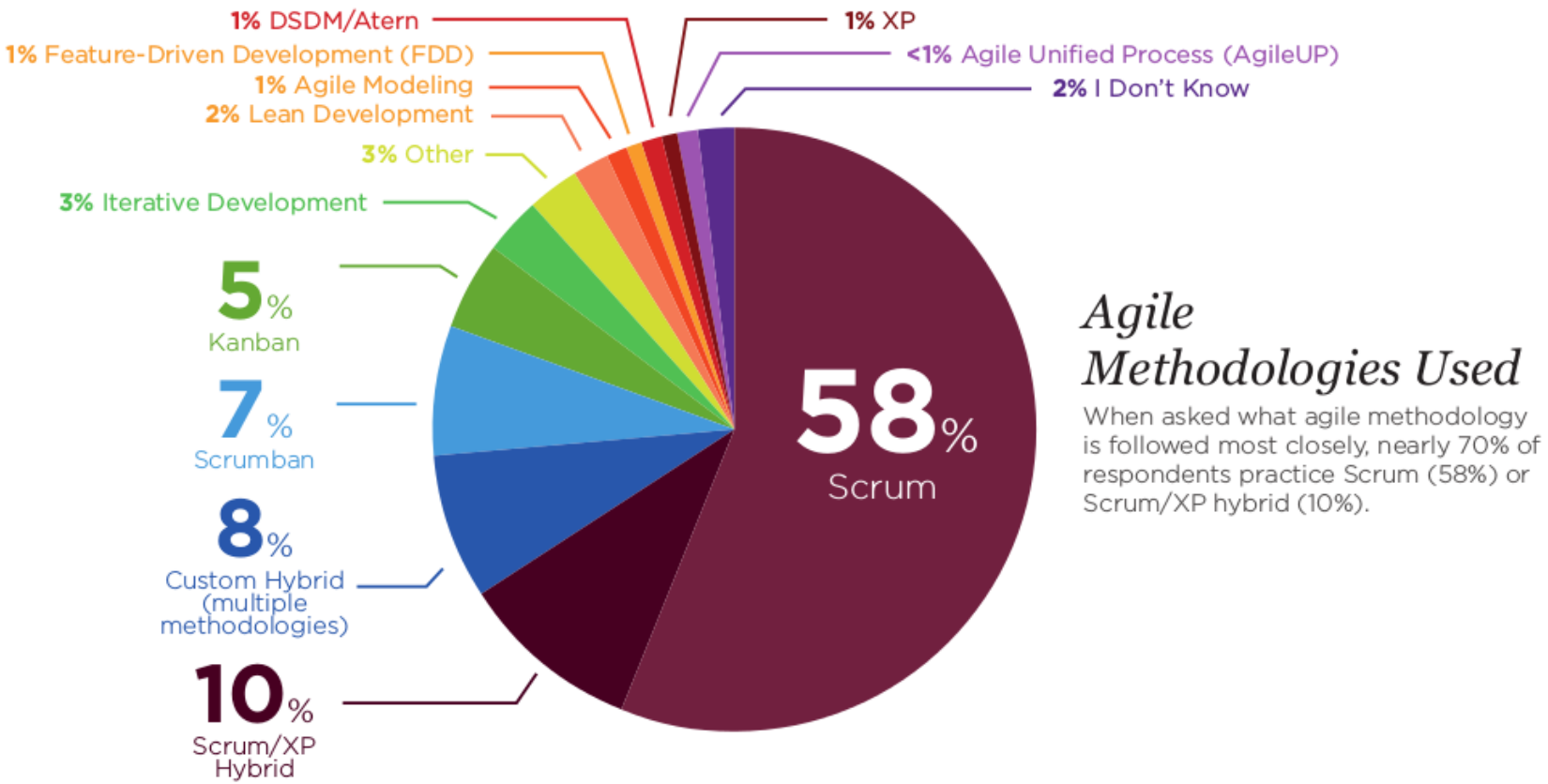
Zwölf Prinzipien agiler Softwareentwicklung (fortgesetzt)

- direkte Gespräche sind die effizienteste und effektivste Form der Informationsübermittlung (➔ Co-Location)
- errichte Projekte rund um motivierte Individuen, gib ihnen das benötigte Umfeld und vertraue darauf, dass sie die Aufgabe erledigen
- ständiges Augenmerk auf technische Exzellenz und gutes Design
- Einfachheit - die Kunst, die Menge nicht getaner Arbeit zu maximieren - ist essenziell
- die besten Ergebnisse entstehen durch selbstorganisierte Teams
- regelmäßige Reflexion der Teams, wie sie effektiver werden können, und Anpassung ihres Verhaltens

Nutzung diverser Agiler Methoden (1)



Nutzung diverser Agiler Methoden (2)



Quelle: VersionOne.com, 10th Annual State of Agile Report, 2016

Scrum

Was ist Scrum?

- Rahmenwerk und Vorgehensmodell zur Entwicklung und Erhaltung komplexer Produkte
- weder ein Prozess noch eine Technik zur Erstellung von Produkten
- schreibt keine konkreten Entwicklungspraktiken vor, sondern erlaubt den Einsatz verschiedener Prozesse und Techniken

Laut „The Scrum Guide“ ist Scrum:

- leichtgewichtig
- einfach zu verstehen
- schwierig zu meistern

Scrum

Basiert auf der Theorie empirischer Prozesssteuerung

- „Empirie“
  - Wissen wird aus Erfahrung gewonnen
  - Entscheidungen werden auf Basis des Bekannten getroffen
- Annahmen
  - die meisten modernen Entwicklungsprojekte sind zu komplex, um durchgängig planbar zu sein
  - wichtige Faktoren sind unberechenbar
  - der perfekte Plan existiert nicht

    ➔ Unsicherheiten und Veränderungen werden akzeptiert

    ➔ Verbesserung von Schätzungen und Verringerung von Risiken durch iterativen, inkrementellen Ansatz

Scrum

Drei Prinzipien der empirischen Prozesssteuerung

Transparenz:
- gemeinsame Sprache
- gemeinsames Verständnis
- wahrheitsgetreue Sichtbarkeit von Fortschritt und Hindernissen

Überprüfung/Inspektion:
- regelmäßige Prüfung der Scrum-Artefakte und des Fortschritts
- Erkennung ungewollter Abweichungen

Anpassung:
- Aspekte des Produktes und des Prozesses werden regelmäßig neu bewertet und bei Bedarf angepasst

Scrum

Ziel

- schnelle, kostengünstige und qualitativ hochwertige Fertigstellung eines Produktes, das einer zu Beginn formulierten Vision entsprechen soll

- Umsetzung der Vision in mehreren Iterationen, je maximal vier Wochen, heute typisch zwei Wochen
  ➔ *Sprints*

- Lieferung einer fertigen Software-Funktionalität am Ende jedes Sprints
  ➔ *Produkt-Inkrement*

- die neu entwickelte Funktionalität sollte in einem Zustand sein, dass sie an den Kunden ausgeliefert werden könnte
  ➔ *Potentially Shippable Code* oder *Usable Software*

Scrum

Das Scrum-Rahmenwerk beschreibt

- drei interne Rollen + drei externe Rollen (später zugefügt)
- Ereignisse/Aktivitäten
  - Sprint Planning
  - Daily Scrum
  - Sprint Review
  - Sprint Retrospective
- Artefakte
  - Product Backlog
  - Sprint Backlog
  - Burn-Down Chart
- Regeln, die Ereignisse und Artefakte miteinander verbinden
- Aktivitäten dienen der unmittelbaren Vorbereitung der nächsten Aktivität
- alle Aktivitäten sind zeitlich beschränkt (*timeboxed*)

Scrum-Rollen im Überblick

Drei interne Kernrollen, das “Scrum Team”:

- Product Owner
- Scrum Master
- Entwicklungsteam

Drei externe Rollen:

- Management
- Kunden
- Nutzer

Kein traditioneller Projektmanager!

➔ Team bestimmt Aufgabenverteilung und Zusammenarbeit selbst

➔ verbleibende Managementaufgaben auf diverse Rollen verteilt

Scrum-Rollen

Rolle „Product Owner“ (PO)

- verantwortet Projekterfolg und Erreichung der wirtschaftlichen Ziele
- kontinuierliche Festlegung und Priorisierung der Anforderungen mit Hilfe des Product Backlog
- darin pflegt er in Zusammenarbeit mit dem Entwicklungsteam die User Stories
- User Stories beschreiben Funktionalitäten aus der Sicht des Benutzers
- der PO versucht, Kundenbedürfnisse und Wünsche optimal in die Entwicklung einfließen zu lassen, ist aber kein Vertreter des Kunden

Scrum-Rollen

Rolle „Product Owner“ (fortgesetzt)

- die Festlegungen des Product Owners sind verbindlich
- nimmt beim Sprint-Review das Entwicklungsergebnis ab
- Entscheidung darüber, ob die vom Entwicklungsteam am Ende jedes Sprints gelieferte Funktionalität akzeptiert wird
- entscheidet über Auslieferungszeitpunkt, Funktionalität und Kosten

Häufige praktische Probleme des Product Owners

- oft nicht bevollmächtigt, Entscheidungen verbindlich zu treffen
- Überlastung mit fremden Aufgaben

Scrum-Rollen

Rolle „Scrum Master“

- unterstützt das Team, aber steuert es möglichst nicht
- treibt den täglichen Prozess, fokussiert das Team auf Scrum
- überwacht den Fortschritt, führt “Sprint Burn-Down Chart”
- stellt optimale Arbeitsbedingungen sicher, beseitigt Hindernisse
- Coach, Moderator und Change Agent; fördert Lernprozesse
- moderiert Besprechungen, z.B. das *Daily Scrum*
- moderiert bei Konflikten
- unterstützt bei der Einführung von testgetriebener Entwicklung, Continuous Integration und Refactoring
- hält Sprint Planning am Beginn und Retrospektive am Ende jedes Sprints
- sollte keine Personalverantwortung für die Teammitglieder haben und keine Leistungsbeurteilungen zu den Teammitgliedern abgeben

Scrum-Rollen

Rolle „Entwicklungsteam“

- setzt User Stories in auslieferbare Produktinkremente um
- arbeitet eigenverantwortlich; organisiert sich weitgehend selbst
- verantwortlich für die Lieferung des Potentially Shippable Code
- reines Scrum erlaubt keine unterschiedlich benannten Rollen im Entwicklungsteam; per Definition sind alle Mitglieder Entwickler (!)
- besteht in der Praxis oft aus Menschen mit cross-funktionalen Fähigkeiten, zum Beispiel:
  - Architekt
  - UI-Designer
  - Entwickler
  - Tester
  - Dokumentierer
- etwa 5 bis 10 Mitglieder

Scrum-Rollen

Drei externe Rollen

- nicht formell eingebunden, aber wichtig für den Erfolg
- auch als “Stakeholder” bezeichnet = vom Projektergebnis betroffen
- Management:
  - verantwortet Rahmenbedingungen (materielle Ressourcen, Unterstützung für den eingeschlagenen Kurs)
  - schützt Scrum-Team vor externen Arbeitsanforderungen
  - unterstützt bei Beseitigung von Hindernissen

- Kunde:
  - Auftraggeber, ermöglicht Projekt und profitiert vom Ergebnis
  - sollte eng mit Product Owner zusammenarbeiten
- Nutzer:
  - kann Produkt aus Nutzersicht beurteilen
  - sollte an Sprint Planning Punkt 1 und Sprint Review teilnehmen

Scrum

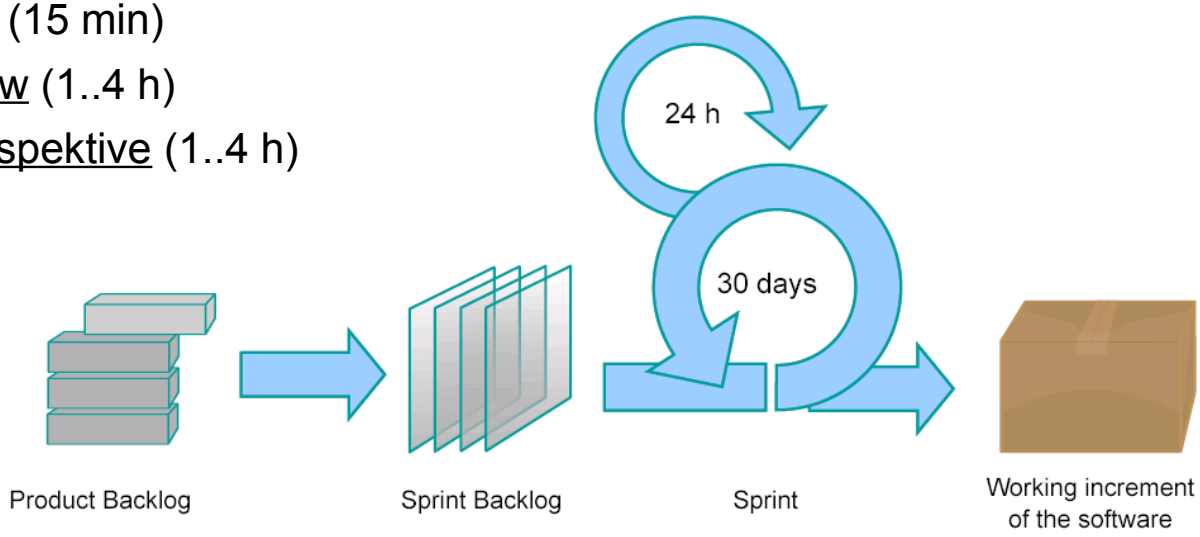
Timeboxing

- Timebox: fester Zeitrahmen für das Projekt und einzelne Aktivitäten
- Idee: jede Aktivität ist auf die festgelegte Dauer beschränkt, auch wenn nicht alle geplanten Inhalte abgeschlossen werden konnten
- noch offene Teile werden in eine folgende Timebox verschoben oder gestrichen
- Zweck: Erhöhung der Effizienz
- „timeboxed“ in Scrum sind alle planmäßigen Meetings, z.B. Daily Scrums, und die Sprints selbst

Scrum

Zyklen

- initiale Erstellung von Produktkonzept („Vision“) und Releaseplan
- kontinuierliche Pflege des Product Backlogs durch den Product Owner
- Sprints dauern 1..4 Wochen; je Sprint durchgeführt:
  - Sprint Planning => Sprint Backlog (2 x 1..4 h)
  - Daily Scrum (15 min)
  - Sprint Review (1..4 h)
  - Sprint-Retrospektive (1..4 h)



Scrum

Produktvision (vom „Scrum Guide“ inzwischen entfernt)

- vor Projektstart erstellt und regelmäßig angepasst
- kann beispielsweise enthalten:
  - Vision
  - Produktidee
  - Funktion
  - Produkteigenschaften
  - Zielgrößen
  - Marktforschungsergebnisse
  - Befragung von Kunden
  - Roadmap
- hilft bei der Kommunikation mit Stakeholdern und bei der Erstellung des Product Backlog

Scrum

Releaseplan (vom „Scrum Guide“ inzwischen entfernt)

- verantwortet vom Product Owner
- wird zu jedem Sprint aktualisiert
- Übersicht zum Zeit- und Kostenrahmen, zu Terminen für Zwischenergebnisse und Fertigstellung
- enthält grobe Reihenfolge der Umsetzung der Anforderungen und die erwartete Anzahl Sprints
- im Projektverlauf iterativ präzisiert
- dabei Risiken berücksichtigt

Scrum

Definition of Done („DoD“)

- gibt genau an, wann eine Anforderung/User Story als fertiggestellt gilt
- vor Projektbeginn definiert von Entwicklungsteam und Product Owner
- enthält auch alle notwendigen Testaktivitäten
- ist verbindlich

➔ Die Umsetzung einer User Story, die nicht alle Kriterien der DoD erfüllt, ist nicht fertig!

Scrum

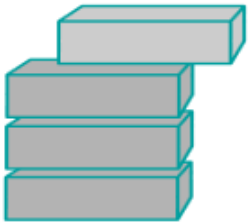
Beispiel für eine DoD

- Code ist fertiggestellt, committed und erfolgreich auf dem CI-System gebaut
- Unit-Tests und Integrationstests sind erstellt und erfolgreich ausgeführt
- Code ist dokumentiert
- Code Review im Team durchgeführt, Coding Guidelines eingehalten
- Release Notes sind erstellt
- QS-Maßnahmen sind beendet
- alle bekannten, noch nicht vollständig behandelten Fehler sind dokumentiert
- es gibt keine kritischen Fehler, die die User Story betreffen
- an der User Story dokumentierte Akzeptanzkriterien sind erfüllt
- Lizenzinformationen verwendeter Module sind dokumentiert
- der Product Owner hat die User Story akzeptiert

Scrum

Product Backlog

- ein zentrales Product Backlog für das Produkt
- Ablage für funktionale und nicht-funktionale Anforderungen
- zu Beginn grobgranular und unvollständig; regelmäßig präzisiert („Refinement“, füher „Grooming“)
- vom Product Owner gepflegt
- umfasst je Anforderung
  - Priorität
  - User Story / Beschreibung
  - Akzeptanzkriterien
  - Risiken
  - Schätzung für Entwicklungsaufwand (vom Team ermittelt)
- liefert den Input für die Sprint Backlogs



Scrum

User Stories

- spezifizieren Anforderungen; eine User Story je Anforderung
- in Alltagssprache bewusst kurz formuliert
- bestehen aus Anforderung und Akzeptanzkriterien
- oft auf Story Cards geschrieben (physisch oder virtuell)

Anforderung

- Typische Form:
  Als <Rolle> **möchte ich** <Wunsch>, **um** <Nutzen/Ziel>
- Beispiel:
  Als registrierter Nutzer **möchte ich** nach dem Login meinen Namen sehen, **um** eine Rückmeldung über das korrekte Login zu erhalten.

Akzeptanzkriterien

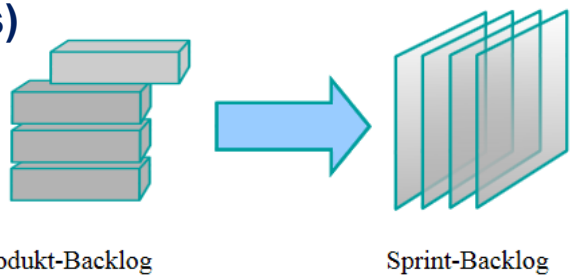
- Liste zu überprüfender Voraussetzungen für die Akzeptanz
- praktisch sind das oft die Anforderungsdetails

Scrum

Sprint Planning Meeting

- Kick-Off zu Beginn jedes Sprints
- Ziel: Erstellung des Sprint Backlog
- Timebox: 2..8 h (2h je Sprint-Woche)
- Input: Product Backlog
- Output: Sprint Backlog
- geteilt in zwei Punkte:

1. Auswahl von Anforderungen (Product Backlog Items)



2. Ermittlung der zur Umsetzung nötigen Aufgaben (Tasks)

Scrum

Sprint Planning Meeting, Punkt 1

- Was kann in diesem Sprint fertiggestellt werden?
- Wie: Verhandlung zwischen PO und Entwicklungsteam
  - PO erläutert Sprint-Ziel
  - Sprint-Ziel vom Team bestätigt
  - PO präsentiert höchstpriorisierte, ausgewählte Product Backlog Items
  - PO und Entwicklungsteam diskutieren diese und priorisieren eventuell neu
  - eventuell auch neue Product Backlog Items erstellt
  - Entwicklungsteam schätzt gemeinsam, welche Product Backlog Items voraussichtlich implementiert werden können (*Forecast*)
  - (Begriff „Commitment“ 2011 aus Scrum Guide entfernt)
- *Tester berücksichtigen den Testaufwand bei Schätzungen*

Scrum

Sprint Planning Meeting, Punkt 2

- Wie wird die ausgewählte Arbeit erledigt?
- Entwicklungsteam (ohne Product Owner) ermittelt die für die Implementierung der Product Backlog Items nötigen Aufgaben (Tasks)
- die Tasks bilden das Sprint Backlog
- Granularität/Aufwand je Task etwa 1-8 h
- *Tester sorgen dafür, dass die Tasks auch Zeit für Unit Tests und andere Entwicklertests enthalten*
- *Tester schreiben separate Karten für Test Tasks ODER*
- *Tester fügen Test Tasks zu Entwicklungs-Task-Karten zu*
  - ➔ *Task ist nicht erledigt, wenn der Test nicht beendet ist*
  - ➔ *versucht, Mini-Wasserfall zu vermeiden*
  - ➔ *funktioniert nur, wenn keine entfernten Kollegen involviert sind*

Kontakt

Xceptance Software Technologies GmbH
Leutragraben 2-4
07743 Jena

Tel.: +49 (0) 3641 55944-0
Fax: +49 (0) 3641 376122

E-Mail: kontakt@xceptance.de
http://www.xceptance.de

Referenzen (1)

Bücher

- Lisa Crispin, Janet Gregory: „Agile Testing – A Practical Guide for Testers and Agile Teams“, Addison Wesley (2009)
- James A. Whittaker, Jason Arbon, Jeff Carollo: „How Google Tests Software“, Addison-Wesley (2012)
- Mike Cohn: „Succeeding with Agile: Software Development Using Scrum“, Addison-Wesley (2009)
- Kent Beck: „Test-Driven Development: By Example“, Addision Wesley (2003)
- Mary & Tom Poppendieck: „Lean Software Development – An Agile Toolkit“, Addison Wesley (2003)
- Whittaker, James, „Exploratory Software Testing“, Addison Wesley (2009)
- Andreas Spillner, Tilo Linz: „Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ISTQB-Standard“, dpunkt.verlag GmbH, 5. Auflage (2012)

Referenzen (2)

Websites zu Agilen Methoden & Scrum

- http://www.scrum.org
- http://www.agilemanifesto.org
- http://www.agilealliance.org
- http://www.controlchaos.com
- http://www.mountaingoatsoftware.com/scrum
- http://www.scrumalliance.org

Foren

- http://tech.groups.yahoo.com/group/agile-testing/
- http://tech.groups.yahoo.com/group/scrumdevelopment/

Blogs

- http://testobsessed.com/ (Elisabeth Hendrickson)
- http://www.kohl.ca/blog/ (Jonathan Kohl)

Referenzen (3)

Websites zum Softwaretest

- http://www.agiletester.ca/ (Lisa Crispin, Janet Gregory)
- http://www.istqb.org/downloads/glossary.html (ISTQB glossary)
- http://www.stickyminds.com
- http://www.satisfice.com (James Bach)
- http://kaner.com/?page_id=7 (Cem Kaner)
- http://testingeducation.org/wordpress/ (Cem Kaner)
- http://www.developsense.com (Michael Bolton)

Copyrights