Scrum eingeführt - und nun?

Software-Entwicklung in Scrum managen

Henrik Rößler

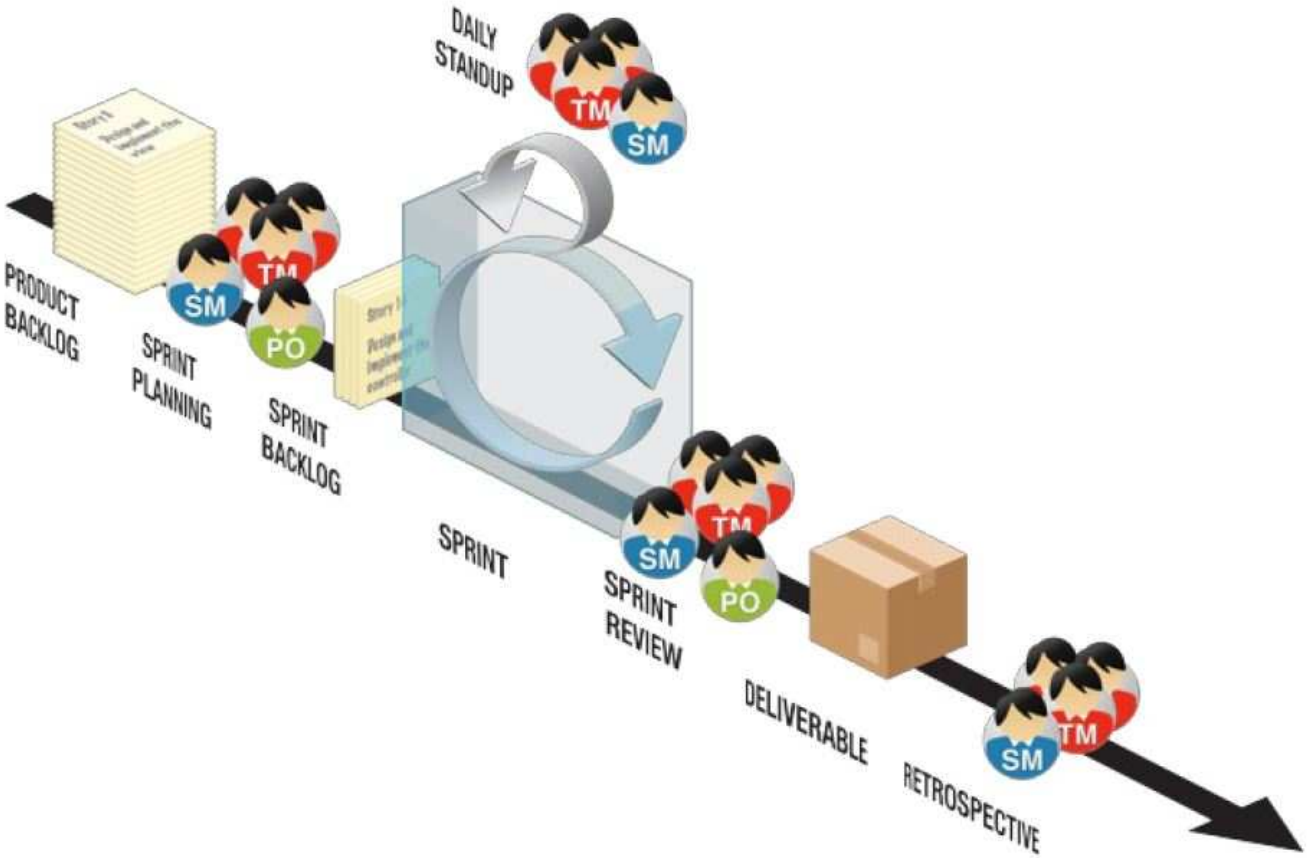
camLine GmbH
http://www.camline.com

Forum 7-it, 22.01.2018

Was gibt es zu hören?

Scrum eingeführt - und nun?

Kurzeinführung Scrum



Unterschiede zwischen Klassisch und Scrum

	Klassisch	**Scrum/ Agile**
Fokus	Große Projekte	Kleine Pakete
Mittel	Viele Features	Inkrementelle Änderungen
Zeiten	Lange Laufzeiten	Kurze Iterationen
Rollouts	Wenige	Viele
Organisation	Projektorganisationen	Kleine Teams
Akteure	Spezialisten	Cross-Funktionale
Feeling	Top-Down	Bottom-Up

Herausforderungen Klassisch

- Hohes Risiko beim Rollout:
  - Ist alles richtig installiert und konfiguriert?
  - Funktioniert die neue Version?
  - Wie nehmen die Benutzer an, was die neue Version bietet?
  - Wie verhält sich das System im Alltag?
  - Wie tief kann direkt nach dem Rollout überhaupt getestet werden?
- Starre Rollen führen zu Problemen bei der Allokation der Ressourcen.
- Änderungen sind schwierig ins Projekt einzuarbeiten.
- Sind eigenartige Verhaltensweisen eigentlich Bugs oder Features?

Herausforderungen in Scrum

- Umstellung der Denkweise in kleine Pakete
- Koordination der kleinen Pakete
- Integration der kleinen Pakete
- Zusammenhalten des Gesamtkonzepts
- Scrum ist unbekanntes Terrain

Nicht jede(r) kann damit umgehen

Scrum erhöht die Transparenz und macht Probleme sichtbar.

Möglichkeiten durch Scrum

- Schnell am Markt
- Kleine Änderungen bedeuten geringeres Rollout-Risiko
- Kurze Rollout-Zyklen
- Arbeit in kleinen und unabhängigen Paketen
- Schnelle Problembehebungen
- Features mit dem Kunden ausprobieren, einfach zurücknehmen oder umbauen
- Kopfmonopole brechen - Kapazitätsengpässe verringern
- Softwarequalität erhöhen - neue Sicht auf Softwarequalität gewinnen

Vielleicht lassen sich Skeptiker überzeugen?

Scrum macht sichtbar gewordene Probleme handhabbar.

Wie anders arbeiten Entwickler in Scrum?

- Das Entwicklungsteam ändert sich selten in seiner Zusammensetzung.
- Das Entwicklungsteam hat eine (nahezu) konstante Velocity.
- Spezialisten werden nur zur Klärung von Detailfragen oder Einführungen hinzugezogen.
- Das Entwicklungsteam schätzt den Aufwand.
- Das Entwicklungsteam arbeitet ausschließlich am Sprint Backlog.
- Änderungen am Sprint Backlog während eines Sprints nimmt nur das Entwicklungsteam vor.
- Das Entwicklungsteams arbeitet während des Sprints selbständig und braucht dafür Vertrauen.

Scrum Teams arbeiten konzentriert und fokussiert.

Never interrupt the team! The Scrum Master will protect the team.

Was ist ein Product Owner? Wie arbeitet er?

- Der Product Owner vertritt die Stakeholder im Scrum Team.
- Ist Besitzer des Product Backlog und hat das finale Sagen zum Product Backlog.
- Definiert Arbeitspakete unabhängig und kleinteilig.
- Detailliert die Arbeitspakete nach Notwendigkeit.
- Definiert paketweise die Abnahme-Kriterien.
- Entwickelt die Definition of Done zusammen mit dem Team.
- Lädt die relevante Stakeholder zum Review ein.
- Aufgabe im Review: Pakete einzeln abnehmen oder zurückweisen.
- Sprintabbruch, wenn das Sprintziel obsolet geworden ist - genau dann, nur dann und nur von ihm.

Und darüber hinaus?

- Qualität von Software und Architektur rücken in den Fokus.
- Im Entwicklungsteam gibt es keine Spezialisten mehr.
- Die Verantwortung für *alles*, was ausgeliefert wird, liegt beim Team — nicht bei einem einzelnen Entwickler.
- Die Gesamtlösung braucht
  - eine robuste Architetur,
  - Paketen die sich in eine handelbarer Größe und Komplexität zusammensetzen,
  - Paketen deren Abhängigkeiten untereinander sich auf handelbare Größe reduziert,
  - jedes in sich funktionierende Pakete und
  - Stabilität im Zusammenspiel der Pakete.
- Jedes dieser Pakete muss
  - in sich funktionieren,
  - einzeln und möglichst automatisch getestet werden können.

Agile Schätzungen - Story-Punkte

Aufwandsschätzungen

Aufwandsschätzungen sind mit erheblichen Unsicherheiten behaftet.

- Die Schätzung von Paketen erfolgt stets durch das Entwicklungsteam.
- Schätzungsergebnisse veralten sehr schnell.
- Schätzungen erfolgen z.B. in Story-Punkten.
- Jedes Team hat ein anderes Verständnis für Story-Punkte.
- Verständnis für Story-Punkte verschiebt sich mit der Zeit.
- Story-Punkte sind in sich nichtlinear.

Velocity

Definition

Velocity bemißt sich anhand der Abarbeitung der in Story-Punkten geschätzten Pakete über die Zeit.

- Velocity läßt sich nur ex post bestimmen.
- Die Fortschreibung von Erfahrungswerten in die Zukunft ist problematisch.
  - Scrum hat das Potential Probleme sichtbar zu machen.
  - Die Themen kommen aufs Tapet und werden mit abgearbeitet.
  - Der komplette Entwicklungsprozess kommt auf den Prüfstand.
  - Inkompatibilitäten zwischen Teammitgliedern werden sichtbar.
  - Verborgene Einstellungen der Teammitglieder zeigen sich.
- Ansprüche an Qualität ändern sich.
- Die Möglichkeiten, Qualität zu erzeugen, ändern sich.

Verschiebung der Sichtweise

Beobachtungen

- Die Durchschnitte der Velocity bleiben in engen Grenzen.
- Product Owner lernen mit der Zeit, Pakete in ungefähr gleiche Größen zu schneiden. Das Entwicklungsteam hilft dabei.

Lemma

Wenn Velocity nahezu konstant ist und Pakete fast gleiche Größen haben, kann man Terminplanung betreiben.

Bitte beachten!

- Product Owner und Stakeholder lernen Paketabhängigkeiten zu managen. Das Entwicklungsteam unterstützt sie dabei.
- Entwicklung des Development-Teams braucht Raum.

Motivation

- Über 30 Jahre Top-Down-Konditionierung von Entwicklern liegen hinter uns.
- Scrum Einführung macht 3 Attitüden von Beteiligten sichtbar:

  1. „Na endlich!“
  2. „Was soll es bedeuten?“
  3. „Was soll'n *der* Scheiß schon wieder!“

Achtung

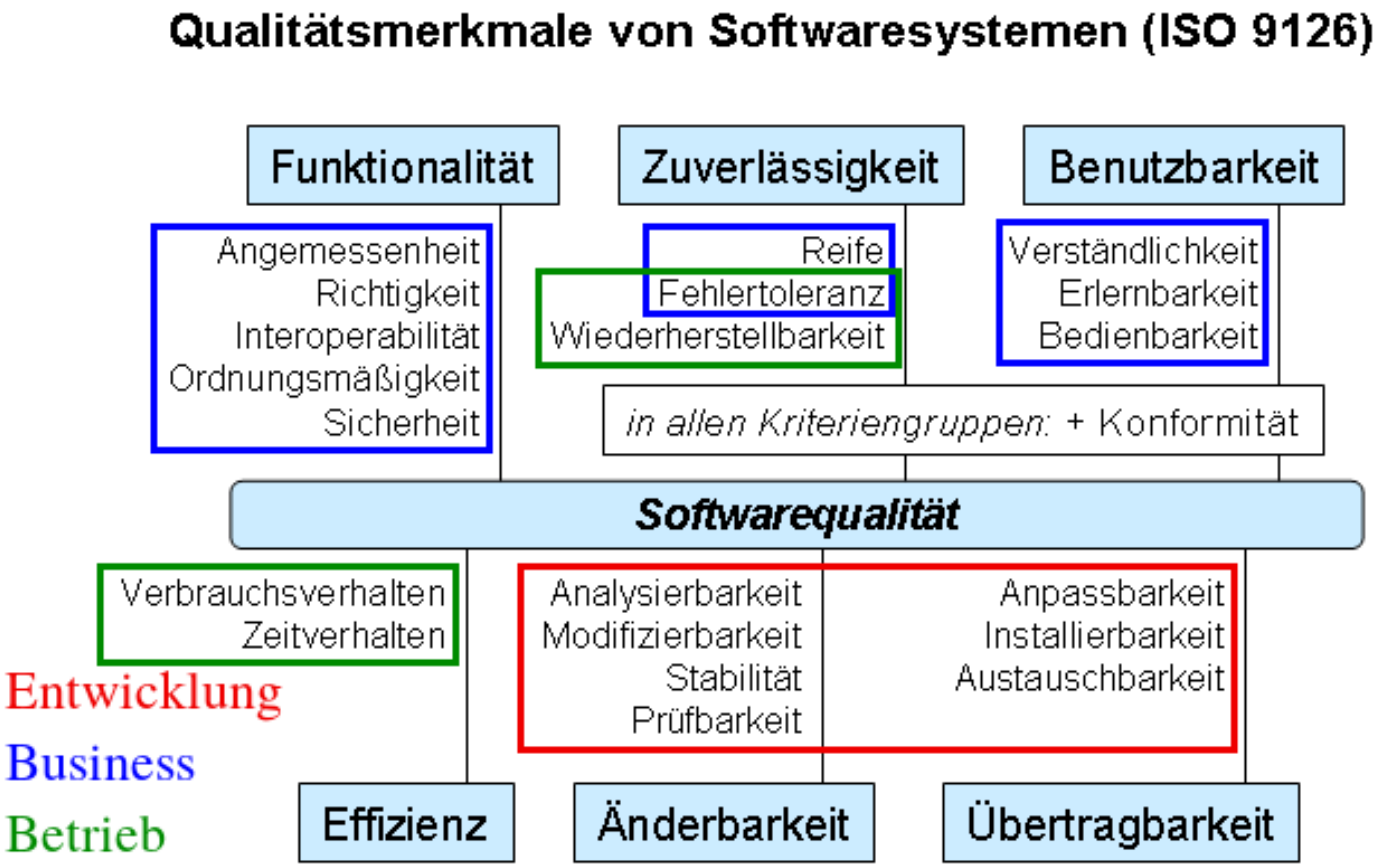
Der Kampf zwischen (1) und (3) kann heftig ausfallen. Ist für die (1)-er Leute keine hinreichende Unterstützung möglich, ist von Scrum dringend abzusehen.

- (2)-Leute sind eigentlich (1)-Leute, die jedoch sehr unsicher sind.

Einstellungen

- Reine Befehlsempfänger sind für Scrum nicht geeignet.
- Entwickler der (3)-Kategorie zollen ihren Respekt eher der Vergangenheit als der Zukunft.
- Die Entwickler der Vergangenheit werden höchstwahrscheinlich ihren Code niemals wieder zu Gesicht bekommen,
- Die Entwickler der Zukunft müssen verstehen, was im Code passiert.
- Wenn ein Entwickler nach 20 Jahren immer noch identisch codiert, hat er sich nicht weiterentwickelt - unsere Programmiersprachen haben sich jedoch durchaus entwickelt.

ISO-9126


Qualitätsmerkmale von Softwaresystemen (ISO 9126)
Funktionalität
Zuverlässigkeit
Benutzbarkeit
Angemessenheit
Richtigkeit
Interoperabilität
Ordnungsmäßigkeit
Sicherheit
Reife
Fehlertoleranz
Wiederherstellbarkeit
Verständlichkeit
Erlernbarkeit
Bedienbarkeit
in allen Kriteriengruppen: + Konformität
Softwarequalität
Verbrauchsverhalten
Zeitverhalten
Analysierbarkeit
Modifizierbarkeit
Stabilität
Prüfbarkeit
Anpassbarkeit
Installierbarkeit
Austauschbarkeit
Entwicklung
Business
Betrieb
Effizienz
Änderbarkeit
Übertragbarkeit

Clean Code

- Unzählige Quellen: http://clean-code-developer.de/, https://de.wikipedia.org/wiki/Clean_Code
- Prinzipien
  - DRY, YAGNI, KISS,
  - SLA, CoC, PoLS, Law of Demeter und
  - SOLID

Kosten von Software

Die Kosten für Software liegen bei der Entwicklung im Wesentlichen im Lesen von Code.

Ziele von Clean Code

- Jede(r) muss den gesamten Code im Verantwortungsbereich des Teams durch selbständiges Lesen verstehen können.
- Niemand darf Angst davor haben, Code zu ändern.

Monolithen

Wahnsinn monolithischer Systeme

In monolithischen Systemen betreiben wir kompletten Wahnsinn: wir ändern eine Zeile Code, erwarten ein Recompile von 10^5, manchmal 10^7 Zeilen Code und jammern dann, dass unsere IDEs zu langsam sind.

- Die IT-Industrie sitzt auf Milliarden Zeilen Code. Unsere Produktivität ist so gering, dass wir aber immer nur einige wenige Dinge anfassen.
- Wir haben keinen Überblick, was alles bereits unabhängig von uns entwickelt und veröffentlicht wurde.
- Wir wissen nur in Ansätzen, wie unser eigenes Systemen funktioniert.

Forderungen an eine effektive Zerlegung

Wir brauchen eine effektive Zerlegung unserer Systeme. Die Teile, die Ergebnis der Zerlegung sind, nennen ich Artefakte.

- Effektiv ist eine Zerlegung, wenn
  - von der Bezeichnung der einzelnen Artefakte auf ihre Verantwortlichkeiten geschlossen werden kann,
  - alle Verantwortlichkeiten vollständig auf die Artefakte abgebildet sind und
  - die Verantwortlichkeiten der Artefakte nicht überlappen.
- Jedes Artefakt, ist dahingehend zu entwickeln, dass es in sich
  - funktionsfähig,
  - getestet und
  - lesbar

  ist.
- Die Gesamtlösung setzt sich aus den Artefakten zusammen: z.B. Dependency Injection.

Prinzipien

Grössenbegrenzung der Artefakte

Kein Artefakt darf zum Bereitstellen (Compilieren, Unit-Testen und Packen) mehr als 5 Minuten „unterwegs“ sein!

Wir haben zwei Typen von Artefakten:

- API-Artefakte
- Implementierungs-Artefakte

Abhängigkeiten zwischen Artefakten

Die Abhängigkeiten zwischen den Artefakten sind zu minimieren. das funktioniert am Besten, wenn

- API-Artefakte nur von anderen API-Artefakten abhängen.
- Implementierungs-Artefakte nur von API-Artefakten abhängen.

Wo bleiben Projektmanager in Scrum?

Der Kampf um Ressourcen und Termine läuft in Scrum auf ganz anderer Ebene als in klassischer Entwicklung. Durchgriff auf einzelne Ressourcen ist weder möglich noch nötig.

- Product Owner
  - enge Zusammenarbeit mit den Entwicklungsteams
  - Fokus: Detailfragen
  - In größeren Organisationen: mehrere Product Owner im Team – ggf. mittels Scrum.
- Interne Stakeholder
  - enge Zusammenarbeit mit den internen und externen Kunden
  - Fokus: Überblick und Strategie

Eckpfeiler zur Struktur

Entscheidungen zu Details müssen Bestand haben. Best Practice: Je Product Backlog muss genau ein Product Owner agieren.

Zusammenfassung

- Management-Ebene kommt in Scrum um eine Änderung des Selbstverständnis nicht herum.
- Scrum ohne Clean Code wird Flacid Scrum.
- Scrum-Rituale/ -Zeremonien/ -Events brauchen Zeit - Arbeits- und damit Projektzeit.
- Denkweisen in Agile und Clean Code brauchen Zeit, sich zu entwickeln.
- Management-Ebene unterstützt die Einführung von Scrum am besten, wenn sie selbst in Scrum arbeitet.

Schlusswort

Management bringt alles mit, was für Scrum nötig ist. Die Schwerpunkt und Aufgaben ändern sich.

Fragen

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?