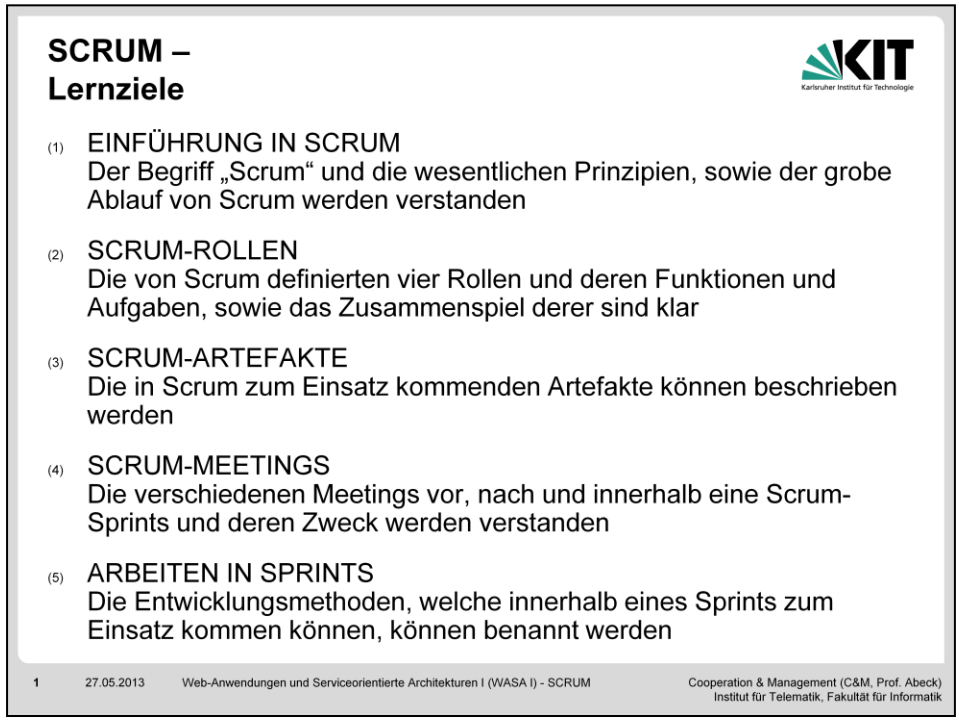


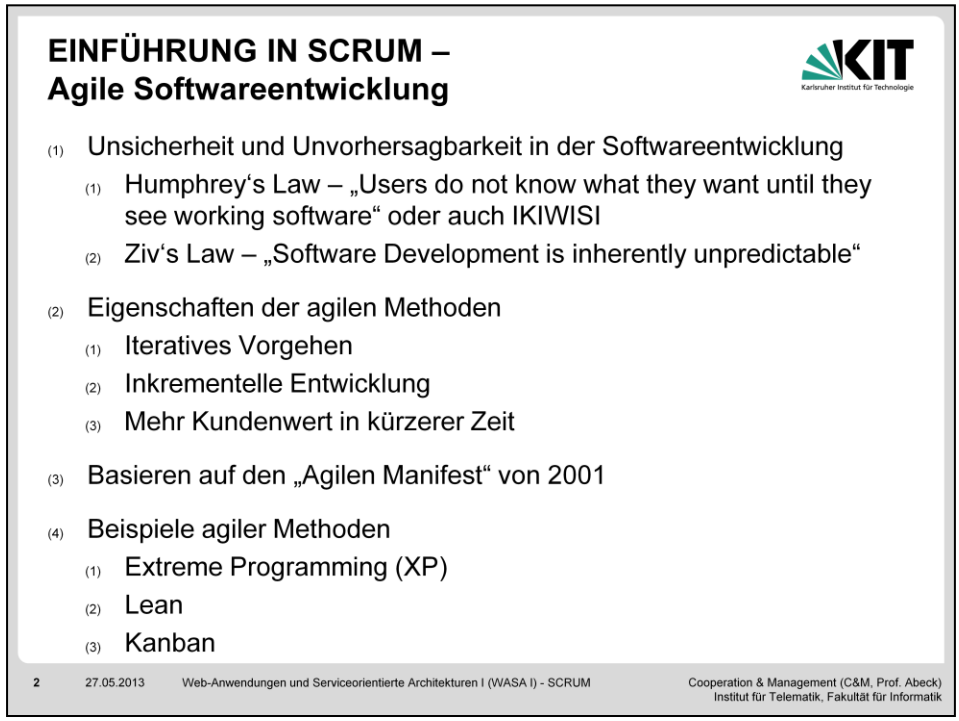
(1) Das erste Lernziel führt in die agile Softwareentwicklung ein und erläutert die Ziele und Prinzipien von Scrum. Ferner wird der Ablauf von Scrum beschrieben.

(2) Die drei Rollen welche das ScrumTeam bilden werden mit ihren Aufgaben, Tätigkeiten und Zielen vorgestellt.

(3) Scrum definiert verschiedene Artefakte, die während der Entwicklungsarbeit zum Einsatz kommen und Transparenz innerhalb des Teams herstellen.

(4) Scrum beschreibt fünf verschiedene Meetings, die vor, während und nach einem Sprint durchgeführt werden, um die Transparenz und Adaption des Teams zu fördern.

(5) Zur konkreten Umsetzung des Produkts können innerhalb eines Sprints verschiedene Methoden aus der agilen Softwareentwicklung zum Einsatz kommen.



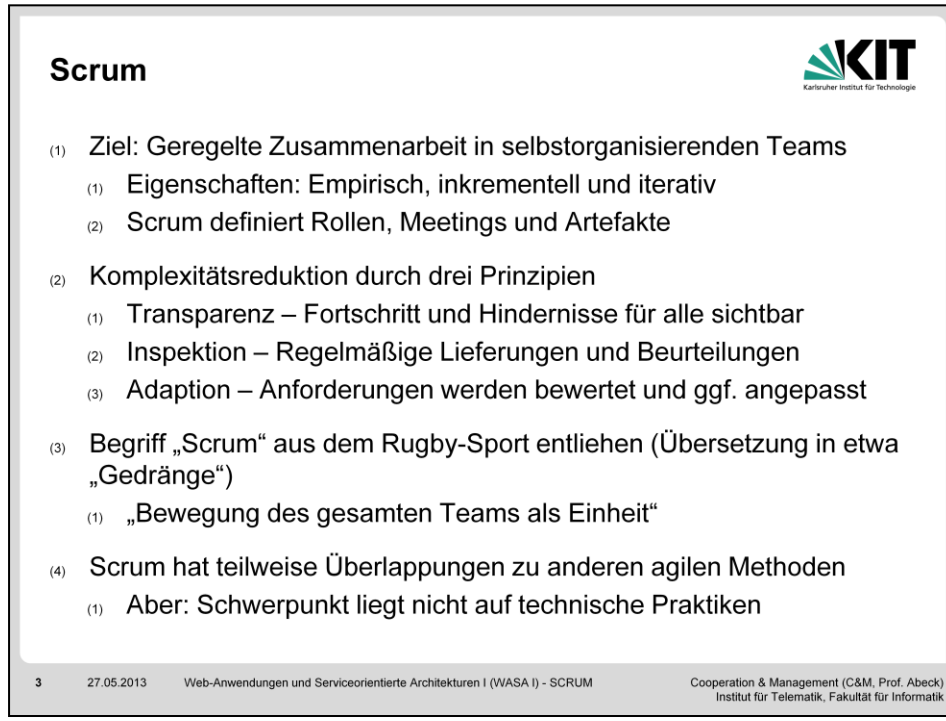
(1) Die zunehmende Unsicherheit und Unvorhersagbarkeit in der Softwareentwicklung, hauptsächlich bedingt durch die komplexer gewordenen Anforderungen, welche heute an Software gestellt werden, hat zu einem Umdenken in dem methodischen Vorgehen zur Softwareentwicklung geführt – hin zu den heute vielfach eingesetzten agilen Methoden. Darüber hinaus sind heute deutlich mehr Personen an einem Softwareentwicklungsprojekt beteiligt, was deren Interaktion unterinander wesentlich komplexer gestaltet. Viele Stakeholder wissen oft nicht genau, was sie konkret von der Software erwarten oder können dies gegenüber den Entwicklern nicht richtig kommunizieren (IKIWISI = „I know it when I see it").

(2.1) Im Kern jeder agilen Methode stehen Iterationen. Das sind Zyklen fester Dauer, in denen sämtliche Aktivitäten aus klassischen Vorgehensmodellen durchgeführt werden, jedoch wesentlich kürzer getaktet und dafür in mehreren Iterationen. Die Verwendung mehrerer Iterationen haben das Ziel zu dem entwickelten Softwareprodukt möglichst schnell Feedback vom Kunden oder Anwender zu erhalten.
(2.2) Bei der inkrementelle Entwicklung werden das nach jeder Iteration entstandene (Teil-)Produkt direkt ausgeliefert bzw. angewendet. Bei jeder Iteration wird ein vertikal durch die Systemarchitektur liegender Querschnitt geliefert.
(2.3) Richtig angewandt liefern agile Methoden bessere Ergebnisse in kürzerer Zeit durch die klare Fokussierung auf die frühe Bereitstellung von Kundenwert. Mit jedem Inkrement steht ein neuer Teil eines Produkts zur Verfügung, der idealerweise direkt vom Kunden verwendet werden kann.

(3) Das „Agile Manifest" von 2001 versucht die Grundprinzipien in der agilen Entwicklung in vier Kernbotschaften zusammenzufassen: Individuen und deren Interaktion über Prozesse und Werkzeuge, funktionierende Software über umfassende Dokumentation, die Zusammenarbeit mit dem Kunden über Vertragsverhandlungen und das Eingehen auf Veränderungen über das Befolgen eines Plans.

(4.1) Exterme Programming (XP) legt den Schwerpunkt auf die Praktiken der Entwicklung bzw. Programmierung, definiert aber auch einen entsprechenden Ablauf. Zu den Praktiken gehören u.a. testgetriebene Entwicklung, kontinuierliche Integration, inkrementeller Entwurf, räumliche Nähe, Pair Programming etc. Viele Ideen aus XP finden sich in Scrum wieder.
(4.2) Lean hat seine Wurzeln im der schlanken Produktion (Toyota). Es basiert auf mehreren Säulen: Kontinuierliche Verbesserung (Kaizen), Selbstreflektion (Hansei) sowie Fokus auf die Entwicklung von selbstorganisierenden Teams. Die Ideen der agilen Entwicklung und Lean ergänzen sich gut.
(4.3) Kanban ist eine Erweiterung der Ideen von Lean und hauptsächlich eine Change-Management-Methode. Grundideen sind Signalkarten (jap. „kan-ban"), verbrauchsgesteuerte Prozesse (Pull-Prinzip) und eine entsprechende Visualisierung der gesamten Wertschöpfungskette.

IKIWISI	„I know it when I see it"
XP	Extreme Programming

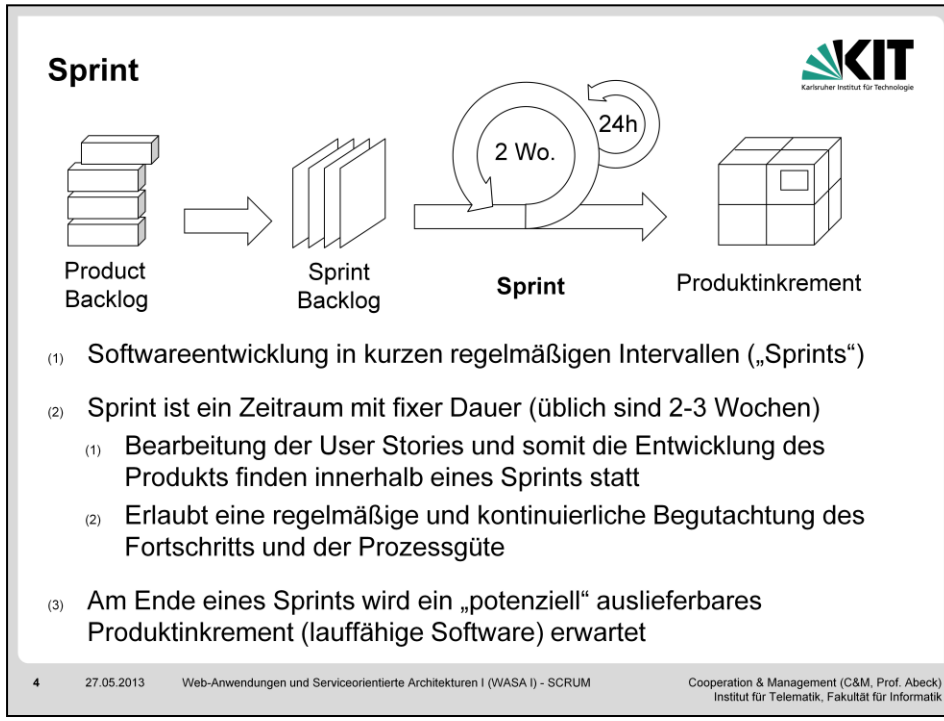


(1) Scrum legt als Entwicklungsmethodik Wert auf die geregelte Zusammenarbeit in selbstorganisierenden Teams. Die Wurzeln von Scrum liegen in der empirischen Prozesssteuerung.
(1.1) Scrum ist empirisch hinsichtlich Anpassungen und Entscheidungen, die auf gewonnenen Erkenntnissen und Erfahrungen beruhen. Ferner forciert Scrum die inkrementelle und iterative Entwicklung durch kurze Entwicklungsintervalle („Sprints“) und regelmäßige Lieferungen von Produktinkrementen an den Kunden.
(1.2) Scrum als Framework definiert drei Rollen, fünf Meetings und mehrere unterstützende Artefakte zur Visualisierung des Prozessfortschritts.

(2) Die drei wesentlichen Eckpfeiler von Scrum, um die Komplexität der Softwareentwicklung zu reduzieren sind Transparenz, Inspektion und Adaption.
(2.1) Die Fortschritte und Hindernisse werden durch entsprechende Artefakte visualisiert und somit für alle sichtbar
(2.2) Durch regelmäßige Lieferungen und entsprechendes Feedback dazu wird der Fortschritt eines Produktes besser überprüfbar.
(2.3) Anforderungen werden nicht zu Beginn einmal festgelegt und nicht mehr geändert, sondern kontinuierlich überprüft, bewertet und ggf. angepasst.

(3) Der Begriff „Scrum“ kommt vom Rugby und heißt übersetzt in etwa „Gedränge“. Scrum im Rugby ist ein Vorgang, bei dem sich das gesamte Teams als eine Einheit bewegt, um den Ball auf der eigenen Seite zu befreien. Dieser Begriff lässt in sinngemäß auf die Softwareentwicklung übertragen, wo ein Team als eine Einheit agieren muss, um ein Produkt von der Anforderungsaufnahme bis zur Auslieferung umzusetzen.

(4) Scrum ist verwandt mit anderen agilen Methoden. Gemeinsamkeiten sind beispielsweise die Selbstorganisation und Verbesserungen durch Feedbackschleifen aus Lean, die Visualisierung des Fortschritts aus Kanban und die Rollen und Iterationen aus XP.
(4.1) Im Unterschied zu XP legt Scrum keinen Schwerpunkt auf technische Praktiken. XP ist auch deutlich regulativer als Scrum. Jedoch werden die technischen Praktiken aus XP für eine erfolgreiche Umsetzung auch in Scrum benötigt.



(1) In Scrum findet die komplette Bandbreite der Softwareentwicklung innerhalb eines immer gleich langen Zeitraums statt, der im Vergleich zu anderen Vorgehensmodellen relativ kurz getaktet ist. Dieser Zeitraum wird in Scrum „Sprint“ genannt. Die kurze Dauer eines Sprints erlaubt es einerseits relativ schnell zu sichtbaren Ergebnissen zu kommen und andererseits kontinuierlich die bisherigen Vorgehensweisen kritisch zu analysieren und beim nächsten Sprint zu verbessern.

(2) Der Sprint ist ein Zeitraum mit fixer und konstanter Dauer, d.h. die Dauer der Sprints sollte stets gleich bleiben und nicht geändert werden. Üblich sind Spints mit einer Dauer von zwei bis drei Wochen.
(2.1) Innerhalb eines Sprints wird eine vorher ausgewählte Menge von User Stories bearbeitet. Sämtliche Entwicklungsarbeiten, von der Analyse bis zur Implementierung, Testen und Integration finden innerhalb des Sprints statt.
(2.2) In Scrum geht es um die ständige Reflexion und Verbesserung, dazu definiert Scrum relativ kurze Feedbackzyklen. Der Sprint erlaubt es dem Team in wenigen Wochen über die erzielten Ergebnisse und den Prozess der Entwicklung zu reflektieren. Während eines Sprints unterstützt außerdem ein tägliches kurzes Treffen (Daily Scrum) die kurzfristige Reflexion und hilft die Arbeit zu organisieren.

(3) Jeder Sprint hat das Ziel ein potentiell auslieferbares Produktinkrement auszuliefern. Ob es tatsächlich an den Kunden ausgeliefert wird, ist dagegen offen. Allgemein sollten aber nach Abschluss des Sprints folgende Eigenschaften erfüllt sein: Das Produktinkrement ist im Sinne der vorher festgelegten Anforderungen funktional vollständig umgesetzt, es ist getestet und frei von Fehlern und auf einer realistischen Systemumgebung vorführbar (nicht nur innerhalb einer Entwicklungsumgebung). Ein weiteres Kriterium ist, dass das Produktinkrement sich möglichst über alle Ebenen einer Softwarearchitektur (Datenhaltung, Geschäftslogik, Präsentation) erstreckt. Ein nur auf eine bestimmte Schicht fokussierter Sprint bietet im Allgemeinen keinen Mehrwert für den Kunden.

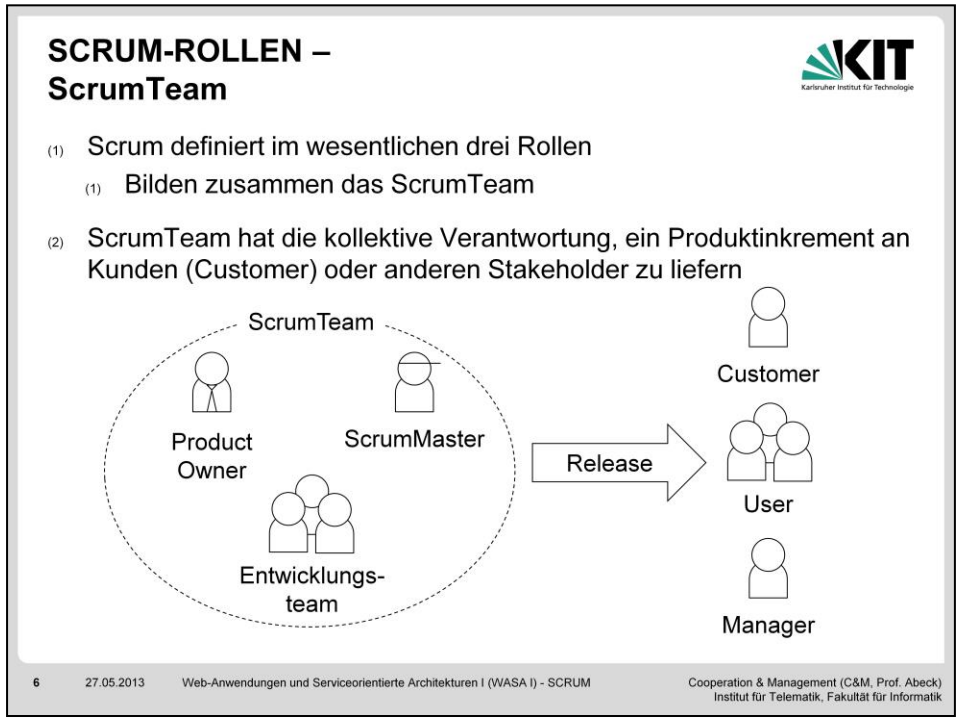
LZ EINFÜHRUNG IN SCRUM – ÜA

(1) Was war die wesentliche Motivation für die Entwicklung von Scrum?

(2) Was sind die drei Prinzipien, auf denen Scrum aufbaut?

(3) Woher stammt der Begriff „Scrum“?

(4) Scrum wird häufig als Prozess oder Methode bezeichnet bzw. dargestellt, dabei sind diese Begriffe genaugenommen nicht treffend. Was wäre ein besser Begriff?



(1) Scrum definiert mehrere Rollen, deren Aufgaben und Verantwortungen genau aufgeteilt sind. Die wichtigsten Rollen sind Product Owner, Entwicklungsteam (meistens nur als „Team“ bezeichnet) und ScrumMaster.
(1.1) Alle drei Rollen bilden das ScrumTeam (nicht mit dem Entwicklungsteam zu verwechseln, welches Teil des ScrumTeams ist). In vielen Darstellung wird der Product Owner mit einer Krawatte abgebildet, da er die Geschäftsseite vertritt, während der ScrumMaster als „Team Coach“ in den Abbildungen eine Kappe trägt.

(2) Das ScrumTeam hat die kollektive Verantwortung ein Produkt an Kunden oder andere Interessensvertretern (Stakeholder), wie beispielsweise Manager oder Fachbereiche zu liefern. Idealerweise werden die drei Scrum-Rollen in vollständiger Wahrnehmung und nicht nur über Rollenverteilung besetzt. Ein ScrumMaster sollte beispielsweise nicht auch gleichzeitig die Rolle des Product Owners oder eines Mitglied des Entwicklungsteams besetzten.

Weitere Rollen, welche auch an der Produktentwicklung maßgeblich beteiligt sind (z.B. Kunden, Anwender, Manager etc.) sind nicht Bestandteil des ScrumTeams. Der Kunde (Customer) und Anwender (User) werden insbesondere vor Beginn und nach Abschluss eines Sprints hinzugezogen. Das Management stellt entsprechende materielle Ressourcen (Räume, Arbeitsplätze etc.) zur Verfügung und unterstützt den ScrumMaster bei der Beseitigung von Hindernissen.

(1) Der Product Owner (oder kurz PO) ist für den geschäftlichen Erfolg des zu entwickelnden Produkts und die Rentabilität des Entwicklungsvorgehens verantwortlich. Er oder sie definiert dazu die Produkteigenschaften und verfeinert diese in konkrete Anforderungen. Hierzu arbeitet er mit dem restlichen ScrumTeam, aber auch mit Kunden, Anwendern und anderen Stakeholdern zusammen und konsolidiert deren Anforderungen in das Product Backlog.

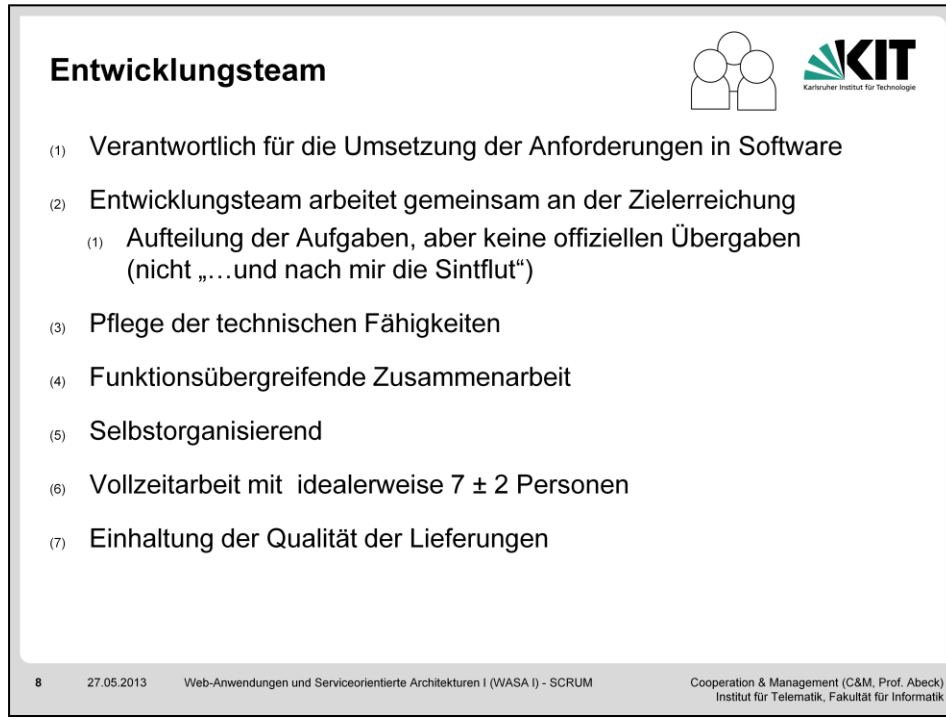
(2) Der Product Owner ist dafür zuständig eine Vision des Produktes zu etablieren, welche an alle Beteiligten kommuniziert werden kann. Die Produktvision soll dabei das grobe Ziel des Produkts aufzeigen, also was das Produkt werden soll und warum es entwickelt wird. Dies dient dem ScrumTeam als Leitlinie durch das gesamte Entwicklungsvorhaben und ist somit eher langfristig zu sehen.

(3) Als Verantwortlicher für den (geschäftlichen) Erfolg des Produkts hat der PO die Aufgabe die vom Entwicklungsteam gelieferten Ergebnisse zu inspizieren und sie dann mittels entsprechenden Feedbacks an das Entwicklungsteam entweder zu akzeptieren oder zurückzuweisen. Das Feedback ist erforderlich, damit die Mitglieder des Entwicklungsteams die Hintergründe zu den Anforderungen verstehen. So gelangt das Team mit der Zeit zu umfassendem Domänenwissen und versteht, wie die Anwender das Produkt in der Praxis verwenden.

(4) Da nicht alle Details zu Anforderungen im Voraus analysiert werden können, werden die Anforderungen im Zuge ihrer Bewegungen im Product Backlog ständig analysiert und verfeinert. Außerdem muss der Product Owner die Anforderungen hinsichtlich ihres Geschäftswerts schätzen können. Die kontinuierliche Pflege des Product Backlogs wird als „Backlog Grooming“ bezeichnet.

(5) Der Product Owner bestimmt auf der Basis von marktwirtschaftlichen Fakten die Zeitpunkte der Lieferung der Produktinkrements. Allerdings hat in der Praxis nicht jeder Sprint auch eine echte Produktlieferung an den Endkunden zur Folge.

(6) Die Rolle des Projektleiters gibt es nicht Scrum nicht. Einige Aufgaben des Product Owners entsprechen denen eines Projektleiters, aber viele klassische Projektmanagementaufgaben gehören nicht zu seiner Agenda, sondern liegen auch beim Entwicklungsteam und ScrumMaster. Der Product Owner muss die Selbstorganisation des Entwicklungsteams respektieren und darauf achten, dass seine Ziele nicht im Widerspruch zu den Zielen des ScrumTeams stehen.



(1) Die Hauptverantwortung des Entwicklungsteams ist die Umsetzung der gewählten Anforderungen in qualitativ hochwertige und lauffähige Software.

(2) Das Entwicklungsteam arbeitet gemeinsam an der Erfüllung des Sprint-Ziels. Die Arbeiten werden zwischen den Teammitgliedern aufgeteilt, aber es findet weiterhin eine enge Zusammenarbeit statt. Zwar gibt es auch innerhalb eines Teams durchaus Spezialisten (z.B. Architekt, GUI-Designer, Datenbankexperte etc.), aber es finden keine offiziellen Übergaben wie in klassischen Entwicklungsprozessen statt.

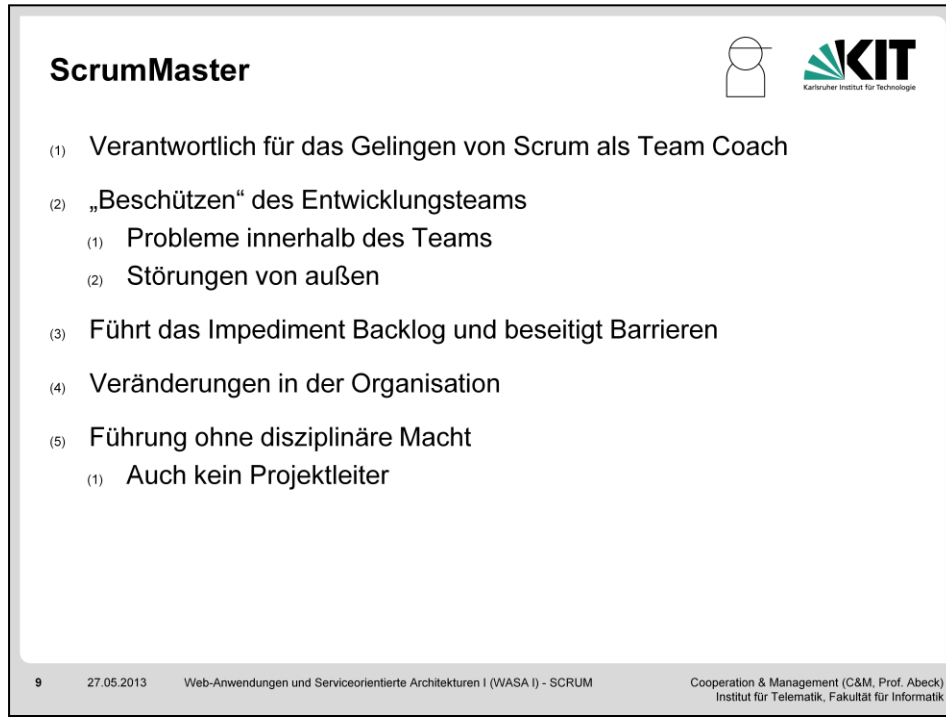
(3) Das Entwicklungsteam muss über die erforderlichen technischen Fähigkeiten verfügen und sich laufend über den aktuellen Stand der Technik informieren. Neben den technologischen Fähigkeiten (z.B. spezielle Frameworks) gehören dazu auch allgemeingültige technische Fähigkeiten, wie der Einsatz von Entwurfsmustern, Refaktorisierung und andere agile Techniken.

(4) Im Entwicklungsteam befinden sich Entwickler ebenso wie Tester, Designer, Architekten und weitere Rollen. Im Sprint muss das Entwicklungsteam funktionsübergreifend zusammenarbeiten, so dass idealerweise sind in einem Entwicklungsteam die ursprünglichen Rollen nicht mehr direkt auf eine Person abbildbar. Scrum definiert auch innerhalb eines Teams keine spezifischen Rollen und es finden auch keine offizielle Übergaben (z.B. Übergabe des Designs in die Implementierung wie in klassischen Vorgehensmodellen) statt.

(5) Ein wichtiger Aspekt von Scrum ist die Selbstorganisation des Entwicklungsteams. Für die Erreichung der vorher selbst festgelegte Ziele ist das Team selbst verantwortlich. Daher ist dafür Sorge zu tragen, dass das Entwicklungsteam während eines Sprints vor negativen Einflüssen von außen geschützt ist.

(6) Entwicklungsteams sind in Scrum eher klein gehalten, um eine effiziente Kommunikation zu gewährleisten. Das Entwicklungsteam sollte nahe beisammen arbeiten, möglichst in einem Raum. Als bewährte Größe haben sich fünf bis neun Personen herausgebildet, ideal sind sieben Personen, die in Vollzeit im Entwicklungsteam mitarbeiten.

(7) Da das Entwicklungsteam sich selbst organisiert und größtenteils auch selbst bestimmt, welche Ziele in einem Sprint umgesetzt werden, ist folglich auch ausschließlich das Entwicklungsteam für die Einhaltung einer hohen Qualität der Lieferung an den Kunden verantwortlich.



(1) Der ScrumMaster hilft dem Entwicklungsteam dessen Probleme zu lösen, indem er die Rolle eines Moderators, Mentors, Trainers oder auch Mediator einnimmt. Zentrale Aufgabe ist die Sicherstellung einer effizienten und effektiven Arbeitsweise des Entwicklungsteams. Er oder sie erkennt menschliche Probleme und führt durch auftretende Konflikte innerhalb des Teams. Der ScrumMaster schafft Transparenz über sämtliche Angelegenheiten des Teams und fördert Reflexionen, ohne dabei Anschuldigungen zuzulassen.

(2) Als ScrumMaster beschützt er das Team vor negativen Einflüssen. Solche Einflüsse können persönliche Konflikte oder mangelnde Kommunikation innerhalb des Entwicklungsteams oder zwischen Entwicklungsteam und Product Owner sein, aber auch Störungen von außen, beispielweise Aufforderungen von anderen Stakeholdern neue Anforderungen in einem laufenden Sprint umzusetzen.

(3) Der ScrumMaster für ein Impediment Backlog, indem alle auftretenden Hindernisse schriftlich festgehalten werden. Die Hindernisse werden mit Datum des ersten Auftretens notiert, sowie das Datum der Beseitigung des Hindernisses.

(4) Neben der Arbeit mit dem ScrumTeam hilft der ScrumMaster auch Scrum in der Organisation richtig einzuführen. Er erklärt betroffenen Abteilungen Scrum und achtet darauf, dass die Scrum-Regeln eingehalten werden. Außerdem arbeitet er mit dem Management zusammen und sorgt dafür, dass die Organisation die durch Scrum entstehenden Änderungen annimmt (Change Management).

(5) Der ScrumMaster hat keine formale Macht, er ist der Coach für das Team – nicht Chef. Das fehlen einer Chefrolle ist notwendig für die Selbstorganisation des Teams und hilft das Vertrauen in den ScrumMaster aufzubauen.
(5.1) Ebenso wie der Product Owner ist auch der ScrumMaster kein Projektleiter.

LZ SCRUM-ROLLEN – ÜA

(1) Welche drei Rollen bilden das ScrumTeam? Wer gehört nicht dazu?

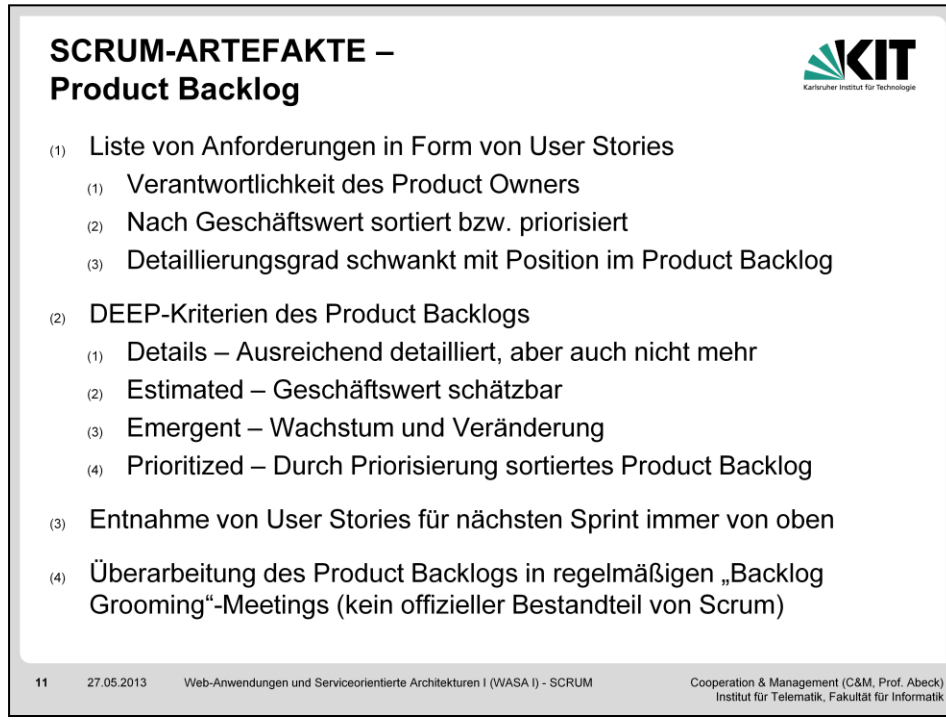
(2) Wer übernimmt (nicht) die Rolle eines Projektleiters in Scrum und warum (nicht)?

(3) Welche Rolle hat Weisungsbefugnisse (innerhalb eines Sprints) gegenüber dem Entwicklungsteam?

(4) Was mach eine gute Vision eines Produkts aus? Beispiel?

1984

heute



(1) In Scrum werden die Anforderungen an das Produkt in Form von User Stories in einer Liste namens „Product Backlog“ festgehalten. Die Liste verändert sich im Laufe der Produktentwicklung, insbesondere werden abgeschlossene User Stories aus dem Product Backlog entfernt und bestehende User Stories „durchwandern“ das Product Backlog nach oben.
(1.1) Die Hauptverantwortung für das Product Backlog liegt beim Product Owner.
(1.2) Das Product Backlog ist eine nach Priorität sortierte Liste. Der Product Owner definiert die Priorität anhand des Geschäftswerts einer User Story (Kundenzufriedenheit, Aufwand, Risiko, Kosten durch Verzögerung etc.). Die Priorisierung einer User Story kann durch veränderte Umstände auch vom Product Owner geändert werden.
(1.3) Eine User Story kann zunächst sehr grob gefasst sein, muss jedoch dann mit wachsender Position nach oben entsprechend in mehrere User Stories verfeinert werden. Daher sind die niedrig priorisierten User Stories häufig gröber gefasst.

(2) Die DEEP-Kriteren definieren ein gutes Product Backlog.
(2.1) Danach enthält ein Product Backlog „nur“ ausreichend Details, wie gerade benötigt werden.
(2.2) Ferner muss der Geschäftswert einer User Story geschätzt werden können.
(2.3) Das Product Backlog ist keine statische Liste, sondern wächst und verändert sich dynamisch.
(2.4) Durch die Priorisierung der User Stories ist das Product Backlog eine sortierte Liste.

(3) Scrum erlaubt nur die Entnahme der jeweils am höchsten priorisierten User Story aus dem Product Backlog für das Sprint Backlog. Eine Entnahme einer User Story aus der Mitte des Product Backlogs ist nicht zulässig, die Priorisierung kann aber vom Product Owner geändert werden.

(4) In sogenannten „Backlog Grooming“-Meetings wird das Product Backlog vom ScrumTeam regelmäßig überprüft und überarbeitet. Neue User Stories werden hinzugefügt, bestehende User Stories werden neu priorisiert und ggf. in mehrere User Stories verfeinert.

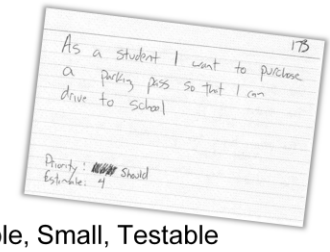
DEEP „Details, Estimated, Emergent, Prioritized“

User Stories

(1) Feature aus der Sicht des Users in Form ein oder zwei kurzer Sätze
  (1) Versprechen für eine Konversation
  (2) Überprüfbar

(2) „Wer“, „Was“ und „Warum“
  (1) Beispiel: „Als Administrator (Wer) möchte ich ein Benutzerkonto sperren (Was) können, um den Zugang zum System für den Benutzer zu verhindern, ohne den Benutzer zu löschen (Warum).“

(3) Weitere Bestandteile
  (1) Nummer (ID)
  (2) Schätzung
  (3) Akzeptanzkriterien (Definition of Done)



(4) INVEST-Kriterien für gute User Stories:
Independent, Negotiable, Valuable, Estimable, Small, Testable

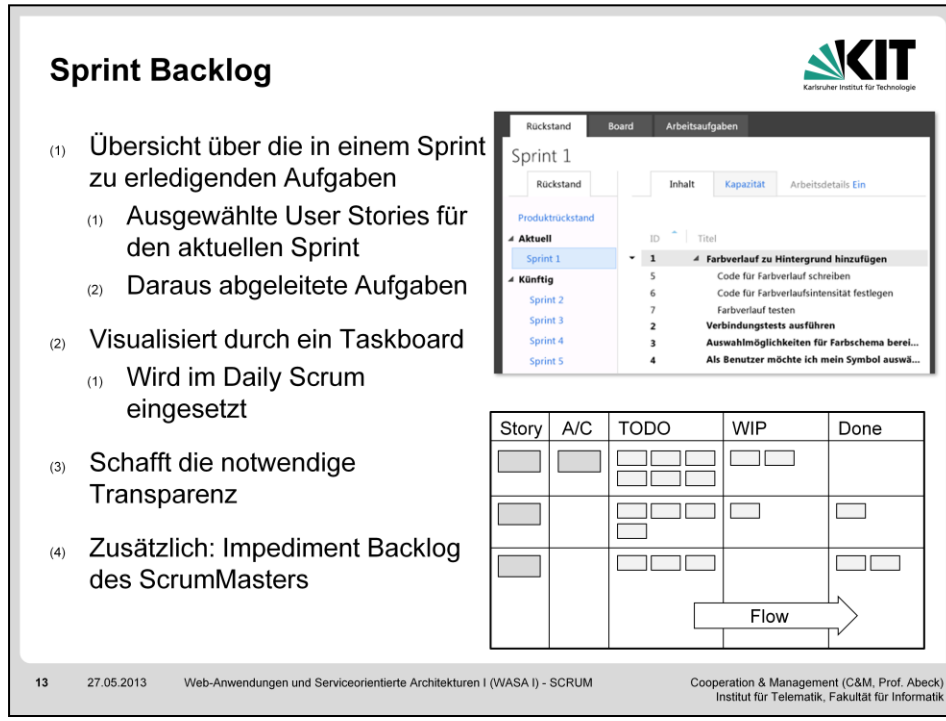
(1) User Stories beschreiben ein Feature aus der Sicht eines Users (Anwenders). Der Inhalt einer User Story kann zunächst sehr grob gefasst sein und eher einer Vision entsprechen (statt User Story als „Epic“ bezeichnet), wird dann aber beim Überarbeiten des Product Backlogs und somit beim „Aufsteigen“ kontinuierlich in mehrere User Stories aufgeteilt und verfeinert.
(1.1) Zunächst ist eine User Story eine kompakte Form einer Notiz, dessen Details in einer späteren Konversation zwischen allen Beteiligten (insbesondere zwischen Product Owner und Entwicklungsteam) festgelegt werden (üblicherweise im Sprint Planning II Meeting).
(1.2) Eine User Story sollte immer durch entsprechend vorher definierte Akzeptanzkriterien (Definition of Done) überprüfbar sein. Auf diese Weise kann nach der Umsetzung festgestellt werden, ob eine User Story erledigt (Done) ist im Sinne der ursprünglichen Anforderung.

(2) Ein bewährtes Schema für eine User Story ist die Form „Als <WER> möchte/muss ich <WAS>, um/sodass/weil <WARUM>.“ Der „Was“-Teil ist dabei das aus User-Sicht beschriebene Feature, der „Wer“-Teil ist die Rolle und der „Warum“-Teil liefert die Begründung oder den Hintergrund für das Feature (kann bei offensichtlichen Hintergrund auch entfallen).

(3) Weitere Bestandteile einer User Story sind eine Identifikationsnummer (insbesondere bei Nutzung von Scrum-Tools), eine Schätzung des Aufwands in einer imaginären Einheit („Story Points“) und Akzeptanzkriterien für die Umsetzung der User Story. Bei der Nutzung von Karteikarten werden die Akzeptanzkriterien häufig auf die Rückseite geschrieben.

(4) Im Zusammenhang mit agilen Entwicklungsmethoden gibt es das Akronym INVEST für gute User Stories. Danach sollten User Stories unabhängig (Independent) sein und sich nicht konzeptionell überlappen. Sie müssen sich unabhängig voneinander implementieren und testen lassen, damit der Product Owner die Priorisierung nach seinem Ermessen vornehmen kann. User Stories sollten auch verhandelbar bleiben (Negotiable), da sie kein Vertrag sind, sondern eher die Essenz der Anforderung einfangen. Details einer User Story werden zwischen Product Owner und Entwicklungsteam bei der konkreten Umsetzung besprochen. User Stories sollen einen Mehrwert für das Geschäft bieten (Valuable). Außerdem ist der Aufwand schätzbar (Estimable). Sie sind in höheren Positionen im Product Backlog hinreichend klein, um sie in einem Sprint zu erledigen (Small) und sie sind testbar, so das Akzeptanztests definiert werden können (Testable).

INVEST „Independent, Negotiable, Valuable, Estimable, Small, Testable“



(1) Das Sprint Backlog enthält die innerhalb eines Sprints umzusetzenden User Stories aus dem Product Backlogs und die aus den ausgewählten User Stories abgeleiteten Aufgaben. Das Sprint Backlog wird während des Sprint Planning I Meetings erstellt.

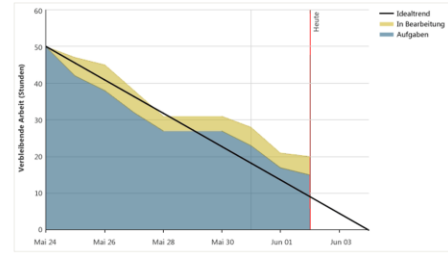
(2) Das Taskboard kann elektronisch mittels eines Werkzeuges oder klassisch mit Karteikarten an einem Whiteboard umgesetzt sein. Im Wesentlichen werden die User Stories aus dem Sprint Backlog in der ersten Spalte angehängt. Die weiteren Spalten enthalten die daraus abgeleiteten Aufgaben entsprechend in der Spalte ihres Status. Eine Aufgabe hat die drei Status „zu erledigen“ (To Do), „in Arbeit“ (Work in Progress) und „erledigt“ (Done). Gelegentlich werden auch die Akzeptanzkriterien noch im Taskboard in einer separaten Spalte aufgeführt.
(2.1) Im Laufe eines Sprints „wandern“ die Aufgaben durch das Taskboard von links nach rechts („Flow“) und erlauben so insbesondere während des Daily Scrums eine realistische Sicht auf den aktuellen Status des Teams und wie weit das Sprint-Ziel noch entfernt ist.

(3) Das Sprint Backlog und das Taskboard schaffen Transparenz, da jedes Mitglied des Teams sich umfassend über den aktuellen Status informieren kann und mögliche Risiken bei der Erfüllung des Sprint-Ziels bereits frühzeitig sichtbar werden und entsprechend reagiert werden kann.

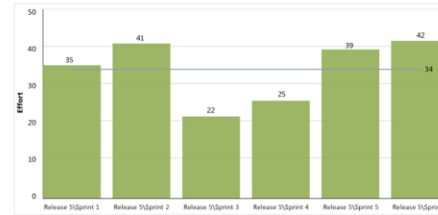
(4) Der ScrumMaster führt ein separates Impediment Backlog, in dem alle während des Daily Scrum aufgetretenen Hindernisse und Blockaden aufgeführt werden.

Burndown- und Velocity-Charts

(1) Burndown-Chart zeigt innerhalb eines Sprints den zu leistenden Restaufwand (Entfernung zum Sprint-Ziel)
  (1) Summe der nicht abgeschlossenen Aufgaben des aktuellen Sprints



(2) Velocity-Chart zeigt die Leistungsfähigkeit des Teams in jedem Sprint
  (1) Summe der „Story Points“ aller abgeschlossenen User Stories



14 27.05.2013 Web-Anwendungen und Serviceorientierte Architekturen I (WASA I) - SCRUM Cooperation & Management (C&M, Prof. Abeck) Institut für Telematik, Fakultät für Informatik

(1) Das Burndown-Chart ist ein Grafik, welche die noch zu leistende Arbeit in einem Sprint zur Erreichung des Sprint-Ziels aufzeigt. Die horizontale Achse ist die Zeitachse und zeigt in der Regel die Arbeitstage des Sprints. Die vertikale Achse sind üblicherweise die Arbeitsstunden. Eine diagonale Linie zeigt den Idealverlauf des Sprints an und erlaubt somit eine Einschätzung der aktuellen Performance des Teams. Das Burndown-Chart zeigt dem Team ob das Erreichen des Sprint-Ziels möglich oder bereits gefährdet ist. Daher ist das Burndown-Chart neben dem Taskboard ein wichtiges Werkzeug, was bei jedem Daily Scrum eingesetzt werden sollte.

(2) Im Velocity-Chart wird die Leistungsfähigkeit des Teams in den bereits abgeschlossenen Sprint gezeigt. Hierbei werden die abgeschlossenen User Stories in „Story Points“ je Sprint aufsummiert und als Balken dargestellt. Eine horizontale Linie zeigt den Mittelwert an. Ein gut eingespieltes ScrumTeam sollte möglichst eine konstante Leistungsfähigkeit aufzeigen. Das bedeutet dass das Velocity-Chart nur geringe Schwankungen (Varianz) aufweisen sollte. Das Velocity-Chart kann bei der Planung eines neuen Sprints helfen, indem die Leistungsfähigkeit des Teams aufgrund der Erfahrung aus vorherigen Sprints richtig eingeschätzt werden kann.

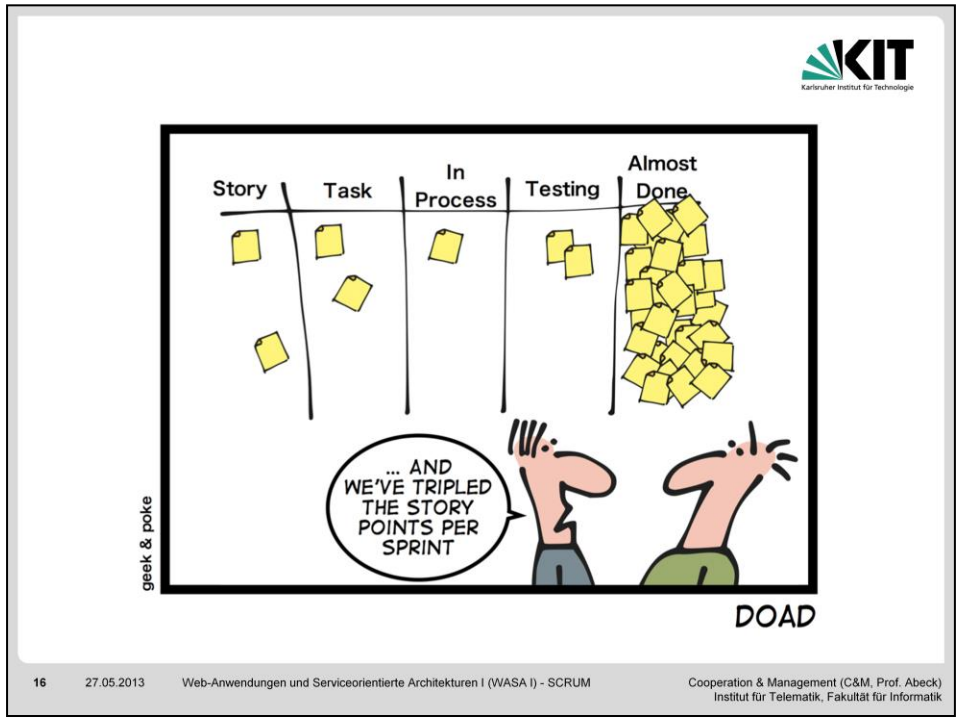
LZ SCRUM-ARTEFAKTE – ÜA

(1) Anhand welcher Kriterien wird das Product Backlog sortiert?

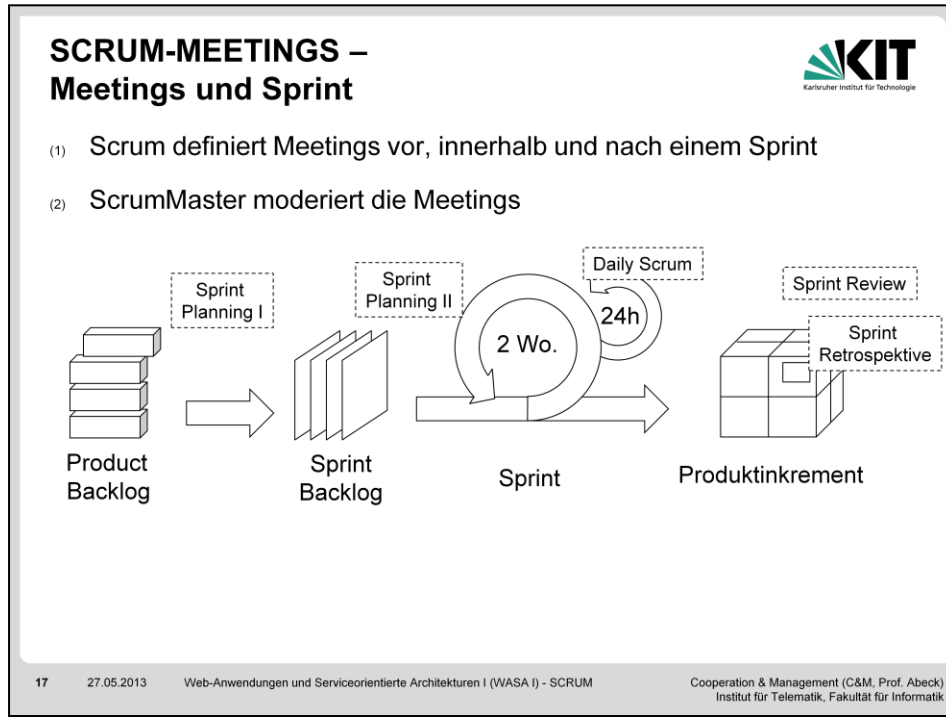
(2) Was sollte eine User Story wiedergeben?

(3) Was ist/sind „Definition of Done“?

(4) Welche Hilfsmittel werden eingesetzt, um den Fortschritt während der Entwicklung zu visualisieren?

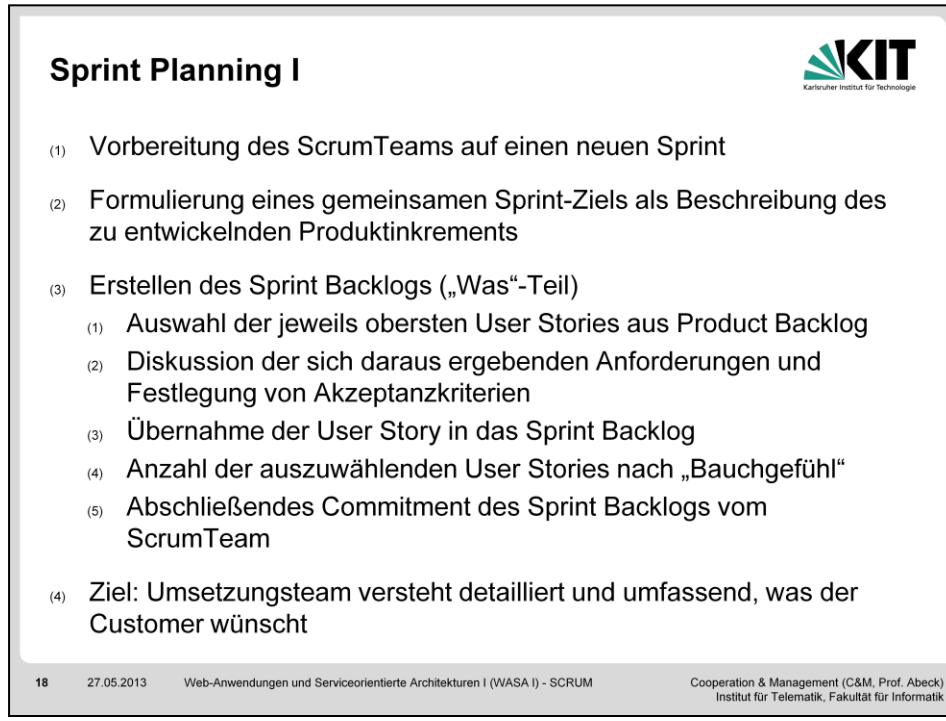


http://www.geek-and-poke.com/2012/10/doad.html



(1) Scrum definiert insgesamt fünf verschiedene Meetings. Das Sprint Planning Meeting findet vor Beginn eines neuen Sprints statt, häufig im Beisein des Kunden und dient dazu die notwendigen Vorbereitungen für den erfolgreichen Abschluss eines Sprints zu treffen. Das Sprint Planning Meeting zerfällt dabei in zwei Teile. Während eines Sprints findet täglich das kurze Daily Scrum Meeting (oder einfach nur Daily Scrum) statt, um sich gegenseitig auf den aktuellen Stand zu bringen. Nach Abschluss eines Sprints werden die erzielten Ergebnisse im Sprint Review zusammen mit dem Kunden analysiert und ggf. Änderungen für neue Sprints beschlossen. Die folgende Sprint Retrospektive dienst als interne Prozessanalyse und -verbesserung. Aufgetretene Probleme und Hindernisse werden aufgedeckt, besprochen und gelöst bzw. beseitigt.

(2) Als Team Coach ist es Aufgabe des Scrum Masters die Meetings zu moderieren. Entsprechend werden vom Scrum Master „Social Skills“ im besonderen Maße erwartet.



(1) Das Sprint Planning Meeting I wird vor dem Start eines neuen Sprints durchgeführt.

(2) Im Sprint Planning Meeting I soll gemeinsam das Ziel des kommenden Sprints bestimmt werden. Hierzu stellt der Product Owner dem Entwicklungsteam die User Stories aus dem Product Backlog vor und legt die Kriterien für die erfolgreichen Umsetzung fest.

(3) Im Kern des Meetings steht das Erstellen des Sprint Backlogs. Das Sprint Backlog besteht aus den ausgewählten User Stories, die im kommenden Sprint umgesetzt werden sollen. Das Sprint Backlog beschreibt also was umgesetzt wird, aber nicht wie die Umsetzung erfolgt (dies wird im Sprint Planning Meeting II festgelegt).
(3.1) Zunächst wählt das Team die oberste User Story aus dem vom Product Owner priorisierten Product Backlog aus. Es besteht dabei keine Wahlmöglichkeit.
(3.2) Dann wird die User Story mit dem Product Owner so lange besprochen, bis dem Entwicklungsteam alle Details und Anforderungen dazu klar sind. Hierzu gehört auch die gemeinsame Festlegung von Akzeptanzkriterien, also welche Bedingungen erfüllt sein müssen, damit eine User Story als erfolgreich umgesetzt gilt.
(3.3) Die User Story wird dann in das Sprint Backlog übernommen.
(3.4) Wenn das Team in dem zu planenden Sprint noch freie Kapazitäten sieht, kann die nächste oberste User Story aus dem Product Backlog ausgewählt werden. Hierbei agiert das Team ohne Druck von außen.
(3.5) Abschließend geht das Team das Sprint Backlog noch einmal hinsichtlich einer realistischen Möglichkeit zur Umsetzung bis zum Sprint-Ende durch und gibt dann eine Art „Versprechen“ ab, das Sprint-Ziel zu erreichen.

(4) Das Umsetzungsteam hat am Ende des Meetings ein umfassendes Verständnis über die Wünsche des Kunden. Alle Unklarheiten sollten beseitigt und offene Fragen sollten beantwortet sein.

Sprint Planning II

(1) Folgt am gleichen Tag auf das Sprint-Planning-I-Meeting

(2) Klärung des Designs und Entwurfs des Produktinkrements

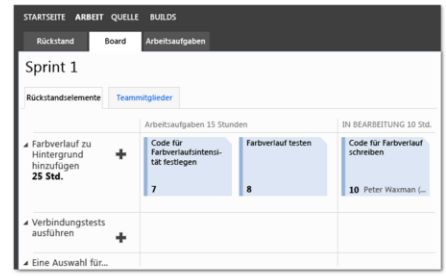
(3) Erstellung von feingranularen Aufgaben anhand des Sprint Backlogs („Wie“-Teil)

  (1) Entwicklung einer grobe Idee zur Umsetzung

  (2) Ableitung von Aufgaben anhand des Entwurfs und Sammlung in einem sortierten Taskboard

  (3) Abschätzen des Aufwands für die Aufgaben

(4) Ziel: Umsetzungsteam hat einen Plan, wie das Produktinkrement erstellt wird und das Sprint-Ziel erreicht werden kann



(1) Das Sprint Planning Meeting II folgt in der Regel direkt auf das Sprint Planning Meeting I, zumindest sollte es aber noch am gleichen Tag durchgeführt werden.

(2) Während das erste Meeting eher der Anforderungsanalyse entspricht, werden im Sprint Planning II Meeting nun auch Design und Entwurfsentscheidungen gemeinsam festgelegt, die das Produktinkrement betreffen.

(3) Für die User Stories im Sprint Backlog muss festgelegt werden, wie die Umsetzung zu erfolgen hat. Hierzu werden feingranulare Aufgaben erstellt, die innerhalb eines Tages von einem Entwickler erledigt werden können.
(3.1) Vor der Erstellung der Aufgaben sind entsprechende Design- und Entwurfsaspekte bei der Umsetzung zu diskutieren und festzulegen. Typische Aspekte sind Komponenten und deren Schnittstellen, externe Systeme, Datenquellen, Protokolle, automatisierte Tests etc.
(3.2) Aus der Diskussion sollte eine Idee der Umsetzung entstanden sein, aus welcher sich nun entsprechende Aufgaben ableiten lassen. Diese werden im Taskboard gesammelt.
(3.3) Der Aufwand für die Erfüllung jeder Aufgabe sollte grob (in Stunden) geschätzt werden.

(4) Das Umsetzungsteam hat am Ende des Meetings nun eine genaue Vorstellung, wie das Produktinkrement erstellt bzw. umgesetzt wird und wie das Sprint-Ziel erreicht werden kann.

Daily Scrum

(1) Findet während des Sprints täglich mit einer max. Dauer von 15 Minuten statt (sollte im stehen abgehalten werden)

(2) Nacheinander beantwortet jedes Teammitglieder mit Hilfe des Taskboards drei Fragen
  (1) Welche Aufgaben habe ich bearbeitet und wie ist deren Status?
  (2) Was habe ich mir heute vorgenommen?
  (3) Was blockiert oder behindert meine Arbeit?

(3) Keine Diskussionen oder Problemlösungen!
  (1) ScrumMaster achtet auf die Einhaltung dieser Regelungen

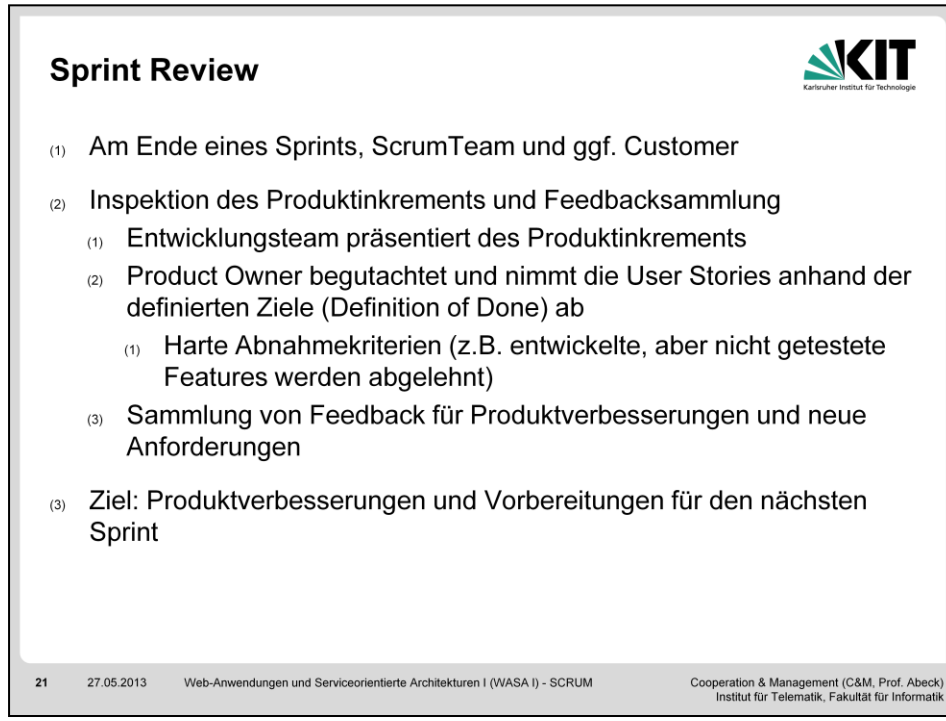
(4) Ziel: Informationsaustausch und Aufsynchronisierung

(1) Das Daily Scrum findet an jedem Arbeitstag zur selben Zeit und am selben Ort statt. Es sollte nicht länger als 15 Minuten dauern und am besten im Stehen abgehalten werden.

(2) Mit Hilfe des Taskboards teilt jedes Team drei Aspekte in kurzer Form dem restlichen Team mit. Hierzu zählen die seit dem letzten Daily Scrum abgearbeiteten Aufgaben und deren aktueller Status und die neuen Aufgaben, welche sich das Teammitglied bis zum nächsten Daily Scrum vorgenommen hat. Außerdem sollte mögliche Blockaden oder Hindernisse bei der Arbeit deutlich gemacht werden werden, die Umsetzung einer Aufgabe verhindern. Der ScrumMaster notiert sich diese Hindernisse im Impediment Backlog und versucht diese später zu lösen.

(3) Im Daily Scrum selbst finden keine Problemlösungen oder Diskussionen statt. Die Teammitglieder berichten sich nur gegenseitig kurz und knapp.
(3.1) Sollten Probleme sichtbar werden, können diese in weiteren separaten Meeting geklärt werden. Der ScrumMaster moderiert dazu das Daily Scrum entsprechend und achtet auf die Einhaltung der Daily-Scrum-Regeln.

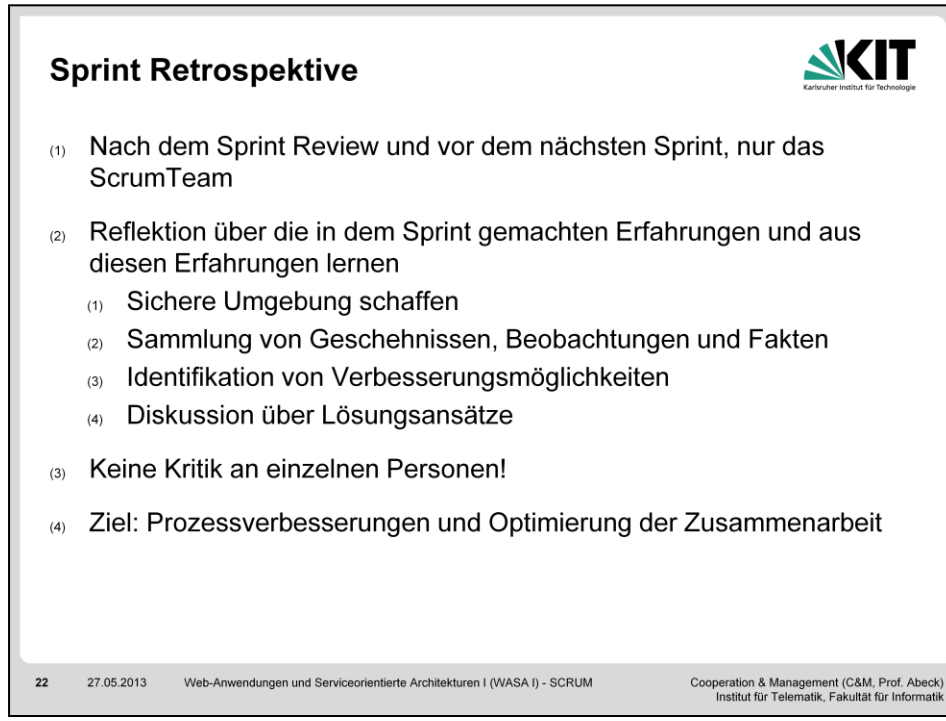
(4) Das Daily Scrum dient als Inspektions- und Adaptionspunkt und ist für die Selbstorganisation des Umsetzungsteams notwendig. Es schafft die geforderte Transparenz, da jedes Teammitglied nun über die Tätigkeiten der anderen Teammitglieder informiert ist. Außerdem sind mögliche Blockaden und Hindernisse identifiziert worden.



(1) Das Sprint Review Meeting findet nach Abschluss des Sprints statt. Teilnehmen können neben dem ScrumTeam auch andere Stakeholder, insbesondere der Kunde ist ein häufiger Teilnehmer im Sprint Review.

(2) Das Ziel des Sprint Reviews ist zum Einen die Präsentation des Produktinkrements und zum Anderen die Sammlung von Feedback zu dem Produktinkrement.
(2.1) Der erste Teil besteht aus der Präsentation des Produktinkrements vom Entwicklungsteam. Das Produktinkrement muss dabei auf eine realistischen Umgebung vorgeführt werden und das Entwicklungsteam muss zeigen, dass die Anforderungen vollständig erfüllt sind.
(2.2) Der Product Owner begutachtet das Produktinkrement und entscheidet, ob die Anforderungen in Form der User Stories korrekt umgesetzt wurden. Hierbei werden die zu Beginn des Sprints definierten Abnahmekriterien angelegt.
(2.3) Nach der Präsentation ist es für das Entwicklungsteam wichtig möglichst genaues und umfassendes Feedback vom Product Owner und anderen Stakeholdern zu bekommen.

(3) Durch das Feedback werden Produktverbesserungen und neue Anforderungen identifiziert, welche vom Product Owner ggf. direkt in das Product Backlog übernommen werden.



(1) Nach dem Sprint Review folgt das Sprint Retrospektive Meeting. Dieses Meeting besteht nur aus dem ScrumTeam selbst, andere Teilnehmer sind nicht zugelassen.

(2) Hauptzweck des Sprint Review Meetings ist die (Selbst)reflektion des ScrumTeams über die im Sprint gewonnenen Erfahrungen und Erkenntnissen. Dabei sollen auf Probleme offen angesprochen und geklärt werden.
(2.1) Das Meeting sollte in einer geschützten Umgebung (z.B. geschlossener Raum) erfolgen, um offen über die gemachten Erfahrungen diskutieren zu können.
(2.2) Zunächst werden die im letzten Sprint gemachten Beobachten und Probleme gesammelt. Das Impediment Backlog des ScrumMaster kann hierbei sinnvoll genutzt werden.
(2.3) Durch eine offene und ehrliche Diskussion werden nur gemeinsam die Probleme gelöst und Verbesserungsmöglichkeiten für den nächsten Sprint identifiziert.
(2.4) Die möglichen Lösungsansätze werden zusammen besprochen und das Team versucht die Lösungen im nächsten Sprint umzusetzen.

(3) Das Spring Retrospektive dient nicht dazu einzelne Personen zu kritisieren oder einen Schuldigen zu suchen. Das ScrumTeam agiert als Einheit und ist somit auch gemeinsam für Erfolge und gemeinsam für Fehlschläge verantwortlich.

(4) Ziel der Sprint Retrospektive ist die Verbesserung der Umsetzung des Scrum-Prozesses durch eine bessere gemeinsame Zusammenarbeit und die Schaffung eines produktiven, aber angenehmen Arbeitsklimas.

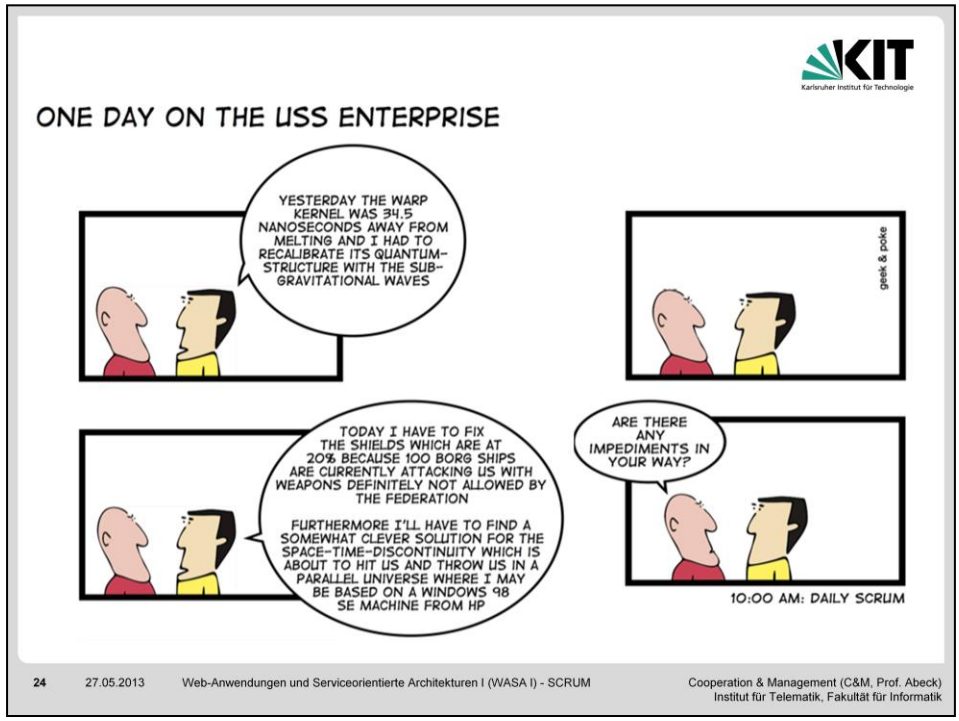
LZ SCRUM-MEETINGS – ÜA

(1) Was ist der (grobe) Unterschied zwischen Sprint Planning Meeting I und II?

(2) Was ist der (grobe) Unterschied zwischen Sprint Review und Retrospektive?

(3) Was ist Sinn und Zweck des Daily Scrums?

(4) Wozu dient das Daily Scrum nicht?



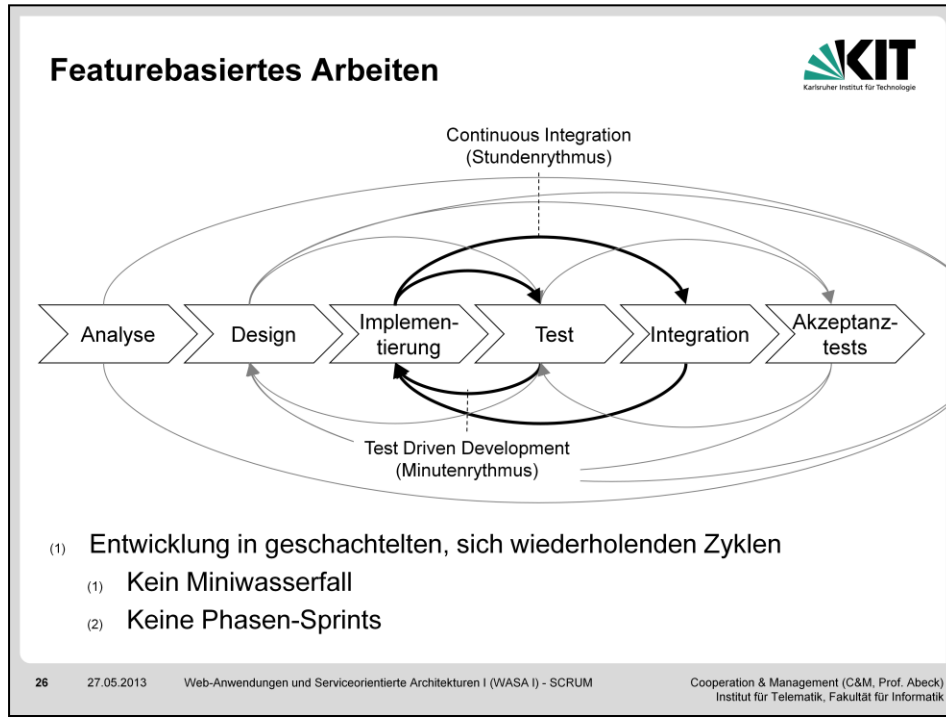
http://www.geek-and-poke.com/2011/11/one-day-on-the-uss-enterprise.html

(1) Alle Mitglieder des Entwicklungsteams sollen die technische Kompetenz haben sauberen und ausreichend dokumentierten Quellcode zu entwickeln. Entsprechende Stilvorgabe, durch „Code Conventions“ festgelegt, müssen eingehalten werden. Außerdem umfasst dies auch das Entwerfen einer sauberen Architektur, umfangreiche Kenntnisse über Entwurfsmuster, den korrekten Einsatz des objektorientierten Paradigma und vieles mehr. Das vernachlässigen dieser Praktiken („Quick‘n‘Dirty“, „Hacks“ etc.), insbesondere bei Zeitknappheit zum Release hin, führt später meistens zu einem entsprechenden zu leistenden Mehraufwand (technische Schulden).

(2) Das Refactoring (Refaktorisierung) sind Praktiken, um das Design einer Software änderbar und flexibel zu halten. Die ständige Refaktorisierung sind für ein gutes Design der Software unverzichtbar. Daher sollte Refaktorisierung insbesondere vom Management nicht als ein großes teures Vorhaben aufgefasst werden ohne einen Geschäftswert, sondern als Bestandteil der täglichen Arbeit eines Entwicklers permanent in entsprechenden kleinen Schritten durchgeführt werden.

(3) In der testgetriebenen Entwicklung wird nicht nur laufend getestet, sondern vor allem noch vor der eigentlichen Implementierung. Bevor eine neue Methode, Funktion oder Klasse entwickelt wird, muss zunächst ein oder mehrere entsprechende Unit-Tests geschrieben werden (rote Phase). Diese Unit-Tests schlagen zunächst fehl bzw. sind überhaupt nicht zu kompilieren. Erst dann ist es gestattet die eigentliche Funktionalität zu implementieren (grüne Phase). Der Umfang der Implementierung sollte dabei nur so umfangreich sein, um den Test gerade erfolgreich zu bestehen. Im weiteren Verlauf kann dann wieder erst der Test und nachfolgend die Implementierung in kleinen Schritten erweitert werden. Dieser Zyklus zwischen Implementierung und Test wiederholt sich dabei häufig im Rhythmus weniger Minuten.

(4) Als kontinuierliche Integration wird die ständige Zusammenführung der Arbeiten aller Teammitglieder verstanden. Das Zusammenführen umfasst dabei nicht nur das regelmäßige Einchecken des Codes in ein Versionsverwaltungssystem in kurzen Zyklen, sondern auch die permanenten Kontrolle der Funktionsfähigkeit der Software durch entsprechende automatisierte Testumgebungen. Der eingecheckte Code wird automatisch kompiliert, analysiert und auf möglichst realistischen Laufzeitumgebungen auf Funktionsfähigkeit getestet. Das Team erhält so kurzfristig Feedback über den aktuellen Zustand des sich in der Entwicklung befindlichen Systems und kann entsprechend reagieren.



(1) In der Entwicklung innerhalb eines Sprints müssen alle Aktivitäten, die aus klassischen Vorgehensmodellen bekannt sind ausgeführt werden, jedoch in kürzeren Abständen. Daher sollte die Entwicklung idealerweise in sich geschachtelten und wiederholenden Zyklen ablaufen. Der innerste und schnellste Zyklus ist der bei der testgetriebenen Entwicklung schnelle Wechsel zwischen Implementierung und Test. Darauf folgen die Zyklen der kontinuierlichen Integration und damit einhergehend auch Akzeptanz- und Regressionstests (weitere Zyklen). Ein gemeinsames Design entsteht nicht nur während der Entwicklung, sondern auch in täglichen Zusammentreffen und im Sprint Planning Meeting II. Der äußerste Zyklus der Analyse sind zunächst die Tätigkeiten des Product Owners, sowie die regelmäßig gemeinsamen Überarbeitungen des Product Backlogs (Backlog Grooming).

(1.1) Die Aktivitäten während der Entwicklung sollten im Sprint nicht als Mini-Wasserfall durchgeführt werden, also streng nacheinander ablaufen. Es gibt in Scrum keine offiziellen Übergaben, beispielsweise zwischen Design und Implementierung.

(1.2) Ebenso sollte ein Sprint immer die komplette Bandbreite der Softwareentwicklung umfassen, um ein möglichst vollständiges vertikales Inkrement auszuliefern. Beispielsweise steht ein Sprint nur zur Entwicklung, gefolgt von einem Sprint nur zum Testen im Widerspruch zu den Prinzipien der Transparenz und Inspektion.

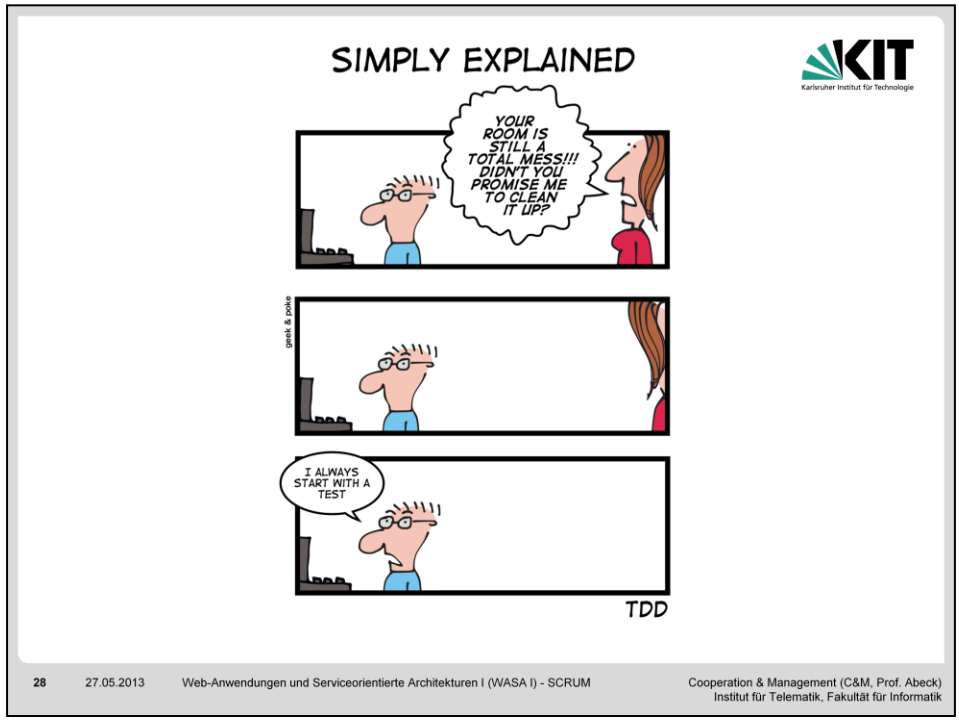
LZ ARBEITEN IN SPRINTS – ÜA

(1) Welche agilen Entwicklungspraktiken werden von Scrum definiert?

(2) Was ist testgetriebene Entwicklung?

(3) Was ist kontinuierliche Integration?

(4) Was gilt es betreffend des Entwicklungsvorgehens in einem Sprint zu vermeiden, was in anderen Vorgehensmodellen Standard ist?



http://www.geek-and-poke.com/2012/09/simply-explained-tdd.html

Literatur

[Gl08] Boris Gloger: Scrum – Produkte zuverlässig und schnell entwickeln, Carl Hanser Verlag München, 2. Auflage, 2008.

[SB02] Ken Schwaber, Mike Beedle: Agile Software Development with Scrum, Prentice Hall, 2002.

[SS11] Ken Schwaber, Jeff Sutherland: Scrum Guide, http://www.scrum.org/Scrum-Guides, 2011.

[Wi12] Andreas Wintersteiger: Scrum Schnelleinstieg, entwickler.press, Software & Supprt Media Verlag Frankfurt, 2012.