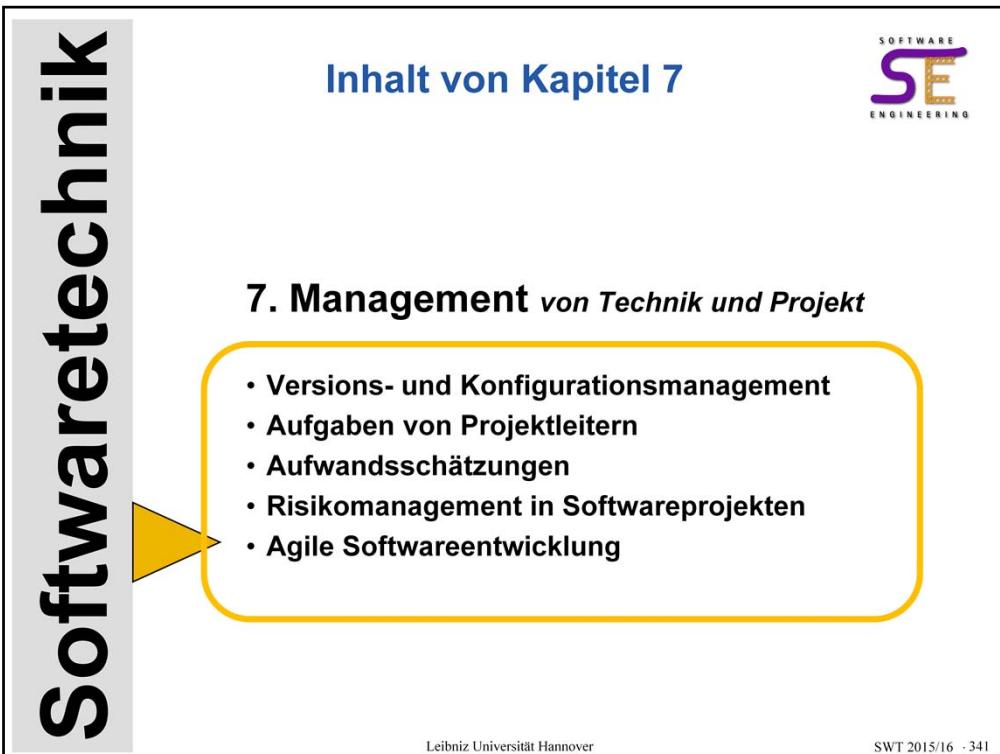


Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



The slide template features a vertical grey sidebar on the left with the word "Softwaretechnik" in large, bold, black font. A yellow triangle points from the sidebar towards the main content area. The main content area has a white background with a thin black border. At the top, the title "Inhalt von Kapitel 7" is centered in blue font. In the top right corner is the logo "SOFTWARE ENGINEERING" with "SE" in a stylized purple and yellow font. Below the title, the chapter heading "7. Management von Technik und Projekt" is in bold black font. A yellow rounded rectangle contains a bulleted list of five items: "Versions- und Konfigurationsmanagement", "Aufgaben von Projektleitern", "Aufwandsschätzungen", "Risikomanagement in Softwareprojekten", and "Agile Softwareentwicklung". The bottom of the slide contains two small text blocks: "Leibniz Universität Hannover" on the left and "SWT 2015/16 · 341" on the right.

Inhalt von Kapitel 7

7. Management von Technik und Projekt

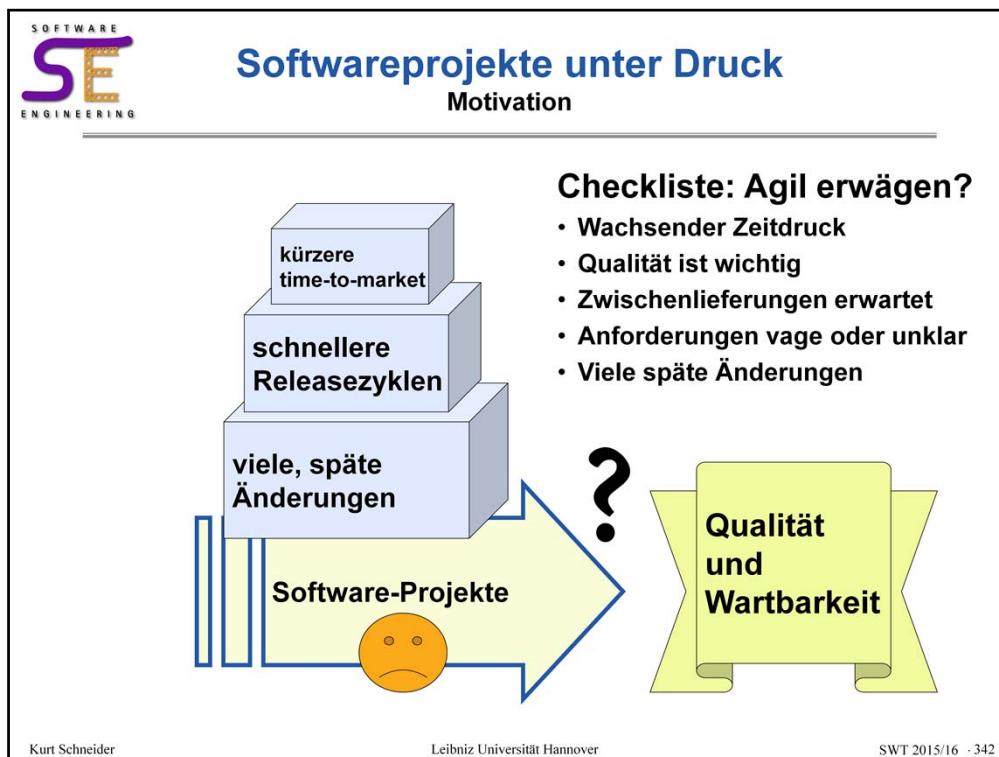
- Versions- und Konfigurationsmanagement
- Aufgaben von Projektleitern
- Aufwandsschätzungen
- Risikomanagement in Softwareprojekten
- Agile Softwareentwicklung

Leibniz Universität Hannover

SWT 2015/16 · 341

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Unter dem zusätzlichen Druck (blau) wird es für SW-Projekte in den letzten immer schwieriger, das Mindestmaß an Qualität und Wartbarkeit zu erzeugen.

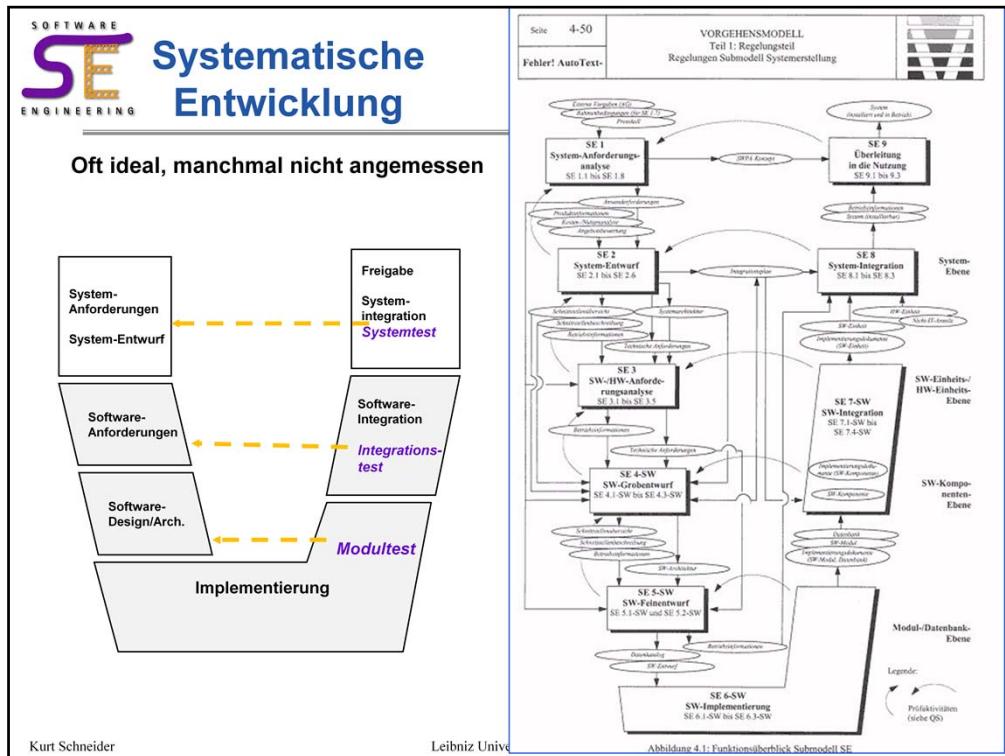
Kurze Entwicklungszeiten, und zwar bei jedem Release wieder, lasten schwer auf den Projekten und den Entwicklern. Am schwersten wirkt sich aus, dass diese besonders innovative Art von Projekten auch den Kunden keine Zeit lässt, ihre Anforderungen vollständig zu verstehen. Daher sind Änderungen unvermeidlich. Sie ergeben sich oft schon daraus, dass Kunden erst nach und nach verstehen, wie sie die Software einsetzen können und wollen.

Daher müssen die Projekte einfach darauf reagieren und können nicht stur auf den ursprünglichen Anforderungen beharren. Es nützt dem Kunden – und damit auch den Entwicklern – wenn die Software den Wünschen des Kunden am Projektende entspricht. Dann ist er zufriedener, und die Entwickler werden wahrscheinlicher wieder einen neuen Auftrag bekommen. Es nützt also allen, wenn es gelingt, hier flexibler zu bleiben.

Wenn man abschätzen möchte, ob man agil arbeiten oder zumindest agile Elemente einbauen will, dann kann man sich die Fragen rechts oben stellen. Je mehr Punkte aufs eigene Projekt zutreffen, desto mehr spricht dafür, sich agile Ansätze anzusehen. Je weniger Punkte zutreffen, desto besser ist man mit reinen systematischen Vorgehensweisen aufgehoben. Achtung: Es wäre ein Missverständnis, agile Methoden als chaotisches Vorgehen ohne Systematik zu betrachten. Das System ist nur anders. Das wird nun angesprochen.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Prozessmodelle wie das V-Modell haben sich sehr bewährt: dadurch sind reife Unternehmen entstanden, man konnte systematische Vorgehensweisen etablieren. Allerdings sind die Regelungen durch das vollständige V-Modell (und viele ähnliche Vorgaben) oft sehr umfangreich und belastend, gerade für kurze, innovative und anfangs noch nicht ganz geklärte Projekte.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

The diagram is titled "Traditionelle Softwareentwicklung". It features two columns: "Stärken +" (Strengths +) on the left and "Probleme -" (Problems -) on the right. The "Stärken +" column lists seven points: Systematisch, nachvollziehbar; Gut dokumentiert; Saubere Spezifikation; UML-Modelle; Projektziele früh klar; Detaillierte Planung, Vorgaben, Abläufe. The "Probleme -" column lists six points: Relativ langsam; Viel Aufwand; Dokumente veralten oft, bevor man sie braucht; Anforderungen oft missverstanden; Änderungen aufwändig.

Stärken +

- Systematisch, nachvollziehbar
- Gut dokumentiert
- Saubere Spezifikation
- UML-Modelle
- Projektziele früh klar
- Detaillierte Planung, Vorgaben, Abläufe

Probleme -

- Relativ langsam
- Viel Aufwand
- Dokumente veralten oft, bevor man sie braucht
- Anforderungen oft missverstanden
- Änderungen aufwändig

Kurt Schneider
Leibniz Universität Hannover
SWT 2015/16 · 344

Die systematischen und reifen Verfahren haben sich im SE sehr bewährt. Über ein Jahrzehnt hinweg haben viele Unternehmen in der ganzen Welt viel investiert, um „reifer“ zu werden – also Erfolge wiederholen zu können, ohne dabei einzige und allein auf hervorragende Mitarbeiter angewiesen zu sein. Natürlich hilft es, wenn Mitarbeiter hoch qualifiziert sind und mitdenken. Aber systematische Arbeitsweise und gut dokumentierte Prozesse können auch dann helfen, wenn man es mit „normalen“ Mitarbeitern zu tun hat, oder wenn im Team jemand kündigt.

Aber die reifen, systematischen Verfahren verlangen einige Zeit und einen Aufwand. Viele moderne, kurze und riskante (d.h. auch: chancenreiche) Projekte müssen schneller fertig werden, müssen auch veränderliche Kundenanforderungen berücksichtigen. Dazu braucht man eine angepasste Arbeitsweise.

Generell kann man sagen: Wer in diese zweite, agile Stufe einsteigen will, sollte zuvor die Grundlage der systematischen Arbeitsweise verstanden und ihre Vorteile erfahren haben. Dann bieten die agilen Ansätze die Chance, die Vor- und Nachteile bewusst abzuwägen und zu einer passenden Mischung zu kommen.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



The slide displays the Agile Manifesto, featuring the Software Engineering logo at the top left. The title "Manifesto for Agile Software Development" is centered above a yellow rounded rectangle containing the manifesto's core values. Below the values, a quote from Kent Beck et al. is presented, followed by the names of the original authors and contributors.

Manifesto for Agile Software Development

www.agilemanifesto.org

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

**Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan**

That is, while there is value in the items on the right, we value the items on the left more."

Kent Beck, Alistair Cockburn
Martin Fowler, Jim Highsmith, Ron Jeffries
Ken Schwaber und 11 andere

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 345

Nachdem einige Prinzipien und Praktiken vorgestellt wurden, hier noch die Werte der agilen Bewegung. Sie sind im Agile Manifesto zusammengestellt, das immer noch im Web steht. Im Kasten steht der Kern, die sogenannten Werte.

Bitte beachten: Am besten, man hat tolle Dokumentation UND laufende Software. Wenn aber nicht beides zu haben ist, dann ziehen die Autoren die linke Seite vor; bei allen vier Werten.

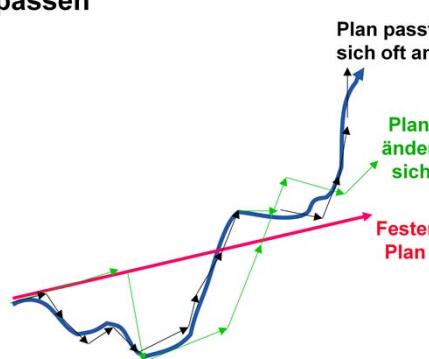
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

**Zusammenfassung
Angemessener Planungshorizont**

- Grober Langfristplan; nur auf kurze Frist genau planen
- Defer Commitment: ständig anpassen
- Lernen und Verbessern

- Gleichmäßiger Durchfluss
- Und geringe Aufgabenzahl
- Führt zu gutem Durchsatz, großer Geschwindigkeit



Plan passt sich oft an

Plan ändert sich

Fester Plan

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 346

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Agile Prinzipien Auswahl

- Einfachheit ist wichtigstes Konstruktionsprinzip
 - Möglichst wenig unnötige Arbeit, nicht auf Vorrat arbeiten
 - Inkrementelle Entwicklung, kurze Releases, wenig Dokum.
- Höchste Priorität: Kundenzufriedenheit
 - durch frühe und häufige Auslieferung laufender Software
 - Enge Kundeneinbindung, Planungssitzung
- Auch späte Änderungen werden willkommen geheißen
 - Änderungen bedeuten Verbesserungen für den Kunden
 - Ständig testen, integrieren, vereinfachen; Pair Programming

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 347

Agile Methoden definieren sich durch Werte (später gezeigt), Prinzipien, nach denen man sich ständig orientiert (oben) und einzelnen Praktiken. Das sind die konkreten Regeln, denen man jeden Tag folgt.

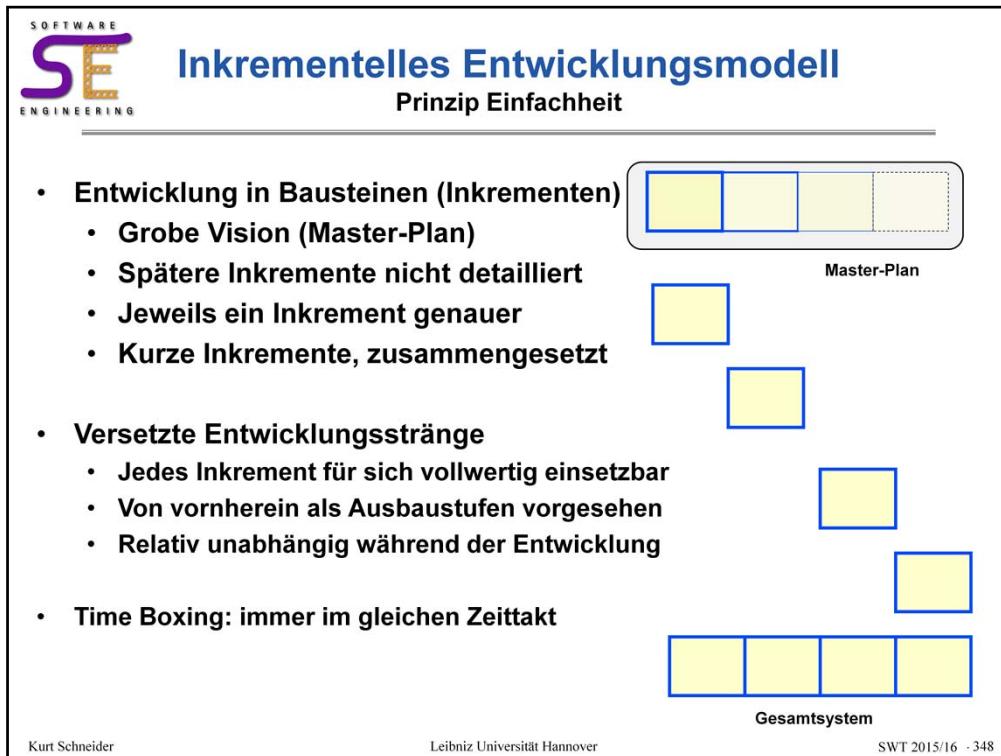
Oben sind drei sehr wichtige Prinzipien ausgewählt.

Zuerst steht jeweils das Prinzip. Dann, in blau, eine kurze Erklärung, was das bedeutet. Und anschließend, hinter dem Pfeil, einige Praktiken, die zur Umsetzung des Prinzips nötig sind. Die hier genannten Aspekte werden im Folgenden kurz angesprochen.

Es ist kaum möglich, agile Vorgehensweisen – und überhaupt echte Projekterfahrung – in einer Vorlesung theoretisch zu vermitteln. Daher wird hier nur der Überblick geboten, den man als Voraussetzung braucht. Später im Studium haben Sie Gelegenheit, diese Erfahrung selbst zu sammeln.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



In Agilen Methoden wird in kleinen Stücken, den sogenannten Inkrementen, entwickelt. Man blickt zwar zu Projektbeginn einmal bis zum möglichen Ende, plant aber nicht bis ins letzte Detail: Das würde sich sowieso wieder ändern und viel Arbeit wäre umsonst.

Aber das nächste Inkrement geht man an und entwickelt darin einen Baustein, den man mit den anderen zusammensetzt. So entstehen immer größere Versionen des Produkts, jede lauf- und einsatzfähig, aber mit immer größerem Funktionsumfang.

Was dabei genau getan wird, unterscheidet sich wahrscheinlich von dem ursprünglichen Plan, bzw detailliert ihn in eine unterwartete Richtung: Man möchte ja die geänderten und präzisierten Kundenanforderungen berücksichtigen.

Ein unscheinbares, aber wichtiges Konzept dabei ist das Time Boxing: Ein Inkrement dauert immer gleich lang (z.B. vier Wochen). Dadurch kommt ein gewisser Takt ins Projekt, der auf Dauer auch die Aufwandsschätzung erleichtert. Man kann gut vergleichen, wie viel man in einem Inkrement schaffen kann: immer etwa gleich viel.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Kurze Release-Zyklen

Prinzip Einfachheit

- Ziel: Häufiges Feedback durch den Kunden
 - Operative Nutzung der Inkreme[n]te (nicht nur Prototypen!)
 - Aufteilung in Teillieferungen scheint schwierig – geht aber meist
- Begriffsklärung
 - „Release“ = „Inkrement“ = Auslieferung zur operativen Nutzung
 - Dazwischen Iterationen: auch lauffähig, gehen nicht an Kunden
- „Saubere Spezifikation für alles ist besser“ – stimmt das?
 - ist aber oft nicht zu erreichen
 - Kostet so viel Aufwand, dass keine Zeit für Umsetzung bleibt
 - Veraltet, bevor Projekt zu Ende ist
- Selbst bei Projekt-Abbruch ist schon Nützliches entstanden

... wenn man das Wichtigste zuerst macht!

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 349

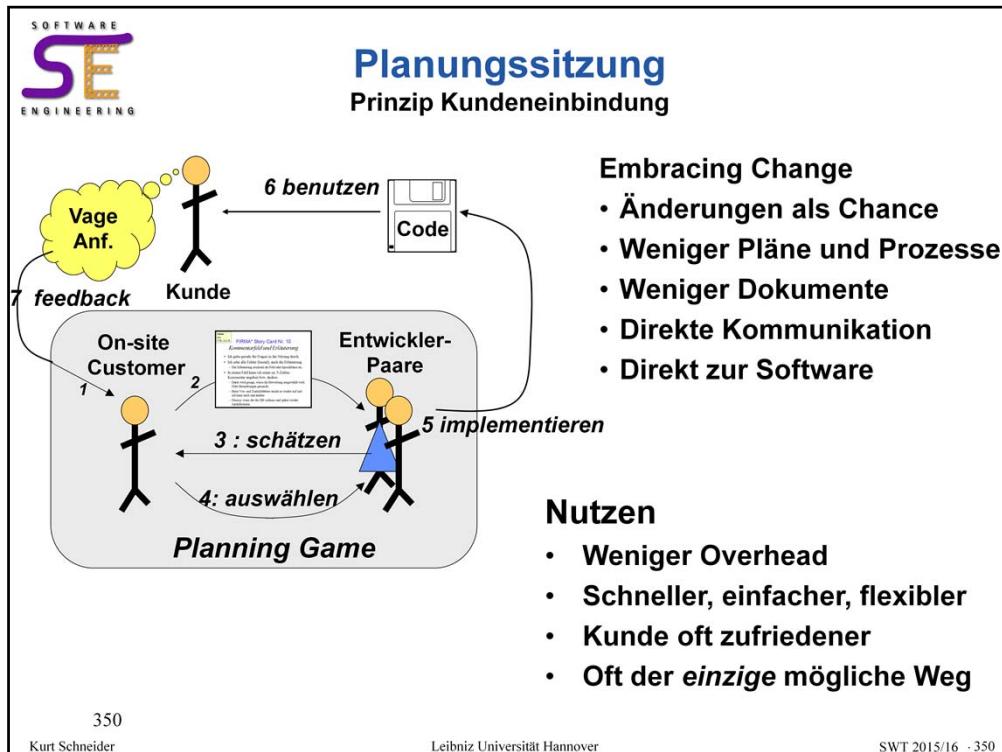
Der letzte Punkt ist interessant:

Wenn nach zwei Inkrementen das Geld ausgeht, hat der Kunde immerhin zwei Bausteine bekommen. Geht man davon aus, dass er sich an die agilen Prinzipien gehalten hat, dann sind das die zwei Bausteine, die ihm am meisten nützen: denn die muss er immer aussuchen.

In einem normalen Projekt hätte man erst lange die Anforderungen und den Entwurf für das Gesamtprojekt entwickelt – und nach derselben Zeit noch nicht viel implementiert. Das wäre erst am Ende gekommen, für den das Geld dann nicht mehr gereicht hat.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Die Folie zeigt auf einen Punkt gebracht die Idee der engen Zusammenarbeit zwischen Entwicklern und Kunden in einem agilen Projekt.

- Es wird ein echter Kunde ins Projektteam geholt und steht dort als „On-site customer“ ständig für Fragen zur Verfügung.
- Die Abstimmung zwischen den Iterationen und Inkrementen geschieht in der Planungssitzung. Ihre wesentlichen drei Schritte (2-4) sind oben gezeigt. Kunden und Entwickler arbeiten zusammen und legen fest, welche Aufgaben zuerst umzusetzen sind.
- Der Kunde beschreibt einige Aufgaben; die Entwickler schätzen, wie aufwändig sie wären. Dann darf der Kunde so viele aussuchen, wie ins nächste Inkrement passen. Er soll dabei aber unbedingt die nehmen, die ihm am meisten nützen würden.
- Nach dem Inkrement bekommen die Kunden diesen nächsten Baustein. Sie verwenden ihn, geben neues Feedback: Und der Kartenstapel wird neu gemischt, ergänzt und verändert.

Auf diese Weise bekommt man Flexibilität, enge Kundeneinbindung und hohen Kundennutzen unter einen Hut.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

Informationen zur Planungssitzung
Prinzip Kundeneinbindung

- Anforderungen werden auf Story Cards gesammelt
 - vom Kunden umgangssprachlich geschrieben
 - Informell, auf A5 od. A4-Karten, evtl. mit kleinen Skizzen
- Entwickler schätzen Aufwand für jede Story Card
 - abstraktes Maß (z.B. 1-5 Punkte)
 - Schätzung durch Erfahrung
 - Kunden, Entwickler interagieren dabei
 - Neue Erkenntnisse auf Story Cards notieren
- Kunde wählt Story Cards für nächste Release(s) aus
 - (Meistens, möglichst!) nach Nutzen priorisiert
 - Bisherige Produktivität des Entwicklungsteams ergibt Limit
- Story Cards sind Grundlage für Akzeptanz-Tests
- Es gibt keine gewohnte, „ordentliche Spezifikation“

Geld abheben
7.6.13, D.Autor

Anforderung

1. Authentifizieren
2. Betrag eingeben
3. Prüfen
4. Auszahlen

Aufwand,
Datum der Erledigung
geschätzt: 6 Pkt./ 1 Wo

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 351

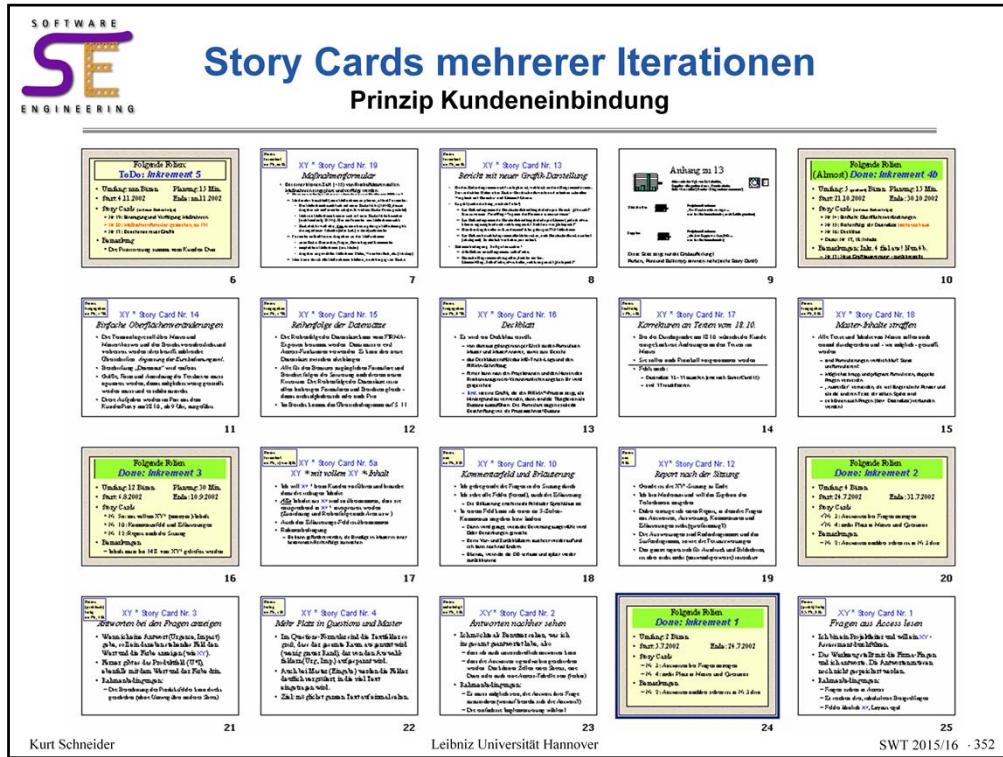
Die Karten heißen „Story Cards“. Es sind wirklich nur einseitige Zettel oder Karten, auf denen jeweils eine „Story“ steht, also ein Ablauf, den der Kunde nachher mit dem System machen will.

Das klingt ganz nach Use Case und ist wirklich sehr ähnlich – nur noch etwas einfacher geschrieben. Im wesentlichen steht hier das Hauptszenario. Wenn die Entwickler mehr wissen wollen (Vorbedingung, Trigger, Erweiterungen usw.), dann können sie den Kunden ja jederzeit fragen, er ist ja on-site.

Die Antworten schreibt man dann z.B. hinten auf die Karte, oder man programmiert sie direkt als Testfälle und Programm ein. Eine herkömmliche Spezifikation kommt auf diese Weise nicht zustande; die agilen Methoden müssen anders gewährleisten, dass nicht alles vergessen wird oder ins Chaos stürzt.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 352

So kann es nach einigen Inkrementen aussehen: Das „Deckblatt“ trennt jeweils die Story Cards eines Inkrementes von dem nächsten. Das sind alles ganz schlichte Folien, die man gerne auch von Hand auf Papier schreiben kann. Kein unnötiger Aufwand, von dem man nichts hat! Heißt die Devise.

In der Regel werden die Story Cards auch nicht mehr aktualisiert, sobald sie einmal umgesetzt sind. Sie haben ihren Zweck erfüllt und dürfen veralten.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

 **On-site Customer / Kunde vor Ort**
Prinzip Kundeneinbindung

- **Kunde ist über die gesamte Projekt-Laufzeit vor Ort**
– zumindest leicht und jederzeit erreichbar
- **Schreibt Story Cards (Entwickler dürfen helfen)**
– Ständige Weiterentwicklung der Anforderungen
- **Beantwortet Fragen der Entwickler, auch unter Ungewissheit**
– Kein Zeitverlust bei Entscheidungen
– Keine Fehlentscheidungen durch Entwickler
- **Voraussetzung: Kompetenz und Entscheidungsgewalt**
– Kunden müssen nach Nutzen sortieren (können und wollen)

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 353

Die enge Kundeneinbindung ist ein Erfolgsrezept, nicht nur für agile Projekte. Auch andere profitieren davon, den Kunden rasch fragen zu können. Folglich kann man das auch übernehmen, wenn man nicht vollständig agil entwickeln kann oder möchte.

Man muss nur den Kunden dazu bringen, so eng zusammenzuarbeiten. Das kommt nicht von alleine. Der Kunde muss merken, dass er davon profitiert.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

**SOFTWARE
SE
ENGINEERING**

Pair Programming

Prinzip „offen für Änderungen“

- Metapher: Fahrer und Beifahrer
- Jedes Paar arbeitet an einer Story Card
 - oder Task Card (Teile einer Story Card)
 - häufiges Abwechseln der Rollen
 - möglichst tägliches Tauschen der Paare
- Ständiges Review
 - Vermeidet vor allem Flüchtigkeitsfehler
 - Eine(r) schreibt, zweite(r) denkt und kontrolliert
 - Vier-Augen-Prinzip
- Ständiges Lernen voneinander
 - Hebt Entwickler kontinuierlich auf ein einheitliches Niveau
- Ist effektiv und effizient
 - Belegen einige Studien; andere belegen das Gegenteil
 - es gibt verschiedene Erfahrungen



Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 354

Pair Programming ist eine Praktik, die gerade im Studium interessant ist, um von Mitstudierenden zu lernen und sich mit ihnen auszutauschen. Aber auch in der Projektpraxis kann es sehr hilfreich sein.

Studien zeigen aber, dass nicht jedes Team unter allen Umständen bereit und in der Lage ist, regelmäßig in Paaren zu arbeiten. Dann muss man eben darauf verzichten. Freilich: mit jeder Praktik, auf die man verzichtet, geht auch ein Stück der Vorteile verloren, die agile Methoden haben.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Kontinuierliche Integration
Prinzip „offen für Änderungen“

- **Integration mehrfach am Tag, mit optimistischer „Sperre“**
 - Jeder hat immer aktuellen Code
 - Doppelte Arbeit wird vermieden
- **Ist Voraussetzung für gemeinsame Code-Verantwortung**
- **Jederzeit lauffähige, getestete Version**
 - Dadurch stets Feedback vom Kunden
 - Basis für Entscheidungen
 - Rückfallposition
 - Refactorings werden kleiner

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 355

Wenn verschiedene Teams an einem gemeinsamen Stück Code arbeiten und es so viele Möglichkeiten für Kollisionen und Missverständnisse mit dem Kunden gibt, hilft nur häufige Integration.

Daran erkennt man gewisse Schwierigkeiten, zum Beispiel mit Schnittstellen. Und nur mit dem integrierten System kann man den Kunden konfrontieren, auf dessen Feedback man ja angewiesen ist.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

The slide features a logo for 'SOFTWARE ENGINEERING' with 'SE' in large letters. The title 'Prinzip von TestFirst: ein Dialog' is at the top, followed by 'Prinzip „offen für Änderungen“'. Below the title is a text box containing Java code and test cases. The code defines a method 'len(int)' that returns 1 for all inputs. The first test case checks for 'len(5)' and fails with a compiler error because the method does not yet exist. Subsequent tests check for 'len(321)' and 'len(12345678)', both failing with JUnit errors indicating the expected value is 3. The final test case for 'len(12345678)' succeeds with JUnit's 'ok.' message. The slide also includes a snippet of the program code showing the implementation of the 'len' method.

Aufgabe: Java-Methode `len(int)` gibt zurück, wie viele Stellen eine int-Zahl hat.

Test beginnt JUnit
„len(5) soll 1 sein!“
`assertEquals(len(5), 1);` COMPILER-FEHLER!
Was soll „len“ bedeuten?

Programm: Das ist einfach:
`public int len (int zahl) { return 1; }`

JUnit: ok. Testfall erfüllt.

Test: Na, warte!
„len(321) soll 3 sein!“
`assertEquals(len(321), 3);` JUnit: Fehler! 1 statt 3

Programm: auch nicht so schwer ...
`if zahl<10 then return 1 else return 3`

Test: Ich glaub es nicht!
„len(12345678) soll 8 sein!“
`assertEquals(len(12345678), 8);` JUnit: ok.

Programm: ... ok, ich sehe das Muster:
eine Schleife: `for (i=...`

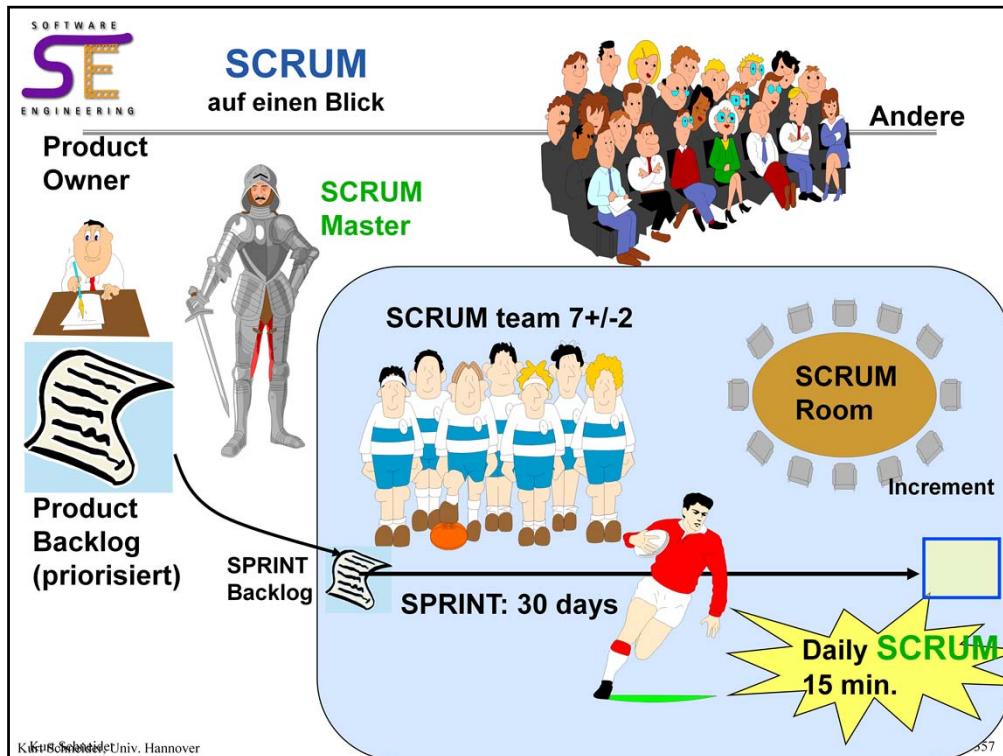
JUnit: Fehler! 3 statt 8

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 356

Diese Folie ist eine Wiederholung, die kennen Sie schon: Das soll daran erinnern, dass auch das ständige, schon sehr früh beginnende Testen dazu beiträgt, stets gut qualitätsgesicherten Code zu haben. Diese Praktik stützt sich gegenseitig mit Pair Programming (die Entwickler lernen voneinander) und anderen Praktiken.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Hier sieht man auf einem Bild alle wesentlichen Elemente von SCRUM.

Der Product Owner (ein Kunde) hütet den Product Backlog. Er darf ihn jederzeit verändern, insbesondere die Reihenfolge von Einträgen verändern.

Wenn ein Sprint beginnt, wird soviel oben vom Product Backlog genommen, dass es für den Sprint langt. Dann schließt sich die Kapsel, in den nächsten 30 Tagen darf niemand dem SCRUM-Team hineinreden oder den Sprint-Backlog ändern. Inzwischen gibt es in vielen Unternehmen kürzere Sprints. Natürlich muss man so lange arbeiten, dass etwas Sinnvolles entstehen kann.

Intern werden jeden Tag SCRUM-Meetings statt finden, am Ende des Sprints wird Software erzeugt. Hier symbolisiert durch einen „Inkrement-Kasten“.

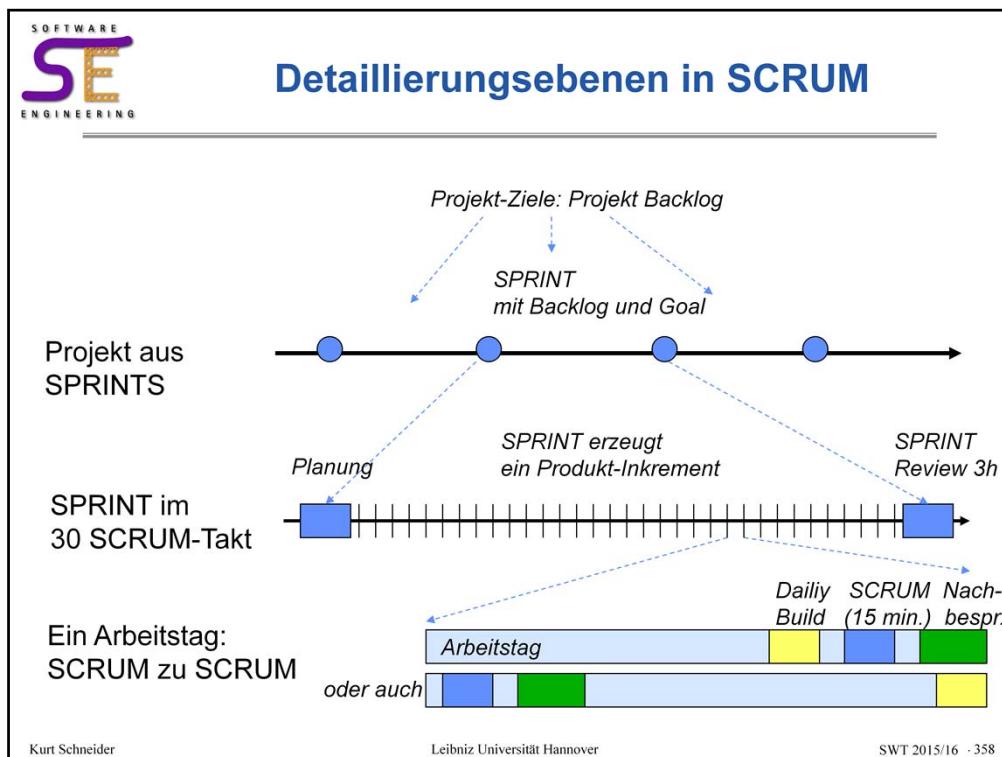
Der SCRUM-Master erinnert an den Projektleiter, ist es aber nicht. Denn ein SCRUM-Team ist selbstorganisierend. Die Hauptaufgabe des SCRUM-Masters ist es, Schwierigkeiten auszuräumen, die im Daily SCRUM geäußert werden.

Andere dürfen zusehen, im Dails SCRUM aber nichts sagen.

Nicht vergessen: Wie im Sprint entwickelt wird, sagt SCRUM nicht. Es wäre zwar gut, agile Praktiken zu nutzen, aber auch ein kleines V-Modell wäre denkbar. SCRUM ist eine Management-Methode für die Entwicklung.

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



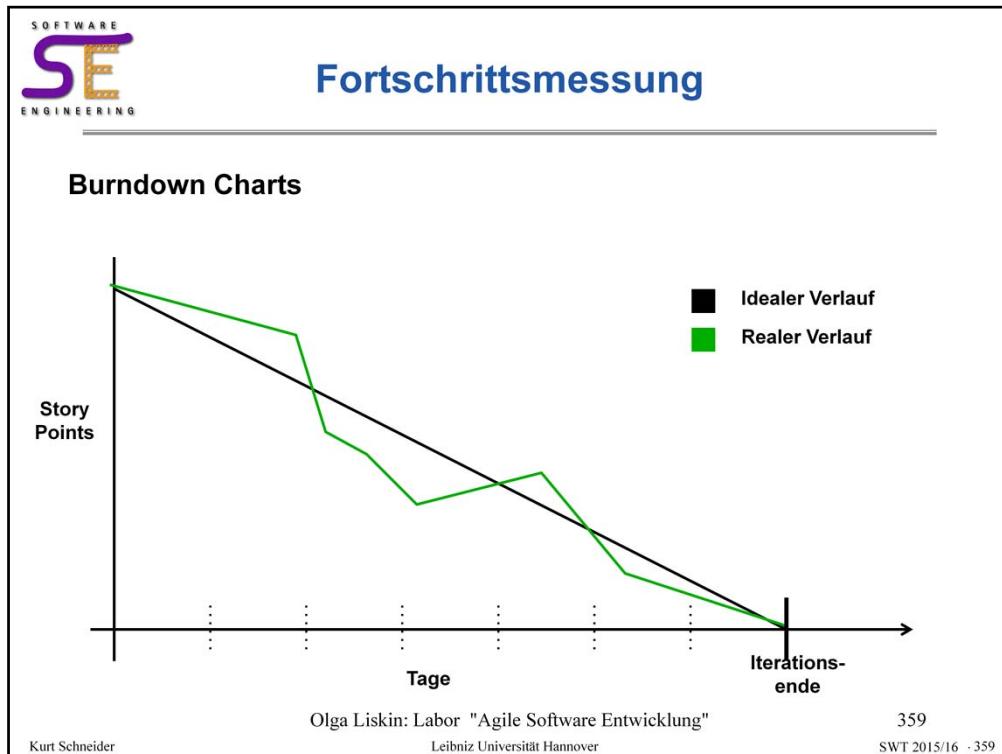
Dies ist die „Seitenansicht“, in der man die Zeiteinteilung sieht.

Die untere Ebene ist jeweils ein Zoom in die obere.

Der Arbeitstag kann unterschiedlich gegliedert sein, solange nur jeden Tag ein Daily SCRUM und ein Daily Build stattfinden. Die sollen dann immer zur gleichen Tageszeit durchgeführt werden.

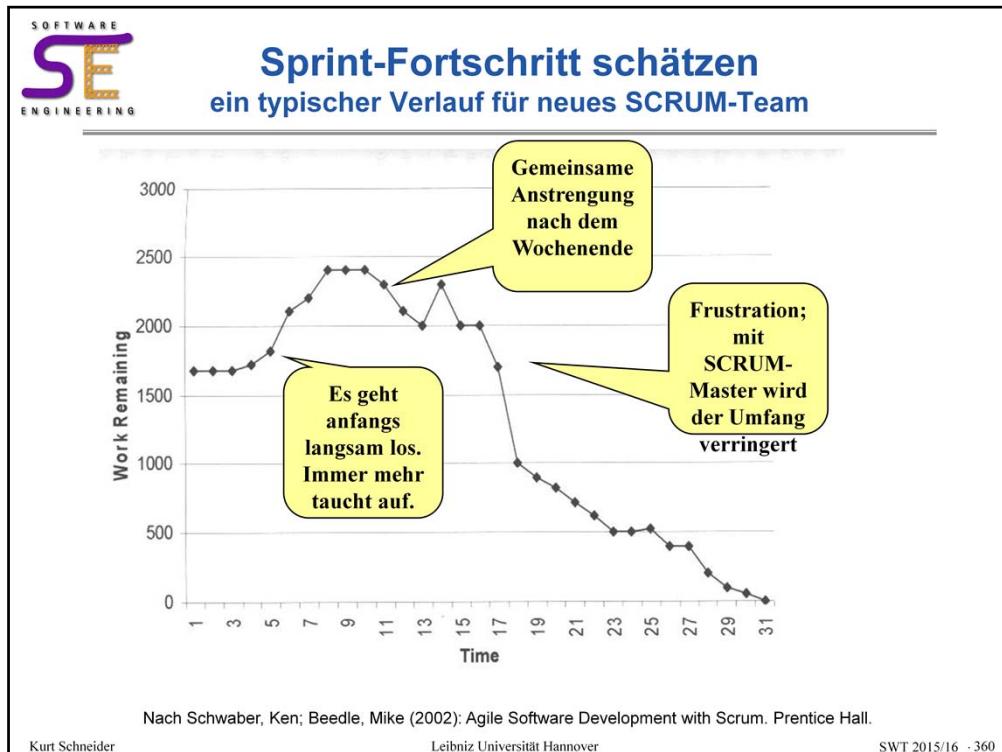
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Nach Schwaber, Ken; Beedle, Mike (2002): Agile Software Development with Scrum. Prentice Hall.

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 360

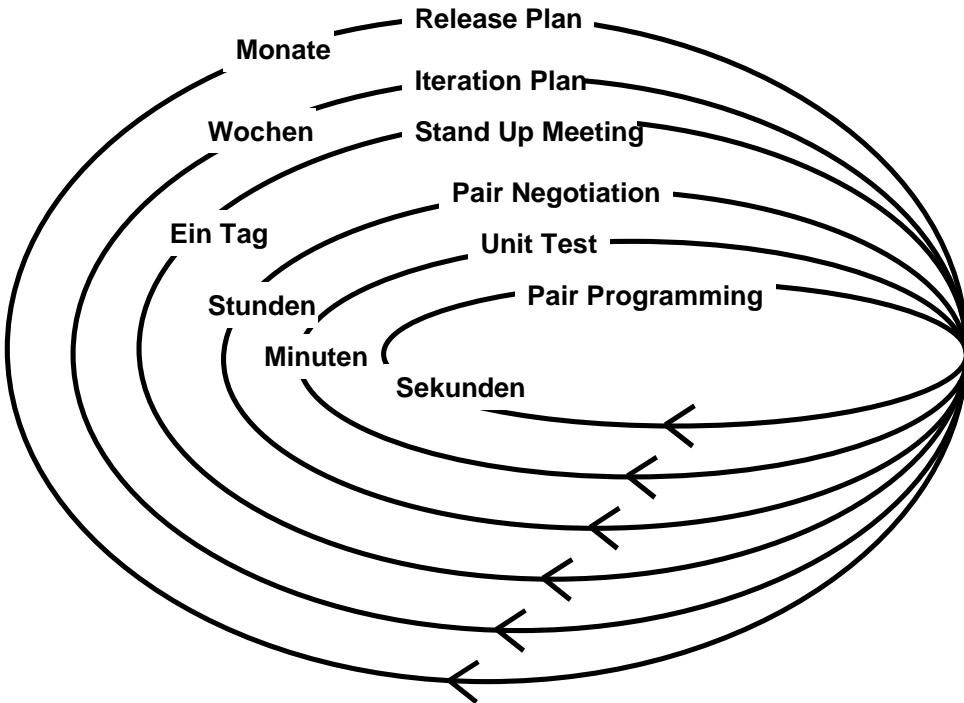
Hier zum Beispiel wieder die Bitte an den SCRUM-Master, mit dem Kunden auszuhandeln, dass die Aufgabe verkleinert wird.

Dadurch sinkt dann die Restarbeit auf einen Schlag.

Am Anfang steigt der geschätzte Restaufwand, weil man durch die beginnenden Arbeiten erst mitbekommt, was noch alles fehlt und was davon wieder abhängt. Man entdeckt also immer mehr Konsequenzen und immer mehr Arbeit.



Feedback beim eXtreme Programming (XP)



SWT II (2004)- 219

Basiert auf Folien von Stefan Roock, it-wps, Hamburg

Kurt Schneider, Univ. Hannover

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Ausblick: Software-Projekt (SWP)

- Idee: Software entwickeln im Team
 - Mit Anforderungserhebung und Kunden
 - Mit Entwurf, Implementierung und Test
 - Mit Besprechungen, Reibereien und Lösungen
 - Mit wirklich einsetzbarer Ziel-Software
 - Mit echtem Feedback
- Früher: Strenge V-Modell; heute: Mischform
 - Spez
 - + Stories
- Heute: Extra angepasstes Vorgehensmodell, wie in Praxis
 - Klassische Spezifikation
 - Auflösung in Story Cards für Priorisierung und Änderung
 - SCRUM-Meetings pro Woche

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 362

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

Interviews verwalten	
Use Case Nr.	02
...	...
Auslöser	Interviewübersicht wird geöffnet.
Beschreibung	<ol style="list-style-type: none">1. System zeigt Interviewübersicht.2. Forscher erstellt ein neues Interview.3. System öffnet Interview-Bearbeiten-Dialog.4. Forscher lädt eine Audio-Datei hoch.5. Forscher klickt auf Speichern.6. System speichert die Änderungen und schließt den Dialog.7. Forscher klickt auf den Bearbeiten-Button von einem Interview.8. System öffnet Interview-Bearbeiten-Dialog.9. Forscher ändert den Wert einer Interview-Eigenschaft.10. Forscher klickt auf Speichern.11. System speichert die Änderungen und schließt den Dialog.12. Forscher löscht ein Interview.
Erweiterung	<p>5a. WENN der Forscher auf Abbrechen klickt, DANN werden die Änderungen nicht gespeichert. 10a. WENN der Forscher auf Abbrechen klickt, DANN werden die Änderungen nicht gespeichert.</p>

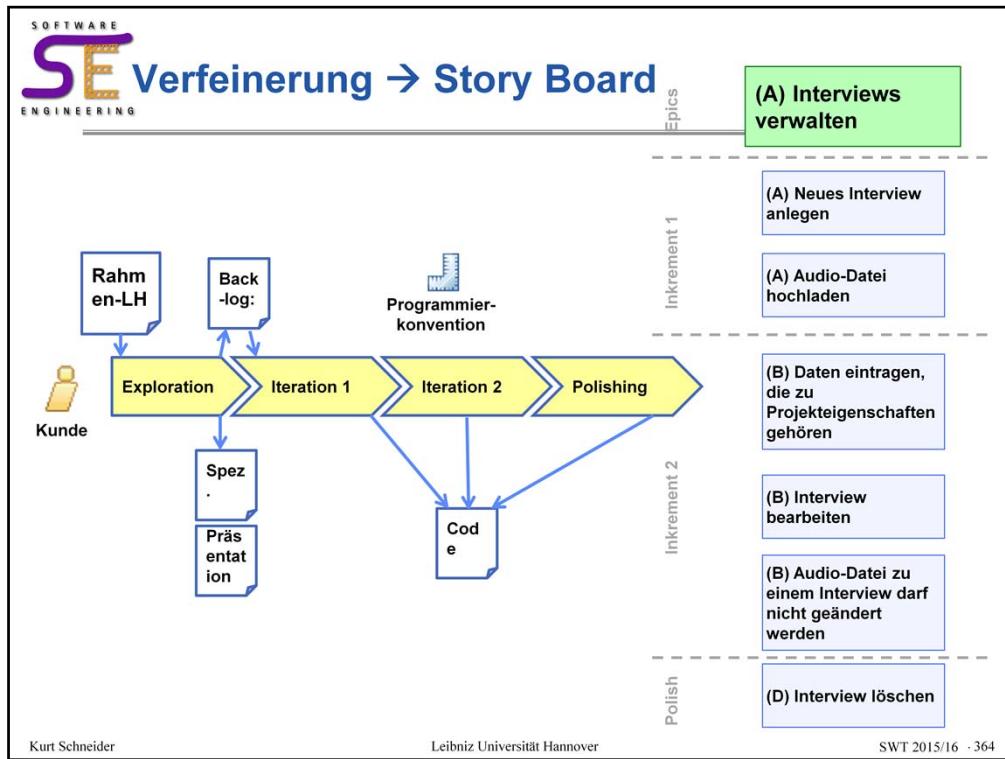
Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 363

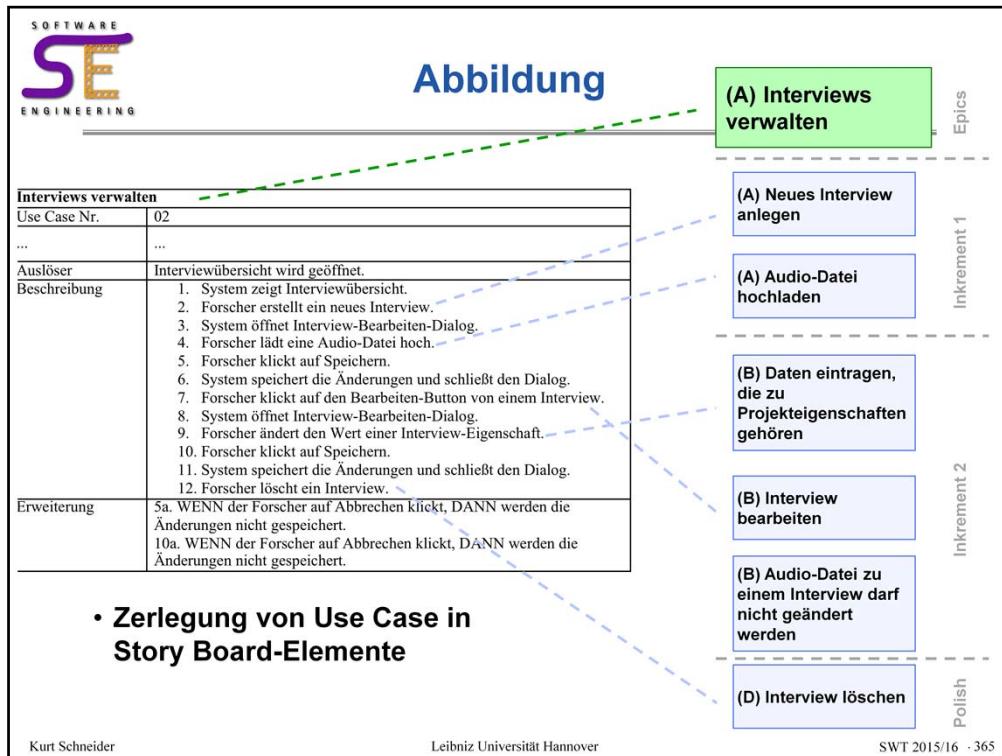
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



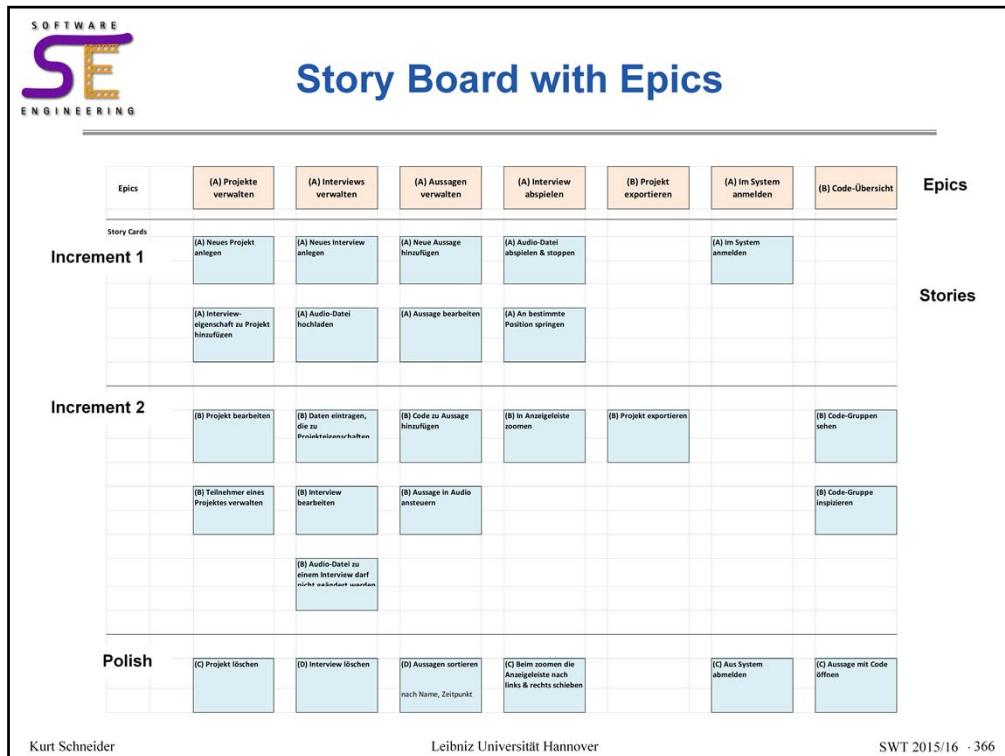
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Planung der Inkremente

**Olga Boruszewski
FG Software Engineering**

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 367

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

Use Case zeigt das gewünschte End-Verhalten	
UC01	Ressourcen verwalten (Beamer, Rechner, Räume)
...	...
Hauptszenario	<ol style="list-style-type: none">1. Mitarbeiter wählt „Ressourcen anzeigen“2. System zeigt alle Ressourcen in einer Liste an3. Mitarbeiter wählt „Neue Ressource anlegen“4. System zeigt leeres Formular an5. Mitarbeiter gibt ID und Name der Ressource an6. Mitarbeiter wählt die passende Kategorie für die Ressource aus (Beamer, Rechner, Raum, ...)7. Mitarbeiter bestätigt die Eingabe8. System trägt die neue Ressource ein und zeigt die Liste mit allen Ressourcen an
Erweiterungen	<p>6b. WENN es die passende Kategorie noch nicht gibt, DANN wählt Mitarbeiter „neue Kategorie erstellen“, gibt Kategorienamen an und bestätigt. Zurück zu Schritt 6.</p> <p>8b. WENN es bereits eine Ressource mit derselben ID gibt, warnt das System den Nutzer. Weiter bei Schritt 5</p>

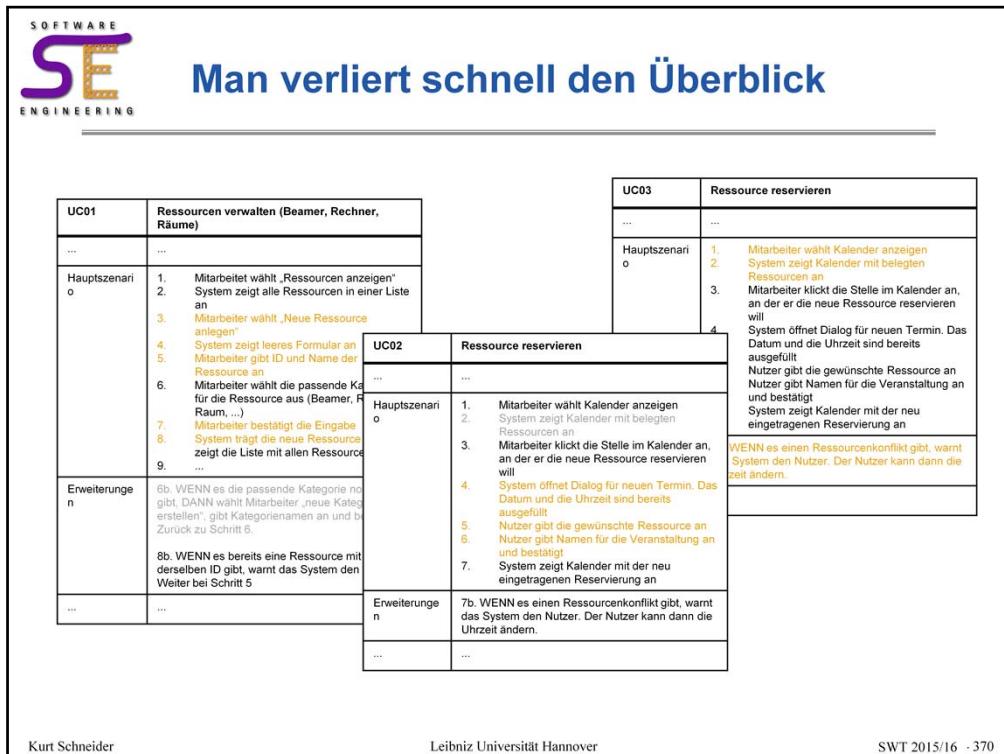
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

... aber nicht jeder Schritt ist gleich wichtig	
SOFTWARE SE ENGINEERING	UC01 Ressourcen verwalten (Beamer, Rechner, Räume)
...	...
Hauptszenario	<ol style="list-style-type: none">1. Mitarbeiter wählt „Ressourcen anzeigen“2. System zeigt alle Ressourcen in einer Liste an3. Mitarbeiter wählt „Neue Ressource anlegen“4. System zeigt leeres Formular an5. Mitarbeiter gibt ID und Name der Ressource an6. Mitarbeiter wählt die passende Kategorie für die Ressource aus (Beamer, Rechner, Raum, ...)7. Mitarbeiter bestätigt die Eingabe8. System trägt die neue Ressource ein und zeigt die Liste mit allen Ressourcen an
Besonders wichtige Schritte	<ol style="list-style-type: none">6b. WENN es die passende Kategorie noch nicht gibt, DANN wählt Mitarbeiter „neue Kategorie erstellen“, gibt Kategorienamen an und bestätigt. Zurück zu Schritt 6.8b. WENN es bereits eine Ressource mit derselben ID gibt, warnt das System den Nutzer. Weiter bei Schritt 5
Besonders unwichtige Schritte	
Kurt Schneider	

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



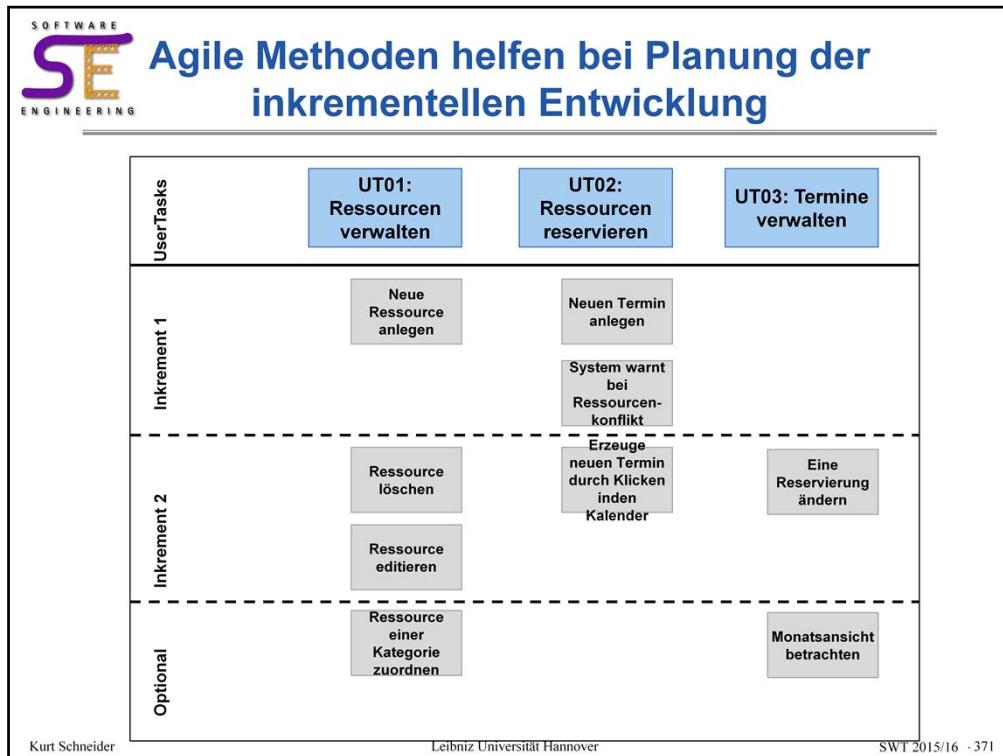
Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 370

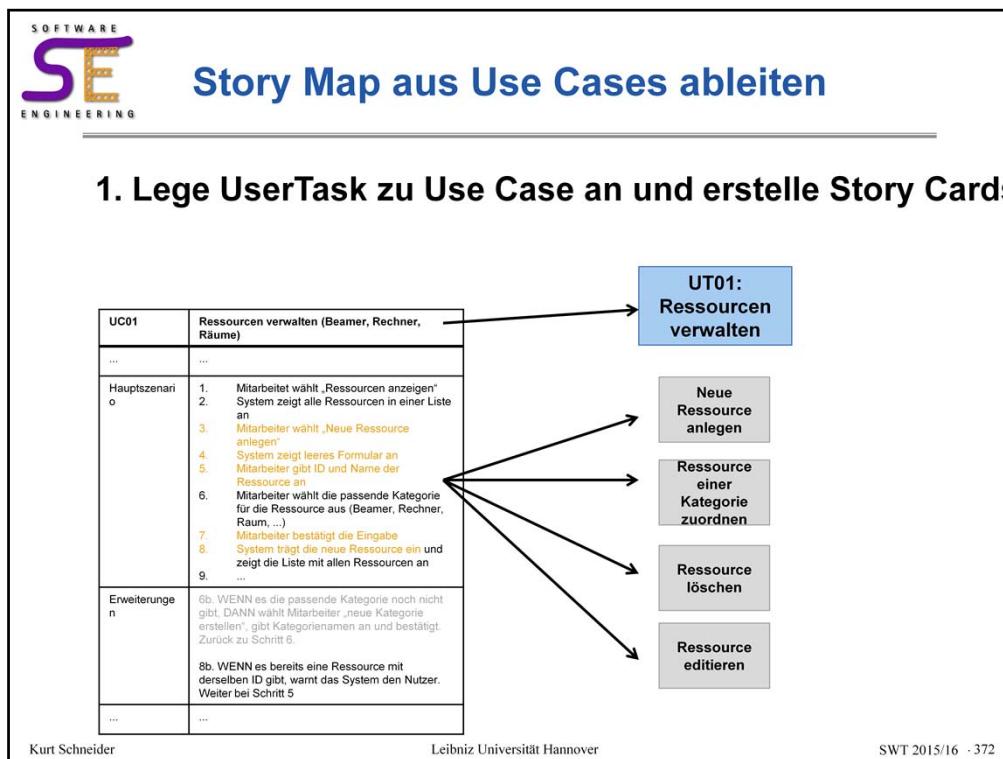
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



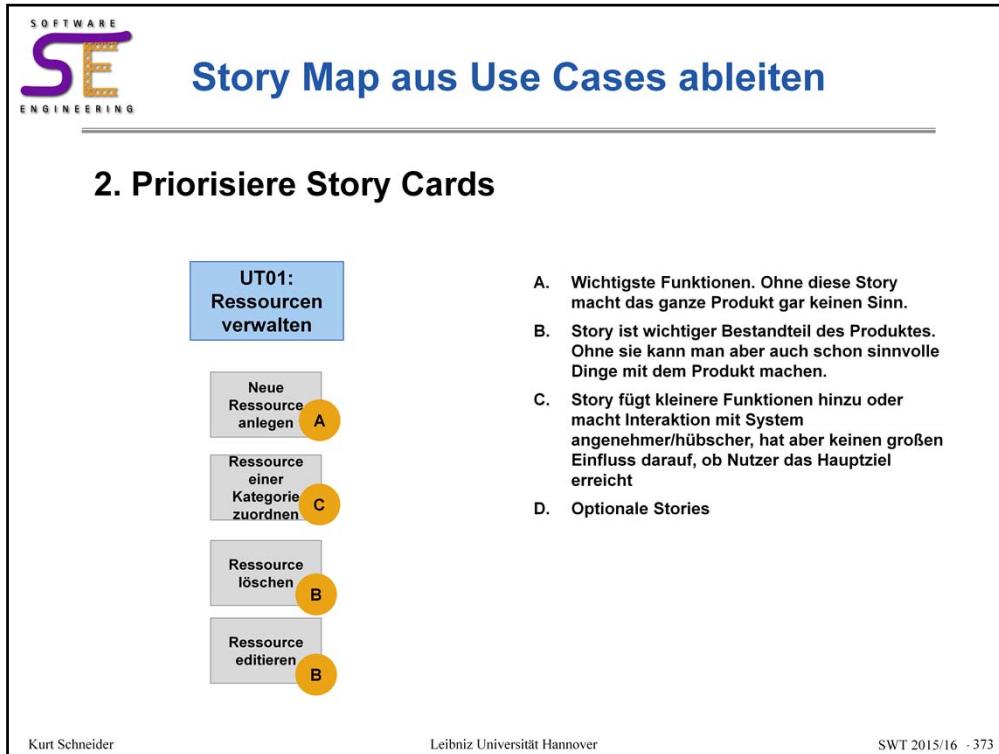
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



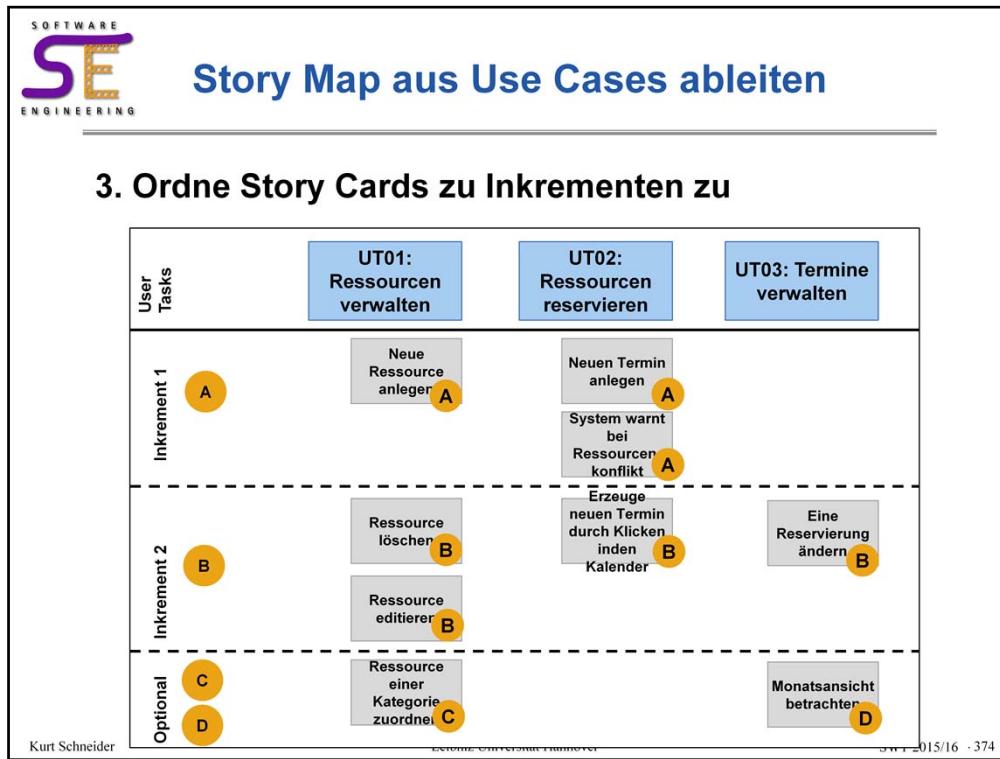
The slide is titled "Story Map aus Use Cases ableiten" and features a logo for "SOFTWARE ENGINEERING" with a stylized "SE". Below the title, it says "2. Priorisiere Story Cards". On the left, there's a box labeled "UT01: Ressourcen verwalten" containing four items: "Neue Ressource anlegen" (labeled A), "Ressource einer Kategorie zuordnen" (labeled C), "Ressource löschen" (labeled B), and "Ressource editieren" (labeled B). To the right of these items is a list of priorities:

- A. Wichtigste Funktionen. Ohne diese Story macht das ganze Produkt gar keinen Sinn.
- B. Story ist wichtiger Bestandteil des Produktes. Ohne sie kann man aber auch schon sinnvolle Dinge mit dem Produkt machen.
- C. Story fügt kleinere Funktionen hinzu oder macht Interaktion mit System angenehmer/hübscher, hat aber keinen großen Einfluss darauf, ob Nutzer das Hauptziel erreicht
- D. Optionale Stories

At the bottom, the slide credits "Kurt Schneider", "Leibniz Universität Hannover", and "SWT 2015/16 · 373".

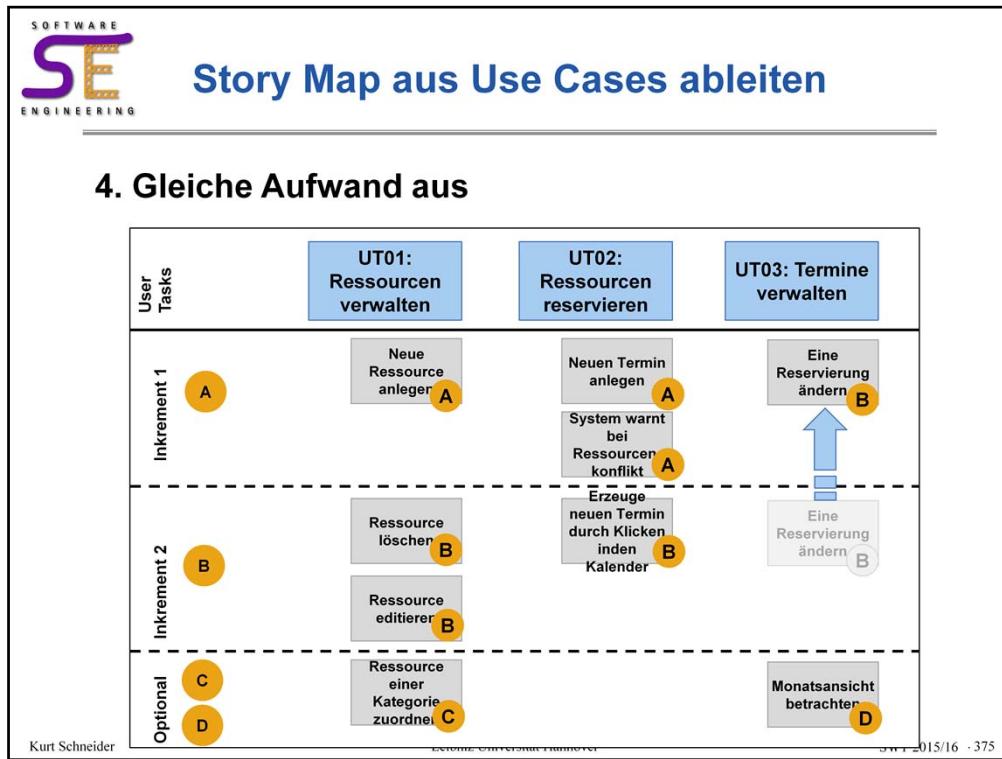
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



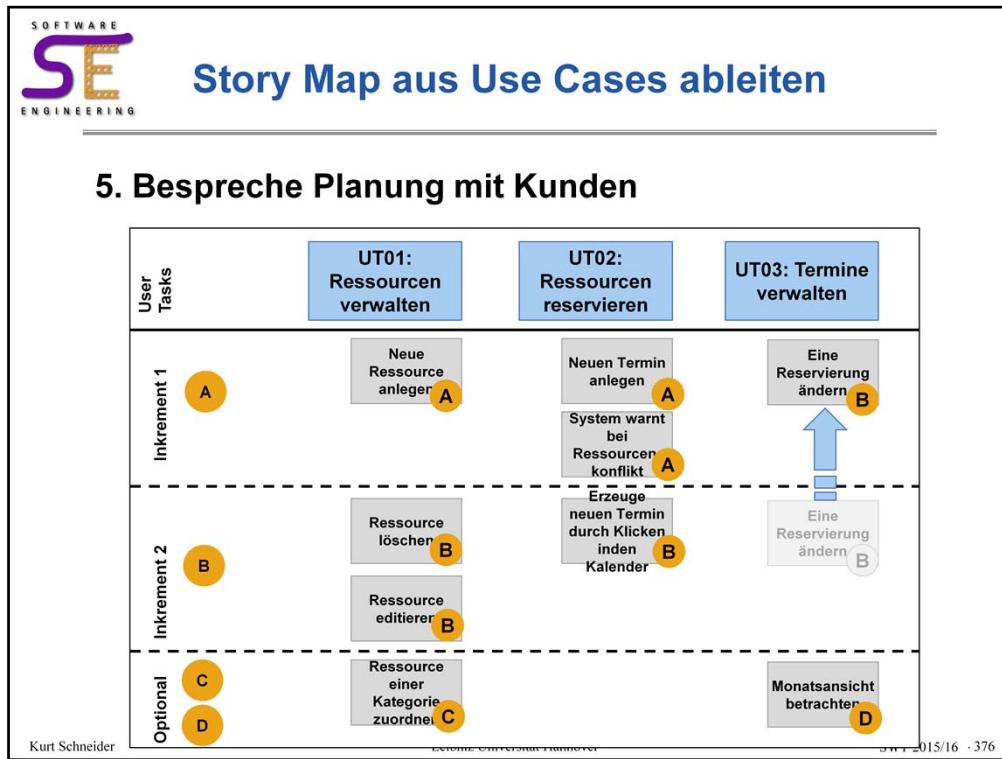
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



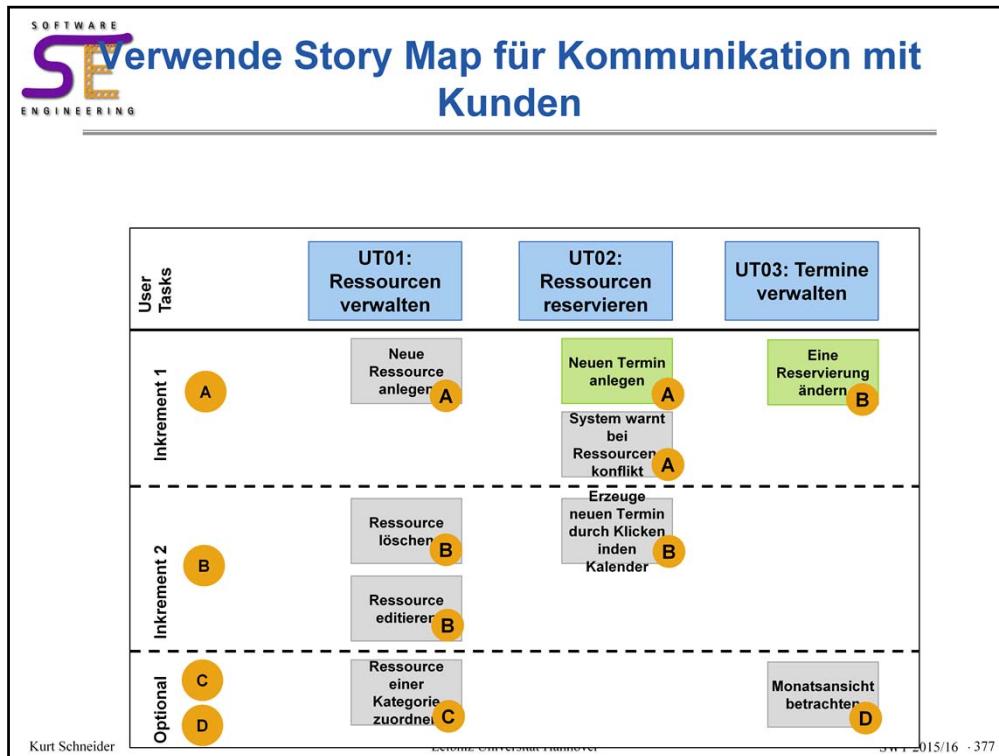
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

**Oft im Einsatz bei agilen Teams:
Stift und Papier**

Leicht Anforderungen:

- neu hinzufügen
- verschieben
- austauschen
- Bilder dazu malen
- markieren
- ...



Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 378

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover

Bei uns: Flipchart und Postits

RPT

UT1 Ressourcen verwalten (Scanner, Laptops, Räume)

UT2 Termine Verwalten

UT3 Alle beliebigen Ressourcen im Kalender anzeigen

UT4 Kategorien Verwalten

UT5 Benutzer Verwalten

UT6 Wiederholbare Termine verwalten

option 1

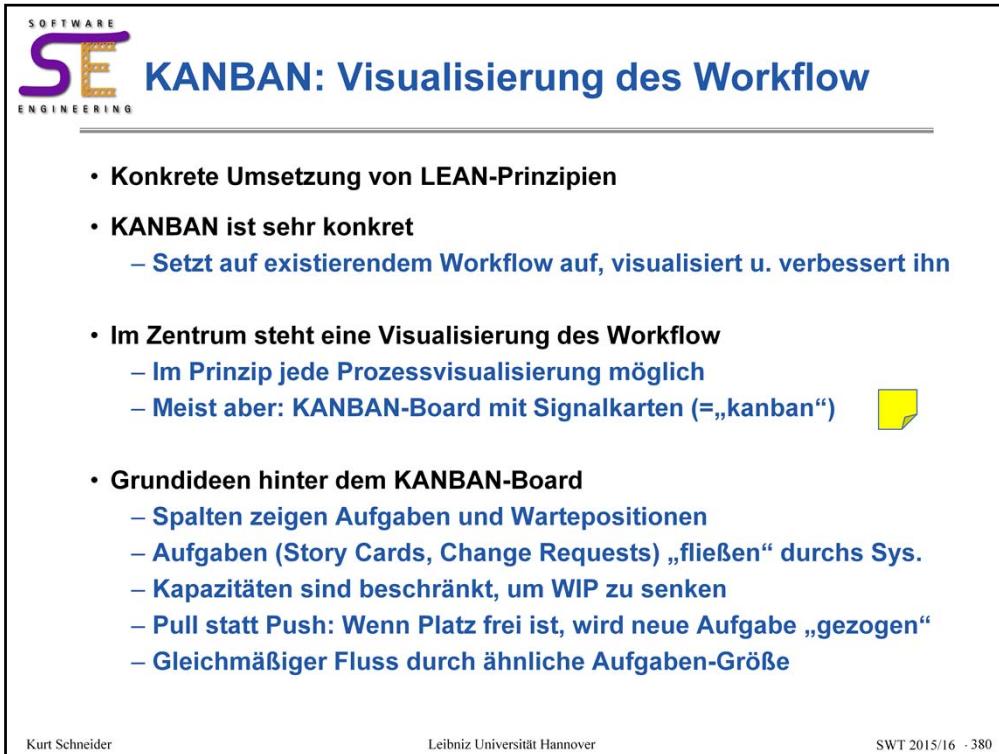
Inkr 1

Inkr 2

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 379

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



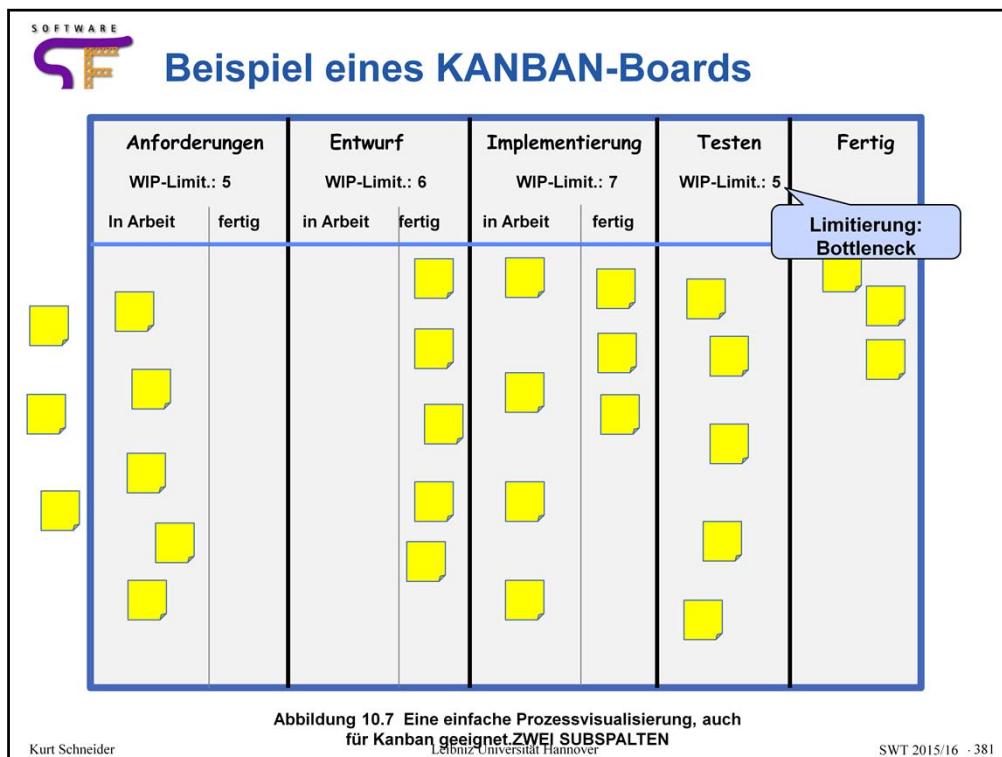
KANBAN: Visualisierung des Workflow

- Konkrete Umsetzung von LEAN-Prinzipien
- KANBAN ist sehr konkret
 - Setzt auf existierendem Workflow auf, visualisiert u. verbessert ihn
- Im Zentrum steht eine Visualisierung des Workflow
 - Im Prinzip jede Prozessvisualisierung möglich
 - Meist aber: KANBAN-Board mit Signalkarten (=„kanban“)
- Grundideen hinter dem KANBAN-Board
 - Spalten zeigen Aufgaben und Wartepositionen
 - Aufgaben (Story Cards, Change Requests) „fließen“ durchs Sys.
 - Kapazitäten sind beschränkt, um WIP zu senken
 - Pull statt Push: Wenn Platz frei ist, wird neue Aufgabe „gezogen“
 - Gleichmäßiger Fluss durch ähnliche Aufgaben-Größe

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 380

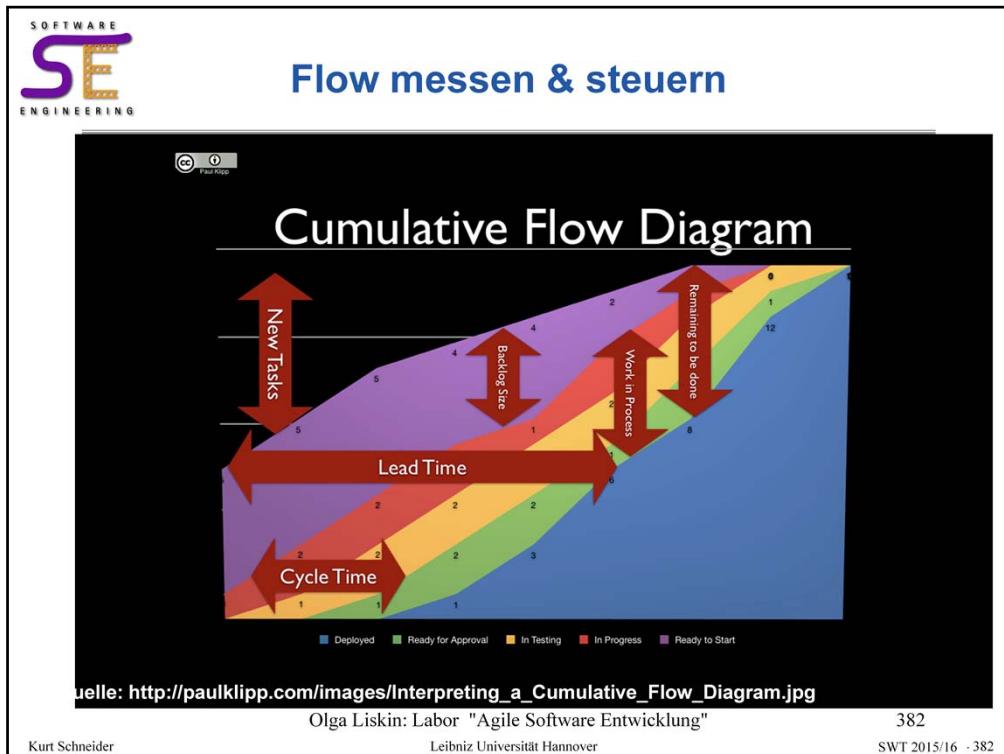
Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Fazit: Agil

- Agile Methoden sind heute allgemein anerkannt
 - Nicht nur „weniger Dokumentieren“
 - Auch: Angemessener Planungshorizont
 - Flexibilität durch abgestimmte Maßnahmen
- Auch nicht-agile Projekte können profitieren
 - Time-Boxing, enger Kundenkontakt
 - inkrementelle Entwicklung
 - Anforderungspakete nach Nutzen sortieren
- SCRUM ist derzeit sehr populär
- Viele Unternehmen mischen und kombinieren

Vorlesung „Grundlagen der Softwaretechnik (SWT)“ im WS 2015/2016

Prof. Dr. Kurt Schneider, FG Software Engineering, Leibniz Universität Hannover



Wir freuen uns auf Sie!

- Vorlesung Software-Qualität
- Software-Projekt SWP
- Bachelor- und Masterarbeiten
(an Uni, mit Unternehmen, ...)
- ... und viele Labore,
Projekte, Vorlesungen
zu aktuellen Themen.



- Für qualifizierte und engagierte Studierende
 - Studentische Hilfskraft- oder Tutorientätigkeit
 - Vermittlung von Auslandsaufenthalten
 - Gutachten für Stipendien

Sprechen Sie uns an!

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 384

Sie erreichen mich unter:

Kurt Schneider

ks@inf.uni-hannover.de

0511/762-19666