

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

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

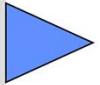
Softwaretechnik

Kapitel 3



1. Wieso Software Engineering?
2. Vom Einzelkämpfer zum Großprojekt

Systematische Softwareentwicklung



3. Anforderungen und Test: *Basis des Projekts*
4. Entwurf: *Strukturen und nicht-funktionale Eigenschaften*
5. Entwürfe notieren mit UML: *Modelle im SE*
6. Design Patterns: *Entwurfserfahrungen nutzen*
7. Management: *Technik und Projektmanagement*

Leibniz Universität Hannover

SWT 2015/16 · 76

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

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

Softwaretechnik

Inhalt von Kapitel 3



Systematische Softwareentwicklung

3. Anforderungen und Test: Basis des Projekts

- Aufgaben und Herausforderungen
- Objekt-Orientierte Analyse (OOA)
- Abläufe erfassen mit Use Cases
- Testen – schon jetzt

Leibniz Universität Hannover

SWT 2015/16 · 77

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

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



Die naive Sicht ist leider weit verbreitet.

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

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

The slide features a cartoon illustration of two men. On the left, a man in a blue suit and bow tie sits in a green office chair, looking up thoughtfully. On the right, another man in a green shirt and tie sits at a desk, looking down at a piece of paper. The background is white with a thin black border around the content area.

„Kunde beschreibt was er möchte“

Software Engineering

Kunde

- Wenn jemand Software will, kommt er zu uns
- er muss eben genau sagen, was er will
- aber wir kennen uns ja auch aus, vielleicht sogar besser als so mancher Kunde

Entwickler

- Vielleicht braucht er vor allem bessere Geschäftsprozesse?
- Man muss schon zum Kunden gehen, und so genau kann der das gar nicht sagen
- Anfängerfehler! In seinem Fach ist der Kunde König und kennt sich aus

und überhaupt ...

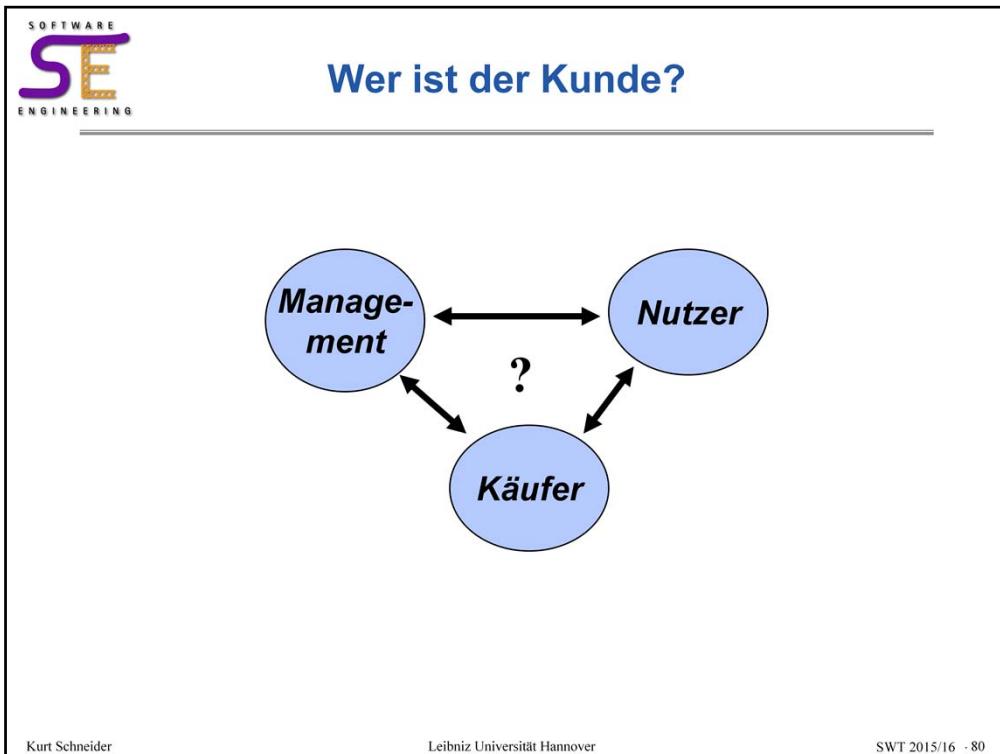
Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 79

In diesem Dialog zeigt sich, dass die naive Sicht die Rolle des Kunden falsch sieht und damit zum Scheitern eines Projekts beitragen könnte.

Da hilft dann die beste Programmierermannschaft nichts, wenn das Falsche programmiert wird.

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

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



Management bestimmt, Käufer ist der Einkauf, der um Rabatte feilscht, und der Nutzer muss mit dem Programm umgehen.

Alle drei sind zu berücksichtigen!

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

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

„Symmetry of Ignorance“ beachten
nach Gerhard Fischer

Unverständnis zwischen Informatikern und Kunden führt sehr oft zu vielen Missverständnissen und zu Frustration

?

?

Unverständnis auf beiden Seiten („symmetrisch“)

- Informatiker wissen nicht
 - wie das Geschäft läuft
 - was Kunden wollen, brauchen
 - was sie meinen
 - was das Problem ist
 - *dass sie es nicht wissen*
- Kunden/Fach-Leute wissen nicht
 - was Software leisten kann
 - was Informatiker können
 - wovon sie reden (UML)
 - welche Lösungen möglich wären
 - *dass die Informatiker sie auch nicht verstehen*

Hüten Sie sich davor zu glauben, Sie wüssten sowieso, was der Kunde will

Hinterfragen Sie scheinbare „Selbstverständlichkeiten“

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 81

Auch der nächste Satz hat es in sich.

Denn nicht jeder Kunde kann so leicht „beschreiben, was er will“.

Ganz allgemein hat man es hier ja mit einem Kommunikationsakt zu tun, von dem hier das Sprechen untersucht wird.

Gerhard Fischer von der CU Boulder hat mit dem Schlagwort der „Symmetry of ignorance“ die Beobachtung beschrieben, dass Entwickler oft die Welt des Kunden nicht gut verstehen und also nicht wissen, was es dort zu lösen gibt.

Auf der anderen Seite kennt und versteht der Kunde nicht die Lösungsmöglichkeiten, die dem Entwickler zur Verfügung stehen. Schon von daher können Kunden oft den späteren Soll-Zustand nicht beschreiben, weil sie ihn sich einfach nicht vorstellen können: Am ehesten kann man sich kleine Veränderungen am bestehenden (abzulösenden) System vorstellen. Aber sind das wirklich die Anforderungen an eine „ideale“ neue Lösung?

Menschen können bei einer Kommunikation nicht alles ständig wiederholen, was sie für selbstverständlich halten. Zwischen den beiden Gruppen gibt es aber kaum gemeinsames Verständnis darüber, so dass wichtige Dinge nicht gesagt werden, weil der eine sie für selbstverständlich hält. Die andere Gruppe weiß aber nichts davon. Wenn Sie hören, etwas sei ja „selbstverständlich“, sollten Sie immer besonders genau zuhören und unbedingt nachfragen, bis Sie ganz sicher sind, dass Sie alles verstanden haben. Sie wissen nicht, wann sich Ihnen diese Gelegenheit zum nächsten Mal bietet. Wörter wie „selbstverständlich“ sollten – besonders bei Anforderungen – immer ein Warnsignal sein.

Wenn nämlich ein Missverständnis aufgetreten ist, merkt man das erst gar nicht.

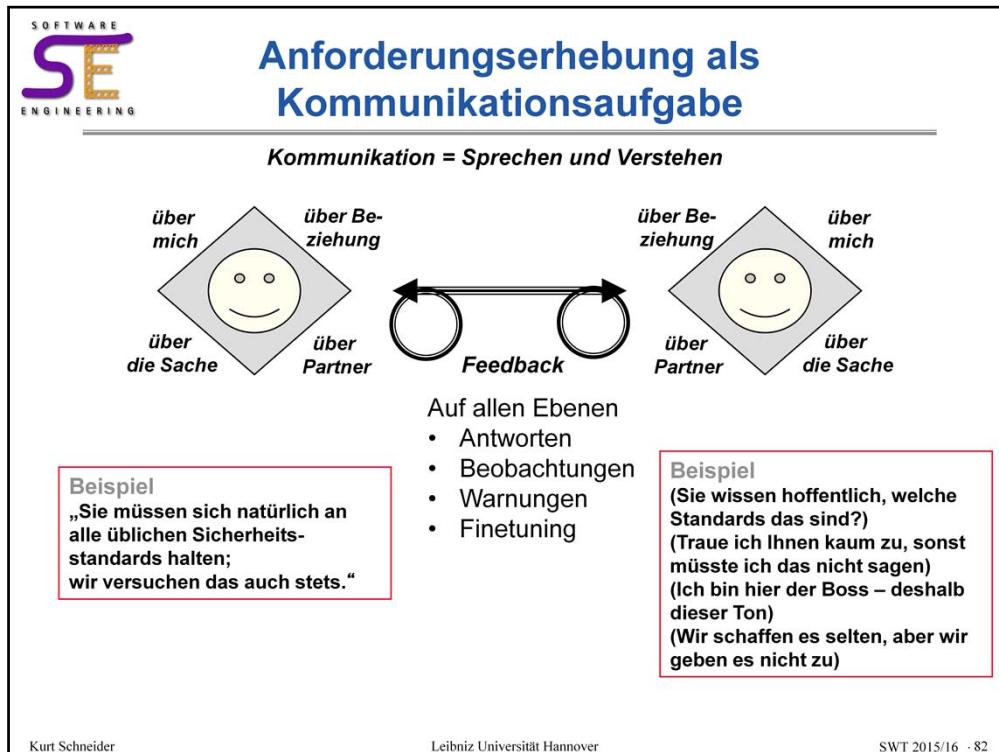
Generell ist die symmetry of ignorance ein allgegenwärtiges Anforderungs-Problem, zu dem sich jeder Projektleiter etwas einfallen lassen muss.

Skript mit erläuterten Folien für die Hörer dieser Vorlesung

Copyright: Kurt Schneider (bzw. zitierte Urheber)

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 · 82

Jawohl, Informatiker müssen sich mit Kommunikationsebenen zwischen Menschen beschäftigen!

Denn sie tragen und verdecken teilweise die Aussagen über Anforderungen.

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

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

**SOFTWARE
ENGINEERING**

Naives Loswerkeln rächt sich

- Stellen Sie früh fest, wer Anforderungen hat
- Versetzen Sie sich in seine/ihre Lage, so gut es geht
- Beschäftigen Sie sich mit dem Arbeitsgebiet des Kunden
- Nehmen Sie nichts für „selbstverständlich“, fragen Sie nach
- Glauben Sie nicht, Sie wüssten schon alles
- Kommunizieren Sie bewusst auf allen Ebenen
- Prüfen Sie, was Sie gefunden haben und wiederholen Sie



Betreiben Sie systematisch Requirements Engineering
(folgt →)

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 83

Wenn man es nicht tut und statt dessen einfach Losprogrammiert („das können wir am besten“), wird man bald in einer Ecke sitzen, aus der man nicht mehr herauskommt: Es gibt viele Vorarbeiten, die aber leider nicht zu den Kundenwünschen passen.

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' (SE) with the letters 'SE' in large, stylized, glowing letters. Below it, the words 'SOFTWARE' and 'ENGINEERING' are written vertically. To the right of the logo, the title '„Requirements Engineering“' is displayed in a large, bold, blue font. Below the title, there is a horizontal line. Underneath the line, the text 'Erinnerung – wieder einmal Engineering:' is written in bold. A quote follows: '„a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of Requirements“'. A section titled 'Konsequenzen auf den Umgang mit Anforderungen' is present, listing five points: '- Kostendenken', '- Qualitätsbewußtsein' (with a bullet point '• nicht nur subjektiv definiert'), '- Anwendung von Normen und Regeln' (with a bullet point '• keine Künstler'), '- Baugruppen und Wiederverwendung' (with a bullet point '• statt „not invented here“'), and '- Probleme durch Zerlegung lösen' (with a bullet point '• „divide et impera“'). At the bottom of the slide, there are three small text elements: 'Kurt Schneider', 'Leibniz Universität Hannover', and 'SWT 2015/16 · 84'.

Dies ist die Folie aus Kapitel 1 (Ingenieursprinzipien),

Hier jedoch mit dem Wort „Requirements“ eingesetzt.

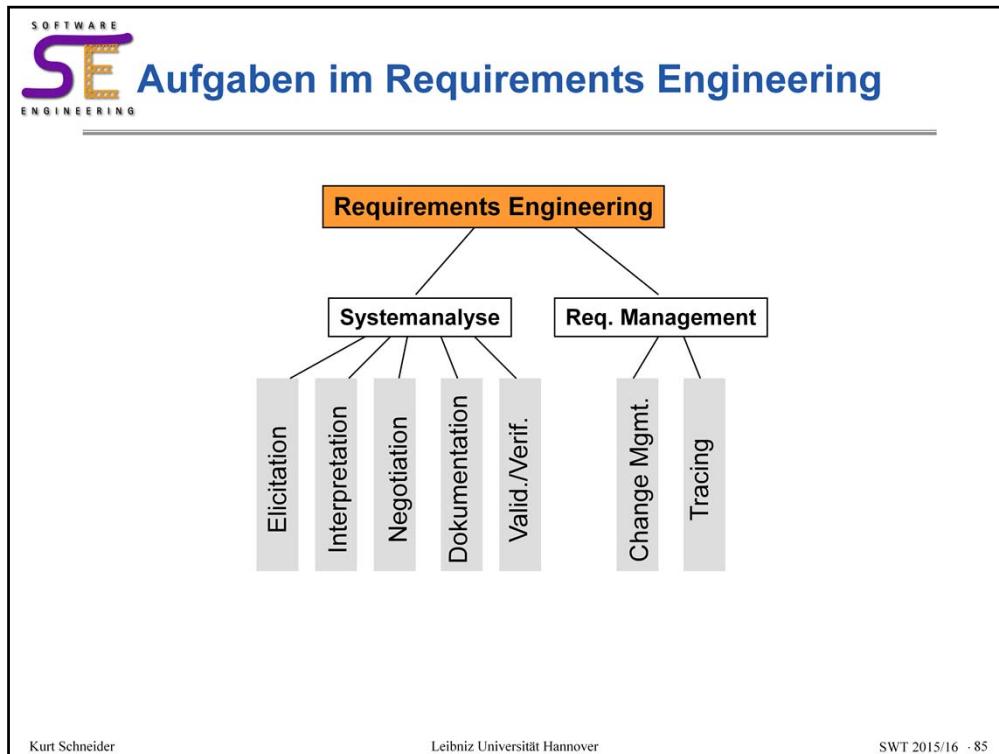
Es mag viele verblüffen, dass schon die Anforderungen *allen* Erfordernissen des ingenieursmäßigen Handelns unterworfen werden sollen.

Die Anwendung von Normen und Regeln oder die Wiederverwendung von Anforderungen liegt vielleicht nicht sofort auf der Hand. Auch darauf kommt es jedoch an: Man kann und sollte die Form der Anforderungsbeschreibung (z.B. UML) an existierenden Normen und (vielleicht firmenspezifischen) Regelungen ausrichten.

Daher ist es gerechtfertigt, von Requirements Engineering zu sprechen.

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

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



Requirements Engineering ist selbst schon wieder ein so kompliziertes Fach, von dem es mehrere Gliederungen und Strukturierungen gibt.

Das hier gezeigte Modell hat DaimlerChrysler auf einem Dagstuhl-Workshop gezeigt. Es unterscheidet zunächst zwischen der Analyse und dem Management von Anforderungen.

Beide Begriffe werden dann noch feiner unterteilt.

Andere Modelle verwenden ähnliche Begriffe, unterscheiden sich aber in der genauen Abgrenzung, auch von Req. Management und Analysis.

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

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



Anforderungen suchen: Elicitation

- „Herauskitzeln“ von Anforderungen
- **Grober Ablauf**
 1. Stakeholder herausfinden: wer ist betroffen?
 2. Umfeld des Systems erheben: Altsysteme, Schnittstellen, Doku.
 3. Sprechen mit Stakeholdern
 - „Sprechen“: Interviews, Workshops, Fragebögen
 - Beobachtung und Aufzeichnung
 - Regelrechte Elicitation-Techniken für „tacit knowledge“
- „Rohanforderungen“ müssen danach veredelt werden
- Beispiel Bankomat
 - Stakeholder: Kunden; die Bank; DB-Admin; Geldbefüller
 - Schnittstellen: zur DB, zu Schalterauszahlung u. Money-Mgmt.
 - Sprechen: Interview mit Schalterbeamten, Kundenfragebogen

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 86

Die Anforderungen zu finden ist schwer.

Aussagen, die man dann bekommt, sind aber oft noch nicht in einer Form, die man direkt umsetzen könnte. Oft (siehe Beispiele) müssen implizite Annahmen explizit gemacht werden, müssen vage Formulierungen geklärt werden.

Die folgenden Folien zeigen weitere Bearbeitungsschritte, die nötig sind, um aus den anfänglichen „Anforderungen“ die Basis und Referenz für weitere Entwicklungen zu machen.

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

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



Interpretation von Anforderungen

- Die Anforderungen werden inhaltlich bearbeitet
 - Dazu ist Interpretation (=Zusprechen von Bedeutung) nötig
 - Diese Interpretation kann falsch sein!
- Spreu vom Weizen trennen
 - Was ist essenziell, was Nebensache
 - Was ist wirklich gefordert, was nur Erläuterung
- Ordnen und Sortieren, Strukturieren
 - Ähnliche Anforderungen zusammen
- Detaillieren und Konkretisieren
 - Formulierungen schärfen (erfordert Interpretation)
 - Prüfbar machen
- Beispiel Bankomat
 - Identifikation: Stornierung ist vor „ok“-Button gefordert, danach nicht mehr
 - Sortieren: Anford. an die Bedienung; an die Geschwindigkeit; an die Funktion
 - Konkretisieren: Jede Kontostandprüfung muss in 5 s. fertig und bestätigt sein

1 Identifikation

2 Strukturierung

3 Konkretisierung

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 87

Anforderungen in ihrer Rohform sind noch nicht immer geeignet. Häufig müssen sie erst die obigen Schritte durchlaufen;

Form, Ausdrucksweise müssen vor dem Hintergrund des Kunden bewertet und oft auch interpretiert (ausgelegt) werden.

Praktisch niemals drückt sich ein Kunde sofort eindeutig und in der Sprache des Entwicklers aus, so dass dieser Schritt unterbleiben könnte.

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

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

Anforderungen verhandeln: Negotiation

- 1. Abhängigkeiten identifizieren**
 - Eine Anforderung erzwingt andere.
 - Beispiel: Online-Überweisung => Alpha-Tastatur => Sicherheitsanf.
 - Anforderungen widersprechen sich
- 2. Widersprüchliche Anforderungen identifizieren**
 - Bsp.: Mächtiges Werkzeug oder billiges Hilfsmittel?
 - Bsp.: Für Laien oder für Experten optimiert?
- 3. Konflikt ist üblich: Verhandlungsprozess, Moderation nötig**
 - Es gibt nicht eine objektive Wahrheit, sondern viele Interessen
- 4. Inkonsistenzen auflösen – oder auch nicht!**
 - Entscheidungsprozess
 - Selten eine technische Entscheidung
 - Mit gewissen Widersprüchen kann man leben

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 88

Wenn man mit verschiedenen Stakeholdern spricht, werden sich unterschiedliche Aussagen ergeben. Manche bedeuten das Gleiche, die sind kein Problem. Aber manchmal meinen Leute auch mit ähnlichen Worten sehr unterschiedliches, je nach ihrem Hintergrund („viele Zugriffe“: 1 Mio für Entwickler, 1000 für Betriebswirte, die das Medium noch wenig kennen).

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

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



Dokumentation

- Anforderungen zu hören reicht nicht 1 Fixieren
- Anforderungen müssen fixiert werden
 - Und trotzdem ändern sie sich!
 - Aber nur bei dokumentierten merkt man es2 Zerlegen in Einzelanforderungen
- Anforderungen dürfen sich ändern
 - Dann muss man nachdokumentieren
 - Und bezahlen muss das der Urheber (meistens)3 Mit Attributen beschreiben
- Sehr wichtig: **mehr** dokumentieren als reine Anforderungen
 - Gesprächspartner, Rollen, Ziele
 - Hintergründe und Randbedingungen
 - Annahmen
 - Absichten, Rivalitäten, alte Fehlversuche4 Anforderungen verbinden

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 89

Über Anforderungen wird im Projekt viel gesprochen.

Auf längere Frist reicht aber weder das gesprochene noch das im Kopf gemerkte Wort – statt dessen müssen Anforderungen auch dokumentiert, also aufgeschrieben werden.

Anforderungen ändern sich. Das macht Arbeit, weil die Dokumente geändert werden müssen. Trotzdem hat es keinen Sinn, Anforderungsänderungen zu verbieten oder so zu tun, als gäbe es sie nicht: man kann den Kunden höchstens daran hindern, sie zu äußern. Wenn Sie an das SW-Quanten-Modell denken, wird aber deutlich, dass dadurch kaum etwas besser wird.

Man wird nicht „einfach nur die Anforderungen“ aufschreiben, sondern sie mit wichtigen Zusatzinformationen anreichern. Diese sind meist Antworten auf die Frage, *Warum* eine Anforderung gestellt worden ist.

Dieser Aspekt wird in unreifen Organisationen und in kleinen Projekten nicht selten unterschätzt. Diese Informationen hat man ja „im Kopf“. Dort verblassen sie allerdings und werden nicht aktualisiert. Wer Hintergründe kennt, kann oft viele konkrete Anforderungen ableiten.

Wenn sich die Situation ändert, ändern sich mitunter mehrere Anforderungen, deren Grundannahme nicht mehr gegeben ist.

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

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



Verbesserte Noteneinsicht
Beispiel

Fiktives Beispiel

- Stellen Sie sich vor, es gibt eine Idee:
 - Noten sollen schon vor Einsichtnahme zu sehen sein
 - Aber noch nicht freigegeben – kann sich ja noch ändern!
- Daran hängen viele weitere Anforderungen
 - Von Studierenden
 - Von Lehrenden
 - Von Fachschaft
 - Von Prüfungsamt
- Werden erhoben wie oben beschrieben (Elicitation, Int., ...)

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 90

Nun folgt ein etwas längeres Beispiel. Die vielen Beispiele sollen Ihnen anschaulich machen, wie wichtig die Aktivitäten des RE sind und was man sich darunter vorstellen kann.

Hier die Aufgabe und das Umfeld.

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

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

 <p>Dokumentation (Auszug)</p> <hr/> <p>[R01] Studierende haben Zugang über Name und PIN Entscheidungstabelle, in Workshop abgesegnet (Negot.-WS)</p> <p>[R02] PIN fünfstellig, nicht mit Matrikelnummer korreliert Matrikelnummer steht öffentlich auf Studentenausweis (Dekan) Fünfstellige PINs ausreichend (Theorie-Vertreter)</p> <p>[R03] Alle Verbindungen zum Notenserver sind verschlüsselt Fordern Dekanat, Prüfungsamt, Dozenten. Studierende implizit.</p> <p>[R04] Zugriff nur auf die eigenen Noten möglich Kein online-Gesamtaushang, damit keine externen Robots (anonyme) Notenprofauswertungen vornehmen können</p> <p>[R05] Statistik ebenfalls nur dort sichtbar, nicht offen Siehe [R04]</p> <p>[R06] Alle eigenen Noten des laufenden Prüfungszeitraums zusammen bei einem Login zu sehen Studenten wollen sich weder mehrfach einloggen noch Klausurnamen merken</p>	<p>Fiktives Beispiel</p>
---	--------------------------

Kurt Schneider

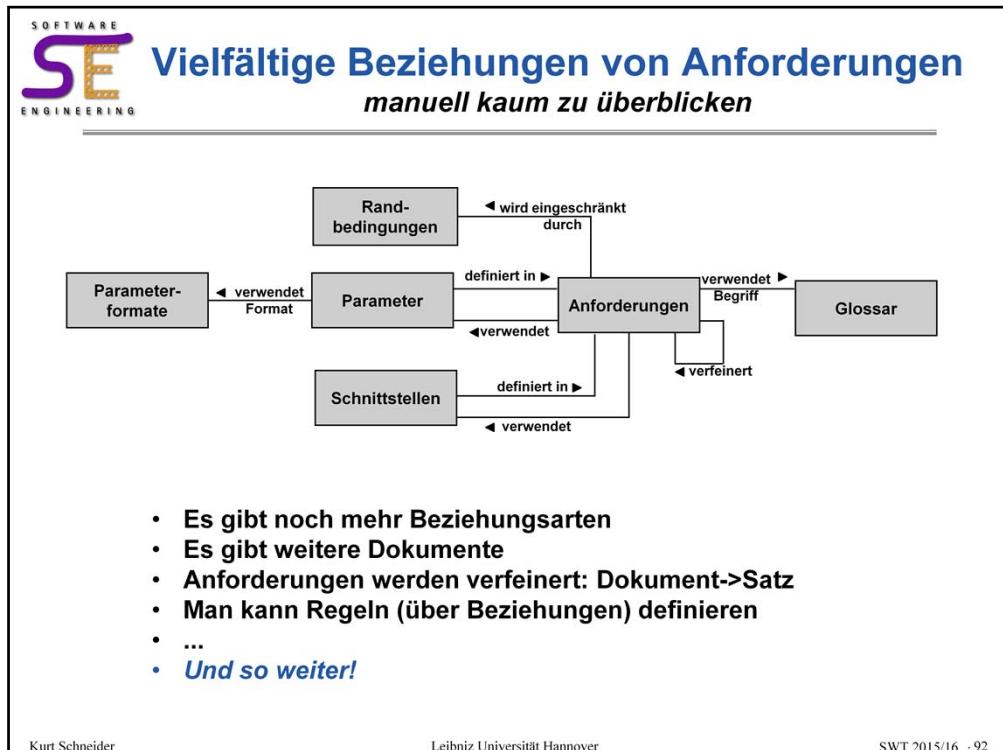
Leibniz Universität Hannover

SWT 2015/16 · 91

Dann ein paar schöne, nummerierte und kleinteilige Anforderungen.
Darunter steht in blau, wo sie jeweils herkommen.

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

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



Dieses Begriffsschema zeigt Anforderungen im Zentrum vieler anderer Begriffe. Im Prinzip muss man alle diese Informationen verwalten und ihre Beziehungen zueinander im Griff behalten. Dazu braucht man in der Regel Werkzeuge – und es gibt auch etliche Werkzeuge für genau diese Aufgabe.

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

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



Validierung und Verifikation

- **Begriffe**
 - **Validierung** :=
„sind die dokumentierten auch die wirklichen Anforderungen?“
 - Egal, was bisher aufgeschrieben wurde
 - **Verifikation** := „stimmen Anf. mit zuvor dokumentierten überein?“
 - Egal, ob der Kunde das (noch) wirklich will
- Validierung: Inhaltliche Prüfung
Ist der Entwurf das, was der Kunde will?


Kunde

Spez.


Entwickler


Entwurf

Erfüllt der Entwurf die Spezifikation?
Verifikation: Formale Prüfung

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 93

Mit Hilfe der SW-Quanten-Metapher werden hier die Begriffe Validierung und Verifikation dargestellt.

Verifikation prüft, ob ein Dokument alle Aspekte eines „verwendeten Referenzdokuments“ berücksichtigt hat. Ob das Referenzdokument (und damit das Ergebnisdokument) „gut“ oder „schlecht“ ist, interessiert dabei nicht.

Validierung dagegen prüft, ob der Inhalt eines Dokuments (oder das, was ein Entwickler denkt) mit der Überzeugung und dem Wunsch des Kunden übereinstimmt. Da der Wunsch des Kunden in seinem Kopf steckt und ja gerade sehr schwer fassbar ist, ist Validierung auch nicht so einfach. Sie ist insbesondere nicht automatisierbar, während bei Verifikation formale Verfahren eingesetzt werden können.

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

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



Requirements Management

- Die Verwaltung der Anforderungen
- Leicht handhabbar ablegen
 - Werkzeug (wie Doors, RequisitePro)
- Änderungen und Versionen im Griff behalten
 - Besonders interessant: auch die Gründe für Änderungen erfassen und aufbewahren!
- Änderungen weitergeben, Verfolgbarkeit in beide Richtungen (Tracing)
 - Das ist sehr schwer!
 - Schon seit Jahrzehnten ungelöst
 - Hier helfen Werkzeuge

```
graph TD; A[Req. Management] --> B[Änderungsmanagement]; A --> C[Verfolgbarkeit  
Tracing]; B --> D["Änderungs-  
wünsche  
  
Versionen von  
Anforderungen  
(Historie)  
  
Propagieren  
der Änderung  
(-> Tracing)"]; C --> E["Festhalten  
der Annahmen,  
Quellen von  
Anforderungen  
(pre-tracing)  
  
Auswirkungen  
von Anforderg.  
im System  
(post-tracing)"]
```

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 94

Das **Tracing** von Anforderungen ist ein wichtiges Thema. Und zwar interessieren beide Richtungen:

- Was ist aus einer Anforderung geworden, wie hat sie also die Entwicklung und das Projekt beeinflusst? Oft gibt es ja Konsequenzen an *mehreren* Stellen, die auf *eine* Anforderung zurückzuführen sind.
- Re-Tracing: Aufgrund welcher Anforderung(en) weist das System eine bestimmte Eigenschaft auf?

Oben geht es darum, von der Anforderung zur Entwicklungseinscheidung zu finden (Tracing), bei der zweiten Frage (Retracing) geht es anders herum.

Beide Verweisarten müssen verwaltet werden.

- Hier kommt man ohne Werkzeuge nicht aus, weil einzelne Anforderungen sehr feingranular (und damit auch zahlreich) auftreten können.
- Richtig interessant wird es, wenn sich dann einzelne Anforderungen oder Grundannahmen ändern: Nun müssen diese Änderungen und ihre Konsequenzen über mehrere Anforderungen hinweg nachgezogen werden. Ein gutes Änderungsmanagementsystem (bestehend aus Werkzeug, Abläufen und eingearbeiteten Leuten) kann auch dies nachvollziehen.

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

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



Diese Klassifikation müssen Sie auswendig kennen.

Oft muss man Anforderungen hier einordnen und sie dann in Spezifikationen schreiben.

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

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



Constraints

- Produkt-Anforderungen beschreiben, welche Funktionen oder Eigenschaften die Software haben soll.
- Prozess- und Projektanforderungen legen fest, mit welchen Mitteln und auf welchem Wege das Projekt ablaufen soll.
- Constraints schränken dagegen den Lösungsraum ein.
 - „MVC muss verwendet werden“
 - „Das Play!-Framework darf nicht verwendet werden“
 - „Unsere hausinterne Bibliothek ist vorzuziehen“
 - „Zur Planung muss MS Project verwendet werden“
- Es gibt Überschneidungen; Abgrenzung ist schwierig

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

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

Klassifikation (Fiktives Beispiel; Auszug)		Fiktives Beispiel
[R01] Studierende haben Zugang über Name und PIN <small>Entscheidungstabelle, in Workshop abgesegnet (Negot.-WS)</small>		Funktional
[R02] PIN fünfstellig, nicht mit Matrikelnummer korreliert <small>Matrikelnummer steht öffentlich auf Studentenausweis (Dekan) Fünfstellige PINs ausreichend (Theorie-Vertreter)</small>		Sicherheit Qualitätsanford.
[R03] Alle Verbindungen zum Notenserver sind verschlüsselt <small>Fordern Dekanat, Prüfungsamt, Dozenten. Studierende implizit.</small>		Sicherheit Qualitätsanford.
[R04] Zugriff nur auf die eigenen Noten möglich <small>Kein online-Gesamtaushang, damit keine externen Robots (anonyme) Notenprofilauswertungen vornehmen können</small>		Funktional
[R05] Dozent kann Jahresstatistik anzeigen <small>Genauer: NUR Dozent kann das (siehe R04)</small>		Funktional
[R06] Noten aller Prüfungen einer Person zusammen sichtbar <small>Studenten wollen sich weder mehrfach einloggen noch Klausurnamen merken</small>		Funktional

Ein paar Zuordnungen in unserem Beispiel.

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

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

Klassifikation		Fiktives Beispiel
(Fiktives Beispiel; Auszug, fortgesetzt)		
[R42]	Der jetzige Arbeitsablauf von Dozenten und Prüfungsamt ist zu beachten. Mehraufwand ist zu vermeiden ...	Funktional (Geschäftsproz.)
[R61]	Die technische Schnittstelle zu den HIS-Programmen steht nicht zur Disposition und muss beachtet werden.	Constraint
...		
[R82]	Das Programm soll bis zum Sommersemester operativ laufen	Projektford.
...		
[R89]	Alle Studierenden der Fakultät sollen dann alle neuen Klausurnoten einsehen können.	Mengengerüst
...		
[R90]	Eine Zugriffsrate von 200 Personen pro Tag, bis zu 20 gleichzeitig muss verkraftet werden und darf nicht zu Antwortzeiten von über 10 sek. führen	Mengengerüst Qualität/Geschw. unklar def.

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 98

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

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



Übersicht: Techniken um Anforderungen

- **Bisher:**
 - Wieso sind Anforderungen so wichtig?
 - Wo sind die Schwierigkeiten?
 - Aufgaben und Forschungsthemen im RE
- **Weiter mit Techniken zum RE:**
 - Denken in Objekten: Objekt-Orientierte Analyse
 - Abläufe im Zentrum: Use Cases
 - Testen und Anforderungen gehören zusammen

Elicitation
Interpretation
Negotiation
Documentation
Validation
Tracing

Anforderungen und Testen hängen eng zusammen

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 99

Softwaretechnik

Inhalt von Kapitel 3



Systematische Softwareentwicklung

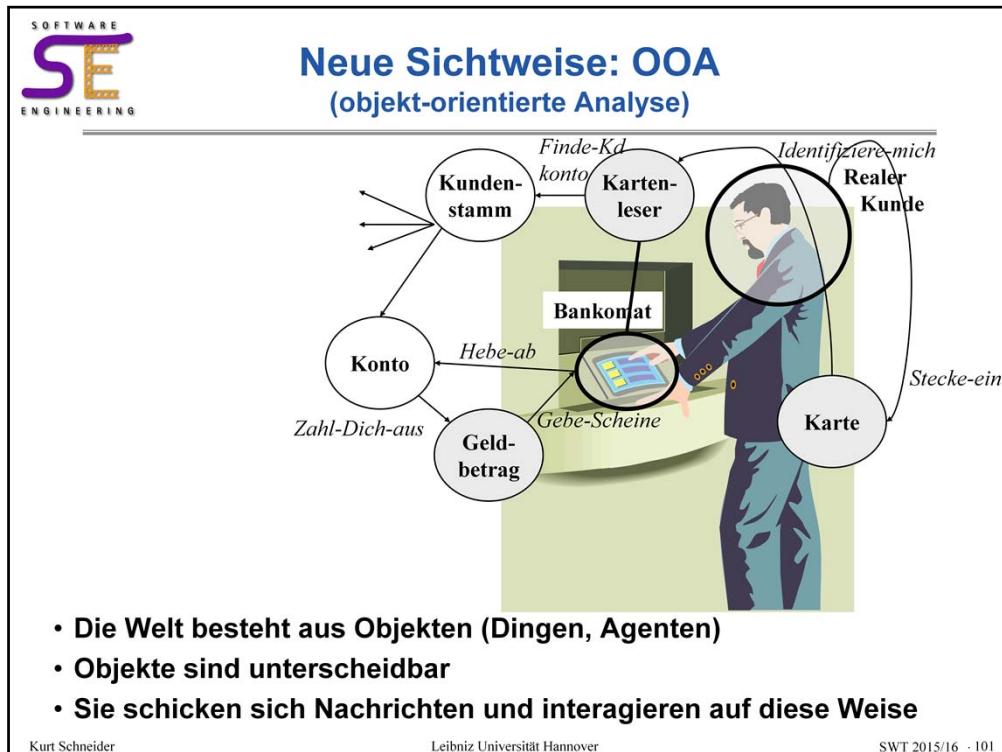
3. Anforderungen und Test: Basis des Projekts

- Aufgaben und Herausforderungen
- Objekt-Orientierte Analyse (OOA)
- Abläufe erfassen mit Use Cases
- Testen – schon jetzt

Leibniz Universität Hannover SWT 2015/16 · 100

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

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



Die OOA sieht überall Objekte.

Die Objekte können einander Nachrichten senden und darauf antworten.

Durch das Zusammenspieler vieler Objekte sollen die Anforderungen erfüllt werden. Offenbar ist es sinnvoll, solche Abläufe zu beschreiben, wenn man einmal die Objekte identifiziert hat.

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

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



Methode im Zusammenhang
Objekt-Orientierte Analyse



Erinnerung:

- es geht hier um die oo Analyse eines Systems und seines Umfeldes

Beschreibung Schritt für Schritt (=Methode)

1. Systemgrenzen festlegen
2. Objekte finden (stehen für reale Dinge, Konzepte etc.)
3. Objekte sortieren und klassifizieren (Domain Concepts)
4. Interaktion und Abhängigkeiten ermitteln

Schritte 1-4 werden zwei Mal absolviert:

- Zunächst IST-Zustand ermitteln
- Dann den Soll-Zustand

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 102

Das Prinzip (erst die Ist-Situation, dann das Soll modellieren) ist das gleiche wie bisher.

Man geht aber anders vor.

Systemgrenzen sind auch hier wichtig, aber dann geht es um Objekte und ihre Einsortierung in Klassen.

Danach betrachtet man, wie die Klassen interagieren.

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

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



Eine „Welt“, in der die Objekte sehr klar zu sehen sind, stellt dieses Computerspiel dar. Daran kann man die Schritte zu Objekten üben.

Jede reale Aufgabe wird aber schwieriger sein, weil die Abgrenzung nicht mehr so einfach ist. Auch liegt in dem Spiel bereits ein Computerprogramm vor, das man nur noch rückwärts analysiert. Das ist etwas anderes als ein von vorne neu modellierter Gegenstandsbereich, den man sich noch nicht so gut als System vorstellen kann.

Fazit: ein paar Schwierigkeiten der Analyse sind hier (absichtlich) weggelassen, damit Sie sich zunächst auf die genaue, konkrete Durchführung der Analyse-Schritte konzentrieren können.

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

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

SOFTWARE
ENGINEERING

1.Schritt: Systemgrenzen festlegen

Wie viele Levels? und so weiter... Highscore?

Musik?

Ladbare Elemente?

Internet-Option?

Spielstandspeicher?

Spieler-erweiterbar?

Kurt Schneider

Blast thru Special Edition, rockSolid Software ca. 2002

SWT 2015/16 · 104

Zunächst: Systemgrenzen sind selbst in diesem einfachen Fall nicht selbstverständlich. Es geht ja nicht nur darum, was zu einem Zeitpunkt gerade auf dem Bildschirm (oder in der Realität) zu sehen ist, sondern auch, welche Funktionen man im Hintergrund noch braucht.

Über die Systemgrenzen muss man *entscheiden*; das ist eine Verhandlungs-Aufgabe. Man wird die Systemgrenzen nicht durch Analyse „herausfinden“ können.

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

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



2. Schritt: Objekte finden

Soll-Analyse

„Entwickeln Sie ein Computerspiel, bei dem man einen Schläger (unten im Bild) lenkt.

Ein Ball wird ins Bild geschossen, er trifft auf Hindernisse (Steine, Bomben) und zerstört diese. Das gibt Punkte. An anderen prallt er ab.

Wenn der Ball am Schläger vorbei unten aus dem Bild läuft, ist er weg. Dann kommt der nächste Ball...“

Objekte suchen:
Substantive
unterstreichen

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 105

Die unterstrichenen Substantive sind gute Kandidaten für Objekte.

Manche kommen mehrfach (ähnlich) vor und können zu Klassen abstrahiert werden.

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 main title is "Objekte festlegen" with the subtitle "Danach kommen Entwurf u. OO Programmierung". A yellow box on the left contains the text "DANN" and a numbered list: "3. Identifizierte Objekte klassifizieren und Klassen ableiten" and "4. Beziehung und Interaktion festlegen", followed by "... weiter mit UML". To the right is a screenshot of a game level. The level shows a dark room with a floor labeled "Boden", a wall labeled "Wand", and several grey rectangular blocks labeled "Steine". A small orange ball is in the center. Two blue ovals, one labeled "Ball" and the other "Schläger", are also present. The bottom of the slide includes the name "Kurt Schneider", the source "Blast thru Special Edition, rockSolid Software ca. 2002", and the page number "SWT 2015/16 · 106".

Objekte zu identifizieren geht erst einmal ganz einfach.

Wichtig für das Spiel (allgemein: die Anwendung des Kunden) sind die Dinge, die in einer kurzen Beschreibung des Kunden von dem, was er möchte, vorkommen.

Wand und Boden sind dagegen nicht so offensichtlich und werden in aller Regel erst auf Nachfrage als Objekte genannt.

Sowohl Wand als auch Boden haben in diesem Spiel sowohl einen Zustand (Position) als auch ein Verhalten (Wand lässt Ball zurückprallen, Boden verschluckt ihn). Damit sind sie richtige Objekte.

Übrigens verhalten sich nicht alle Steine gleich; hier könnte man also von verschiedenartigen Objekten sprechen, die andererseits wieder viel gemeinsam haben (z.B. verschwinden sie, wenn sie getroffen werden und sie haben rechteckige Form).

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

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

Softwaretechnik

Inhalt von Kapitel 3



Systematische Softwareentwicklung

3. Anforderungen und Test: Basis des Projekts

- Aufgaben und Herausforderungen
- Objekt-Orientierte Analyse (OOA)
- Abläufe erfassen mit Use Cases
- Testen – schon jetzt

Leibniz Universität Hannover

SWT 2015/16 · 107

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

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

The diagram illustrates the definition and example of a Use Case. It features a logo for "SOFTWARE ENGINEERING" with "SE" in large letters. The title "Wichtig: Use Cases" is at the top. Below it, a list includes a definition in a yellow box: "Beschreibung einer Interaktion zwischen (min.) einem Akteur und dem System, durch das ein Ziel des Hauptakteurs erreicht wird." Another list item is "Beispiel Bankomat (Kern eines Use Case)". A detailed sequence diagram follows:

Akteur („Actor“): Kunden

Ablauf („Ablauf“):
1. Kunden führt Karte in Kartenschlitz ein
2. System fragt nach PIN
3. Kunden gibt PIN ein
...
4. System fragt nach gewünschter Aktion
5. Kunden wählt Abhebung und gibt Betrag ein
6. System gibt zuerst Karte aus, dann Geld
7. Kunden kann sich Quittung geben lassen
8. System geht in den Ausgangszustand über

Use Case „Geld abheben“

Ziel: Geld haben

Interaktion

Absichten, keine Interface-Details!

Sys-Grenzen? („Scope“)

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 108

Damit kommen wir zu den Use Cases. Die beschreiben Abläufe.

Sie vertragen sich zwar mit objekt-orientierten Konzepten – sind aber keineswegs auf sie zugeschnitten

In der Vorlesung gilt die gelb gerahmte Definition für Use Cases.

Unten sehen Sie an einem Beispiel, wie sich diese Definition in der Praxis niederschlägt.

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

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



Wozu Use Cases?

- Beschreiben das Systemverhalten
 - Das ganze, extern wahrnehmbare Systemverhalten
 - Aber nicht mehr als das (keine Interna, keine Qualitätsanforderungen)
- Meist von Entwicklern geschrieben
 - In Zusammenarbeit mit Fach-Experten
 - Diskussion führt zu Reflexion, klärt vieles („elicitation“)
- Sind Basis für alle weiteren Aktivitäten
 - Stellen Analyse-Schritt dar (Ist-Use Cases)
 - Repräsentieren Requirements (Soll- Use Cases)
 - Design muss Use Cases ermöglichen
 - Testfälle können aus Use Cases abgeleitet werden
 - Hinweis: Einfacher Use Case ist nahe an Testfall – ist aber keiner
 - Use Case: „Klasse“ von Abläufen. Testfall: ein ganz konkreter
 - Als „Contract“ (Vertrag) dienen sie beim Abnahmetest

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 109

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

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

**SOFTWARE
SE
ENGINEERING**

Formate für Use Cases

- Im Prinzip ist jedes Format möglich - wenn es hilft
- In der Literatur dominieren die „graphischen UML-Modelle“

Primary Actor Use Case Beziehung Actor (System)

- Aber Achtung!
 - Diagramme zeigen nur den Überblick
 - Welche use cases gibt es?
 - In welcher Beziehung stehen Actors und use cases?
 - Sie zeigen nicht: den eigentlichen „Kern des Use Cases“
 - die Abfolge der Schritte, Vor- u. Nachbedingung etc.

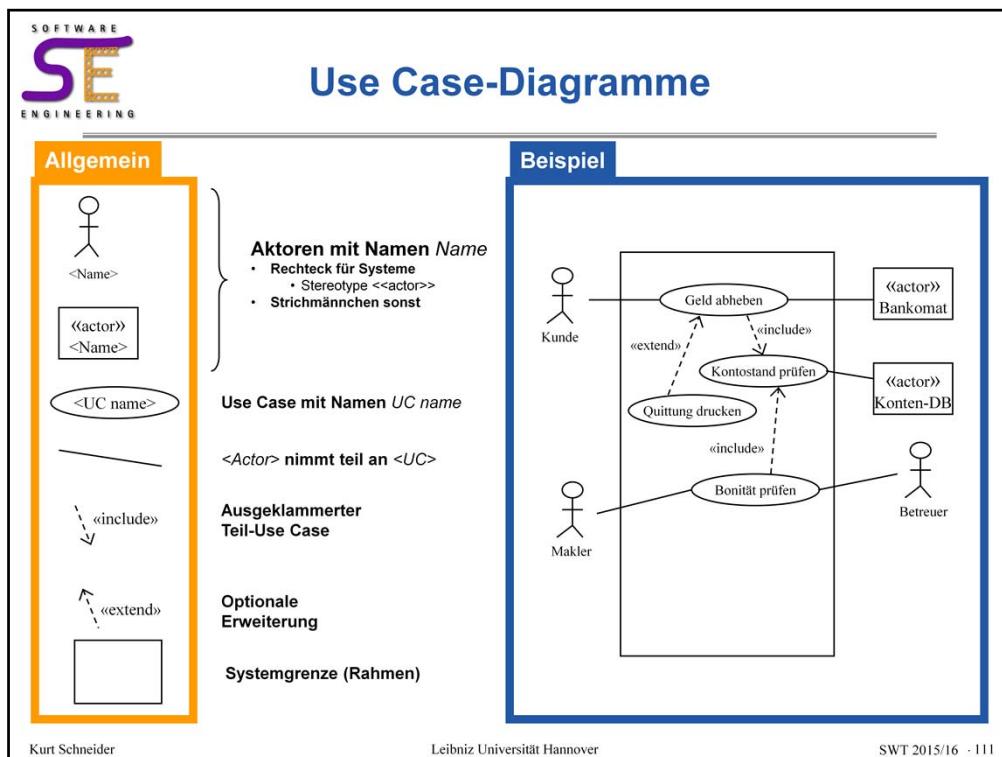
Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 110

Zwei Aspekte:

- Welche Use Cases gibt es zu einem System und wer ist in sie eingebunden? Dafür gibt es die UML-Diagramme mit den Strichmännchen. Man sieht darin aber nicht, was sich in dem Use Case ereignet.
- Was passiert in einem Use Case? Dieser mindestens ebenso spannende Aspekt, der sich durch Diagramme und UML nicht erfassen lässt. Auf ihn gehen wir noch ein.

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

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



An Use Cases sind die Diagramme nicht das Interessanteste.

Aber sie werden zu Übersichtszwecken oft gezeigt.

Beachten Sie die Richtung und Strichelung der Pfeile!

Die Striche zu den Akteueren sind keine Pfeile.

Includes und extends laufen in verschiedene Richtungen. Der Satz muss sich sinnvoll lesen:

Reise-buchen includes dafür-zahlen

Kreditkartenzahlung extends (für manche Fälle) Reise-zahlen (andere zahlen bar).

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

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



Formate für Use Cases

- Die Interaktionen können in vielen Notationen beschrieben werden
 - Ablaufdiagramm
 - Struktogramme
 - Petri-Netze
 - Pseudo-Code
 - Echter Code
- Wichtig: ihre Aufgabe ist Kommunikation unter Menschen
- Daher eignet sich für viele Use Cases besonders: *Text*
- *Aber den sollte man strukturieren*



Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 112

Da Use Cases (Innenansicht) ein „UML-freier Raum“ sind, kann man verschiedenste Darstellungsformen wählen.

Wie so oft im Software Engineering geht es hauptsächlich um den Austausch mit Menschen, nicht mit Computern. Danach sollte man die Wahl ausrichten.

Ein anerkannter Experte im Thema Use Cases (Alistair Cockburn) betont ausdrücklich, dass dafür Text oft das geeignete Medium sein kann.

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

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



Stile textbasierter Use Cases
nach Alistair Cockburn / (meine deutschen Entsprechungen)

- **Narrative / Geschichtchen** (noch kein Use Case)
„Peter braucht schnell Geld fürs Kino. Er betritt den Bankomatenraum und steckt seine ec-Karte ein. Dadurch wird er identifiziert und durch seine PIN authentifiziert. Er gibt den Betrag ein, erhält Geld und Karte.“
- **Brief / Kurzfassung**
– Ein Absatz: Zusammenfassung, oft des Haupt-Erfolgsszenarios.
- **Casual / Informell**
– Mehrere informell geschriebene Absätze, darin mehrere Abläufe.
- **Fully dressed / Detailliert**
– Am ausgefeiltesten.
In Tabellenform sind alle Schritte und Varianten detailliert aufgeführt. Zusatzangaben wie „Voraussetzungen“, „Garantien“.

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 113

Zunächst stellt er fest, dass man einen Use Case unterschiedlich ausführlich schreiben kann.

Man wird mit den kürzeren Formen beginnen und sich – bei Bedarf und NUR bei Bedarf – zu den längeren vortasten.

Es gibt ganze Bücher darüber, wie man gute Use Cases schreibt.

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

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



Elemente von Use Case Beschreibungen

- Akteure / Actors
 - außerhalb der Systemgrenzen
 - Interagiert mit dem System im Rahmen des Use Case
 - Hauptakteur (Primary Actor): der, dessen Ziel erreicht werden soll
- Stakeholder
 - Jemand mit Interesse am Use Case - muss nicht daran teilnehmen
 - Wer ist das? Fertigkeiten, Aufgaben, Hintergrund? -> Separate Tabelle
- Systemgrenzen / Scope
 - Was gehört noch zum modellierten System, was nicht mehr?
- Erfolgsfall und Misserfolgsfall / success and failure
 - Wenn der gewünschte Fall eintritt bzw. wenn er nicht eintritt
- Garantie
 - Was man auch im Misserfolgsfall noch sicherstellen kann

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 114

Auf jeden Fall kommen diese Elemente darin vor.

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

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

Formular für einen vollen Use Case		
Use Case Nr. <nr>	<Name: kurzer, aktiver Satz>	
Umfeld	<Wo, unter welchen Umständen ausgeführt>	
Systemgrenzen	<Scope: was gehört dazu, was nicht?>	
Ebene	<Überblick, Aufgabe oder Teilfunktion>	
Hauptakteur	<Kurzbeschreibung dessen, der Ziel hat>	
Stakeholder u. Interesse	Stakeholder <erste> <zweiter>	Interesse <ihre Interessen> <seine Interessen>
Voraussetzung	<wovon kann man bei Beginn ausgehen?>	
Garantien	<was in jedem Fall gewährleistet ist>	
Erfolgsfall	<Zustand bei erfolgreicher Beendigung>	
Auslöser	<wann der Use Cases startet>	
Beschreibung	Schritt	Aktion 1 <von Auslöser bis Aufräumen> 2 <z.B.: „...zurück in Ausgangsz.“>
Erweiterungen	1a	WENN ... DANN <and. Use Case>
Technologie ...	<Variationen durch Technik>	

in Anlehnung an Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, 2001

Leibniz Universität Hannover

SWT 2015/16 · 115

**Szenario :=
Ein Ablauf in
einem Use Case**

Das ist eine wichtige Folie.

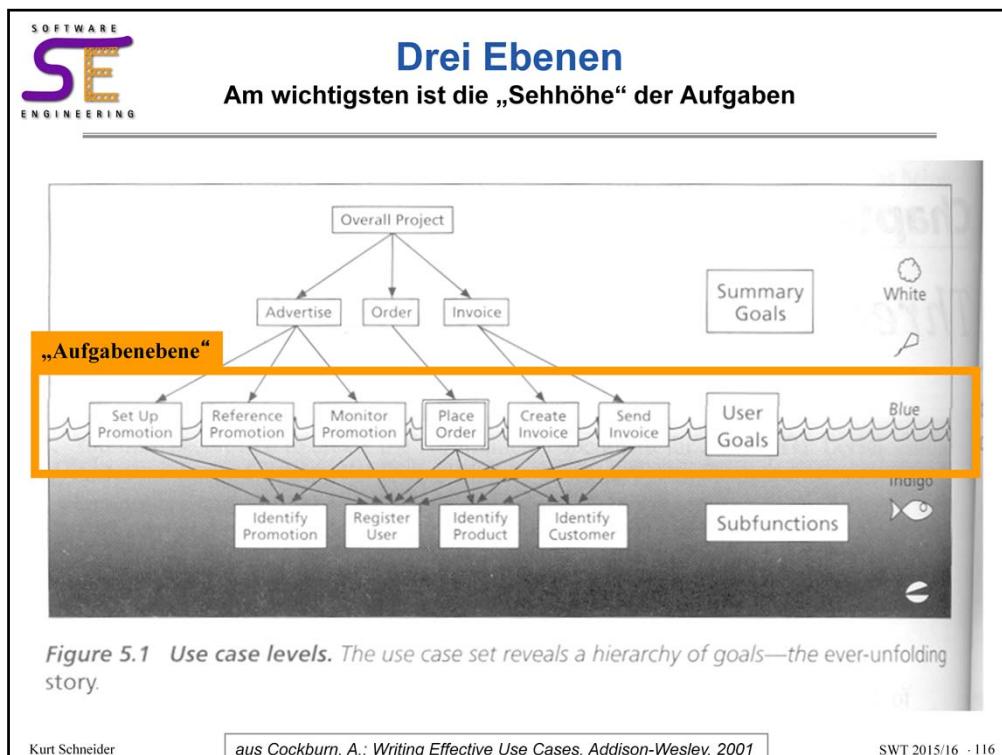
Man kann die wichtigen Elemente natürlich ganz verschieden aufschreiben.

Wie so oft hat es aber Vorteile, sich dabei an eine gemeinsame Vorlage zu halten. Es ist dann müßig, darüber zu streiten, ob man die Vorlage noch ein wenig optimieren könnte. Der Vorteil einer gemeinsamen Vorlage ist fast immer größer als der Nachteil, vielleicht nicht die absolut ideale Vorlage gefunden zu haben.

Daher: das obige Formular sollten Sie ein paar mal verwenden, bis Sie es gut kennen.

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

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



Der Titel der Folie ist ein Wortspiel:

„Seehöhe“ ist in diesem Bild die Ebene, auf der der Kunde sich zu Hause fühlt.
„Sehhöhe“ ist das, was er sieht.

Die Darstellung sagt:

Man kann natürlich Use Cases auf vielen verschiedenen Ebenen formulieren.
Beginnen sollte man aber auf der See/Sehhöhe.

Das wird Sie an die Strukturierte Analyse erinnern, wo man auch nicht top-down oder streng bottom-up vorgeht, sondern von der bedeutungsreichsten Ebene nach oben *und* unten.

Freilich kann das zu sehr vielen Use Cases führen; um den Überblick zu wahren, braucht man dann die UML-Diagramme von Use Cases (Außensicht).

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

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

Beispiel: Bankomat UC Geld abheben (Teil 1)	
UC 1	Geld abheben
Umfeld	Bankautomat im Freien oder im Foyer
Systemgrenzen	Kontoverwaltung und Bankomaten-HW existieren bereits
Ebene	Hauptebene
Hauptakteure	Kunde (dieser oder einer anderen Bank)
Stakeholder u.	Kunde möchte schnell und sicher Geld erhalten
Interessen	Bank möchte Personalaufwand reduzieren
Voraussetzungen	Kunde hat gültige Karte Bankomat läuft und hat Verbindung zum Banksystem
Garantie	Es wird entweder (Geld ausgezahlt und abgebucht) oder (weder ausgezahlt noch abgebucht)
Erfolgsfall	Der Kunde hat den gewünschten Geldbetrag erhalten; dieser Betrag wurde von seinem Kont
Auslöser	Kunde steckt Karte in den Leser
Beschreibung	<ol style="list-style-type: none">1. Kunde steckt Karte in den Leser2. System fragt nach PIN3. Kunde gibt PIN ein4. System fragt nach gewünschter Aktion5. Kunde wählt "Abhebung" aus6. System bietet Beträge an7. Kunde gibt Wunschbetrag an8. System prüft Verfügbarkeit und zahlt Betrag aus9. System schließt die Sitzung, gibt Karte aus.
	<p>Der hier beschriebene UC-Ablauf soll möglich sein. Weitere Anforderungen kann man annotieren/anfügen</p>
	[R27] verfügbar ist Guthaben plus Kreditrahmen
	Fortsetzung folgt ...

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 117

So kann man Use Cases zum Beispiel in Excel beschreiben. Dann ist es leicht, Zeilen einzufügen oder zu löschen. Den Use Case kann man in die Spezifikation kopieren.

Oder man verwendet im Textverarbeitungssystem gleich eine Tabelle.

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

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

Beschreibung	
1.	Kunde steckt Karte in den Leser
2.	System fragt nach PIN
3.	Kunde gibt PIN ein
4.	System fragt nach gewünschter Aktion
5.	Kunde wählt "Abhebung" aus
6.	System bietet Beträge an
7.	Kunde gibt Wunschbetrag an
8.	System prüft Verfügbarkeit und zahlt Betrag aus
9.	System schließt die Sitzung, gibt Karte aus.
Erweiterungen	
3a	WENN keine gültige PIN eingegeben, DANN zurück zu 2 (nach drei Fehlversuchen)
8a	WENN Wunschbetrag nicht verfügbar ist DANN
8a.1	bietet das System den höchsten verfügbaren Betrag an
8a.2	Kunde bestätigt, dass er ihn akzeptiert
8a.3	8a.2a: Falls nicht akzeptiert, beende Sitzung (9).
	System zahlt Betrag aus, weiter bei 9
Technologie	
Bei Geräten mit Irisensor ist alternativ zur PIN-Eingabe auch ein Iris-Scan möglich	

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 118

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

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

Beispiel 2: UC Überweisung tätigen	
Teil 1	
UC 2	Überweisung tätigen
Umfeld	Bankautomat mit Alpha-Tastatur im Foyer
Systemgrenzen	HW und Bankensoftware werden unverändert übernommen
Ebene	Hauptebene
Hauptakteure	Kunde dieser Bank
Stakeholder u.	Kunde möchte schnell und direkt vor Ort Überweisung durchführen
Interessen	Bank möchte Aufwand für Beleglesen sparen
Voraussetzungen	Kunde hat ein überweisungsfähiges Konto bei dieser Bank Bankomat läuft und hat Verbindung zum Banksystem Kunde hat sich über seine Karte und PIN identifiziert und authentifiziert
Garantie	Am Ende der Sitzung wird unmissverständlich angezeigt, ob die Transaktion erfolgreich war
Erfolgsfall	Die Überweisung ist abgeschickt und ins normale Banksystem übergeben
Auslöser	Kunde wählt Aktion "Überweisung" aus
Beschreibung	<ol style="list-style-type: none">1. Kunde wählt Aktion "Überweisung" aus2. System weist darauf hin, dass mTAN erforderlich ist3. Kunde bestätigt4. System zeigt Formular wie auf Papier an5. Kunde füllt das Formular aus6. System verschickt mTAN per SMS, fragt Kunden danach7. Kunde gibt mTAN ein8. System übergibt Überweisung an das Banksystem9. System <u>bestätigt erfolgreiche Überweisung</u>10. System beendet die Sitzung
	Fortsetzung folgt ...

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 119

Unterstrichene Teile stehen für Use Cases, die hier inkludiert sind, also „aufgerufen“ werden.

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

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



Beispiel 2: UC Überweisung tätigen

Teil 2

Beschreibung	<ol style="list-style-type: none">1. Kunde wählt Aktion "Überweisung" aus2. System weist darauf hin, dass mTAN erforderlich ist3. Kunde bestätigt4. System zeigt Formular wie auf Papier an5. Kunde füllt das Formular aus6. System verschickt mTAN per SMS, fragt Kunden danach7. Kunde gibt mTAN ein8. System übergibt Überweisung an das Bankensystem9. System bestätigt erfolgreiche Überweisung10. System beendet die Sitzung
Erweiterungen	<ol style="list-style-type: none">3a. WENN Kunde nicht bestätigt, DANN Abbruch (10)5a. WENN Pflichtfelder nicht ausgefüllt, DANN weist System auf Fehlendes, zurück zu 55b. WENN Kunde abbricht, DANN beende Sitzung (10)7a. WENN keine gültige mTAN, DANN zurück zu 6. Sperrung nach drei Fehlversuchen9a. WENN Kunde Beleg wünscht, DANN 9a.1 System druckt Beleg, beendet Sitzung (10)
Technologie	In Filialen mit Fingerkuppenleser ist keine TAN nötig

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 120

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

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

Use Case-Template		
hier beispielsweise für Excel; finden Sie auf StudIP		
UC <Nr.>	<Name: kurzer, aktiver Satz, was erreicht werden soll>	
Umfeld	<informell: wo und unter welchen Umständen ausgeführt>	
Systemgrenzen	<was gehört dazu, was nicht?>	
Ebene	<Überblick, Aufgabe bzw. Hauptaufgabe oder Teilfunktion>	
Hauptakteure	<wer hat das Ziel und welches?>	
Stakeholder u. Interessen	<Bezeichner>	<Interessen>
	mehrere üblich	
Voraussetzungen	<davon kann man bei Beginn ausgehen>	
Garantie	<In allen erwähnten Erfolgs- und Misserfolgsfällen sichergestellt>	
Erfolgsfall	<was im Idealfall erreicht wird>	
Auslöser	<Wodurch wird die Interaktion gestartet?>	
Beschreibung	1	erster Schritt ist meist der Auslöser
	2	
	3	
Erweiterungen	2a.	WENN <Bedingung, um in 2 abzuweichen>
		DANN <Aktion in diesem Fall, evtl. mehrere Schritte: 2a.1, 2a.2
Technologie	3a	Besonderheit wg. Technologievarianten

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 121

Die blauen Teile sind Platzhalter. Sie erläutern, was man in das jeweilige Feld schreiben soll.

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

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



Wie man Use Cases einsetzt

- 1. Akteure identifizieren**
 1. Wer benutzt das System? Wozu? → Stakeholder-Profile-Tabelle
 2. Wer ist mit Ein- und Ausgaben konfrontiert? → Akteure
- 2. Systemgrenzen festlegen**
 - Akteure sind außen, Use Cases beschreiben das Hin und Her
- 3. Wichtigste Use Cases identifizieren**
 - Narrative, Beziehung zwischen Use Case und Actor herstellen (UML)
- 4. Zunehmend genauer beschreiben**
 1. Narrative (Geschichtchen)
 2. Kurzfassung -> Informell oder Detailliert
 1. Erfolgsszenarien
 2. Fehlerszenarien
 3. Use Cases mit dem Kunden diskutieren
 4. Bei Bedarf: grafische Übersicht (UML, Beziehungen) ergänzen

Zwei Schleifen: über alle Use Cases, durch die Ebenen; mit Kunden

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 122

Use Cases entstehen. Es ist auch wichtig zu wissen, wie man dabei vorgeht.

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

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



Wann ist man fertig?

- Fertig, sobald
 - alle Hauptakteure identifiziert wurden
 - ihre Ziele genannt haben
 - Jedes Ziel von mindestens einem Use Case abgedeckt ist.
- Und wenn alle Use Cases klar und verständlich sind:
 - Sponsoren bestätigen („agree“), damit Abnahme machen zu können
 - Nutzer bestätigen, dass Systemverhalten zutreffend beschrieben
 - Sponsoren bestätigen, dass die Use Cases (vorerst) vollständig sind
- Natürlich ist man dann *nicht endgültig* fertig!

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 123

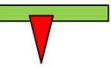
Wieso ist man nicht wirklich fertig? Man ist nicht fertig, weil ein SW-Projekt, solange es nicht „tot“ ist, sich ständig ändert.

Hilfreiche Techniken eignen sich daher auch für Änderungen.

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

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

 **Tipps und Tricks**

- Use Cases erfassen *nur funkt.* Anforderungen, die aber *alle*
- Zuerst in die Breite arbeiten, dann in die Tiefe 
- Eher zu wenig als zu viel schreiben
 - Sonst liest es doch keiner, dann hilft es nichts
- Es geht um das Geschriebene (UC), nicht das Gemalte (Diag.)
- Diskussion über Formalität und Stil nicht übertreiben
 - Es kommt letztlich halt auf Klarheit an - egal, wie
- Keine „if-Konstrukte“ in Hauptszenarios
 - Sond. Extensions/Abzweig-Szenarien

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 124

Sie sollten sich mit dem Formular – als dem Träger vieler Konzepte – gut vertraut machen und auch die Übungen und die Übungsklausur dazu noch einmal durchsehen.

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

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



Wiederholung: Anforderungen

- Anforderungen: Bezugspunkt von SW-Entwurf, -Entwicklung, -Test
- Anforderungen sind aber nicht leicht zu bekommen
 - Sie erfordern Kommunikation mit „Fach-Leuten“
 - „Symmetry of ignorance“
 - Alle Schwierigkeiten üblicher Kommunikation eingeschlossen
- Requirements Engineering umfasst mehrere Aktivitäten
 - Elicitation: Herauszitzen
 - Interpretation: Das Gehörte deuten
 - Negotiation: Priorisieren, Widersprüche auflösen, Verhandeln
 - Dokumentieren: Aufschreiben
 - Validation & Verification: Prüfen
 - Managen und Tracen: Verfolgen, Änderungen einarbeiten
- Wichtige RE-Techniken
 - Structured Analysis (SA): Anforderungen aus Datensicht [historisch]
 - Objekt-orientierte Analyse: Die Welt aus interagierenden Objekten
 - Use Cases: Abläufe im Zentrum

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 125

Nach dem Blick auf SE werden wir nun feststellen, dass SA auch manche Schwächen hat. Da kommt die Essenzielle Systemanalyse (in diesem Abschnitt) zu Hilfe, denn sie versucht, diese Fehler zu vermeiden.

Das Schlüsselkonzept ist die Konzentration auf den eigentlichen Kern eines Systems, auf seine Essenz. Alles darum herum kann man erst einmal vernachlässigen. Der Begriff der Essenz ist genau definiert.

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

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

Softwaretechnik

Inhalt von Kapitel 3



Systematische Softwareentwicklung

3. Anforderungen und Test: Basis des Projekts

- Aufgaben und Herausforderungen
- Objekt-Orientierte Analyse (OOA)
- Abläufe erfassen mit Use Cases
- Testen – schon jetzt

Leibniz Universität Hannover

SWT 2015/16 · 126

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

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



Was ist Testen?

=_{Def} „Ausführen eines Programms mit dem Ziel, Fehler zu finden“

- Das heißt also:
 - das Programm muss ausgeführt werden
 - jeder gefundene Fehler ist ein Erfolg für den Tester
 - Für den Entwickler nur indirekt
- Testen ist also nicht:
 - Wunsch zu zeigen, dass das Programm in Ordnung ist
 - Herumprobieren-Ändern-wieder Probieren
 - ein unmittelbar konstruktiver Verbesserungsschritt
- Fazit: Testen erscheint zunächst destruktiv
 - verlangt gewisse Schadenfreude: Autor hat die nicht!
 - enthält viele kreative Elemente (was könnte schiefgehen?)

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 127

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

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



Hier werden die Testvarianten nur kurz und im Überblick besprochen. Lediglich Black-Box-Testfälle werden Sie danach erstellen können.

Dabei kann man noch sehr viel raffinierter vorgehen, kompliziertere Techniken einsetzen und mit Glass Box Test kombinieren. Das wird in der Vorlesung Software-Qualität beschrieben.

Hier geht es darum, den Sinn und den Nutzen von Tests allgemein einzusehen und wenigstens einfache (Black Box-) Tests schon einmal entworfen zu haben. Tests sind kein Anhängsel am Ende sondern das logische Gegenstück zu Programmen: Nur ein getestetes Programm kann man ausliefern. Tests helfen, Fehler zu finden, bevor sie Schaden anrichten.

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

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



Beispiele für Testfälle

Einfache Fälle: mathematisch und klar

```
public int quersum (int zahl)
    - [ R01 ] die Methode gibt die Quersumme einer zahl aus
```

– Testfälle: (was geht rein? → was muss dann rauskommen?)

- T01: quersum (123) → 6
- T02: quersum (1234) → 10
- T03: quersum (12345) → 15
- T04: quersum (123456) → 21
- ... und so immer weiter?

Req. Engineer:
„Aha, keine einstellige / iterierte
Quersumme, denn da wäre
 $itQuer(1234) =$
 $itQuer(10) = 1$ (< 10 !)
Anforderung R01 ist jetzt klarer“

```
– [ R02 ] ein negatives Vorzeichen wird ignoriert
    • T42: quersum (-529) → -16
```

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 129

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

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

**SOFTWARE
ENGINEERING**

Beispiel: Kassensoftware testen

K-Market Citykumpu
Kummunkatu 9
83500 OULU
Puh. 013-316 712

Sprite virvoitusjuoma 1,5l	2.09	B
Pullopantti 0,40	0.40	E
VAASAN 6 viljan leipä 520 g	2.20	B
Pullopalaatus	-0.40	E
=====		
YHTEENSÄ	4.30	
ANNETTU	5.00	
TAKAISIN	0.70	

ALV NETTO VERO BRUTTO
17.00 % 3.67 0.62 4.29 B

Palvelemme Ma-Pe klo 7-21, La 7-18
Sallittuina aikoina Su klo 12-21
Mitä tään syötäisiin?
Ruokavinkit RUOKA-PIRKKA -lehdestä!
Y-tunnus 1759047-3
#840055 C:000009 K:02 22.07.2008 14:56

4.30

- Einzelposten gegeben
- Gesamtsumme ermitteln (YHTEENSÄ)
- Und Barrückgabe (TAKAISIN)
- Zu einfach?
 - Es sind aber finnische Belege!
 - Kunde ist ein finn. Supermarkt
- Na und?
 - Dann sehen Sie mal genau hin.

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 130

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

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

The screenshot shows a receipt from Alko Oy. At the top right, there is a blue banner with the text "Finnland verwendet keine 1c, 2c Münzen". Below this, a bulleted list states: "Daher wird gerundet", "Auf- und Ab; das mittelt sich aus", and "Idealweise explizit ausgewiesen - „Pyöristys 0,01“". The receipt itself has a red box around the total amount "38.25".

Alko Oy
Oulu Linnanmaa
Y-TUNNUS: 1505551-4

002042 LAPONIA TYRNI 0,5 2 10.49 D
000041 KARJALA A 0,33 1.28 D

=====
YHTEENSÄ 11.77
ANNETTU 50.00
TAKAISIN 38.25

ALV NETTO VERO BRUTTO
22.00 % 9.65 2.12 11.77 D

TUOTELKM 2
Kiitos käynnistä, tervetuloa uudelleen!
Palvelenne: ark 9-20, la 9-18
Puhelin: 020 711 2830 Faksi 020 711 4830
ml2830@alko.fi, www.alko.fi
#130801 C:013280 K:03 19.07.2008 16:04

EUROMARKET LINNANMAA
Kauppalinnaankuja 1-3
90570 OULU
puh. 08 - 8181 300

Kassa 10/46 Kuitti 326472 19.07.08 15:58

	A 4	-0,10
PULLONPALAUTUS	3	0,89
FAZER SALMIAKKI		

Pyöristys 0,01
Maksettava yhteenä 0,80
Käteinen 1,00
TAKAISIN 0,20

Tradeka Oy	Y-tunnus	1905481-8	
ALV.PROS	VEROTON	VERO	YHTEENSÄ
3 17,00	0,76	0,13	0,89
4 22,00	-0,07	-0,02	-0,09
YHT.	0,69	0,11	0,80

KIITOS KAYNNISTA
TERVETULOA UUDELEEN !
SÄILYTÄ KUITTI

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 131

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

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

**SOFTWARE
ENGINEERING**

Aber das scheint noch nicht alles zu sein

- Nach einiger Zeit stoßen Sie auf diesen „Testfall“:

Was tun Sie jetzt?

[R02] Bei Kartenzahlung wird nicht gerundet

Rationale (Begründung):
„Bei Kartenzahlung muss man ja keine Münzen herausgeben – da kann man exakt zahlen“

ELDOR PEPPERONI 300/1	3	1,49
ELD MARGARINI 60%	3	0,85
F RANSKALAINEN PATON	3	1,49
VALKOTINEN SUKLAAN	3	1,09
RUUKKUSAL. I LK	3	0,99
SIPULI SUOMI	3	1,45
ESKIMO PAKPSS 1L	4	1,55
MS VAPAA KANAMUN M10	3	2,19
MAKUMAASTA 300G SPEC	3	2,09
CREME BON RUOHOS	3	1,35
SVEITSINPAH. 200G 2,000 x 1,79	3	3,58
SM KUMINA 18G PUSSI	3	0,85
SM BASILIKÄ 11G	3	1,25
3 VÄRIMIX PAPRIKA	3	2,49
Maksettava yhteensa		56,22
Luottokortti 30500		56,22
EUROCARD/MASTERCARD		
T = Tarjous		
Tradeka Oy		Y-tunnus 1905481-8
ALV, PROS	VEROTON	VERO YHTEENSA
Kurt Sc	3 17,00	47,84 8,13 55,97

Leibniz Universität Hannover

SWT 2015/16 · 132

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

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



Finnische Quittung

Beispiel aus dem echten Leben

In Finnland gibt es keine 1c und 2c-Münzen. An der Kasse im Supermarkt wird der Endbetrag daher auf- oder abgerundet.

[R01] Die Summe der Einzel-
beträge wird **auf volle 5c**
mathematisch gerundet.

[R02] Bei Kartenzahlung
wird nicht gerundet

Eingaben	Soll/Ausgaben
T01: finCash (8,42 €)	→ 8,40 €
T02: finCash (1.043,43 €)	→ 1.043,45 €
T03: finCash (18,00 €)	→ 18,00 €
T04: finCard (1.043,43 €)	→ 1.043,43 €
T05: finCash (-1,21 €)	→ 1,20 € Pfand zurück

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 133

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

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

**Software
Engineering**

Von der Anforderung zum Testfall direkter Zusammenhang

Phishing, nnichtsahn

ID	Kurzbeschreibung	Anforderung	Rationale
[R01]	Phishing verhindern	ec-Karte soll mehrfach teilweise eingezogen und ausgegeben werden	Vorsatzgeräte können dann nicht mehr den Streifen lesen
[R02]	PINs haben genau vier Stellen	Akzeptiere nur PINs mit exakt vier Stellen	Internationaler Standard; auch nicht mehr Stellen
[R03]	Alle vierstelligen Zahlen (0..9) sind als PIN erlaubt	Alle vierstelligen Zahlen aus den Ziffern 0 bis 9 sind als PIN erlaubt	Einige Banken erlauben das, also muss überall diese Eingabe erlaubt sein

	Testfall: SetUp	Soll
T01.1	Karte in Spalt geben	Wird min. zwei Mal in jede Richtung bewegt
T02.1	Beim Verändern der PIN vierstellige wählen	wird akzeptiert
T02.2	Beim Verändern der PIN fünfstellige wählen	wird zurückgewiesen
T03.1	PIN Zahl im Mittelfeld wählen (z.B. 4711)	wird akzeptiert
T03.2	Als PIN den Grenzwert 0000 wählen	wird akzeptiert

- Kann noch hinzukommen:**
 - setUp() von Datenbanken, Ausgangssituation
 - Aufwändige Beschreibung der Vorbereitung/Durchführung
 - Angabe von Toleranzen

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 134



Gute Anforderungen und Testfälle

- **Gute Testfälle**
 - Konkret
 - Spezifisch
 - Leicht anwendbar
 - Objektiv beurteilbar
 - Testbar
 - So ausgewählt, dass sie viele Fehler finden!

- **Anforderung:**
 - was soll Programm tun?
- **Test:**
 - Tut das Programm, was der Test verlangt?
- **Ideal:**
 - Programm besteht alle Tests
 \Leftrightarrow
 es erfüllt die Anforderungen

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 135

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

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

Use Case vs. Testfall ähnlich, aber nicht dasselbe	
Beschreibung	Schritt Aktion 1 Kd drückt „Überweisung“ 2 Hinweis erscheint, dass er TAN braucht. Kd quittiert. 3 Formular wie Papier erscheint 4 Kd füllt die Felder aus 5 Ist Kd fertig, schaltet er weiter 6 Ausgeföllte Überweisung wird gezeigt, nach TAN gefragt 7 Kd gibt (verdeckt) TAN ein 8 Sys fragt, ob Beleg gewünscht 9 Sys druckt Beleg 10 Sys verabschiedet sich, Ruhezust.
Erweiterungen	2a WENN Kd keine da hat DANN beendet er die Aufgabe 5a WENN ein Feld fehlt oder falsch ist DANN fragt System nur nach diesem Feld. Weiter: 6 7a WENN ungültige TAN, DANN Fehlermeldung, zurück zu 6
Technologie ...	In Filialen mit Fingerkuppenleser keine TAN nötig

Und hier den zweiten. Die *Beschreibung* ist am ausführlichsten. Hier steht eigentlich der Ablauf.

Beachten Sie, dass nur der normale Hauptablauf in der Beschreibung steht; jede Variante oder jeder Sonderfall wird in die Erweiterungen ausgelagert. Das ist gewöhnungsbedürftig, aber nun einmal so üblich.

Die vielen anderen Angaben sind aber genauso wichtig!

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

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



Vollständig testen: fast nie möglich

- Abschätzung:
 - Für einen 16-bit-Parameter gibt es 2^{16} mögliche Werte
 - Für drei Parameter schon $2^{48} = 2,8 \cdot 10^{14}$
 - Bei 100 Tests pro Sekunde wären das über 10^{12} Sekunden - fast 90.000 Jahre!
- Oft gibt es noch mehr Permutationen
 - Viele Programme haben mehr als drei Parameter
 - Dazu kommen Sonder- und Fehlerfälle
 - Oberflächen führen zu noch mehr Varianten
- Es reicht nicht, die *Eingaben* zu erzeugen! Auch *Sollwerte* für Test nötig!
- Erstaunlich: in Einzelfällen ist vollständiges Testen trotzdem möglich; und es lohnt sich! Beispiel: Kleines Steuergerät
 - Parameterwerte nur 0 bis 255
 - In der Regel nur ein oder zwei Parameter ($255^2 = 65.025$, im 10 Min.-Bereich)
 - Ergebnisse leicht ableitbar / formal spezifiziert (über Formel)

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 137

Vermeiden Sie den Ausdruck „ausgetestet“ und ähnliche Aussagen. Sie zeigen höchstens, dass der Sprecher nicht weiß, wovon er redet.

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

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



Einige Definitionen

Prinzip

Prüfling =_{Def} zu prüfende Software (*nicht*: deren Autor!)

Autor =_{Def} Verfasser des Prüflings

Laufzeitversuch =_{Def} Entwickler spielt herum, um zu sehen, „ob Programm läuft“
selbstverständliche Voraussetzung für Test - nicht mehr

Test =_{Def} Ausführung eines Programmes mit dem Ziel, Fehler zu finden
„Test“ wird von manchen auch anders verstanden: oft Missverständnisse
Testvorschrift: Anweisung, wie Test abzuwickeln ist (inkl. Setup)
Testprotokoll: standardisiertes Formular zur Doku. über jeden gefundenen Fehler

Sonderfall Abnahme(test) =_{Def} Offiziell/juristischer Akt zur Feststellung,
ob Software die an sie gestellten Anforderungen erfüllt.
– Kriterium: ausschließlich die Abnahmetestfälle
– Abnahme erfolgt, falls Abnahmetestfälle ohne Fehler durchlaufen
– Hier geht es also nicht vor allem darum, Fehler zu finden

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 138

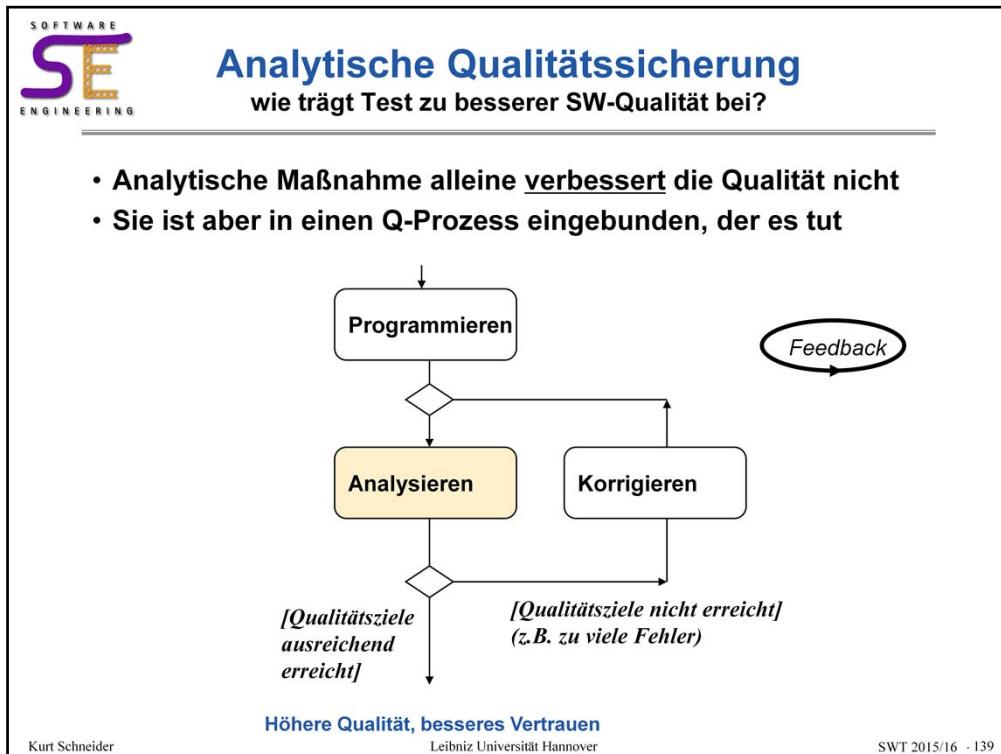
Diese Definitionen müssen Sie unbedingt kennen.

Beachten Sie bitte, dass Abnahmetests einen anderen Anspruch und folglich auch eine ganz andere Herangehensweise haben als normale Entwicklertests!

Grob gesprochen möchte beim Abnahmetest der Entwickler gerade keinen Fehler mehr finden – ein fundamentaler Unterschied.

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

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



Scheint trivial zu sein, ist es aber nicht.

Letztendlich geht es um Fehlerbeseitigung und ganz am Ende um besseres Vertrauen, das der Kunde – und die Entwickler! – in ihre Software haben können.

Das ist also wieder eine Feedback-Schleife: die Befunde aus den Analytischen Verfahren werden verwendet, um korrigieren zu können.

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

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



Fehler in Tests Hintergrund

Abweichung von Soll und Ist bedeutet Fehler in

falschem	Ist	(„falsche Ausgabe“)
oder falschem	Soll	(„falsche Vorgabe“)
oder ungeeignetem	Vergleich	(„falscher Test“)

Nicht jede Abweichung ist daher Indiz für Programmfehler obwohl man die eigentlich sucht!

Andere Möglichkeiten

- Testfall war falsch Programm korrekt, **Soll falsch aus Anf. ermittelt**
- Anforderung falsch interpretiert Soll korrekt ermittelt, **Anf. missverstanden**
- Vergleich entspricht nicht den Erfordernissen
Toleranz, Kriterien oder Typ falsch
- Kombination mehrerer Gründe (das ist am übelsten)
Problem: woher weiß man, welcher Fall vorliegt?

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 140

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

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

„Testbarkeit“ eines Programms

„Ist das (Teil-) Programm so gestaltet, dass Testen einfach ist?“

- Leicht mit Eingaben und Parametern zu versorgen 
- Ergebnisse leicht erkennbar 
- Programmcode leicht verständlich (Stil und Struktur) 
- Evtl. Zwischenergebnisse „abzweigbar“ 
- Nicht viele Außenbeziehungen 
- Wenige Einflussgrößen vorhanden (nicht viele Kombinationen)
 - Diese sind alle gut kontrollierbar
 - Damit sind Ergebnisse reproduzierbar
- Ausführung des Prog. ist technisch und organisatorisch einfach 
 - Zugänglich, nötige Hardware vorhanden
 - Billig (Rechenzeit, Bediener etc.)
 - Wenig Zusatzausrüstung/Vorlaufzeit nötig

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 141

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

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

Tätigkeiten und Dokumente beim Testen

Vorgehensweise

- **Test planen**
 - **Resultat:** Testplan
 - Vorgehen: wer tut wann was?
 - Testgeschrif
- **Testfälle erstellen**
 - **Resultat:** Testfallbeschreibung
 - Setup, Voraussetzungen
 - Eingaben und Sollausgaben
- **Tests nach Plan durchführen**
 - **Resultat:** Testberichte
- **Gefundene Fehler zusammenfassen**
 - **Resultat:** Fehlerberichte

Wichtig:
Tests dokumentieren

Resultate: Testplan bis Fehlerbehebungsberichte

Siehe auch: Zuser, Biffl, Grechenig, Köhle (2001): Software Engineering mit UML und dem Unified Process. Pearson Studium

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 142

Dokumentation der Tests umfasst sowohl deren Planung als auch die Resultate, wie die Folie zeigt.

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

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



Testplanung

Vorgehensweise

- Im Prinzip: Wer führt wann welche Testfälle durch
 - Testmethoden (Black-/White-Box-Test, Regressionstest usw.)
 - Konkrete Testfälle (Eingabe, Soll-Ausgabe)
 - Konfiguration der Testplattform (auch Testdaten in Datenbank)
 - Bereitstellung des Testgeschrirrs („Hilfsgerüst für die Testfälle“)
- Kriterien für den Start der Testdurchführung
 - Version für den Test freigegeben
- Kriterien für Testunterbrechung
 - Normalerweise wird bei Fehlern weitergetestet
 - Kriterien, wann das nicht mehr sinnvoll ist
- Kommunikationswege
 - Wer erfährt in welcher Form von Testergebnissen?
 - Was tun diese Rollen daraufhin (selbstständig); korrigieren sie?

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 143

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

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



Testfälle erstellen Prinzip

- **Black-Box-Test: Aus der Spezifikation**
 - zu allem, was in Spezifikation verlangt ist (*und nur dazu!*), muss geprüft werden, ob es wirklich so funktioniert
 - Spezifikation muss dazu die Soll-Werte vorgeben
 - Soll ist prinzipiell *nicht* aus dem Code ableitbar!
 - SW-Verhalten muss Spezifikation genügen
 - Was „intern“ passiert, ist uns egal
 - **Teststrategie**
 - Zu jeder Anforderung mindestens einen Testfall
 - Aber auch nicht viel mehr
 - Daher Testfälle geschickt wählen



Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 144

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

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



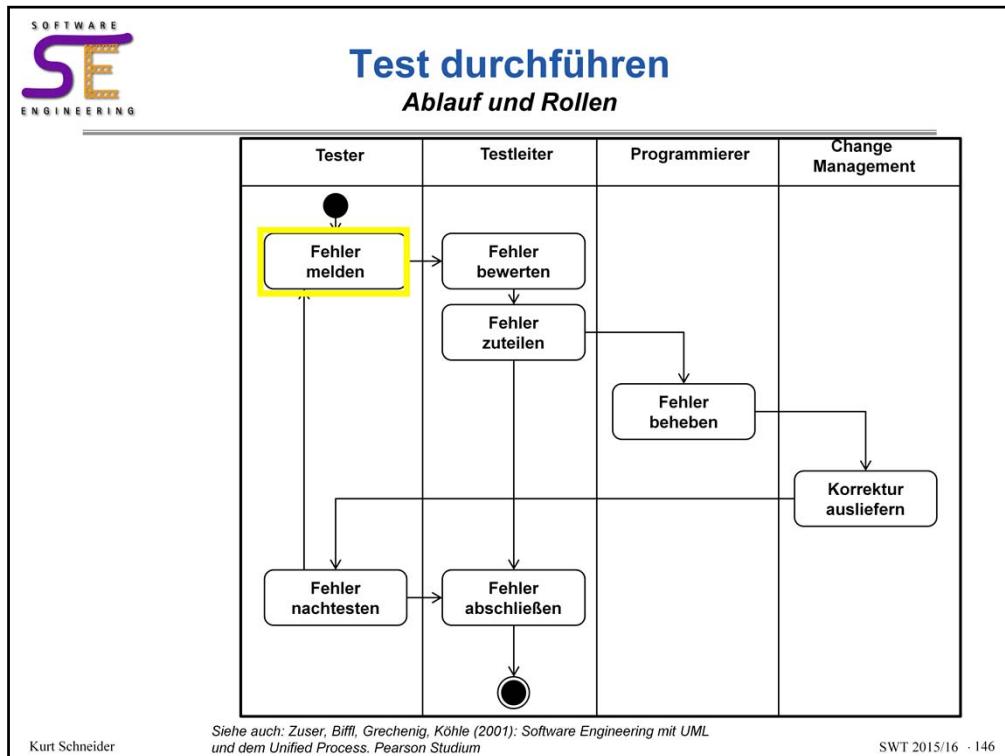
Testfälle praktisch ermitteln

1. Ausgangspunkt: Anforderung
 - Beispiel: „Der Eintrittspreis ist 3€, für Ermäßigte 2€“
2. Testfall: Anforderung anwenden, Sollwert ermitteln
 - Testfall 1: Knopf „Erwachsener“ drücken. Soll-Resultat: 3€
 - Testfall 2: Knopf „Ermäßigt“ drücken. Soll-Resultat: 2€
3. Mehr und kompliziertere Anforderungen: mehr Testfälle nötig
 - Unklarheiten ausräumen: quersum oder itQuer?
 - Kombinationen von Eingabeparametern
 - Sonderfälle: quersum (-12)? finCash (-1,28) ?
 - Was soll bei Falscheingaben geschehen?
 - (Vorher abgefangen, also unmöglich)
 - Warnung
 - Fehler
 - Raten, was gemeint war

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 145

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

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



Testen hat mehrere Rollen. Der Prozess beginnt mit „Fehler melden“ und endet mit „Fehler abschließen“.

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

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

Vor- und Nachteile von Tests	
Prinzip	
Vorteile	Nachteile
<ul style="list-style-type: none">• Reproduzierbar• Mehrfach nutzbar<ul style="list-style-type: none">– Rechenzeit ist billig• Umgebung wird mitgeprüft<ul style="list-style-type: none">– Bibliotheken– Virtuelle Maschine• Systemverhalten veranschaulicht• Falls automatisiert: schnell und billig	<ul style="list-style-type: none">• Bedeutung „fehlerloser“ Tests wird überschätzt („schlecht gesucht“)• Nicht alle Eigenschaften von Software sind testbar („Lesbarkeit“ ?)• Nicht alle Situationen reproduzierbar• Test zeigt Fehlerursache nicht• Falls <i>nicht</i> automatisiert: Aufwändig, insbesondere Wiederholung

Kurt Schneider

Leibniz Universität Hannover

SWT 2015/16 · 147

Die Frage ist ja nicht: Testen oder nicht?

Jeder professionelle Softwareentwickler testet selbstverständlich seine Programme. Aber wie intensiv und auf welche Eigenschaften hin? Die obige Aufstellung soll helfen, die Schwächen und Stärken von Tests bewußt einzusetzen.

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

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

Testebenen
von unten nach oben testen!

Systemtest

- Funktioniert das ganze Gerät?
- Incl. Funktions- und Leistungstest
- Wiederholung der Black Box Tests
- Regressionstest
- Ressourcen, Antwortzeit, Last, Durchsatz

Integrationstest:

- Funktionieren Module zusammen?
- Incl. Installationstest
- Lässt sich SW installieren?

Modul-/Unit Test:

- Funktioniert ein Modul?

Weitere Tests

- Benutzbarkeit, Sicherheit, Interoperabilität, Neustart nach Absturz

Kurt Schneider Leibniz Universität Hannover SWT 2015/16 · 148

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

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

Softwaretechnik

Kapitel 4



1. Wieso Software Engineering?
2. Vom Einzelkämpfer zum Großprojekt

Systematische Softwareentwicklung

3. Anforderungen und Test: Basis des Projekts
4. **Entwurf: Strukturen und nicht-funktionale Eigenschaften**
5. Entwürfe notieren mit UML: Modelle im SE
6. Design Patterns: Entwurfserfahrungen nutzen
7. Management: Technik und Projektmanagement

Leibniz Universität Hannover

SWT 2015/16 · 149