

Software Engineering

Requirements Engineering

Authors of slides:
Norbert Siegmund
Janet Siegmund
Oscar Nierstrasz
Sven Apel

- Können wir den Kurs auf 9:30 verschieben?

Schritt 1

Softwaretechnologie _WS19/20

Einzelne Seite

Einschreibung

Material


☒ Wählen Sie den Kurs 1

E-Mail

Performance results

Groups

Rights management



TU Chemnitz | Wintersemester 2019 / 2020

Softwaretechnologie _WS19/20

Tutor: [Shadi Saleh](#) | [Janet Siegmund](#)

Softwaretechnologie _WS19/20

[Display more information](#)

Einzelne Seite

Einzelne Seite

Willkommen im Kurs "Softwaretechnologie WS19/20"

Dies ist die Informationsseite zum Kurs "Softwaretechnologie WS19/20".
Klicken Sie auf "Bearbeiten", um diese Seite gemäß Ihren Vorstellungen anzupassen.

Schritt 2

Softwaretechnologie _WS19/20

Einzelne Seite

Einschreibung

Material

☒ Wählen Sie den Kurs

☐ E-Mail

Performance results

Groups

Rights management

▼ Wählen Sie den Kurs

Wählen Sie den Kurs, den Sie teilnehmen möchten (SE oder SE mit SE-P)

SE: Softwaretechnologie

SE mit SE-P: Softwarepraktikum und Softwaretechnologie

You are allowed to **Start test**, **See the assessment**, and **Get your results**.

▼ Assessment

Score: 0.0 / 1.0

Result:

Last result will be scored.

Show evaluation

Test with assessment:

All information about your submitted solution can be viewed by the supervisor of the course.

Start test

2

Schritt 3

The screenshot shows a quiz interface with a dark header bar at the top containing a 'Finish test' button. Below the header is a light gray bar with the text 'New question'. The main area contains two radio button options: 'SE mit SE-P: Softwarepraktikum und Softwaretechnologie' and 'SE: Softwaretechnologie'. A red box labeled '3' highlights the 'New question' button and the two options. At the bottom, a gray bar displays 'Question 1/1' and another 'Finish test' button, which is highlighted by a red box labeled '4'.

Finish test

New question

☐ SE mit SE-P: Softwarepraktikum und Softwaretechnologie

☐ SE: Softwaretechnologie

3

Question 1/1

Finish test

4


Poll - Industrievorlesung



- <https://terminplaner4.dfn.de/vpqVV5Sh7utMXoBF>



Votes 

January 2020

Wed 8Wed 15

Your name


(✓)


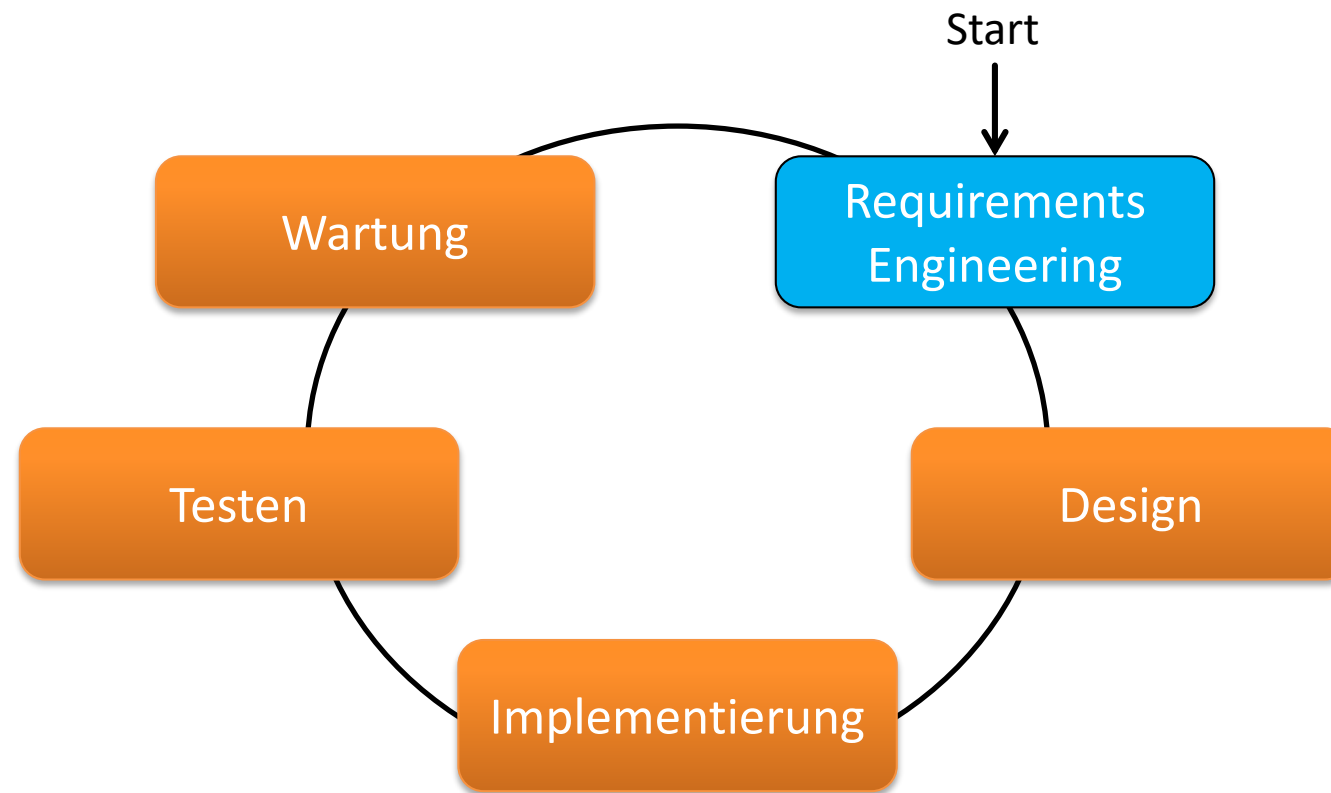

(✓)


Save

Die Deadline ist Montag, 28. Oktober 2019

- Was haben wir in der letzten Vorlesung besprochen?

Einordnung



Lernziele

- Notwendigkeit von Requirements Engineering verstehen
- Typische Probleme bei der Anforderungsanalyse kennen
- Vorgehen für systematisches Finden von Anforderungen verstehen
- Anforderungen beschreiben können



Warum Requirements Engineering?



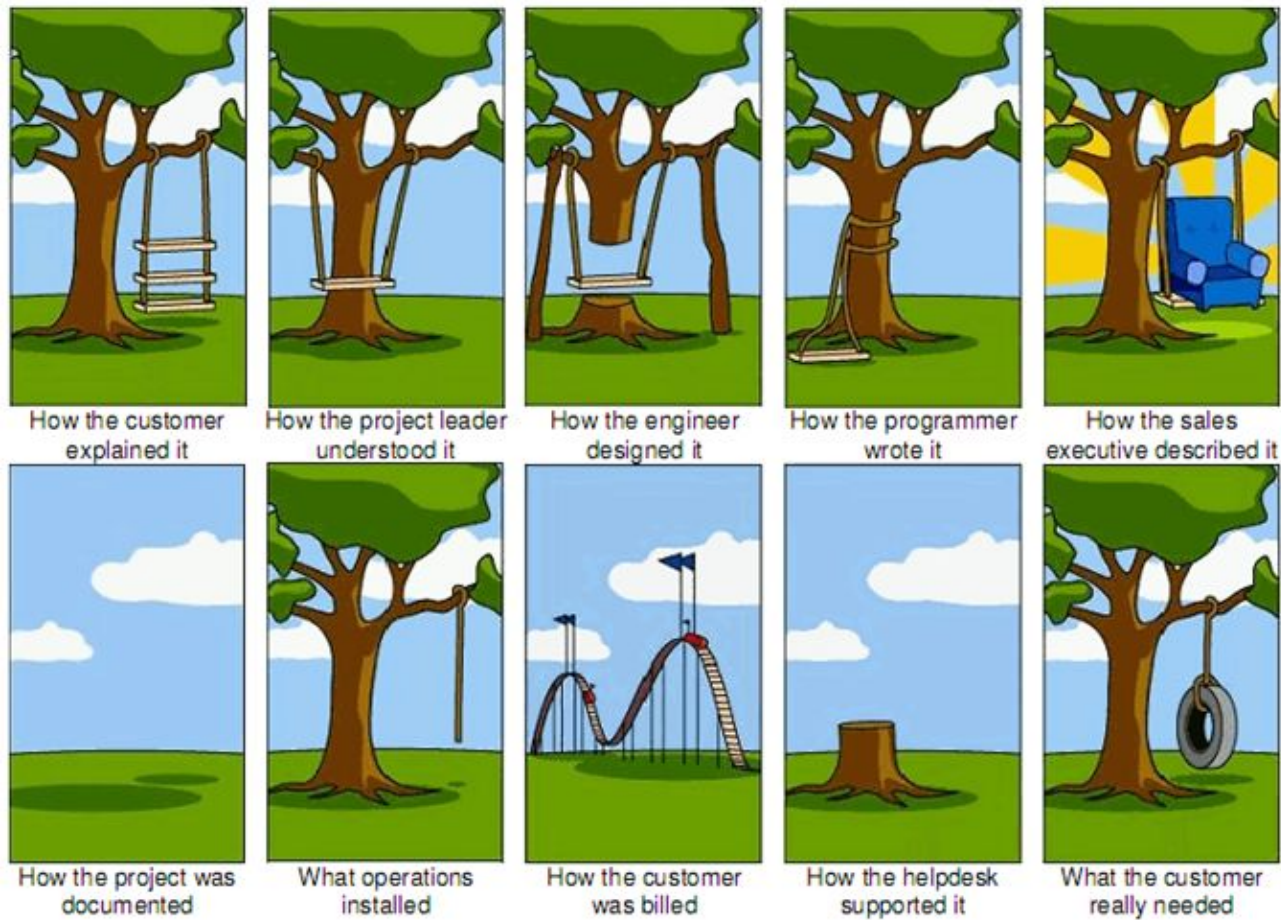
Was sind Requirements? I

- Bedingung oder Eigenschaft, die ein System benötigt,
 - um ein Problem zu lösen
oder
 - um ein Ziel zu erreichen
oder
 - um einem Vertrag, Standard oder Ähnlichem zu genügen
- [Pohl. Requirements Engineering]

Was sind Requirements? II

- Werden an ein Programm gestellt
- Spezifizieren:
 - Eigenschaften
 - Funktionalität
 - Einsatzsszenario
 - Qualität
- Werden von Stakeholdern bestimmt

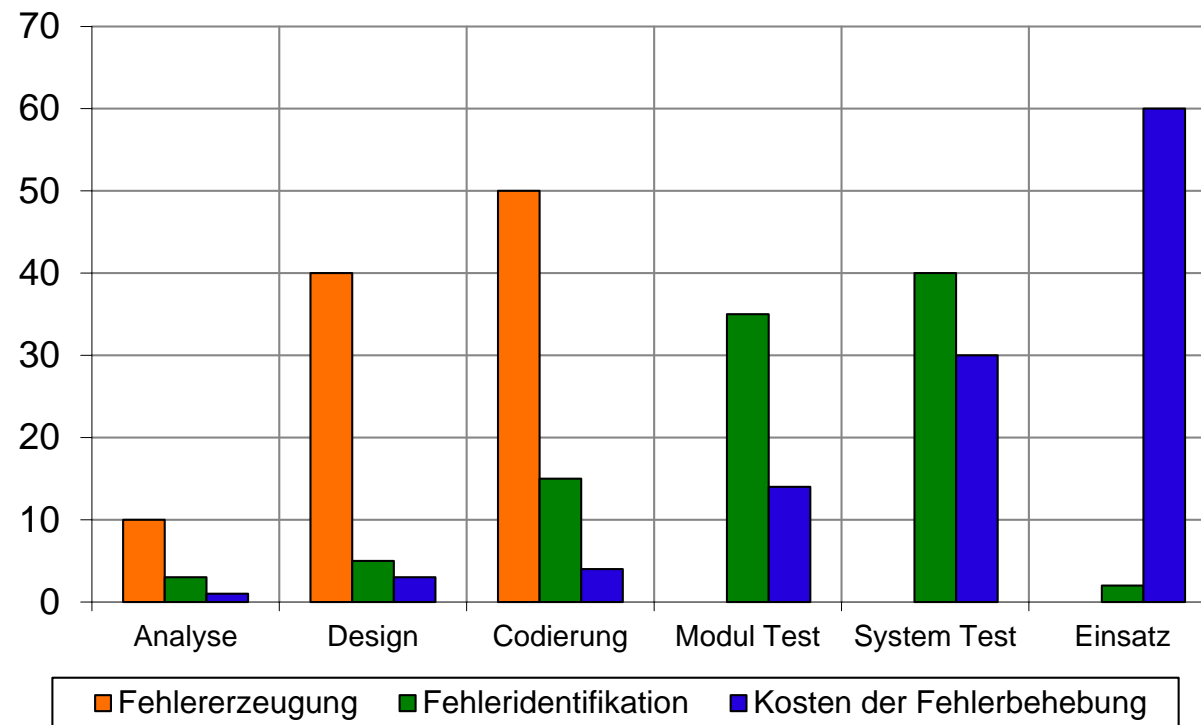
Warum Requirements Engineering? I



[Quelle: <http://www.interface-gmbh.de/RequirementsEngineering.htm>]

Warum Requirements Engineering? II

- Bauen wir das Richtige?



Überprüfbarkeit

- Szenario: Neue Firma, 3 Mitarbeiter (Gehalt 45.000 EUR)
- Bekommen Auftrag von Firma \$BigCooperation
 - Geschätzter Aufwand: 9 Bearbeiterjahre
 - Festpreis 500.000 EUR, 250.000 EUR sofort, Rest nach Abnahme
- Fertigstellung nach 3 Jahren
- Firma verweigert Abnahme, fordert Nacharbeiten

Probleme

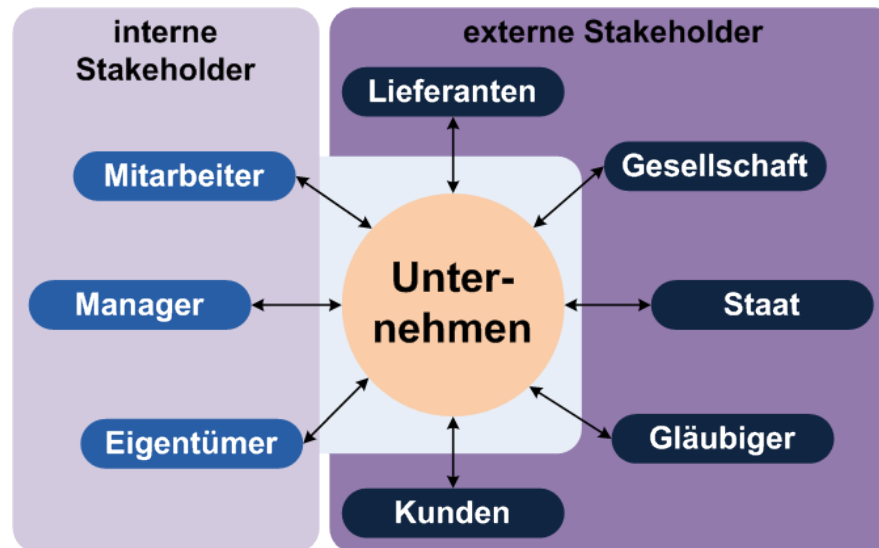


Probleme -- Kunden

- Kunden wissen nicht, was sie wirklich wollen
- Kunden benutzen ihre eigene Fachsprache
- Politische und organisatorische Faktoren können Anforderungen beeinflussen

Probleme -- Widersprüche

- Verschiedene Stakeholder können widersprüchliche Anforderungen haben
- Neue Stakeholder mischen sich ein



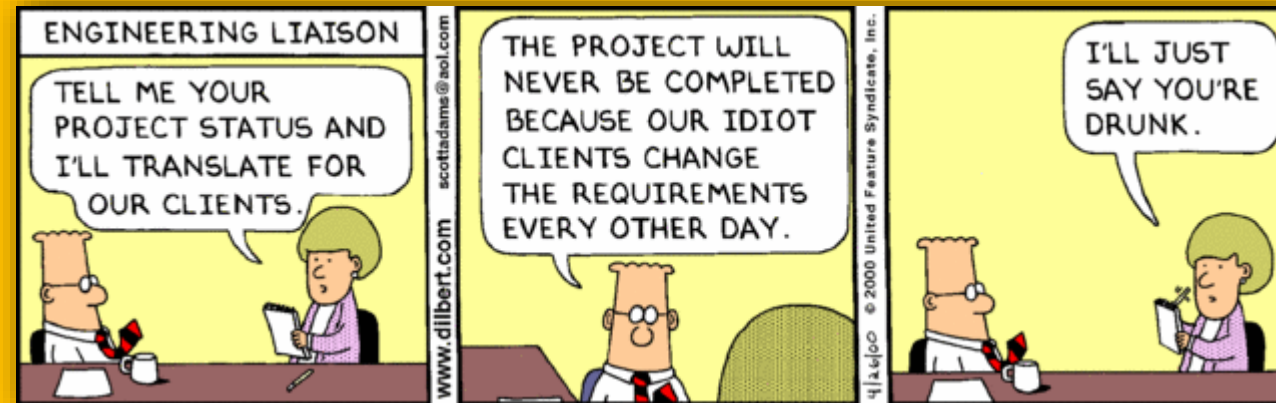
© Grochim
[<http://de.wikipedia.org/wiki/Stakeholder>]

Probleme -- Evolution

- “Requirements *always evolve*”
 - Durch besseres Verständnis darüber, was der Kunde wirklich braucht
 - Durch Änderungen in den Zielen der Organisation
- Daher wichtig: “*plan for change*” in den Requirements



Wie funktioniert Requirements Engineering?



4 Schritte

1. Anforderungsermittlung
 - Sammeln von Anforderungen
 - z.B. durch Anwendergespräche, Dokumenten-Studium
2. Anforderungsanalyse
 - Klassifizierung, Bewertung, Vergleich und Prüfung
 - z.B. Kosten-/Nutzen-Aspekte, Konsistenz, Vollständigkeit
3. Anforderungsbeschreibung
 - Beschreibung in einheitlicher Form (z.B. als Anwendungsfälle)
4. Anforderungsrevision
 - Erneute Prüfung/Änderung von Anforderungen
 - ggf. nur in formellem Änderungsverfahren

1. Anforderungsermittlung

- Stakeholder-basiert:
 - Identifikation von Stakeholdern
 - Gespräche mit allen Stakeholdern
- Szenario-basiert:
 - Szenario: konkreter, fiktiver (Arbeits-) Ablauf eines Systems
 - Erstellen von allen relevanten Szenarien

Beispiel: NoMoreWaiting (NMW)

Die Studierenden der TU Chemnitz sind mit dem Problem konfrontiert, dass sie zur Mittagszeit keinen Tisch in der Mensa finden. Dies liegt daran, dass zu bestimmten Stoßzeiten besonders viele Studierende Essen gehen, die Mensa jedoch nur eine beschränkte Zahl von Sitzplätzen aufweist.

Findige Studierende der Lehrveranstaltung „Software Engineering“ möchten aus diesem Grund eine App entwickeln, mit der Studierende einen Sitzplatz zu einer bestimmten Zeit für sich reservieren können. Damit dies auch in der Praxis funktioniert, werden in der Mensa Tische ausgewiesen, die ausschließlich von Nutzern der Mensa-App genutzt werden dürfen. Die Anzahl der ausgewiesenen Tische kann vom Mensapersonal entsprechend der Reservierungen vorab angepasst werden.

Die App kann kostenlos aus dem Market heruntergeladen werden. Für die Nutzung der App ist es erforderlich, dass sich jeder Nutzer einen Account mit seiner Matrikelnummer anlegt. Ob die Matrikelnummer auch wirklich existiert, wird über das zentrale Hochschulverwaltungssystem der Universität geprüft.

Sobald die Registrierung abgeschlossen wurde, können Nutzer damit beginnen, Plätze zu reservieren. Plätze können am Tag der Nutzung ab 09:00 reserviert werden. Dabei kann der Nutzer über eine Karte, die die Anordnung der Tische in der Mensa zeigt, wählen, welchen Platz er gern hätte. Reservierungen sind immer nur für 20 Minuten gültig und sind aufgeteilt in Slots. In einer Stunde kann ein Platz also für drei Slots vergeben werden.

In der Mensa müssen Nutzer auf ihrem Platz „einchecken“. Dafür scannen sie mit ihrem Smartphone einen am Platz fixierten QR-Code. Über das Einchecken werden Statistikdaten gesammelt. So wird für jeden Nutzer ermittelt, wie oft er einen Platz reserviert, diesen dann aber nicht in Anspruch nimmt. Nutzer, die reservierte Plätze häufig nicht in Anspruch nehmen, werden über ein Credit-System bestraft. Entsprechend der Credit-Zahl der Nutzer müssen diejenigen Nutzer mit wenigen Credits bei der Reservierung einen oder gar mehrere Tage aussetzen.

Stakeholder für NMW

- Studierende, die App benutzen
- Studierende, die App nicht benutzen
- Mensa-Angestellte
- Datenschutzbeauftragte
- Mitarbeiter
- Professoren

Szenarien für NMW

- "Ich mache mich JETZT mit meinen X Freunden auf den Weg. Das System soll einfach und schnell Plätze finden und reservieren."
- „Es soll möglich sein, Plätze sehr einfach wieder freizugeben.“

Aufgabe

- Welches sind die wichtigsten Anforderungen für NoMoreWaiting?
- Einfach
- Usability
- Mehrsprachigkeit
- Jeder Platz darf nur einmal vergeben werden
- Verschiedene OS
- Geringer Datenverbrauch
- Concurrency
- Serverstabilität
- Kosten und Nutzenverhältnis
- Datenschutz (MatNummer)
- Gründe für Nichtnutzung der Reservierung
- Zusätzliche Authentifizierung neben Matrikelnummer (zusätzlicher Schutz)
- Effiziente Verteilung der Plätze
- Zusammenspiel zwischen Scanner und App
- Gruppenfunktion
- Freie Slots danach, damit mehr Zeit bleibt
- Nutzer kann nicht alle Plätze für einen Tag reservieren
- Statusmeldung (z.B. Server geht nicht)



Stakeholder- oder szenariobasiert?

- Stakeholder-basiert:
 - Vorteil: Anforderungen werden nach Personen gebündelt aufgenommen
 - Nachteil: Nutzen/Sinn/Ziel des Systems tritt in den Hintergrund
- Szenario-basiert:
 - Vorteil: Anforderungen orientieren sich an den „normalen“ Abläufen des Systems
 - Nachteil: Seltene Abläufe oder indirekt betroffene Institutionen werden leichter vergessen

Use Cases und Szenarien

- Ein use case ist die *Spezifikation* von einer *Sequenz von Aktionen*, einschließlich *Varianten*, dass ein System, *interagierend mit Aktoren* des Systems, ausführen kann”.
 - Beispiel: kaufen einer DVD im Internet
- Ein Szenario ist ein *bestimmter Pfad von auftretenden Aktionen*, startend von einem bekannten, initialen Zustand.
 - Beispiel: Verbinde zu myDVD.com, Gehe zur “search” page; Gebe ein “...”, ...

Unified Modeling Language

UML ist der Industriestandard für die Dokumentierung objekt-orientierter Modelle

<i>Class Diagrams</i>	visualize <i>logical structure</i> of system in terms of <i>classes, objects, and relationships</i>
<i>Use Case Diagrams</i>	show external <i>actors and use cases</i> they participate in
<i>Sequence Diagrams</i>	visualize <i>temporal message ordering</i> of a <i>concrete scenario</i> of a use case
<i>Collaboration (Communication) Diagrams</i>	visualize <i>relationships</i> of objects exchanging messages in a <i>concrete scenario</i>
<i>State Diagrams</i>	specify the <i>abstract states</i> of an object and the <i>transitions</i> between the states

Use Case Diagramme

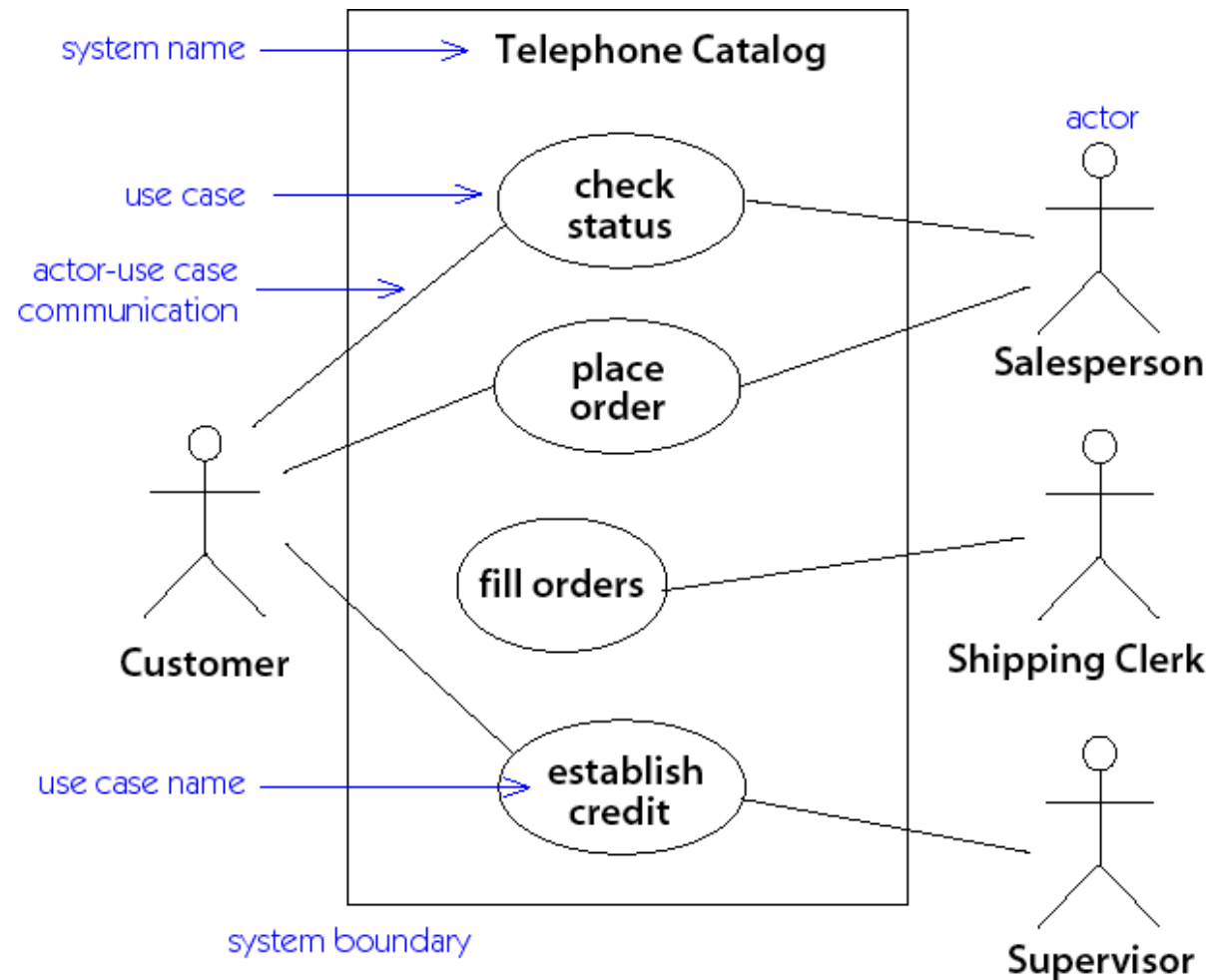


Figure 5-1. Use case diagram

Sequenzdiagramme

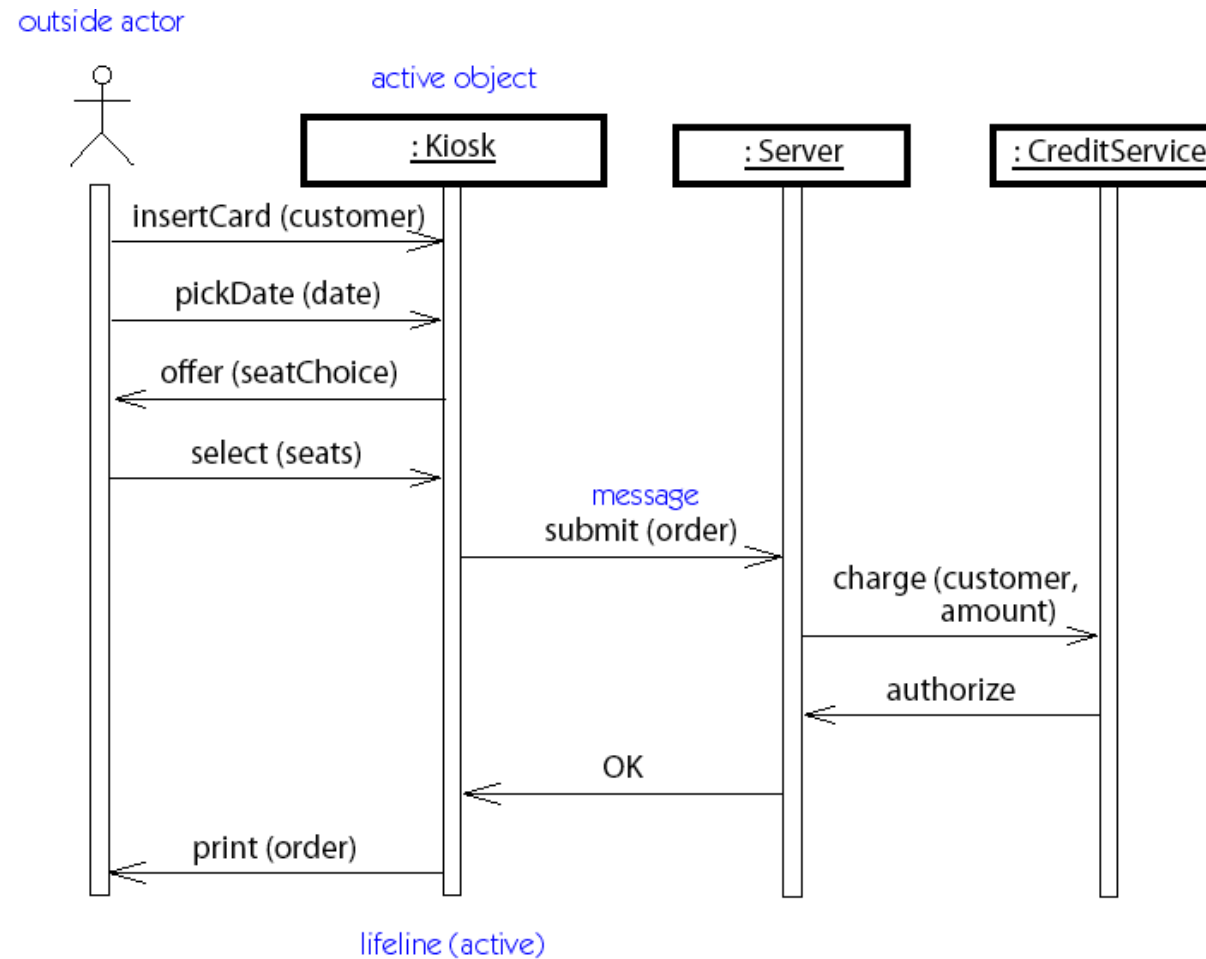


Figure 8-1. Sequence diagram

2. Anforderungsanalyse

- Funktionale Anforderungen:
 - **Was** soll das System leisten?
 - Welche Dienste soll es anbieten?
 - Eingaben, Verarbeitungen, Ausgaben
 - Verhalten in bestimmten Situationen, ggf. was soll es explizit nicht tun
- Nicht-funktionale Anforderungen:
 - **Wie** soll das System/individuelle Funktionen arbeiten?
 - Qualitätsanforderungen wie Performance und Zuverlässigkeit
 - Anforderungen an die Benutzbarkeit des Systems

Anforderungen für NoMoreWaiting

- Stakeholder-basiert

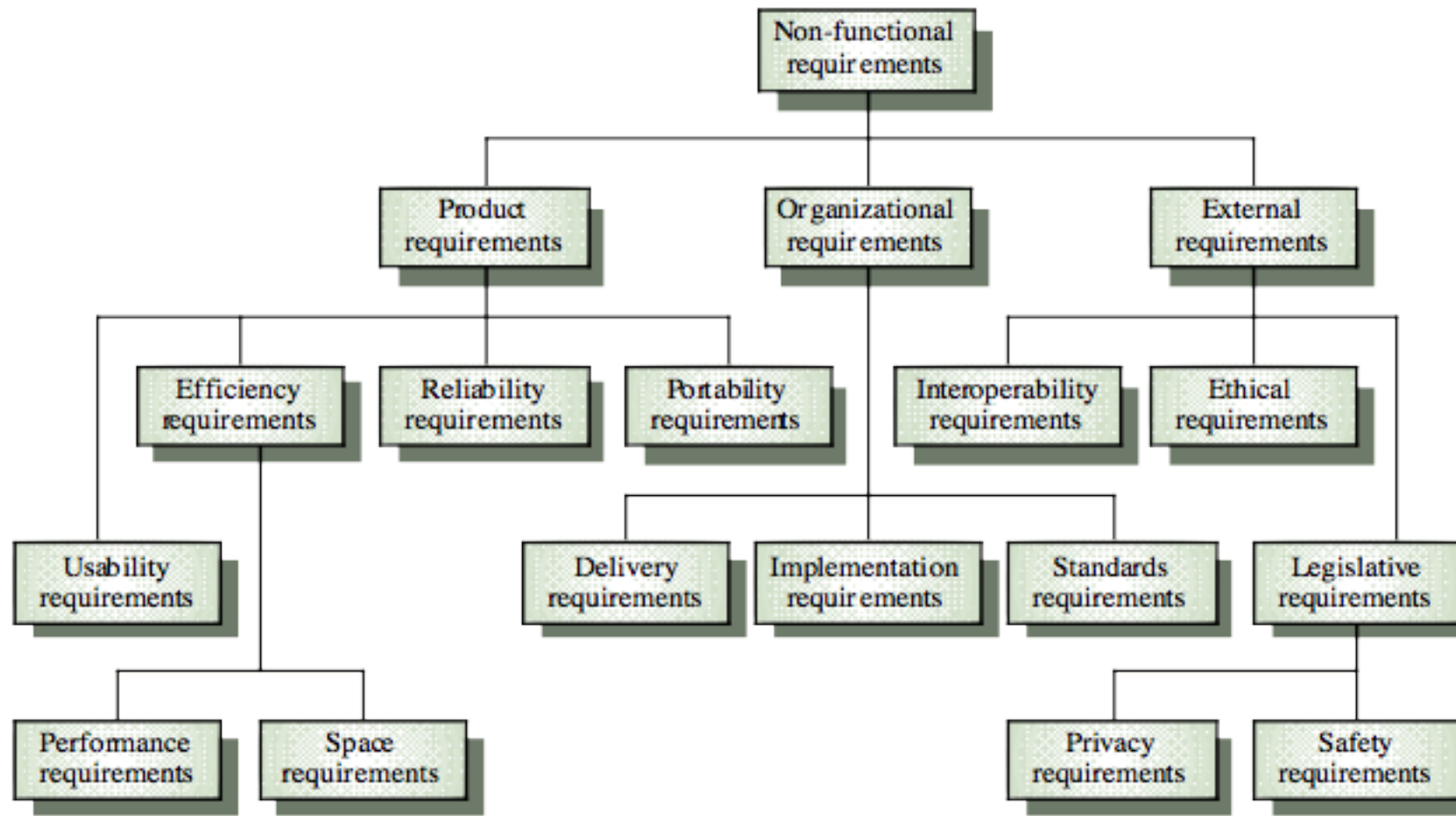
- Schnell
- Einfach
- Übersichtlich
- Schnelle Abmeldung
- Rückmeldung über Credits und Reservierungsstatus
- Mitarbeiter der Mensa sollen es einfach handeln können
- Nichtbenutzer sollen nicht eingeschränkt werden
- Mehrfachbuchen soll nicht möglich sein
- Datenschutz (anonym)
- Student soll Platz einfordern können, wenn andere reservierten Platz benutzen
- Gruppenreservierung

- Szenario-basiert

- Reservieren:
 - Kostenlos
- Freigeben
 - System muss wissen, dass Platz frei ist
- Registrierung
 - Ohne Matrikelnummer?
 - Fake-Reservierung?
 - Schnittstelle zur Univerwaltung zum Validieren von MatNr.
- Sammeln
 - Verfälschung von Statistken bei Gruppenbenutzung verhindern
- Feedback
 - Andere müssen Gruppenreservierung annehmen
- Verschlüsselung der Daten
- Max. Platzbeschränkung bei Reservierung



Arten von Nicht-Funkt. Anforderungen



2. Anforderungsanalyse

- Stakeholder- oder szenariobasiert?
 - Funktionale Anforderungen: Szenario-basiert
 - Nicht-funktionale Anforderungen: Stakeholder-basiert
- Widersprechen sich Anforderungen?
- Sind Anforderungen konsistent?
- Sind Anforderungen umsetzbar?

Erfüllbarkeit von Anforderungen

- Generell: Anforderungen sollten so formuliert werden, dass sie objektiv verifiziert werden können
- Unpräzise (Negativbeispiel):
 - Das System sollte von erfahrenen Kontrolleuren *einfach zu benutzen* sein und sollte so organisiert sein, dass *Benutzerfehler minimiert* werden.
 - Ausdrücke wie „einfach zu benutzen“ sind sinnlos
- Verifizierbar (Positivbeispiel):
 - Erfahrene Kontrolleure sollten in der Lage sein alle Funktionen des Systems *nach einem 2-stündigen Training* nutzen zu können. Die *mittl. Anzahl* von getätigten Fehlern sollte nach dem Training *nicht höher als 2 pro Tag* sein.

Präzise Metriken

<i>Eigenschaft</i>	<i>Metrik</i>
<i>Performanz</i>	Processed transactions/second User/Event response time Screen refresh time
<i>Größe</i>	K Bytes; Number of RAM chips
<i>Handhabbarkeit</i>	Training time Rate of errors made by trained users Number of help frames
<i>Zuverlässigkeit</i>	Mean time to failure Probability of unavailability Rate of failure occurrence
<i>Robustheit</i>	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
<i>Portabilität</i>	Percentage of target dependent statements Number of target systems

Beispiele

- Das System soll zügig reagieren
 - 80 % aller Anfragen sollen unter 0.1 Sekunde beantwortet werden, 99.99 % aller Anfragen unter 2 Sekunden
 - Test-Hardware spezifizieren!
- Das System soll auch im Mehrbenutzerbetrieb schnell reagieren
 - Auf XY-Server Verarbeitung von 200 Anfragen pro Sekunde von 10 unterschiedlichen Systemen mit 100mbit Ethernet-Anbindung
- Das System soll stabil und ausfallsicher sein
 - Verfügbarkeit von 99.99 %, Neustart innerhalb von 30 Sekunden, das System darf nicht aufgrund falscher Eingaben abstürzen

Aufgabe

- Wie kann folgende Requirements spezifizieren?

Stakeholder-basiert

- Schnell
- Einfach
- Übersichtlich
- Schnelle Abmeldung
- Rückmeldung über Credits und Reservierungsstatus
- Mitarbeiter der Mensa sollen es einfach handeln können
- Nichtbenutzer sollen nicht eingeschränkt werden
- Mehrfachbuchen soll nicht möglich sein
- Datenschutz (anonym)
- Student soll Platz einfordern können, wenn andere reservierten Platz benutzen
- Gruppenreservierung

Szenario-basiert

- Reservieren:
 - Kostenlos
- Freigeben
 - System muss wissen, dass Platz frei ist
- Registrierung
 - Ohne Matrikelnummer?
 - Fake-Reservierung?
 - Schnittstelle zur Univerwaltung zum Validieren von MatNr.
- Sammeln
 - Verfälschung von Statistken bei Gruppenbenutzung verhindern
- Feedback
 - Andere müssen Gruppenreservierung annehmen
- Verschlüsselung der Daten
- Max. Platzbeschränkung bei Reservierung



3. Anforderungsbeschreibung

- Anforderungen müssen systematisch und einheitlich beschrieben werden
- Beispiele:
 - Volere-Template (Snow card)
 - IEEE Std 830-1998
 - FURPS/FURPS+

Volere

- 5 Hauptkategorien, in denen Anforderungen genau beschrieben werden
 - Project Drivers, Project Constraints, Functional Requirements, Non-Functional Requirements, Project Issues
- Eingeteilt in Subkategorien
- <https://www.volere.org/templates/volere-requirements-specification-template/>

Volere: Project Drivers

- Warum machen wir das Projekt?
 - Zweck des Projekts
 - Stakeholder

Volere: Project Constraints

- Welchen (gesetzlichen) Rahmenbedingungen müssen eingehalten werden?
 - Einschränkungen
 - Namenskonventionen und Terminologie
 - Relevante Fakten und Annahmen

Volere: Functional Requirements

- Was ist der Sinn des Systems?
 - Rahmen der Arbeit
 - Datenmodell und Data-dictionary
 - Rahmen des Produkts
 - Funktionelle Anforderungen und Anforderungen an Daten

Volere: Non-functional Requirements

- Was sind (selbstverständliche) Erwartungen an das System?
 - Look and feel
 - Usability and humanity
 - Performance
 - Wartbarkeit- und Support
 - Sicherheit
 - Kulturell und politisch
 - Gesetzliche

Volere: Project Issues

- Sonstige Eigenschaften:
 - Offene Probleme
 - Off-the-Shelf Lösungen
 - Neue Probleme
 - Aufgaben
 - Migration auf neues Produkt
 - Risiken
 - Kosten
 - Nutzerdokumentation und –training
 - Waiting room
 - Ideen für Lösungen

Model: Snow Card

Req-ID:	Eindeutige ID	Req-Type:	Kategorie	Events/UCs:	Use cases/events, die diese Anforderung benötigen
Description:	Informelle Beschreibung				
Rationale:	Begründung, warum diese Anforderung wichtig ist				
Originator:	Stakeholder, der die Anforderung stellt				
Fit Criterion:	Wie kann man die Erfüllung der Anforderung messen/testen?				
	Wie wichtig bzw. wie kritisch ist die Anforderung				
Customer Satisfaction:		Customer Dissatisfaction:		Priority:	
Supporting Material:	Verweise auf Dokumente, die diese Anforderung ausführlich beschreiben		Conflicts: In Konflikt/Konkurrenz stehende Anforderungen		
History:	Wann erstellt, welche Änderungen, letzte Bearbeiter,...				

Bewertungskriterien

- Welche Kriterien sollten gute Anforderungen erfüllen?
- Sind die gefundenen Anforderungen gute Anforderungen?



Bewertungskriterien

- Korrekt
- Eindeutig
- Vollständig
- Konsistent
- Gewichtet nach Wichtigkeit und Stabilität
- Überprüfbar
- Modifizierbar
- Nachvollziehbar
- Quelle: IEEE Std 830-1998

In der Praxis

- Anforderungsdefinition bestehen gewöhnlich aus *natürlicher Sprache* mit zusätzlichen *Diagrammen und Tabellen* (z.B. UML)
- 3 Arten von Probleme:
 - **Lack of clarity:** Schwierig *präzise und gleichzeitig leicht-verständliche* Dokumente zu schreiben
 - **Requirements confusion:** *Funktionale und nicht-funktionale* Anforderungen sind oft *vermischt*
 - **Requirements amalgamation:** *Mehrere unterschiedliche* Anforderungen werden *zusammen ausgedrückt*.

4. Anforderungsrevision

- Erneute Prüfung und ggf. Anpassung der Anforderungen

Validität	Does the system provide the functions <i>which best support</i> the customer's needs?
Konsistenz	Are there any <i>requirements conflicts</i> ?
Vollständigkeit	Are <i>all functions</i> required by the customer included?
Realisierbarkeit	Can the requirements be implemented given <i>available budget and technology</i> ?

Checkliste kann helfen

http://www.wis.win.tue.nl/2M390/rev_req.html

- Hat die Software **prägnanten Namen** und **klar beschriebenen Zweck**?
- Sind die Eigenschaften der Nutzer und **typischen Szenarien** beschrieben?
- (Keine Nutzer-Kategorie fehlt)
- Sind alle **externen Interfaces** der Software explizit erwähnt?
- (Keine Interfaces fehlen)
- Hat jede Anforderung einen **eindeutigen Namen**?
- Ist jeder Anforderung **atomar** und **einfach formuliert**?
- (Typischerweise ein Satz. Zusammengesetzte Anforderungen müssen geteilt werden)
- Sind die Anforderungen in **zusammenhängenden Gruppen** organisiert?
- (Wenn notwendig, hierarchisch; nicht mehr als ca. 10 pro Gruppe)
- Hat jede Anforderung eine **Priorität**?
- (Ist die Bedeutung der Levels klar?)
- Sind alle **instabilen Anforderungen** entsprechend markiert?
(TBC=`To Be Confirmed', TBD=`To Be Defined')

Zusammenfassung

- Notwendigkeit von Requirements Engineering verstehen
- Typische Probleme bei der Anforderungsanalyse kennen
- Vorgehen für systematisches Finden von Anforderungen verstehen
- Anforderungen beschreiben können



Mögliche Klausurfragen

- Warum brauchen wir Requirements Engineering?
- [Beschreibung von Anwendung X]
 - Nennen Sie X Stakeholder. Erklären Sie Ihre Auswahl.
 - Beschreiben Sie X Szenarien. Erklären Sie Ihre Auswahl.
 - Nennen und beschreiben Sie X funktionale und X nicht-funktionale Eigenschaften. Erklären Sie Ihre Entscheidung.
 - Sind Ihre Anforderungen gute Anforderungen? Warum?
- Würden Sie den stakeholderbasierten oder szenariobasierten Ansatz zum systematischen Finden von Requirements empfehlen?

Literatur

- Pohl. Requirements Engineering: Grundlagen, Prinzipien, Techniken. 2008
- Sommerville. Software Engineering. Kapitel 6-10.