

Zusammenfassung Requirements Engineering

Philipp Jäcks

11. Januar 2016

Inhaltsverzeichnis

1	Einleitung	2
1.1	non-functional R	2
1.2	functional R	2
1.3	Requirements Specification (RS)	3
1.4	Software Requirements Specification (SRS)	3
1.5	Main Activities in RE	3
2	Use Case Analysis	5
2.1	Glossar	7
3	Requirements Elicitation - "Anforderungserhebung"	9
3.1	Techniken und Ansätze	10
3.1.1	Task Analysis and Domain Analysis	10
3.1.2	Repertory Grids	10

1 Einleitung

Requirements sind nicht gegeben, sondern müssen erarbeitet werden und im Laufe des Entwicklungsprozess verfeinert und verbessert werden. Interdisziplinäre Disziplin mit Menschenkontakt. Abbildung der informalen Welt der Stakeholder auf die formale Welt der Entwickler/Verhalten der Software.

1.1 non-functional R

- Bedingungen die an das System als Ganzes gestellt werden, z.B. Zeitbeschränkungen, Performance, Standards, den Entwicklungsprozess
- **Aufteilung in**
 - *Product* Effizienz (Performance, Speicherbedarf), Usability, Zuverlässigkeit
 - *Organisational* Standards, Entwicklungsprozess
 - *External* Ethische, Gesetzliche, Interoperabilität
- oftmals kritischer als functional, da User bei Fehlverhalten eines Dienstes (=functional R) einen Weg drum herum findet. Bei fehlerhafter NFR ist dies nicht so leicht möglich.
- quantisierbar spezifizieren um objektiv testen zu können. Dazu folgende Kriterien

Eigenschaft	Messeinheit
Speed	Operationen/Sekunde; Reaktionszeit auf Usereingabe
Ease of Use	Einarbeitungszeit; Hilfsfenster
Zuverlässigkeit	Durchschnittszeit bis Fehler eintritt
Robustheit	Was passiert mit Daten nach Fehler? Wie schnell lässt sich das System nach Fehler neu

1.2 functional R

- Aussagen über Dienste die das System liefern muss, speziell konkrete Features, wie reagiert das System auf Eingaben, bei bestimmten Situationen.
- Manchmal Aussage darüber was das System in Situation grade *nicht* machen soll.
- **Aspekte:**
 - *Data*: Struktur, Verwaltung, Zugriff, Übertragung
 - *Functions*: Input, Output, Verarbeitung
 - *Verhalten*: beobachtbares Verhalten des Systems (auch in Fehlersituationen)

Aufteilung der Anforderungsanalyse in RS (vglbar mit Lastenheft) und SRS (vglbar mit Pflichtenheft).

1.3 Requirements Specification (RS)

Product Constraints bestimmen

1. Zweck des Produkts
Aufgabe, Ziel, Was soll das Produkt bringen? Wichtige Punkte
 - Welchen Vorteil liefert das Produkt?
 - Ist der Vorteil messbar?
 - Ist der Aufwand für den Vorteil rechtfertigbar?
2. Client, Kunde, Stakeholder
3. Nutzer des Produkts
4. Requirements Bedingungen
5. Definitionen, Nomenklatur
6. relevante Fakten
7. Annahmen

Functional Req. bestimmen

- Scope des Produkts
The features and functions that characterize a product, service, or result.
- Functional and Data Req
Spezi der Dienste und Datenstruktur

Non-Functional Req. bestimmen

1.4 Software Requirements Specification (SRS)

Eigenschaften einer guten SRS:

- Korrekt, Komplett, Konsistent, Verifizierbar...

1.5 Main Activities in RE

- Erhebung der R. (informal)
Welche Probleme bewältigen? Wer ist Kunde, Stakeholder, Nutzer? Techniken: Interviews, Brainstorming, Prototyping
- Model and Analysing der R -> Ergebnis: Formale R
- Kommunikation der R
Formale R müssen auch für Stakeholder verständlich sein

- Abgleich/Verifikation der formalen R mit Vorstellungen der Stakeholder
- Weiterentwicklung der R
Vorstellung des Softwaresystems kann Stakeholder/Nutzer/Kunde dazu veranlassen
R umzuformulieren

2 Use Case Analysis

Ziel: Abbildung der Realität in Anwendungsfalldiagramm. Dabei wird in der Regel ein konkretes Szenario beschrieben.

- Use Case = Menge von Aktionssequenzen, die ein Nutzer oder externes System ausführt um ein bestimmtes Ziel zu erreichen. Beschreibt inhaltlich was beim Versuch der Zielerreichung passieren und schiefgehen kann.
- Helfen in der **Planungsphase**:
 - verdeutlichen wichtige Ziele des Produkts
- in der **Entwicklungsphase**
 - fassen Kernpunkte eines Features für Entwickler gut zusammen
- Erfassen der funktionalen R aus Sicht des Nutzers

Actor:

- interagiert mit dem System; entweder Person oder wieder ein System
- gibt Input und erhält Output vom System
- extern; keine Kontrolle über den Use Case
- Vererbung möglich
- **Primär** und **Sekundär** Actor
 - Primär: will ein Ziel mithilfe des Systems erreichen
 - Sekundär: wird vom System benötigt um das Ziel des Primären zu erreichen

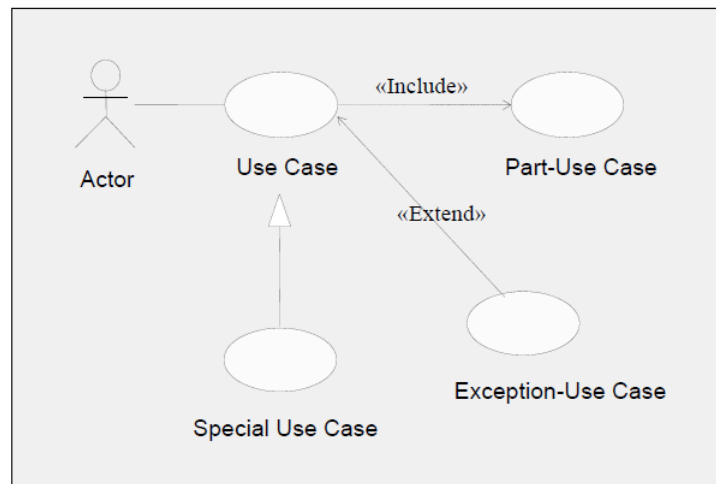


Abbildung 1: Elements of Use Case

Use Case begleitet die ganze Entwicklung. Wird *realisiert* vom Design Model, *Implementiert* vom Implementation Model und *Verifiziert* vom Test Model.

Flow of Events

- **Happy Path** normal basic Flow
- alternative flows
 - exception flows for error situations
 - reguläre Alternativen ein Ziel zu erreichen
 - odd cases

Scenario = Sequenz von Aktionen die die Interaktion zwischen Actor und System beschreibt (quasi ein Element des Use Case)

Use Case Reports

Beschreibt den Use Case im Detail. Einige Bestandteile nach Cockburn

- Ziel
- Context of Use: In welchem technischen Kontext tritt der Use Case auf?
- Level: Level of Detail, Summary, Primary Task etc
- Folgende sind optional
- Actor
- Stakeholder
- Conditions
- Trigger

Report ist wichtiger als Use Case Model (\leq Das Diagramm).

Dazu gehören noch zusätzliche Spezi (mit Usability, Zuverlässigkeit Performance etc pp, quasi NFR zum Use Case) und Glossar -> Begriffsklärung.

Use Case Analysis

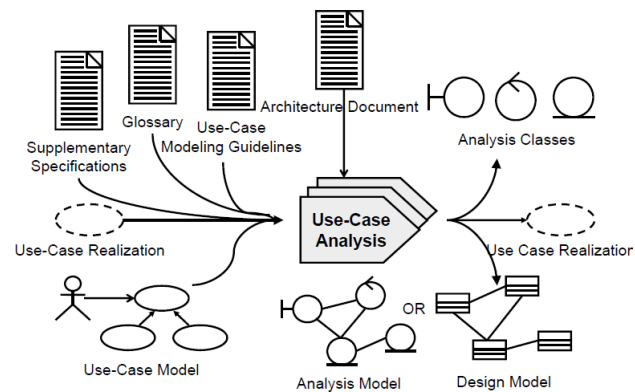


Abbildung 2: Use Case Analysis

Summary

Vorteile	Nachteile
zeigen functional R in einfacher, verständlicher Art	zeigen ausschließlich functional R
Erstellen Framework ¹ für non-functional R und Projekt Details.	

2.1 Glossar

- **Sequenzdiagramm** = Graphische Darstellung eines Szenarios, das Interaktion zw. Objekten zeitlich darstellt

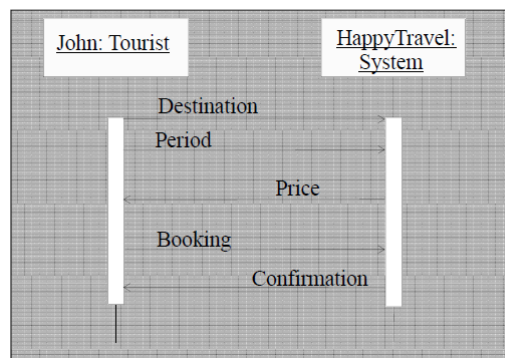


Abbildung 3: Ex Sequence Diagram

- **Collaboration Diagram** = Interaktionsdiagramm - zeigt eine Sequenz von Nachrichten, die eine Operation oder Transaktion implementieren

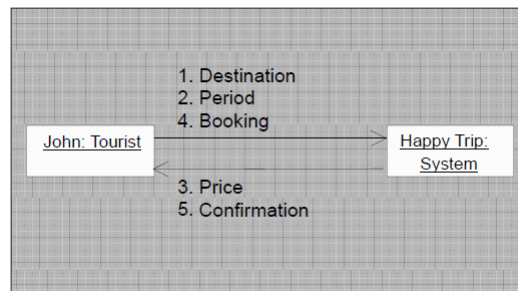


Abbildung 4: Ex Collaboration Diagram

3 Requirements Elicitation - "Anforderungserhebung"

- Prozess des *Suchens, Aufdeckens, Erwerbens, Ausarbeiten* von R für computerbasierte Systeme
- Teil der frühen Entwicklungsphase, aber fortlaufend und kritisch
- komplex, verschiedenste Techniken - vor allem aus Sozialwissenschaften - finden Anwendung
 - großer Problemfaktor liegt in Kommunikation zw. Engineers und Kunde
 - Iterative Erarbeitung, setzt stark auf Kommunikationskills der Engineers und Bereitschaft der Stakeholder
 - Problem: Konzepte, die für A verständlich sind, könnten für B komplett unverständlich sein
- Probleme der Erhebung
 - Scope: Systemgrenzen sind falsch definiert; unnötige technische Details spezifiziert, die eher verwirren als helfen
 - Understanding: Kunde weiß nicht genau was er will; was seine Computer-Umgebung her gibt; versteht das Problem nicht ganz; Lässt Infos aus die "offensichtlich" sind
 - Flüchtigkeit: R verändern sich mit der Zeit

5 Fundamentale Aktivitäten zur Erhebung

1. Verstehen der Anwendungsdomäne
Beschreibung existierender Arbeitsprozesse und die zugehörigen Probleme, die vom System gelöst werden sollen.
2. Quellen der R identifizieren
verschiedenste Quellen -> Stakeholder (meistens); Existierende Systeme und Prozesse; exist. Doku über aktuelles System und Prozesse
3. Analyse Stakeholder
Identifizieren wer der "offensichtlichste" Stakeholder ist -> mal der Endkunde, mal der Geldgeber etc
4. Methoden, Ansätze, Tools auswählen
Gründe für Methodenauswahl: einzige, die der Analyst kennt; sein Favorit; vorge-schrieben;...
5. R erheben von Stakeholder und anderen Quellen
Ergebnis = detaillierte Menge von R in natürlicher Sprache und einfachen Dia-grammen

Wir benötigen unterschiedliche Techniken, da jede Technik anderes Wissen als Ergebnis liefert.

	Interviews	Domain	Groupwork	Ethnography	Prototyping	Goals	Scenarios	Viewpoints
Understanding the domain	x	x	x	x		x	x	x
Identifying sources of requirements	x	x	x			x	x	x
Analyzing the Stakeholders	x	x	x	x	x	x	x	x
Selecting techniques and approaches	x	x	x					
Eliciting the Requirements	x	x	x	x	x	x	x	x

Abbildung 5: Which techniques and approaches should be used for a given requirements elicitation activity?

3.1 Techniken und Ansätze

3.1.1 Task Analysis and Domain Analysis

Beschreibung der Aufgaben, Rollen und Aufgabendomains in der aktuell vorgestellten Situation.

Task Modeling

- *Hierarchische Beschreibung* in SubTasks und deren Beschreibung
- Tasks werden ausgeführt um ein Ziel zu erreichen; meist von Actor in bestimmter Rolle
- Alternative: *Temporale Beschreibung*

Task Domain

oft objektorientiert modelliert

Wurzel eines Task Models ist ein Use Case

Analysis Synthesis Bridge Model

3.1.2 Repertory Grids

liefert impliziertes Wissen

basiert auf den persönlichen Konstrukten eines Individuums, in denen die Interaktion mit anderen eingeordnet wird -> hat Einfluss auf die weitere Interaktion mit anderen Menschen

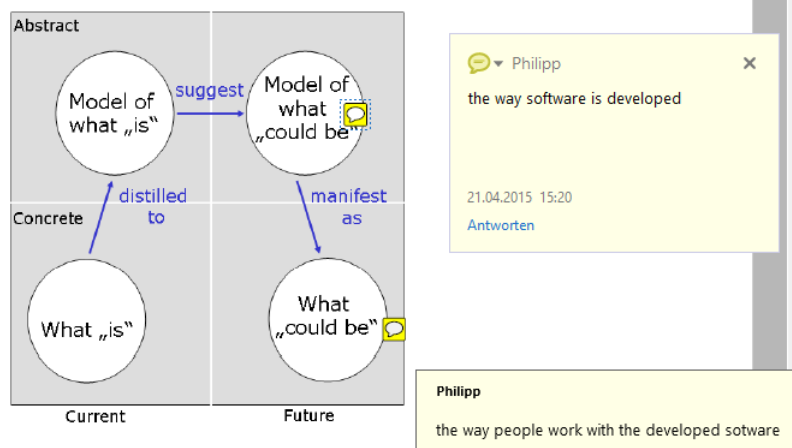


Abbildung 6: Analysis Synthesis Bridge Model