Zusammenfassung Requirements Engineering

Philipp Jäcks

13. Januar 2016

Inhaltsverzeichnis

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.
In RE muss der Analyst verstehen:

- System-as-is
- System-to-be
- System-to-be-next

Daraufhin wird ein Model der aktuellen Situation erstellt und das in ein Model der vorgestellten Situation überführt. (Grob zsmf)

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 neus

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 Domain Requirements

Ergeben sich aus der Anwendungsdomäne des Systems und spiegeln die Charakteristik und Bedingungen der Domäne wieder. Kann sowohl non-functional als auch functional R sein.

1.4 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

1.5 Software Requirements Specification (SRS)

Eigenschaften einer guten SRS:

- Korrekt, Komplett, Konsistent, Verifizierbar...

1.6 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

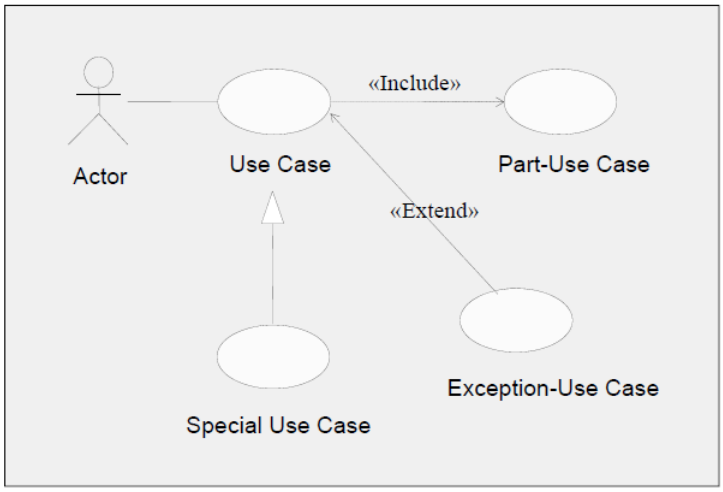

Abbildung 1: Elements of Use Case

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 Detailt, Summary, Primary Task etc
- Folgende sind optional
- Actor
- Stakeholder
- Conditions
- Trigger

Report ist wichtiger als Use Case Model (<= Das Diagramm).
Quasi Semi-Structured textuelle Beschreibung

- Use Case Name
- Primary Actor
- Stakeholder and Interests
- Preconditions
- Success Garantie
- Trigger
- Main Success Scenario
- Extensions (zum main scenario)

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

Use Case Analysis

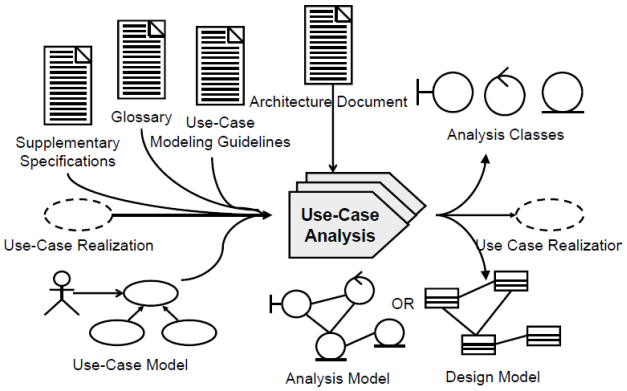

Abbildung 2: Use Case Analysis

Summary

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

2.1 Glossar

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

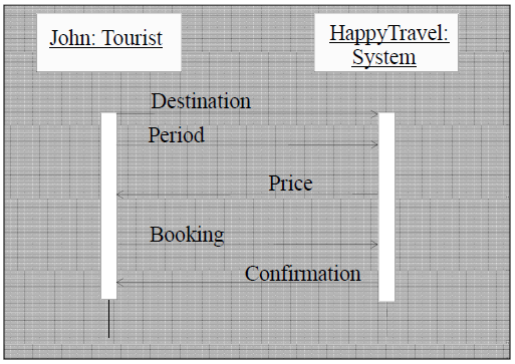

Abbildung 3: Ex Sequence Diagram

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

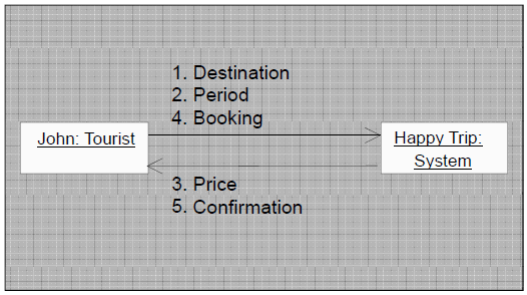

Abbildung 4: Ex Collaboration Diagram

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" sind0
  - 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; vorgeschrieben;...
5. R erheben von Stakeholder und anderen Quellen
   Ergebnis = detaillierte Menge von R in natürlicher Sprache und einfachen Diagrammen

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 Aufgabendomain in der aktuell vorgestellten Situation.

Task Modeling

- *Hierarchisches 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

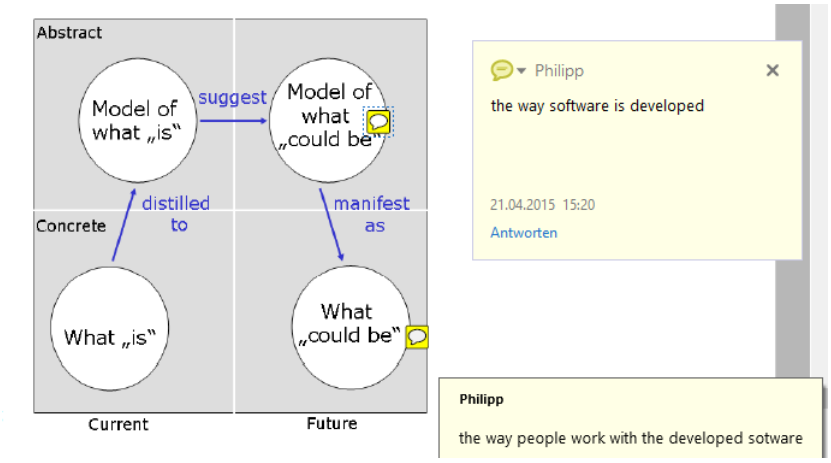

Abbildung 6: Analysis Synthesis Bridge Model

- Zuordnung von Werten zu einer Domain Entität
  Ergebnis => System wird als Matrix modelliert bei der die Elemente des Systems kategorisiert werden
- Ziel ist das Identifizieren von Unterschieden und Gemeinsamkeiten zwischen den einzelnen Domainbereichen

3.1.3 Interviews

- meist verbreitet und oft eingesetzt
- Human Based social Activity -> eher informal und Ergebnis hängt stark vom Zsm-spiel der Beteiligten ab
- große Datenmenge in kurzer Zeit mgl
- 3 Arten:
  - unstructured: kein vordefinierter Ablauf von Fragen etc; oft eingesetzt wenn Domäne etc noch sehr unklar formuliert; Risiko das manche Themen komplett wegfallen
  - semi-structured: Zwischending
  - structured: vorgefertigter Fragenkatalog, der abgearbeitet wird; Erfolg hängt von den richtigen Fragen ab, wann und wem sie gestellt werden

3.2 Contextual Design Approach

Grundidee: Daten vom Kunden sammeln und daraus entscheiden was das System können soll
Ablauf in 2 Phase

1. Kontextuelle Anfrage:
    - Kombination aus verschiedenen Elicitation Techniken (Interview, Observation)
    - Führende/Lenkende Prinzipien
        - Kontext: analysieren des Kundenarbeitsplatzes
        - Partnership: Analyst sucht nach Strukturen/Abläufen in der Arbeit; Kunde erläutert wie Arbeit tatsächlich abläuft (Technik: Kunde bringt dem Analysten den Arbeitslauf bei)
        - Interpretation: Analyst interpretiert gesammelte Daten über Arbeitsplatz und gleicht mit dem Eindruck des Kunden ab
        - Fokus: jede Elicitation Technik fokusiert auf einen Teilaspekt. Am Ende muss das "Große Ganze" daraus entstehen
2. Erstellung von 5 Work Models
    - Flow Models: Beschreiben Arbeitsteilung und -koordination aus Sicht eines Arbeiters
      oft: mehrere Flow Models (aus unterschiedlichen Perspektiven) benötigt
    - Sequence Models: Beschreibt konkrete Aufgaben der Arbeit
      Zweck, Trigger, Arbeitsschritte, Unterbrechungen, Probleme,
    - Artefact Models: beschreibt Artefakte
      Wann/von wem erstellt?; Struktur; Inhalt des Artefakts; Wie werden sie präsentiert?
    - Cultural Models: Organisationskultur -> beschreibt Erwartungen, Wünsche der Arbeiter; organisatorische Vorschriften
    - Physical Models: beschreibt den physikalischen Arbeitsplatz -> Aufbau; Bewegung innerhalb; eingesetzte Technik;
3. Analyse der 5 Modelle um die 'System-to-be' zu bestimmen; Quasi das Redesign der Arbeit erbringt das Design der Software

4 Erweitern der Requirements Erkenntnisse

RE = koordinierte Menge von Aktionen zum Entdecken, Evaluieren, Dokumentieren, Konsolidieren, Überarbeiten von Zielen, Bedingungen, Annahmen, Fähigkeiten die das 'System-to-be' erfüllen sollte um Probleme des 'System-as-is' zu lösen und neue Möglichkeiten zu liefern.

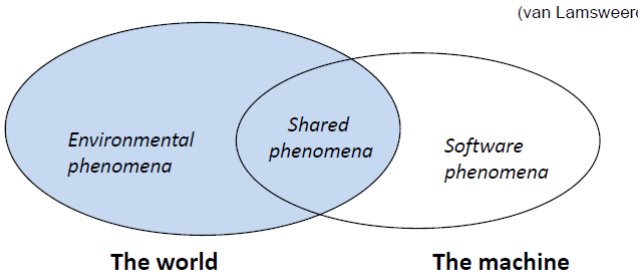

Abbildung 7: Why, What and Who Dimension of RE

R beziehen sich auf die Umwelt und betreffen die Maschine nur indirekt.

- *System R*: vorgeschrieben Angaben, die das 'System-to-be' u.U. mit anderen System Komponenten, leisten kann
- *Software R*: vorgeschriebene Angaben, die **ausschließlich** das 'System-to-be' leisten kann;
  - Domain Properties: Invariante Bedingungen
  - Annahmen: müssen erfüllt werden von Umwelt

4.1 Why, What and Who Dimension of RE

- **Why-Dimension** kontextbezogene Gründe für ein neues System müssen gegeben werden - in Bezug auf die Ziele, die das neue System erfüllen soll
  Ziele anhand der Beschränkungen des aktuellen Systems herausarbeiten
  Zielfindung liefert oft Konflikte, da Ziele aus unterschiedlichen Perspektiven def. werden
- **What-Dimension** Bestimmen der Dienste, Bedingungen, Annahmen des System-to-be anhand der bestimmten Ziele mit Hilfe von Elicitation Techniken
- **Who-Dimension** Bestimmt die Kompetenzen, die notwendig sind um Ziele zu erreichen

4.2 Darstellung der R

- natürliche Sprache für einzelne R nutzen
  Verbesserung durch: Strukturierung der R; Regeln für das beschreiben (z.b. Verbot

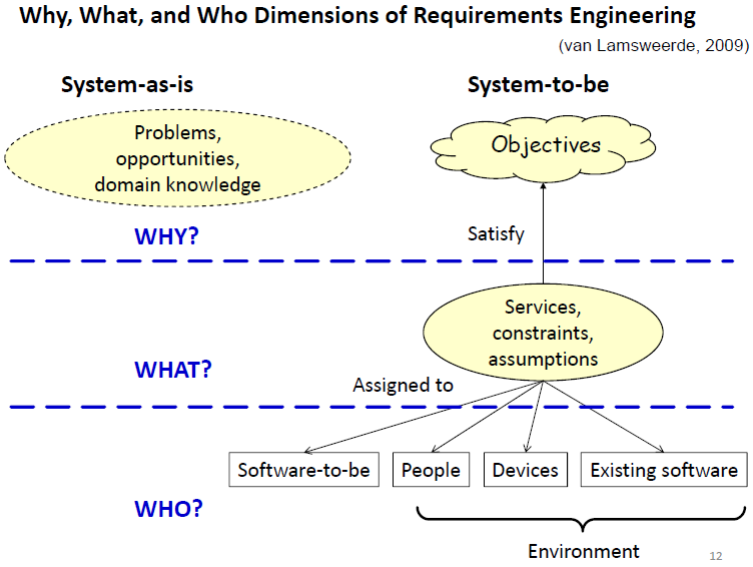

Abbildung 8: Why, What and Who Dimension of RE

bestimmter Wörter); Redundanzprüfung
Limitierung: nicht zwingend eindeutig; Verknüpfung zw R kann schwer zu ziehen sein

- Graphische Notation, die Daten, Funktionen, Verhalten beschreiben
- Formale Modelle
- Formalisierung R durch
  Designation = Formale Grundterm (e.g. Prädikat) durch informelle Beschreibung erklären
  Definition Anhand definierter Termine neue Termine def.; lässt sich besser begründen, aber ist eingeschränkter in der Beschreibung

4.3 Zusammenfassung

- R beschreibt Bedingung über ein Phänomen der Umwelt
- Beschreibung der R muss die Trennung zw. Maschine und Umwelt respektieren und zw
  Umwelteigenschaften, die gegeben sind (indicative) und die von der Maschine geleistet werden (optative)

5 Z-Notation

- basiert auf Mengenlehre und Prädikatenlogik 1. Stufe
- beschreibt Zustände (durch Menge, Objekte, Prädikate) und Übergänge (durch Beschreibung des Zustands vor und nach Übergang) des Systems
- Schema beschreibt Zustand bzw. Übergang
- strukturiertes Wissen über Domäne wird als Ausgangspunkt zum Modellieren benutzt - *Domäne* = exist. und aufgestellte Systeme die ähnliche Ziele haben
  Aufgestelltes System = System (Entitäten die miteinander interagieren um ein Domainziel zu erreichen) + Umwelt(Elemente, die Input geben und auf Output reagieren)
- **Modelling the domain**
  - als Graph; V = Aussagen über zu erreichende Ziele; formale Spezi, das Verhalten des Systems beschreibt; Erklärungen (natürliche Sprache)
  - E = Beziehungen zw Zielen verschiedener $v \in V$; 4 Arten:
    is_supported_by; is_undermined_by(untergraben; x wird ruch y gestört); may_be_specialised_by; must_be_considered_before(geprüft)
  - iterativer prozess; Aufstellen von high und low level goals:
    high-level: Knoten haben selten Fragmente formaler Spezi
    low-level: Knoten haben Fragmente formaler Spezi
  - Basic System Node: sollte die Basisfunktionalität des Systems beinhalten; mit Stakeholder abgeglichen werden; Ausgangspunkt für weitere Spezifikationen

Pros	Cons
Formal, Strukturiert besseres Verständnis der R durch Domain Model	R des Kunden formalisieren nicht trivial iterative Entwicklung kann inkonsistente Spezi hervorrufen