



Strukturiertes Requirements Engineering:

Modellbasierte Anforderungsbeschreibung

Focus on Requirements

Manfred Broy

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany

Beobachtung - Problemfelder in der Softwareentwicklung

- Anforderungen: Requirements Engineering
 - ◊ Die Anforderungen bestimmen den Funktionsumfang eines Systems und damit seinen Wert (Nutzen für den Verwendungszweck) und die Kosten für seine Entwicklung
- Architekturentwurf
 - ◊ Die Architektur bestimmt die Beherrschbarkeit eines Systems aus Entwicklersicht
- Qualitätssicherung
 - ◊ Die Qualität der Realisierung eines Systems bestimmt das Risiko und den Overhead bei seiner Nutzung

TUM, October 17th, 2003 Manfred Broy TUM 2

Hauptaufgaben im RE

- Ziel- und Anforderungserfassung
 - ◊ Von allen relevanten Interessengruppen („Stakeholder“) werden alle Ziele und Anforderungen zusammengetragen
- Anforderungsbewertung
 - ◊ Die Ziele und Anforderungen werden bewertet und priorisiert
- Anforderungsbeschreibung
 - ◊ Anforderungen werden präzise formuliert
- Anforderungsrealisierung
 - ◊ Die Anforderungen werden in den Architekturentwurf
- Anforderungsverfolgung und Verifikation
 - ◊ Die Realisierung wird in Hinblick auf die Erfüllung der Anforderungen überprüft

TUM, October 17th, 2003 Manfred Broy TUM 3

Hotspots in RE

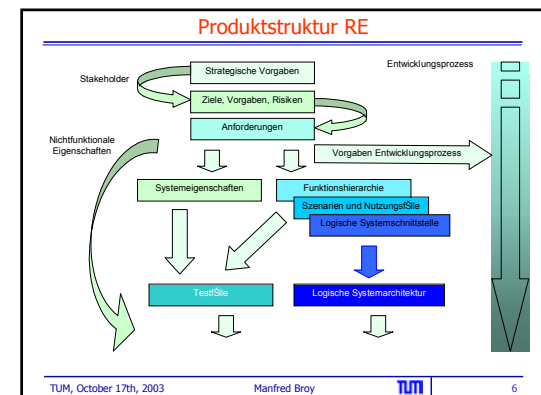
- Ziele festlegen
- Risiken und Optionen analysieren
- Anforderungen ableiten bzw. vollständig erfassen
- Anforderungen bewerten (Kosten/Nutzen)
- Anforderungen strukturieren
- Anforderungen präzisieren
- Anforderungen in Architektur reflektieren
- Anforderungen sicherstellen
- Anforderungsänderungen managen
 - ◊ Änderungsbedarf erfassen
 - ◊ Änderungen entscheiden
 - ◊ Änderungen umsetzen

TUM, October 17th, 2003 Manfred Broy TUM 4

Von Zielen zu Anforderungen

- Ziel: Festlegung allgemeiner Vorgaben
 Beispiel:
 ◊ Das System muss international einsetzbar sein
 ◊ Die Entwicklung findet mit Systematik statt
- Anforderung: Aus Zielen abgeleitete Detailfestlegungen zu Eigenschaften an
 - ◊ das zu entwickelnde System (funktional/nichtfunktional)
 - ◊ den Prozess
 Beispiel:
 ◊ Die Nutzerschnittstelle muss unterschiedliche Sprachen unterstützen
 ◊ Der Prozess muss CMMI Level 3 zertifiziert sein
 Anforderungen legen fest, wie Ziele erreicht werden

TUM, October 17th, 2003 Manfred Broy TUM 5



Ziele/Anforderungen sammeln

- Interessengruppen bestimmen (Stakeholder: Marketing, Auftraggeber, Nutzer, Entwickler, Betreiber, Wartung)
- Ziele/Anforderungen erfassen
 - ◊ Auf Hintergrund und Fähigkeiten der jeweiligen Interessengruppe ausgerichtet
 - ◊ Anforderungen grob dokumentieren
 - ◊ Gewichtung ermitteln
 - ◊ Ziele/Anforderungen strukturieren

Wichtige Aufgabe:
Von Anforderungen zu Zielen abstrahieren

Ziele festlegen

- Auf Basis der gesammelten Zielvorstellungen und Anforderungen werden Ziele festgelegt und priorisiert

Wesentliche Aufgaben

- Ziele herausarbeiten
 - Ziele vor dem Hintergrund der strategischen Vorgaben bewerten
 - Risiken abschätzen
 - Kosten/Nutzenbewertung
 - Ziele festschreiben und priorisieren
 - Zielfestlegungen begründen
- Hierbei werden wesentliche Richtungsentscheidungen getroffen

Von Zielen zu Anforderungen

- Jedes Zielvorgabe wird in eine Reihe von Anforderungen umgesetzt

Aufgaben

- Zu Zielen Anforderungen herausarbeiten
- Gegebenfalls Alternativen entwickeln
- Zielerreichung und Kosten bewerten
- Anforderungen festschreiben

Hierbei werden wesentliche Entscheidungen getroffen, wie ein Ziel konkret erreicht werden soll

Funktionale Anforderungen

- Ein Hauptziel des Requirements Engineering ist die Festlegung der funktionalen Anforderungen

Die funktionalen Anforderungen bestimmen

- Das Verhalten des Systems aus Sicht der Nutzung
- Den Funktionsumfang des Systems
- Seine fachliche Anwendungsmöglichkeiten

Hierbei ist festzulegen, welche Einzelfunktionen ein System zur Verfügung stellt und welches Verhalten dies für das System genau erfordert.

Funktionale Anforderungen bewerten

Klassifizierung von funktionalen Anforderungen

- Unverzichtbare Basisfunktionen (Sine qua non)
 - ◊ Systemeigenschaften, die ein System unbedingt haben muss, damit es seine Ziele erreicht
- Nützliche Zusatzfunktionen (Nice to Have)
 - ◊ Systemeigenschaften, die wünschenswert, aber nicht entscheidend sind
- Alleinstellungsmerkmale (Unique Selling Points)
 - ◊ Systemeigenschaften, die dem System einen Vorsprung vor anderen Systemen einräumt

Zielorientiertes RE

Ein erstes RE-Dokument hat die Ziele festzulegen und zu bewerten und zu priorisieren

Daraus sind die Anforderungen abzuleiten

- Funktionale Anforderungen
- Nichtfunktionale Anforderungen
 - ◊ Zuverlässigkeitsanforderungen
 - ◊ Wartung und Betriebsdauer
 - ◊ Realisierungsanforderungen (Hardware etc.)
 - ◊ Performanzanforderungen
 - ◊ Prozessanforderungen
- Ausbaukonzept (gestufte Anforderungen)
- Risiken

Funktionsorientiertes RE

- Das System wird in eine Hierarchie von Funktionen strukturiert (Funktionsnetze, Feature Trees)

Aufgaben

- Funktionenhierarchie erstellen
- Einzelfunktionen modellieren
- Querbezüge beschreiben
- Nutzerschnittstellenverhalten Gesamtsystem daraus kombinieren

Multifunktionale Systeme

Funktionale Komplexität:

- Typischerweise bieten heute Systeme viele unterschiedliche Funktionen an
- Jede Einzelfunktion dient einem Verwendungszweck (Nutzungsfall) des Systems
- Zwischen der Einzelfunktionen bestehen unterschiedliche Abhängigkeiten

Das RE muss auf diesen Umstand ausgerichtet sein.

Multifunktionale Systeme können in Funktionenhierarchien strukturiert werden

Funktionenhierarchie

Eine Funktionenhierarchie besteht aus folgenden Angaben

- Funktionsgruppen
- Liste der Funktionen erstellen
- Funktionen klassifizieren
 - ◊ Einzelfunktionen/ Grundfunktionen
 - ◊ Kombinierte Funktionen/Zusammengesetzte Funktionen
 - ◊ Crossfunktionalität
 - ◊ Sekundärdienste
- Abhängigkeiten zwischen Funktionen

Abhängigkeiten zwischen den Funktionen

In der Funktionenhierarchie werden Abhängigkeiten und Beziehungen zwischen Einzelfunktionen durch Relationen dargestellt.

Typische Relationen:

- Funktion A ist unabhängig von Funktion B
- Funktion B ist abhängig von Funktion A
 - ◊ Funktion A ermöglicht Funktion B (B nutzt A)
 - ◊ Funktion A verändert das Verhalten der Funktion B

Modellierung von Einzelfunktionen: Spezifikation

Ergänzend zur Funktionenhierarchie wird das Verhalten der Einzelfunktionen modelliert

- Informelle Szenarien
- Ausschnitt aus Datenmodell
- Sequenzdiagramme zur formalen Darstellung der Interaktion (exemplarisch, repräsentativ)
- Schnittstellenbeschreibung der Einzelfunktion
 - ◊ Durch Zustandsmaschinen
 - Zustandsübergangdiagramme
 - Tabellen
 - ◊ Durch Angabe von Eigenschaften

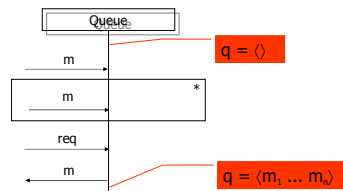
Nutzungsszenarien

- Jede Einzelfunktion wird in Nutzungsszenarien beschrieben

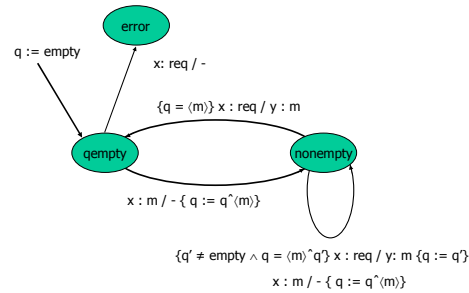
Bestandteile:

- Beteiligte Nutzer (Personen und Systeme)
- Informelles Nutzungsszenario
- Ausschnitt aus Datenmodell
- Sequenzdiagramme
- Zustandsautomat

Szenarien für Nutzungsfälle



Queue als Zustandsmaschine



Beispiel: Password Funktion

```

Let
type PswIn = { quit } ∪ { psw(w); w ∈ Password }
type PswOut = { pok, qok, perror }

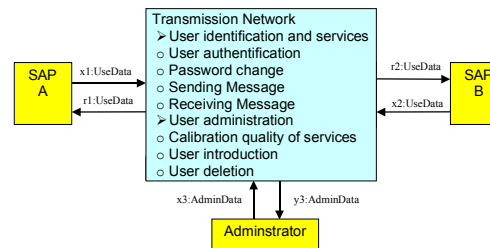
PswService(type M)
in x: M | PswIn
out x: M | PswOut
x = P(x') where for z: Seq M
valid(w) ⇒ P((psw(w))^z^quit^x) = ⟨pok^z^qok⟩P(x)
valid(w) ⇒ P((psw(w))^z^psw(w')^x) = P((psw(w))^z^x)
¬valid(w) ⇒ P((psw(w))^x) = ⟨perror⟩P(x)
P(z^x) = P(⟨quit^x) = P(x)
    
```

Schnittstellenmodellierung

Prinzipiell bestimmen die funktionalen Anforderungen in ihrer Gesamtheit das Schnittstellenverhalten aus Nutzersicht

- Das Gesamtverhalten des Systems wird aus Schnittstellen/Nutzungssicht durch eine abstrakte Zustandsmaschine dargestellt

Beispiel: Multi-funktionales System



Nutzungssichten

Die Schnittstelle/Nutzungssicht kann aus ganz unterschiedlichen Perspektiven beschrieben werden:

- Nutzeroberfläche
- Abstrakte (logische) Nutzungssicht

Die detaillierte Dialogsicht ist eine Verfeinerung der logischen Nutzungssicht und nicht Teil der Anforderungen

Übergang zur Architektur

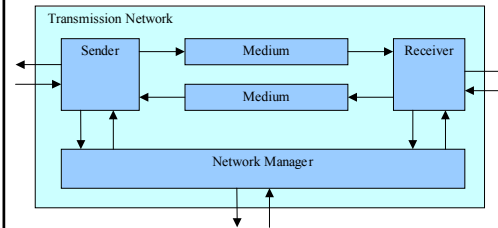
Aus dem RE wird die Systemarchitektur festgelegt

Wir unterscheiden:

- Direkte Architektur Anforderungen (Portierung, Schichtung, Einbezug vorgefertigter Systemteile und von Infrastruktur)
- Indirekte Architektur Anforderungen (Skalierung, Performanz, Strukturierung in Hinblick auf Nutzungsfunktionalität)

Der kontrollierte Übergang von den Anforderungen zur Architektur wird noch ungenügend beherrscht (Rückkopplung Anforderungsfestlegung/Architektur)

Architektur



Testfälle

Aus dem RE ergeben sich Testfälle:

- Ein Testfall ist ein Systemablauf (Sequenzdiagramm), der Eingabestimuli und Ausgabereaktionen festlegt
- Zusätzlich wird dokumentiert, welche Eigenschaft ein Testfall überprüft
- Negative Testfälle dienen der Überprüfung von Anforderungen, die festlegen, dass bestimmtes Verhalten auf keinen Fall auftreten darf
- Testfälle sind auf eine bestimmte Abstraktionsebene in der Modellierung eines Systems ausgerichtet

Funktionshierarchie für Komponenten

Wie auf dem Gesamtsystem kann für jede Komponente der Architektur eine Funktionshierarchie definiert werden

Zwischen den Funktionen der Funktionshierarchie des Gesamtsystems und den Funktionen der Funktionshierarchie der Teilsysteme (Komponenten) können wieder Beziehungen definiert werden.

Vorteile der Modellierung

Eine Modellbildung auf RE Ebene erlaubt:

- Präzise Strukturen zur Beschreibung der Anforderungen
- Überprüfung der informellen Anforderungsdokumente auf Vollständigkeit und Konsistenz („Anforderungvalidierung“)
- Überprüfung der verhaltensorientierten Anforderungsmodellierung auf wesentliche Eigenschaften („Anforderungsverifikation“)

Ziel- und Funktionsorientiertes RE/SE

- Grundlagen:
 - ◇ Multi-funktionale Systeme
 - ◇ Strukturierte Darstellung der Systemfunktionen
 - ◇ Abstraktionsebenen
 - ◇ Funktionshierarchie
 - ◇ Übergang zur Architektur
- Vision: Ziel und Funktionsorientiertes Software Engineering
 - ◇ Funktionen als Anforderungsbausteine
 - ◇ Funktionen als Aspekte
 - Schnittstellen
 - Architekturen