Software Engineering und Projektmanagement 2.0 VO

Projektphase Analyse (Was bauen wir?)

„If you don't know where you're going, you're unlikely to end up there.“ - Forrest Gump

www.inso.tuwien.ac.at

INSO - Industrial Software
Institut für Rechnergestützte Automation | Fakultät für Informatik | Technische Universität Wien

Lernziele

- Die Rolle der Stakeholder in einem Softwareprojekt
- Prüfung der Machbarkeit mittels Machbarkeitsstudie
- Die Entscheidung, ob Software entwickelt oder gekauft wird *(Make or Buy)*
- Welche Anforderungstypen gibt es und was sind gute Anforderungen?
- Wie können Anforderungen modelliert und beschrieben werden?
- Wie können Anforderungen verwaltet und die Nachverfolgbarkeit sichergestellt werden?
- Organisatorische und soziale Aspekte bei der Analyse

Beispiel Anforderungsqualität

- Die beschriebenen Anforderungen sind zueinander widersprüchlich.
- Es existieren weitere „unsichtbare“ Vorgaben, z.B. Gesetzestexte, die in diesem Fall eine Relevanz haben.
- Die „richtige“ Interpretation (und damit die Auflösung des Widerspruches) ist nur mit Wissen über die weiteren Vorgaben möglich.
- Es existiert die Gefahr, dass unterschiedliche Interpretationen in den Projektphasen und von verschiedenen Projektbeteiligten getroffen werden.
- Die Interpretation wird oftmals nicht gesondert dokumentiert.

Definition Anforderung

Folgende Definitionen werden nach dem IEEE-Standard 610.12 (1990) gegeben:

1. „A condition or capability needed by a user to solve a problem or achieve an objective.“
2. „A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.“
3. „A documented representation of a condition or capability as in (1) or (2)“

Fragestellungen in der Analyse

1. **Woher kommen Anforderungen?**
   - Was sind Stakeholder?
   - Andere wichtige Anforderungsquellen
2. **In welchem Kontext wird gebaut?**
   - Machbarkeit und Machbarkeitsstudie
   - Make or Buy
   - Abgrenzung
3. **Was wird gebaut?**
   - Anforderungserhebungs-Methoden
   - Klassifikation von Anforderungen
   - Was sind gute Anforderungen
   - Anforderungsmodellierung
   - Anforderungsmanagement

Woher kommen Anforderungen?

Stakeholder und Anforderungsquellen

Stakeholder

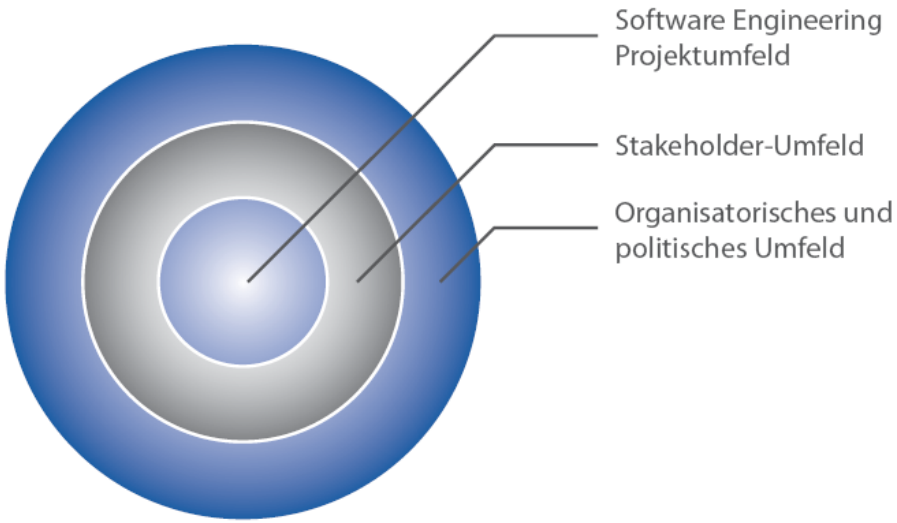
- „Anteilhalter“ der ein Interesse im Projekt vertritt und damit eine Anforderungsquelle darstellt

Stakeholdergruppe

- Gruppe von Stakeholdern mit gleichem Interesse
- Jede Stakeholdergruppe hat eine eigene Sicht auf das Projekt
- Jede Stakeholdergruppe hat einen eigenen Lingo (z.B. Technikerlingo)

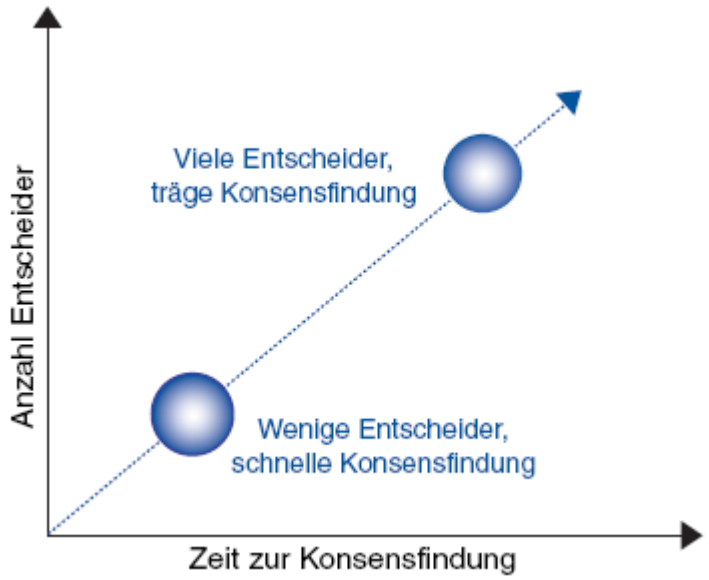
Konflikte → Anforderungspriorisierung (!)

Stakeholder ist jede Person/Institution, der/die ein Projekt beeinflussen kann oder die durch das Projekt beeinflusst wird



Typische Stakeholder

- Kunde/Auftraggeber/Sponsor
- Geschäftsführung/Abteilungsleiter (Auftraggeber- und Auftragnehmerseite)
- Legacy Owner (Besitzer des Altsystems oder der Altdaten)
- Betreiber/Entwickler von Schnittstellensystemen (Systemen, mit denen integriert wird)
- Benutzer/Benutzervertreter (pro Benutzergruppe)
- Projektleiter
- Projektsteuergruppe
- Systemarchitekten/IT-Stratege
- Softwareentwickler
- Qualitätssicherung und Test
- Betrieb
- Wartung



Weitere wichtige Anforderungsquellen

Standards und Normen

Datenschutz

Gesetze

Projektumfeld

- Anforderungen und Funktionsweisen von alternativen Systemen (z.B. Konkurrenzanbieter)
- Unternehmenskultur

Verdeckte Regeln

- Glaubensgrundsätze/Tabus/Macht/Werte/Einstellungen/ Status

In welchem Kontext wird gebaut?

Zielsetzung der Analyse

Strategische Ziele

- langfristige Unternehmensziele, z.B. Hebung des Unternehmensimages

Operative Ziele

- mittel- bis kurzfristige Ziele, z.B. Einhaltung der Projektkosten

Führungsziele

- lang-, mittel-, oder kurzfristige Ziele, die sich mit internen Abläufen und dem Faktor Mensch befassen, z.B. Steigerung der Mitarbeitermotivation im Unternehmen/im Projekt

Vision und Scope

Vision (üblicherweise stabil)

- Zweck, Nutzen, Business Case – Beispiel: Im Jahr 2020 werden infrastrukturelle Änderungen notwendig, die Anpassungen an einem end-of-life Softwareprodukt erfordern. Von dem neuen Produkt wird erwartet, dass die 30 Jahre fachliches Wissen der alten Lösung enthalten sind, sowie dass möglichst rasch die Lizenzkosten für das alte System eingespart werden können.

Scope (üblicherweise veränderbar)

- (fachlicher) Umfang des Projekts – Beispiel: Eine Funktionsliste der existierenden Software; Eine Ansammlung von Use Cases, die umgesetzt werden müssen; Achtung: fachlicher Scope != Scope des gesamten Projekts (inkludiert z.B. Deployment, PM Overhead, QS, etc.)

Nachdem sich der Umfang im Projekt ändern kann, wird üblicherweise eine *Projektsteuergruppe* für die Freigabe der Änderung des Scopes eingerichtet → Aufgabe des Projektmanagement

Machbarkeit und Machbarkeitsstudie

- Identifikation des Inventionsanteils
- Identifikation von ähnlichen Systemen (z.B. auch von Konkurrenzanbietern)
- Technische Modellierung
- Technische Prototypen
- Rechtliche Prüfung (z.B. landesweit, europaweit oder weltweit)
- Akzeptanzanalyse bei den zukünftigen Benutzern
- Identifizierung der wesentlichen Kostentreiber (z.B. nichtfunktionale Anforderungen)
- Kostenanalyse

Make or Buy – Einflussfaktoren

- Abdeckung des aktuellen und des antizipierten Projekt-Scope
- Entwicklungskosten
- Lizenzkosten (Anschaffung und laufender Betrieb)
- Anpassungsmöglichkeiten und Anpassungsaufwand
- Schulungsaufwand für Entwickler und Betreiber
- Hardwareanforderungen
- Support und Wartung
- Nutzung von offenen oder proprietären Protokollen oder Formaten
- Herstellerabhängigkeit (Vendor Lock-In)
- Besitzverhältnisse
- Weitere strategische Elemente, z.B. Weiterverkaufsmöglichkeit einer Neuentwicklung

Zusammenwirkung von Grobarchitektur und Analyse

Die Grobarchitektur beeinflusst die Analyse

- Wesentliche nicht-funktionale Eigenschaften werden bestimmt
- Einfluss auf Anforderungsanalyse und –Beschreibung
  - Kann durch robuste Analysemethodik komplett abgefangen werden
- Strukturierung der Anforderung ausgehend von wesentlichen Systemkomponenten
  - Kann auch Geschäftsprozess-orientiert geschehen
- Applikationstyp wird festgelegt
- Flexible oder feste Systemteile werden eingeteilt

Schnittstellen werden bestimmt

- Wichtig für Stakeholdermanagement
- Extrem wichtig für Projektplanung

Schnittstellenidentifikation

Die meisten aktuellen Software (SW)-Systeme interoperieren mit anderen vorhandenen Systemen.

Viele Projekte haben einen höheren Integrationsaufwand (Gluing) als Implementierungsaufwand (Building).

Strukturiere Notationen (z.B. UML-Sequenzdiagramme) sind effektiv zur Beschreibung von Schnittstellen.

Eine exakte Schnittstellenbeschreibung ist notwendig für die rechtliche Abgrenzung von Systemteilen und Verantwortlichkeiten.

Schnittstelle = Bruchstelle: Schnittstellentestfälle (Szenarien über die Schnittstellengrenzen hinaus) werden von der Schnittstellenbeschreibung abgeleitet.

Das Management des Fehlverhaltens (Exceptionhandling) muss über die Schnittstellen hinaus spezifiziert werden.

Was wird gebaut?

Ermitteln von Anforderungen – Auszug von Methoden

Workshops

- Analyst trifft mehrere Stakeholder um Ergebnisse zu erarbeiten
- Unterstützt den Aufbau eines effektiven Teams, das gemeinsam auf ein Ziel hinarbeitet
- Alle Stakeholder werden einbezogen → schafft Stakeholder Buy-In
- Erarbeitung eines Konsens an die Eigenschaften des zu entwickelnden Systems
- Identifizierung und Lösung von politischen/verborgenen Aspekten, die Projekterfolg gefährden würden
- Ergebnis ist Vorabversion der funktionalen oder nicht-funktionalen Anforderung

Interviews

- Analyst trifft einen Vertreter der Stakeholder (Opinion Leader, Key User) um dessen Wünsche und Anforderungen aufzunehmen
- Weniger offener, strukturierter als Workshops
- Wichtig: kontextfreie, also allgemeine Fragen stellen
  - Warum existiert dieses Problem?
  - Wie lösen Sie das Problem heute?
  - Wie würden Sie das Problem gerne lösen?

Beobachtung

- Bei existierenden Systemen/Prozessen wird Vorgehen der Anwender beobachtet
- Analyst versucht durch Befragung Vorgehen der Anwender zu verstehen

Mitarbeit

- Ähnlich zu Beobachtung, aber Analyst muss tatsächlich mitarbeiten

Bei allen Interaktionen mit Stakeholdern sollten Protokolle/Mitschriften erstellt werden

Anforderungsvolatilität

Stabile Anforderungen

- Anforderungen, die sich (voraussichtlich) nicht oder nur geringfügig ändern werden

Volatile Anforderungen

- Anforderungen, die sich (voraussichtlich) während der Anforderungsanalyse oder während des Betriebes des Systems ändern werden

Systemdesignauswirkungen:

- Stabile Anforderungen können mit simpleren Designs bzw. „*hard-wired*“ *implementiert* werden.
- Die Umsetzung volatiler Anforderungen erfordert ein adaptiveres Systemdesign („*soft-wired*“).
- Stabile Anforderungen können früher und detaillierter dokumentiert bzw. modelliert werden.
- Stabile Anforderungen können früher im Entwicklungsprozess implementiert werden.

Baseline und Änderungen an selbiger

Es existiert niemals ein echtes Einfrieren (Freeze) der Anforderungen

Trotzdem sind stabile Zustände in der Analyse unabdingbar → Baseline

- „A specification or system that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development and can be changed only through formal change control procedures.“ – IEEE-Standard 610.12 (1990)

Nach dem Setzen einer Baseline wird bei Änderungen Folgendes durchgeführt:

- Change Impact Analysis
- Re-Costing

Achtung: Änderungen der Baseline sind Teil des Projektmanagements

- Im Rahmen der agilen Softwareentwicklung werden ständig Trade-offs vereinbart – einen vernünftigen Change Request Prozess hierfür zu finden liegt im Rahmen des Projektmanagements bzw. des vertraglichen Rahmenwerks

Klassifikation von Anforderungen

Unterscheidung in Benutzeranforderungen und Systemanforderungen und desweiteren in:

Funktionale Anforderungen:

- Beschreibung der Funktionen/Services, die das System zur Verfügung stellen soll. Dabei muss geklärt werden, wie das System auf bestimmte Eingaben und spezielle Situationen reagieren soll.

Nichtfunktionale Anforderungen:

- Anforderungen an die Funktionen/Services, die nicht deren Verhalten, sondern Vorgaben etwa an Qualitätseigenschaften (z.B. nach der ISO9126), den Entwicklungsprozess, einzuhaltende Standards und Normen und gesetzliche Rahmenbedingungen beschreiben.

Domänenanforderungen:

- Funktionale oder nichtfunktionale Anforderungen, die von der Domäne des Systems definiert werden und die besonderen Eigenschaften/Einschränkungen dieses Gebietes darstellen. Domänenanforderungen sind oft inhärent und nicht explizit pro Projekt dokumentiert.

Was sind gute Anforderungen?

Gute Anforderungen sollen in erster Linie komplett und konsistent sein:

- Vollständigkeit: Alle Anforderungen sollen abgedeckt sein.
- Konsistenz: Anforderungen enthalten keine Konflikte und keine Widersprüche.

Überprüfung von Anforderungen

Anforderungs-Review:

- strukturierte und systematische Überprüfung der Qualitätseigenschaften von Anforderungen.
- Konflikte zwischen Stakeholdern gilt es zu erkennen und qualifiziert aufzulösen (z.B. mittels Priorisierung).
- Die Beteiligung unterschiedlicher Stakeholder (fachliche und technische) ist erforderlich.

Prototyping:

- Dabei wird geprüft, ob die Anforderungen umsetzbar und insbesondere auch eindeutig und vollständig sind.

Abnahmetestfallerstellung:

- Dadurch wird die Testbarkeit (Testability) von Anforderungen erreicht und eine Prüfung auf Eindeutigkeit und Vollständigkeit ermöglicht.

Typische Probleme bei Anforderungen

Verständlichkeit

- Anforderungen werden in der Sprache der Applikations-Domäne repräsentiert. Diese wird von Softwareingenieuren oft nicht verstanden.

Selbstverständlichkeit

- Domain-Spezialisten sind mit der Domäne so vertraut, dass sie bestimmte Anforderungen als selbstverständlich erachten und nicht explizit darstellen und kommunizieren.

Wenig Präzision

- Es ist schwierig, Anforderungen präzise zu definieren, ohne diese schwer lesbar zu gestalten.

Vermischung

- Funktionale und nichtfunktionale Anforderungen werden häufig vermischt.

Zusammenführung

- Verschiedene Anforderungen werden gemeinsam beschrieben, ohne diese genau zu identifizieren und voneinander abzugrenzen.

Mehrdeutigkeit

- Spezifikationen in natürlicher Sprache sind oft mehrdeutig.

Überflexibilität

- Die Beschreibungsart der Anforderung (z.B. die natürliche Sprache) ist oft mehrdeutig interpretierbar und somit zu flexibel, um exakte Beschreibungen zu erzeugen.

Beispiele für Anforderungen

Input

- Ich will, dass das System immer funktioniert
- Wieso dauert das so lang bis ich auf den Knopf drücken kann?
- Jeder Anwender MUSS sich dem System gegenüber eindeutig identifizieren
- Als Anwender möchte ich eine Nachricht eingeben, die auf allen Anzeigen möglichst prominent angezeigt wird, damit ich die Gäste unsere Shopping Mall auf einen Brand hinweisen kann

Output

- Nicht-funktionale Anforderung: Das System MUSS eine Verfügbarkeit von 99,98% bei 24x7h Betriebszeit gewährleisten
- Nicht-funktionale Anforderung: Das System MUSS mindestens 90% der Requests innerhalb von 0,5s beantworten, bei 300 gleichzeitig aktiven Nutzern
- Wurzelanforderung: Jeder Anwender MUSS sich dem System gegenüber eindeutig identifizieren
  - Jeder Anwender MUSS einen eindeutigen Usernamen haben, mit dem er sich dem System gegenüber identifiziert
  - Jeder Anwender MUSS ein Passwort zusätzlich zu seinem Usernamen verwenden
  - Jedes Passwort MUSS mindestens 6 Zeichen lang sein und alpha-numerisch sein, sowie mindestens eine Zahl, sowie einen Groß- bzw. Kleinbuchstaben enthalten
- User Story: Weitere Analyse Schritte sind notwendig
  - Formulieren von Abnahmekriterien – z.B. Nachricht wird nach Anzeigeaktualisierung angezeigt, Nachricht entspricht abgenommenen Design
  - Abgestimmtes und abgenommenes Design mit Usern

Qualitätsattribute von Anforderungen

Vollständig

- Alle Anforderungen müssen explizit beschrieben sein, es darf keine impliziten Annahmen eines Stakeholder über das zu entwickelnde System geben.

Eindeutig definiert/abgegrenzt

- Präzise Definitionen helfen, Missverständnisse zwischen Stakeholdern (z.B. Entwickler und Auftraggeber) zu vermeiden.

Verständlich beschrieben

- Dadurch können alle Stakeholder unter vertretbarem Aufwand die gesamte Anforderung lesen und verstehen.

Atomar

- Es darf nur eine Anforderung pro Anforderungs-ID beschrieben sein.

Identifizierbar

- Jede Anforderung muss eindeutig identifizierbar sein (z.B. über eine ID).

Einheitlich dokumentiert

- Die Anforderungen und ihre Quellen sollten nicht in unterschiedlichen Dokumenten stehen oder unterschiedliche Strukturen haben.

Qualitätsattribute von Anforderungen

Notwendig

- Klären, ob Anforderungen unabdingbar sind.

Nachprüfbar

- Die Anforderungen sollten mit Abnahmetestfällen verknüpft werden, damit bei der Abnahme auch geprüft werden kann, ob die Anforderungen erfüllt wurden.

Rück- und vorwärtsverfolgbar

- Dadurch ist einerseits erkennbar, ob jede Anforderung vollständig erfüllt wurde, und andererseits ist für jede implementierte Funktionalität erkennbar, aus welcher Anforderung sie resultiert, so dass nichts Überflüssiges entwickelt wird.

Priorisiert

- Die Priorität muss festgelegt werden, damit diese im Entwicklungsprozess berücksichtigt werden kann (z.B. durch die Entwicklungsreihenfolge) bzw. konkurrierende Anforderungen aufgelöst werden können.

Anfoderungsmodellierung

Beispielhaft Modellierungsformen/sprachen:

Unified Modeling Language (UML)

(Logische) Datenmodellierung

Use Cases

Geschäftsprozessmodellierung

Unified Modeling Language (UML)

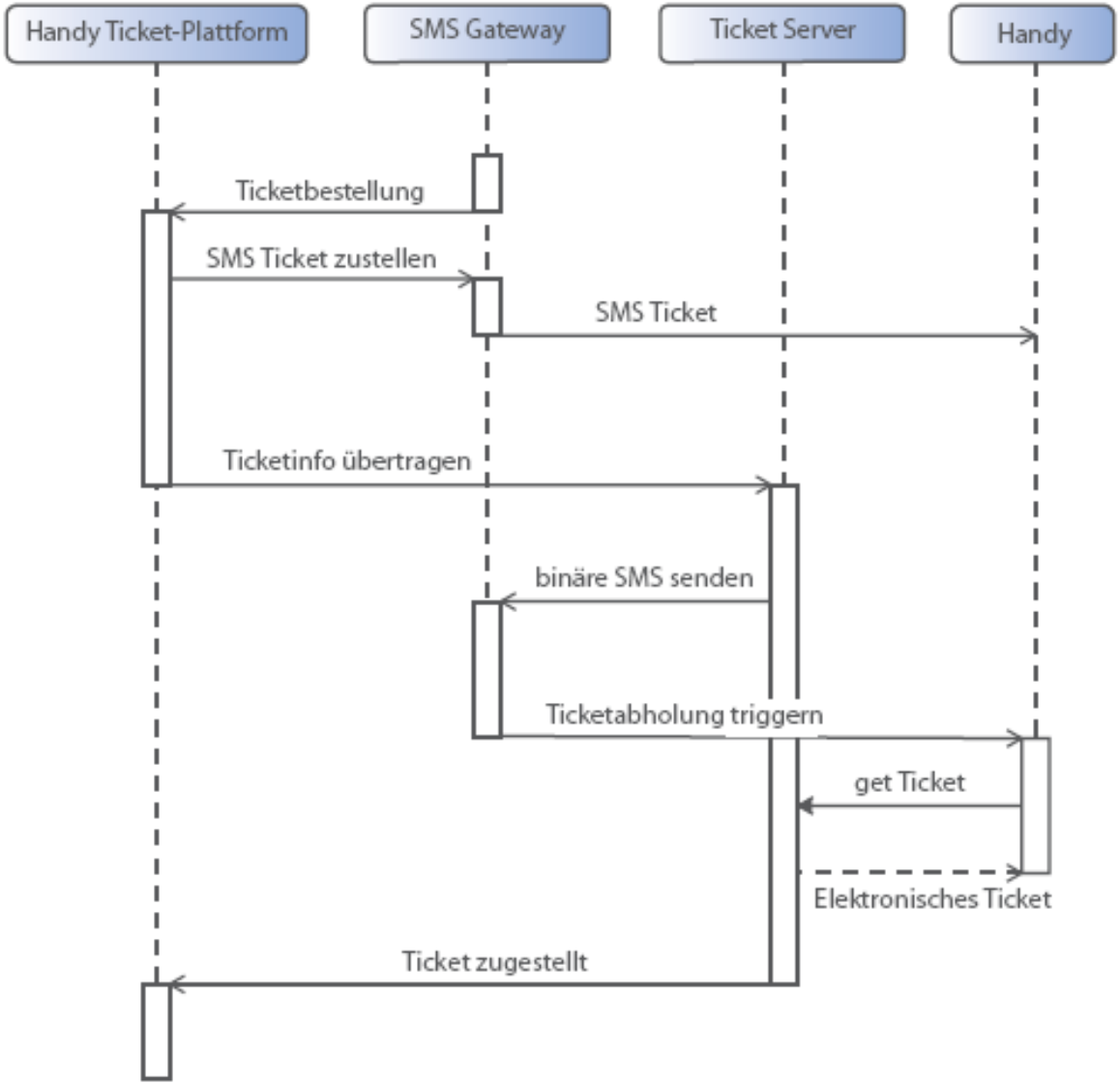
Verhaltensmodelle

- Anwendungsfalldiagramm (Usecase Diagram)
- Aktivitätsdiagramm (Activity Diagram)
- Zeitverlaufdiagramm (Timing Diagram)
- Zustandsdiagramm (Statechart Diagram)
- Kommunikationsdiagramm (Communication Diagram)
- Interaktion-Übersichts-Diagramm (Interaction Overview Diagram)
- Sequenzdiagramm (Sequence Diagram)

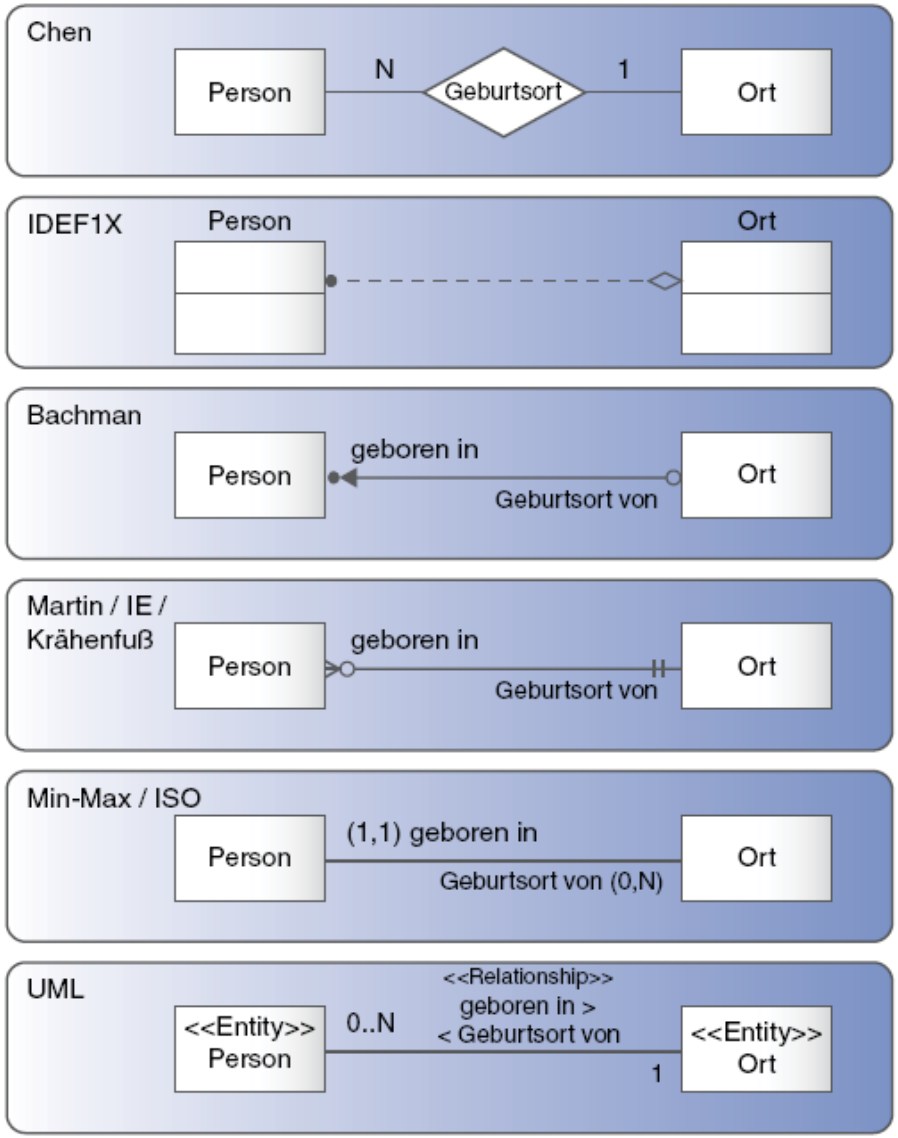
Strukturmodelle

- Klassendiagramm (Class Diagram)
- Paketdiagramm (Package Diagram)
- Komponentendiagram (Component Diagram)
- Verteilungsdiagramm (Deployment Diagram)
- Kompositions-Struktur-Diagramm (Composite Structure Diagram)
- Objektdiagramm (Object Diagram)

Beispiel UML Sequenzdiagramm



(Logische) Datenmodellierung / Informationsmodellierung



- Informationsmodell
- Persistenzmodell
- Häufig als relationales Modell
- Oftmals aber auch XML Dokumente

Anforderungsanalyse

Nicht strukturierte Methoden:

- Die natürliche Sprache ist die häufigste Methode für die Anforderungsanalyse.

Semi-strukturierte Methoden:

- Semi-strukturierte Methoden basieren im Kern auf der natürlichen Sprache, werden aber durch Vorlagen, ein eingeschränktes Vokabular und ergänzende Eigenschaften (Properties) vorgegeben.

Strukturierte Methoden:

- Strukturierte Methoden folgen einer definierten Syntax, Notation oder formalen Spezifikation.

Anwendungsfälle (Use Cases) „Was aber nicht wie“

User Stories/Kunde vor Ort

Geschäftsprozessmodellierung insb.

- Business Processing Modeling Notation (BPMN)

Entwurf und Prototyping der Anwenderschnittstelle

Anwendungsfälle (Use Cases) nach Cockburn 2001

Name und Identifier:

- Benennung von Anwendungsfällen und geordnete Nummerierung

Beschreibung:

- Beschreibung des Anwendungsfalles

Beteiligte Akteure:

- Akteure sind involvierte Personen oder Systeme wie z.B. Benutzer, Kunde, Lieferanten oder ein System. Grundsätzlich werden zwei Arten von Akteuren unterschieden:
  - primäre Akteure, die eigentlichen Benutzer des Systems, und
  - sekundäre Akteure, die das System überwachen, warten und die primären Akteure bei der Zielerreichung unterstützen.

Status:

- Sagt aus, wie weit die Arbeit an dem Anwendungsfall fortgeschritten ist. Beispiele hierfür können sein: in Arbeit, bereit zum Review, im Review, abgelehnt und abgenommen.

Verwendete Anwendungsfälle:

- Wenn ein Anwendungsfall auf andere Anwendungsfälle zurückgreift, werden diese Fälle, mit Name und Identifikation, aufgezählt.

Auslöser:

- Gründe für das Ausführen eines Anwendungsfalles

Anwendungsfälle (Use Cases) nach Cockburn 2001 cont.

Vorbedingungen

- Bedingungen, die erfüllt sein müssen, damit dieser Anwendungsfall ausgeführt werden kann

Invarianten

- Bedingungen, die durch den Anwendungsfall nicht verändert werden dürfen. Diese Bedingungen müssen in einem Misserfolgs- oder Fehlerszenario immer noch erfüllt werden.

Nachbedingung/Ergebnis

- der zu erwartende Zustand nach einem erfolgreichen Durchlauf des Anwendungsfalles.

Standardablauf

- Darstellung eines typischen Szenarios, das leicht zu verstehen oder der am häufigsten vorkommende Fall ist. Am Ende des Anwendungsfalles steht die Zielerreichung des Primärakteurs. Die Ablaufschritte werden nummeriert und häufig als Ablaufpläne dargestellt. Mittels der UML können diese Ablaufpläne in Aktivitätsdiagrammen oder Sequenzdiagrammen dargestellt werden.

Alternative Ablaufschritte

- Szenarien, die neben dem Standardablauf auch auftreten können, unabhängig davon, ob das Ziel erreicht wurde oder nicht. Sie werden häufig als bedingte Verzweigungen der normalen Ablaufschritte dargestellt.

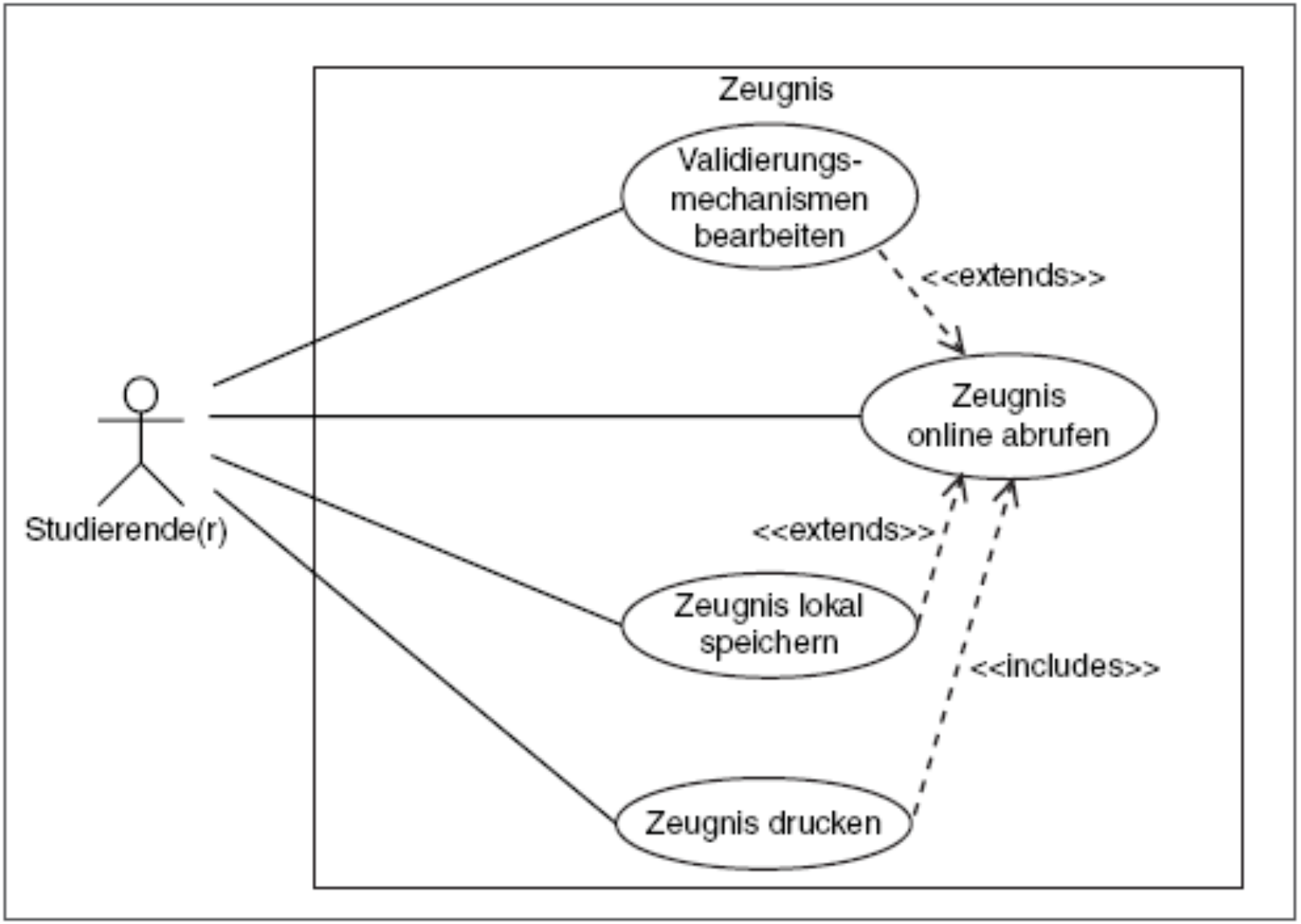
Hinweise

- kurze Beschreibungen und Erklärungen zum besseren Verständnis des Anwendungsfalles, möglicherweise auftretende Nebeneffekte und alles andere, das nicht unter die bereits dargestellten Punkte fällt.

Änderungsgeschichte

- Versionierung des Anwendungsfalles mit Name des Autors, Datum etc.

Beispiel Anwendungsfalldiagramm



Beispiel Anwendungsfallbeschreibung (Ausschnitt)

Name	**Selbstausdruck Zeugnis**
Beschreibung	Studierende können sich Prüfungszeugnisse online abrufen, selbst ausdrucken, lokal speichern und Validierungsmechanismen für Dritte bearbeiten.
Beteiligte Akteure	StudentIn
Status	Abgenommen
Verwendete Anwendungsfälle	Zeugnis online abrufen
Auslöser	StudentIn möchte sich ein Zeugnis ausdrucken.
Vorbedingungen	StudentIn hat eine Prüfung absolviert, die Prüfungsdaten wurden im System erfasst und StudentIn ist im System angemeldet.
Invarianten	Das Zeugnis muss im System gespeichert bleiben.
Ergebnis	Das Zeugnis wurde gedruckt.
Standardablauf	1. StudentIn loggt sich in HISS ein. 2. StudentIn ruft das gewünschte Zeugnis in HISS ab. 3. StudentIn bearbeitet Validierungsmechanismen nicht. 4. StudentIn öffnet das Zeugnis im Browser. 3. StudentIn druckt das Zeugnis.

User Stories/Kunde vor Ort

Vorteile von User Stories:

- User Stories sind kurz und prägnant gehalten und damit gut wartbar.
- User Stories fördern die Diskussion/Abstimmung zwischen Kunde und Entwickler während der gesamten Projektdauer.
- User Stories reduzieren damit das Risiko von Fehlentwicklungen aus Benutzersicht.
- User Stories funktionieren insbesondere in Kombination mit einer Metapher (gemeinsame Sicht auf das System).

Nachteile von User Stories:

- Ohne zugehörige präzisierte Akzeptanz-Testfälle sind User-Stories sehr stark interpretierbar und können ein Risiko für eine erfolgreiche Abnahme darstellen.
- User Stories setzen eine starke Involvierung des Kunden/Benutzers während des gesamten Prozesses voraus. Dies ist jedoch oft unrealistisch.
- User Stories (bzw. die erforderliche Kundeninvolvierung) sind personenzentriert.
- User Stories können kaum als Vertragsgrundlage dienen.

Beispiel Story Card

Datum: 18.09.2009 Aktivität: Neu: X Fix: ___ Verbessern: ___

Story Nr.: 132 Priorität: Benutzer: X Tech.: ___

Referenzen: 30 Risiken: _____

Aufwandsschätzung: 9 Tage

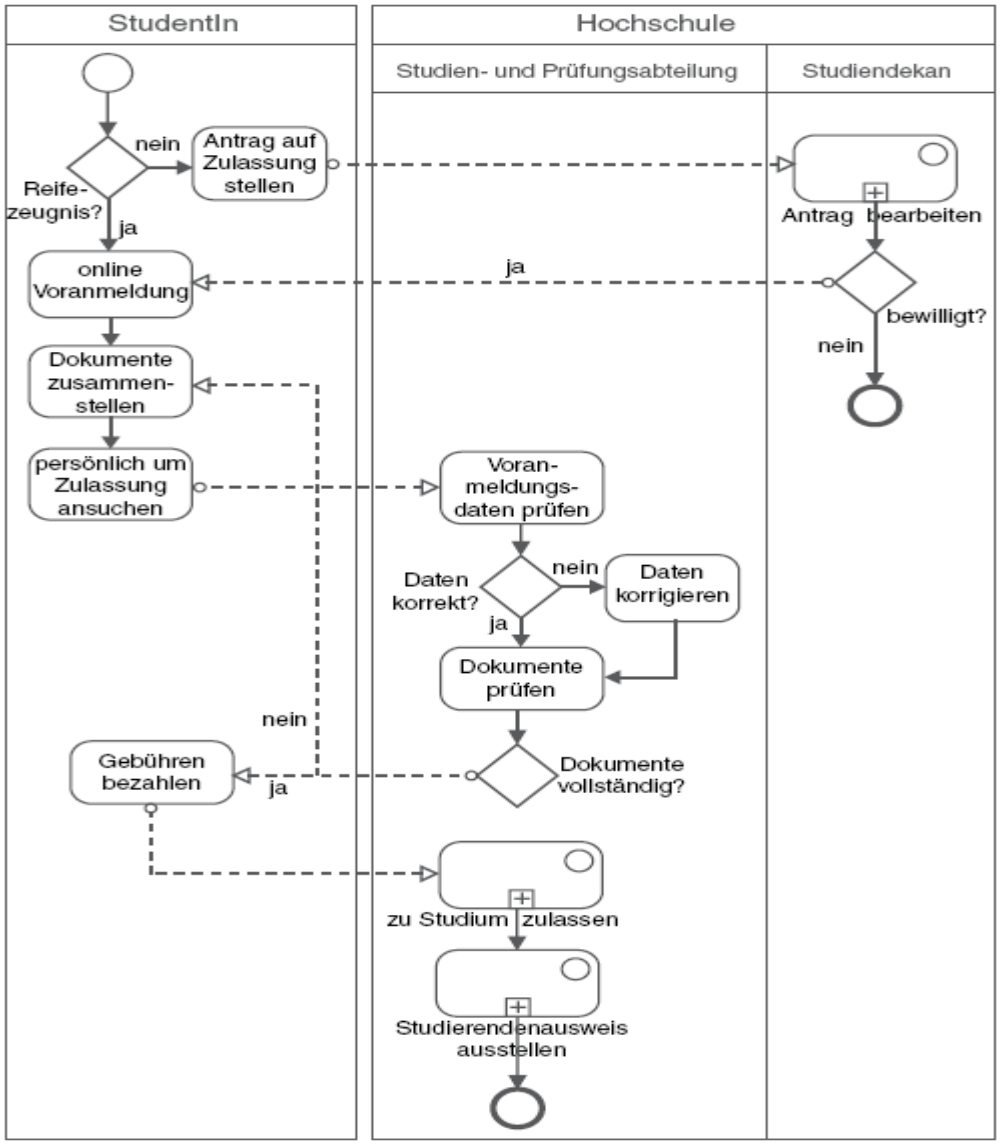
Aufgabenbeschreibung:

Es soll eine Möglichkeit geben, zu einem Termin via E-Mail benachrichtigt zu werden (als Arzt). Dies soll in den persönlichen Einstellungen (de)aktivierbar sein.

Notizen:

Datum	Status	To Do	Kommentar

Geschäftsprozessmodellierung Beispiel in der BPMN



Anforderungsdokumentation

- Spezifikationsdokument
- Projektglossar
- Anforderungsverwaltung
- Nachverfolgbarkeit von Anforderungen (Traceability)

Anforderungsdokument

Offizielles Dokument und Vertragsgrundlage

Wichtige Methode RFC2119

- MUSS (MUST): absolute Festlegung einer Anforderung
- DARF NICHT (MUST NOT): absoluter Ausschluss einer Anforderung
- SOLL (SHOULD): Empfehlung für eine Anforderung. Abweichungen zu diesen Festlegungen sind in begründeten Fällen möglich.
- SOLL NICHT (SHOULD NOT): Empfehlung, eine Anforderung auszuschließen. Abweichungen sind in begründeten Fällen möglich.
- KANN (MAY): Die Anforderung ist optional.

Software Requirement Specification (SRS) nach IEEE830-1998

Einleitung

- Zweck
- Abgrenzung (Systemname, Allgemeine Systembeschreibung, Vorteile und Nutzen)
- Definitionen, Akronyme und Abkürzungen
- Referenzen
- Überblick

Beschreibung

- Produktperspektive: Schnittstellen des Systems, Benutzerschnittstellen, HWSchnittstellen,
- SW-Schnittstellen, Kommunikationsschnittstellen, Speicherumfang
- Produktfunktionen
- Nutzermerkmale
- Rahmenbedingungen
- Voraussetzungen und Abhängigkeiten

Spezifische Anforderungen

- Externe Schnittstellen
- Funktionen
- Anforderungen an die Performanz
- Logische Datenbankanforderungen
- Designvorgaben
- Zu erfüllende Standards
- Systemattribute
  - Zuverlässigkeit
  - Verfügbarkeit
  - Sicherheit
  - Wartung und Pflege
  - Portierbarkeit

Anhang

Projektglossar = gemeinsame Sprache

Administrator

siehe Rollen UsersAndRoles

Anonyme Daten

Anonyme Daten sind solche Daten, für welche sich durch niemanden mit vernünftigen Mitteln ein Personenbezug herstellen lässt. Als vernünftige Mittel sind jene anzusehen, welche nach Art und Aufwand nicht als vollkommen ungewöhnlich anzusehen sind. Dies wäre beispielsweise eine Liste, welche ausschließlich Körpergröße und Gewicht sämtlicher Stellungswerber eines Jahrgangs enthält. Eine Rückführung könnte zB ev. dadurch möglich sein, dass sämtliche Stellungswerber nochmals vorgeladen werden und die Daten nochmals erhoben werden -> jedoch nach Art und Aufwand völlig ungewöhnlich. Über anonymisierte Daten kann frei verfügt werden, da für sie das DSG 2000 gänzlich nicht anwendbar ist.

Attribut

Ein Attribut eines Dokumentationsblattes z.B. ein Textfeld oder eine Dropdownbox

Attributgruppe

Ein Zeile in einer Dokumentation bestehend aus mehreren Attributen

Contributor

siehe Rollen UsersAndRoles

Creator

siehe Rollen UsersAndRoles derzeit nicht verwendet!

Direktpersonenbezogene Daten

sind solche Daten, bei welchen die betreffende Person bestimmt oder ohne nennenswerte Mühe bestimmbar ist. Bestimmt: Udo Jürgens, Schlagerkönig; Bestimmbar: Sieger Wien Marathon 2009. Für diese Art der Daten ist das DSG in vollem Umfang anwendbar.

Dokumentation

Eine Dokumentation besteht aus n Dokumentationsblätter, n Patienten und n Partner.

Dokumentationsblatt

Dokumentationsblatt heißt ab Juli 2009 "Formular"

Formular

Ein Formular, das für ein Patient ausgefüllt wird. Es gibt Einfach-Formulare (wird nur ein mal pro Patient ausgefüllt) und Mehrfach-Formulare (kann pro Patient mehrmals ausgefüllt werden). Ein Formular besteht aus mehreren Attributsgruppen.

GDA (Gesundheitsdienstean bieter)

Auftraggeber und Dienstleister[!! => auch die RISE) gemäß DSG 2000, deren regelmäßige Verwendung von Gesundheitsdaten Bestandteil ihrer Erwerbstätigkeit, ihres Betriebszwecks oder ihres

Anforderungsmanagement-Werkzeug

- Verwaltung der Anforderungen in einer Anforderungsdatenbank
- Versionsmanagement der Anforderungen
- Kollektives Editieren der Anforderungen
- „Festklammern“ (Tagging) von Analyse-Baselines
- Import und Export in Dokumentenform
- Integration von Systemmodellen (z.B. UML-Modelle)
- Selektive Notifikationen der Stakeholder bei Anforderungsänderungen
- Nachverfolgbarkeit (Traceability) von Anforderungen; Darstellung einer Traceability-Matrix
- Integration von weiteren relevanten Werkzeugen wie z.B. einem Testmanagement-Werkzeug
- Verwaltung von Eigenschaften und Metadaten (z.B. die Priorität der Anforderung oder Aufwandsschätzungen)
- Auswertungen der Anforderungen
- Unterstützung für Anforderungs-Reviews und Kommentierungsverfahren
- Geordnete Anforderungswiederverwendung
- Verwaltung eines Projektglossars

Anforderungsnachverfolgbarkeit (Traceability)

Nach dem IEEE- Standard 610.12 (1990) wird Traceability wie folgt definiert:

- *„The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; e.g., the degree to which the requirements and design of a given system element match.“*

Nachverfolgbarkeit der Quelle:

- Zusammenhang zum Stakeholder, der diese Anforderung eingebracht hat. „Wann, wie?“

Nachverfolgbarkeit zu anderen Anforderungen:

- Zusammenhänge zwischen interdependenten Anforderungen

Nachverfolgbarkeit zu abhängigen Artefakten:

- Zusammenhänge der Anforderungen z.B. zu Elementen des Systemdesigns oder auch die Nachverfolgbarkeit zu den Testfällen, die diese Anforderung abdecken

Integration von unterschiedlichen Anforderungsquellen

Use Cases vs. Requirements

- Methoden zur Anforderungserhebung sind komplementär einsetzbar
- In formalen Anforderungslisten fehlt ein Narrativ
- Use Cases sind schwer auf Vollständigkeit von formalen Anforderungen prüfbar
- Mapping von Requirements zu Use Cases in Requirementsmanagement Werkzeug
- N zu N Beziehung zwischen Requirements und Use Cases

Beispiel I: Kunde beauftragt ein Smartcard-Verwaltungssystem, das später sicherheitszertifiziert werden soll

- Anforderungen des Kundens an den Workflow der Verwaltung in Form von Use Cases und Aktivitätsdiagrammen
- Anforderungen der Sicherheitszertifizierung in Form von formalen Anforderungslisten
- Abnahme des Gesamtsystems durch Abnahmetests und formale Reviews

User Stories vs. Use Cases

- Integration von User Stories und Use Cases auf zwei Arten
  - Use Cases verfeinern User Stories
  - User Stories verfeinern Use Cases
- User Stories sind wesentlich informeller als Use Cases, benötigen also meistens Zusatzinformationen

Beispiel II: Pflichtenheft in Form von Use Cases, agile Vorgehensweise im Projekt gewünscht

- Use Cases aus Pflichtenheft werden in Aktivitätsdiagrammen abgebildet
- Aktivitätsdiagramme werden in 1 bis N User Stories (möglichst horizontal) geschnitten und umgesetzt

Zusammenfassung

Vision, Scope und Terminologie müssen klar definiert und abgegrenzt sein.

Alle relevanten Stakeholder müssen identifiziert und qualifiziert eingebunden werden.

Die Machbarkeit muss vor dem formellen Projektbeginn sichergestellt sein.

Alle Komponenten müssen auf „Make or Buy“ evaluiert werden.

Eine Architektur sollte während der Analyse schon grob feststehen und den Stakeholdern qualifiziert dargestellt werden.

Alle Schnittstellen müssen bekannt und geprüft sein.

Alle funktionalen, nichtfunktionalen und Domänenanforderungen müssen identifiziert und qualitativ beschrieben sein.

Anforderungsbeschreibungen müssen für alle Stakeholder (gegebenenfalls über unterschiedliche Sichten) verständlich gemacht werden.

Ein Prozess zum Ändern von Anforderungen (inklusive Re-Costing) muss etabliert sein.

Passende Werkzeuge zum Anforderungsmanagement – insbesondere für die Anforderungsnachverfolgbarkeit – sollten zweckmäßig eingesetzt werden.