Basiswissen Requirements Engineering

Zusammenfassung (Kapitel 1-6)

Patrick Bucher

18.01.2019

Inhaltsverzeichnis

1 Einleitung und Grundlagen

- Wozu Requirements Engineering?
  - Der Umgang mit Anforderungen ist eine signifikante Ursache für Projektfehlschläge.
  - Bessere Kommunikation von Anforderungen reduziert die Anzahl gescheiterter Projekte.
  - Ein grosser Teil der Fehler in Systemprojekten entsteht bereits im Requirements Engineering.
  - Mangelhafte Anforderungen werden durch die Entwickler (unbewusst) vervollständigt und (falsch) interpretiert. Mängel an Anforderungen sind:
    * fehlende oder unklar formulierte Anforderungen
    * Anforderungen widerspiegeln nicht Kundenwünsche und erfüllen nicht dessen Bedürfnisse
    * die Formulierungen von Anforderungen erlauben Interpretationsspielraum
    * Stakeholder verschweigen Sachverhalte, die sie als selbstverständlich erachten
    * Beteiligte haben unterschiedlichen Erfahrungs- und Wissensstand

  - Je später ein Fehler im Entwicklungsprozess behoben wird, desto höher fallen die Kosten für dessen Korrektur aus.
  - Fehlerfreie und vollständige Anforderungen sind die Basis für eine erfolgreiche Systementwicklung.
- Was ist eine Anforderung?
  - Eine dokumentierte Repräsentation einer Bedingung oder Fähigkeit
    * die ein Benutzer zur Lösung eines Problems/zur Erreichung eines Ziels benötigt
    * die ein (Teil)system zur Erfüllung einer Vorgabe (Vertrag, Norm, Spezifikation) benötigt
- Was ist ein Stakeholder?
  - Ein Stakeholder ist direkt oder indirekt von einem Projekt betroffen.
  - Ein Stakeholder hat direkten oder indirekten Einfluss auf die Anforderungen eines Projekts.
- Was ist die Aufgabe des Requirements Engineering?
  - Die Anforderungen der Stakeholder
    * zu ermitteln (detaillieren, verfeinern)
    * zweckmässig zu dokumentieren (natürliche Sprache, Modelle)
    * zu überprüfen und abzustimmen (bestimmten Qualitätskriterien entsprechend)
    * über den ganzen Projektlebenszyklus hinweg zu verwalten (strukturieren, für Rollen aufarbeiten, konsistent ändern und umsetzen)
- Wie ist das Requirements Engineering in verschiedene Vorgehensmodelle eingebettet?
  - Schwergewichtige Vorgehensmodelle (V-Modell, Wasserfallmodell): abgeschlossene, zeitlich befristete erste Phase
  - Leichtgewichtige Vorgehensmodelle (eXtreme Programming, Scrum): kontinuierlicher, phasenübergreifender Prozess
- Welche Rolle spielt die Kommunikation beim Requirements Engineering?
  - Der Austausch von Informationen benötigt eine gemeinsame Sprache (Glossar, formale Beschreibungssprache wie UML).
  - Vereinfachungen im sprachlichen Ausdruck führen zu Ungenauigkeiten und ergeben dadurch Interpretationsspielraum.
- Wie sieht die Rolle des Requirements Engineers aus?
  - Der Requirements Engineer pflegt meist als Einziger direkten Kontakt zu allen Stakeholdern.
  - Er muss sich in die verschiedenen Fachgebiete der Stakeholder einarbeiten, um die Bedürfnisse hinter den Aussagen der Stakeholder zu erkennen.
  - Er dient als Übersetzer zwischen dem Fachgebiet und den Entwicklern/Architekten.
- Welche Fähigkeiten braucht ein Requirements Engineer?
  1. Analytisches Denken: schnelles Verstehen unbekannter Fachgebiete
  2. Empathie: erkennen, was der Stakeholder tatsächlich benötigt
  3. Kommunikationsfähigkeit: zuhören, die richtigen Fragen stellen und nachfragen
  4. Konfliktlösungsfähigkeit: Konflikte erkennen, zwischen Parteien vermitteln und

den Konflikt auflösen
5. Moderationsfähigkeit: zwischen verschiedenen Meinungen vermitteln, Diskussionen leiten
6. Selbstbewusstsein: mit Kritik umgehen können, selbstsicher auftreten
7. Überzeugungsfähigkeit: Anforderungen der Stakeholder verteidigen, Konsens unter Stakeholdern herstellen und Entscheidungen herbeiführen

- Welche Arten von Anforderungen gibt es?
  1. Funktionale Anforderungen: geforderte Funktion, die ein System anbieten soll
     - Beispiel: Zu jeder Person sollen zwei Adressen erfasst werden können.
  2. Qualitätsanforderung: betrifft Performanz, Verfügbarkeit, Zuverlässigkeit, Skalierbarkeit, Portabilität
     - grosser Einfluss auf die Systemarchitektur
     - auch als «nicht funktionale Anforderung» bezeichnet
     - werden häufig unzureichend dokumentiert
     - sollen möglichst frühzeitig ermittelt, dokumentiert und unter Stakeholdern abgestimmt werden
     - sollen möglichst objektiv überprüfbar und mit quantitativen Angaben konkretisiert sein
     - können durch funktionale Anforderungen konkretisiert werden (z.B. Forderung nach erneuter Passworteingabe bei bestimmten Aktionen zur Erhöhung der Sicherheit)
     - Beziehung zu funktionalen Anforderungen sollten explizit festgehalten werden.
     - Beispiel: 95% aller Anfragen sollen in weniger als einer Sekunde beantwortet werden können.
  3. Randbedingungen/Rahmenbedingungen: von Projektbeteiligten nicht beeinflussbare Gegebenheiten (Systemlandschaft, Freigabetermin, Budget, Projektpersonal)
     - werden nicht umgesetzt, müssen aber berücksichtigt werden
     - schränken die Umsetzungsmöglichkeiten ein
     - Beispiel: Das Projekt muss innert drei Monaten abgeschlossen sein.
- Welche Arten von Qualitätsanforderungen gibt es?
  1. Performanz: Antwortzeiten, Ressourcenverbrauch
  2. Sicherheit: Nachweisbarkeit, Authentizität, Vertraulichkeit, Integrität
  3. Zuverlässigkeit: Verfügbarkeit, Fehlertoleranz, Wiederherstellbarkeit
  4. Benutzbarkeit: Barrierefreiheit, Erlernbarkeit, Bedienbarkeit
  5. Wartbarkeit: Wiederverwendbarkeit, Analysierbarkeit, Modifizierbarkeit, Prüfbarkeit
  6. Übertragbarkeit: Anpassbarkeit, Installierbarkeit, Austauschbarkeit
- Was ist das Ziel des Requirements Engineering?
  - Das vollständige Dokumentieren von Kundenanforderungen in guter Qualität.
  - Das frühzeitige Erkennen und beheben von Fehlern.

2 System und Systemkontext abgrenzen

- Was ist ein Systemkontext?
  - Ausschnitt der Realität
  - für die Anforderungen des Systems massgebend
  - und für das Verständnis derselben relevant
- Aus welchen Aspekten besteht ein Systemkontext?
  - Personen: Stakeholder und -gruppen
  - Systeme im Betrieb: Hardware und Software
  - Prozesse: technische und Geschäftsprozesse
  - Ereignisse: technische oder physische
  - Dokumente: Gesetze, Standards, Dokumentationen
- Warum ist der Systemkontext kritisch für den Projekterfolg?
  - Ein falscher oder unvollständig berücksichtigter Systemkontext führt zu fehlerhaften Anforderungen.
  - Ein unter fehlerhaften Anforderungen entwickeltes System kann im Betrieb versagen.
  - Anforderungen sind immer in einem bestimmten Kontext definiert und können nur innerhalb dieses Kontexts richtig interpretiert werden.
- Wie erfolgt die System- und Kontextabgrenzung?
  - Systemabgrenzung: welche Aspekte müssen durch das geplante System abgedeckt werden?
    - grenzt den durch das Projekt veränderbaren Teil von unveränderbaren Aspekten der Umgebung ab
    - legt den Scope (Projektumfang) fest
    - ermöglicht die Identifikation von Schnittstellen (Quellen und Senken ermitteln; werden zu Eingabe- und Ausgabeschnittstellen)
  - Kontextabgrenzung: welche Aspekte in der Systemumgebung sind für dieses relevant?
    - Grenze zwischen relevantem Kontext zur irrelevanten Umgebung
    - grenzt den für das Projekt relevanten Teil von der Umgebung (Einfluss auf das System) vom irrelevanten Teil der Umgebung (kein Einfluss auf das System) ab
  - Graubereich: Die Grenzen verlaufen zeitweise unscharf.
    - Systemabgrenzung: die Grenze kann oft erst gegen Ende des Requirements-Engineering-Prozesses präzise festgelegt werden.
    - Kontextabgrenzung: der Graubereich muss nicht vollständig aufgelöst werden können.

3 Anforderungen ermitteln

- Welche Anforderungsquellen gibt es?
  - Stakeholder: Person mit direktem oder indirektem Einfluss auf Anforderungen

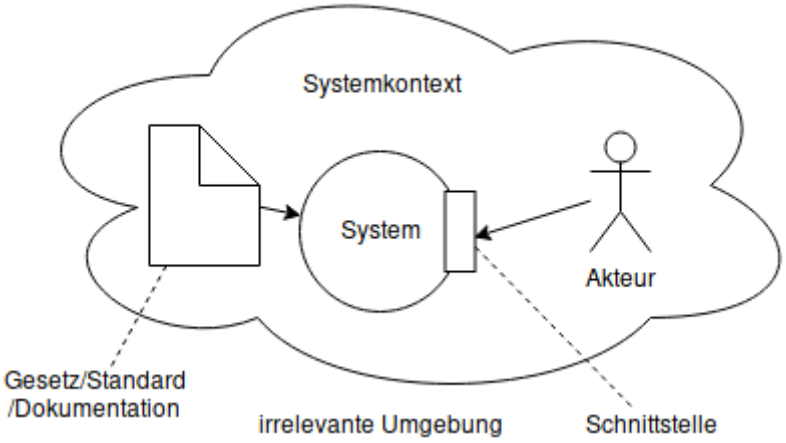

Abbildung 1: Kontextdiagramm (Beispiel)

    - Nutzer, Betreiber, Entwickler, Architekten, Auftraggeber, Tester
  - Dokumente: Normen, Standards, Gesetzestexte; Anforderungsdokumente, Fehlerberichte des alten Systems
  - Systeme im Betrieb: alte und Vorgängersysteme, Konkurrenzsysteme
    - Eindruck durch Ausprobieren erhalten
- Was ist im Umgang mit Stakeholdern im Projekt zu beachten?
  - Geeignete Stakeholder müssen zunächst selektiert werden.
  - Stakeholder sollen dokumentiert werden.
    - Name, Funktion/Rolle, Kontaktdaten, Verfügbarkeit, Relevanz, Wissensgebiet, projektbezogene Ziele und Interessen
  - Durch deren Integration ins Projektgeschehen werden Projektbetroffene zu Projektbeteiligten.
  - Stakeholder können sich einem Projekt entgegenstellen, müssen vom Projekt überzeugt werden.
    - Gründe: Motivationsmangel, Zufriedenheit mit Ausgangszustand, Angst vor Veränderungen, negative Erfahrungen mit früheren Projekten
  - Vereinbarung mit Stakeholdern betreffend Aufgaben, Verantwortungsbereiche, Weisungsbefugnisse, Ziele, Kommunikationswege sind zu treffen.
- Welche Rechte und Pflichten hat der Requirements Engineer?
  - die Sprache der Stakeholder sprechen
  - sich in das Fachgebiet einarbeiten
  - das Anforderungsdokument erstellen
  - die Arbeitsergebnisse verständlich machen
  - respektvollen Umgang mit Stakeholdern pflegen
  - Ideen, Alternativen und deren Realisierung präsentieren
  - den Ansprüchen der Stakeholder entsprechendes System spezifizieren
- Welche Rechte und Pflichten hat der Stakeholder?

  - den Requirements Engineer in das Fachgebiet einführen
  - den Requirements Engineer mit Anforderungen versorgen
  - Anforderungen zielgerecht und gewissenhaft formulieren
  - Entscheidungen zeitgerecht treffen
  - Einschätzungen des Requirements Engineers bzgl. Kosten und Machbarkeit respektieren
  - Anforderungen priorisieren
  - Anforderungen des Requirements Engineers überprüfen
  - Anforderungsänderungen kommunizieren
  - Änderungsprozess befolgen
- Was leistet das Kano-Modell?
  - Teilt Produktmerkmale in drei Kategorien ein:
    1. Basisfaktoren: selbstverständlich, vorausgesetzt (unterbewusst)
       * müssen vollständig erfüllt werden
       * mittels Beobachtungstechniken ermittelbar
    2. Leistungsfaktoren: explizit gefordert (bewusst)
       * Erfüllung erhöht Zufriedenheit der Stakeholder
       * mittels Befragungstechniken ermittelbar
    3. Begeisterungsfaktoren: angenehme Überraschungen (unbewusst)
       * werden durch eigenes Ausprobieren erkannt
       * mittels Kreativitätstechniken ermittelbar
- Welche Ermittlungstechniken gibt es?
  - Befragungstechniken: möglichst genaue und unverfälschte Aussage zu Anforderungen von Stakeholdern erfahren
    * Interview: vorgegebene Fragen, Gesprächsverlauf anpassbar
      · vollständige Antworten durch gezieltes Nachfragen
    * Fragebogen: offene oder geschlossene Fragen
      · Abfragen, was der Requirements Engineer bereits kennt/vermutet
  - Kreativitätstechniken: innovative Anforderungen und Begeisterungsfaktoren ermitteln, Vision festlegen
    * Brainstorming: beurteilungsloses Sammeln von Ideen in einer Gruppe mit anschliessender Analyse
    * Brainstorming paradox: Brainstorming zum Sammeln negativer Anforderungen (was nicht eintreten soll) um Risiken zu erkennen und Gegenmassnahmen zu entwickeln
    * Perspektivwechsel: Stakeholder dazu bringen neue Sichtweisen einzunehmen
    * Analogiedenken: Suchen von analogen Strukturen innerhalb (Bionik) oder ausserhalb (Bisoziation) der Natur
  - Dokumentzentrierte Techniken: Lösungen und Erfahrungen bestehender Systeme wiederverwenden
    * Systemarchäoloie: Analyse des bestehenden Systems zur Herausarbeitung der tatsächlichen Fachlogik (aufwändig)
    * Perspektivenbasiertes Lesen: Dokumente aus einer bestimmten Perspektive

lesen
    * Wiederverwendung: Bereits erhobene Anforderungen von bestehenden Systemen wiederverwenden
  - Beobachtungstechniken: bei Mangel an Fähigkeiten und Zeit zur Weitergabe von Wissen an den Requirements Engineer
    * Feldbeobachtung: Beobachten und Dokumentieren der stattfindenden Prozesse
      · bewusste Demonstration oder blosse Beobachtung des Arbeitsvorgangs
      · Hinterfragen bestehender Abläufe
      · Identifikation ineffizienter Prozesse
      · Herausarbeiten von Basisfaktoren
      · Video- und Audioaufzeichnung zur späteren Analyse möglich
    * Apprenticing: Erlernen und Ausführen von Tätigkeiten durch den Stakeholder
  - Unterstützende Techniken
    * Mindmapping: grafische Darstellungen von (hierarchischen) Beziehungen und Abhängigkeiten zwischen Begriffen
    * Workshops: Zusammentreffen von Requirements Engineer mit Stakeholdern
    * CRC-Karten: Notieren von Kontextaspekten mit Eigenschaften und Beziehungen
    * Audio- und Videoaufzeichnungen: Beobachtung von Prozessabläufen reproduzierbar machen
    * Use-Case-Modellierung: Aussensicht des zu erstellenden Systems dokumentieren
    * Prototypen: Hinterfragen von erarbeiteten Anforderungen, Vorstellungen überprüfen und konkretisieren

4 Anforderungen dokumentieren

- Was ist eine Anforderungsspezifikation?
  - eine systematisch dargestellte, vorgegebenen Kriterien genügende Sammlung von Anforderungen
- Welche Rolle spielt die Dokumentation in einem Projekt?
  - unterstützende Funktion bei der Kommunikation
  - Festhalten der Sachverhalte in einer für alle Adressaten genügenden Qualität
- Wozu werden Anforderungen dokumentiert?
  - Anforderungen als Basis der Systementwicklung: Anforderungen wirken sich direkt oder indirekt auf das Projekt und dessen Erfolg aus.
  - Rechtliche Relevanz: Anforderungen sind rechtlich verbindlich und helfen bei der Klärung rechtlicher Konflikte.
  - Komplexität: Anforderungen werden dokumentiert, um den Überblick über komplexe Systeme behalten zu können.

    - Zugreifbarkeit: Eine für alle Beteiligten verfügbare Dokumentation ermöglicht diesen eine schnelle Einarbeitung.
- Aus welchen Perspektiven können Anforderungen dokumentiert werden?
    1. Strukturperspektive: statisch-strukturelle Perspektive auf das System
        - Struktur von Ein- und Ausgabedaten
        - Nutzungs- und Abhängigkeitsbeziehungen (zu Diensten externer Systeme)
    2. Funktionsperspektive: Informationen/Daten des Systems
        - die im Systemkontext manipuliert werden
        - die in den Systemkontext einfliessen
    3. Verhaltensperspektive: zustandsorientierte Reaktion des Systems auf Ereignisse
        - Bedingungen für Zustandswechsel
        - Effekte des Systems auf den übrigen Systemkontext
- Welche Arten von Dokumentation gibt es?
    - natürliche Sprache (Prosa)
        * + vielseitig einsetzbar
        * + funktioniert für alle Arten von Anforderungen
        * + Notation allen Stakeholdern von Anfang an bekannt
        * - mehrdeutig
        * - Vermischung der Perspektiven
    - konzeptuelle Modelle
        * + isolierte Betrachtung der Perspektiven
        * + Kompaktheit
        * + für geübten Leser verständlicher
        * + Eindeutigkeit
        * - setzt spezifische Modellierungskenntnisse voraus
    - zweckmässige Kombinationen der beiden
        * Nachteile der einen Form durch Vorteile der anderen Form verringern
        * Beispiel: natürlichsprachliche Kommentare in UML-Diagrammen
- Was sind die wichtigsten Diagramme zur Dokumentation?
    - Use-Case-Diagramm: Überblick über ein System
        * zur Verfügung gestellte Funktionen
        * Verbindungen zu externen Interaktionspartnern
    - Klassendiagramm: statische Struktur von Daten
        * dokumentiert Abhängigkeiten zum Systemkontext
        * stellt komplexe Beriffsysteme von Fachgebieten strukturiert dar
    - Aktivitätsdiagramm: Geschäftsprozesse und andere Abläufe
        * dokumentiert Ablauflogik eines Use Case
        * spezifiziert Verarbeitungslogik einzelner Operationen
    - Zustandsdiagramm: ereignisgesteuertes Verhalten
        * Zustände eines Systems
        * Ereignisse und Bedingungen für Zustandsübergänge
        * Effekte auf den Systemkontext
- Wie kann eine Dokumentation strukturiert werden?
    - Standardisierte Dokumentstrukturen

        * erleichtern die Einarbeitung
        * ermöglichen eine schnelle Erfassung von Inhalten
        * ermöglichen selektives Lesen/Überprüfen
        * ermöglichen automatisches Prüfen (auf Vollständigkeit)
        * ermöglichen einfache Wiederverwendung von Inhalten
    - Angepasste Standardinhalte
        * Adaption standardisierter Dokumentstrukturen
        * erfordern minimale Inhalte
- Was sind die wichtigsten standardisierten Dokumentstrukturen?
    - Rational Unified Process (RUP): für objektorientierte Methoden entwickelt
        * Auftraggeber erstellt Business Model (Geschäftsregeln, -anwendungsfälle, -ziele)
        * Auftragnehmer erstellt Software Requirements Specification (Dokumentation der Softwareanforderungen)
    - Standard ISO/IEC/IEEE 29148:2011: für Dokumentation von Softwareanforderungen, Kapitel:
        * einführende Informationen (Zweck, Abgrenzung)
        * Auflistung referenzierter Dokumente
        * spezifische Anforderungen (funktionale, nicht-funktionale)
        * geplante Verifikationsmassnahmen
        * Anhänge (getroffene Annahmen, identifizierte Abhängigkeiten)
    - V-Modell
        * Auftraggeber erstellt Lastenheft (Gesamtheit der Forderungen zu Lieferung und Leistungen: «wozu»; Forderungen aus Anwendersicht)
        * Auftragnehmer erstellt Pflichtenheft auf Basis des Lastenhefts (Konkretisierung des Lastenhefts: Realisierungsvorgaben)
- Was sind die minimalen Inhalte für angepasste Dokumentstrukturen?
    - Einleitung: dokumentübergreifende Informationen zur Gewährung eines Überblicks
        * Zweck: «warum?», Zielgruppe, Leserkreis
        * Systemumfang: Name, Vorteile und Ziele des zu entwickelnden Systems
        * Stakeholder: Auflistung mit relevanten Informationen
        * Definitionen, Akronyme, Abkürzungen
        * Referenzen: auf andere Dokumente
        * Übersicht: Erläuterung weiterer Inhalte und Struktur/Aufbau des Anforderungsdokuments
    - Allgemeine Übersicht: Verständnis der Anforderungen erhöhen
        * Systemumfeld: System- und Kontextabgrenzung
        * Architekturbeschreibungen: Schnittstellen, Beschränkungen
        * Systemfunktionalität: grobe Funktionalität (Use Cases)
        * Nutzer und Zielgruppen
        * Randbedingungen: mögliche Beeinträchtigungen
        * Annahmen: Annahmen über den Systemkontext, auf denen die Anforderungen beruhen

  - Anforderungen: funktionale und nicht-funktionale
  - Anhang: weiterführende Unterlagen, Hintergrundinformationen
  - Index: Inhaltsverzeichnis, nachgetragener(!) Index
- Wozu werden Anforderungsdokumente verwendet?
  - grundsätzlich: als Grundlage für verschiedene Projektaufgaben
  - Planung: Ableitung von Arbeitspaketen und Meilensteinen
  - Architekturentwurf: auf Basis detaillierter Anforderungen und Randbedingungen
  - Implementierung: auf Basis des Architekturentwurfs
  - Test: Testfälle zur Überprüfung des Systems
  - Änderungsmanagement: Ausmass von Änderungen abschätzen
  - Systemnutzung und Systemwartung: Art des Fehlers (Implementierung, Bedienung) analysieren
  - Vertragsmanagement: Anforderungsdokument als Vertragsgegenstand
- Welche Qualitätskriterien gelten für das Anforderungsdokument?
  - Eindeutigkeit und Konsistenz: Anforderungen eindeutig und untereinander widerspruchsfrei
  - Klare Struktur:
    * grosser Umfang erfordert gute Struktur
    * gute Struktur erlaubt selektives Lesen
  - Modifizierbarkeit und Erweiterbarkeit:
    * Anforderungen müssen geändert, hinzugefügt und entfernt werden können
    * Anforderungsdokument sollte der Versionsverwaltung des Projekts unterliegen
  - Vollständigkeit:
    * alle Anforderungen mit allen relevanten Informationen (Eingaben, Ereignisse, Reaktionen, Fehler- und Ausnahmefälle) dokumentiert
    * formale Gesichtspunkte: Beschriftung von Grafiken, Diagrammen, Tabellen; Quellen- und Abkürzungsverzeichnisse
  - Verfolgbarkeit (Traceability): zwischen Anforderungsdokument und anderen Dokumenten
- Welche Qualitätskriterien gelten für einzelne Anforderungen?
  - abgestimmt: für alle Stakeholder korrekt und notwendig
  - eindeutig: Interpretationsspielraum ausgeschlossen
  - notwendig: von allen Stakeholdern akzeptiert, die Gegebenheiten im Systemkontext widerspiegelnd
  - konsistent: in sich widerspruchsfrei
  - prüfbar: durch Test oder Messung nachweisbar
  - realisierbar: mit gegebenen organisatorischen, rechtlichen, technischen und finanziellen Rahmenbedingungen
  - verfolgbar:
    * Ursprung und Beziehungen zu anderen Dokumenten nachvollziebar
    * eindeutige Identifikation vorhanden und konsequent genutzt
  - vollständig: Anforderung ist vollständig beschrieben oder als unvollständig

(«tbd»: «to be determined») markiert (suchbarer Text, Statusfeld)
    - verständlich: für alle Stakeholder
- Was sollte ein Glossar beinhalten?
    - generell: Definition einer konsistenten projektspezifischen und/oder projektübergreifenden Terminologie zur Vermeidung von Missverständnissen
    - Kontextspezifische Fachbegriffe
    - Abkürzungen und Akronyme
    - Alltägliche Begriffe mit spezieller Bedeutung im Projektkontext
    - Synonyme (verschiedene Begriffe gleicher Bedeutung)
    - Homonyme (Begriff mit verschiedenen Bedeutungen)
- Welche Anforderungen sollte ein Glossar erfüllen?
    - zentrale Verwaltung des Glossars: genau ein verbindlich gültiges Glossar
    - Verantwortlichkeit: eine konkrete Person
    - projektübergreifend gepflegt
    - allgemein zugänglich
    - Herkunft von Begriffen ausgewiesen
    - unter allen Stakeholdern abgestimmt
    - einheitlich strukturiert (mittel Schablone)

5 Anforderungen natürlichsprachlich dokumentieren

- Welche Bedeutung haben Transformationseffekte bei der Wahrnehmung und Darstellung auf das Requirements Engineering?
    - Der Requirements Engineer kann aus der Oberflächenstruktur (formulierte Anforderungen) durch gezieltes Nachfragen die Tiefenstruktur (tatsächlich Gemeintes) ermitteln.
- Welche Transformationsprozesse treten beim Formulieren von Anforderungen auf?
    1. Nominalisierung
        - umfassender Prozess wird unter Weglassung von Details zu einem Ereignis gemacht
        - Beispiel: «die Eingabe» statt «eingeben»
        - Ausnahmen und Ein-/Ausgabeparameter müssen ausreichend definiert sein
    2. Substantive ohne Bezugsindex
        - Begriffe sind unvollständig spezifiziert
        - Beispiel: «die Daten», «der Anwender»
        - Substantive müssen um weitere Informationen ergänzt werden
    3. Universalquantoren
        - Aussagen werden über sämtliche Objekte einer Menge gemacht
        - Beispiel: «nie», «immer», «alle», «keine»
        - Aussagen müssen hinterfragt und Ausnahmen definiert werden
    4. Unvollständig spezifizierte Bedingungen
        - bei Bedingungen wird nur der Positivfall definiert
        - Beispiel: «wenn …, dann …»

        - Negativfall («sonst») muss spezifiziert werden
    5. Unvollständig spezifierte Prozesswörter
        - bei Verben fehlen Subjekt und Objekte
        - Beispiel: «die Daten werden übertragen»
        - Subjekt (mittels Aktivformulierung) und weitere Parameter («von wo», «wohin» beim Verb «übertragen») müssen definiert werden
- Wozu dienen Satzschablonen/Requirement Templates?
    - Zur Reduzierung obengenannter sprachlicher Effekte
    - Zur Erreichung syntaktisch eindeutig formulierter Anforderungen
- Wie muss eine Satzschablone aufgebaut sein?
    1. Verbindlichkeit festlegen: bindend, empfohlen, wünschenswert?
        - Modalverben: «müssen», «sollen», «können»
    2. Kern der Anforderung bestimmen: geforderte Funktionalität (Prozess) bezeichnen
        - Vorgänge oder Tätigkeiten: ausschliesslich Verben
    3. Aktivität des Systems charakterisieren
        1. selbständige Systemtätigkeit: «Das System muss *Prozesswort* …»
        2. Benutzerinteraktion: «Das System muss *wem?* die Möglichkeit bieten *Prozesswort* …»
        3. Schnittstellenanforderung: «Das System muss fähig sein *Prozesswort* …»
    4. Objekte einfügen
        - Prozesswort um Objekte ergänzen: was, wie, wo etc.
    5. Logische und zeitliche Bedingungen formulieren
        - «sobald» für zeitliche Bedingungen
        - «falls» für logische Bedingungen
        - «wenn» zu vermeiden, da es eine logische *und* zeitliche Bedeutung hat
        - Qualitätsanforderungen mittels Nebensatz an den Anfang der Anforderung stellen
- Was ist bei der Arbeit mit Satzschablonen zu beachten?
    - Satzschablonen normieren das Vorgehen und schränken den Stil ein.
    - Satzschablonen sollten nicht als Methode vorgeschrieben sondern als Hilfsmittel angesehen werden.

6 Anforderungen modellbasiert dokumentieren

- Welche drei Ausprägungen von Anforderungen gibt es?
    1. Ziele: beschreiben Intentionen von Stakeholdern
    2. Use Cases und Szenarien: dokumentieren beispielhafte Abläufe der Systemnutzung
    3. Systemanforderungen: beschreiben detaillierte Funktionalitäten und Qualitäten des zu entwickelnden Systems
- Was ist ein Modell?
    - Ein Modell ist ein abstrahiertes Abbild

        1. einer existierenden Realität oder
        2. ein Vorbild für eine zu schaffende Realität.
- Welche drei wesentliche Eigenschaften haben Modelle?
    1. Abbild der Realität
        - deskriptiv: Aspekte einer bestehenden Realität abbilden
        - präskriptiv: Aspekte einer zu schaffenden Realität abbilden
    2. Verkürzung der Realität
        - Selektion: Aspekte auswählen
        - Verdichtung: Aspekte zusammenfassen
    3. Pragmatische Eigenschaft
        - in spezifischem Verwendungskontext
        - für spezifischen Verwendungszweck
- Wie ist eine konzeptuelle Modellierungssprache definiert?
    - Syntax: Modellelemente und deren gültigen Kombinationen
    - Semantik: Bedeutung der Modellelemente
- Was ist ein Ziel?
    - die intentionale Beschreibung eines von Stakeholdern erwünschten charakteristischen Merkmals des zu entwickelnden Systems
- Welche Abschnitte gehören zu einer Use-Case-Spezifikation?
    1. Bezeichner
    2. Name
    3. Autoren
    4. Priorität
    5. Kritikalität (bezüglich Schadensausmass bei Fehlverhalten)
    6. Quelle (Stakeholder, Dokument, System)
    7. Verantwortlicher (Stakeholder)
    8. Beschreibung (zusammengefasst)
    9. Auslösendes Ereignis
    10. Akteure
    11. Vorbedingungen
    12. Nachbedingungen (für Hauptszenario)
    13. Ergebnis (erzeugte Ausgaben)
    14. Hauptszenario
    15. Alternativszenarien
    16. Ausnahmeszenarien
    17. Qualitäten
- Welche Modelle eignen sich für welche Perspektive einer Anforderung?
    - Strukturperspektive: Klassendiagramme, ER-Diagramme
    - Funktionsperspektive: Aktivitätsdiagramm
    - Verhaltensperspektive: Zustandsdiagramm, Datenflussdiagramm, Statechart

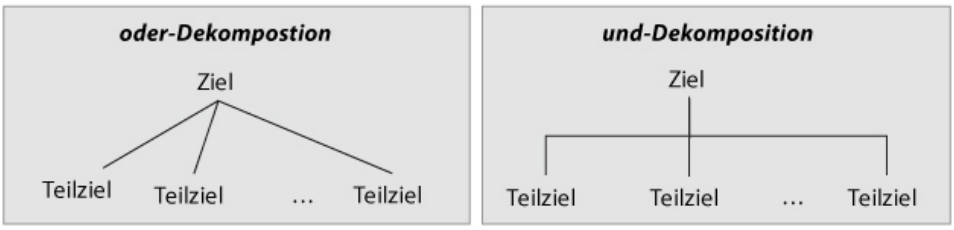

Abbildung 2: Zielmodell (Und/Oder)

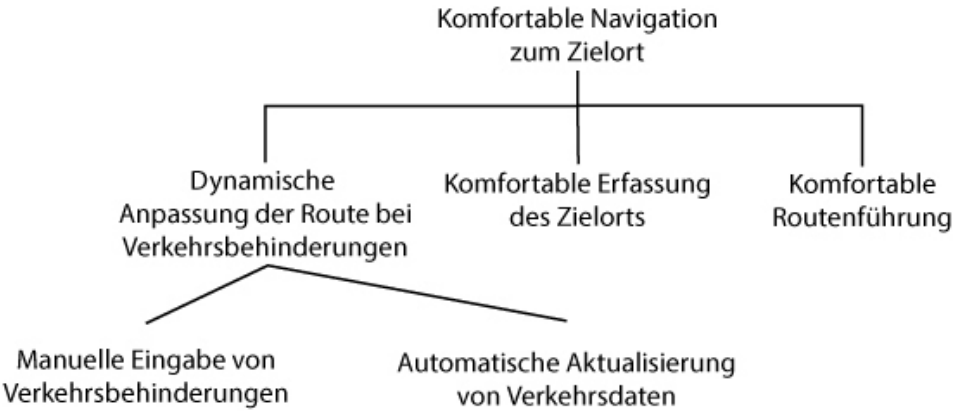

Abbildung 3: Zielmodell (Beispiel)

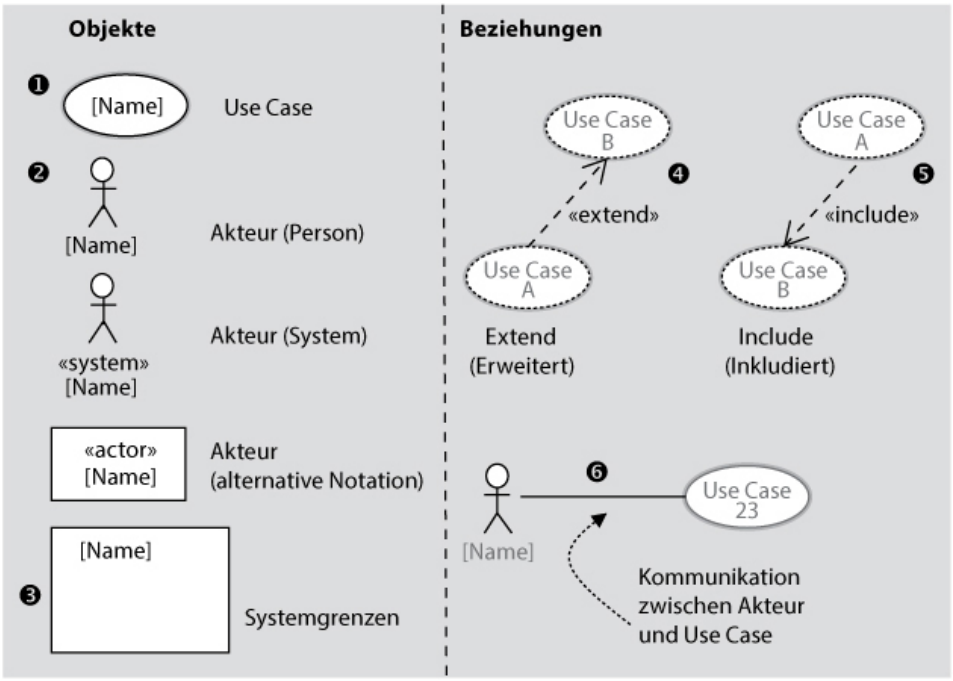

Abbildung 4: Use Case (Elemente)

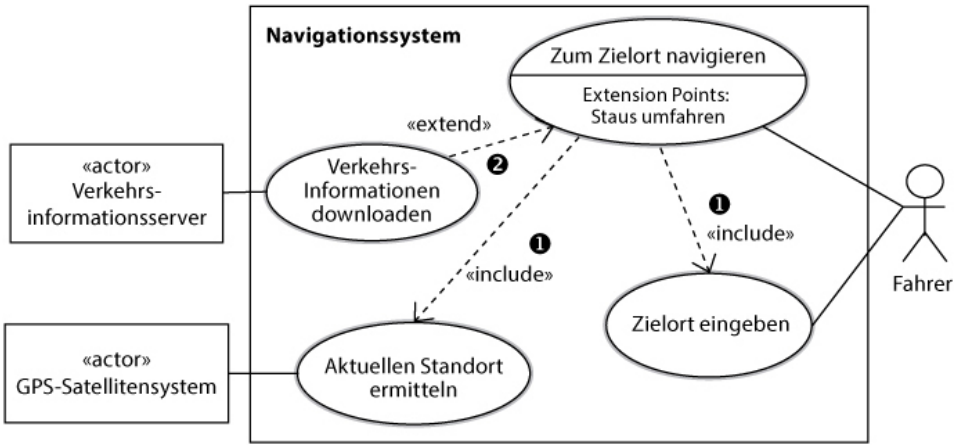

Abbildung 5: Use Case (Beispiel)

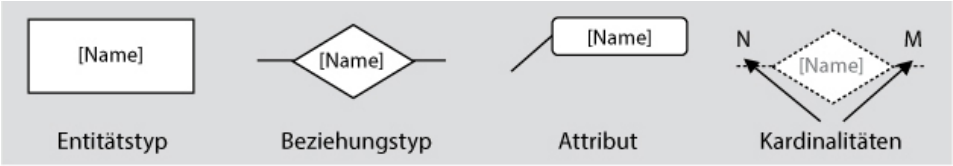

Abbildung 6: ER-Diagramm (Elemente)

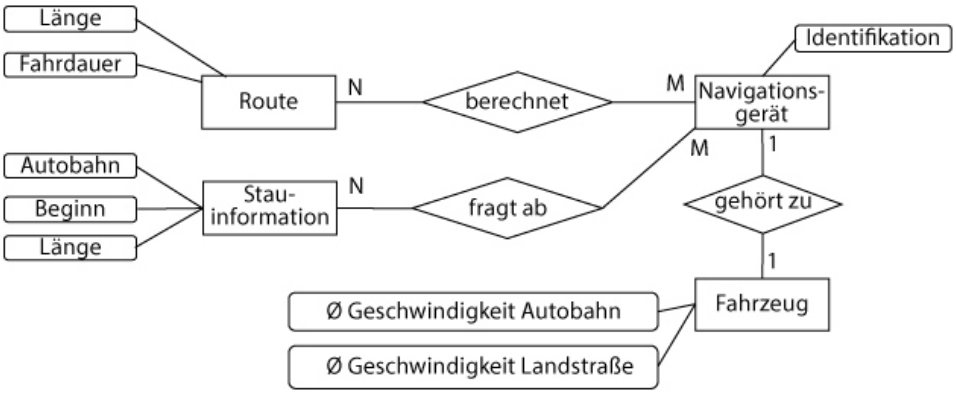

Abbildung 7: ER-Diagramm (Beispiel)

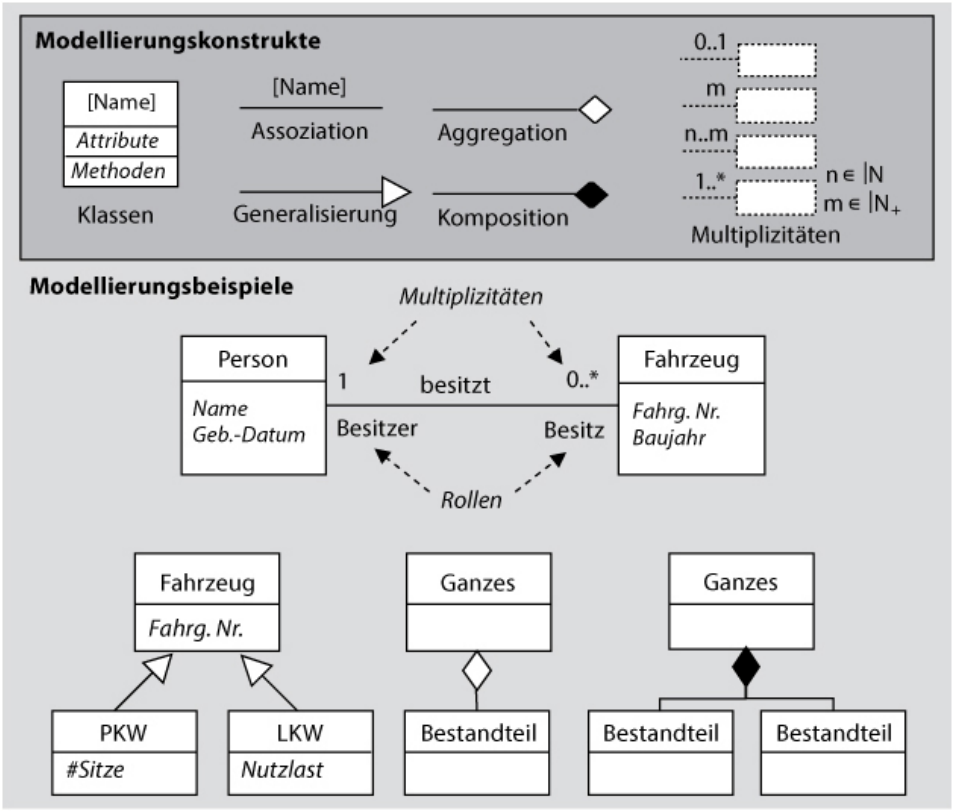

Abbildung 8: Klassendiagramm (Elemente)

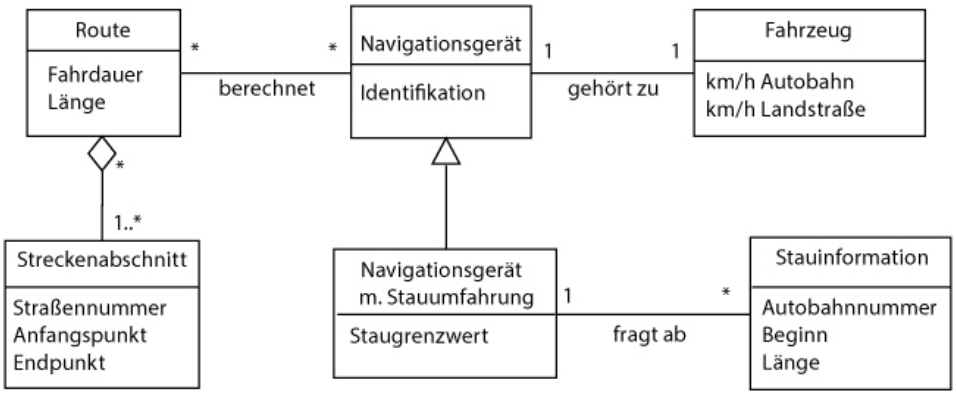

Abbildung 9: Klassendiagramm (Beispiel)

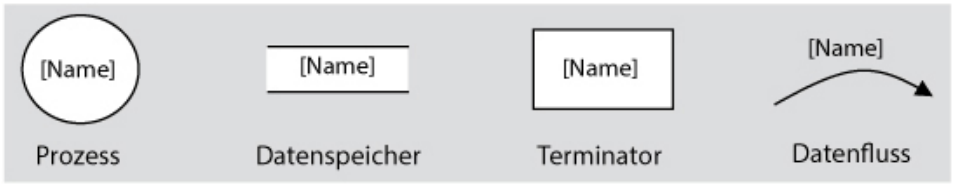

Abbildung 10: Datenflussdiagramm (Elemente)

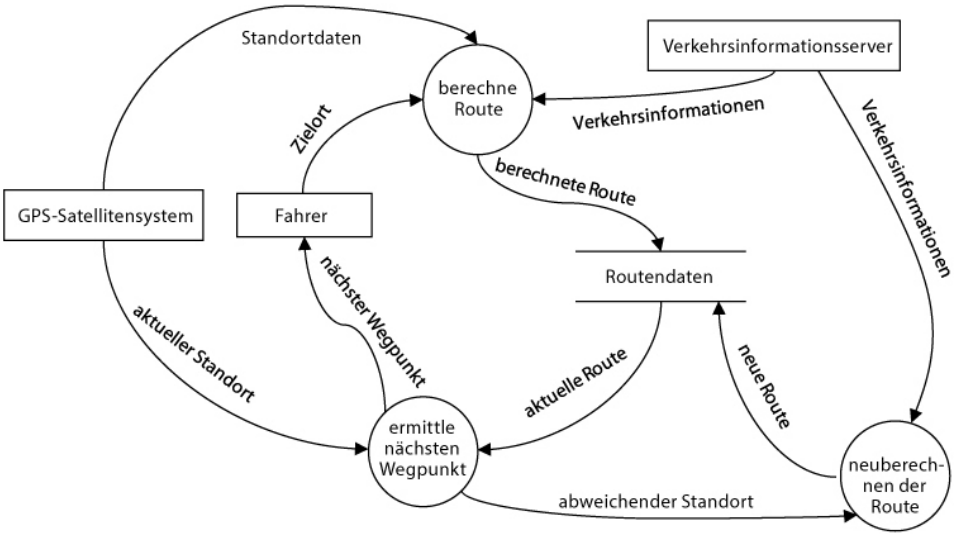

Abbildung 11: Datenflussdiagramm (Beispiel)

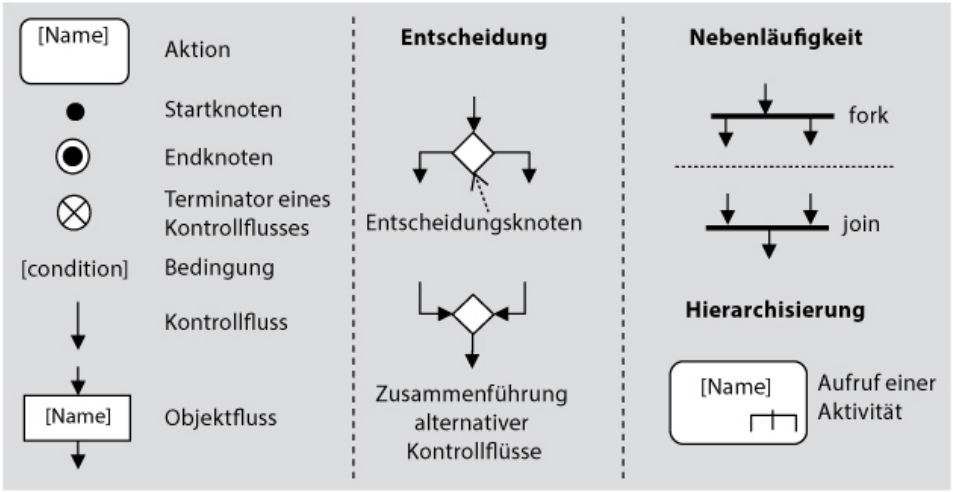

Abbildung 12: Aktivitätsdiagramm (Elemente)

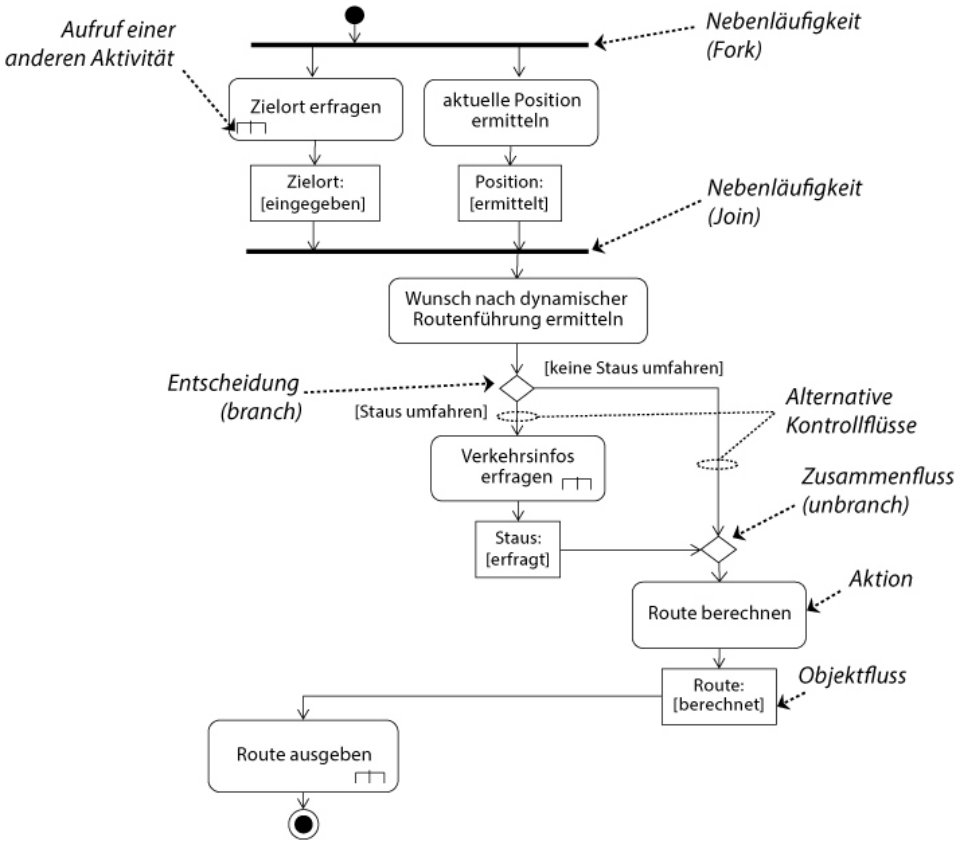

Abbildung 13: Aktivitätsdiagramm (Beispiel 1)

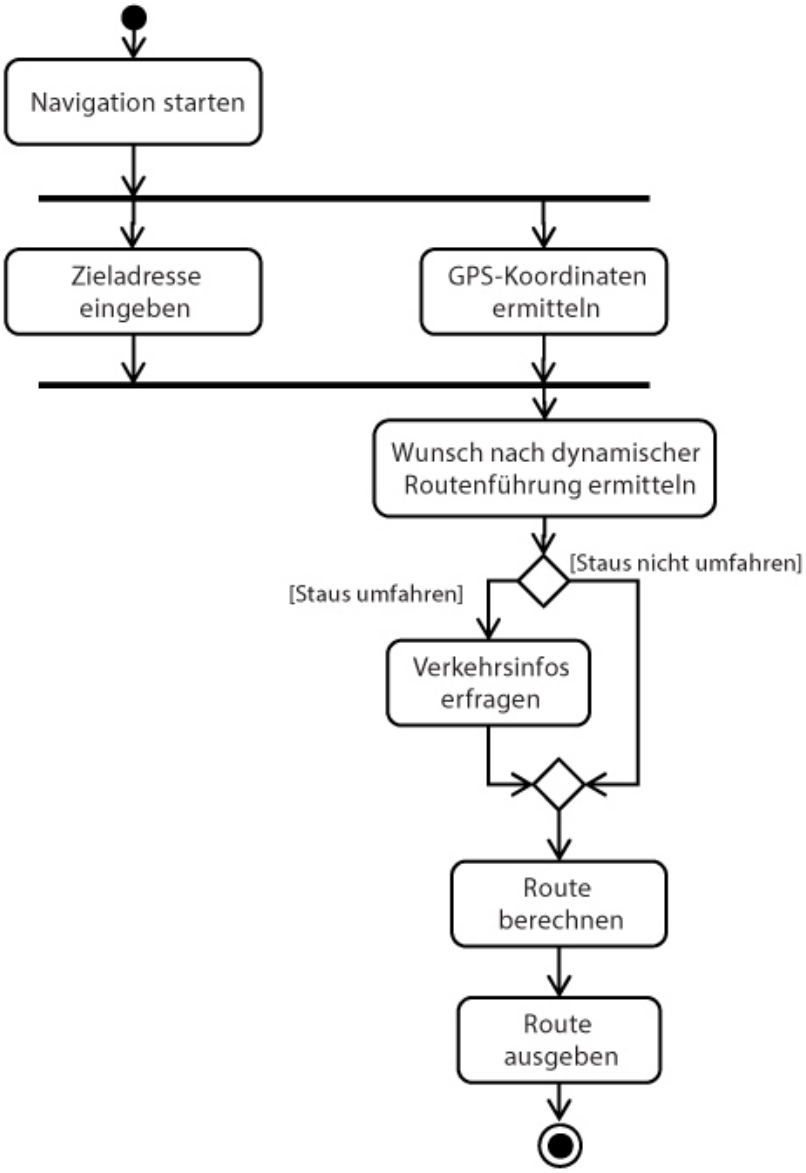

Abbildung 14: Aktivitätsdiagramm (Beispiel 2)

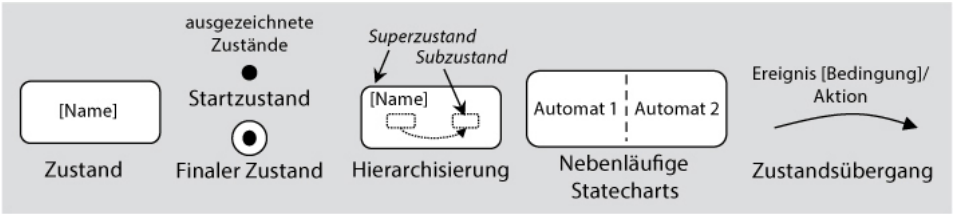

Abbildung 15: Statechart (Elemente)

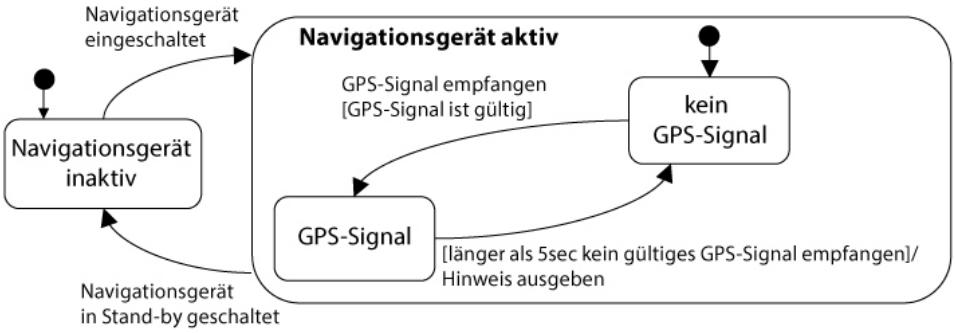

Abbildung 16: Statechart (Beispiel)

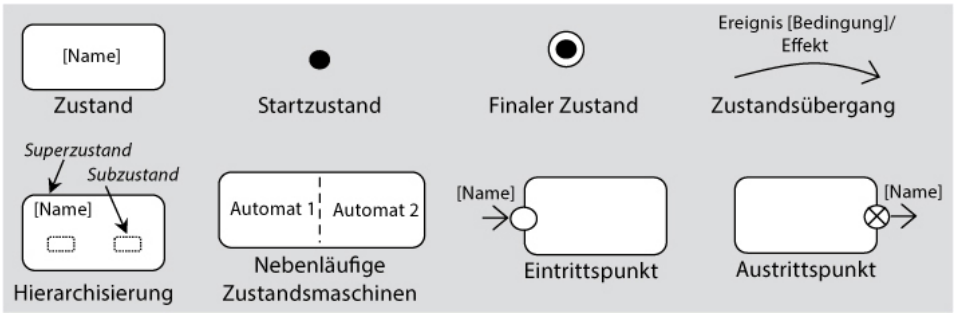

Abbildung 17: Zustandsdiagramm (Elemente)

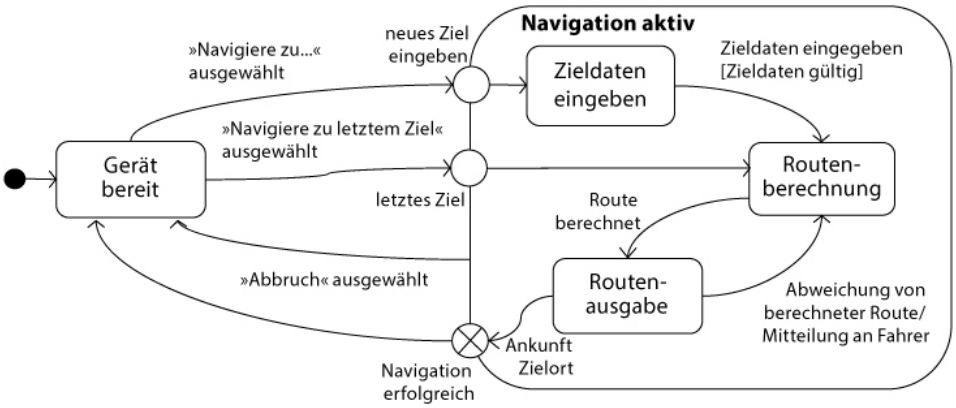

Abbildung 18: Zustandsdiagramm (Beispiel)