Vorlesung Softwaretechnik
(SS 2005)

Prof. Dr. O. Drobnik Michael Berschin, Michael Lauer

Professur für Telematik / ABVS
J. W. Goethe-Universität / Frankfurt am Main

Kapitel 3

Software-Anforderungen

3.1 Einführung

- Festlegung von Anforderungen (*Requirements Engineering*) ist ein Prozess, in dem bestimmt wird, welche Dienste ein System erbringen soll und unter welchen Randbedingungen *(constraints)* es operieren muss.
- Der Begriff *Engineering* wird hier lose gebraucht (⇐ es wird eine systematische Vorgehensweise verwendet).
- Systemanforderungen beschreiben, *was* das System leisten soll, nicht *wie*.
  - **Funktionale Anforderungen:** beschreiben, welche Funktionalität das System in Form von Diensten erbringen sollte (Dienst: Menge zusammengehörender Funktionen)
  - **Nichtfunktionale Anforderungen:** beschreiben Randbedingungen des Systems (Antwortzeit) oder des Entwicklungsprozesses (bestimmter Sprachstandard)

Anforderungsdokument

- Anforderungen an ein neues System werden auf unterschiedlichen Ebenen benötigt:
  - Die Ausschreibung eines großen Projekts muss so abstrakt sein, dass sich mehrere Anbieter darum bewerben können (mit unterschiedlichen Vorschlägen).
  - Nachdem ein Anbieter den Zuschlag bekommen hat, muss dieser die Anforderungen so detailliert aufschreiben, dass der Kunde verstehen und validieren kann, was das System leisten wird.
- Diese Dokumente können beide als *Anforderungsdokument* aufgefasst werden.

Anforderungsdokument

- Wir unterscheiden:
  - **Anforderungsdefinition** (*requirements definition*): Aussage (in natürlicher Sprache mit Diagrammen) darüber, welche Dienste das System erbringen soll und unter welchen Randbedingungen es operieren muss.
  - **Anforderungsspezifikation** (*requirements specification*): Strukturiertes Dokument, das die Dienste im Detail darstellt.
    - Wird oft auch funktionale Spezifikation (*functional specification*) genannt.
    - Sie dient als „Vertrag“ zwischen Kunde und Systementwickler.
  - **Softwarespezifikation** (*software specification*): Abstrakte Beschreibung der Software als Basis für Entwurf und Implementierung. Fügt der Anforderungsspezifikation weitere Details hinzu.
- Üblicherweise beginnt man mit der Anforderungsdefinition und erweitert diese zu einer Anforderungsspezifikation. Die Softwarespezifikation kommt dann in einem späteren Schritt hinzu.

Anforderungsdokument

- Diese Aufteilung der Systemspezifikation ist sinnvoll, weil sie auf die unterschiedlichen Arten von Lesern Rücksicht nimmt:
  - **Anforderungsdefinition:** Manager-Ebene
    - Kunden- und Anbieter-Manager,
    - Endbenutzer,
    - Technisches Personal beim Kunden,
    - Systemarchitekten.
  - **Anforderungsspezifikation:** Projektmanager-Ebene und leitendes technisches Personal
    - Endbenutzer,
    - Technisches Personal beim Kunden,
    - Systemarchitekten,
    - Softwareentwickler.
  - **Softwarespezifikation:** Beteiligte Softwareentwickler
    - Systemarchitekten,
    - Softwareentwickler,
    - eventuell technisches Personal beim Kunden.

Schwieriges Problem

- Ein schwieriges Problem (*wicked problem*) ist ein Problem, das so komplex ist und aus so vielen Teilbereichen besteht, dass es keine endgültige Problemspezifikation gibt; die wahre Natur des Problems kommt erst während der Entwicklung der Lösung zu Tage.
- Extrembeispiel: Katastrophenschutz
  - Niemand kann vorhersagen, wo und wann z.B. ein Erdbeben auftritt, welche Auswirkungen es auf die Umgebung haben wird, usw. Man kann darum nicht vollständig spezifizieren, wie mit einem schweren Erdbeben umgegangen werden soll; das kann erst passieren, nachdem das Erdbeben eingetreten ist.

Erdbeben in der Türkei
vom 27. Juni 1998

Probleme bei der Spezifikation

- Große Softwaresysteme werden meist entwickelt, um schwierige Probleme anzugehen. Ihre Spezifikationen sind darum schwierig zu erstellen und mit großer Wahrscheinlichkeit unvollständig.
- Einige Gründe, warum es fast unmöglich ist, die Anforderungen an die Lösung eines Problems vollständig und konsistent zu definieren:
  - Das System soll den Status quo verbessern. Man kennt unter Umständen dessen Schwächen, aber man kann nichts darüber sagen, welche Auswirkungen das verbesserte System haben wird.
  - Große Systeme haben oft eine uneinheitliche Benutzerbasis mit unterschiedlichen Anforderungen und Prioritäten, die sich widersprechen oder ausschließen. Die endgültigen Anforderungen sind ein Kompromiss.
  - Die Benutzer des Systems und die, die dafür bezahlen, sind nicht identisch. Die Anforderungen letzterer entsprechen nicht unbedingt denen der Benutzer.

Probleme bei der Spezifikation

- Es ist auch schwierig, Spezifikationen zu entwickeln, wenn die Anwendung noch nicht gut verstanden ist. In so einem Fall kann ein Prozessmodell auf der Basis von Prototypenentwicklung eher angebracht sein als das klassische „Wasserfallmodell“.

3.2 Prozess der Entwicklung von Anforderungen

- Der Prozess der Entwicklung von Anforderungen ist die Menge der Aktivitäten, die zur Erstellung der Anforderungsdefinition und Anforderungsspezifikation führen. Gleichzeitig werden dabei oft auch noch weitere Dokumente (Machbarkeitsstudie, Softwarespezifikation) erstellt.
- Wir unterscheiden 4 Hauptstufen:
  1. Durchführbarkeitsstudie
  2. Anforderungsanalyse
  3. Anforderungsdefinition
  4. Anforderungsspezifikation

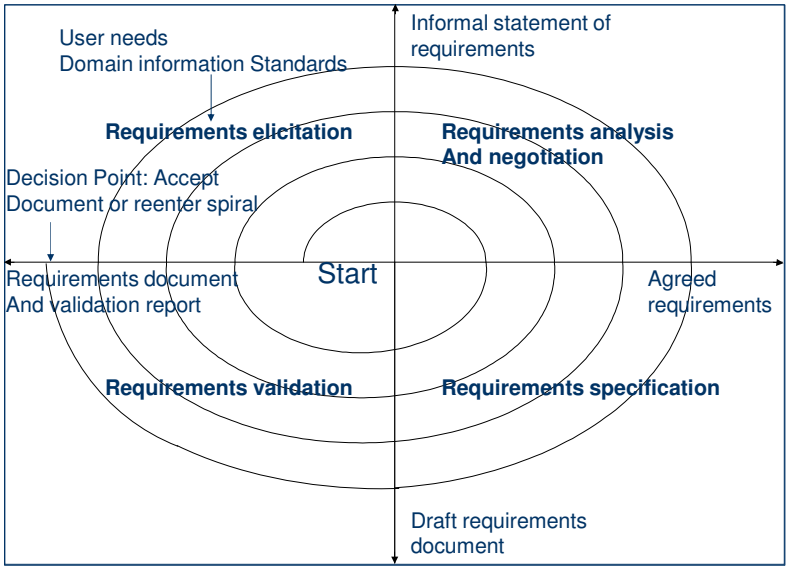
Hauptstufen

- **Durchführbarkeitsstudie:** Abschätzung, ob die Benutzer-wünsche mit der aktuellen Soft- und Hardwaretechnik erfüllt werden können. Entscheidet, ob das vorgeschlagene System kosteneffektiv ist. Sollte billig und schnell zu erstellen sein und die Entscheidung ermöglichen, ob eine tiefer gehende Untersuchung sinnvoll ist.
- **Anforderungsanalyse:** Ableiten der Systemanforderungen durch Beobachtung existierender Systeme, Diskussionen mit potentiellen Käufern und Benutzern, usw. Entwicklung von Systemmodellen, um das System besser zu verstehen; evtl. Entwicklung von Prototypen

Hauptstufen

- **Anforderungsdefinition:** Übersetzung der während der Analyse gesammelten Informationen in ein Dokument, das eine Reihe von Anforderungen definiert. Diese sollten möglichst genau wiedergeben, was der Kunde haben will. Das Dokument muss von Kunden und Endbenutzern verstanden werden können.
- **Anforderungsspezifikation:** Eine detaillierte und genaue Beschreibung der Systemanforderungen wird als Basis eines Vertrags zwischen Kunden und Anbieter niedergelegt. Dies geschieht oft gleichzeitig mit dem abstrakten Systementwurf; Spezifikation und Entwurf beeinflussen sich gegenseitig. Hierbei werden fast immer Fehler in der Anforderungsdefinition festgestellt, die dann geändert werden muss.
- Diese Vorgänge werden nicht einfach in Folge ausgeführt, sondern iteriert.

Iterierte Entwicklung von Anforderungen



Klingonische Softwareentwickler

- Die 5 häufigsten Aussagen eines klingonischen Software-entwicklers:

5. „Klingonische Funktionsaufrufe haben keine ‚Parameter' – sie haben ‚Argumente'. Wage ja nicht zu widersprechen!"

Klingonische Softwareentwickler

- Die 5 häufigsten Aussagen eines klingonischen Software-entwicklers:

4. „Einrückungen im Code?! Ich zeige dir wie man einrückt, wenn ich Deinen Schädel einrücke!"

Klingonische Softwareentwickler

- Die 5 häufigsten Aussagen eines klingonischen Software-entwicklers:

3. „Die Abteilung vom technischen Qualitätsmanagement hat von mir verlangt, meinen Quelltext zu kommentieren! Sie werden uns nie wieder belästigen…"

Klingonische Softwareentwickler

- Die 5 häufigsten Aussagen eines klingonischen Softwareentwicklers:

2. „Spezifikationen sind für die Schwachen und Ängstlichen!“

Klingonische Softwareentwickler

- Die 5 häufigsten Aussagen eines klingonischen Softwareentwicklers:

1. „Debugging? Klingonen debuggen nicht. Unsere Software ist nicht dazu gedacht, die Schwachen zu verhätscheln!“

3.3 Ermittlung von Anforderungen

- Quellen für Anforderungen:
  - Ziele des Unternehmens, z.B. kritische Erfolgsfaktoren für Unternehmen,
  - Fachwissen über Anwendungsgebiet,
  - Akteure,
  - Betriebsbedingungen für das System,
    - **Beispiel:** Einhalten von Reaktionszeiten auf Anfragen, Interoperabilität mit anderen Systemen
  - Organisatorische Bedingungen.
    - **Beispiel:** Einbettung des Systems in Geschäftsprozesse
- Ermittlungstechniken:
  - Interviews mit Akteuren (Einzeln oder in Gruppen)
  - Szenarien: Exemplarische Abläufe in der Benutzung des Systems
  - Prototypen zur Verdeutlichung unklarer Anforderungen
  - Moderierte Sitzungen / Gesprächsrunden mit Akteuren, die unterschiedliche Anforderungen haben (Konfliktregelung).
  - Beobachtung: Ethnographische Methode

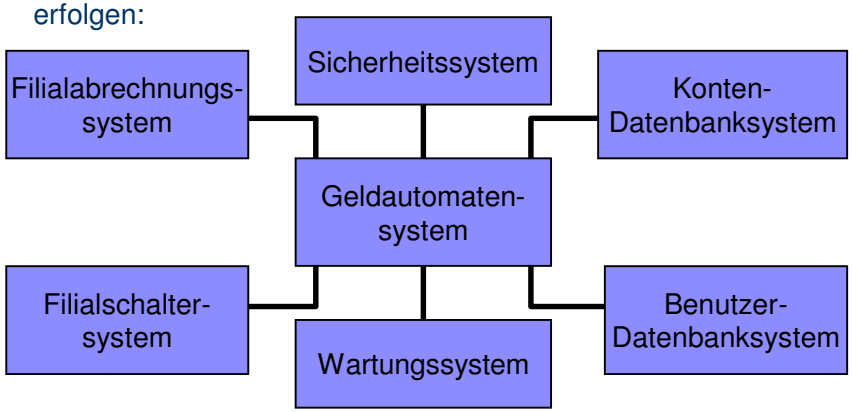
Ermittlung von Anforderungen

Systeme und ihr Umfeld

- Schon früh im Analyseprozess muss entschieden werden, wo das System aufhört und das „Umfeld“ des Systems anfängt, um die Kosten des Systems und die Analysedauer zu begrenzen bzw. die Entwicklung anderer Systeme im Umfeld zu ermöglichen.
- Typische Szenarien:
  - Ein automatisiertes System soll ein manuelles ersetzen. Dann ist die Umgebung des alten Systems auch die des neuen.
  - Oftmals ist die Trennung einigermaßen willkürlich: Ein CASE-System kann eine eigene Datenbank enthalten oder auf ein existierendes Datenbanksystem zurückgreifen. Wo die Trennlinie gezogen wird, ist nicht nur ein technisches, sondern auch ein „politisches“ Problem.

Systemumfeld

- Ein Teil der Analyse besteht daraus, das Umfeld des Systems zu identifizieren und die Abhängigkeiten zwischen dem System und seinem Umfeld festzustellen.
- Dies kann z.B. durch die Erstellung eines Blockdiagramms erfolgen:



Abhängigkeit vom Kontext

- Softwaresysteme existieren nicht in einem Vakuum, sondern werden im Kontext einer Gesellschaft oder Organisation benutzt.
- Dieser Kontext kann sowohl die Anforderungen an das System als auch den Analysevorgang beeinflussen oder gar dominieren. (Die meisten Analysemethoden ignorieren diese Tatsache.)
- **Szenario:** Ein vorgeschlagenes Softwaresystem ermöglicht es der „Chefetage“, auf Informationen direkt (und nicht über das mittlere Management) zuzugreifen. Top-Manager benutzen normalerweise nicht selbst Computer, sondern haben dafür eine Sekretärin oder ähnliches Personal.

Abhängigkeit vom Kontext

- Mögliche Einflüsse auf die Anforderungen:
  - **Der Status des Managers:** Manager könnten glauben, dass sie zu wichtig sind, um selbst eine Tastatur zu benutzen. Außerdem können sie vielleicht nicht schnell Maschine schreiben, und das ist ihnen gegenüber ihren Sekretärinnen peinlich. Systeme mit einer kommandoorientierten Schnittstelle sind deshalb möglicherweise nicht akzeptabel.
  - **Die Verantwortung des Managers:** Manager können sich vielleicht nicht lange genug von anderen Verpflichtungen befreien, um zu lernen, wie man das neue System benutzt. Darum sind Systeme, die eine (längere) Einarbeitungszeit benötigen, möglicherweise nicht akzeptabel.
  - **Persönliche Interessen:** Die Absicht, die Anzahl der mittleren Manager zu reduzieren, könnte einen Einfluss auf die Anforderungsanalyse haben. Das mittlere Management könnte dadurch ein verstecktes Interesse daran haben, dass aus dem geplanten System nichts wird. Das mittlere Management ist aber eine wichtige Informationsquelle für die Systemanforderungen. Die Manager können sich weigern, bei der Analyse zu kooperieren, oder wichtige Informationen zurückhalten.

Ethnographie

- Eine Methode, die Soziologen und Sozialanthropologen in den Analyseprozess einbindet, basiert auf der Technik der *Ethnographie.* Bei dieser wird die tägliche Arbeit beobachtet und dokumentiert, welche Aufgaben die Beteiligten erledigen.
- Der Vorteil der Ethnographie ist, dass die Beteiligten nicht erklären müssen, was sie tun. Die meisten Leute haben damit große Schwierigkeiten, weil sie über ihre Arbeit nicht bewusst nachdenken und u. U. auch nicht wissen, wie ihre Tätigkeit mit der anderer Stellen in der Organisation zusammenhängt.
- Einflüsse des gesellschaftlichen Umfelds bzw. der Organisation, welche die Arbeit beeinflussen, aber den Beteiligten nicht auffallen, können von einem neutralen Beobachter bemerkt werden.
- Die Ethnographen sollten in der Lage sein, die wirklichen Abläufe anstelle der formal festgelegten herauszufinden, und so Anforderungen für effektive Software vorzuschlagen.

Ethnographie

- Während in der „klassischen“ Ethnographie der Sozialwissenschaftler eine Zeitlang beobachtet und dann seine Ergebnisse berichtet, muss die Ethnographie für die Zwecke der Anforderungsanalyse gezielter vorgehen. Eine gute Methode ist die Kombination ethnographischer Techniken mit der Entwicklung von Prototypen.
- Ein Risiko ethnographischer Methoden ist das „Eisenbahnparadoxon“ (Weinberg 1982):

Eisenbahnparadoxon: Eine Bahnlinie wird nicht gut bedient, und die Kunden beschweren sich. Sie finden Alternativen (z.B. Autos), und die Linie wird darum noch weniger benutzt. Bei der Überprüfung der Beschwerden kommt heraus, dass die Linie nicht populär ist, und es wird entschieden, dass man sie gar nicht braucht.

Ethnographie

- Wenn man sich nur die offensichtlichen Eigenschaften eines Systems ansieht, kann man leicht einen falschen Eindruck erhalten.
- Ein Beispiel betrifft Flugsicherungssysteme, bei denen ein Warnton ertönt, wenn Flugzeuge auf potenzielle Kollisionskurse platziert werden. Die Systeme können aber nicht zwischen Fehlern der Fluglotsen und ihren bewussten Entscheidungen unterscheiden. Viele Fluglotsen schalten den Warnton darum aus.
- Eine oberflächliche Analyse könnte entscheiden, dass der Warnton nicht gebraucht wird (niemand benutzt ihn), wenn in Wirklichkeit die Bedingungen für das Ertönen der Warnung verbessert werden sollten.

3.4 Anforderungsanalyse

3.4.1 Analyseprozess

- Die Anforderungsanalyse ist die erste große Phase des *Requirements-Engineering.* Sie ist wichtig, da die Akzeptanz des fertigen Systems davon abhängt, wie gut es den Anforderungen des Kunden entspricht.
- Beteiligte Personenkreise und deren Rollen:
  - Endbenutzer,
  - deren Vorgesetzte,
  - andere vom System betroffene Personen beim Auftraggeber,
  - Entwickler und Wartungspersonal für Systeme, die mit dem neuen System zusammenarbeiten,
  - Experten für den Problembereich und
  - Vertreter der relevanten Gewerkschaften.

Analyseprozess

- Die Beteiligten wissen oft nur schemenhaft, was sie überhaupt vom „Computer“ erwarten, können konkrete Vorstellungen nicht artikulieren oder haben unrealistische Erwartungen.

“The computer salesman said we can do all of our banking online. I slipped my paycheck into the slot, but nothing happened.”

Analyseprozess

- Die Beteiligten verwenden ihre eigene Begriffswelt und setzen ihr eigenes Wissen über den Problembereich voraus.

Analyseprozess

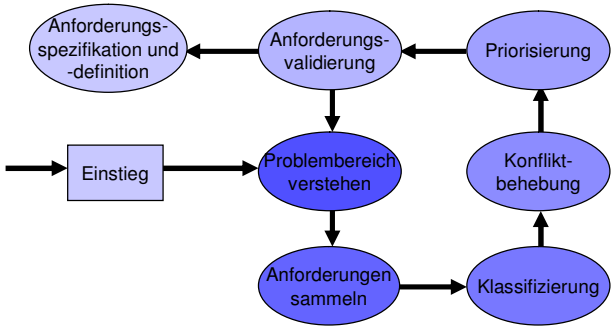
- Verschiedene Beteiligte haben unterschiedliche Anforderungen und drücken diese unterschiedlich aus; Gemeinsamkeiten und Konflikte müssen erkannt werden.

Analyseprozess

- Innerhalb einer Organisation wird die Analyse von politischen Faktoren beeinflusst, von denen die Endbenutzer nichts wissen. Diese können z.B. auf Entscheidungsträger zurückgehen, die ihre eigenen Ziele bei der Systemerstellung verfolgen.
- Die Analyse findet in einem dynamischen wirtschaftlichen und geschäftlichen Kontext statt. Die Prioritäten zwischen verschiedenen Anforderungen können sich ändern. Neue Anforderungen können von neuen Beteiligten kommen, die vorher nicht gefragt wurden.
- Für eine erfolgreiche Anforderungsanalyse muss man als Systementwickler versuchen, den Problembereich möglichst gründlich zu verstehen.

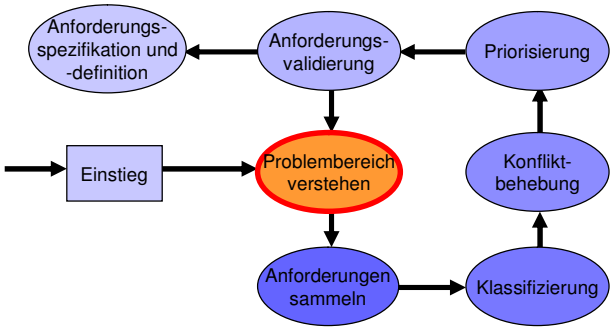
Analyseprozess

- Das Modell für die Anforderungsanalyse muss als Kreislauf verstanden werden, der mit dem Verstehen des Problembereichs beginnt und mit der Validierung der Anforderungen endet. Das Verständnis der Anforderungen steigt mit jedem Durchlauf.



Analyseprozess

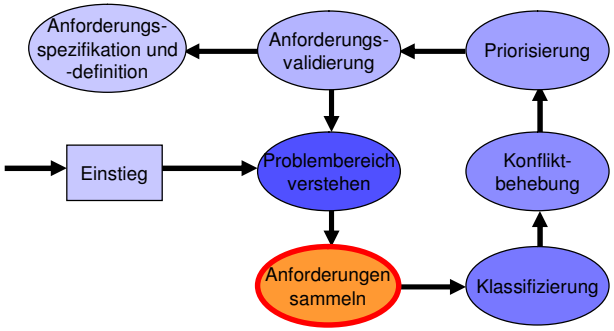
- Das Modell für die Anforderungsanalyse muss als Kreislauf verstanden werden, der mit dem Verstehen des Problembereichs beginnt und mit der Validierung der Anforderungen endet. Das Verständnis der Anforderungen steigt mit jedem Durchlauf.



- **Problembereich verstehen:**
  Wenn man ein System für einen Supermarkt entwickelt, muss man so viel über Supermärkte lernen wie möglich / nötig.

Analyseprozess

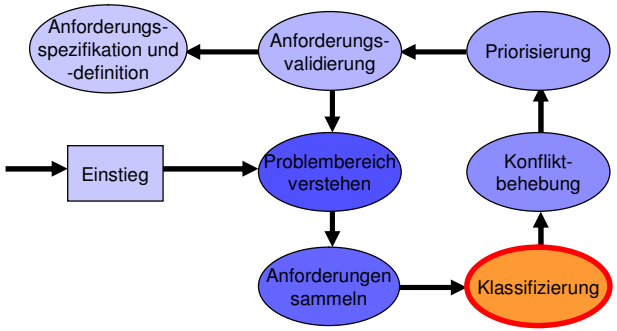
- Das Modell für die Anforderungsanalyse muss als Kreislauf verstanden werden, der mit dem Verstehen des Problembereichs beginnt und mit der Validierung der Anforderungen endet. Das Verständnis der Anforderungen steigt mit jedem Durchlauf.



- **Anforderungen sammeln:**
  Man unterhält sich mit den Beteiligten, um ihre Anforderungen herauszufinden. Hierbei lernt man natürlich das Problem besser verstehen. (Wichtig: Trainieren von Interviewtechniken)

Analyseprozess

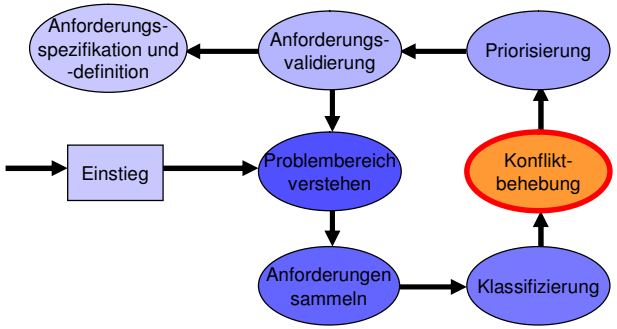
- Das Modell für die Anforderungsanalyse muss als Kreislauf verstanden werden, der mit dem Verstehen des Problembereichs beginnt und mit der Validierung der Anforderungen endet. Das Verständnis der Anforderungen steigt mit jedem Durchlauf.



- **Klassifizierung:**
  Die unstrukturierten Anforderungen werden sortiert und in zusammengehörende Gruppen eingeteilt.

Analyseprozess

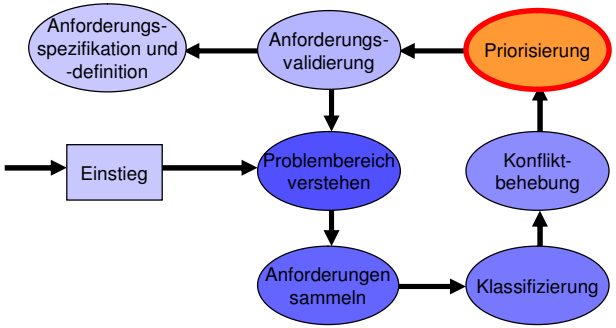
- Das Modell für die Anforderungsanalyse muss als Kreislauf verstanden werden, der mit dem Verstehen des Problembereichs beginnt und mit der Validierung der Anforderungen endet. Das Verständnis der Anforderungen steigt mit jedem Durchlauf.



- **Konfliktbehebung:**
  Die bei mehreren Beteiligten unvermeidlichen Konflikte zwischen Anforderungen werden gefunden und aufgelöst.

Analyseprozess

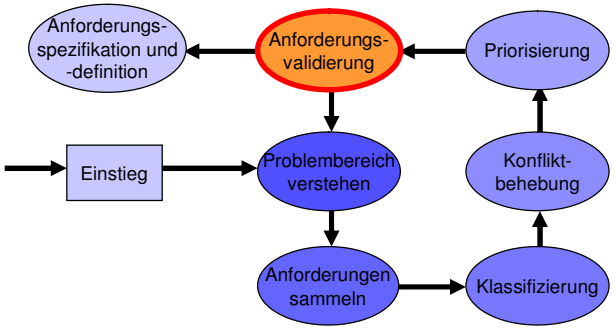
- Das Modell für die Anforderungsanalyse muss als Kreislauf verstanden werden, der mit dem Verstehen des Problembereichs beginnt und mit der Validierung der Anforderungen endet. Das Verständnis der Anforderungen steigt mit jedem Durchlauf.



- **Priorisierung:**
  In jeder Menge von Anforderungen sind manche wichtiger als andere. In diesem Stadium interagiert man mit den Beteiligten, um herauszufinden, welche am wichtigsten sind.

Analyseprozess

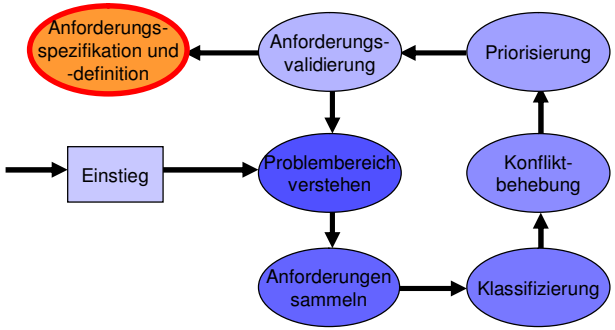
- Das Modell für die Anforderungsanalyse muss als Kreislauf verstanden werden, der mit dem Verstehen des Problembereichs beginnt und mit der Validierung der Anforderungen endet. Das Verständnis der Anforderungen steigt mit jedem Durchlauf.



- **Validierung von Anforderungen:**
  Die resultierenden Anforderungen werden auf Vollständigkeit und Konsistenz geprüft und ob sie den wirklichen Erwartungen der Beteiligten an das System entsprechen.

Analyseprozess

- Das Modell für die Anforderungsanalyse muss als Kreislauf verstanden werden, der mit dem Verstehen des Problembereichs beginnt und mit der Validierung der Anforderungen endet. Das Verständnis der Anforderungen steigt mit jedem Durchlauf.



- **Anforderungsspezifikation und -definition:**
  Mit den nach mehreren Iterationen gewonnenen Erkenntnissen lässt sich schließlich die Anforderungsspezifikation und -definition erstellen.

Strukturierungsaktivitäten

- Der Analyseprozess muss immer drei wichtige Strukturierungsaktivitäten enthalten (Davis 1990):
  1. **Partitionierung:** Feststellung der strukturellen Zusammenhänge zwischen Systembestandteilen („*a* ist Teil von *b*“).
     ⇒ *Entity-Relationship-Modell*
  1. **Abstraktion:** Feststellung von Gemeinsamkeiten zwischen Systembestandteilen.
     ⇒ *Objektmodell*
  1. **Projektion:** Identifizierung verschiedener Ausgangspunkte zum Betrachten eines Problems.

(Die verschiedenen Modelle werden später in der Vorlesung behandelt.)

3.4.2 Klassifikation von Anforderungen

- Anforderungen an Software-Systeme lassen sich unterteilen in funktionale Anforderungen und nichtfunktionale Anforderungen.
- **Funktionale Anforderungen** (functional requirements)**:**
  - Beschreiben die Funktionalität der zu erbringenden Dienste:
    - Welche Dienste sind zu erbringen,
    - wie sollte das System auf bestimmte Eingaben reagieren,
    - wie sollte das System sich in bestimmten Situationen verhalten.
  - Zudem sollte ggf. explizit spezifiziert werden, was das System nicht tun sollte.
  - **Wichtige Bedingungen:**
    - Vollständigkeit,
    - Konsistenz bzw. Widerspruchsfreiheit.

Nichtfunktionale Anforderungen

- **Nichtfunktionale Anforderungen** (non-functional requirements)**:**
  - Beschreiben *wie* das System seine funktionalen Anforderungen erbringen soll und definieren Eigenschaften und Randbedingungen des Systems.
- Beispiele für Eigenschaften:
  - Zuverlässigkeit,
  - Antwortzeit,
  - Speicherbedarf.
- Beispiele für Randbedingungen:
  - Fähigkeiten der E/A-Geräte,
  - Daten-Austauschformate.
- Nichtfunktionale Anforderungen sind manchmal wichtiger als funktionale Anforderungen.
  - **Beispiel:** Ein System, das in einem Flugzeug installiert werden soll, muss von den Luftfahrtorganisationen als sicher zertifiziert sein – andernfalls darf es überhaupt nicht benutzt werden.

Nichtfunktionale Anforderungen

- Spezifische Anforderungen aus den Anwendungsbereichen (*domain requirements*):
  - Anforderungen bzgl. Verwendung bestimmter Standards,
  - Berichtigung von Urheberrechtsregelungen,
  - Beachtung technischer Details aus dem Anwendungsbereich (Anwendungsprämissen).
- **Beispiel:** Ein Programm zum Erstellen einer Steuererklärung muss die geltenden Steuergesetze kennen und beachten.
- **Problem:** Domänenspezifische Anforderungen können im Einzelfall jedoch auch zu den funktionalen Anforderungen gezählt werden, z.B. im Medizinischen Bereich.

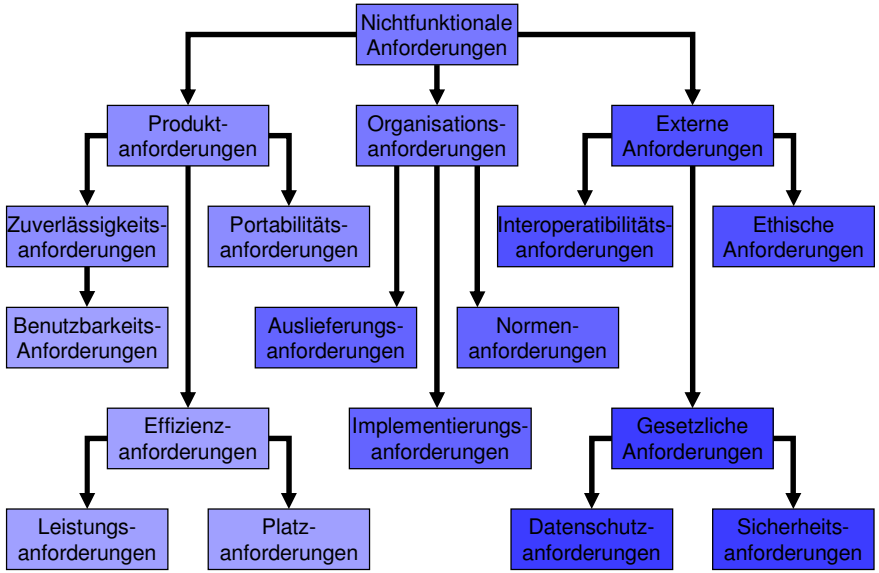
Nichtfunktionale Anforderungen

- Nichtfunktionale Anforderungen können sich statt auf das Produkt auch auf den Prozess der Systementwicklung beziehen:
  - Qualitätsstandards für die Entwicklung,
  - Spezielle zu benutzende Werkzeuge,
  - Zu verfolgendes Prozessmodell.
- Kunden stellen diese Anforderungen aus Qualitäts- und Wartungsgründen.

Nichtfunktionale Anforderungen

- Nichtfunktionale Anforderungen lassen sich grundsätzlich in drei Kategorien aufteilen:
  1. **Produktanforderungen:** Das Produkt soll sich auf eine bestimmte Weise verhalten. Diese Anforderungen lassen sich direkt aus Benutzerwünschen ableiten.
  1. Beispiel: Alle notwendige Kommunikation zwischen dem System und den Benutzern soll im ISO-Latin-1-Zeichensatz abgewickelt werden.
  1. **Organisationsanforderungen:** Diese ergeben sich aus Grundsätzen und Vorschriften der beteiligten Organisationen (Kunde und Entwickler).
  1. Beispiel: Der System-Entwicklungsprozess und die abzuliefernden Dokumente sollen den Definitionen des `XYZCo-SP-STD-03` entsprechen.
  1. **Externe Anforderungen:** Diese ergeben sich aus Faktoren, die extern zum Produkt und dem Prozess seiner Erstellung sind, z.B. Interoperabilität, gesetzliche und ethische Anforderungen. Das System muss für seine Benutzer und die Gesellschaft im Ganzen akzeptabel sein.
  1. Beispiel: Das System soll jedem Benutzer die Möglichkeit geben, zu prüfen, ob personenbezogene Daten für ihn gespeichert werden. Benutzer müssen ihre personenbezogenen Daten einsehen und jegliche Fehler darin korrigieren können.

Nichtfunktionale Anforderungen



Nichtfunktionale Anforderungen

- Ein typischer Fehler besteht darin, nichtfunktionale Anforderungen mit allgemeinen Zielen des Kunden zu verwechseln. Ein Ziel könnte sein, dass das System „benutzerfreundlich“ ist. Dies ist nicht formal verifizierbar. Eine entsprechende (verifizierbare) Anforderung könnte sein, dass die Befehlsauswahl durch den Benutzer über Menüs stattfindet.
- Die beste Methode, um die Verifizierbarkeit zu sichern, ist, sie *quantitativ* auszudrücken, indem man ein messbares Attribut der Software mit der gewünschten Eigenschaft in Verbindung bringt.

Nichtfunktionale Anforderungen

Eigenschaft	Maß
Geschwindigkeit	Transaktionen pro Sekunde, Antwortzeit auf Ereignis, Zeit für Bildschirmaufbau.
Größe	KBytes, Anzahl von RAM-Chips.
Einfachheit	Ausbildungszeit, Anzahl der Hilfeseiten (?).
Zuverlässigkeit	Mittlere Zeit zwischen Fehlern, Wahrscheinlichkeit für Ausfall, Fehlerrate, Verfügbarkeit.
Robustheit	Zeitdauer für Neustart, Prozentsatz von Ereignissen, die Ausfall verursachen, Wahrscheinlichkeit für Datenverlust bei Ausfall.
Portabilität	Anteil von systemabhängigem Code, Anzahl der Zielsysteme.

Anforderungen

- Die Unterscheidung verschiedener Klassen von Anforderungen wird in konkreten Spezifikationen oft stiefmütterlich gehandhabt.
- Oft bestehen Wechselwirkungen oder Konflikte zwischen verschiedenen (funktionalen und nichtfunktionalen) Anforderungen.
  - **Beispiel:** Die Software für ein System an Bord einer Raumsonde soll höchstens 256K groß sein, weil sie in ein ROM passen muss. Außerdem soll sie in Ada geschrieben sein. Vielleicht ist es nicht möglich, ein Ada-Programm mit der gewünschten Funktionalität in 256K ROM unterzubringen. In diesem Fall muss man mehr ROM vorsehen oder eine andere Sprache wählen.
- In der Spezifikation sollte man zwischen funktionalen und nichtfunktionalen Anforderungen trennen. Dies ist schwierig, weil einzelne nichtfunktionale Anforderungen als Anforderungen an das System ausgedrückt werden können – anstatt als Anforderungen an einzelne Funktionen des Systems. Dies ist insbesondere dann der Fall, wenn es sich um Organisations- oder externe Anforderungen handelt.

3.4.3 Das Anforderungsdokument

- Das Anforderungsdokument enthält die Anforderungen an das System und ist die verbindliche Aussage darüber, was von den Entwicklern verlangt wird.
  - Enthält die Anforderungsdefinition und die Anforderungsspezifikation (nicht immer separat).
  - Ist kein Entwurfsdokument – soll beschreiben, *was* das System tut, aber nicht *wie.*
  - Verbindung zum endgültigen Design sollte hergestellt werden können.
  - „Sollte“ vollständig und konsistent sein (ist schwierig).

  ⇒ muss einfach zu ändern sein (zur Behebung von Fehlern)

Anforderungsdokument

- K. L. Heninger (1980) definiert sechs Anforderungen an ein Anforderungsdokument:
  1. Es sollte nur externes Systemverhalten beschreiben.
  2. Es sollte Randbedingungen der Implementierung beschreiben.
  3. Es sollte leicht zu ändern sein.
  4. Es sollte als Nachschlagewerk für das Wartungspersonal dienen.
  5. Es sollte Überlegungen über den Lebenszyklus des Systems enthalten.
  6. Es sollte akzeptable Reaktionen auf unerwünschte Ereignisse charakterisieren.

Anforderungsdokument

- Laut Sommerville besteht die beste Organisation des Anforderungsdokuments aus einer Folge von Kapiteln mit der detaillierten Spezifikation als Anhang.
- **Generische Struktur:**
  - **Vorwort**
    - Empfohlene Zielgruppe der Leser des Dokuments
    - Versionshistorie mit Änderungen gegenüber früheren Versionen
  - **Einleitung**
    - Wofür wird das System gebraucht?
    - Funktion und Interaktion mit anderen Systemen
    - Wie passt das System in die Gesamtziele des Auftraggebers?
  - **Glossar**
    - Definition der Fachausdrücke im Dokument
      (Keine Annahmen über das Vorwissen der Leser treffen.)

Anforderungsdokument

- **Definition funktionaler Anforderungen**
  - Beschreibung der vom System erbrachten Dienste
  - In natürlicher Sprache mit Diagrammen
  - Reaktion auf unerwünschte Ereignisse (Fehler)
  - Für Kunden verständlich
- **Definition nichtfunktionaler Anforderungen**
  - Der Software auferlegte Randbedingungen
  - Restriktionen der Freiheit des Entwicklers, z.B. Datenformate, Antwortzeiten, Speicherbedarf
  - Produkt- und Prozessstandards
  - Vorhergesehene Änderungen wegen Hardwareweiterentwicklung
  - Veränderte Benutzerwünsche
- **Anforderungsspezifikation**
  - Detaillierter Beschreibung der funktionalen Anforderungen
  - Evtl. zusätzliche Informationen über nichtfunktionale Anforderungen

Anforderungsdokument

- **System- und konzeptuelle Modelle**
  - Ein oder mehrere Modelle, die die Zusammenhänge zwischen Komponenten des Systems und dem System und seiner Umwelt aufzeigen.
  - Objektmodelle, Datenflussmodelle, semantische Datenmodelle
- **Systemevolution**
  - Fundamentale Annahmen über das System
- **Anhang**
  - **Hardware**
    - System- und Schnittstellenbeschreibung spezieller Hardware
    - Minimale und optimale Konfigurationen generischer Hardware
  - **Datenbasis-Anforderungen**
    - Logische Beschreibung der verwendeten Daten und ihrer Zusammenhänge, z.B. Entity-Relationship-Modell
  - **Index**
    - Alphabetisch
    - Diagramme und Abbildungen
    - Tabellen
    - Funktionen und Komponenten

3GPP TS 23.107 V5.3.0 (2002-01)

- Beispiel
  - 3rd Generation Partnership Projekt
    Technical Specification Group Services and Systems Aspects
    QoS Concept and Architekture
    (Release 5.3.0)

(http://www.3gpp.org/ftp/Specs/archive/23_series/23.107/23107-530.zip)

3.4.4 Anforderungsdefinition

- Zur Erinnerung: Die Anforderungsdefinition
  - ist eine abstrakte Beschreibung der Dienste, die das System erbringen soll, und der Randbedingungen für deren Funktion,
  - beschreibt nur das externe Verhalten,
  - ist für „Kunden“ ohne detailliertes Vorwissen verständlich,
  - besteht aus natürlicher Sprache, Formularen und Diagrammen.
- In natürlicher Sprache geschriebene Anforderungsdefinitionen haben einige Probleme:
  - **Mangelnde Klarheit:** Wenn man präzise und unmissverständlich schreiben will, wird das Dokument meist lang und schwer zu verstehen.
  - **Verwirrung:** Funktionale und nichtfunktionale Anforderungen, Ziele des Systems und Entwurfsinformationen werden nicht klar unterschieden.
  - **Verschmelzung:** Mehrere verschiedene Anforderungen werden zu einer zusammengeworfen.

Anforderungsdefinition

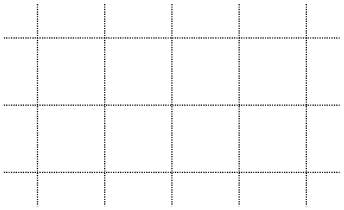
- Ein Beispiel für die Verschmelzung von Anforderungen kommt aus dem Stoneman-Dokument (Buxton 1980), wo die Anforderungen für eine Ada-Programmierumgebung definiert werden:

> **4.A.5** Die Datenbank soll die Erzeugung und Manipulation von Konfigurationsobjekten unterstützen, also Objekten, die Gruppierungen anderer Objekte in der Datenbank darstellen. Die Manipulationsmöglichkeiten sollen den Zugriff auf die Objekte in einer Versionsgruppe über unvollständige Namen zulassen.

- Hier werden konzeptuelle Information (die Existenz der Konfigurationskontrolle) und detaillierte Information (unvollständige Namen) durcheinander geworfen. Letzteres ist eher Material für die Anforderungsspezifikation.

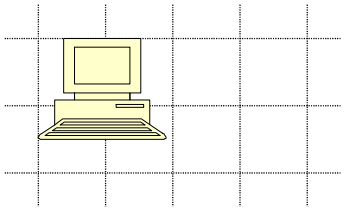
Anforderungsdefinition

- Beispiel: Anforderungsdefinition für eine „Rasterfunktion“



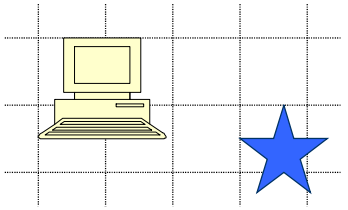
Anforderungsdefinition

- Beispiel: Anforderungsdefinition für eine „Rasterfunktion“



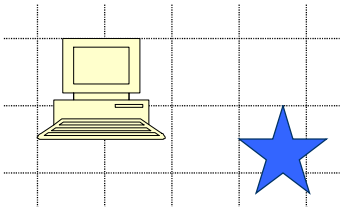
Anforderungsdefinition

- Beispiel: Anforderungsdefinition für eine „Rasterfunktion“



Anforderungsdefinition

- Beispiel: Anforderungsdefinition für eine „Rasterfunktion“



- Mögliche Anforderungsdefinition dazu:

> **2.6 Rasterfunktionen:** Für die Positionierung von Objekten auf dem Diagramm kann der Benutzer über einen Eintrag im Kontrollfeld ein Raster in Zentimetern oder Zoll einschalten. Zu Anfang ist das Raster ausgeschaltet. Das Raster kann jederzeit während einer Sitzung ein- oder ausgeschaltet oder zwischen Zentimetern und Zoll umgeschaltet werden. Auch bei der verkleinerten Darstellung gibt es eine Rasterfunktion, aber die Anzahl der gezeigten Rasterlinien wird verkleinert, um das Diagramm nicht zu überladen.

Anforderungsdefinition

> **2.6 Rasterfunktionen:** Für die Positionierung von Objekten auf dem Diagramm kann der Benutzer über einen Eintrag im Kontrollfeld ein Raster in Zentimetern oder Zoll einschalten. Zu Anfang ist das Raster ausgeschaltet. Das Raster kann jederzeit während einer Sitzung ein- oder ausgeschaltet oder zwischen Zentimetern und Zoll umgeschaltet werden. Auch bei der verkleinerten Darstellung gibt es eine Rasterfunktion, aber die Anzahl der gezeigten Rasterlinien wird verkleinert, um das Diagramm nicht zu überladen.

- Oft versucht man unbewusst, ein einziges Anforderungsdokument zu erstellen, das sowohl als Anforderungsdefinition als auch als Anforderungsspezifikation dienen soll. Die Vermischung von Definitionen (für nicht-technisches Publikum) und Anforderungen (für technische Leser) ist ein oft beobachtetes Phänomen.

Anforderungsdefinition

> **2.6 Rasterfunktionen:** Für die Positionierung von Objekten auf dem Diagramm kann der Benutzer über einen Eintrag im Kontrollfeld ein Raster in Zentimetern oder Zoll einschalten. Zu Anfang ist das Raster ausgeschaltet. Das Raster kann jederzeit während einer Sitzung ein- oder ausgeschaltet oder zwischen Zentimetern und Zoll umgeschaltet werden. Auch bei der verkleinerten Darstellung gibt es eine Rasterfunktion, aber die Anzahl der gezeigten Rasterlinien wird verkleinert, um das Diagramm nicht zu überladen.

- Der erste Satz dieser Definition vermischt drei Arten von Anforderungen:
  - Eine konzeptuelle funktionale Anforderung (das Programm soll ein Raster liefern). Es wird auch erklärt, warum ein Raster sinnvoll ist.
  - Eine nichtfunktionale Anforderung über die Maßeinheiten.
  - Eine nichtfunktionale Anforderung über die Benutzungsschnittstelle.
- Außerdem ist die Definition unvollständig: Es wird gesagt, dass das Raster ausgeschaltet ist, aber die Maßeinheit, die das Raster beim ersten Anschalten hat, ist nicht festgelegt – genauso wenig wie der Abstand zwischen den Linien.

Anforderungsdefinition

- **Bessere Version:**

> **2.6 Rasterfunktionen**
>
> 1 2.6.1 Der Editor hat eine Rasterfunktion, bei der ein Gitter von horizontalen und vertikalen Linien im Hintergrund des Fensters erscheint. Dieses Raster ist passiv; die Anordnung von Objekten ist die Sache des Benutzers.
>
> 1 Hintergrund: Die Rasterfunktion hilft bei einer ordentlichen Ausrichtung von Objekten. Obwohl ein aktives Raster, bei dem Objekte an Rasterlinien „springen“, nützlich sein kann, ist der Benutzer am besten in der Lage, über die Positionierung zu entscheiden.
>
> 1 2.6.2 In der verkleinerten Darstellung (siehe 2.1) muss der Abstand zwischen den Rasterlinien vergrößert werden.
>
> 1 Hintergrund: Wenn man den Abstand nicht vergrößert, dann wird der Fensterhintergrund mit Rasterlinien überladen.

(aus Spezifikation: ECLIPSE/WS/Tools/DE/FS 5.6)

Anforderungsdefinition

- Eine Methode zum Aufstellen brauchbarer Anforderungsdefinitionen besteht in der Festlegung eines Standardformats, an das man sich dann bei allen Anforderungsdefinitionen hält. Die Wahrscheinlichkeit für Auslassungen ist dann geringer und die Anforderungen können leichter miteinander verglichen werden:
- Eigenschaften:
  - Fett- oder Kursivschrift geben dem Text Struktur.
  - Eng verwandte Anforderungen werden gruppiert.
  - Die Hauptanforderung steht in einem kurzen fetten Satz.
  - Der restliche Text fügt Informationen hinzu.
  - Der Hintergrund ist wichtig:
    - Anforderungen erscheinen nicht „willkürlich“
    - Die Wichtigkeit der Anforderungen für Implementierung und Wartung wird klargestellt.
  - Enthält Querverweise zu anderen Teilen der Definition.
  - Enthält einen Querverweis zur detaillierten Spezifikation.

Anforderungsdefinition

- Das folgende Beispiel zeigt eine Liste von Benutzeraktionen. Dies kann helfen, damit mehrere Funktionen auf konsistente Weise zur Verfügung gestellt werden. Implementierungsdetails fehlen auch hier. (Es wird nicht gesagt, wie der Typ gewählt oder der Cursor bewegt werden kann.)

3.5.1 Hinzufügen von Objekten

- 3.5.1.1 Der Editor lässt Benutzer Objekte eines bestimmten Typs zum Entwurf hinzufügen. Objekte werden selektiert (siehe 3.4), wenn sie hinzugefügt werden.
- 3.5.1.2 Der Ablauf von Aktionen zum Hinzufügen ist wie folgt:
  - 1. Der Benutzer wählt den Typ des hinzuzufügenden Objekts.
  - 2. Der Benutzer bewegt den Cursor in die Nähe der gewünschten Position und signalisiert, dass das Objekt dort eingefügt werden soll.
  - 3. Das Objekt wird an seinen endgültigen Platz bewegt.
- Hintergrund: Der Benutzer weiß am besten, wo das Objekt hin soll. Dieser Ansatz gibt dem Benutzer direkte Kontrolle über Typauswahl und Positionierung.

(aus Spezifikation: ECLIPSE/WS/Tools/DE/FS 3.5.1)

3.5 Anforderungsspezifikation

- **Erinnerung: Anforderungsspezifikationen**
  - führen die Anforderungsdefinitionen weiter aus,
  - beschreiben (zusammen mit System- und konzeptuellen Modellen), wie das zu entwerfende System aussieht,
  - enthalten alle notwendigen Informationen über funktionale und nichtfunktionale Anforderungen.
- Anforderungsspezifikationen sind oft in natürlicher Sprache abgefasst. Daraus ergeben sich Probleme:
  - Gebrauch missverständlicher oder unklarer Terminologien.
  - Derselbe Sachverhalt kann auf völlig unterschiedliche Weise ausgedrückt werden. (Übereinstimmung von Anforderungen ist schwer feststellbar.)
  - Abhängigkeiten zwischen Anforderungen sind oft nicht klar.

Anforderungsspezifikation

- Alternativen zum Gebrauch natürlicher Sprache:
  - **Strukturierte natürliche Sprache** verwendet standardisierte Formulare oder Schablonen (siehe voriger Abschnitt).
  - **Entwurfsbeschreibungssprachen** ähneln Programmiersprachen, aber mit abstrakteren Eigenschaften, um ein operationales Modell des Systems festzulegen.
  - **Anforderungsspezifikationssprachen** sind auf die Zwecke von Anforderungsspezifikationen zugeschnitten und werden durch Softwarewerkzeuge unterstützt. Praktisch werden sie jedoch nicht verwendet.
  - **Graphische Notationen** (am bekanntesten: SADT (Ross 1977)) ähneln den Notationsformen für System- und konzeptuellen Modelle (nächstes Kapitel). Sie sind recht populär, aber komplex in der Handhabung.
  - **Mathematische Spezifikationen** (z.B. endliche Automaten, Petrinetze) sind eindeutig, aber schwierig zu verstehen – vor allem für die (meisten) Kunden.

Anforderungsspezifikation

- Wichtig: Querverweise zwischen verwandten Anforderungen
  - Jede Anforderung bekommt eine eindeutige Nummer.
  - Anforderungen sollten explizit auf verwandte Anforderungen verweisen.
  - Jedes Anforderungsdokument sollte eine Querverweismatrix enthalten. Für unterschiedliche Arten von Beziehungen können unterschiedliche Matrizen verwendet werden.
- Strukturierte natürliche Sprache
  - Eigenschaften (nicht zwingend):
    - Eingeschränkte Terminologie
    - Standardisierte Formulare
    - Kontrollstrukturen aus Programmiersprachen
    - Graphische Hervorhebung
  - Ziel:
    - Ausdrucksmöglichkeiten und Verständlichkeit natürlicher Sprache
    - Einheitlichkeit

Anforderungsspezifikation

- Formularstruktur kann an den vom System manipulierten Objekten, an den Funktionen des Systems oder den auftretenden Ereignissen ausgerichtet sein. Sie hängt auch davon ab, wie die Anforderungen strukturiert werden.
- Funktionsorientierte Spezifikationen sind am gebräuchlichsten.
- In einem standardisierten Formular sollten die folgenden Informationen enthalten sein:
  - Beschreibung der spezifizierten Funktion/des Objekts.
  - Beschreibung der Eingaben (und wo sie herkommen).
  - Beschreibung der Ausgaben (und wo sie hingehen).
  - Welche anderen Objekte werden benötigt?
  - Im funktionalen Ansatz: Beschreibung der Vor- und Nachbedingungen.
  - Beschreibung der Nebeneffekte.
- Der Formularbasierte Ansatz kann auch für formale mathematische Spezifikationen benutzt werden. Eine formale Spezifikation kann in einer Reihe von Formularen definiert und umschrieben werden, die auch für die Kunden verständlich sind.

Anforderungsspezifikation

- Strukturierte Formulare können auch als erste Stufe des Entwurfsprozesses verwendet werden. Felder im Formular können, wo passend, in mathematisch formaler Notation ausgefüllt werden.
- Der große Vorteil eines formularbasierten Ansatzes ist, dass die Anforderungen übersichtlicher werden, aber für Nichtfachleute verständlich bleiben. Die inhärente Mehrdeutigkeit natürlicher Sprache wird aber nicht immer völlig beseitigt.

Anforderungsspezifikation

Funktion: Objekt hinzufügen

- **Beschreibung:** Fügt ein Objekt zu einem existierenden Entwurf hinzu. Der Benutzer wählt den Objekttyp und seine Position. Wenn es hinzugefügt wird, dann wird das Objekt die aktuelle „Auswahl“. Der Benutzer wählt die Position, indem er den Cursor dorthin bewegt, wo das Objekt eingefügt werden soll.
- **Eingaben:** Objekttyp, Objektposition, Entwurfs-ID
- **Quelle:** Objekttyp und -position: Benutzereingabe; Entwurfs-ID: Datenbank
- **Ausgabe:** Entwurfs-ID
- **Ziel:** Die Entwurfsdatenbank. Am Ende der Operation wird der neue Entwurf geschrieben.
- **Benötigt:** Am eingegebenen Entwurfs-ID verwurzelter Entwurfsgraph
- **Vorbedingung:** Der Entwurf ist geöffnet und wird auf dem Bildschirm gezeigt
- **Nachbedingung:** Der Entwurf ist bis auf das eingefügte Objekt unverändert
- **Nebeneffekte:** Keine

(aus Spezifikation: ECLIPSE/WS/Tools/DE/FS 3.5.1)

Programm-Beschreibungssprachen

- Programm-Beschreibungssprachen (*Program Description Languages – PDLs*) sind von Programmiersprachen abgeleitete operationale Beschreibungen von Anforderungen.
- Ziel:
  - Milderung der Mehrdeutigkeiten natürlicher Sprache.
  - Erhöhung der Ausdruckskraft gegenüber Programmiersprachen.
  - Syntax- und Semantikprüfung mit geeigneten Werkzeugen.
  - Automatische Feststellung von Inkonsistenzen und Auslassungen.
- Das Systemmodell wird durch zusätzliche Informationen erweitert, welche die Angaben der Modellierungssprache ergänzen. Eine PDL kann in den folgenden Fällen nützlich sein:
  - Eine Operation besteht aus einfacheren Teiloperationen, deren Reihenfolge wichtig ist. Deren Beschreibung in natürlicher Sprache ist oft verwirrend.
  - Hard- und Softwareschnittstellen müssen spezifiziert werden. In einer PDL ist das detaillierter möglich als in natürlicher Sprache.

Beispiel: Eine PDL Beschreibung einer ATM Operation

```
class ATM {
  // declarations here
  public static void main (String args[]) throws InvalidCard {
    try {
      thisCard.read () ; // may throw InvalidCard exception
      pin = KeyPad.readPin () ; attempts = 1 ;
      while (!thisCard.pin.equals (pin) & attempts < 4 )
        {  pin = KeyPad.readPin () ; attempts = attempts + 1 ;
        }
        if (!thisCars.pin.equals (pin))
          throw new InvalidCard („Bad PIN“) ;
      thisBalance = thisCard.getBalance () ;
      do {Screen.prompt („Please select a service“) ;
        service = Screen.touchKey () ;
        switch (service) {
          case Services.withdrawalWithReceipt:
            receiptsRequired = true ;
          case Services.withdrawalNoReceipt:
            amount = KeyPad.readAmount () ;
            if (amount < thisBalance)
            {  Screen.printmsg („Balance insufficient“) ;
               break ;
            }
```


```
          case Services.withdrawalNoReceipt:
            amount = KeyPad.readAmount () ;
            if (amount < thisBalance)
            {  Screen.printmsg („Balance insufficient“) ;
               break ;
            }
            Dispenser.deliver (amount) ;
            newBalance = thisBalance – amount ;
            if (receiptsRequired)
              Receipt.print (amount, newBalance) ;
            break ;
          // other service descriptions here
          default: break ;
        }
      }
      while (service!=Services.quit) ;
        thisCard.returnToUser („Please take your card“) ;
    }
    catch (InvalidCard e )
    { Screen.printmsg („Invalid card or PIN“) ;
    }
    // other exception handling here
  } //main ()
} //ATM
```


- **Vorteile:**
  - Wenn der Leser mit der PDL vertraut ist, sind die Spezifikationen leichter zu benutzen.
  - Wenn die PDL auf der Implementierungssprache beruht, ist der Übergang zum Entwurf einfacher.
- **Nachteile:**
  - Die PDL hat möglicherweise zu wenig Ausdruckskraft für die anwendungsspezifische Beschreibung des Problems.
  - Personal auf der Kundenseite, das sich nicht mit Programmiersprachen auskennt, kann verschreckt werden.
  - Die Spezifikation wird als Entwurf missverstanden und nicht als Systembeschreibung gesehen. Entwurfsentscheidungen werden zu früh getroffen und behindern die Systemarchitekten beim Einhalten anderer (nichtfunktionaler) Anforderungen.

Schnittstellen

- Die meisten Softwaresysteme müssen mit anderen (u. U. bereits vorhandenen) Systemen interoperieren. Dazu gehört, dass die Schnittstellen zwischen ihnen präzise spezifiziert werden.
- Diese Schnittstellen sollten frühzeitig festgelegt und der Anforderungsspezifikation (z.B. als Anhang) beigefügt werden.
- Es gibt drei Arten von Schnittstellen:
  1. **Prozedurale Schnittstellen:** Ein existierendes Subsystem bietet eine Reihe von Diensten an, die über Schnittstellenprozeduren aufgerufen werden.
  1. **Datenstrukturen:** Ein Subsystem tauscht mit einem anderen Daten aus.
  1. **Datendarstellungen:** Formate, die für ein existierendes Subsystem festgelegt wurden.

Prozedurale Schnittstelle

- Beispiel für ein abstraktes Modell eines Druckerservers, das keine Implementierungsdetails enthält:

```
package Print_server is
    procedure Initialize (P : Printer);
    procedure Print (P : Printer; F : Print_File);
    procedure Display_Queue (P : Printer);
    procedure Cancel_Job (P : Printer; N : Job_ID);
end Print_server;
```


Schnittstellen

- Beispiel der Spezifikation einer Datenstruktur für Systemnachrichten:

```
type MESSAGE is record
      Sender: SYSTEM_ID;
      Receiver: SYSTEM_ID;
      Dispatch_time: DATE;
      Length: MESSAGE_LENGTH;
      Terminator: CHARACTER;
      Message: TEXT;
end record;
type SYSTEM_ID is range 20_000. . . 30_000;
type YEAR_TYPE is range 1980. . . 2080;
type DATE is record
      Seconds: NATURAL;
      Year: YEAR_TYPE;
end record;
type MESSAGE_LENGTH is range 0. . . 10_000;
type TEXT is array (MESSAGE_LENGTH) of CHARACTER;
```


Schnittstellen

- Sprachen wie Ada sind dafür geeignet, solche Datenstrukturen festzulegen. Beispielsweise können Größen für die verschiedenen numerischen Typen angegeben werden:

```
for SYSTEM_ID'SIZE use 2*BYTE;
for YEAR_TYPE'SIZE use 2*BYTE;
for MESSAGE_LENGTH'SIZE use 2*BYTE;
```

- Für die Definition von Schnittstellen kann man auch formale mathematische Notationen benutzen. Diese sind aber oft nicht ohne spezielle Kenntnisse zu verstehen. PDLs sind weniger formal, aber stellen einen Kompromiss zwischen Präzision und Verständlichkeit dar. Zumindest für Techniker mit Programmiererfahrung sollten sie zu verstehen sein.

Fazit

- Heutzutage haben PDLs und Sprachen wie Ada kaum mehr praktische Bedeutung – sie wurden im Wesentlichen durch weiterentwickelte konzeptuelle Modelle und leistungsfähige Skriptsprachen abgelöst.
- Mit der *Unified Modelling Language* (UML) lassen sich beispielsweise Abläufe sehr gut beschreiben.
- Für die Beschreibung von Schnittstellen gibt es dedizierte Beschreibungssprachen, z.B. *Interface Definition Languages* (IDLs).
- Skriptsprachen haben die Ausdrucksfähigkeit von PDLs und sind darüber hinaus direkt ausführbar.
- Auf UML und IDLs wird im Laufe der Vorlesung noch ausführlich eingegangen.

3.6 Validierung von Anforderungen

- Durch die Validierung der Anforderungen soll gezeigt werden, dass die definierten Anforderungen tatsächlich beschreiben, was der Kunde haben möchte.
- Unzureichende Validierung von Anforderungen bedeutet, dass etwaige Fehler erst im Entwurfs- oder Implementierungsstadium entdeckt werden. Die Behebung wird dann unter Umständen um mehrere Größenordnungen teurer!
- Anforderungen an die Anforderungen:
  - **Gültigkeit:** Systeme haben verschiedene Benutzer mit verschiedenen Vorstellungen. Die endgültigen Anforderungen sind normalerweise ein Kompromiss, bei dem bestimmte Prioritäten berücksichtigt wurden.
  - **Konsistenz:** Keine Anforderung sollte einer anderen widersprechen.
  - **Vollständigkeit:** Alle gewünschten Anforderungen und Randbedingungen sollten berücksichtigt sein.
  - **Realismus:** Die Anforderungen sollten tatsächlich umsetzbar sein.

Validierung von Anforderungen

- Es ist schwierig, auf der Basis von Anforderungsdefinition und -spezifikation zu zeigen, dass ein System den Anforderungen der Benutzer entspricht.
- Eine wichtige Validierungstechnik ist die Erstellung eines Prototyps.
- Die Formulierung der Anforderungen und ihre Validierung sollten Hand in Hand gehen und sowohl Programmierer als auch Benutzer einbeziehen.

Validierung von Anforderungen

- Neben den oben genannten Punkten sollten untersucht werden:
  - **Überprüfbarkeit:** Kann sinnvoll geprüft werden, ob die Anforderung erfüllt wird?
  - **Verständlichkeit:** Können die Besteller und / oder Endbenutzer des Systems die Anforderung verstehen?
  - **Verfolgbarkeit:** Ist klar, warum die Anforderung existiert? Dies ist besonders wichtig, wenn Anforderungen sich ändern. Es kann notwendig sein, die Auswirkungen einer Änderung auf die Ursache der Anforderung zu untersuchen.
  - **Anpassbarkeit:** Kann die Anforderung geändert werden, ohne dass sich weit reichende Folgen für andere Anforderungen ergeben?
- Widersprüche, Irrtümer und Auslassungen müssen bei der Validierung notiert werden, damit eine Lösung ausgehandelt werden kann.

3.7 Evolution von Anforderungen

- Beim Aufstellen des Anforderungsdokuments für ein Softwaresystem werden die Wünsche der Benutzer mit der Zeit besser verstanden. Ferner kann die Anforderungsanalyse bei größeren Projekten Jahre dauern; die Anforderungen können sich innerhalb dieses Zeitraums durchaus ändern, sowohl was die Systemumgebung als auch was die Ziele des Auftraggebers angeht. Diese Tatsache sollte man berücksichtigen.
- Vom Standpunkt der Evolution aus gibt es zwei Klassen von Anforderungen:
  1. **Dauerhafte Anforderungen** sind relativ stabil, haben mit der Haupttätigkeit des Auftraggebers zu tun und beziehen sich direkt auf das Problemgebiet.
  1. Beispiel: Im Krankenhaus hat man es immer mit Patienten, Ärzten, Pflegepersonal und Therapien zu tun.
  1. **Flüchtige Anforderungen** ändern sich wahrscheinlich während der Systementwicklung oder nach der Inbetriebnahme.
  1. Beispiel: Anforderungen, die auf die Gesundheitspolitik zurückgehen.

Evolution von Anforderungen

- Flüchtige Anforderungen lassen sich weiter unterteilen:
  1. Anforderungen, die sich aufgrund der Arbeitsweise der Organisation ändern,
     1. **Beispiel:** Geänderte Abrechnungsweise der Krankenkassen erfordert Erhebung anderer Therapiedaten.
  1. Anforderungen, die sich während der Entwicklung herausstellen, weil der Auftraggeber das System besser versteht,
  1. Anforderungen, die sich als Konsequenz aus der Einführung des Systems ergeben,
  1. Anforderungen, die Kompatibilität zu Prozessen oder Systemen beim Auftraggeber erzielen sollen und sich ändern, wenn diese sich ändern.

Evolution von Anforderungen

- Das Anforderungsdokument sollte so organisiert sein, dass Änderungen ohne großes Umschreiben eingebaut werden können (⇒ Modularisierung, Vermeiden übermäßiger Querverweise, Quelltext- und Konfigurationsverwaltung).
- Ein komplizierter Prozess für Änderungen führt dazu, dass Änderungen der Implementierung nicht im Anforderungsdokument nachgeführt werden. Später stimmen Anforderungen und System dann nicht mehr überein.
- Es ist vorteilhaft, das Anforderungsdokument in elektronischer Form zu halten.
- Anmerkung: Während man früher mangels geeigneter Standards am Dokumentenaustausch „auf Papier“ nicht vorbei kam, existieren mittlerweile diverse Austauschformate, z.B. XML oder PDF.

3.8 Analysemethoden

3.8.1 Blickwinkelorientierte Analyse

- Bei mittelgroßen oder großen Systemen gibt es normalerweise verschiedene Arten von Endbenutzern, die alle unterschiedliche Anforderungen an das System haben. Hieraus ergeben sich verschiedene **Blickwinkel** (*viewpoints*), die bei der Analyse beachtet werden müssen.
- Die verschiedenen Blickwinkel, unter denen das System erscheint, sind nicht völlig unabhängig, sondern überlappen sich. Ihre Gemeinsamkeiten und Widersprüche sind für die Analyse wichtig.
- Die wichtigsten Blickwinkel sollten identifiziert und für die Strukturierung der Anforderungsanalyse ausgenutzt werden (⇐ Projektion).

Beteiligte

- Bei einem Geldautomatensystem gibt es unter anderem die folgenden Beteiligten:
  - **Bankkunden:** Sie benutzen die Automaten.
  - **Vertreter anderer Banken:** Es gibt Abkommen über gegenseitige Automatenbenutzung.
  - **Filialleiter** und **Kundenbetreuer** in den betroffenen Filialen
  - **Datenbankadministratoren**
  - **Bank-Sicherheitsbeauftragte**
  - **Kommunikationsingenieure**
  - **Marketingabteilung der Bank**
  - **Wartungspersonal**
  - **Personalabteilung der Bank**
- Nicht alle diese Personen sind in derselben Weise betroffen.

Blickwinkelorientierte Analyse

- Was man genau unter einem Blickwinkel verstehen will, hängt von der verwendeten Analysemethode ab:
  - **Datenquelle oder –senke:** Inhaber von Blickwinkeln *produzieren* oder *konsumieren* Daten. In der Analyse werden alle solchen Blickwinkel identifiziert. Man stellt fest, welche Daten wie verarbeitet werden. Anschließend fügt man ein Gesamtbild zusammen und prüft, ob irgendwo Daten produziert, jedoch nie konsumiert werden oder umgekehrt. (CORE-Methode, Mullery 1979)
  - **Repräsentationsschema:** Ein Blickwinkel ist eine bestimmte Art von konzeptuellem Modell. Man stellt verschiedene solche Modelle auf und vergleicht sie, um Anforderungen zu bemerken, die man sonst übersehen würde. (VOSE-Methode, Finkelstein *et al.* 1990)
  - **Empfänger von Diensten:** Blickwinkel sind außerhalb des Systems angeordnet und nutzen dessen Dienste. Sie liefern Eingabedaten oder Steuersignale für diese Dienste. Bei der Analyse werden die verschiedenen Dienste zusammengetragen und analysiert und Konflikte behoben. (VORD-Methode, Kotonya und Sommerville 1992)

Vor- und Nachteile

- Diese Ansätze haben unterschiedliche Stärken und Schwächen.
- Der datenbasierte und der repräsentationsbasierte Ansatz sind sehr leistungsfähig und universell einsetzbar, helfen aber nicht so sehr bei der Strukturierung der Anforderungsanalyse, da zwischen den Prozessbeteiligten und den Blickwinkeln kein einfacher Zusammenhang besteht.
- Der dienstorientierte Ansatz ist am besten für Systeme geeignet, die Dienste für Leute erbringen. Manche Systeme haben wenig Benutzerinteraktion, so dass die Blickwinkel weniger intuitiv sind. Konflikte zwischen Diensten können bemerkt werden, Konflikte zwischen verschiedenen konzeptuellen Modellen dagegen nicht.

Blickwinkelorientierte Analyse

- Der dienstorientierte Ansatz ist auch als VORD-Methode bekannt.
- Für ihn sprechen die folgenden Vorteile:
  - Bei der Mehrzahl interaktiver Systeme bietet es sich an, Endbenutzer als Dienstempfänger anzusehen.
  - Da die Blickwinkel extern sind, können sie die Erhebung der Anforderungen strukturieren.
  - Es ist gut erkennbar, ob ein Beteiligter auch einen zu berücksichtigenden Blickwinkel darstellt. Beteiligte ohne Interaktion mit dem System führen nicht zu neuen Blickwinkeln.
  - Blickwinkel und Dienste sind nützlich zur Strukturierung nichtfunktionaler Anforderungen. Die nichtfunktionalen Anforderungen eines Dienstes können sich allerdings von Blickwinkel zu Blickwinkel unterscheiden.

3.8.2 Methodenbasierte Analyse

- Der am weitesten verbreitete Ansatz zur Anforderungsanalyse verwendet strukturierte Methoden.
- Das Ergebnis dieses Ansatzes ist eine Menge von konzeptuellen Modellen, die durch die jeweilige Methode definiert werden.
- Manche Analysemethoden eignen sich vor allem für die Anforderungserhebung, während andere mehr auf das Design ausgerichtet sind.

Methodenbasierte Analyse

- Die meisten strukturierten Methoden enthalten eine Teilmenge der folgenden Bestandteile:
  - **Prozessmodell:** Definiert die Aktivitäten der Methode.
  - **Modellierungsnotation:** Diagramme, Formulare oder Sprache, z.B. Datenflussdiagramme, Objektdiagramme.
  - **Regeln für Systemmodelle:** Entweder für ein einzelnes Modell oder Modell übergreifend.
  - **Entwurfshinweise:** Durch ihre Anwendung soll schlechtes Design vermieden werden.
  - **Schablonen für Ergebnisse:** Geben vor, wie die erhobenen Informationen präsentiert werden sollen.
- Bei der Analyse können erhebliche Datenmengen aufkommen. Dies ist als *Informationsmanagementproblem* bekannt.
- *CASE-Tools* können helfen, diese Datenmengen zu kanalisieren.

Methodenbasierte Analyse

- Verbreitete Methoden:
  - Objektorientierte Methoden (z.B. UML, Booch & Co. 1996) oder
  - Datenflussorientierte Methoden (DeMarco 1978, Ward und Mellor 1985).
- Diese sind aber eher entwurfsorientiert, da sie theoretische Konzepte aus der Informatik verwenden.
- Ein Problem mit solchen Methoden ist, dass die Endbenutzer und ihre Manager weder objektorientiert noch in „Datenflüssen“ denken – statt dessen wollen sie wissen, wie das System ihnen bei ihrer Arbeit hilft. Insbesondere Benutzerinteraktionen werden von den meisten strukturierten Methoden leider ignoriert.
- Der wesentliche Teil der Anforderungsanalyse besteht in der Erhebung und Integration der Vorstellungen vieler verschiedener Beteiligter. Weder die objektorientierten noch die datenflussorientierten Methoden bieten hierbei besondere Unterstützung.

3.8.3 Anforderungsanalyse mit der VORD-Methode

- Die VORD-Methode dient vor allem zur Erhebung und Analyse von Anforderungen. Sie definiert auch Schritte zur Umsetzung der Anforderungen in ein objektorientiertes konzeptuelles Modell.
- Die wesentlichen Schritte dieser Methode sind:
  - **Identifikation von Blickwinkeln:** Man findet die Stellen, an denen Systemdienste in Anspruch genommen werden, und die betreffenden Dienste.
  - **Strukturierung von Blickwinkeln:** Verwandte Blickwinkel werden in eine Hierarchie eingeordnet. Gemeinsame Dienste werden weiter oben in der Hierarchie untergebracht und von Diensten auf niedrigerer Ebene „geerbt“.
  - **Dokumentation von Blickwinkeln:** Identifizierte und strukturierte Blickwinkel werden auf das System abgebildet und dokumentiert.
- Blickwinkel- und Dienstinformationen werden mit standardisierten Formularen erhoben.
- Im allgemeinen kann ein Formular nicht durch eine einzige Aktivität vollständig ausgefüllt werden, da in jeder Stufe der Methode neue Informationen zum Vorschein kommen.
- Das Gesamtbild ist erst am Ende der Analyse vollständig.

Anforderungsanalyse mit der VORD-Methode

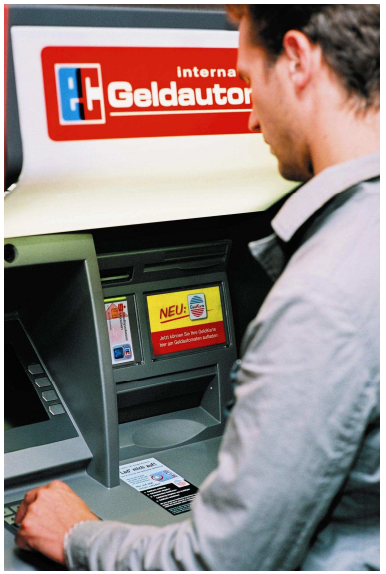
Schablone für Blickwinkel	
Bezeichnung	Der Name des Blickwinkels
Attribute	Informationen über den Blickwinkel
Ereignisse	Verweise auf eine Menge von Szenarien, die beschreiben, wie das System auf Ereignisse am Blickwinkel reagiert
Dienste	Verweise auf eine Menge von Dienstbeschreibungen
Unter-Blickwinkel	Die Namen von untergeordneten Blickwinkeln

Anforderungsanalyse mit der VORD-Methode

Schablone für Dienste	
Bezeichnung	Der Name des Dienstes
Hintergrund	Warum der Dienst erbracht wird
Spezifikation	Verweise auf eine Liste von Dienstspezifikationen
Blickwinkel	Liste der Blickwinkel, die den Dienst in Anspruch nehmen
Nichtfunktionale Anforderungen	Verweise auf eine Menge von nichtfunktionalen Anforderungen an den Dienst
Erbringer	Verweise auf Objekte im System, die den Dienst erbringen

Beispiel: Bankautomat

- Zur Illustration betrachten wir die Analyse der Anforderungen an ein Steuerungssystem für Bankautomaten.
- Die Automaten akzeptieren Anfragen der Benutzer und geben Bargeld und Informationen über den Kontostand heraus.
- Kunden können standardisierte Nachrichten an ihre Bankfiliale schicken, um einen Kontoauszug, Schecks und ähnliches zu erbitten.
- Die Automaten einer Bank gestatten auch Kunden anderer Banken Zugriff auf eine Teilmenge der Funktionalität.

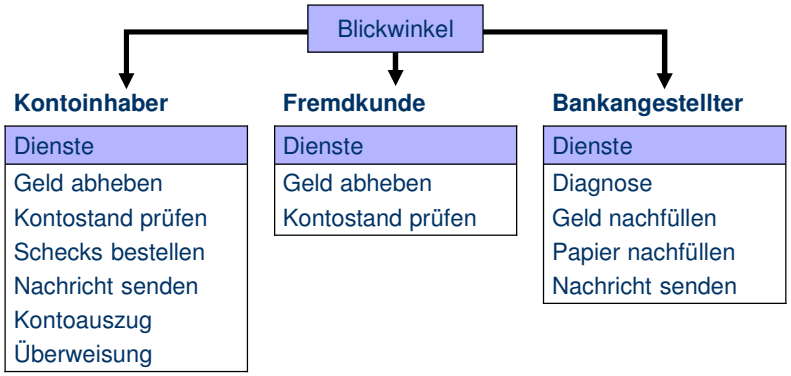


Identifikation von Blickwinkeln

- Der erste Schritt der Analyse ist die Identifikation von Blickwinkeln.
- Eine mögliche Vorgehensweise ist das *Brainstorming*.
- Hierbei wird alles zu Diensten, interagierenden Bestandteilen des Systems, Eingabedaten, Fehlerzuständen, nichtfunktionalen Anforderungen, etc. einfallende notiert.
- Informationsquellen hierfür sind:
  - Dokumente, die die abstrakten Ziele des Systems angeben,
  - Wissen aus früheren Projekten,
  - eigene Erfahrungen oder
  - Gespräche mit Beteiligten.
- Wichtig ist, in diesem Schritt eine weitergehende Strukturierung zu vermeiden. Dies ist die Aufgabe in weiteren Schritten.

Identifikation von Diensten

- Als nächstes sucht man aus den gesammelten Daten die Blickwinkel und die Dienste heraus. Die Dienste werden Blickwinkeln zugeordnet; ein Dienst kann zu mehreren Blickwinkeln gehören.
- Übrigbleibende Dienste deuten auf unidentifizierte Blickwinkel hin. Dasselbe gilt für Eingabedaten und Steuerinformationen.



Identifikation von Eingabedaten

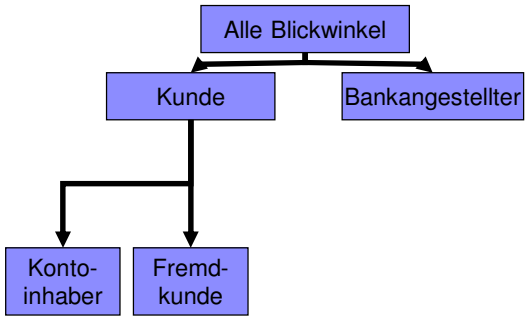
- In diesem Stadium werden die Eingabedaten und Steuer-informationen nur namentlich identifiziert:

Kontoinhaber

Steuereingaben	Dateneingaben
Beginne Transaktion	Kartendaten
Stoppe Transaktion	Geheimzahl
Beende Transaktion	Geldbetrag
Wähle Dienst	Nachricht

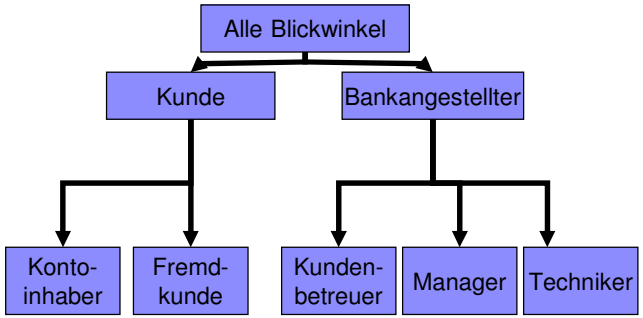
Blickwinkelhierarchie

- Mit den so gewonnenen Informationen werden die Blickwinkel in eine *Blickwinkelhierarchie* eingeordnet.



Blickwinkelhierarchie

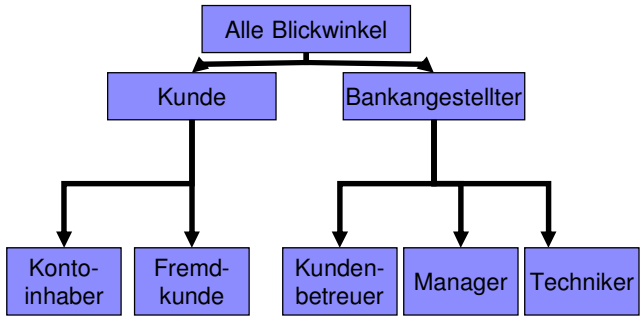
- Mit den so gewonnenen Informationen werden die Blickwinkel in eine *Blickwinkelhierarchie* eingeordnet.



- Durch Diskussion mit Experten oder den Endbenutzern können die Blickwinkel verfeinert bzw. weitere, noch nicht identifizierte Blickwinkel, hinzugefügt werden.

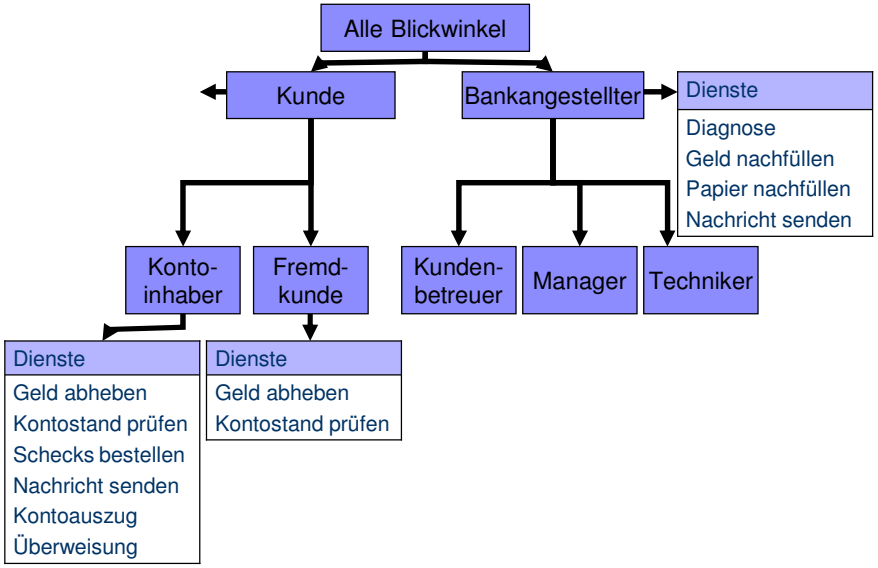
Blickwinkelhierarchie

- Mit den so gewonnenen Informationen werden die Blickwinkel in eine *Blickwinkelhierarchie* eingeordnet.

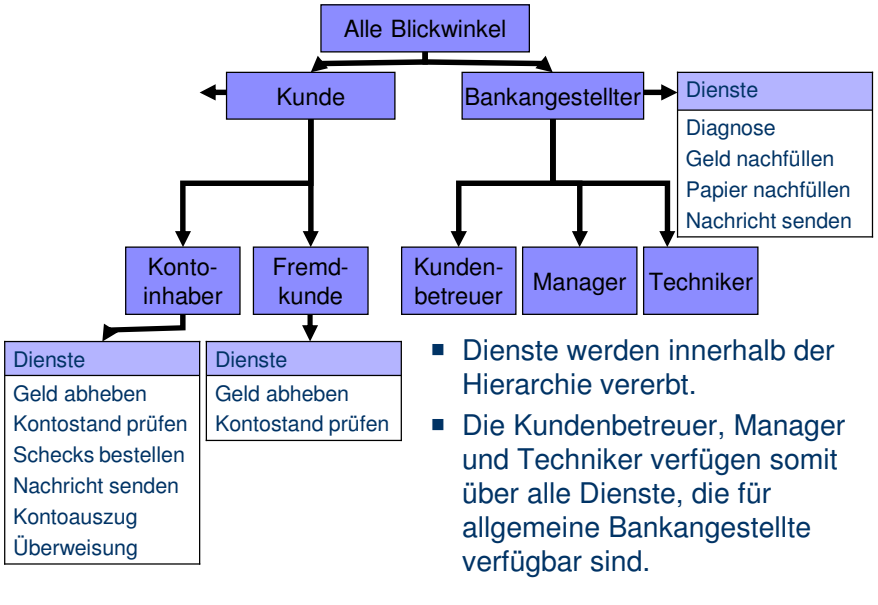


- Im nächsten Schritt werden für jeden Blickwinkel die ermittelten Dienste in die Hierarchie eingetragen.

Blickwinkelhierarchie

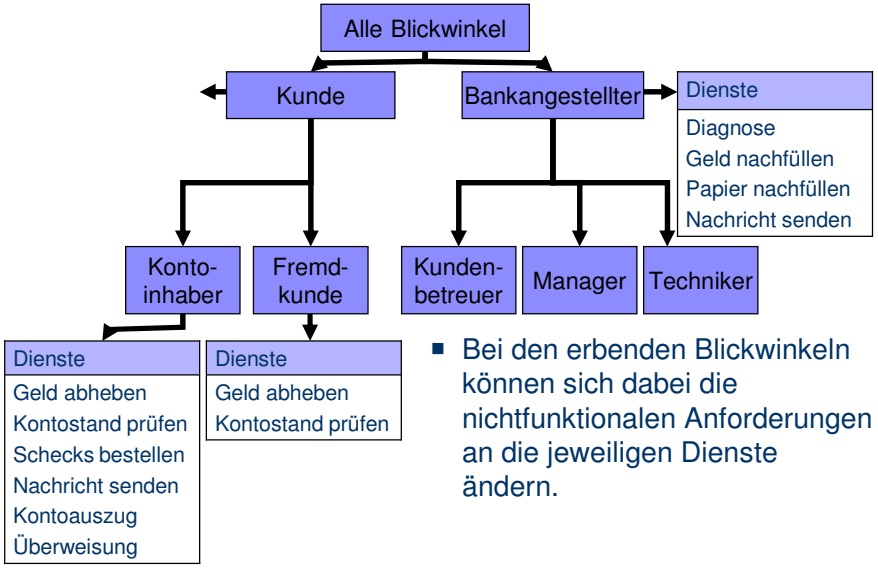


Blickwinkelhierarchie



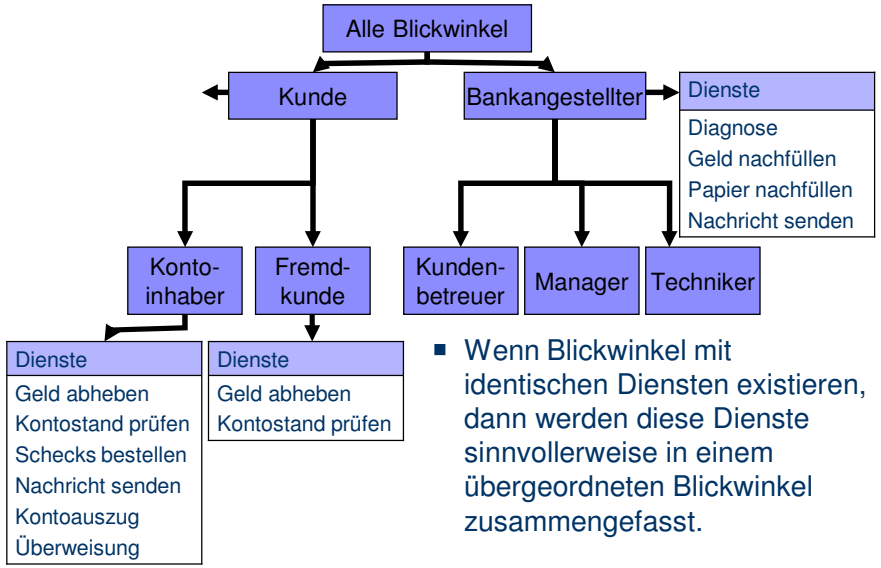
- Dienste werden innerhalb der Hierarchie vererbt.
- Die Kundenbetreuer, Manager und Techniker verfügen somit über alle Dienste, die für allgemeine Bankangestellte verfügbar sind.

Blickwinkelhierarchie



- Bei den erbenden Blickwinkeln können sich dabei die nichtfunktionalen Anforderungen an die jeweiligen Dienste ändern.

Blickwinkelhierarchie



- Wenn Blickwinkel mit identischen Diensten existieren, dann werden diese Dienste sinnvollerweise in einem übergeordneten Blickwinkel zusammengefasst.

Blickwinkelhierarchie

- Wenn Blickwinkel mit identischen Diensten existieren, dann werden diese Dienste sinnvollerweise in einem übergeordneten Blickwinkel zusammengefasst.

Blickwinkelhierarchie

- Die Blickwinkelhierarchie ist die erste Aufteilung eines Systems in seine Bestandteile. Ausgehend von ihr können Pläne für das weitere Vorgehen bei der Analyse gemacht und Anforderungen priorisiert werden.

Anforderungsanalyse mit der VORD-Methode

- Mit den so gewonnenen Blickwinkelinformationen werden Schablonen für die Blickwinkel ausgefüllt.

Schablone für Blickwinkel	
Bezeichnung	Kunde
Attribute	Kontonummer Geheimzahl
Ereignisse	Starte Transaktion Wähle Dienst Stoppe Transaktion Beende Transaktion
Dienste	Stoppe Transaktion Beende Transaktion
Unter-Blickwinkel	Kontoinhaber Fremdkunde

Anforderungsanalyse mit der VORD-Methode

- Im nächsten Schritt werden die Dienste und die von ihnen benötigten Daten und Steuerinformationen weiter präzisiert.

Schablone für Dienste	
Bezeichnung	Der Name des Dienstes
Hintergrund	Verbessern des Kundendiensts; Verringerung der Papierflut
Spezifikation	Benutzer wählen diesen Dienst mit der „Geld abheben“-Taste. Sie geben dann den gewünschten Betrag ein. Dieser wird bestätigt und – wenn der Kontostand es zulässt – ausgezahlt.
Blickwinkel	Kunde
Nichtfunktionale Anforderungen	Geld spätestens 1 Min. nach Bestätigung auszahlen.
Erbringer	*Ist noch nicht ermittelt*

Anforderungsanalyse mit der VORD-Methode

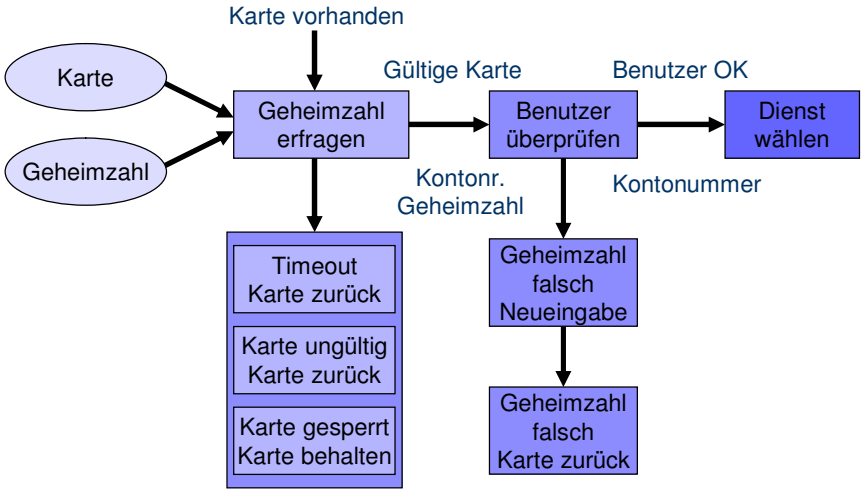
- Man verwendet die Blickwinkel, um diesen Prozess zu strukturieren, und diskutiert jeden Dienst mit Endbenutzern oder Experten.

Schablone für Dienste	
Bezeichnung	Der Name des Dienstes
Hintergrund	Verbessern des Kundendiensts; Verringerung der Papierflut
Spezifikation	Benutzer wählen diesen Dienst mit der „Geld abheben“-Taste. Sie geben dann den gewünschten Betrag ein. Dieser wird bestätigt und – wenn der Kontostand es zulässt – ausgezahlt.
Blickwinkel	Kunde
Nichtfunktionale Anforderungen	Geld spätestens 1 Min. nach Bestätigung auszahlen.
Erbringer	*Ist noch nicht ermittelt*

Ereignisszenarien

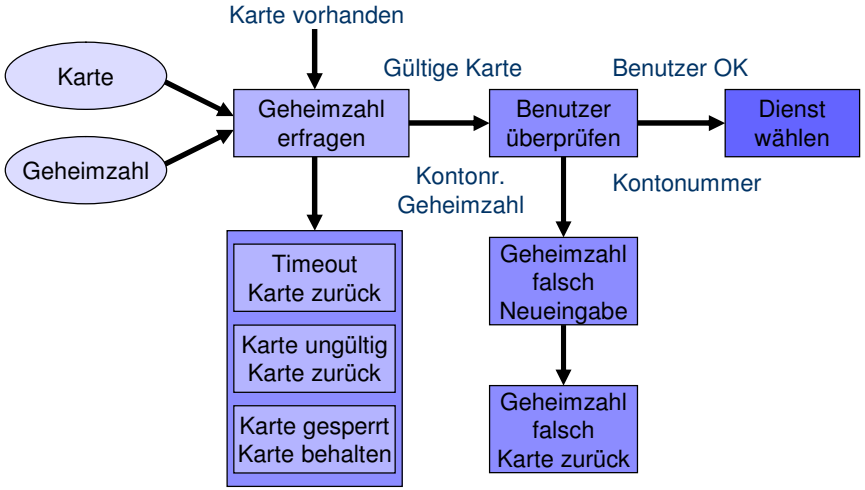
- **Ereignisszenarien** beschreiben das Systemverhalten beim Auftreten bestimmter Ereignisse. Sie geben den Datenfluss und die Aktionen des Systems an.
- Ein wichtiger Bestandteil dieser Szenarien ist die Dokumentation von Fehlersituationen, die auftreten können.

Ereignisszenarien



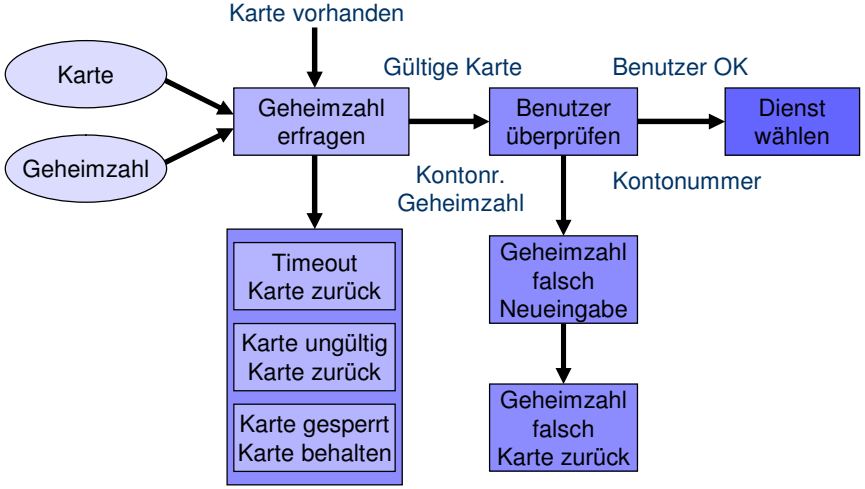
- Daten, die ein Blickwinkel liefert oder erhält, werden in Ellipsen dargestellt.

Ereignisszenarien



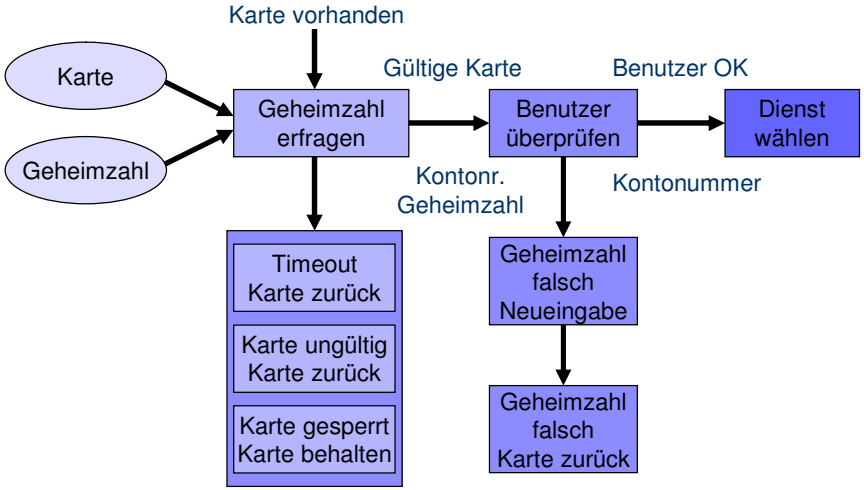
- Steuerinformationen kommen an der Oberkante eines Rechtecks hinein und heraus.

Ereignisszenarien



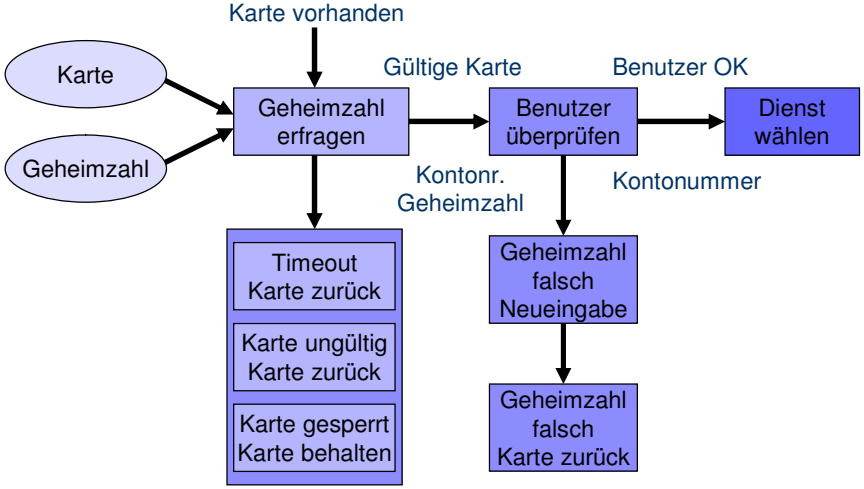
- Ausgabedaten verlassen ein Rechteck an der rechten Seite. Interne Daten des Systems werden dabei nicht eingerahmt.

Ereignisszenarien



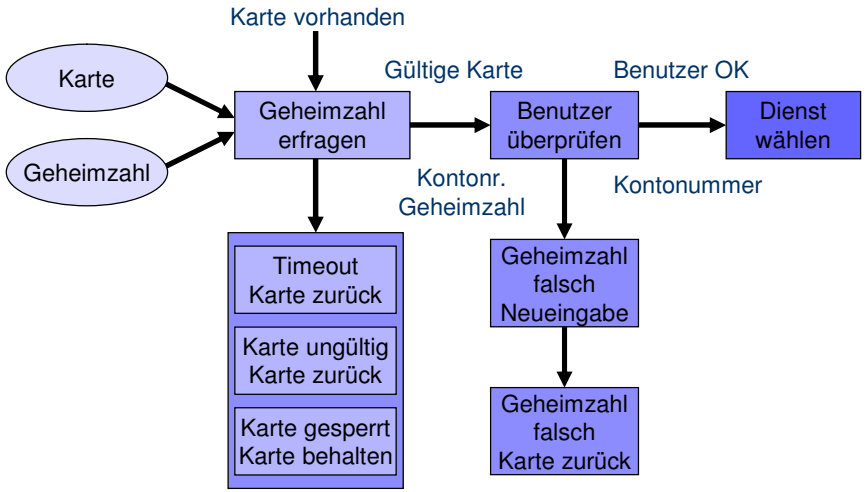
- Fehlersituationen stehen unterhalb jedes Rechtecks.

Ereignisszenarien



- Wenn es mehrere mögliche Fehlersituationen gibt, werden diese in ein weiteres Rechteck eingeschlossen.

Ereignisszenarien



- Der Name des nächsten erwarteten Ereignisses am Ende eines Szenarios steht in einem Rechteck mit einem dicken Rand.
  (In diesem Beispiel nicht verwendet.)

Vor- und Nachteile

- Für alle Blickwinkel und Dienste werden Schablonen ausgefüllt und Ereignisszenarien aufgestellt. Durch Vergleich der gesammelten Informationen können Fehler bei der Analyse und Konflikte durch mehrfache Dienstdefinitionen erkannt werden.
- Das Datenaufkommen ist erheblich, und diese Methode kann – wie andere Analysemethoden auch – in der Praxis nur mit der Unterstützung durch CASE-Tools angewendet werden.
- Der Vorteil der methodenbasierten Analyse ist, dass sie systematisch angewendet werden kann.
- Der Nachteil ist, dass die Systemmodellierung in einen (möglicherweise künstlichen) Rahmen eingezwängt wird.
- Je nach dem Problembereich können verschiedene Analysemethoden unterschiedlich gut geeignet sein.
- Da Organisationen aus Kostengründen meistens nur eine Methode anwenden, kann es vorkommen, dass die Ergebnisse nicht zufrieden stellend ausfallen.