Subjekt, Prädikat und Objekt in der Grammatik der Software

Dr. Albert Fleischmann den 18. Sep. 2008

„ Die Grenzen meiner Sprache bedeuten die Grenzen meiner Welt“

Ludwig Wittgenstein Tractatus logico-philosophicus, 5.6

„In der Logik ist nichts zufällig: Wenn das Ding im Sachverhalt vorkommen kann, so muß die Möglichkeit des Sachverhaltes im Ding bereits präjudiziert sein.“

Ludwig Wittgenstein, Tractatus logico-philosophicus, 2.012

Inhalt

1 Einleitung

Entsprechend der Struktur von natürlichen Sprachen besteht die Wirklichkeit aus Subjekten, Prädikaten und Objekten. Basierend auf diesen Elementen wird die Wirklichkeit beschrieben. Diese Struktur wird bei der menschlichen Kommunikation ob schriftlich oder mündlich laufend verwendet und ist den Menschen vertraut. Es stellt sich nun die Frage warum in formalen Sprachen zur Beschreibung von Modellen der Wirklichkeit wie z.B. Geschäftsprozessen oder Computerprogrammen nicht Subjekt, Prädikat und Objekt verwendet werden und ob damit nicht die Kommunikationslücke zwischen Fachbereichen und IT-Bereichen verkleinert werden kann.

In dieser Abhandlung wird zunächst untersucht wie sich Subjekt, Prädikat und Objekt in formalen Sprachen zur Beschreibung von Modellen der Wirklichkeit einschließlich Programmiersprachen wiederfinden und zu welchen Verständnisproblemen und Beschreibungslücken es führt wenn eines oder mehrere dieser Basiselemente fehlen.

Danach wird eine formale Methode zur Beschreibung von Modellen wie Geschäftsprozessen vorgestellt mit der sich vollständige Sätze im Sinne von natürlichen Sprachen bilden lassen. Diese Sprache PASS (Parallel Activity Specification Schema) enthält Subjekt, Prädikat und Objekt. Die Erfahrung zeigt, dass diese Modellbeschreibungssprache für nicht IT-Personen verständlicher ist und schneller verwendet werden kann als andere gängige formale Sprachen. Modelle dieser Sprache können auch ohne Programmierung automatisch in ausführbare Programme umgesetzt werden.

Am Ende wird gezeigt, dass die Hauptebenen der Service Orientierten Architektur (SOA) den Elementen der Standardsatzsemantik entsprechen. Der Benutzerzugang entspricht dem Subjekt, die Services den Prädikaten und die jeweiligen Daten den Objekten.

2 Allgemeines zu Modellbeschreibungen und Programmen

Die Entwicklung eines Anwendungsprogramms beginnt mit der Modellierung eines Ausschnitts der Wirklichkeit, welcher durch das zu entwickelnde Anwendungsprogramm unterstützt werden soll. Modelle beschreiben Eigenschaften und Verhaltensalternativen und die Wechselwirkung mit ihrer technischen und/oder organisatorischen Umgebung. Modelle werden Schritt für Schritt zu einem Programm.

Der Vorgang der Modellbildung wird im Allgemeinen als Analyse bezeichnet. Wichtig bei der Analyse ist, welche Modellelemente als wesentlich herausgestellt und betont werden und welche nur als ergänzend betrachtet werden. In (Scholz & Holl, 1999) werden die wesentlichen Modellelemente als essentiell und die Ergänzenden als akzidentell (Akzidenzien) bezeichnet.

Je nachdem, welche Modellelemente als essentiell betrachtet werden, werden bei der Anwendungsentwicklung verschiedene Ansätze zur Modellbildung verwendet. Um diese essentiellen Elemente werden dann die Akzidenzien gruppiert. Für eine heute weitgehend akzeptierte Einteilung der einzelnen Aspekte der Modellbildung in der Softwareentwicklung werden derzeit folgende Ansätze verwendet (siehe z.B. (Denert, 1991) und (Scholz & Holl, 1999)

- Beim *funktionalen Ansatz* werden Akzidenzien um Funktionen gruppiert.
  Beispiele für funktionsorientierte Modelle sind die allgemein bekannten Kontrollflussdiagramme, Datenflussdiagramme nach deMarco etc..
- Beim *datenorientierten Ansatz* werden Akzidenzien um Daten gruppiert.
  Ein allgemein bekanntes Beispiel für datenorientierte Modellierungsmethoden sind Entity Relationship Diagramme.
- Beim *objektorientierten Ansatz* werden Akzidenzien um Objekte gruppiert.

Der objektorientierte Modellierungsansatz gilt als der zur Zeit am meisten akzeptierte. Eine dazugehörige, weitestgehend bekannte Beschreibungsmethode ist UML.

Wichtig bei der Modellbildung ist, dass die erstellten Modelle beschrieben und dokumentiert werden. Nur dann haben Modelle einen Sinn. In der obigen Auflistung wurden bereits einige allseits bekannte Sprachen zum Dokumentieren der Analyseergebnisse angegeben. Modellbildung und Analysieren heißt also letztlich, einen Ausschnitt der Wirklichkeit mit einer „künstlichen“ Sprache zu beschreiben. Diese Sprachen orientieren sich an dem zu erstellenden Programm und damit mehr oder weniger an den Bedürfnissen der Informationstechnik.

Modelle sind immer auch in natürlichen Sprachen formulierbar. Der Vorteil dieser Beschreibungen ist, dass sie von allen sofort verwendet werden können und über eine vollständige Standardsatzsemantik mit Subjekt, Prädikat und Objekt verfügen[1]. Diese Standardsatzsemantik ist jedem vertraut, da er sie täglich in seiner Kommunikation benutzt.

Der Nachteil ist, dass natürliche Sprachen nicht präzise und nicht scharf genug sind um Missverständnisse zu vermeiden. Die formalen Sprachen haben dagegen eine eindeutige Wortsemantik. Allerdings wird in formalen Modellen zur Vereinfachung eine reduzierte „Satzsemantik“ verwendet. Dies führt zu großen Schwierigkeiten beim Verstehen formaler Modelle. Letztlich werden sie dann in natürlicher Sprache interpretiert, was wegen ihrer Unvollständigkeit zu Übersetzungsproblemen führen kann. Menschen sind gewohnt, in vollständigen Sätzen der Form Subjekt-Prädikat-Objekt zu kommunizieren.

In den folgenden Überlegungen wollen wir die Nachteile der natürlichen Sprache ausklammern und uns auf die Standardsatzsemantik Subjekt-Prädikat-Objekt fokussieren. Das *Subjekt* ist der Ausgangspunkt, von dem das Geschehen ausgeht, das durch das *Prädikat* bezeichnet ist. Das *Objekt* ist der Zielpunkt des durch das Prädikat bezeichneten Handelns.

Wie oben bereits erläutert wurde, wird bei der Modellbildung zwischen essentiellen und akzidentellen Aspekten unterschieden. Dies zeigt sich auch in der natürlichen Sprache. Dort werden Passivsätze verwendet, wenn es unerheblich ist, von wem eine Aktion ausgeht. Um verständliche vollständige Sätze zu bilden, wie man es gewohnt ist, empfiehlt es sich, formale Modellierungssprachen zu schaffen, die sich einer solchen vollständigen Standardsatzsemantik bedienen, um Übersetzungsprobleme zu vermeiden.

In den folgenden Ausführungen wollen wir verschiedene Modellierungsmethoden in Anlehnung an die Standardsatzsemantik betrachten. Denn auch in formalen Modellen für Anwendungssysteme muss es so etwas wie Subjekte, Prädikate und Objekte geben, denn sonst wären Erklärungen zu formalen Sprachen in einer natürlichen Sprache nicht möglich. Über die Zeit haben sich in der Modellbildung die essentiellen Aspekte vom Prädikat bis zum Objekt verschoben, während Subjekte nur rudimentär behandelt wurden. Der Ausgangspunkt einer Handlung, das Subjekt, rückt erst jetzt durch die zunehmende Bedeutung von Geschäftsprozessen in den Blickpunkt der Betrachtung.

Anwendungsprogramme die heute entwickelt werden, werden immer im Zusammenhang mit den Tätigkeiten von Mitarbeitern in einem Unternehmen gesehen. Neue oder angepasste Anwendungsprogramme sollen helfen Tätigkeiten zu beschleunigen bzw. zu verbessern. Die Tätigkeiten Einzelner werden heute in einen größeren Zusammenhang betrachtet. Mitarbeiter in einem Unternehmen arbeiten bei der Erbringung der einzelnen Leistungen zusammen. Die Definition dieser Zusammenarbeit erfolgt in Form von Geschäftsprozessen. Diese legen fest was von wem in welcher Reihenfolge erledigt werden muss, wobei bestimmte Tätigkeiten natürlich automatisiert durch Anwendungsprogramme erledigt werden können. Mitarbeiter kom-

[1]Was hier als Standardsatzsemantik bezeichnet wird ist in der die zweithöchste Ebene der Satzsemantik mit den semantischen Rollen Agens, Prädikation und Thema (Max spielt den Ball). Die Ebene eins entspricht Aussagen wie „der Ball ist rund“. Die letzte und dritte Ebene entspricht satzsemantischen Strukturen innerhalb von Satzgliedern (Peters Freude am Fussball brachte Glück). Einzelheiten siehe Schmidt et.al. 2005.

munizieren untereinander oder benutzen Anwendungsprogramme bzw. andere Werkzeuge wie z.B. Papier und Bleistift. Die Kommunikation der Menschen untereinander kann akustisch, mit Papier oder elektronisch etc. erfolgen. In solchen Prozessen gibt es also Handelnde z.B. die Mitarbeiter, Prädikate, die die Tätigkeiten der Mitarbeiter definieren und Objekte, die das Ziel dieser Tätigkeiten darstellen. Eine wesentliche Tätigkeit der Mitarbeiter ist natürlich die Kommunikation untereinander.

Geschäftsprozesse sind eine Abfolge von Geschehnissen in einem Unternehmen, die in Form eines Modells beschrieben werden.

In dieser Abhandlung soll am Beispiel eines Prozesses zur Beantragung von Urlaub dargestellt werden, in welchen in der Informatik üblichen Modellen welche Teile der Standardsemantik Subjekt-Prädikat-Objekt essentiell und welche akzidentell sind und wie der Beispielsprozess in der jeweiligen Modellierungsart beschrieben wird.

Die Abbildung 1 zeigt die natürlich sprachliche Beschreibung des Urlaubsantragsprozesses. Dieses natürlich sprachliche Modell wird nun in verschiedenen formalen bzw. semiformalen Modellierungsmethoden beschrieben.

Urlaubsprozess:
Ein Mitarbeiter füllt den Urlaubsprozess aus. Er gibt an wann der Urlaub beginnen soll und wann er endet. Dier Urlaubsantrag wird dann vom Vorgesetzten geprüft. Der Vorgesetzte informiert den Mitarbeiter ob er den Urlaub genehmigt oder ablehnt.
Genehmigte Urlaubsanträge gehen an die Verwaltung und diese aktualisiert das Urlaubskonto des Mitarbeiters.

Abbildung 1: Natürlich sprachliche Beschreibung des Urlaubsantragsprozesses

Am Ende der Untersuchung von verschiedenen repräsentativen Sprachen zur Beschreibung von Modellen wird ein Konzept vorgestellt in dem Subjekt, Prädikat und Objekt gleichberechtigt neben einander stehen. Damit wird es möglich ein Modell der Wirklichkeit zu bilden, das in einer vollständigen Grammatik so beschrieben wird wie es in jeder natürlichen Sprache möglich ist , aber trotzdem so präzise ist, dass daraus automatisch der entsprechende Programmcode generiert werden kann, der problemlos auf die heute gängige Service Orientierte Architektur abgebildet werden kann.

3 Das Prädikat oder die Handlung

Am Anfang der Datenverarbeitung war die Aktion, also das Prädikat, der essentielle Aspekt. Das Subjekt und Objekt wurden als Akzidenzien betrachtet. Der Grund dafür, dass am Anfang der Datenverarbeitung die Algorithmik im Mittelpunkt stand.Die ersten Rechner wurden entworfen um komplexe Rechenprobleme zu lösen. Konrad Zuse wollte als gelernter Bauingenieur seine Statikberechnungen automatisieren. Daten spielten Anfangs in der Informatik nur eine untergeordnete Rolle.

3.1 Flussdiagramme

Eine der ersten Methoden, um Modelle für solche algorithmischen Aufgaben zu beschreiben, waren Flussdiagramme oder Programmstrukturpläne. Flussdiagramme beschreiben eine Folge von Operationen oder Tätigkeiten zur Lösung einer Aufgabe. Solche Flussdiagramme können dann zur Beschreibung des Arbeitsablaufs bei der Abarbeitung eines Prozesses, z.B. Beantragung von Urlaub verwendet werden.

Werden Flussdiagramme zur Beschreibung eines Rechenalgorithmus verwendet, so ist klar, wer die einzelnen Aktionen in dem Flussdiagramm anstößt. Es ist die ausführende Person

bzw. der ausführende Rechner. Diese Standardsubjekte werden nicht explizit erwähnt. Auch die Daten, die bei der Ausführung eines Flussdiagramms benötigt werden, werden nur sehr rudimentär angegeben.

Werden Flussdiagramme zur Beschreibung eines Modelles verwendet, erlauben sie es jedoch nur unzureichend, meistens durch natürlichsprachliche Ergänzungen, Subjekte und Objekte in das Modell einzubringen.

Die Abbildung 2 zeigt ein Beispiel für die sicherlich allseits bekannten Flussdiagramme. Im linken Flussdiagramm sind nur die Aktionen und teilweise die betroffenen Objekte enthalten. Das rechte Beispiel enthält in den Aktionen sowohl die Subjekte als auch die Objekte in den Tätigkeitssymbolen (Rechtecke). Allerdings wurden diese Aspekte in natürlichsprachlicher Form hinzugefügt.

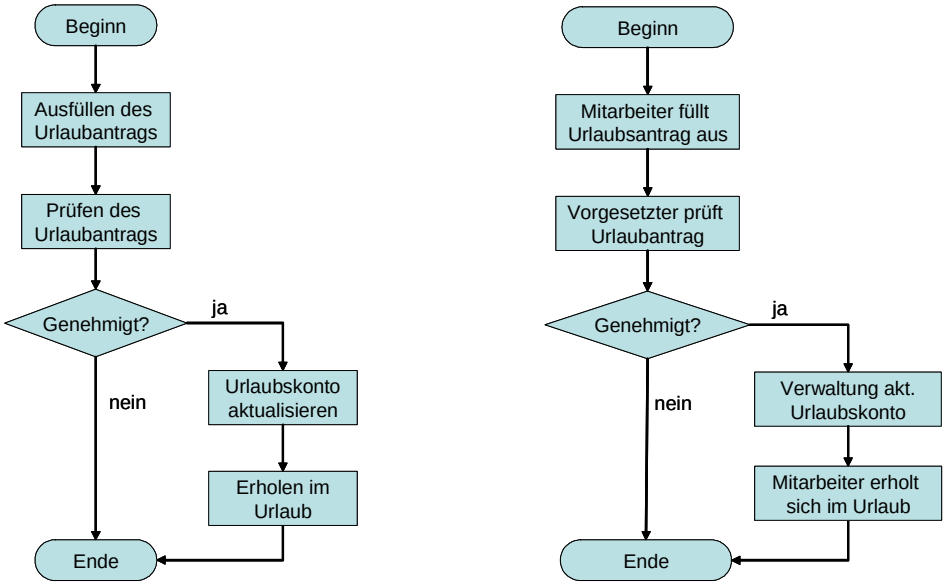

Abbildung 2: Beispiel Flussdiagramm

In erweiterten Formen von Flussdiagrammen werden neben den Verben auch die Subjekte und Objekte als Symbole direkt oder indirekt mit angegeben. Das folgende Bild zeigt dasselbe Flussdiagramm, das indirekt die Subjekte als Eingaben enthält und die Objekte durch entsprechende Symbole darstellt.

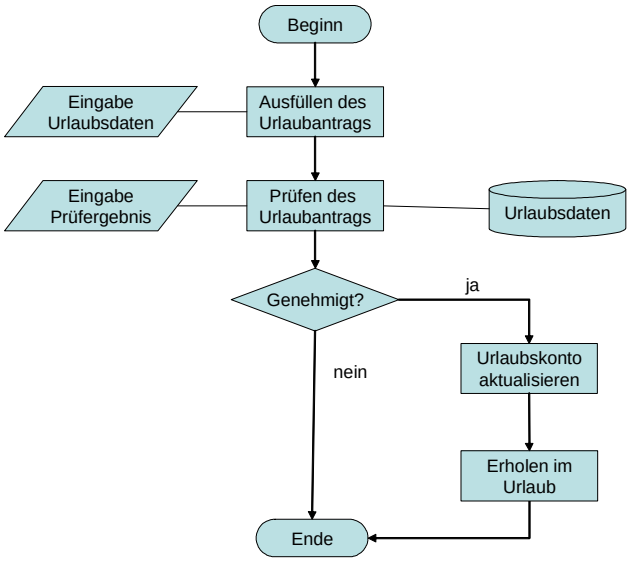

Abbildung 3: Flussdiagramm mit Daten (Objekten)

3.2 Ereignisgesteuerte Prozessketten

Eine kontrollflussbasierte Methode zur Darstellung von Geschäftsprozessen sind ereignisgesteuerte Prozessketten. Das folgende Bild zeigt den Prozess des Urlaubsantrags als EPK.

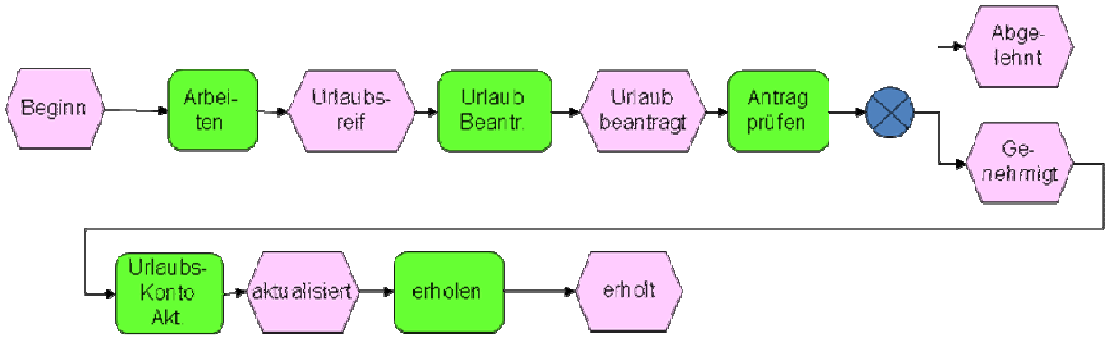

Abbildung 4: Urlaubsantragsprozess als EPK

Die Vierecke repräsentieren die Aktionen eines Prozesses. Wobei zur Verdeutlichung die natürlichsprachliche Beschriftung der Aktionen die Objekte enthalten kann. Den einzelnen Funktionen geht immer ein Ereignis voraus (Sechsecke), das den Anstoß zur Ausführung einer Funktion repräsentiert bzw. für die vorausgehende Funktion das Ergebnis. Mit Hilfe von Konnektoren können die Ergebnisse einer Funktion zu unterschiedlichen Ereignissen führen. Die Funktion „Antrag prüfen“ kann entweder zum Ereignis „Abgelehnt“ oder „Genehmigt“ führen (XOR). Neben dem XOR gibt es noch weitere Konnektoren. Einzelheiten zu EPKs und ihren Einsatz finden sie in (Scheer, 2001).

In der Praxis werden EPKs fast nie eingesetzt, vielmehr werden erweiterte EPKs (eEPKs) verwendet. eEPKs wurden um Elemente der Organisations-, Daten- und Leistungsmodellierung erweitert. Diese Erweiterungen entsprechen im Wesentlichen den Subjekten und Objekten. Organisationseinheiten sind der Ausgangspunkt von Handlungen und Daten sind deren Zielpunkt. Das folgende Bild zeigt eine erweiterte EPK des Urlaubsantragprozesses.

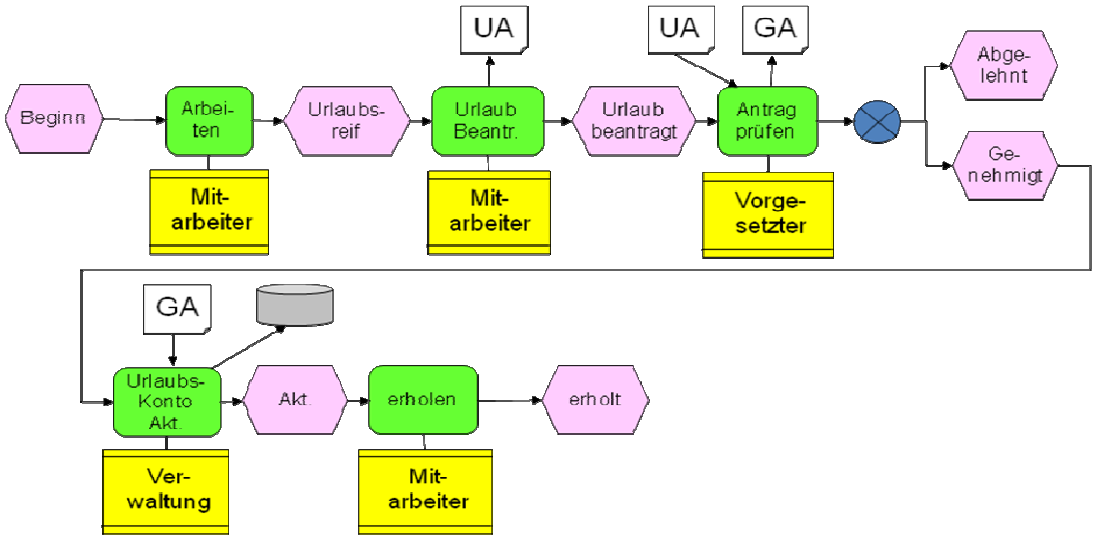

Abbildung 5: eEPK mit Subjekt, Prädikat und Objekt

3.3 Petrinetze

Eine in der theoretischen Informatik sehr wichtige Form der Modellbildung sind Petrinetze (Einen Überblick mit entsprechenden Literaturhinweisen findet sich z.B. in Wikipedia: Petrinetze). Petrinetze sind eine aktionsorientierte Modellbildungsmethode, d.h. Petrinetze sind prädikatorientiert. Sie erlauben im Gegensatz zu Kontrollflussdiagrammen, dass mehrere Aktionen parallel ausgeführt werden können.

Um auch Datenaspekte zu unterstützen wurden attributierte Petrinetze entwickelt. Allerdings gibt es keine Ansätze, Subjekte in Petrinetzen zu repräsentieren.

Die Abbildung 6 zeigt ein Petrinetz das den Urlaubsprozess beschreibt. Ein Petrinetz besteht aus Stellen, Transitionen, Übergängen zwischen Transitionen und Stellen, wobei sich Stellen und Transitionen abwechseln müssen und einer Anfangsmarkierung. Im Allgemeinen werden Transitionen als Aktionen interpretiert und Stellen als die Anfangsbedingung damit eine Transition schalten kann. Eine Transition kann schalten, wenn sich in ihren Eingangsstellen mindestens ein Token befindet. Nach dem Schalten erhält jede Ausgangsstelle einen Token. Die Anfangsmarkierung legt fest, welche Stellen beim Start mit Tokens belegt sind. In Abbildung 6 enthält als Anfangsmarkierung die Stelle *Mitarbeiter hat Urlaub* einen Token.

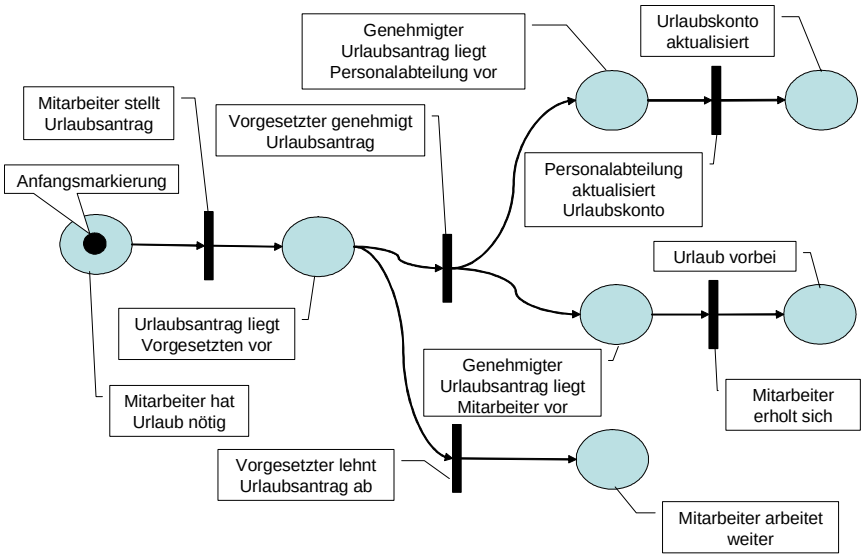

Abbildung 6: Urlaubsantrag als Petrinetz mit Anfangsmarkierung

Nach dem Schalten der Transition *Mitarbeiter stellt Urlaubsantrag* ergibt sich eine Tokenbelegung wie sie in Abbildung 7 dargestellt ist. Der Token wird von der Stelle *Mitarbeiter hat Urlaub nötig* abgezogen und ein Token erscheint in der Stelle *Urlaubsantrag liegt Vorgesetzten vor*.

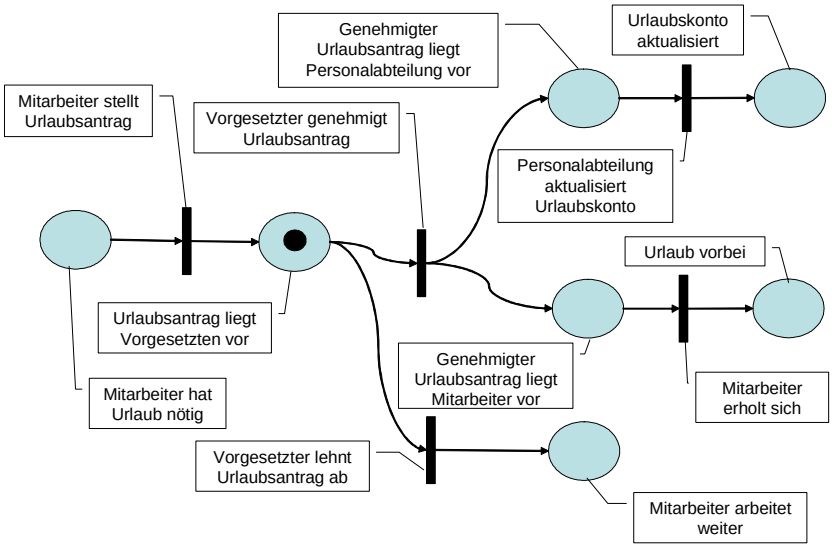

Abbildung 7: Tokenbelegung nach dem Schalten von „Mitarbeiterstellt Urlaubsantrag“

Danach kann entweder die Transition *Vorgesetzter lehnt Urlaubsantrag ab* oder die Transition *Vorgesetzter genehmigt Urlaubsantrag* schalten. Das Petrinetz ist somit nicht deterministisch. Schaltet die Transition *Vorgesetzter genehmigt Urlaubsantrag*, so werden die Stellen *Genehmigter Urlaubsantrag liegt Personalabteilung vor* und *Genehmigter Urlaubsantrag liegt Mitarbeiter vor* mit einen Token belegt (siehe Abbildung 8).

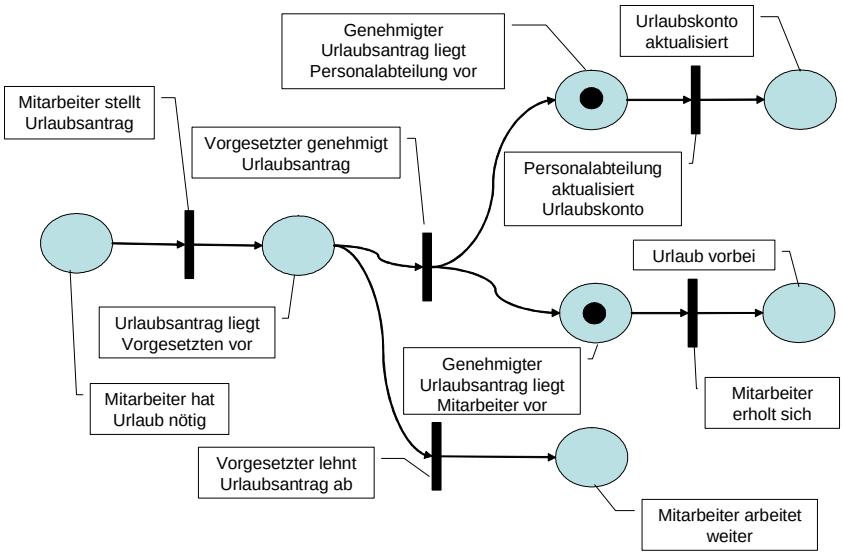

Abbildung 8: Tokenbelegung nach dem Schalten von „Vorgsetzter genehmigt Urlaubsantrag“

Das obige Beispiel zeigt, dass Petrinetze die Reihenfolge der Aktionen in den Mittelpunkt stellen. Die Subjekte und Objekte kommen durch entsprechende natürlichsprachliche Ergänzungen hinzu. In diesem Fall geschieht dies durch die Wahl von entsprechenden Bezeichnungen für die Stellen und Transitionen. Der Vorteil von Petrinetzen gegenüber Flussdiagrammen ist, dass sie theoretisch fundiert sind und Nebenläufigkeit dargestellt werden kann.

4 Das Objekt, oder das Ziel der Handlung

Mit dem zunehmenden Einsatz von Rechnern in der Industrie wurde der Aspekt der Datenverarbeitung immer wesentlicher. In Unternehmen werden umfangreiche Datenbestände gespeichert und bearbeitet. Dies bedeutet, dass der Aspekt *Objekt* in den Mittelpunkt der Betrachtung rückte, d.h. essentiell wurde. Um diesen Anforderungen gerecht zu werden, wurden Modellierungssprachen entwickelt, die das Ziel der Handlung, die Objekte bzw. Daten, in den Mittelpunkt rückten.

4.1 Entity-Relationship-Modell

Das Entity-Relationship-Modell (ER-Modell oder ERM) dient dazu, im Rahmen der Modellbildung, die in dem betrachteten Ausschnitt der Wirklichkeit vorhandenen Daten mit ihren Beziehungen zu beschreiben. Das ER-Modell besteht meistens aus einer Grafik und einer Beschreibung der darin verwendeten einzelnen Elemente. Für die Grafiken können unterschiedlich Darstellungsformen verwendet werden.

Grundlage der Entity-Relationship-Modelle ist die Typisierung von Datenelementen und deren Beziehungen untereinander. Bei der Erstellung wird aber meistens mit den konkreten Entitäten und deren Beziehungen gearbeitet:

- Entität: Objekt der Wirklichkeit, materiell oder abstrakt (zum Beispiel Angestellter „Meyer“, Vorgesetzter „Huber“,)
- Beziehung (Relationship): semantische Beziehung zwischen zwei Objekten (zum Beispiel „Angestellter Meyer ist Mitarbeiter von Vorgesetzten Huber “)

Das Modell selbst besteht immer ausschließlich aus Entitätstypen und Beziehungstypen.

- Entitätstyp: Typisierung gleichartiger Entitäten (zum Beispiel Angestellter, Vorgesetzter)
- Beziehungstyp: Typisierung gleichartiger Beziehungen (zum Beispiel „ist Mitarbeiter von“). Die inhaltliche Bedeutung der Beziehungstypen zwischen Entitätstypen kommt

im ER-Diagramm durch einen kurzen Text als Beschriftung der Kante zum Ausdruck, wobei es dem Ersteller freigestellt ist, welche Bezeichnung er verwendet.

Das folgende Bild zeigt das ERM des Urlaubsantragsprozesses. Jeder Mitarbeiter hat nur genau einen Vorgesetzten und jeder Vorgesetzter ist Chef von ein bis N Mitarbeitern. Jeder Mitarbeiter hat keinen oder U Urlaubsanträge gestellt. Jeder Urlaubsantrag enthält genau ein Datum für den Urlaubsbeginn und das Urlaubsende. Ein Vorgesetzter hat 0 bis M Urlaubsanträge zu prüfen.

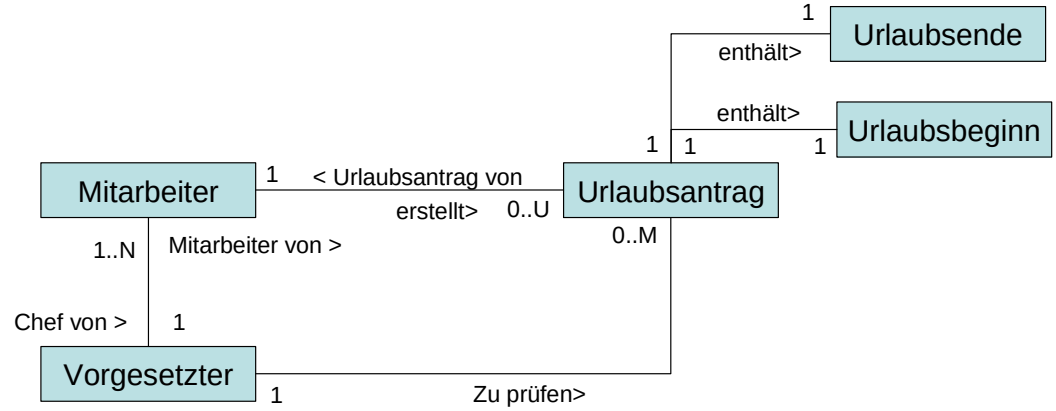

Abbildung 9: ERM für den Urlaubsantragsprozess

ERM fokussiert auf Objekte. Prädikate und Subjekte kommen nur indirekt durch die Bezeichnung der Beziehungen in das Spiel. Wird für eine Beziehung ein Prädikat verwendet, kann ein vollständiger Satz entstehen. Wie das obige Beispiel zeigt, ist es nicht vorgeschrieben, dass zur Bezeichnung einer Beziehung ein Prädikat benutzt werden muss. Subjekt und Prädikat werden deshalb nur durch die Disziplin des Modellierers eingeführt. Ein ERM enthält keinerlei Kontrollfluss, d.h. daraus ist nicht ersichtlich, wann welche Aktionen ausgeführt werden (Prädikat) und nur bei der disziplinierten Verwendung von entsprechenden Bezeichnungen der Beziehungen kann geschlossen werden, wer der Ausgangspunkt einer Aktion (Subjekt) ist.

4.2 Relationales Datenmodell

Bei relationalen Datenmodellen werden wie beim ERM nur die Datenobjekte betrachtet. Subjekt und Prädikat sind Akzidenzien.

Als Strukturelemente stehen bei relationalen Datenmodellen ausschließlich Relationen zur Verfügung, die sich durch Tabellen darstellen lassen. Die einzelnen Zeilen der Tabellen bilden die Datensätze und die jeweiligen Spalten entsprechen den einzelnen Datenfeldern. Ein Datenmodell besteht in der Regel aus mehreren Tabellen, wobei Beziehungen zwischen beliebigen Datensätzen auch in verschiedenen Tabellen eines Modells über gleiche Feldinhalte hergestellt werden. Der Zugriff auf bestimmte Datensätze erfolgt über die Feldinhalte. Die Abbildung 10 zeigt ein Datenmodell für unser Urlaubsantragsproblem, wobei die Aktualisierung des Urlaubskontos aus Platzgründen nicht berücksichtigt ist. Das Datenmodell besteht aus den drei Tabellen *Mitarbeiter*, *Vorgesetzte* und *Urlaubsanträge*. Die Tabelle *Vorgesetzte* enthält alle Vorgesetzten, die Tabelle *Mitarbeiter* enthält alle Mitarbeiter mit einem Verweis auf ihren Vorgesetzten in der Spalte *V-Nr.*. Die Tabelle *Urlaubsanträge* enthält alle gestellten Urlaubsanträge. Die Spalte *MA-Nr.* in der Tabelle *Urlaubsanträge* enthält einen Verweis auf den Mitarbeiter, der diesen Urlaubsantrag gestellt hat.

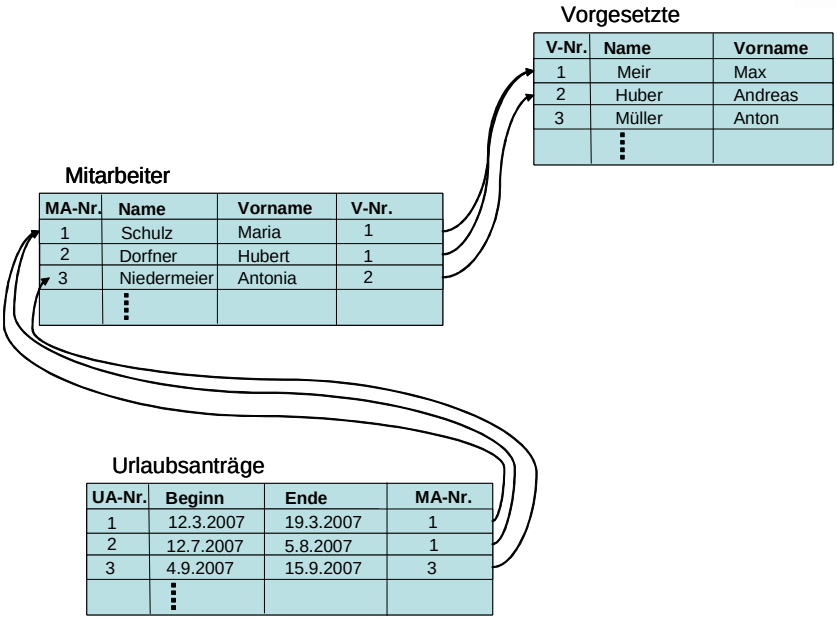

Abbildung 10: Relationales Datenmodell für den Urlaubsantragsprozess

Auf relationalen Datenmodellen sind logische, mengentheoretischen Abfragen definiert (Prädikate), die von den Benutzern (Subjekte) verwendet werden. Ein relationales Datenmodell enthält nicht, welche Benutzer (Subjekte) es in dem betrachteten Ausschnitt der Wirklichkeit gibt. Die möglichen Prädikate, die von den Benutzern angestoßen werden, sind durch die sogenannte Abfragesprache, im allgemeinen SQL, festgelegt.

Im obigen Beispiel ermittelt der Vorgesetzte *Meier, Max* (ein Benutzer, Subjekt) durch eine entsprechende Abfrage (Prädikat) seine Mitarbeiter aus der Tabelle *Mitarbeiter* (Objekte). Die Mitarbeiter von Meier, Max sind alle diejenigen in der Tabelle *Mitarbeiter,* die in der Spalte *V-Nr.* eine 1 enthalten. Danach werden in der Tabelle *Urlaubsanträge* alle Urlaubsanträge identifiziert, die in der Spalte *MA-Nr.* eine Nummer eines Mitarbeiters von Meier, Max enthalten. Somit erhält Meier, Max alle Urlaubsanträge seiner Mitarbeiter, die dann entsprechend bearbeitet werden können.

Relationale Datenmodelle sind sehr nahe an der Implementierung. Sie können mehr oder weniger direkt durch eine relationale Datenbank realisiert werden, so dass als Modellierungssprache ERM verwendet wird und das relationale Modell bereits als Programmierung gilt. In beiden Modellsprachen gilt aber, dass Subjekte nur sehr am Rande betrachtet werden. Für eine Datenbankanwendung gibt es immer nur den Anwender, wer immer das ist. Der Subjektgedanke kommt durch Berechtigungskonzepte in das Spiel: Welcher Benutzer darf auf welche Daten wie zugreifen?

Durch die Abfragesprache für relationale Sprachen ist das Prädikat vorhanden, während es bei ERM vollständig fehlt.

5 Prädikat und Objekt, oder die Handlung und das Ziel der Handlung

In den oben beschriebenen Modellierungsmethoden wurde entweder das Objekt oder das Prädikat vernachlässigt. Bei den prädikatzentrierten Methoden gab es Probleme bei der Programmierung, weil die Objektaspekte unzureichend beschrieben waren. Bei den objektunterstützenden Methoden gab es Probleme mit den Prädikaten, da es zwar eine Abfragesprache

gab, mit der Prädikate gebildet werden können, aber keine Möglichkeiten, Kontrollflüsse (d.h. Sequenzen von Prädikaten) zu definieren.

Es lag deshalb nahe, Modellierungskonzepte zu entwickeln, die Aktions- und Datenaspekte gleichwertig betrachten, d.h. Modellierungssprachen, die Prädikate und Objekte enthalten. In gewisser Weise können damit vollständige Sätze im Sinne der Standardsatzsemantik gebildet werden, nämlich Passivsätze. Passivsätze werden in natürlichen Sprachen benutzt, wenn das Subjekt eine untergeordnete Rolle spielt. Eine Passivbeschreibung des Urlaubsantragsprozesses würde in etwa wie folgt aussehen: Der Urlaubsantrag wird ausgefüllt, der Urlaubsantrag wird geprüft, das Prüfergebnis wird dokumentiert, das Urlaubskonto wird aktualisiert.

5.1 Datenflussdiagramm – Strukturierte Analyse (SA) nach deMarco

Mit Datenflussdiagrammen wird der Fluss von Daten zwischen Funktionen, Datenspeichern und externen Beteiligten, die nicht zum Betrieb des Systems gehören, dargestellt. Die Strukturierte Analyse nach Tom DeMarco ist die Anwendung der Datenflussdiagramme zur Modellbildung.

In Datenflussdiagrammen (DFD) werden die folgenden grafischen Elemente verwendet:

- Externe Schnittstelle (Externer Partner, Beteiligter, Terminator).

Externe Schnittstellen werden als Rechtecke dargestellt. Sie stehen für die Beziehungen des betrachteten Systems zu dessen Außenwelt. Sie senden oder empfangen die Daten, verarbeiten diese jedoch nicht. Externe Schnittstellen stoßen durch das Bereitstellen von Daten das System an und können somit unter gewissen Einschränkungen als Subjekte betrachtet werden.

- Funktion (Prozess, Aufgabe, Function, Process)

Funktionen werden als Kreise oder Ovale dargestellt. Sie haben die Aufgabe, Eingabedaten in Ausgabedaten zu verarbeiten und enthalten die dazu notwendigen Algorithmen. Die Funktionen entsprechen in unserer Satzsemantik den Prädikaten. Die höheren komplexen Prädikate werden dann noch durch die Prädikate eines Kontrollflussdiagramms verfeinert.

- Datenspeicher (Speicher, Store)

Datenspeicher werden als zwei Parallele dargestellt. Sie bilden eine Ablagemöglichkeit für Daten, bei denen sich der Erstellungszeitpunkt vom Gebrauchszeitpunkt unterscheidet. Sie können als spezielle Funktionen zum Speichern von Daten betrachtet werden.

- Datenfluss (Informationsfluss, Dataflow)

Der Datenfluss wird durch Pfeile zwischen Funktionen bzw. Datenspeicher dargestellt. Die Pfeile werden mit den Namen der fließenden Daten bezeichnet. In einem Datenlexikon werden die Strukturen aller verwendeten Informationen definiert. Die Definition der Datenstrukturen erfolgt in Backus-Naur-Form. Hier könnte natürlich auch ein ERM verwendet werden. Die Daten entsprechen den Objekten der Standardsatzsemantik.

- Kontextdiagramm

Das folgende Bild zeigt das Kontextdiagramm der Urlaubsantragsbearbeitung. Im Kontextdiagramm werden die externen Schnittstellen identifiziert und das zu erstellende System als eine Funktion dargestellt. Das Kontextdiagramm zeigt, dass die Anwendung Daten von einer externen Schnittstelle erhält und das Ergebnis an diese externe Schnittstelle zurückgibt. In diesem Beispiel kann die externe Schnittstelle als Subjekt betrachtet werden. Allerdings fehlt hier der Vorgesetzte, da dieser ja Teil des Systems ist. Verlagert man den Vorgesetzten und die Aktualisierung der Urlaubsdaten auch nach Extern bleibt von der Anwendung nichts mehr übrig, da alles extern ist.

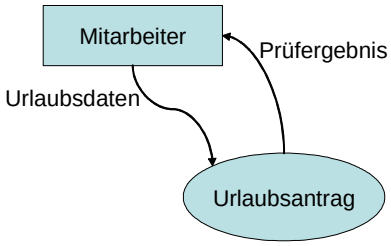

Abbildung 11: Kontextdiagramm für den Urlaubsantragsprozess

Die folgende Abbildung 12 zeigt die Verfeinerung der Urlaubsbearbeitung. Sie zeigt den Datenfluss zwischen den einzelnen Funktionen und Datenspeichern. Wichtig ist, dass mit dem Datenfluss kein Kontrollfluss verbunden ist, wobei ein Kontrollfluss durchaus suggeriert wird.

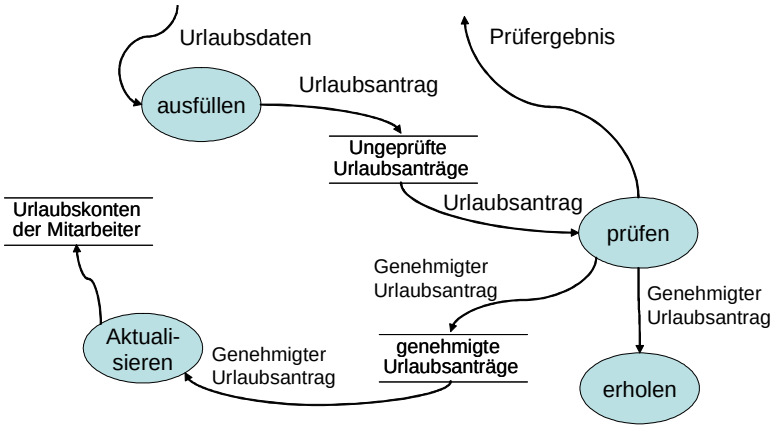

Abbildung 12: Urlaubsantragsprozess als Datenflussdiagramm

Obwohl Datenflussdiagramme bereits in den siebziger Jahren entwickelt wurden, decken sie Prädikat und Objekt der Standardsatzsemantik ab. Subjekte können nur über Hilfskonstruktionen, die zu Verfälschungen führen, eingeführt werden. Datenflussdiagramme werden heute in der Praxis nicht mehr eingesetzt. Die Kombination Prädikat-Objekt hat sich weiterentwickelt und führte zu den objektorientierten Modellierungs- und Implementierungsmethoden.

5.2 Objektorientierte Methoden

Die Grundidee der objektorientierten Programmierung ist, Funktionen, die auf diese Daten angewendet werden können, möglichst eng mit den zu verarbeitenden Daten und ihrer Eigenschaften zusammenzufassen und nach außen hin zu kapseln. Die Funktionen zusammen mit den Daten bilden ein Objekt im Sinne der objektorientierten Modellierung. Auf die Daten eines Objekts kann nur mit den entsprechenden Methoden zugegriffen werden. Ergänzt wird dies noch durch das Konzept der Klasse, bei dem Objekte aufgrund ähnlicher Eigenschaften zusammengefasst werden. Aus einfachen Objekten (bzw. Klassen) können durch entsprechende Operationen wie Vererbung, Polymorphie, Aggregation, Assoziationen usw. komplexe Objekte und Klassen aufgebaut werden.

Weitere Einzelheiten zu objektorientierten Methoden findet man in der sehr zahlreichen Literatur. Objektorientierte Ansätze sind der aktuelle Stand der Modellbildung und Programmierung.

Im Vergleich zu Ansätzen, bei denen Eigenschaften und Funktionen nicht gemeinsam betrachtet werden, erhebt dieses Modellierungsparadigma den Anspruch, die reale Welt besser nachzubilden.

Die Objektorientierte Modellierungsansatz deckt die Konzepte Prädikat und Objekt der Standardsatzsemantik ab. Ein Objekt besteht aus Daten und Funktionen. Dabei entsprechen die

Funktionen des Objekts den Prädikaten, während die Daten dem Objekt der Standardsatzsemantik entsprechen.

Das folgende Bild zeigt das Objekt *Urlaubsantrag* mit den Daten *Urlaubsbeginn*, *Urlaubsende* und *Prüfergebnis* sowie den Funktionen *Ausfüllen*, *Prüfen* und *Prüfergebnis* eintragen. Ist das Prüfergebnis *Urlaub wird genehmigt* wird das Urlaubskonto aktualisiert, wobei das Urlaubskonto durch ein eigenes Objekt dargestellt wird.

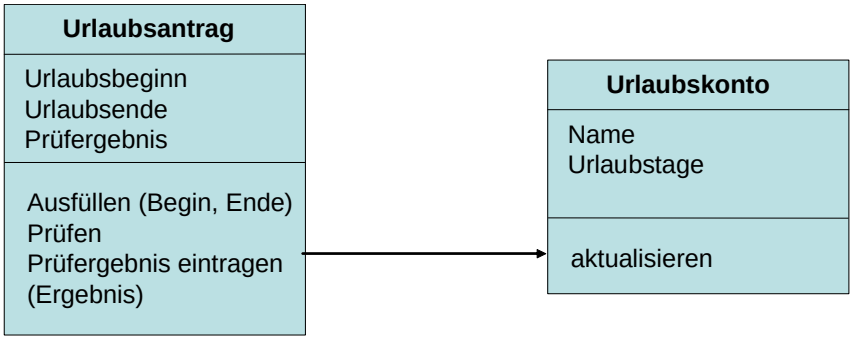

Abbildung 13: Objekt bzw. Objektklasse Urlaubsantrag

Das Objekt Urlaubsantrag erlaubt es nun, unvollständige Sätze der Form *Ausfüllen Urlaubsantrag* oder *Urlaubsantrag prüfen* zu formulieren. Um jedoch vollständige Sätze bilden zu können, gab es in den ursprünglichen objektorientierten Ansätzen nur die Möglichkeit, Subjekte durch entsprechende natürlichsprachliche Ergänzungen in das Modell einzufügen.

Durch die Einführung von Use Cases bzw. Anwendungsfalldiagrammen, wie sie in UML (Unified Modelling Language) enthalten sind, sollte dieser Nachteil beseitigt werden. UML ist eine von der Object Management Group (OMG) entwickelte und standardisierte Sprache für die Modellierung von Software und anderen Systemen. UML enthält 13 verschiedene Diagrammtypen. Einer dieser Diagrammtypen ist der Use Case bzw. das Anwendungsfalldiagramm. Weitere Einzelheiten zu UML sind in der reichlich vorhandenen Literatur zu finden.

Die Einführung der Subjekte in die Grammatik der Modellbildung durch Anwendungsfalldiagramme wird in einem späteren Abschnitt betrachtet.

6 Das Subjekt, oder der Ausgangspunkt einer Handlung

In der Informatik gibt es das Konzept der parallelen Prozesse. Ein Prozess führt in einem bestimmten Zeitintervall Aktionen aus, um ein bestimmtes Ziel zu erreichen (Havey, 2005). Eine Prozessbeschreibung definiert das Verhalten eines Prozesses.

In der Standardsatzsemantik ist das Subjekt der Ausgangspunkt der durch das Prädikat definierten Handlung. Somit repräsentieren Subjekte die aktiven Elemente der Wirklichkeit. Subjekte können definierte Reihenfolgen von Aktionen (Prädikate) ausführen. Subjekte sind unabhängig voneinander und kommunizieren bei Bedarf miteinander, d.h. sie tauschen Informationen aus. Subjekte entsprechen daher in weiten Teilen den Prozessen der Informatik. Mit dem Prozesskonzept besteht die Möglichkeit, die Subjekte aus der Wirklichkeit auf ein entsprechendes Konstrukt im Modell abzubilden.

In den folgenden Abschnitten werden zwei Konzepte vorgestellt, die Prozesse in das Zentrum der Betrachtung stellen. Dabei werden parallele Prozesse definiert, die sich über den Austausch von Nachrichten synchronisieren, d.h. ein Prozess kann Nachrichten über sogenannte Ports senden oder empfangen. Senden und Empfangen sind somit die einzig möglichen Prädikate. In gewisser Weise können zum Nachrichtenaustausch verwendeten Ports als Objekte der Standardsatzsemantik interpretiert werden.

6.1 Calculus of Communicating Systems

Calculus of Communicating Systems (CCS) ist eine von Robin Milner (Milner, 1980) entwickelte Prozessalgebra. Eine Prozessalgebra dient zur algebraischen Modellierung von parallelen Prozessen und besteht aus elementaren Aktionen und Verknüpfungsoperatoren für Aktionen. Elementare Aktionen können nicht weiter detailliert werden.

Prozesse können mit dem Nachbarn interagieren oder unabhängig parallel Aktionen ausführen. Ziel von CCS ist die Modellierung der Kommunikation zwischen Prozessen z.B. auf Äquivalenz.

Ein Prozess benutzt Ports, über die er mit anderen Prozessen kommuniziert, wobei jeder Port einen Namen hat. Es wird zwischen Sende- und Empfangsports unterschieden. Die folgende Abbildung 14 zeigt die einzelnen Prozesse bzw. Subjekte aus dem Urlaubsantragsprozess. Der Mitarbeiter sendet den Urlaubsantrag an den Vorgesetzten. Bei Sendeports wird der Portnamen mit einem Querstrich versehen. Der Vorgesetzte sendet das Prüfergebnis an den Antragsteller und u.U. den genehmigten Urlaubsantrag an die Verwaltung.

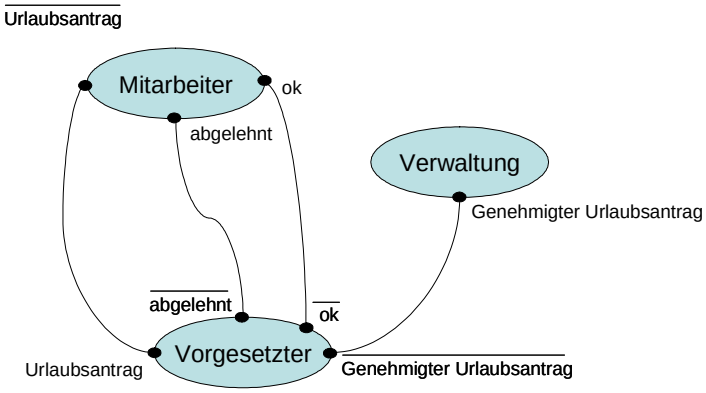

Abbildung 14: CCS Prozesse im Urlaubsantrag

In der Abbildung 14 werden nur die beteiligten Prozesse und deren Beziehungen gezeigt. Das interne Verhalten ist noch nicht sichtbar. Das interne Verhalten wird mit verschiedenen Operatoren beschrieben. In diesem Beispiel werden nur Einige benutzt, für eine vollständige Liste sei auf die angegebene Literatur verwiesen. Die Abbildung 15 zeigt die Verhaltensbeschreibung der einzelnen Prozesse und deren Verknüpfung zum Urlaubsantragsprozess.

$$\text{Mitarbeiter} = \overline{\text{Urlaubsantrag}}.(\text{abgelehnt} + \text{ok}).\text{NIL}$$

$$\text{Vorgesetzter} = \text{Urlaubsantrag}.(\overline{\text{abgelehnt}} + \overline{\text{ok}}.\overline{\text{Genehmigter Urlaubsantrag}}).\text{NIL}$$

$$\text{Verwaltung} = \text{Genehmigter Urlaubsantrag}.\text{NIL}$$

$$\textbf{Urlaubsprozess} = \textbf{Mitarbeiter} \mid \textbf{Vorgesetzter} \mid \textbf{Verwaltung}$$

Abbildung 15: Beschreibung des Urlaubsantragsprozesses in CCS

Gemäß Abbildung 15 sendet der Prozess *Mitarbeiter* zunächst den Urlaubsantrag und danach wartet er entweder auf die Nachricht *abgelehnt* oder *ok*. Erhält der Mitarbeiter einer dieser Nachrichten, führt er die Operation *NIL* aus, was bedeutet dass der Prozess stoppt. Die Beschreibung der Prozesse *Vorgesetzter* und *Verwaltung* kann ähnlich interpretiert werden.

Die letzte Zeile in Abbildung 15 zeigt die Komposition des Gesamtprozesses mit dem entsprechenden Operator.

Das Urlaubsbeispiel zeigt, dass in CCS das aktive Element, der Akteur, als essentiell gesehen wird, während Prädikat und Objekt nur eine untergeordnete Rolle spielen. So gesehen kann CCS als eine subjektorientierte Methode betrachtet werden.

6.2 Communicating Sequential Processes

Communicating Sequential Processes (CSP) ist ebenfalls eine Prozessalgebra. Sie wurde von Tony Hoare (Hoare, Communicating Sequential Processes, 1984) entwickelt. CSP wurde zunächst als programmiersprachliches Konstrukt veröffentlicht (Hoare, Communicating Sequential Processes, 1978) und wurde dann in den folgenden Jahren auch durch den Einfluss von Milner formalisiert. Zunächst wird bei CSP, im Gegensatz zu CCS, nicht zwischen Senden und Empfangen unterschieden. Werden Prozesse durch entsprechende Operatoren verknüpft, werden auch Ereignisse gleichen Namens aus den verbundenen Prozessen verknüpft.

Die Abbildung 16 zeigt den Urlaubsantragsprozess in CSP beschrieben. Beim Mitarbeiter ist das Ereignis *Urlaubsantrag* möglich und danach entweder das Ereignis *abgelehnt* oder *akzeptiert*. Das Ereignis *SKIP* beschreibt, dass der Prozess beendet ist. Im Prozess *Vorgesetzter* ist ebenfalls das Ereignis *Urlaubsantrag* möglich und danach entsprechende Folgeereignisse. Wird nun der Prozess *Mitarbeiter* mit dem Prozess *Vorgesetzter* durch den || Operator verknüpft, so haben beide dasselbe Anfangsereignis und in beiden Prozessen wird der entsprechende Übergang (Pfeilsymbol) ausgeführt (siehe letzte Zeile in Abbildung 16).

```
Mitarbeiter   = Urlaubsantrag → (abgelehnt → SKIP | akzeptiert → SKIP)
Vorgesetzter  = Urlaubsantrag → (abgelehnt → SKIP | akzeptiert → Genehmigung → SKIP)
Verwaltung    = Genehmigung → SKIP
Urlaub        = Mitarbeiter || Vorgesetzter || Verwaltung
```

Abbildung 16: Urlaubsantragsprozess in CSP beschrieben

In einer Detaillierungsstufe von CSP ist es möglich, Ereignisse in Sende- und Empfangsoperationen, die über Ports laufen und Daten übertragen können, aufzulösen. Damit gibt es in CSP die Prädikate „Senden“ und „Empfangen“, sowie Objekte (Nachrichten), die auf diese einfachen Prädikate ausgeführt werden können.

In CSP ist analog zu CCS das Subjekt der essentielle Teil. Prädikat und Objekt spielen eine sehr untergeordnete Rolle. Ohne natürlichsprachliche Ergänzungen im Bereich Prädikat und Objekte kann mit CSP kein vollständiges Modell des Urlaubsantragsprozesses erstellt werden. Natürlich sinnvolle Namen auch bei den Prozessen für das verstehen essentiell, sie sind aber kein Beitrag für die Semantik.

7 Subjekt, Prädikat und Objekt, oder vollständige Sätze

Im Folgenden werden zwei Methoden vorgestellt, die der Standardsatzsemantik Subjekt-Prädikat-Objekt nach Kenntnissen des Autors am nächsten kommen.

7.1 Anwendungsfalldiagramme

Anwendungsfalldiagramme (Use Cases) sind einer von 13 Diagrammtypen aus der UML. Diese unterteilen sich in 6 Strukturdiagrammtypen und 7 Verhaltensdiagrammtypen. Mit den Verhaltensdiagrammen werden die dynamischen Aspekte eines Programms beschrieben. Diese 7 Strukturdiagrammtypen überlappen sich in ihren Darstellungsaspekten sehr stark, wobei eine systematische Überführung ineinander nicht möglich ist. Alle dieser 7Diagrammtypen beinhalten Subjektaspekte allerdings in unklarer Form. In UML sind alles Objekte. Im Folgenden wer-

den die zwei Diagrammtypen genauer erläutert in denen der Subjektaspekt am klarsten zu Tage tritt. Dies sind Anwendungsfalldiagramme und Aktivitätsdiagramm.

Anwendungsfalldiagramme erlauben es, die Benutzung eines Systems aus der Sicht der Nutzer zu beschreiben. Ein Use Case zeigt auf, welcher Anwender (Akteur, =Subjekt) welche Handlungen (Prädikat) auf dem System ausführt. Ein Anwendungsfall beschreibt das nach außen sichtbare Verhalten des betrachteten Elements (System, Klasse, etc.) und kapselt eine in sich geschlossene Sammlung von Aktionen, die in einer festgelegten Reihenfolge ablaufen. Ein Anwendungsfall zeigt nicht, welche Klassen und welche einzelnen Operationen an den Aktionen beteiligt sind. Eine vollständige Anwendungsfallbeschreibung ist dann vollständig, wenn die dahinterliegenden Abläufe definiert sind. Dafür kann eine entsprechende Methode der Verhaltensmodellierung aus UML oder eine natürlich sprachliche Beschreibung verwendet werden.

Akteure werden in UML als spezielle Klassen mit bestimmten Eigenschaften gesehen, so dass in UML Akteure nicht definitiv als aktiv gelten, d.h. es ist nur festgelegt, welche Aktionen zwischen dem Akteur und dem System ablaufen, aber es ist nicht definiert, wer der Ausgangspunkt der Aktionen ist. Allerdings bietet es sich an, Akteure als Ausgangspunkt von Aktionen zu betrachten.

Die Abbildung 17 zeigt das Anwendungsfalldiagramm für den Urlaubsantragsprozess. Die Aktionsfolge für „Antrag ausfüllen“ kann heißen: Urlaubsbeginn eintragen, Urlaubsende eintragen, Vorgesetzten zur Prüfung auffordern. Analog können die anderen Anwendungsfälle beschrieben werden. Für komplexe Reihenfolgen gibt es in UML das Aktivitätsdiagramm, in dem Elemente der Datenflussdiagramme, Petrinetze, Flussdiagramme etc. kombiniert sind. Allerdings ist die Zusammenarbeit von mehreren Aktivitätsdiagrammen durch den Austausch von Signalen und Ereignissen nur sehr rudimentär möglich. Dies bedeutet, dass der Zusammenhang der einzelnen Anwendungsfälle in unserem Beispiel auf Use Case Diagrammebene gar nicht und in den Aktivitätsdiagrammen nur sehr eingeschränkt möglich ist. Ein Beispiel ist das alternative Warten auf die Genehmigung oder Ablehnung.

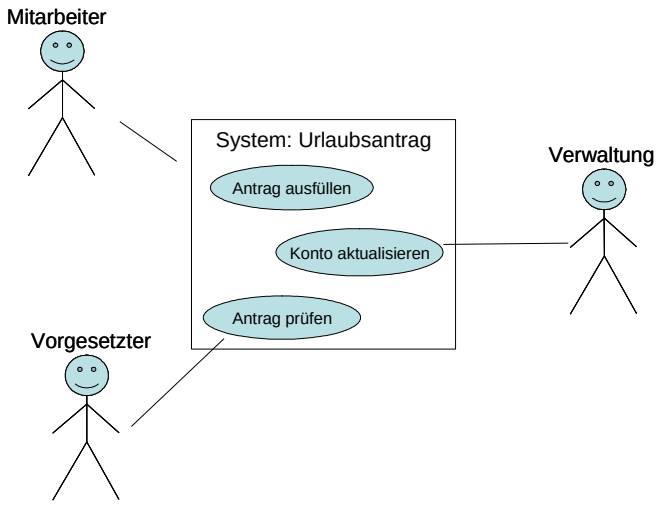

Abbildung 17: Anwendungsfalldiagramm für den Urlaubsantragsprozess

Trotz der oben beschriebenen Unzulänglichkeiten bietet UML mit den Anwendungsfalldiagrammen und den anderen Diagrammtypen die eingeschränkte Möglichkeit, vollständige Sätze im Sinne der Standardsatzgrammatik zu bilden. In UML gelten Akteure nicht als Bestandteil des Modells, so dass deren Verhalten im Detail nicht weiter betrachtet, insbesondere eine mögliche Kommunikation von Akteuren untereinander. Dies wird auch dadurch deutlich dass die Akteure in den übrigen Diagrammtypen von UML mit Ausnahme beim Time Sequence Diagramm nicht mehr auftauchen.

Da die Handelnden in einem Geschäftsprozess wesentlich sind, wird hier ein wesentlicher Teil der Wirklichkeit im Modell nur sehr unzureichend berücksichtigt. Das Verhalten der Handelnden in einem Prozess ist wesentlich für das Ablaufen eines organisationsübergreifenden Geschäftsprozesses.

7.2 Aktivitätsdiagramme in UML

Anwendungsfalldiagramme in UML werden häufig mit Hilfe von Aktivitätsdiagrammen weiter verfeinert.

Das folgende Beispiel zeigt ein Aktivitätsdiagramm für den Urlaubsantrag. Die einzelnen Aktivitäten werden dabei mit Hilfe von sogenannten Schwimmbahnen gruppiert je nachdem wer die Tätigkeit ausführt. In unserem Beispiel gibt e seine Schwimmbahn für den Mitarbeiter, den Vorgesetzten und die Personalabteilung. Diese Schwimmbahnen können als Subjekte betrachtet werden, die die zugeordneten Aktivitäten ausführen. Die Reihenfolge der Aktivitäten wird durch den Kontrollfluss analog den Flussdiagrammen definiert. Es besteht die Möglichkeit durch fork und join Operationen einen einzelnen Kontrollfluss in mehrere parallele Kontrollflüsse aufzuspalten (fork) bzw. Wieder zusammen zu führen (join). Im Urlaubsbeispiel spaltet sich nach der Genehmigung des Urlaubsantrags durch den Vorgesetzten der Kontrollfluss auf (im Bild durch einen schwarzer Balken dargestellt). Dies bedeutet dass der Mitarbeiter und die Personalabteilung die Urlaubsgenehmigung parallel erhalten. Die parallelen Kontrollflüsse werden danach wieder zusammengeführt bevor der Endeknoten erreicht wird.

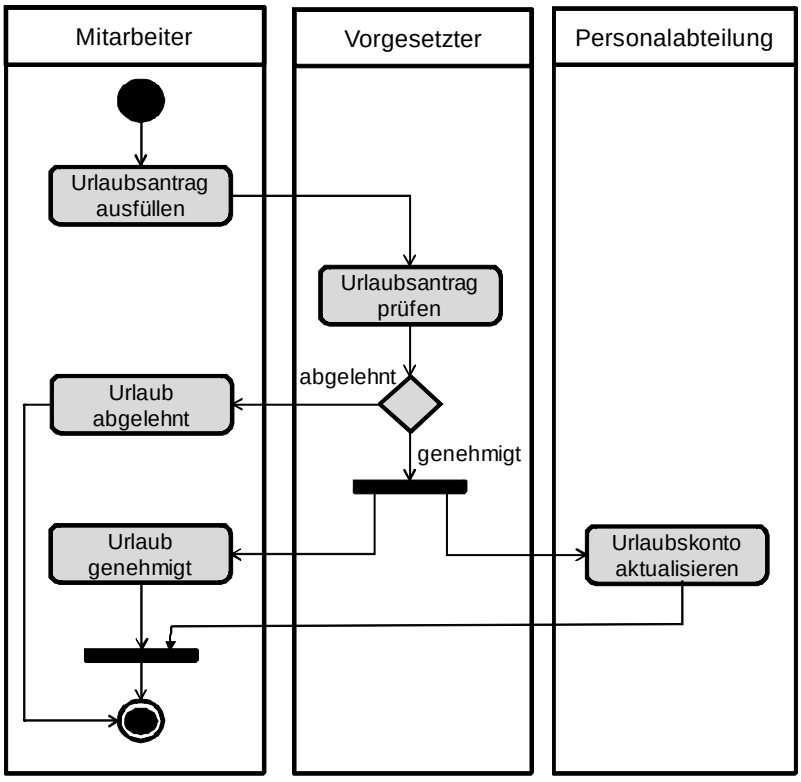

Abbildung 18: Aktivitätendiagramm des Urlaubsantragsprozesses

Die Koordinierung der einzelnen Aktivitäten erfolgt durch den Wechsel des Kontrollflusses zwischen den einzelnen Schwimmbahnen. Wobei es etwas wirklichkeitsfremd erscheint wen man sagt nach dem ausfühlen des Urlaubsantrags durch den Mitarbeiter wechselt der Kontrollfluss zum Vorgesetzten. Dieser Übergang eines Kontrollflusses von einem Prozessteilnehmer auf einen anderen erscheint nicht sehr natürlich. Normalerweiser tauschen Prozessteilnehmer Nachrichten aus. Die Definition von Prozessen auf der Basis des Nachrichtenaustauschs zwischen Prozessteilnehmern wird im folgenden Abschnitt behandelt.

7.3 PASS (Parallel Activity Specification Schema)

Die Prozessbeschreibungsmethode PASS lehnt sich an die theoretischen Konzepte von Milner und Hoare an. Deren Prozessalgebra wurde durch eine entsprechende Symbolik für die Praxis zugänglich gemacht, die Kommunikation und Synchronisation wurde ausgebaut und die objektorientierten Konzepte (siehe Abschnitt 5.2) integriert.

Die Prozesse der Prozessalgebra werden in PASS Subjekte genannt. Die Subjekte tauschen Nachrichten aus und benutzen Operationen auf Objekten. Die Möglichkeiten von PASS werden am Beispiel des Urlaubsantragsprozesses dargestellt. Für weitergehende Informationen wird auf (Fleischmann, Distributed Systems-Software Design and Implementation, 1994) und (Fleischmann, Subjektorientiertes Geschäftsprozess-management, White Paper, 2007) verwiesen.

Die subjektorientierte Beschreibung eines Prozesses beginnt mit der Identifikation der im Prozess beteiligten prozessspezifischen Rollen, den Subjekten, und die zwischen diesen ausgetauschten Nachrichten. Die Handelnden sind die wesentlichen Bestandteile eines Prozesses, durch sie wird ein Prozess getrieben. Damit das gemeinsame Handeln abgestimmt und strukturiert wird, kommunizieren sie untereinander und benutzen geeignete Werkzeuge.

Die Abbildung 19 zeigt die in einem Prozess beteiligten Subjekte, die zwischen ihnen ausgetauschten Nachrichten und welche Informationen in den Nachrichtenparametern übertragen werden. Abhängig von den Tätigkeiten der Subjekte werden Schritt für Schritt die benötigten Prädikate mit zugehörigen Daten identifiziert.

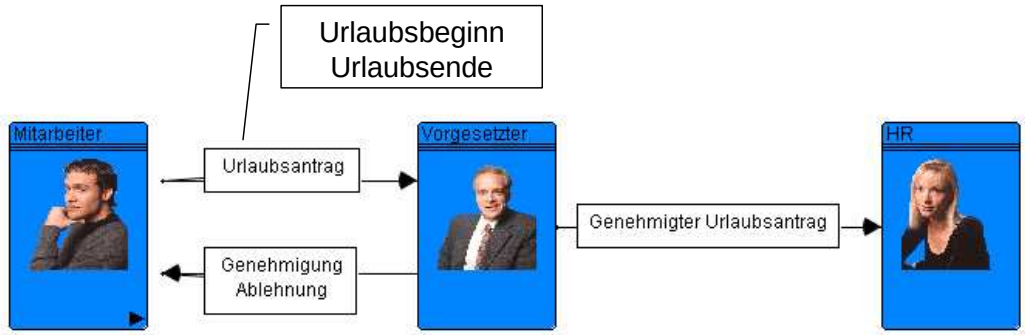

Abbildung 19: Am Urlaubsantragsprozess beteiligte Subjekte und deren Interaktionen

Beim Senden von Nachrichten werden die vom Empfänger benötigten Daten mit übermittelt. So wird mit der Nachricht *Urlaubsantrag,* die der Mitarbeiter an den Vorgesetzten sendet, der Urlaubsbeginn und das Urlaubsende übertragen.

Hinweis:

Der hier verwendete Begriff „Nachrichten senden“ bzw. „Nachrichten übertragen“ soll noch keine Realisierung der Interaktionen implizieren. Der Urlaubantrag kann durchaus in Papierform vorliegen und per Hauspost gesendet werden bzw. es kann eine Webmaske ausgefüllt werden, der Inhalt wird abgespeichert und dem Vorgesetzten zugänglich gemacht. Der hier verwendete Begriff des Nachrichtenaustauschs bezieht sich auf eine rein logische Handlung und soll keine Realisierung durch Nachrichtensysteme nahe legen.

In Abbildung 19 wird nur die Interaktionsstruktur eines Prozesses wiedergegeben. In einer weiteren Verfeinerungsstufe muss nun beschrieben werden in welcher Reihenfolge die einzelnen Interaktionen ausgeführt werden, d.h. das Verhalten der einzelnen Subjekte muss definiert werden.

Zunächst wird das Verhalten des Mitarbeiters näher betrachtet. Die folgende Abbildung 20 zeigt die Reihenfolge, in der der Mitarbeiter Nachrichten sendet, empfängt und interne Aktionen ausführt.

Der Anfangszustand ist grün umrandet. In diesem Zustand füllt der Mitarbeiter seinen Urlaubsantrag aus. Ist dieser fertig ausgefüllt, kommt der Mitarbeiter über den Zustandsübergang *Urlaubsantrag ausgefüllt* in den nächsten Zustand. Dieser Folgezustand ist ein Sendezustand. In diesem Zustand wird der Urlaubsantrag an den Vorgesetzten gesendet. Nach dem erfolgreichen Senden wird in den Zustand *Antwort vom Vorgesetzten* übergegangen. Dieser Zustand ist ein Empfangszustand. In diesem Zustand wartet der Mitarbeiter auf die Antwort des Vorgesetzten. Bekommt er eine Nachricht *Ablehnung vom Vorgesetzten* ist der Prozess beendet.

Erhält der Mitarbeiter die Nachricht *Genehmigung vom Vorgesetzten* tritt er an dem vereinbarten Datum den Urlaub an, Nach Antritt des Urlaubs ist der Urlaubsantragsprozess ebenfalls abgeschlossen.

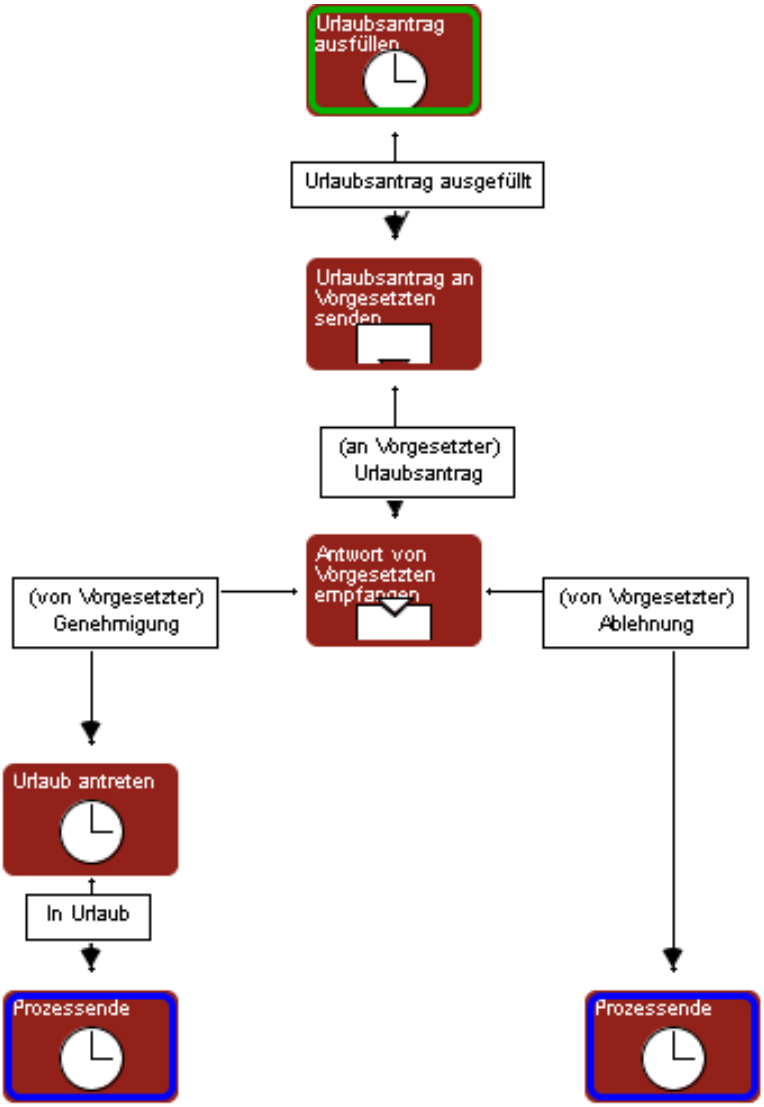

Abbildung 20: Verhalten des Mitarbeiters

Das Verhalten des Vorgesetzten ist gegenüber dem Mitarbeiter komplementär. Nachrichten, die der Mitarbeiter sendet, empfängt der Vorgesetzte und umgekehrt. Abbildung 21 zeigt das entsprechende Verhalten des Vorgesetzten. Der Vorgesetzte erwartet zunächst den Urlaubsantrag des Mitarbeiters. Erhält er den Urlaubsantrag, geht er in den Zustand über, in dem der Urlaubsantrag geprüft wird. Diese Prüfung kann zum Ergebnis *Genehmigung* oder *Ablehnung* führen. Je nach Ergebnis wird in den entsprechenden Folgezustand übergegangen, in dem das jeweilige Prüfungsergebnis dem Mitarbeiter mitgeteilt wird. Wird der Urlaub genehmigt, wird noch die Verwaltung (Personalabteilung) informiert.

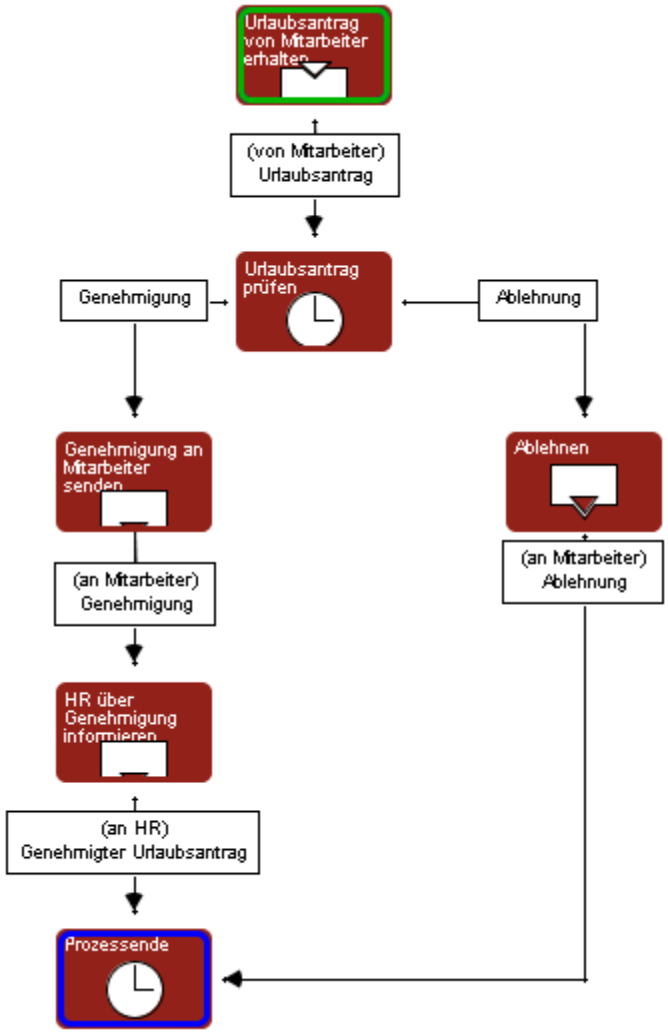

Abbildung 21: Verhalten des Vorgesetzten

Das folgende Bild zeigt das Verhalten der Verwaltung. Diese bekommt den genehmigten Urlaubsantrag und legt ihn ab. Danach ist für die Verwaltung der Prozess beendet.

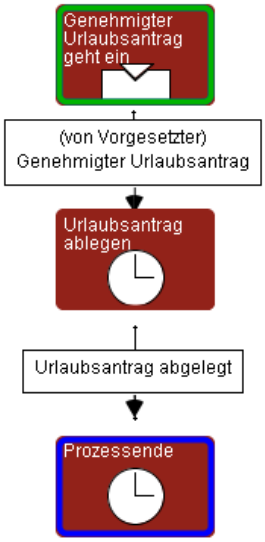

Abbildung 22: Verhalten der Verwaltung

Bis hierher wurden

- die an einem Prozess beteiligten Subjekte
- die Interaktionen die sie austauschen
- die Daten die sie bei jeder Interaktion senden bzw. empfangen
- und das Verhalten der einzelnen Subjekte

beschrieben.

Die Beschreibung eines Subjekts definiert, in welcher Reihenfolge Nachrichten gesendet und empfangen, bzw. interne Funktionen ausgeführt werden, d.h. in einem Subjekt wird festgelegt, in welcher Reihenfolge ein Subjekt welche Prädikate anstößt. Diese Prädikate können die Standardprädikate Senden oder Empfangen sein oder irgendwelche Prädikate, die auf entsprechenden Objekten definiert sind. Der Mitarbeiter füllt den Urlaubsantrag aus (siehe Startzustand in Abbildung 20).

Den einzelnen Knoten und Übergängen in einer Subjektbeschreibung muss also noch eine Operation zugeordnet werden, wobei es nicht von Bedeutung ist, wie die Operation definiert wird. Dies kann durch ein Objekt, wie in Abschnitt 5.2 beschrieben erfolgen oder natürlichsprachlich. Deshalb wird im Folgenden nicht der Begriff Methode bzw. Operation verwendet, sondern der allgemeine Begriff Service.

- Interne Funktion
  Einem internen Funktionsknoten wird ein Service zugeordnet. Wird dieser Zustand erreicht, wird der zugeordnete Service angestoßen. Der Service wird ausgeführt und die Endebedingungen des ausgeführten Service entsprechen den Kanten, die den jeweiligen internen Funktionsknoten verlassen.
- Sendeknoten
  Dem Ausgang eines Sendeknoten ist über den Nachrichtennamen ein Service zugeordnet, mit dem die Werte der gesendeten Daten ermittelt werden. Vor dem Übertragungsvorgang wird dieser Service angestoßen. Der Service ermittelt die Werte der Nachrichtenparameter, die mit der Nachricht übertragen werden.
- Empfangsknoten

Jedem Ausgang eines Empfangsknoten wird über den Nachrichtennamen ein Service zugeordnet. Wird eine in diesem Zustand vorgesehene Nachricht angenommen, wird der zugeordnete Service angestoßen. Bei diesem Anstoß werden die erhaltenen Parameterwerte an den Service übergeben. Der zugeordnete Service verarbeitet die erhaltenen Nachrichtenparameterwerte.

Diese Services werden also benutzt, um den einzelnen Schritten in einem Subjekt eine klare fachliche Bedeutung zu zuweisen. Mit Hilfe der Services werden die in einem Subjekt verwendeten Prädikate definiert. Diese „Benutztservices“ werden synchron angestoßen, d.h. ein Subjekt geht erst dann zum entsprechenden Folgezustand, wenn der benutzte Service auch vollständig abgearbeitet ist.

Das folgende Bild zeigt, wie die in einem Subjekt verwendeten Prädikate mit Hilfe eines Objekts definiert werden.

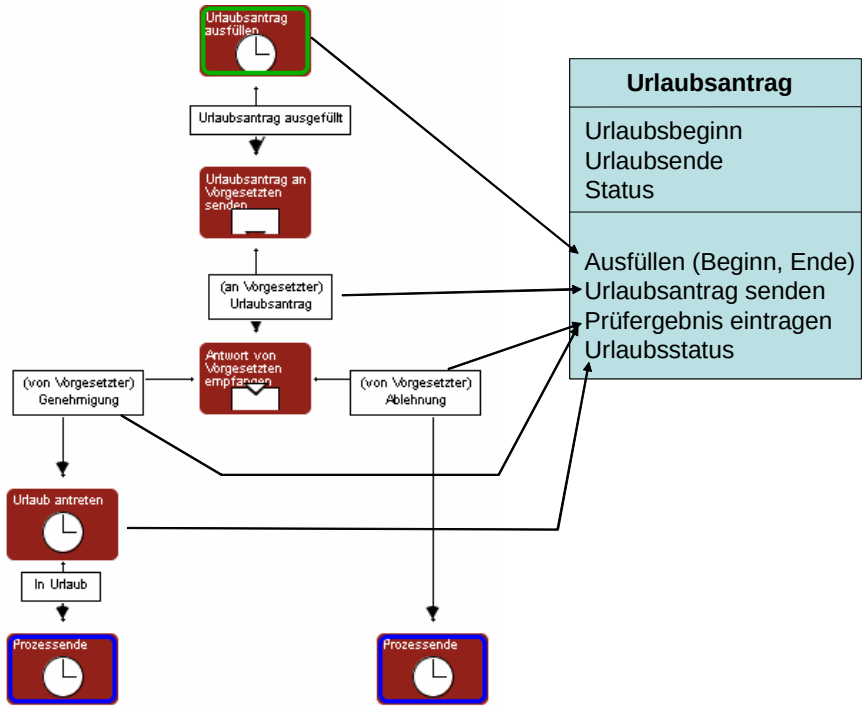

Abbildung 23: Ein Subjekt mit Prädikaten und Objekt

8 Umsetzung von Modellen in ablauffähige Programme

Bisher wurde die Analysephase betrachtet. Diese ging von dem zu betrachteten Teil der Wirklichkeit aus und bildete diesen in ein Modell ab, in dem die wesentlichen Aspekte nicht verloren gehen durften. Legt man die Standardsatzsemantik zugrunde, so zeigt sich, dass in allen Modellierungssprachen, mit Ausnahme von PASS, nicht alle Bestandteile eines Satzes betrachtet werden.

Bei der Anwendungsentwicklung muss nun als nächstes das Modell in ein Anwendungsprogramm übergeführt werden.

Während in der Modellierung immer mindestens ein Teil der Standardsatzsemantik sehr schwach ausgeprägt ist, findet man in nahezu allen Programmiersprachen sowohl Subjekte, als auch Prädikate und Objekte. Dies ist auch der Grund, warum es mit unvollständigen Modellierungssprachen gar nicht gelingen kann, ausführbaren Code aus dem Modell abzuleiten. Um ausführbaren Code ohne weitere Zwischenschritte aus dem Modell erzeugen zu können, braucht es eine vollständige Modellierungssprache im Sinne von Subjekt, Prädikat und Objekt wie bei PASS(siehe /FLEI07/).

Der Code eines Programms wird durch einen Betriebssystemprozess (oder durch einen Thread, einer vereinfachten Form eines Betriebssystemprozesses) ausgeführt. Zur Ausführung ist das Speicherabbild des Programms, der Speicher für die Daten und weitere vom Betriebssystem bereitgestellte Ressourcen wie z.B. Ein-/Ausgabegeräte oder der Prozessor notwendig. All diese Betriebsmittel werden als zum Betriebssystemprozess zugehörig betrachtet. Der Betriebssystemprozess ist das Subjekt, das Programm enthält die auszuführenden Aktionen also die Prädikate und die verwendeten Daten und Ressourcen sind die Objekte auf die die Prädikate wirken.

In einem Rechner können mehrere Betriebssystemprozesse konkurrierend ablaufen. Konkurrierend ablaufen heißt, dass sie um die verschiedenen Betriebsmittel konkurrieren. Ein wesentliches Betriebsmittel ist natürlich der Prozessor. Da es in den meisten Rechnern mehr Betriebsystemprozesse als Prozessoren gibt, wird der Prozessor immer nur für eine bestimmte Zeit einem Betriebssystemprozess zugeordnet. Dieser gibt den Prozessor entweder an einer Synchronisationsstelle wieder frei oder der Prozessor wird ihm durch das Betriebssystem nach Ablauf einer Zeitscheibe entzogen. Diese Art der überlappten Ausführung wird auch als concurrent oder parallel processing bezeichnet.

Ein Betriebssystemprozess ist damit der Ausgangspunkt der durchgeführten Aktionen und hat auch Zugriff auf die verwendeten Daten. Die einzelnen Programmanweisungen entsprechen im Wesentlichen den Prädikaten. Wobei natürlich eine Programmanweisung in der Regel auch einen Bezug zu Daten hat und damit auch einen Bezug zu Objekten.

Damit können vollständige Sätze gebildet werden, da es ein Standardsubjekt, den Betriebsystemprozess, gibt.

Existieren mehrere Betriebssystemprozesse, die in einem Rechner parallel ablaufen und zusammenarbeiten, hat man mehrere Subjekte die ihre Abläufe synchronisieren müssen. Dazu werden, wie oben in CCS und CSP gezeigt, Synchronisationsmechanismen verwendet. Das sind entweder Ereignisse, mit denen sich Prozesse direkt synchronisieren, oder Mechanismen wie z.B. Semaphore und Monitore, mit denen der Zugriff auf die Betriebsmittel geregelt wird. Eine ausführliche Beschreibung und Gegenüberstellung von Synchronisationskonzepten findet sich in (Fleischmann, Distributed Systems-Software Design and Implementation, 1994) und (Fleischmann, Entwicklung verteilter Realzeitprogramme, in PEARL 93, 1993)

Parallelität, d.h. Prozesse und Threads sind heute eine Eigenschaft des Betriebssystems und nur in wenigen Programmiersprachen enthalten. Dies sind in der Regel Programmiersprachen, die im Bereich der Realzeitverarbeitung wie z.B. PEARL und Ada eingesetzt werden. In Java ist Parallelität eine Eigenschaft des Main Objekts. Durch die Verwendung entsprechender Operationen der Klasse THREADS können in Java parallele Programme entwickelt werden. In Java werden also über entsprechende Operationen auch Eigenschaften des Betriebssystems oder der virtuellen Javamaschine benutzt. Es gibt keine direkten Sprachkonstrukte für Parallelität wie in Ada oder PEARL.

9 SPO und SOA

Modelle, die in der Standardsatzsemantik Subjekt-Prädikat-Objekt (SPO) modelliert wurden, müssen im nächsten Schritt umgesetzt, d.h. als Softwareanwendung von der IT implementiert werden. Diese Umsetzung muss schnell und kostengünstig durchgeführt werden. Können bei der Umsetzung eines Modells Prädikate auf bereits existierende Funktionen abgebildet werden, so kann die Abbildung des Modells in eine Anwendung wesentlich effizienter und effektiver erfolgen. Wenn nun eine Menge von Bausteinen existiert, die Prädikate und Objekte aus den Modellen realisieren, können diese von den Anwendern, den Subjekten, entsprechend dem Modell ohne jegliche Programmierarbeit zusammengebaut werden. Ein Konzept das dies unterstützt, ist die in den letzten Jahren stark diskutierte Service orientierte Architektur (SOA).

In diesem Kapitel soll gezeigt werden, wie sich Modellbeschreibungen, die sich an der Standardsatzsemantik orientieren, für die Implementierung auf eine Service orientierte Architektur abbilden lassen. Damit wird ein direkter Bezug von der Standardsatzsemantik Subjekt-Prädikat-Objekt, wie sie täglich von Menschen benutzt wird, zur IT-Architektur hergestellt.

Das grundlegende Konzept hinter SOA ist, Softwareanwendungen in logische Bausteine aufzubrechen. Diese Bausteine lassen sich dann nach Belieben und sehr schnell zu neuen konkreten Anwendungen gemäß den Anforderungen des Geschäftsprozesses zusammenbauen. Anstatt für einen neuen oder abgewandelten Geschäftsprozess umfassende Änderungen am zugehörigen Anwendungsprogramm vorzunehmen, soll durch die Aneinanderreihung von Serviceaufrufen eine möglichst große Wiederverwendbarkeit erreicht werden.

SOA ist ein Konzept für die Softwaregestaltung, das Geschäftsanwendungen in einzelne Funktionen oder „Services“ zerlegt, z. B. die Prüfung des Urlaubsantrags oder die Aktualisierung des Urlaubskontos. Diese Funktionen lassen sich unabhängig von den Anwendungen und IT-Plattformen, auf denen sie ausgeführt werden, nutzen.

Wenn einzelne Funktionen innerhalb von Anwendungen durchgängig als eigenständige Bausteine verfügbar sind, haben Unternehmen die Möglichkeit, diese auf unterschiedliche Weise zu gruppieren, um völlig neue Funktionalitäten zu erstellen. Die gesamte Programmlogik ist somit nicht zwingend in einem einzigen Programm zu finden, sondern kann über mehrere Systeme und sogar Unternehmen verteilt sein.

Um die verfügbaren Funktionen entsprechend den Anforderungen zu gruppieren, gehören zu einer vollständigen SOA Kontrollinstanzen, die definieren, für welche Aufgabe welcher Dienst verwendet wird und die die Prozesssteuerung übernehmen, damit die Dienste in der richtigen Abfolge und unter den richtigen Bedingungen aufgerufen werden. Diese Kontrollinstanzen sind also der Ausgangspunkt der Handlungen und entsprechen somit den Subjekten.

Für die Definition der Prozesssteuerung gibt es zur Zeit zwei Verfahren, Service Orchestrierung und Service Choreographie.

9.1 Service Orchestrierung und Service Choreographie

Für das Zusammensetzen von Geschäftsprozessen werden zwei Vorgehensweisen unterschieden, die Orchestrierung und die Choreographie.

Orchestrierung:

Die Orchestrierung beschreibt die Reihenfolge, in der die benötigten Operationen ablaufen. Orchestrierung von Operationen oder Services heißt, dass mit einer Art Flussdiagramm die möglichen Ausführungsreihenfolgen der verwendeten Operationen beschrieben werden. Bei der Orchestrierung wird die Reihenfolge der auszuführenden Operationen durch einen zentralen Kontrollfluß festgelegt. Die folgende Abbildung zeigt wie die der zentrale Kontrollfluß die Abarbeitung der einzelnen Serviceoperationen definiert.

Als formale Beschreibungssprache hat sich BPEL (Business Process Execution Language) durchgesetzt (Einen guten Überblick zu BPEL findet man z.B. in Havey, 2005). Die Ausführung einer Orchestrierung erfordert bei der Implementierung eine zentrale Kontrollinstanz, die den Kontrollfluss gemäß der Definition steuert. Bisher ist die Einbindung von Benutzerinteraktionen in die Orchestrierung sehr aufwändig. BPEL ist gut geeignet bei der Orchestrierung von vollständig automatisch durchlaufenden Prozessen. Allerdings hat in der Regel ein Prozess mindestens ein oder zwei Benutzer. Derjenige, der den Prozess anstößt und derjenige, der das Ergebnis will.

Choreographie:

Die Choreographie lehnt sich an die Konzepte von Hoare und Milner an (siehe obige Abschnitte). Jeder Ablauf wird ähnlich der Orchestrierung definiert, allerdings synchronisieren sich die einzelnen Abläufe durch den Austausch von Nachrichten.

Unter der Choreographie von Services versteht man die Beschreibung der direkten Interaktion der an einem Prozess beteiligten Parteien zum Beispiel als Abfolge von Nachrichten. Jede an einem Prozess beteiligte Partei hat seinen eigenen Kontrollfluß. Die jeweiligen Kontrolleflüsse synchronisieren sich durch den Nachrichtenaustausch. Jeder beteiligte Anwender ist für die Ausführung der einzelnen Schritte selbst verantwortlich

Eine sehr anschauliche Definition des Unterschieds zwischen den Begriffen Choreography und Orchestration findet sich im Blog von Stefan Tilkov (übersetzt aus dem Englischen):

Bei der Orchestrierung, gibt es jemanden - den Dirigenten - der den Orchestermitgliedern sagt, was sie zu tun haben und sicherstellt, dass der Takt eingehalten wird. Bei der Choreographie folgen die Tänzer einem definierten Plan - aber jeder unabhängig voneinander.

Die folgende Tabelle fast die Unterschiede zwischen Orchestrierung und Choreographie zusammen.

Orchestrierung	Choreographie
▪ Zentrale Stelle für Prozessausführung verantwortlich	▪ Jeder Beteiligte ist für die korrekte Ausführungsreihenfolge selbst verantwortlich
▪ Zentrale Prozessausführung lässt sich bei organisationsüber-greifenden Prozessen nahezu nicht realisieren (Probleme beim Outsourcing)	▪ Dezentrale Prozessausführung lässt sich bei organisationsüber-greifenden Prozessen einfach realisieren
▪ Impliziert starke Sequentialisierung der Aktivitäten eines Prozesses.	▪ Impliziert starke Parallelisierung der Aktivitäten der am Prozess Beteiligten.

9.2 Service Orientierte Architektur (SOA)

Die Konzepte von Serviceoperationen deren Orchestrierung bzw. Choreographie und die Einbindung von Anwendern lässt sich in einem Drei Ebenen Modell darstellen.

. Das folgende Bild zeigt die wesentlichen Ebenen einer serviceorientierten Architektur.

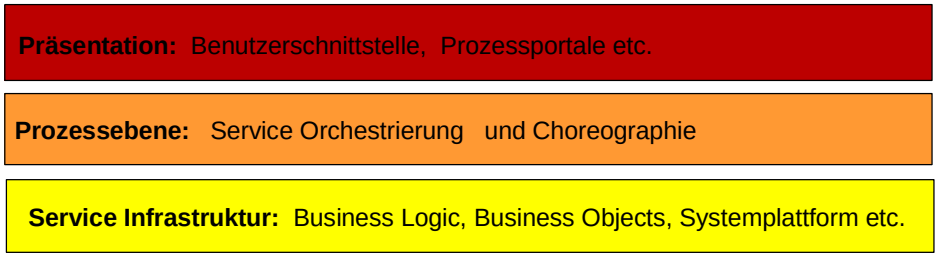

Abbildung 24: Basisebenen von SOA

Die oberste Ebene ist die Präsentationsebene, über die Anwender auf die einzelnen Funktionen eines Prozesses zugreifen können bzw. informiert werden, dass entsprechende Eingaben erwartet werden.

Die mittlere Ebene enthält die Implementation der einzelnen Geschäftsprozesse. In dieser Ebene wird definiert in welcher Reihenfolge Funktionen aus der darunterliegenden Ebene ausgeführt werden bzw. wann über die Ebene 1 vom Anwender Eingaben erwartet werden. Die Ebene 2 enthält also im Wesentlichen die Geschäftsprozesslogik.

Die unterste Ebene stellt die in der zweiten Ebene verwendeten Serviceoperationen zur Verfügung.

9.3 Anwendungsfalldiagramme und SOA

Abbildung 25 zeigt den Zusammenhang zwischen SOA und den SPO bei Anwendungsfalldiagrammen. Die Subjekte bzw. Akteure entsprechen der Präsentationsebene in der SOA Architektur. Die Prädikate den Operationen, die durch die Orchestrierung zu komplexeren Operationen zusammengebaut werden. In der Abbildung 25 entspricht die Prozessbeschreibung als EPK der Orchestrierung der einzelnen Services einschließlich der Benutzerinteraktionen. Die Serviceinfrastruktur enthält die Objekte, deren Operationen als Prädikate benutzt werden.

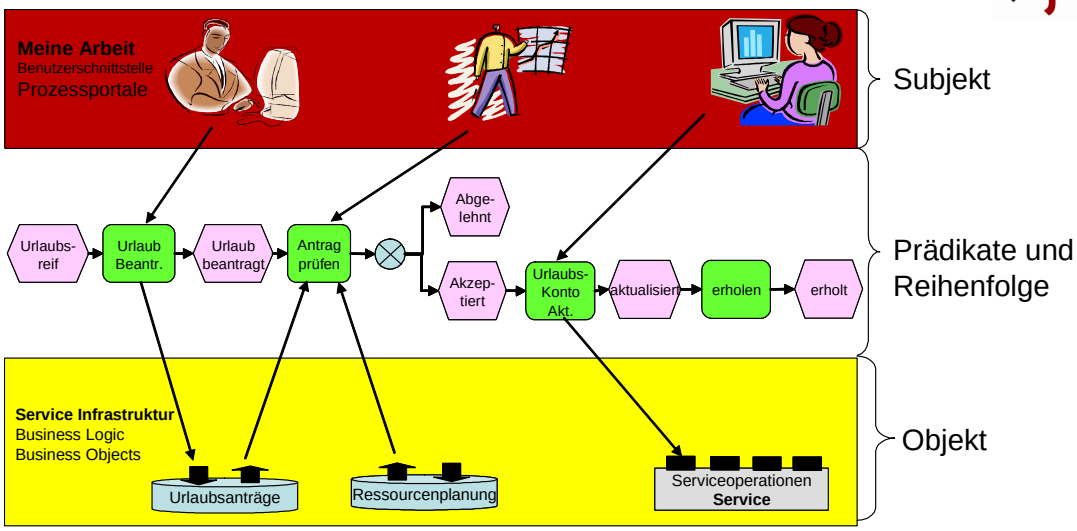

Abbildung 25: Orchestrierung und SOA

9.4 PASS und SOA

Wie in Abschnitt 0 beschrieben, werden in PASS Operationen benutzt, um den einzelnen Schritten in einem Subjekt eine klare fachliche Bedeutung zu zuweisen. Diese „Benutztservices“ werden synchron angestoßen, d.h. ein Subjekt geht erst dann zum entsprechenden Folgezustand, wenn der benutzte Service auch vollständig abgearbeitet ist. Dies entspricht dem Vorgehen wie in der Orchestrierung. Allerdings gibt es in PASS noch eine direkte Kommunikation zwischen den Subjekten analog den Prozessalgebren von Milner und Hoare. In jPASS! schließen die Subjekte die Benutzer mit ein. Es gibt also reine Subjekte ohne menschlichen Anteil und Subjekte, die die entsprechenden Benutzer mit einschließen. Diese Benutztservices können entweder komplexe Geschäftslogiken (Business Logic) oder einfache lesende oder schreibende Zugriffe auf Daten sein.

Die den einzelnen Knoten zugeordneten Services können als Teil ihrer Abarbeitung auch Benutzerein- und -ausgaben enthalten. Die Benutzer werden somit ein Teil der Services für einen Prozess. So kann ein bestimmter Service notwendige Daten bereitstellen, die mit der entsprechenden Nachricht an ein anderes Subjekt gesendet werden sollen. Dies kann dadurch realisiert werden, dass der Benutzer über eine entsprechende Maske die Daten eingibt. In unserem Beispiel wäre in der internen Funktion *Urlaubsantrag ausfüllen* ein solcher Service hinterlegt. Der Service erfüllt also die fachliche Anforderung, die Daten für den Urlaubsantrag bereitzustellen. Benutzereingaben können auch mit errechneten oder aus Datenbeständen extrahierten Informationen kombiniert werden.

Um solche benutzerunterstützten Services standardisiert zugänglich zu machen, werden heute überwiegend Portale benutzt. Dient ein Portal auch dem Zugang zu den jeweiligen Prozessen bzw. Prozessinstanzen, wird es als *Prozessportal* bezeichnet.

Subjekte als Ganzes können auch als Services gesehen werden. Ein Subjekt kann ein anderes Subjekt beauftragen einen bestimmten Service zu erbringen. Subjekte können als Dienstleister gesehen werden, die aktiv und eigenständig die geforderten Dienste erbringen. Subjekte bieten dann so genannte „Beauftragtservices“ an. Diese werden durch eine Nachricht beauftragt. Der Beauftragende kann mit seinen Aufgaben weitermachen während der Dienstleister die Beauftragung abarbeitet. Das Ergebnis sendet dieser per Nachricht an den Beauftragenden zurück. Das beauftragende Subjekt kann die Nachricht mit dem Ergebnis gemäß seinem Ablauf zu einem geeigneten Zeitpunkt entgegennehmen.

Die folgende Abbildung 26, fasst die einzelnen Aspekte von subjektorientierter Prozessbeschreibung, serviceorientierter Architektur (SOA) und Portalen nochmals zusammen.

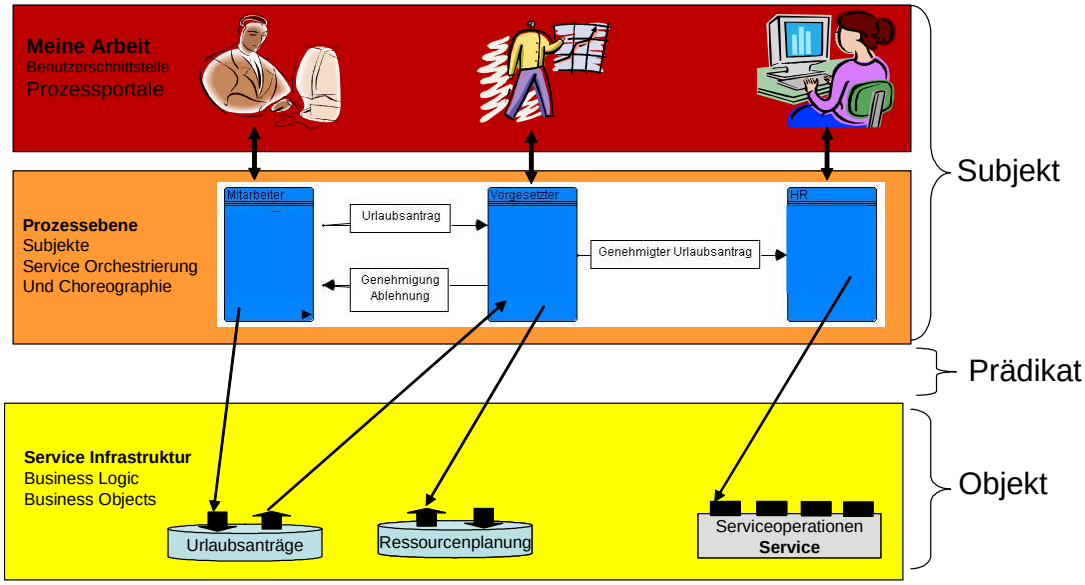

Abbildung 26: jPASS und SOA

In Unternehmen bieten Portale eine benutzerspezifische Sicht auf die Informationen und Geschäftsprozesse eines Unternehmens. Durch eine subjektorientierte Betrachtung von Prozessen wird gleichzeitig definiert, was ein Benutzer von welchen Prozessen sehen soll. Durch die Zuordnung der Subjekte zu Aufgabenträgern (Benutzern) wird gleichzeitig festgelegt, welche Prozesse bzw. Services im Portal zur Verfügung stehen müssen, damit ein Benutzer seine Arbeit erledigen kann.

10 Schlussbemerkung und Ausblick

Die Anwendung der Standardsatzsemantik soll die Kommunikation zwischen den Anwendern, den Analysten und den Softwareentwicklern erleichtern. Durch die Nutzung einer vollständigen Sprachstruktur zur Beschreibung formaler Modelle wie wir sie auch in unserer täglichen Sprache verwenden zeigte sich: In vielen Projekten wird die Kommunikation zwischen Fachabteilung und IT-Abteilung konfliktfreier. Die große Akzeptanz der Anwendungsfalldiagramme in der industriellen Praxis dürfte mit darauf zurückzuführen sein, dass damit vollständige und von allen gleichermaßen verstandene Sätze gebildet werden können. In den letzten Jahren wird aber die organisationsübergreifende Zusammenarbeit immer wichtiger, was erhebliche Auswirkungen auf die geforderte Softwarearchitektur hat. Möchte man Modelle bilden, die die Zusammenarbeit zwischen verschiedenen Organisationen mit definiert, müssen auch entsprechende Konstrukte bereitgestellt werden. Die Kommunikation zwischen den am Prozess Beteiligten muss tTeil des Modells werden. Ein solches Modellierungstool, das auf der Subjekt-Prädikat-Objekt Methode von PASS beruht, wurde mit jPASS! realisiert. Mit jPASS! wurden inzwischen viele Geschäftsprozessmodelle gebildet und implementiert und es zeigte sich, dass die vollständigen Sätze die Akzeptanz deutlich steigerten. Durch die vollständige Semantik ist es mit jPASS! ohne weiteres möglich, lauffähige Anwendungsprogramme direkt aus dem Modell zu generieren. Auch hat sich gezeigt, dass die Subjekte hervorragend zur Choreographie….

Im nächsten Schritt wird untersucht, wie das Wissensmanagement in einer SPO Struktur gestaltet werden kann. Hier fließen Ansätze ein, wie sie in (Stary, 2004) beschrieben sind.

Es zeigten sich natürlich auch erste Lücken. So gilt es zu untersuchen, wie das Wissensmanagement in einer SPO Struktur gestaltet werden kann. Hier könnten Ansätze, wie sie in (Stary, 2004) beschrieben sind, einfließen.

Des Weiteren müssen noch Möglichkeiten für eine verbesserte Ausnahmebehandlung gefunden werden.

Hinsichtlich der Simulation von Geschäftsprozessen um z.B. Zeit- und Ressourcenbedarfe zu ermitteln ergeben sich ebenfalls neue Fragestellungen und auch neue Lösungsansätze. Bedingt durch die obigen Überlegungen resultieren zwingend Fragestellungen nach der Gültigkeit betriebwirtschaftlicher Konzeptionen und Methoden, welche grundsätzlich auf der Prädikat-Objektrelation basieren. Denken wir lediglich an so viel gepriesene und auch durchgeführte Simulationen, die letztendlich für Maschinen richtig sind, denn hier wird das Subjekt wieder implizit angenommen.

Wird allerdings von betriebswirtschaftlichen Simulationen gesprochen, so müssen die Theorien um die explizite Definition der Subjekte erweitert werden. Nehmen wir als Beispiel Ansätze der stochastischen Simulation:

Kern jeder stochastischen Simulation ist die Erzeugung von Zufallszahlen. Die Zufallszahlen werden im Systemmodell dazu benutzt, Stichproben zufälliger Ereignisse zu simulieren, die bestimmten statistischen Häufigkeitsverteilungen unterliegen. Die Verteilungen können sowohl theoretisch als auch empirisch ermittelt worden sein. Beispiele für empirisch ermittelte Verteilungen wären die Zugangshäufigkeiten der verschiedenen Erzeugnisse in einem Fertiglager oder die Verteilung der Hauptnutzungszeiten an einer Drehbank. Durch die Zuordnung (Transformation) der einzelnen Zufallszahlen zu den einzelnen Verteilungswerten werden durch das Ziehen von Zufallszahlen zufällige Ereignisse erzeugt, die der statistischen Verteilung unterliegen.

Wir gehen davon aus, dass ein bestimmter Zustand eine bestimmte Aktion auslöst, basierend auf wohldefinierten mathematischen Modellen.

Nur, ein Subjekt (ein nicht maschinelles) kann Zustände wechseln und für jede Instanz der betrachtenden Prädikat-Objektrelation alterierende Zustände, ja sogar indefinite Zustände erzeugen. Daher führen sich Modelle, die basierend auf Kontrollflüssen oder der Objektorientierung zu Simulationen herangezogen werden letztendlich ad absurdum, da die Werte, die diese Simulation liefert, keinen Bezug zum Subjekt haben.

11 Literatur:

Für die Darstellung der einzelnen Methoden wurde sich an den jeweiligen Stichwörtern in Wikipedia orientiert. Die folgende Literatur wurde verwendet, ohne dass dies immer explizit erwähnt wurde. Natürlich wurden wörtlich Zitate entsprechend gekennzeichnet. Bei allseits bekannten Konzepten wie UML etc. wurde auf die Angabe spezieller Literatur verzichtet.

Denert, E. (1991). *Software Engineering,.* Berlin: Springer Verlag.

Fischer H., Fleischmann, A., Obermeier S. (2006). *Geschäftsprozesse realisieren*, Vieweg Verlag

Fleischmann, A. (1994). *Distributed Systems-Software Design and Implementation.* Springer Verlag.

Fleischmann, A. (1993). *Entwicklung verteilter Realzeitprogramme, in PEARL 93.* Informatik aktuell: Springer Verlag, Hrsg. P. Holleczek.

Fleischmann, A. (2007). *Subjektorientiertes Geschäftsprozessmanagement, White Paper.* siehe www.jcom1.com: jCOM1 AG.

Havey, M. (2005). *Essential Business Process Modelling.* O´Reilly .

Hoare, C. (1978). *Communicating Sequential Processes.* CACM.

Hoare, C. (1984). *Communicating Sequential Processes.* Prentice Hall.

Holl, A. (1999). *Empirische Wirtschaftsinformatik und evolutionäre Erkenntnistheorie* (Bde. Becker, Jörg et al. (ed.)). Wiesbaden: Gabler.

Meyers Lexikon Online 2007. (2007). http://lexikon.meyers.de/meyers/ Subjekt Objekt Problem.

Milner, R. (1980). *Calculus of Communicating Systems.* Springer Verlag.

Scheer, A.-W. (2001). *ARIS - Modellierungsmethoden, Metamodelle, Anwendungen* (Bde. 4. Auflage, ISBN 3-540-41601-3). Berlin : Springer.

Scholz, M., & Holl, A. (1999). *Objektorientierung und Poppers Drei-Welten-Modell als Theoriekerne der Wirtschaftsinformatik* (in: Schütte, Reinhard et al. (ed.): Ausg.). Essen: Universität Essen.

Stary, C. (2004). *Partizipatives organisationales Lernen – Ein prozessorientierter Ansatz.* DUV.

Schmidt J.E., Rabanus S., Vilmos A. (2005), *Systax*

web.uni-marburg.de/dsa//Direktor/Rabanus/SS2005/Grundlagen.pdf