
# Subjekt - Prädikat - Objekt

## Ein Ansatz zur Beschreibung von informationstechnisch Gestützten Geschäftsprozessen angelehnt an die natürliche Sprache

Stephan H. Sneed
Dr. Albert Fleischmann

jCOM1 AG
Lilienthalstrasse 17
85296 Rohrbach

stephan.sneed@jcom1.com
albert.fleischmann@jcom1.com

**Abstract:**

*Ein effektives und effizientes Prozessmanagement gelten heutzutage als wichtiger Erfolgsfaktor für Unternehmen. Als Folge wächst der Markt für Business Process Management (BPM) Tools nach einer Studie von Gartner weltweit um ca. 25 % und beträgt aktuell um die 2,5 Milliarden Dollar[1]. Parallel zu dieser wachsenden Nachfrage steigt aber auch die Unzufriedenheit mit dem heutigen Werkzeugangebot, in der Praxis wächst ebenso die Kluft zwischen den Erwartungen der Anwender und der Leistung der verfügbaren Produkte. Statt die Effektivität und Effizienz der Prozesse zu steigern und ihre Flexibilität zu sichern, tragen die heutigen BPM Tools zunächst einmal dazu bei, den Aufwand zu erhöhen und die Flexibilität zu reduzieren.*
*In diesem Beitrag werden die Autoren zunächst in Form einer Soll-Analyse untersuchen, welche organisatorischen und technischen Anforderungen an optimale BPM Werkzeuge gestellt werden. Aus diesen Forderungen lassen sich dann drei Theoreme ableiten, welche von einem BPM Ansatz erfüllt werden müssen, wenn er sich im Sinne der Herleitung optimal nennen möchte. Im Weiteren wird darauf eingegangen, in wie weit führende BPM Werkzeuge diese Theoreme erfüllen und was für Konsequenzen sich daraus ergeben. Schließlich wird die Standardsemantik der natürlichen Sprache als eine solche, effiziente Methode identifiziert und auf ihre praktische Verwendung mit der Business Process Management Suite jPASS demonstriert.*

**Keywords**: *Geschäftsprozesse, Anforderungen an Prozessmanagement, Effizienz, Modellbildung, automatische Code-Generierung, Prototypen, formale Theorien, natürliche Sprache, Programmieren in ganzen Sätzen*

## Einleitung

Am Anfang war das Wort.[2] So steht es geschrieben über die Entstehung der Welt. Nicht anders verhält es sich bei der Entstehung von Geschäftsprozessen und IT-Systemen. Am Anfang steht immer ein Gedanke, geäußert in natürlicher Sprache. Am Ende des Entstehungsprozesses sollen eine Reihe von IT-Systemen in Zusammenarbeit mit den Fachanwendern einen definierten Geschäftsprozess implementieren, der auf möglichst

[1] [Gart06]: Gartner, Dataquest Inside: BPMS Software , Market Size and Forecast, Worldwide 2006-2011

[2] Genesis 1.1, Altes Testament. Man streitet, in wie weit es sich hier um ein Übersetzungsfehler aus dem Hebräischen handelt und es nicht „Im Anfang..“ heißen müsste. Dies soll die unbestimmte Natur des Anfangs ausdrücken und wäre für die Betrachtungen zur Entstehung eines IT-Systems genau so treffend.

effiziente Weise zur Wertschöpfung eines Unternehmens beiträgt. Allerdings stellt die Entwicklung nur eines allein stehendes Systems, einschließlich Spezifikation, Entwurf, Programmierung und Test, eine gewaltige Herausforderung an alle Beteiligten. Laut dem jährlichen CHAOS Berichten scheitern 57% aller Entwicklungsprojekte [Cha07]. EAI und SOA tragen nicht gerade dazu bei, die ohnehin schwierige Lage zu erleichtern.
Die Integration dutzender Systeme hybrider Unternehmensformen zu durchgängigen Geschäftsprozessen, welche sich wiederum ständig an den sich immer rascher verändernden Markt anpassen müssen, überfordert manche IT-Abteilungen, vor allem wenn einen Großteil ihrer Kapazitäten - laut letzten Untersuchungen mindestens 60% - allein an die Erhaltung ihrer Legacy-Systeme gebunden sind. Nicht wenige Unternehmen befinden sich in einer Falle aus der sie mit eigener Kraft nicht herauskommen.
Sowohl die Änderung alter als auch die Einführung neuer Geschäftsprozessen verursacht einen überproportional hohen Aufwand, den die Anwender nicht in der Lage zu bezahlen sind. Neue Geschäftsprozesse ziehen je nach Modularität der jeweiligen IT-Landschaft etliche andere bestehender Systeme in ihre Strudel hinein. Es sind nicht nur die Anwendungssysteme, die betroffen sind, sondern auch die betrieblichen Datenbanken und die bereits implementierten Schnittstellen die angepasst oder gar neu implementiert werden müssen. Hinzu kommt, dass in einen unternehmensübergreifenden Geschäftsprozess neben strategischen Zielen und anderen Abhängigkeiten das detaillierte Know-How ganzer Fachabteilungen einfließt, was zu einem extrem hohen Abstimmungsbedarf führt.
Kostendruck und eine möglichst geringe Time-to-Market führen dabei in ein Dilemma, bei dem sich nüchtern betrachtet die Frage eröffnet, warum es nach wie vor Menschen gibt, die sich für den Leitungsposten einer IT-Abteilung freiwillig zur Verfügung stellen. Änderungen oder Neueinführungen von Geschäftsprozessen stellen ein Unternehmen aufgrund von technischen und psychologischen Fallstricken vor nicht unerhebliche Risiken, für die en général der IT-Leiter gerade stehen muss. Um so wichtiger erscheint für diesen die Frage nach dem optimalen BPM Ansatz, der, wie wir noch sehen werden, den mannigfaltigsten Anforderungen gerecht werden muss, um auf dem Prüfstand der Kosten-Nutzen-Analyse zu bestehen. Wir werden im Folgenden diese Anforderungen mittels eines stringenten Top-Down-Ansatzes herleiten und beginnen somit auf der Ebene des Managements.

# 1. Geschäftsprozesse aus Sicht des Managements

Spätestens seit Hammer und Champy in ihrem „Manifest for Business Reengineering“ den Kunden in die Mitte des betrieblichen Interesses gestellt haben [HaCh93], gelten für die meisten Bereiche der Wirtschaft heute die Bedingungen von Käufermärkten. Die Variantenvielfalt der Produkte (und auch der Prozesse, als Diensteilstunksprodukte verstanden) nimmt zu, während ihr Lebenszyklus sich verkürzt, ebenso wie die Bereitschaft der Kunden sinkt, lange Lieferzeiten in Kauf zu nehmen [ToAc03]. Aus diesen Forderungen lassen sich auf Management-Ebene drei wesentliche Punkte herauskristallisieren:

## 1.1 Der Geschmeidigkeitsfaktor: Flexibilität

Der Kunde hat den Anspruch, so geschmeidig wie irgend möglich mit seinem Anliegen durch die betrieblichen Hierarchien, Abteilungen und Organisationseinheiten zu gelangen. Betriebsinterne administrative oder informationstechnologische Barrieren sind dem Kunden heute nicht mehr zu vermitteln. Wer heute wegen einem einzigen Anliegen seine Daten

zweimal angeben muss, akzeptiert dies höchstens von einer Behörde noch ein drittes mal. Letztlich läuft heute alles auf den für den Kunden einzig entscheidenden Faktor hinaus: Den Geschmeidigkeitsfaktor. Für ihn stellt sich nur die eine Frage: Welcher Anbieter erfüllt meine Wünsche am geschmeidigsten? Diese Forderung gibt das Management an die Geschäftsprozessmodellierung weiter, welche als eine Form der IT-Anforderungsanalyse die Anforderungen eines Unternehmens unabhängig von den jeweilig darunter liegenden Technologien modellieren sollte [Rob99]. Sie bietet im idealen Fall eine größtmögliche Flexibilität und „sollte als offenes Konzept ganzheitlich angelegt sein“ [Sche07]. Nur so unterstützt sie die Kreativität der Mitarbeiter und stellt somit letztlich die Kunden zufrieden. Leider gibt es auf dem deutschen IuTK-Markt immer noch Betriebe mit großem Namen, die dies nicht begriffen haben und sich über schwindende Marktanteile beklagen. Damit Geschäftsprozesse flexibel sind, muss die Modellierung in der Lage sein, jede Situation der Wirklichkeit zu erfassen. Es kann nicht angehen, die Wirklichkeit zu verbiegen, nur um einem Tool gerecht zu werden, doch dies ist oft der Fall. So ist in etwa von sequentiellen Wertschöpfungsketten die Rede, obwohl die Realität eher aus Netzwerken sich synchronisierenden Aktivitäten besteht, weshalb wir in Zukunft von Wertschöpfungsnetzwerken sprechen werden. Generell kommen wir aber zu der Forderung:

> *F1: Der optimale BPM Ansatz muss ein Höchstmaß an Flexibilität schon bei der Beschreibung erlauben.*

### 1.2 Der Zeitfaktor: Time-to-Market

Neben der vom Kunden geforderten Flexibilität spielt noch ein zweiter Faktor eine Rolle: Der Zeitfaktor. In dem harten Wettbewerb, in dem die Unternehmen heute stehen, ist Flexibilität, so wichtig sie als Qualitätsfaktor auch sein mag, fast noch dem zeitlichen Faktor Time-to-Market unterzuordnen. „Studien aus verschiedenen Branchen belegen Verkürzungen von Produktlebenszyklen, Restlaufzeiten von Patenten und Marktschöpfungsdauern. Unter diesen Bedingungen gewinnt der Faktor Zeit an Bedeutung und tritt neben oder sogar vor die Wettbewerbsfaktoren Preis und Produktqualität“[3].

[3] Siehe [ScKR04]: S. 67, Organisatorische Trends, Organisation im Spannungsfeld zwischen IT und Wettbewerb

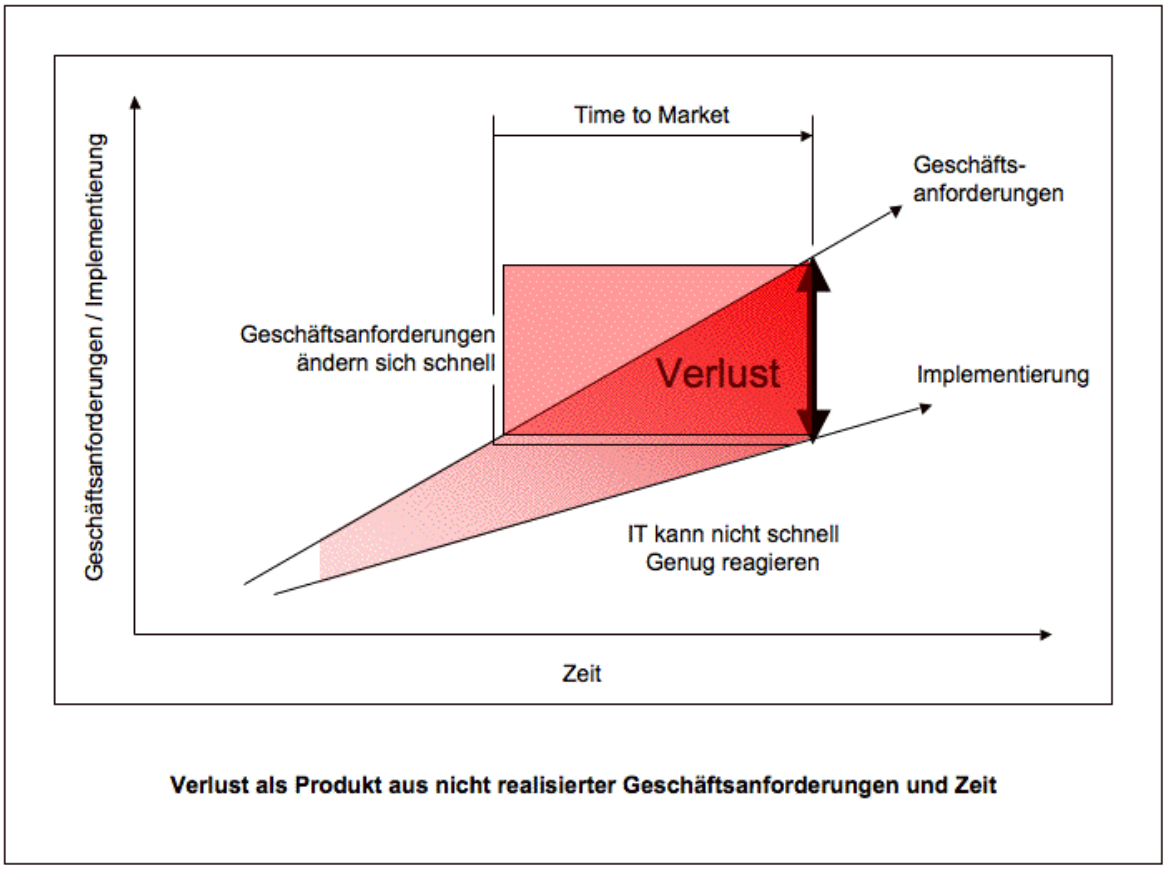



**Abbildung 1: Verlust als Produkt nicht realisierter Geschäftsanforderungen und Zeit**

Um auf den Kunden und seinem Wunsch nach Geschmeidigkeit zurückzukommen: In hart umkämpften Käufermärkten heißt die Frage also nicht mehr „Welcher Anbieter erfüllt meine Wünsche am Geschmeidigsten?“ sondern besser „Wer ist jetzt schon wie geschmeidig?“
In dem Moment, in dem ein Konkurrenzunternehmen im Gegensatz zum eigenen in der Lage ist, einen beim Kunden beliebten Service anzubieten, tickt die Uhr gegen das eigene Unternehmen (siehe Abb. 1: Verlust als Produkt aus nicht realisierten Geschäfts-anforderungen). Das Management ist somit gezwungen, für eine schnelle und somit effiziente Implementierung von Prozessen zu sorgen. Effizienz zahlt sich in diesem Fall für das Unternehmen zweifach aus, denn eine schnelle Implementierung bedeutet weniger Kosten für die Implementierung und weniger Verlust durch eine zu hohe Time-to-Market, während es im Fall von Ineffizienz doppelt bestraft wird. Natürlich kommt es letztlich darauf an, wie schnell die Prozesse operativ sind. Dies hängt jedoch nicht nur von dem gewählten BPM Ansatz von einer Reihe weiteren Faktoren ab, wie in etwa der Modularität der darunter liegenden IT-Systemen.
Diese liegen aber nicht im Fokus dieses Aufsatzes, weshalb wir den Zeitfaktor anders definieren müssen. Was ein BPM Ansatz diesbezüglich höchstens leisten kann, ist die Zeit von der ersten Definition der Prozessanforderungen in natürlicher Sprache ab bis maximal hin zu einem bereits ausführbarem Prozessmodell, welches schon die Schnittstellen für die Anbindung an die darunter liegende Systemlandschaft bereit hält, zu verkürzen. Innerhalb dieser Projektmeilensteine kann ein BPM Ansatz helfen, Zeit zu sparen. Somit fordern wir:

*F2: Der optimale BPM Ansatz muss schnell zu einer Implementierung führen*

## 1.3 Der Wettbewerbsfaktor: Effizienz

So wichtig zunächst die beiden Faktoren Flexibilität und Time-to-Market sind, um sich einen Wettbewerbsvorteil zu verschaffen, so wichtig ist auch die Effizienz, d.h. die Wirtschaftlichkeit, um diesen Vorteil auch langfristig zu wahren. Wirtschaftlichkeit wiederum ist das Verhältnis aus Ertrag und Aufwand [ThAc03]:

$$Wirtschaftlichkeit = \frac{Ertrag}{Aufwand}$$

In dem Moment, in dem ein Konkurrenzunternehmen dieselbe Flexibilität und Time-to-Market mit weniger Kosten erreicht, wird es langfristig an einem vorbeiziehen. Es ist daher nicht nur dafür zu sorgen, dass die Geschäftsprozesse eine hohe Flexibilität und eine kurze Time-to-Market erreichen, sondern dass sie auf eine Weise definiert werden, die eine möglichst effiziente bzw. wirtschaftliche Abarbeitung der Gesamtaufgabe ermöglicht. Wir müssen davon ausgehen, dass die Prozessbeteiligten in einem Unternehmen, also die Organisationsgruppen und Personen wissen, wie sie in diesem Sinne Effizienz erreichen können[4]. Sie müssen aber in der Lage sein, ihre detaillierten Kenntnisse zur Effizienzsteigerung in den Prozess einbringen zu können. Im besten Fall können die Fachanwender ihr persönliches Verhalten in einem Prozess direkt in das Prozessmodell einfließen lassen. Wie auch immer dies erreicht wird, in jedem Fall müssen die definierten Prozesse den höchst möglichsten Grad an Wirtschaftlichkeit bzw. Effizienz ereichen. Wir müssen daher fordern:

*F3: Der optimale BPM Ansatz muss zu effizienten Prozessen führen*

## 1.4 Der ewige Faktor: Die Kosten

Es ist eigentlich müßig zu erwähnen, dass das Management sich neben den Forderungen an ein mögliches Investitionsobjekt auch immer für dessen Preis interessiert, der sich diesen gegenüber in optimaler Weise stets indirekt proportional verhält. Die maßgebliche Größe ist hier die Produktivität als Verhältnis zwischen ausgebrachten Faktorkombinationen und eingebrachten Produktionsfaktoren [ThAc03]:

$$Produktivität = \frac{Ausbringungsmenge\ der\ Faktorkombinationen}{Einsatzmenge\ an\ Produktionsfaktoren}$$

Optimal im Sinne der Produktivität ist also eine möglichst geringe Einsatzmenge an Produktionsfaktoren wie Software-Tools bzw. geringe Kosten für diese. Einer kleinen Ausführung bedarf es aber vielleicht um zu klären, woraus sich diese Kosten bei einem PBM Ansatz eigentlich zusammensetzen. Zunächst ist da der Anschaffungspreis sowie die Kosten für den Betrieb was, je nach Lizenzmodell, auch zusammenfallen kann. Hinzu kommen aber noch anderen Kosten, nämlich jene Kosten, welche durch die Benutzung eines BPM Ansatzes entstehen. Jedes Prozessmodell verursacht Kosten. Zunächst verursacht es Kosten bei der Erstellung. Anforderungen in natürlicher Sprache müssen in einem Prozessmodell ausgedrückt werden.

[4] Ist dies nicht der Fall, dann wäre dies ein Problem der Personalabteilung und somit nicht im Fokus von Geschäftsprozessmanagement.

Anschließend entstehen Kosten, das Prozessmodell zu implementieren. Dabei müssen aus dem Modell konkrete Anforderungen für die einzelnen Prozessbeteiligten abgeleitet werden. Dies sind entweder Handlungsanweisungen an Personen oder Konzepte für IT-Systeme in natürlicher Sprache oder weiteren Modellen. In beiden Fällen, bei den Kosten der Erstellung wie von den Kosten für die Implementierung können wir von Transformationskosten[5] sprechen. Die Frage hier lautet: Was kostet es, Anforderungen in natürlicher Sprache in ein Prozessmodell zu transformieren, und was kostet es, das Prozessmodell in Form eines zumindest auf abstrakter Eben ausführbaren Prozesses zu implementieren und spezifizierende Modelle für die letztendliche Implementierung daraus abzuleiten? Die Kosten für die letztendliche Implementierung sind, wie bei dem Faktor Zeit, von weiteren, den BPM Ansatz nicht betreffenden Faktoren abhängig und werden in unserer Betrachtung daher vernachlässigt, nicht jedoch Kosten für jede Transformation, zunächst in das Prozessmodell, dann aus dem Prozessmodell heraus. Somit müssen wir noch eine letzte Forderung aus dem Management aufstellen:

*F4: Der optimale BPM Ansatz verursacht geringe Kosten*

## 2. Geschäftsprozesse aus Sicht der IT

In Kapitel 1 haben wir die Sicht des Managements dargestellt und vier Forderungen definiert, die, so gut es eben geht, umgesetzt werden sollen. Diese Aufgabe fällt den IT-Abteilungen der jeweiligen Unternehmen zu, denn die IT wird „in der Rolle des Ermöglichers (enabler) - oder des Behinderers (disabler)“[6] des Prozessparadigmas gesehen. Die IT-Abteilung trägt somit die Hauptverantwortung und nicht zuletzt auch den Aufwand und die Kosten für die Definition und die Umsetzung von Geschäftsprozessen. Für den Fall, das neue Prozesse definiert bzw. alte geändert werden sollen, stellen sich die bereits erwähnten zwei Transformationen als Aufgaben (siehe Abb. 2: Die zwei Transformationen von Geschäftsprozessen).

[5] Nicht zu verwescheln mit der betriebswirtschaftlichen Definition von Transformationskosten

[6] [Picot03]: S. 193: Informations- und kommunikationsorientierte Gestaltung der Unternehmensprozesse

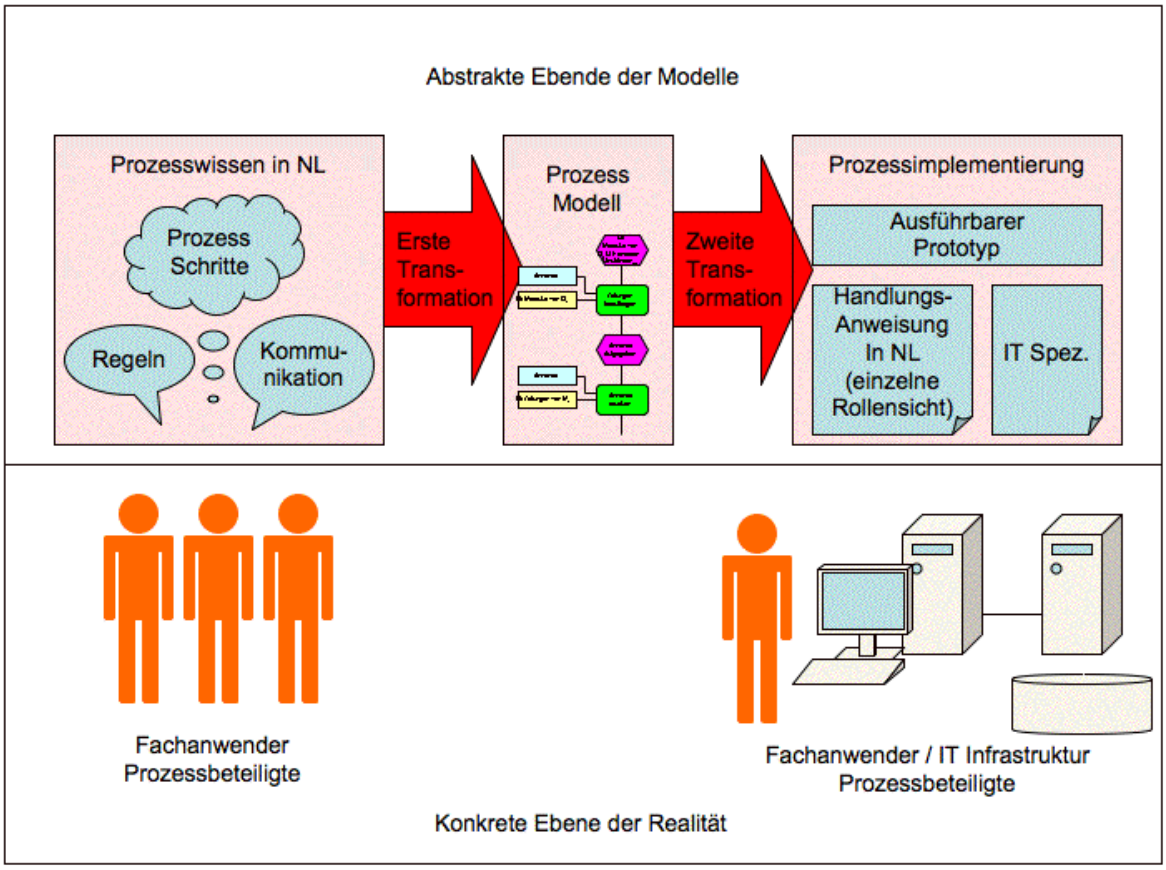



**Abbildung 2: Die zwei Transformationen von Geschäftsprozessen**

## 2.1 Die erste Transformation: Erstellen der Modelle

Zunächst müssen die Prozessmodelle erstellt werden. Das Prozesswissen steckt, wie bereits erwähnt, in den Köpfen der Fachanwender, welche sich in natürlicher Sprache mitteilen. Addieren wir alles Prozesswissen aller Fachanwender, so haben wir den (mit diesem Wissensstand) optimalen Prozess, vorliegend in natürlicher Sprache. Theoretisch. In der Praxis hat sich gezeigt, dass kaum jemand in der Lage ist, sein gesamtes Wissen zu einem Thema zu einem Zeitpunkt X auf den Tisch zu legen. Die weite Verbreitung von inkrementellen Projektansätzen in der IT mag hier als Beweis genüge tun. Was sich in der komplexen Welt der IT aus demselben Grund ebenso einer gewissen Beliebtheit erfreut, ist der Prototyp. Durch ihn kann man eine Vorstellung entwickeln, eine Vorstellung, von dem was sein wird, und mit ihm kann diese Vorstellung kostengünstig optimiert werden. Bei beiden Vorgehen, inkrementell oder prototypisch, wird derselbe Ansatz verfolgt: Es gilt, zunächst mit möglichst wenig Mitteln und unter Vernachlässigung von allen Details etwas zu implementieren, was man als „den Kern der Sache“ oder „das Wesentliche“[7] bezeichnen könnte. Hat man dann einen Eindruck davon, was diesen Kern nun ausmacht und was das Wesentliche genau ist oder zu sein hat, kann man sich daran machen, eine Schale um den nun ausgereiften Kern zu bauen. So viel in der Theorie zur der Methode, welche die Forderung F4 (geringe Kosten) erfüllt. In der Praxis eines IT Projektes bedeutet dies so viel wie dass die Erfassung des Wesentlichen in den frühen Projektphasen abgeschlossen sein muss. Das

[7] Das Wesentliche als das „die Wesenheit einer Sache beschreibende“.

Wesentliche ist in unserem Fall der abstrakte Prozess, das Prozessmodell. Bevor an dessen Implementierung gegangen wird, muss sicher gestellt werden, dass man mit dem Prozess, so wie er im Fachkonzept oder dem DV-Konzept definiert ist, den Nagel auf den Kopf trifft, denn laut einer alten und allseits bekannten Studie von Gartner kostet die Beseitigung von Fehlern in den Spätphasen eines IT-Projektes das 10 bis 100fache gegenüber deren Beseitigung in den frühen Projektphasen (siehe Abb. 2: Kosten der Fehlerbehebung)
Zwar wird gefordert, auch den Test der Geschäftsprozesse in den Systemtest mit einzubeziehen [Sne07], aber auch hier gilt: Getestet wird gegen die Spezifikation. Ist diese fehlerhaft, werden diese Fehler erst im laufenden Betrieb offenbar und sorgen so für die maximalen Kosten.

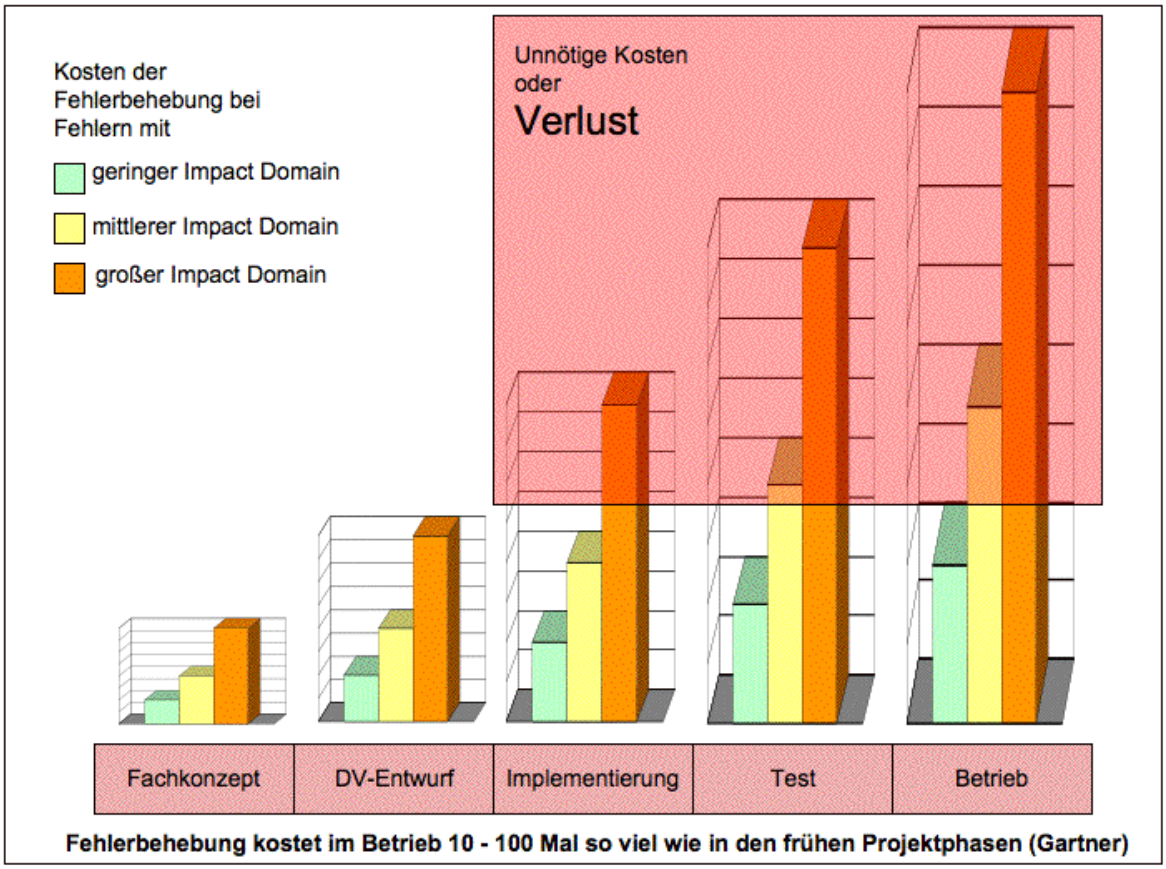



**Abbildung 3: Kosten der Fehlerbehebung**

Wir werden zu einem späteren Zeitpunkt noch genau darauf eingehen, wie die Transformation von Wissen in Form von natürlicher Sprache in ein Prozessmodell transformiert werden kann und wie eine im Sinne von Effizienz optimale Lösung aussehen könnte. An dieser Stelle genügt es jedoch zu sagen, dass in jedem Fall das Ergebnis dieser Transformation inhaltlich korrekt sein muss. Dies wiederum bedeutet jedoch nichts anderes, als dass die inhaltliche Qualitätssicherung der Prozessmodelle vor deren Aufnahme in eine Spezifikation (Fachkonzept, spätestens DV-Konzept) abgeschlossen sein muss. Dadurch aber fällt die Ermöglichung einer inhaltlichen Qualitätssicherung in den Bereich der Prozessdefinition und wird somit zur Anforderung an einen BPM-Ansatz, denn wo oder durch was soll sie sonst erfolgen? Wir formulieren also:

*F5: Der optimale BPM Ansatz unterstützt eine inhaltliche Qualitätssicherung*

### 2.2 Die zweite Transformation: Prototyp und Spezifikation

Im obigen Kapitel haben wir uns mit der Transformation in das Modell beschäftigt. Wie dies im Einzelnen geschieht und geschehen kann, wird später behandelt werden, zunächst genügte uns hier eine Zieldefinition, nämlich die eines inhaltlich qualitätsgesicherten Prozessmodells. Unter demselben Aspekt werden wir die zweite Transformation betrachten, die Transformation aus dem Prozessmodell heraus. Wenn ein inhaltlich qualitätsgesichertes Prozessmodell das optimale Ergebnis der ersten Transformation ist, was ist dann das optimale Ergebnis der zweiten Transformation? Nun. Aus einem Prozessmodell sollen möglichst eindeutige Handlungsanweisungen für Personen und Personengruppen sowie Spezifikationen für maschinelle Akteure ausgehen. Wenden wir uns zunächst den Personen zu. Wenn der Prozess von den beteiligten Personen „gelebt“ werden soll, setzt dies voraus, das jede der einzelnen Personen oder Personengruppen aus dem Prozess die eigene Handlungsweise ohne Probleme ableiten kann. Wir sind bereits darauf eingegangen, wie schwierig es ist, komplexe Sachverhalte so darzustellen, dass sich einem das Wesen des Sachverhalts offenbart. Dabei hat sich das eigene „Erleben“ von Sachverhalten als eine der wirkungsvollsten Methoden herausgestellt. Dies wird hinreichend belegt durch den bei den Kunden so beliebten Extreme Programming Ansatz von Kent Beck, mit dem ein möglichst schnelles Erreichen eines ausführbaren Prototyps angestrebt wird. [BecXX]. Wir fordern also für den optimalen BPM Ansatz:

*F6: Der optimale BPM Ansatz liefert einen ausführbaren Prozessprototyp*

Prozessmanagement ist aber nur dann interessant, wenn maschinelle bzw. automatische Akteure zum Einsatz kommen. Einfach gesprochen handelt es sich hierbei um Systeme, welche Daten vom Prozess bekommen, sie verarbeiten und an den Prozess zurückgeben. Allerdings müssen diese Systeme in der Regel angepasst werden, um sich auf die gewünschte Weise in den Prozess integrieren zu können. Daher ist in diesem Fall das Ergebnis eine Spezifikation für eben jene Anpassung, in seltenen Fällen eine Spezifikation für eine Neuentwicklung. So oder so besteht diese Spezifikation aus einer Sammlung von Anwendungsfällen für jedes System, denn jeder Systemverwalter interessiert sich zunächst einmal dafür, welche Funktionalität er dem Prozess bereitstellen muss. Daher müssen wir fordern:

*F7: Der optimale BPM Ansatz liefert Anwendungsfälle für IT-Systeme*

## 3. Geschäftsprozesse aus Sicht der Beteiligten

Nach dem wir die Sicht des Managements und der IT auf Geschäftsprozesse beleuchtet und einige Forderungen identifiziert haben, wenden wir uns nun der eigentlich wichtigsten Gruppe zu: Den Fachanwendern. Geschäftsprozesse haben für sie eine zweifache Bedeutung: Zunächst bestimmt ihr Handeln den Geschäftsprozess (erste Transformation) und anschließend bestimmt der Geschäftsprozess ihr Handeln (zweite Transformation). Trotz dieser engen Bindung zwischen Prozess und Prozessbeteiligten auf theoretischer Ebene sieht es in der Praxis oft ganz anders aus: Die Prozessbeteiligten oder Fachanwender interessieren sich nur mäßig bis gar nicht für die Definition von Geschäftsprozessen. Dies ist im Wesentlichen ein Anreizproblem und ein klassisches Beispiel von Dilemma Strukturen der

Spieltheorie [Hom02]. Die meisten Fachanwender sehen ihr operationales Geschäft als wichtiger an als die Spezifikation von Prozessen und IT-Systemen, deren tatsächlicher Einsatz sowieso in den Sternen steht.[8] Es wird von ihnen verlangt, sich mehr oder weniger in ihrer Freizeit an der Definition von Geschäftsprozessen zu beteiligen, dabei ihr Know-How offen zulegen und sich in für sie abstrakte Prozessmodelle einzuarbeiten. Die Dilemmastruktur, die sich hier anwenden lässt, entspringt dem klassischen Gefangenendilemma. Das Beste wäre es für alle Fachanwender, sie würden sich aktiv beteiligen und einen ordentlichen Prozess definieren. Beteiligt sich aber einer nicht, schleichen sich Inhaltliche Fehler ein, was, wie wir bereits wissen, teuer werden kann und ab einer gewissen Quantität für das Projekt zur Gefahr wird. Am Ende kommt nichts dabei heraus und diejenigen, die sich beteiligt haben, waren die Dummen. Im Gefangendilemma ist daher „nicht kooperieren“ die dominante Strategie. Dies sind natürlich in erster Linie Management-Probleme, zu denen ein BPM Ansatz wenig beitragen kann, aber in einem Punkt kann er es doch: Er kann die Hürde, die es für einen nicht an abstrakten Modellen interessierten Menschen bedeutet, sich mit abstrakten Modellen zu befassen, entsprechend hoch oder niedrig ansetzen und somit den Anreiz, dies zu tun, beeinflussen. Somit ist es ein Anliegen der Fachanwender, dass die definierten Prozesse leicht und intuitiv verständlich dargestellt werden.
Intuitiv verständlich für eine Fachabteilung ist etwas dann, wenn es in seiner Struktur nahe an der Struktur der natürlichen Sprache liegt. Dies ist die Sprache in der eine Fachabteilung seine „Modelle“ beschreibt. Es gilt also die Vorteile der natürlichen Sprache wie allgemeine Verfügbarkeit und auch gewisse Formalisierungsansätze wie Grundbedeutungen für Worte, Stabilität (Goethe versteht man heute noch) und Standardsatzsemantiken wie Subjekt, Prädikat und Objekt zu nutzen, aber die Unschärfen wie Mehrdeutigkeiten von Worten zu vermeiden.[9] Bei der Anwendung von natürlicher Sprache für Anforderungsbeschreibung jeder Art gelten ganz generell die üblichen Qualitätskriterien für Anforderungen [Rup07]. Unter Berücksichtigung dieser Kriterien fordern wir zunächst unabhängig von der Art und Weise, wie dies geschehen kann:

*F8: Der optimale BPM Ansatz stellt die Prozesse intuitiv verständlich dar*

Wir können nun also davon ausgehen, dass die Prozessbeteiligten sich nun einen Überblick über den Prozess verschaffen können. Dies ist nicht zuletzt auch für sie entscheidend, da sie ja dennoch mit ihrer Unterschrift unter dem jeweiligen Fach- oder DV-Konzept, zu dem auch die Prozesse gehören, dafür einstehen, letztere für implementierungswürdig zu halten. Noch wichtiger als die Übersicht ist jedoch die Möglichkeit für sie, ihre eigene Rolle entsprechend im Prozess wieder finden zu können. Erst wenn ein Prozessbeteiligter aus dem Prozessmodell entnehmen kann, was auf ihn ganz persönlich zu kommt, wird er sich optimal einbringen können. Erst, wenn er sich in dem Prozess wieder findet, wird er die Frage, in wie weit der definierte Prozess die Sachlage korrekt widerspiegelt, mit Sicherheit beantworten und so einer inhaltlichen Qualitätssicherung genüge tun können. Wenn wir den optimalen Prozess als die Aggregation von optimal beschrieben Verhaltensweisen aller seiner Prozessrollen auffassen, müssen wir fordern:

*F9: Der optimale BPM Ansatz kann die Sicht einzelner Rollen darstellen*

[8] Wie bereits erwähnt scheitern laut den jährlichen CHAOS Berichten 57% aller Entwicklungsprojekte [Cha07].

[9] siehe [Hol99]: Holl, A. (1999). Empirische Wirtschaftsinformatik und evolutionäre Erkenntnistheorie (Bde. Becker, Jörg et al. (ed.)).

## 4. Zusammenfassung bisheriger Forderungen: Code-Generierung

In den Kapiteln zwei und drei haben wir die Forderungen des Managements an die IT und die Prozessbeteiligten weitergegeben und daraus weitere Forderungen abgeleitet, von denen wir nun einige zusammenfassen können:

> Um sicher zu gehen, dass keine inhaltlichen Fehler mehr in den Prozessmodelle stecken, forderten wir eine inhaltliche Qualitätssicherung:
> *F5: Der optimale BPM Ansatz unterstützt eine inhaltliche Qualitätssicherung*
>
> Als ideales Medium für einen humanen Akteur, der sein Verhalten aus einem abstraktem Prozess ableiten soll, haben wir den Prozessprototyp identifiziert:
> *F6: Der optimale BPM Ansatz liefert einen ausführbaren Prozessprototyp*
>
> Als ideales Medium für einen Systemverwalter, der aus einem abstrakten Prozessmodell die entsprechenden Funktionszugriffe auf sein System ableiten soll, haben wir eine Aufzählung der Anwendungsfälle identifiziert:
> *F7: Der optimale BPM Ansatz liefert Anwendungsfälle für IT-Systeme*
>
> Um den Anreiz für die Prozessbeteiligten zu erhöhen, sich an der Gestaltung optimaler Prozesse zu beteiligen, haben wir eine möglichst einfache Darstellung der Prozessmodelle gefordert:
> *F8: Der optimale BPM Ansatz stellt die Prozesse intuitiv verständlich dar*
>
> Um zu garantieren, dass die Prozessbeteiligten sich im Prozess wieder finden und das Verhalten ihrer Prozessrolle mit dem ihrer tatsächlichen Rolle im Unternehmen abgleichen zu können, haben wir gefordert:
> *F9: Der optimale BPM Ansatz kann die Sicht einzelner Rollen darstellen*

Wie wir sehen, fallen F5 und F6 bereits zusammen. Ein ausführbarer Prozessprototyp (F6) liefert bereits alle Möglichkeiten für eine inhaltliche Qualitätssicherung (F5), da die Fachanwender den Prozess simulieren und das Erlebnis der Simulation auf das ihrer täglichen Arbeit projizieren können. Ebenso verhält es sich mit F8. Die Möglichkeit eines „Erlebens“ eines Prozesses durch Simulation an einem Prototyp ist neben der graphischen sicher eine der intuitivsten Möglichkeiten der Darstellung. Auf F7 werden wir in einem späteren Kapitel noch eingehen und F9 folgt gleich im Anschluss. Zunächst können wir jedoch aus F5 (Qualitätssicherung), F6 (Prozessprototyp) und F8 (Intuitive Darstellung) die Konsequenzen ziehen: Wir möchten aus dem Prozessmodell ausführbaren Programmcode ableiten können. Wenn wir dabei ein iteratives Vorgehen unterstützen und die Forderungen F2 (Geschwindigkeit) und F4 (Geringe Kosten) erfüllen wollen, muss diese Ableitung vollständig automatisch geschehen.
Automatische Code-Generierung ist hier somit mehr als ein modisches Schlagwort. Es ist die Konsequenz der stringent entwickelten Forderungen, deren Ursprünge sich aus dem ökonomischen Zwang zur Optimierung, Effizienzsteigerung und Kostenreduzierung ableiten lassen. Zusammenfassend läst sich also sagen:

> *F10: Der optimale BPM Ansatz liefert automatisch ausführbaren Code*

Kommen wir nun zu F9 (Einzelne Rollensicht). Wir haben bereits erläutert, wie wichtig es für die Prozessbeteiligten und eine inhaltliche Qualitätssicherung ist, dass die Prozessbeteiligten sich bei der Prozesssimulation in ihrer eigenen Rollensicht wieder finden. Dies bedeutet, das der ausführbare Code der Simulation diese Rollensicht einschließen muss. Wir fordern schlussendlich:

*F11: Der ausführbare Code spiegelt das Verhalten der einzelnen Rollen wieder*

## 5. Drei BPM Theoreme

Betrachten wir die letzten beiden Forderungen. Einerseits wollen wir aus einem Prozessmodell vollautomatisch Code ableiten, andererseits soll dieser Code das Verhalten der einzelnen Rollen widerspiegeln. Im Sinne einer modularen IT-Architektur wäre es somit ein nahe liegender Gedanke, den Code anhand der einzelnen Prozessrollen zu organisieren. Jede Prozessrolle entspricht einem Modul oder einer Klasse. Das Verhalten einer einzelnen Klasse im System spiegelt das Verhalten der einzelnen Rolle im Prozess wider. Vielleicht gäbe es auch noch andere Möglichkeiten, doch diese scheint intuitiv die jenige zu sein, welche die reelle Situation am besten beschreibt. Außerdem, und dies mag hier genügen, wäre durch diese Organisation des Codes die Forderung F11 erfüllt, wonach der ausführbare Code das Verhalten der einzelnen Rollen bzw. Prozessbeteiligten widerspiegelt. In diesem Sinne würde sich also aus einem Prozessmodell ein verteiltes System aus Programmen (bzw. Modulen oder Klassen) ableiten lassen, in dem jedes dieser Programme das Verhalten einer Prozessrolle repräsentiert. Wir fassen dies in einer zentralen Forderung zusammen:

*F12: Ein BPM Ansatz ist optimal, wenn sich aus einem Prozessmodell dieses Ansatzes vollautomatisch ein ausführbares System aus Programmen generieren lässt, von denen jedes Programm einen Prozessbeteiligten repräsentiert, dessen Verhalten durch eben dieses Programm (im Sinne des Prozessmodells) vollständig und abgeschlossen definiert ist und welches System bei Ausführung den modellierten Prozess widerspiegelt.*

Nehmen wir diesen Satz als eine Zusammenfassung unserer zentralsten Forderungen hin. Damit liefern wir jedoch nichts Neues. Diese Forderung ist letztendlich nichts weiter als eine präzisere Formulierung des alten Gedankens, aus einem Prozessmodell automatisch das Verhalten einzelner Rollen abzuleiten. Die Anfänge der teilweise glücklosen Versuche einiger BPM Hersteller, dieses Ziel zu erreichen, reichen nun ja auch schon einige Jahre zurück. Für die Umsetzung eines Geschäftsprozesses von seiner natürlichsprachlichen ersten Formulierung durch die Fachabteilung bis zum IT-gestützten Workflow sind häufig mehrere Umsetzungsschritte notwendig. So wird die fachliche Geschäftsprozessbeschreibung, in Deutschland häufig erstellt mit ARIS, manuell in eine IT-taugliche Beschreibung umgesetzt.“[10] Häufig kann die Fachabteilung nicht beurteilen ob die IT-taugliche Prozessbeschreibuung noch der ursprünglichen fachlichen Intension entspricht. Wir werden im Weiteren einige Prämissen einführen und daraus drei Theoreme für BPM Ansätze

[10] Prof. Dr. Thomas Allweyer Der Weg vom Geschäftsprozess zum Workflow Fachlichen Modellierung und Entwicklung eines ausführbaren Workflows mit Intalio|BPMS, siehe http://www.bpm-netzwerk.de/content/articles/

herleiten, bei deren Berücksichtigung einer erfolgreichen Code-Generierung direkt aus dem fachlichen Modell im Sinne des obigen Satzes nichts mehr im Wege steht.

Damit wir automatisch aus einem Prozessmodell Code generieren können, benötigen wir eindeutige Zuweisungsregeln, welche sich dann im Sinne eines Transformationsprogramms implementieren lassen. Daraus ergibt sich P1:

> *P1: Eine vollautomatische Ableitung von Code aus Prozessmodellen entspricht einer bijektiven Abbildung, also einer sowohl injektiven als auch surjektiven.*
> - *Alle Elemente der Definitionsmenge, d.h. Konstrukte des Prozessmodells werden eindeutig in Elemente der Wertemenge, d.h. Programme überführt (injektiv.)*
> - *Für jedes generierte Programm kann eindeutig festgelegt werden, welchem fachlichen Modell es entspricht (surjektiv).*

Wenn es sich bei der Generierung von Code aus einem Prozessmodell also um eine bijektive Abbildung (Funktion) handelt, gelten entsprechend auch die Bedingungen für Definitions- und Wertemengen derartiger Abbildungen.

> *P2: Es existieren nur dann bijektive Funktionen der Menge A (Prozessmodelle) auf die Menge B (Systeme aus Programmen), wenn beide Mengen auf formale Weise eindeutig und vollständig definiert sind.*

In Bezug auf die Wertemenge, also die Menge aus Systemen, welche die Prozessmodelle widerspiegeln, brauchen wir uns diesbezüglich nicht zu sorgen. Jede Programmiersprache ist formal definiert und der Compiler sorgt dafür, dass ein Programm oder ein System aus Programmen nur aus korrekt Gebildete Anweisungen besteht. Würde man eine Programmiersprache als Theorie auffassen, dann wären korrekt gebildete Anweisungen Formeln dieser Theorie. Wir können also P3 einführen:

> *P3: Ein ausführbares System aus Programmen einer Programmiersprache besteht immer aus korrekten Anweisungen (Formeln) dieser Sprache (Theorie).*

Die Definitionsmenge erfüllt also bereits per Definition unsere Anforderungen. Wenn wir also nach eine bijektive Abbildung von Prozessmodellen auf Systeme aus Programmen suchen, und behaupten, das eine derartige existiert, vorausgesetzt es gibt eine definierte Definitions- und Wertemenge und wir ferner behaupten, dass dies für die Wertemenge immer der Fall ist, können wir schließen: Die Wertemenge, d.h. die Prozessmodelle müssen ebenfalls Konstrukte einer formalen Theorie sein.

**Erstes Geschäftsprozesstheorem:**

Eine automatische Ableitung eines Programms (bzw. eines verteilten Systems, bestehend aus Programmen) aus einem Prozessmodell ist genau dann möglich, wenn den Prozessmodellen eine streng formale Theorie zu Grunde liegt (d.h. wenn sie aus korrekt gebildeten Formeln einer formalen Theorie bestehen).

Mit diesem Theorem sind wir also in der Lage, überhaupt compilierbaren Code aus einem Geschäftsprozessmodell abzuleiten. Wie wir sehen werden, ist diese Bedingung für die Erfüllung der zentralen Forderung F12 zwar notwendig, jedoch nicht hinreichend. Genau genommen: Wir können nun aus einem Prozessmodell, welchem eine formale Theorie zu Grunde liegt, ein System aus Programmen ableiten. In wie weit sich diese Programme in ihrem Zusammenspiel so verhalten, wie die einzelnen Rollen im Prozess, ist noch offen. Die weitere Argumentation liegt jedoch auf der Hand. Damit diese Programme überhaupt im Sinne des Prozesses ausgeführt werden können, müssen sie miteinander kommunizieren, denn jedes dieser Programme repräsentiert ja eine Prozessrolle, welche in der Realität ebenfalls miteinander kommunizieren. Andernfalls könnten immer nur Teile des Prozesses, nämlich die jenigen, die von einer einzelnen Rolle allein bewerkstelligt werden, ausgeführt werden. Somit gilt auch für die Programme, welche die Rollen repräsentieren bzw. simulieren:

> *P4: Ein System einzelner Programme, welche die Prozessbeteiligten repräsentieren, ist nur dann im Sinne des Prozesses ausführbar, wenn diese Programme die im Prozess festgelegte Kommunikation abbilden.*

Es ist also notwendig, dass der Kommunikationsaspekt innerhalb der Wertemenge, also innerhalb der verteilten Systeme, existiert. Wenn wir das Kommunikationsverhalten der Programme wiederum zu den Ergebnissen einer bijketiven Abbildung zählen, gelten hier im speziellen natürlich die selben Bedingungen wie für die gesamte Abbildung im Allgemeinen: In die Wertemenge abgebildet werden kann nur das, was formal in der Definitionsmenge definiert wurde. Somit ergibt sich das zweite Theorem:

**Zweites Geschäftsprozesstheorem:**

Eine automatische Ableitung eines im Sinne des Prozesses ausführbaren Programms (bzw. eines verteilten Systems, bestehend aus Programmen) aus einem Prozessmodell ist nur dann möglich, wenn die Kommunikation zwischen den Prozessbeteiligten in dem Prozessmodell durch die zu Grunde liegende Theorie formalisiert wird.

Aus dem zweiten Theorem lässt sich jedoch noch ein drittes ableiten, welches streng genommen zwar nur eine Konsequenz des zweiten ist, wir jedoch auf Grund seiner Aussagekraft nicht darauf verzichten wollen, es als eigenständigen Satz aufzuführen. Wir haben also behauptet, dass die Kommunikation zwischen den Prozessbeteiligten im Prozessmodell formalisiert sein muss. Unter Kommunikation verstehen wir im einfachsten Sinne Konstrukte der Form: „Rolle A“ sendet „Nachricht XY“ an „Rolle B“.

Würde man derartiges mit den Mitteln der Prädikatenlogik formalisieren, so hätten wir zunächst einmal eine Logik dritter Stufe. Der Ausdruck wäre dann von der Form: P(x,y,z), wobei P für das Prädikat (Die Begriffe Subjekt und Prädikat haben in der Prädikatenlogik und im Bereich der Grammatik der natürlichen Sprachen unterschiedliche Bedeutungen)[11] steht, in diesem Fall „senden“, die erste Variable (x) für das Subjekt (Rolle A), die zweite (y) für das

[11] [HoyXX]: P. Hoyningen-Huene, Formale Logik, Eine philosophische Einführung, Reclams Unversalbibliothek 9692 W. Detel, Grundkurs Philosophie, Band 1 Logik, Reclams Universalbibliothek 18468

Akkusativobjekt „Rolle B“ und die dritte Variable (z) für das weitere Akkusativobjekt, die „Nachricht xy“.

Wie die Informatik beweist, ist es immer möglich, an sich dreistufige Sachverhalte auch durch eine zweistufige Logik auszudrücken. Dies gelingt durch eine Schachtelung der Ausdrücke, setzt aber entsprechend eindeutige Relationen voraus. Man könnte Kommunikation also auch zweistufig formalisieren: P(x,y). Dabei wäre P weiterhin das Prädikat „senden“, die Variable x entspräche dem Subjekt und y einer Relation auf ein weiteres Tuppel, nämlich der Nachricht, bestehend aus Inhalt und Empfänger. Hierbei benötigt man jedoch eine Tabelle der Nachrichtentuppel. Wie ersichtlich ist es also möglich, Kommunikation mit einer zweistufigen Logik auszudrücken, allerdings unter der Nebenbedingung einer eindeutigen Schachtelung von Relationen. Prinzipiell ist Kommunikation dreistufig. Auch wenn diese Dreistufigkeit, wie ersichtlich, mit einer zweistufigen Logik implementiert werden kann, muss bei einem Geschäftsprozess en général zunächst von einer Dreistufigkeit ausgegangen werden.

**Drittes Geschäftsprozesstheorem:**

Eine automatische Ableitung eines im Sinne des Prozesses ausführbaren Programms (bzw. eines verteilten Systems, bestehend aus Programmen) aus einem Prozessmodell ist nur dann möglich, wenn das Prozessmodell direkt oder indirekt dreistufige Ausdrücke formalisiert.

# 6. Konsequenzen aus den drei BPM-Theoremen

Wir haben im vorigen Kapitel also die Vorraussetzungen für eine Code-Generierung aufgestellt und behauptet, dass diese nur möglich ist, wenn die drei Theoreme erfüllt sind. Gilt jedoch im Umkehrschluss, dass eine Code-Generierung aus BPM Ansätzen, welche die Theoreme nicht erfüllen, unmöglich ist? Die Antwort lautet: Nicht zwangsläufig. Nach den Gesetzen der formalen Logik kann aus etwas Wahrem immer nur etwas Wahres geschlossen werden, wie in etwa „sind die drei Theoreme erfüllt, ist Code-Genierung möglich“. Aus etwas Falschem jedoch muss nicht zwangsläufig wieder etwas Falsches folgen. Sind die drei Theoreme nicht erfüllt, so bedeutet dies nicht unbedingt, dass Code-Generierung unmöglich ist. Unter Umständen ist sie doch möglich. Diese Umstände lassen sich sogar ziemlich präzise beschreiben. Zunächst wollen wir aber, mit den drei Theoremen im Hinterkopf, einen Blick auf die Welt der BPM Ansätze werfen. So geschehen lassen sich im Wesentlichen zwei Ansätze erkennen: Die zentralistische und die verteilte Sichtweise.

## 6.1 Die zentralistische Sichtweise: Der endliche Automat

Bei der zentralistischen Sichtweise wird der Prozess, welcher in der Realität ja nur ein abstrakter Begriff ist, als eine eigenständige Entität aufgefasst. BPM Ansätze, welche die zentralistische Sichtweise vertreten (Aris, Adonis), haben ihre Wurzeln in den Kontrollflussdiagrammen und beschreiben einen Prozess auf die gleiche Weise wie ein Kontrollflussdiagramm das Verhalten eines Programms beschreibt: Als eine Sammlung von

internen Zuständen und Zustandsübergängen. Wichtig für das Verständnis ist, dass in der ursprünglichen Konzeption diese Diagramme für die Beschreibung von endlichen Automaten verwendet wurden. Die Zustände und die Zustandsübergänge sind als interne Bestandteile eines solchen endlichen Automaten zu interpretieren. Die Folge der Zustände und Zustandsübergänge wird durch ein gewisses Ereignis ausgelöst, das so genannte „Start-Event“. In dieser Sicht der Dinge ist das Unternehmen oder die Organisationseinheit ein endlicher Automat, der nach Eintreten eines entsprechenden Ereignisses eine gewisse Folge von internen Zuständen durchläuft.[12] In Aris können mit der eEPK diese Zustände nun wiederum einzelnen, dem Prozess untergeordneten Akteuren zugewiesen sein. Doch was ist das? Handelt es sich nun um Zustände des endlichen Automaten oder um Zustände von Akteuren, welche innerhalb eines endlichen Automaten existieren? Letzteres ist wohl schon daher abzulehnen, da die Existenz von Akteuren innerhalb eines endlichen Automaten dessen Definition widerspricht. Handelt es sich also doch nicht um einen endlichen Automaten, sondern um ein verteiltes System aus verschiedenen Akteuren, die miteinander kommunizieren und so einen Prozess darstellen? Auch dies ist nicht der Fall, denn, wie wir gesehen haben, würde dies eine Formalisierung der Kommunikation zwischen diesen Akteuren erfordern. In Aris jedoch existieren weder dreistufige Relationen noch werden zweistufige Relationen entsprechend geschachtelt. Eine Kante vom Typ „führt aus“ verbindet einen Zustand (State, Function) mit einer Prozessrolle (Person, OrgUnit, etc..). Es handelt sich hierbei um eine zweistellige Relation, also ein P(x,y) wobei P für das Prädikat „führt aus“ und die Variablen x und y für den Zustand und die Prozessrolle stehen. Damit ist aber die Prozessrolle aus Sicht des Zustands nichts weiter als ein Attribut. Formal besteht kein Unterschied zu F(x,y) wobei F für das Prädikat „hat die Farbe“ sowie x für „die Funktion“ und y für „grün“ steht. Ein formaler Unterschied zwischen P(x,y) und F(x,y) würde nur dann bestehen, wenn alle P(x,y) für eine Formalisierung der Kommunikation herangezogen würden. Man könnte sich durchaus vorstellen, dass ein Event, welches zwischen zwei Zuständen steht, die mit unterschiedlichen Akteuren verbunden sich, als Nachricht zwischen diesen Akteuren zu interpretieren wäre. Dies wäre auch ohne Probleme durch einen Algorithmus abzubilden.

[12] Es ergibt sich zum einen die Frage woher dieses Ereignis kommt. Gibt es noch einen weiteren Automaten der dieses Ereignis produziert? Ferner ergibt sich die Frage wie dieser Automat in der Realität zu laufen anfängt und wie seine Ausführung zustande kommt.

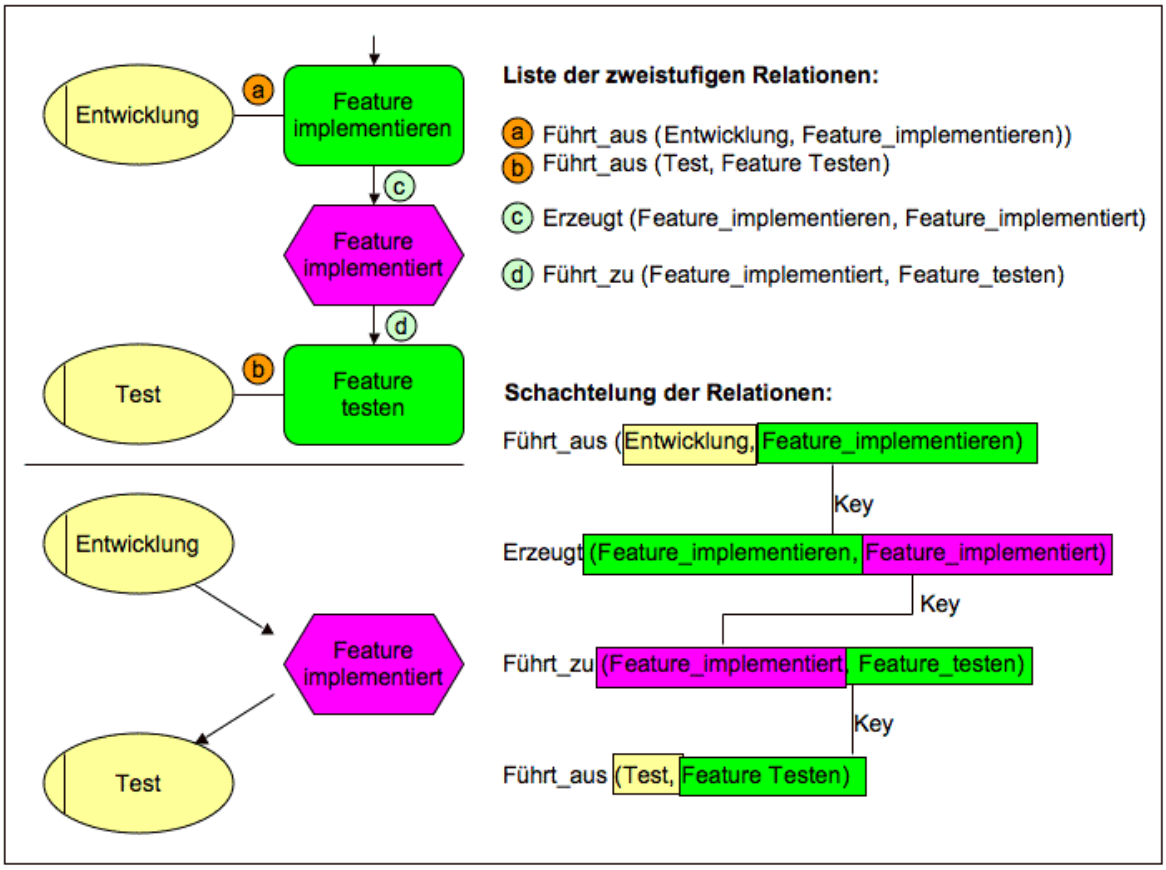



**Abbildung 4: Schachtelung mehrerer zweistufiger Relationen**

Allerdings wäre hier eine Schachtelung, also eine systematische Interpretation mehrerer zweistufiger Relationen, vorauszusetzen, die über einen Key miteinander verbunden sind. Es wird von jeder Funktion über die „führt_aus“ Relation die Rolle eingeholt und mit jener der Folgefunktion verglichen. Unterscheiden sich diese, könnte man zwei Programme ableiten, die sich eine Nachricht zuschicken, welche dann von dem empfangenden Programm entsprechend ausgewertet wird. Eine Bedingung dafür jedoch wäre aber wiederum, dass die Existenz dieser zweistufigen Relationen immer in der selben Anordnung gegeben ist (siehe Abb. 4: Schachtelung mehrerer zweistufiger Relationen). Wie wir wissen erzwingt Aris jedoch nicht einmal die Modellierung von Ereignissen zwischen Funktionen oder die Modellierung von Rollen zu den Funktionen. Es hängt also ganz vom Modellierer ab, ob ein Prozessmodell mit Hilfe derartiger Konstrukte als ein verteiltes System angesehen werden kann oder nicht. Aus der zentralisitischen Sichtweise wäre es ja nicht einmal verwerflich, wenn der Modellierer Ausdrücke wie „Rolle A sendet Nachricht XY an Rolle B“ in eine einzige Funktion packt. Für den Prozess als endlicher Automat ist dies nichts weiter als einer seiner internen Zustände. Daraus allerdings ein System aus zwei Programmen abzuleiten, wobei Programm „Rolle A“ dem Programm „Rolle B“ die „Nachricht XY“ sendet und dieses nach deren Auswertung entsprechend reagiert, wäre hier nur mit einem Interpreter für natürliche Sprache möglich. Allein gibt es so etwas leider immer noch nicht.

### 6.2 Die verteilte Sichtweise: Ein System aus mehreren endlichen Automaten

Die verteilte Sichtweise hat einen anderen Ansatz. Hier entsteht der Prozess quasi als Nebenprodukt, aus der Summe der Verhaltensweisen der einzelnen Akteure. Bei Ansätzen dieser Art (jPASS, Mega) besitzt der Prozess hier keine eigenen Zustände. Sie gehören den Akteuren. Diese wiederum sind durchaus mit endlichen Automaten zu vergleichen. In der verteilten Sichtweise ist der Prozess also ein System aus endlich vielen endlichen Automaten. Dies hat gewisse Vorteile. Zunächst wird hier per Definition die geforderte Rollensicht (F9) erreicht. Zweitens bringt ein als verteiltes System angelegter Ansatz eine Formalisierung der Kommunikation zwischen seinen Komponenten zwangsweise mit sich. Es liegt somit die Vermutung nahe, dass die verteilte Sichtweise eher dazu tendiert, die drei Theoreme zu erfüllen, da die Kommunikationsaspekte hier nicht in Form von Anotationen ausgedrückt werden können. Wenn man von einem verteilten System spricht, bedeutet dies implizit immer, dass eine streng formalisierte Kommunikation zwischen den Komponenten vorliegen muss. Außerdem kommt die verteilte Sichtweise letztlich der Realität am nächsten. Dies wird am deutlichsten, wenn wir uns mit der Praxis einer Prozessdefinition befassen. Dies aber ist Teil eines späteren Kapitels. Für den Moment genügt uns, dass die verteilte Sichtweise unsere Forderungen weit aus besser erfüllt. Fahren wir also fort mit der Argumentation, denn, wie wir sehen werden, sind immer noch nicht alle Probleme auf dem Weg zu einem optimalen Ansatz aus dem Weg geräumt.

## 7. Das modelltheoretische Dilemma

Wir haben also die verteilte Sichtweise als die unsere Forderungen optimal erfüllende identifiziert. Zumindest was einige dieser Forderungen angeht. Betrachten wir noch einmal zwei der ersten vier Forderungen aus der Ebene des Managements. F1 besagt, dass der optimale BPM Ansatz ein Höchstmaß an Flexibilität bereitstellten soll. Das Schlagwort „Flexibilität“ bedeutet hier eigentlich nichts anderes, als dass mit dem BPM Ansatz jede denkbare Situation der Realität abgebildet werden kann. Formulieren wir diese Forderung nun anders und verleihen ihr etwas mehr konkrete Handhabung für das spezifische Problem. Ein Prozessmodell soll also in der Lage sein, jede denkbare Situation auszudrücken. Im Sinne von Modelle können wir also von Mächtigkeit sprechen, und fordern:

*F13: Der optimale BPM Ansatz besitzt eine hohe Mächtigkeit*

Doch neben Flexibilität fordert das Management grundsätzlich niedrige Kosten (F4). Wir haben bereits auf die Möglichkeiten zur Entstehung von Kosten gesprochen und einmalige Lizenzgebühren nicht als Thema der Betrachtung deklariert. Wenden wir uns den anderen, ebenfalls bereits angesprochenen Kosten zu: Den Kosten für die beiden Transformationen, die eine in das Geschäftsprozessmodell und der anderen aus dem Geschäftsprozessmodell. Beide Transformationen verursachen einen gewissen Personalaufwand und stellen an das ausführende Personal gewisse Anforderungen. Qualifiziertes Personal ist teurer und man könnte die Kosten als Produkt von Personaleinsatz und Personalqualifikation auffassen. Jedes Prozessmodell verursacht zweimal Kosten mit diesem Produkt. Geringe Kosten bedeuten also in diesem Fall einerseits, dass auf umfangreiche Schulungen für die Arbeit mit dem BPM Ansatz nach Möglichkeit nicht notwendig sein sollen, und andererseits, dass die Transformationen mit möglichst geringem Aufwand zu erledigen sein sollen.

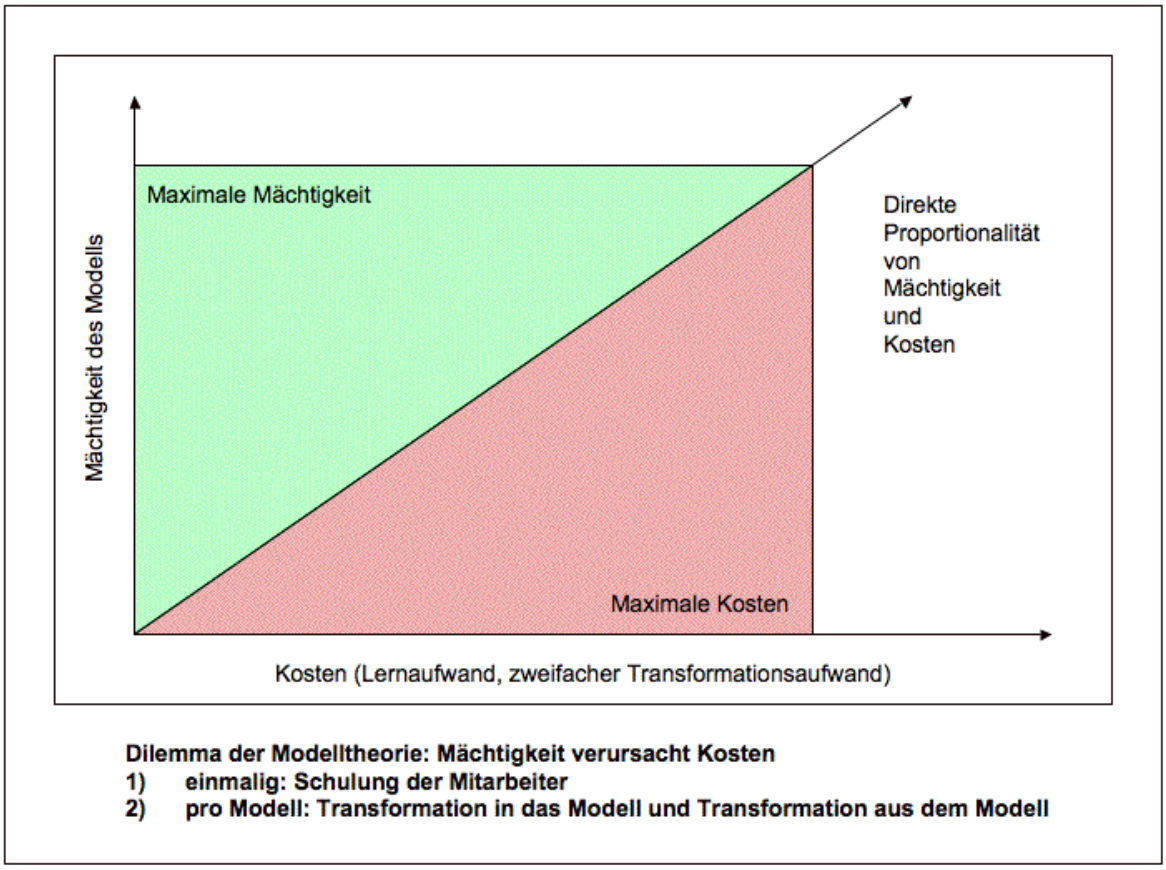



**Abbildung 5: Das modelltheoretische Dilemma**

Ein komplexer BPM Ansatz verursacht beides: Einen großen Schulungsaufwand sowie einen entsprechenden Transformationsaufwand. Aus diesem Grund können wir an dieser Stelle nur nochmals auf die Forderung F8 verweisen: Der optimale BPM Ansatz stellt die Prozesse intuitiv verständlich dar.
Wenn wir aber einerseits mit F13 eine hohe Mächtigkeit fordern und andererseits aber einen geringe Komplexität, steht dies nicht im Widerspruch? Augenscheinlich ja. Denn ein Modell wird mit zunehmender Mächtigkeit auch zunehmend komplex (siehe Abb. 5: Das modelltheoretische Dilemma). Kosten für die Arbeit mit einem Modell scheinen direkt proportional zu seiner Mächtigkeit zu steigen.

## 8. Der Ausweg: Die Standardsemantik der natürlichen Sprache

Im letzten Kapitel haben wir auf das Dilemma der Modelltheorie hingewiesen. Doch es gibt einen Ausweg. Dieser Ausweg heißt: Natürliche Sprache. Es gibt wohl kaum etwas, dass nicht in natürlicher Sprache beschrieben werden kann, und wenn doch, ist es mit Sicherheit für Geschäftsprozesse ohne Relevanz. Natürliche Sprache erfüllt somit unsere Forderung F13 (Hohe Mächtigkeit). Was aber ist mit F4 (Geringe Kosten)? Würde man natürliche Sprache extra zum Zweck von Geschäftsprozessmodellierung erlernen müssen, so hätten wir damit

einmal mehr das Modelldilemma bewiesen.[13] Mit der hohen Mächtigkeit von NL (Natural Language) geht ein entsprechend hoher Lernaufwand einher. Der Widerspruch zwischen F13 und F4 bleibt also auch hier bestehen. Allerdings hat NL in diesem Fall den Vorteil, dass bei jedem Menschen der Schulungsaufwand bereits im Kindesalter geleistet wurde. NL ist also eine Anomalie im Modelldilemma und erfüllt als somit einziges F4 wie auch F13! Aber ist NL ein formales Modell, so wie wir es in den Theoremen der Geschäftsprozessmodellierung gefordert haben? Die Antwort lautet natürlich nein. Natürliche Sprache ist unter anderem kontextabhängig und mehrdeutig und damit alles andere als ein formales Modell. Das bedeutet aber nicht, dass NL nicht für ein formales Modell verwendet werden könnte, denn auch in der NL steckt ein formaler Kern. Einfache Sätze werden mit der Standardsemantik Subjekt-Prädikat-Objekt gebildet. Es wird hier bewußt der Begriff Standardsatzsemantik gewählt. Wir wollen keine Aussage treffen über die Reihenfolge in der diese Satzteile in einem Satz angeordnet werden. Dies ist in der Grammatik der jeweiligen Sprache festgelegt. In diesem Artikel benutzen wir die gewohnte grammatikalische Reihenfolge Subjekt, Prädikat, Objekt. Auf derartige Sätze kann ein überschaubarer Formeltest angewandt werden, der entscheidet, ob ein Satz entsprechend dieser Standardsatzsemantik vollständig ist. Sätze auf dieser einfachen Ebene sind, wie wir noch sehen werden, durchaus mit einem formalen Modell vereinbar. Baut man zudem ein Begriffsuniversum auf, in dem jedem Begriff eine eindeutige Bedeutung zukommt, eliminiert man Mehrdeutigkeit und Kontextabhängigkeit. Um die NL als formales Modell zu verwenden, sind also die folgenden beiden Schritte notwendig:

1) Definition eines Begriffsuniversums
2) Reduzierung der NL auf Sätze in Standardsemantik

Wir werden in einem späteren Kapitel darauf eingehen, wie die beiden Schritte tatsächlich durchgeführt werden können und dies an einem Beispiel durchexerzieren. Zunächst aber prüfen wir, in wie weit in der natürlichen Sprache unsere Anforderungen erfüllt sind: Wir haben die Forderungen der ersten Kapitel (F1-F11) in der zentralen Forderung F12 zusammengefasst und aus dieser letztlich die drei BPM Theoreme entwickelt. Außerdem haben wir einen Widerspruch zwischen der Forderung an einen Modellierungsansatz nach hoher Mächtigkeit und niedriger Kostenverursachung identifiziert. Die Standardsemantik der NL löst diesen Widerspruch auf. Erfüllt sie jedoch die drei Theoreme? Wenn ja, wären wir mit unserer Suche nach dem optimalen Modellierungsansatz am Ziel. Zunächst wäre im Sinne des ersten BPM Theorems die Bedingung zu erfüllen, dass jeder Satz in NL Standardsemantik (SPO) eine Formel sein muss, d.h. mit den Bildungsgesetzen einer formalen Theorie gebildet werden kann. Diese Theorie ist in dem Fall die deutsche Grammatik, deren Regeln wir als unsere Axiome betrachten:

1) Ein Aussagesatz besteht aus Subjekt, Prädikat und mindestens einem Objekt
2) An erster Stelle steht das Subjekt, der Ausgangspunkt der Handlung
3) An zweiter Stelle steht das Prädikat
4) Nun folgt mindestens ein Objekt (mit entspr. Präpositionen)[14]

Nehmen wir den Satz in Standardsemantik: „Mitarbeiter füllt Reisekostenantrag aus“. Dabei handelt es ich um einen Ausdruck der Form T(x,y,z), wobei T als Prädikat

[13] Wir wissen alle noch aus dem Französisch- oder Lateinunterricht, wie viel Aufwand es bedeutet, auch nur die armseligsten Konstrukte in einer Fremdsprache korrekt zu bilden.

[14] Siehe [Dud95]: Der Duden: Die Grammatik: Satzbaupläne im Einzelnen, S. 655 ff.

(Tätigkeitsprädikat) im Sinne der Prädikatenlogik angibt dass es sich um einen volltändigen Satz handelt der aus folgenden Bestandteilen besteht:

*x:* *Subjekt: Mitarbeiter*
*y:* *Prädikat: ausfüllen*
*z:* *Objekt 1: Reisekostenantrag*

Der Satz erfüllt alle Axiome der NL in Standardsemantik: Er besitzt ein Subjekt, ein Prädikat und ein Objekt (A1) wobei an erster Stelle das Subjekt (A2), an zweiter Stelle das Prädikat (A3) und an letzter Stelle mindestens ein Objekt (A4) steht. Damit ist der Satz eine Formel und die NL in Standardsemantik erfüllt unser erstes BPM Theorem. Wie sieht es mit der Formalisierung des Kommunikationsaspektes aus? Nehmen wir wieder den Satz „Rolle A sendet Nachricht XY an Rolle B“. Wie wir bereits gesagt haben, handelt es sich hier um einen Ausdruck der Form S(x,y,z) (Sendeprädikat)wobei

*x:* *Subjekt: Rolle A*
*y:* *Objekt 1: Rolle B*
*z:* *Objekt 2: Nachricht XY*

Indieser Relation ist das grmmatikalische Prädikat senden identisch mit dem prädikatenlogischen Prädikat (Prädikator). Analog ist auch der Prädikator Empfangen aufgebaut. Der Aussage „Rolle A empfängt Nachricht XY von Rolle B“ entspricht dem dem Prädikator E(x,y,z) (Empfangen).

*x:* *Subjekt: Rolle A*
*y:* *Objekt 1: Rolle B*
*z:* *Objekt 2: Nachricht XY*

Ein derartiger Satz ist nach den Axiomen ebenso eine Formel und in der Lage, Kommunikation zu formalisieren, da er eine dreistufige Relation aufbaut, nämlich ein Prädikat mit Sender, Empfänger und Nachricht. Somit erfüllt die Standardsemantik der NL die drei BPM Theoreme. Damit wären wir also mit unserer Suche nach einem optimalen BPM Ansatz am Ende. Wie wir im nächsten Kapitel sehen werden, verfügen die gängigen Modelle in der Informationstechnologie bereits über Prädikate und Objekte. Jetzt, im Zeitalter der Geschäftsprozesse, im Zeitalter von SOA, Webservices und automatisierten Akteuren, lässt sich die Einführung des Subjekts in die IT-Modellierung nicht mehr hinauszögern. Die Zeit ist reif für eine Programmierung in ganzen Sätzen, für den Modellierungsansatz Subjekt – Prädikat – Objekt.

## 9. SPO und die Evolution der IT-Modelle

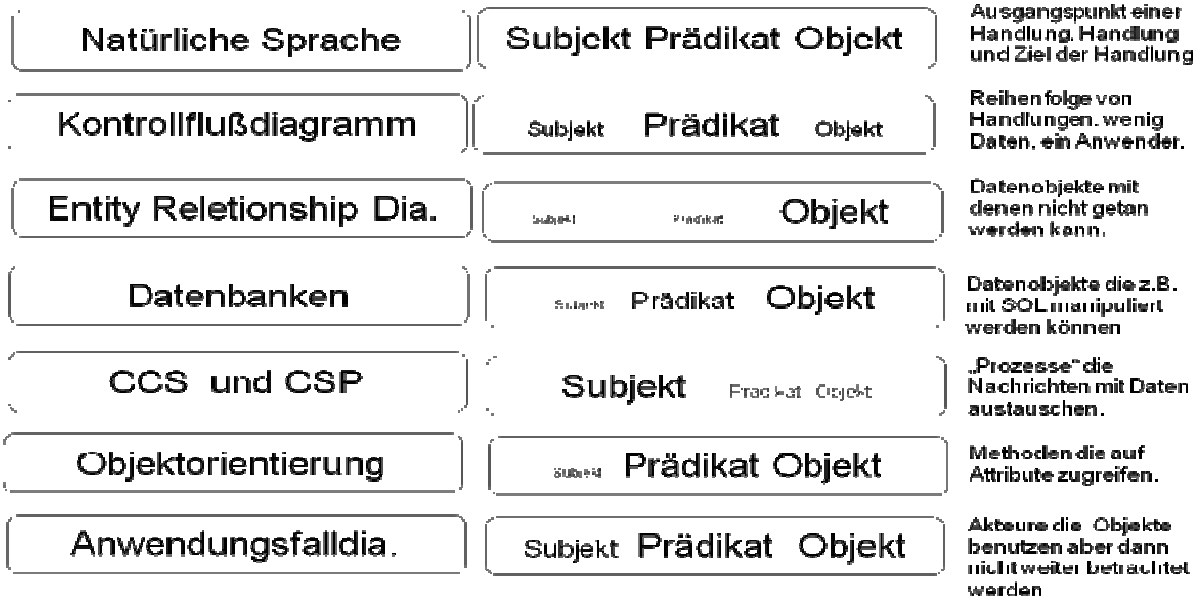
Nach dem wir im vorigen Kapitel aus einer Reihe von Forderungen und Theoremen die Standardsemantik SPO der NL als die grundlegende Theorie für optimale BPM Ansätze identifiziert haben, können wir dieses Ergebnis auch von einer ganz anderen Seite erreichen. Wenn wir die Evolution der Modellierungsansätze für IT-Systeme betrachten, werden wir ebenfalls feststellen, dass die Einführung des Subjekts auch aus dieser Sicht der nächste zwingend logische Schritt ist. Modellbildung bedeutet immer für einen bestimmten Zweck einen Ausschnitt der Wirklichkeit betrachten und die für den Zweck wesentlichen Aspekte in

einem Modell erfassen. (siehe auch Wikioedia Stichwort Modell).[15] In der Informatik dienen Modell dazu eine Wirklichkeit zu betrachten die dann in einem Programm nachgebildet wird und durch die Nutzung des Programms durch Menschen (Anwender, Nutzer) wieder Teil der Wirklichkeit wird. Diese neue Wirklichkeit kann weiter den Wünschen von in der betrachteten Wirklichkeit lebenden Menschen angepasst werden. Je nach den betrachteten Aspekten der Wirklichkeit werden verschiedene Modellbildungskonzepte verwendet. Ausgehend von der Standardsatzsemantik der natürlichen Sprache als ein Unversalkonzept der Modellbildung lassen sich die in der Informatik die jeweiligen Modellbildungsmethoden bewerten inwieweit sie die einzelnen Aspekte der Standardsatzsemantik Subjekt, Prädikat, Objekt unterstützen.[16] [17]

| Methode | Wirklichkeit |
|---|---|
| Kontrollflußdiagramm | Ein bestimmter Anwender gibt eine begrenzte Anzahl Daten an einen Rechner und erhält eine Lösung z.B. Numerische Lösungen einer Differentialgleichung |
| Entity Reletionship Diagramm | Die betrachteten Datenobjekte sind sehr komplex. Die Methoden zur Bearbeitung sind nur erzeugen, lesen und löschen und werden deshalb nicht betrachtet |
| Datenbanken | Die betrachteten Datenobjekte sind komplex und kommen in großen Mengen vor. Zur gleichartigen Bearbeitung dieser Datenmengen gibt es entsprechende Abfrage- und Bearbeitungssprachen z.B. SQL |
| CCS und CSP | Es gibt nur aktive Elemente die parallel laufen. Diese Elemente tauschen Nachrichten mit Daten auswie es in Realzeitanwendungen notwendig ist |
| Objektorientierung | Die Wirklichkeit besteht aus komplexen Daten die mit komplexen Operationen beliebig bearbeitet werden sollen. Der Anwender der Methoden wird nicht betrachtet. |
| Anwendungsfalldiagramm | Akteure benutzen Objekte. Eine direkte Interaktion der Akteure wird nicht betrachtet |

Die obige Tabelle gibt einen Überblick über die betrachteten Wirklichkeiten im Laufe der Entwicklung der Softwareentwicklung und welche Methoden dabei zum Einsatz kamen. In der Algorithmik liegt der Schwerpunkt auf dem Prädikat, in der Informationsverarbeitung auf dem Objekt. Da das Geschäftsprozessmanagement ein umfassendes Konzept ist benötigt man die Subjekte als Ausgangspunkt von Handlungen, das Prädikat als Handlung und das Objekt als Ziel der Handlung.
Die folgende Tabelle stellt die jeweiligen Methoden der Standardsatzsemantik der natürlichen Sprache gegenüber. Sie zeigt welche Aspekte der natürlichen Sprachen in dem jeweiligen Modellierungskonzept berücksichtigt sind.

[15] Siehe Wikipedia unter dem Stichwort Modell
[16] Albert Fleischmann, Subjekt, Prädikat und Objekt in der Grammatik der Software, siehe http://www.jcom1.com/cms/downloads.html
[17] Albert Fleischmann, Christian Stary, Subject-Driven Specification: Bridging the Gap between Natural Language and Formal Semantics for Automated Code Generation and Service-Oriented Architecting, Publikation in Vorbereitung




## 10. Modellierung mit SPO (die erste Transformation)

Nach dem wir hinreichend die Einführung des Subjektes in die Modellierung der Informationstechnologie begründet haben, wenden wir uns nun der praktischen Arbeit zu. Wie wir bereits angedeutet haben, steckt in der NL ein formaler Kern, den wir uns für die Verwendung der NL als Modellierungsinstrument bei Geschäftsprozessen zu Nutze machen wollen: Die Standardsemantik Subjekt-Prädikat-Objekt. Ebenfalls haben wir eingangs behauptet, jede Spezifikation von Geschäftsprozessen oder IT-Systemen liegt zunächst in NL vor. Ferner sagten wir, dass es möglich sei, jeden beliebigen Text durch Definition eines Begriffsuniversums und durch Reduktion der Sätze auf Standardsemantik in das formale System SPO zu überführen, ohne dabei Informationsverluste in kauf nehmen zu müssen. Somit lässt sich jeder beliebig komplexe Text auf ein formales Modell reduzieren, ohne dass der Charakter und die Vorteile der NL verloren gehen. Wir demonstrieren dies anhand eines Satzes aus dem Repertoire einer der unübertroffenen Meister komplexer und geschwungener Sätze deutscher Sprache, einem Satz des Heinrich von Kleist:

*„In M., einer bedeutenden Stadt im oberen Italien, ließ die verwitwete Marquise von O., eine Dame von vortrefflichem Ruf, und Mutter von mehreren, wohlerzogenen Kindern, durch die Zeitungen bekannt machen: dass sie, ohne ihr Wissen, in andere Umstände gekommen sei, dass der Vater zu dem Kinde, das sie gebären würde, sich melden solle; und dass sie, aus Familienrücksichten, entschlossen wäre, ihn zu heiraten"*[18].

Soweit der erste Satz der Erzählung „Die Marquise von O." des Heinrich von Kleist. Ein Genuss für den Liebhaber sprachlich gehobener Abendliteratur, aber lässt er sich auch in das formale Modell Subjekt-Prädikat-Objekt überführen und schließlich durch ein Prozessmodell ausdrücken? Dazu zunächst die Definition eines Begriffsuniversums.

[18] Heinrich von Kleist: Die Marquise von O. in: Sämtliche Werke und Briefe, hrsg. Helmut Sembdner, Deutscher Taschenbuchverlag, München, 2001

## 10.1 Definition eines Begriffsuniversums

Wie wir bereits erwähnt haben, benötigen wir ein fixes Begriffsuniversum, um eine Eindeutigkeit und Kontextunabhängigkeit zu erreichen. Dies wird von der IEEE bei der Spezifikation für IT-Systeme ohnehin gefordert und ist Teil der von Christine Rupp aufgelisteten Qualitätskriterien für Anforderungen [Rup07]. Es handelt sich einfach somit um einen allgemein notwendigen Schritt auf dem Weg von NL zu einem formalen System. Wir extrahieren Subjekte, Prädikate und Objekte (Subjekte sind der Ausgangspunkt von Handlungen, das Prädikat die Handlung und das Objekt das Ziel der Handlung):

Subjekte: Die Marquise von O.
Die Zeitungen von M. (im oberen Italien)

Prädikate: etwas tun lassen = beauftragen
etwas bekannt machen = drucken
verwitwet sein
eine Dame von vortrefflichem Ruf sein
Mutter mehrerer wohlerzogener Kinder sein

Objekte: Die Bekanntmachung

Wie ersichtlich haben wir nur ein reines Objekt identifiziert, und noch dazu eines, das derart im Text gar nicht vorkommt, jedoch haben wir dabei nichts anderes getan, als den Text nach dem Doppelpunkt, also die tatsächliche Bekanntmachung, als solche zu deklarieren. Als nächsten Schritt zerlegen wir den Satz in einfache Aussagesätze der Form SPO.

## 10.2 Reduzierung der NL auf Sätze in Standardsemantik

Hierbei bedienen wir uns nun ausschließlich aus unserem Begriffsuniversum von Subjekten, Prädikaten und Objekten. Durch Aufnahme von Begriffen in das Begriffsuniversums treffen wir die Entscheidung, ob Information aus dem Text für einen Geschäftsprozess formalisiert werden soll, oder ob sie nur kommentierenden Charakter besitzt und uns in Prosaform genügt.

*Die Marquise von O. ist verwitwet.*
*Die Marquise von O. ist eine Dame von vortrefflichem Ruf.*
*Die Marquise von O. ist Mutter mehrerer, wohlerzogener Kinder.*
*Die Marquise von O. beauftragt die Zeitungen von M. mit einer Bekanntmachung*
*Die Zeitungen von M. drucken die Bekanntmachung*

Nun haben wir also den komplexen Satz des von Kleist in eine Reihe einfacher Aussagesätze umgeformt, welche nur auf Elemente unseres Begriffsuniversums zurückgreifen. Wir erkennen dabei auch drei im Wesen voneinander unterscheidbare Satzkategorien:

- Zunächst sind da die drei Sätze mit Kopula (ist). Diese haben die Funktion, einem Objekt (bzw. dem grammatikalischen Subjekt) Attribute zuzuweisen.
- Als nächstes haben wir einen Satz, bei dem eines der Akkusativobjekte in der Liste der Subjekte auftaucht (die Zeitungen von M.), was bedeutet, dass es sich hier um

einen Satz handelt, bei dem zwischen zwei Akteuren ein (Nachrichten-) Austausch stattfindet.

- Der letzte Satz und die dritte Kategorie ist von funktionalem Charakter. Hier taucht nur ein Akkusativobjekt auf, welches nicht in der Liste der Subjekte bzw. Akteure vorkommt, also ein reines Objekt oder ein passiver Gegenstand ist (die Bekanntmachung). [19]

Diese drei Satzkategorien reichen prinzipiell völlig aus, um jeden beliebigen Sachverhalt eines Geschäftsprozesses auszudrücken. Das einzige, das noch fehlt , ist die Definition von Regeln, nach dem Motto: Wenn ein Auftrag mit einer Bekanntmachung eingeht, dann wird die Bekanntmachung gedruckt. Darauf werden wir im nächsten Kapitel eingehen, in dem wir zunächst die Grundzüge eines optimalen BPM Ansatz vorstellen und anschließend mit ihm den obigen Sachverhalt modellieren werden. Festzuhalten ist, dass die Umformungen, die bis hierher getan wurden, von jedem Menschen ohne spezielle Qualifikation durchgeführt werden können.

## 11. Modelle eines optimalen BPM Ansatzes: jPASS

jPASS ist ein BPM Ansatz, welcher als formales und verteiltes System alle Forderungen sowie die drei BPM Theoreme erfüllt und aufgrund der Anlehnung an die Standardsemantik der NL mit einem einfachem Konzept das Modelldilemma auflöst. Es ist nicht Ziel dieses Beitrags, eine detaillierte Werkzeugbeschreibung abzugeben.[20] In diesem Aufsatz geht es vielmehr um den Kern der Sache, also um die Frage, in wie fern ein Konzept geeigneter ist als das andere. In diesem Sinne werden wir auch nur das wesentliche Konzept von jPASS vorstellen. Ein Prozess in jPASS besteh aus zwei Ebenen:

- Die Kommunikationsebene, welche illustriert, welches Subjekte untereinander welche Nachrichten austauschen.
- Die Verhaltensebene, in welcher das interne Verhalten der einzelnen Subjekte modelliert wird.

Diese beiden Ebenen stehen natürlich in streng formaler Beziehung zueinander. Die Modellierung beginnt auf der Kommunikationsebene. Zunächst werden dort zwei Subjekte angelegt (die Marquise von O. und die Zeitungen von M.) und die Nachrichten (in unserem Fall: Auftrag mit Bekanntmachung) modelliert, welche die Subjekte miteinander austauschen (diese bilden eine Liste der überhaupt verfügbaren Nachrichten). Anschließend kann für jedes Subjekt ein internes Verhalten festgelegt, d.h. ein endlicher Automat definiert werden. Dabei kennt jPASS drei Zustände:

Sendezustand: Hier wartet das Subjekt auf eine Nachricht eines anderen Subjekts. Trifft die Nachricht ein, kann ein Folgezustand definiert werden.

[19] Die Unterscheidung zwischen den letzten beiden Kategorien (die wir auch schon zu Beginn bereits angedeutet haben) entspricht dem Unterschied zwischen Delegation oder Ausführung eines Prozessschrittes, in den Wirtschaftswissenschaften auch bekannt unter dem Schlagwort „make or buy“.

[20] Für weitere Informationen zum Tool jPASS und seiner Add-Ons bitte die Internetseite der jCOM1 AG konsultieren: www.jcom1.com

Empfangszustand: Hier sendet das Subjekt eine Nachricht an ein anderes Subjekt.

Interne Funktion: Hier führt das Subjekt eine interne Funktion aus.

Das interne Verhalten der Marquise von O. besteht in diesem Fall (d.h. unter alleiniger Betrachtung des ersten Satzes der Erzählung) nur aus einem Zustand, einem Sendezustand: Sie sendet einen Auftrag mit einer Bekanntmachung an die Zeitungen von M. Diese wiederum (jPASS unterstütz für diesen multiplen Fall sog. Subjekt-Arrays) besitzen zwei Zustände: Einen Empfangszustand, in dem auf einen Auftragseingang etwa in Form einer Bekanntmachung gewartet wird, und einen zweiten Zustand, eine interne Funktion, welche die Bekanntmachung eben bekannt macht bzw. druckt und publiziert. Kommen wir nun zu den Regeln. Diese werden in jPASS schlicht über die Kanten ausgedrückt. In diesem Fall gibt es nur eine davon, die wir bereits erwähnten: Wenn ein Auftrag mit einer Bekanntmachung eingeht, dann wird die Bekanntmachung gedruckt (siehe Abb. 6: Die Marquise von O. in jPASS). Zieht man eine Kante von einem Empfangszustand zu einem Folgezustand, so muss man für die Kante angeben, für welche Nachricht (ein Tripel aus Sender, Inhalt und Empfänger) sie gilt. Somit kann der Kontrollfluss innerhalb der Subjekte gesteuert werden, nach dem Motto: Empfange dies von jenem und tue das oder empfange etwas anderes von diesem und tue jenes. Interne Funktionen können ebenfalls ausgewertet werden und auf eine ähnliche Weise den Kontrollfluss bestimmen.
Nach dem wir nun die Marquise von O. auf oberster Ebene soweit modelliert haben, können wir uns detailliertere Gedanken machen. Eine Zeitung ist ja eigentlich kein Ein-Mann-Betrieb sondern ein Unternehmen, das sich aus mehreren Organisationseinheiten zusammensetzt und das Drucken einer Bekanntmachung ist keine einzelne Funktion, sondern vielmehr ein Prozess. jPASS löst derartiges mit der Eleganz der natürlichen Sprache. Ein Subjekt kann in jPASS entweder ein Akteur im klassischen Sinne sein oder aber wieder ein Prozess mit neuen Akteuren, die auf der oberen Ebene gar nicht vorkommen. Sie sind interne Akteure der Zeitungen von M. und auf dieser Ebene auch gar nicht relevant, den die Marquise von O. wird sich kaum von den eingangs erwähnten Kunden unterscheiden und sich ebenso wenig für die interne Struktur eines Zeitungsverlegers interessieren. Sie hat, wie aus dem ersten Satz der Erzählung zu entnehmen, andere Sorgen.

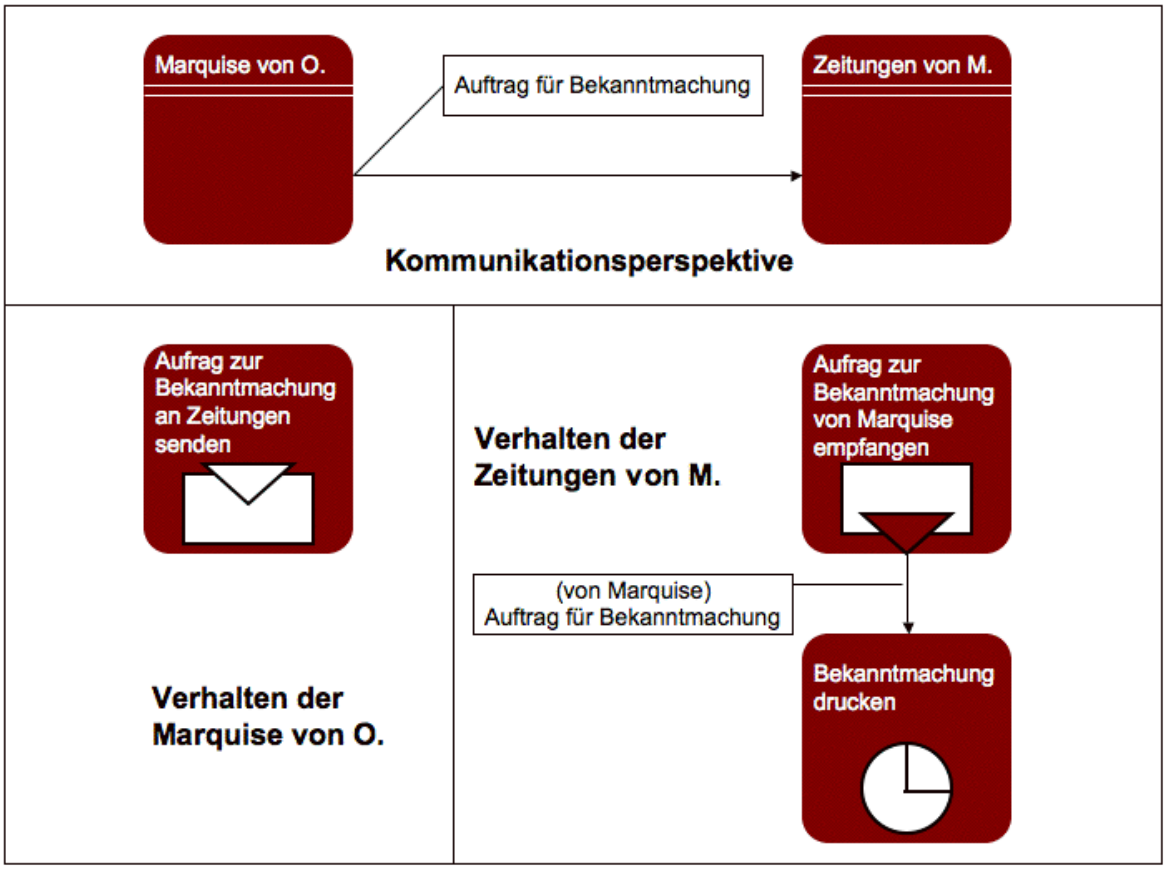



**Abbildung 6: Die Marquise von O. in jPASS**

Für die Zeitungen jedoch sind interne Prozesse sehr wohl von Relevanz und sie täten gut daran, die Schritte für eine Bekanntmachung zu formalisieren und die Verantwortlichkeiten intern zuzuweisen.
Der Punkt jedoch ist der Folgende: Wo andere Modellierungsansätze fixe Ebenen einführen (Wertschöpfungsebene, etc..) die wiederum mit neuen Symbolen und Modellierungsregeln (oder der traurigen Abstinenz solcher) einhergehen, bleibt die NL in ihrer Flexibilität völlig uneingeschränkt, ohne dabei auch nur ein einziges neues Symbol oder eine neue Regel einführen zu müssen. SPO lässt sich auf die Marquise von O. und die Zeitungen von M. genauso anwenden, wie zeitungsintern auf den Redakteur, den Drucker und deren Maschinen. Es gibt ein System für alle Ebenen, und diese werden bereits bei der Definition des Begriffsuniversums angelegt. Die Subjekte und Prädikate lassen sich hier schon anhand der Hierarchien eines Prozesses oder einer Organisationsstruktur unterteilen. Dabei sollte man wiederum im Kopf behalten, dass es sich hier ebenso wenig um eine Leistung handelt, für die eine spezielle Qualifikation von Nöten wäre. Wir setzen eine konsequente Unterscheidung zwischen solchen Ebenen in jeder Diskussion von unserem Gegenüber sogar in Echtzeit implizit voraus. Somit haben wir aufgezeigt, wie die Standardsemantik der NL als formales Modell für Geschäftsprozesse tatsächlich das modelltheoretische Dilemma durchbricht. Als nächstes wenden wir uns wieder einem anderen Aspekt zu, wenn auch unter dem selben, die Effizienz und Kosteneinsparung betreffenden Blickwinkel: Der zweiten Transformation, also der Transformation aus dem Modell in Richtung Implementierung.

# 11. Modellierung mit SPO: Die zweite Transformation

Wie in Abb. 2 dargestellt, erfährt jeder Geschäftsprozess eine zweifache Transformation. Zum einen aus der NL in ein Modell, zum zweiten aus dem Modell in Richtung Implementierung. Beide Transformationen haben wir als Verursacher von Kosten identifiziert, weshalb wir letztendlich für beide eine größtmögliche Effizienz forderten. Die Ziele für die zweite Transformation haben wir bereits genannt:

- Handlungsanweisungen für humane Akteure (NL)
- Spezifikationen (Use Cases) für die Erstellung bzw. Anpassung von IT-Systemen
- Einen Prozessprototyp

## 11.1 Handlungsanweisungen für humane Akteure

Damit der Prozess letztendlich so gelebt werden kann, wie in einem Modell spezifiziert wurde, müssen die Menschen, die ihn ausführen, das Modell in irgend einer Weise vermittelt bekommen. Dem voran geht aber noch eine inhaltliche Qualitätssicherung, welche allerdings genau die selben Ansprüche stellt. Da bei einem iterativen Vorgehen mehrere Runden gedreht werden, ist gerade dabei eine effiziente Vermittlung von Vorteil. Die Zielsprache der Handlungsanweisungen ist natürlich die NL. Liegt ein BPM Ansatz vor, der, wie in F9 gefordert, die einzelnen Rollensichten darstellen kann und wie im Fall von SPO bereits auf der NL basiert, ist dies natürlich ein leichtes und kann automatisiert geschehen, also ohne den geringsten Aufwand. Wir können aus dem obigen Prozessmodell folgendes ableiten:

*Marquise von O: Senden: Auftrag zur Bekanntmachung*

*Zeitungen von M: Empfangen: Auftragseingang Bekanntmachung*
*Zeitungen von M: Wenn Auftragseingang Bekanntmachung*
*Bekanntmachung drucken*

Und so weiter und so fort. Der chronologische Aspekt, der hier möglicherweise zu entnehmen ist, kann und soll dabei keine Rolle spielen. Ein verteilter Ansatz mit einem Message-System ist nicht an eine chronologische Abfolge gebunden. Dies bleibt ein künstliches Konstrukt der zentralisitischen Sichtweise. Im verteilten Ansatz interessiert nur die logische Abfolge der Schritte, wobei die Subjekte sich ganz automatisch optimal synchronisieren, so lange bei der Modellierung die einfache Regel angewandt wurde, jeden Prozessschritt so früh als möglich ausführen zu lassen.
Doch zurück zu unserem Output. Die oben aus dem SPO Modell generierte deutsche Sprache entspricht vielleicht nicht gerade dem Niveau eines Heinrich von Kleist, aber sie reicht alle mal aus, um von einem Fachanwender verlangen zu können, zu überprüfen in wie weit sein Verhalten in einem Prozess korrekt erfasst wurde. Ein inkrementelles Vorgehen und eine inhaltliche Qualitätssicherung kann so ohne einen ständigen, aufwendigen da manuellen Übersetzungsprozess gewährleistet werden.

## 11.2 Ableitung von IT Spezifikationen

Das Modell der Wahl für die Spezifikation von IT-Systemen ist UML, und dessen Einstiegsmodell sind die Use Cases. Aus Sicht eines Systemverwalters ist in Bezug auf Prozesse letztlich nur eines relevant: Was gibt der Prozess meinem System und was will er von ihm. Taucht ein Anwendungssystem im Prozess auf, so tut es dies in Form eines Akteurs oder, im Falle von SPO eines Subjekts. Die internen Funktionen dieses Subjekts entsprechen den Anwendungsfällen des Prozesses auf das System. Da es sich hier immer um zweistellige Relationen handelt, ließen diese sich prinzipiell aus allen BPM Ansätzen, zentralistischen wie verteilten, ableiten.

### 11.3 Generierung eines Prototyps

Wie bereits eingehend dargestellt ist Code-Generierung mehr als nur ein Schlagwort. Prozesse als verteiltes System ausführen zu können ist der Königsweg bei der Vermittlung von Prozessen, sei es als Möglichkeit zur internen Qualitätssicherung, zu Schulungszwecken oder letztlich gar zur automatisierten Abwicklung des Prozesses selbst. Für die Generierung eines ausführbaren Prozessprototyps ist letztlich jeder verteilte Ansatz prinzipiell in der Lage, zumindest, so lange er die drei Theoreme berücksichtigt. Einen wesentlichen Vorteil bringt SPO demgegenüber nicht. jPASS ist somit schon auf Grund seines verteilten Ansatzes und aufgrund der drei erfüllten Theoreme in der Lage, binnen Minuten einen Prozess von der Modellierung zur Ausführung zu bringen.

## 12. Zusammenfassung

Beginnend auf der Ebene des Managements haben wir zunächst vier Forderungen an einen BPM Ansatz aufgestellt. Diese waren:

*F1: Der optimale BPM Ansatz muss ein Höchstmaß an Flexibilität erlauben.*
*F2: Der optimale BPM Ansatz muss schnell zu einer Implementierung führen.*
*F3: Der optimale BPM Ansatz muss zu effizienten Prozessen führen*
*F4: Der optimale BPM Ansatz verursacht geringe Kosten*

Damit aus Sicht der IT-Abteilung ein iteratives Vorgehen effizient (F2, F4) unterstütz werden kann und eine teure Fehlerbehebung in den späten Projektphasen erspart bleibt (F4), forderten wir:

*F5: Der optimale BPM Ansatz unterstützt eine inhaltliche Qualitätssicherung*

Außerdem ist es der IT ein anliegen, dass sich die Fachanwender mit ihrem Prozesswissen optimal einbringen können und so für die besten und effizientesten Prozesse sorgen können (F3) forderten wir:

*F6: Der optimale BPM Ansatz liefert einen ausführbaren Prozessprototyp*

Um eine möglichst schnelle Implementierung zu ermöglichen (F2) und der IT so weit es geht entgegen zu kommen, forderten wir:

*F7: Der optimale BPM Ansatz liefert Anwendungsfälle für IT-Systeme*

Um generell Transformationskosten zu sparen (F4) und die Abneigung der Fachanwender gegen eine Arbeit mit den Prozessmodellen zu minimieren (F3), forderten wir:

*F8: Der optimale BPM Ansatz stellt die Prozesse intuitiv verständlich dar*

Ferner sagten wir, die für die Fachanwender optimale Weise, einen Prozess zu beurteilen, ist aus der Sicht ihrer eigenen Rolle. Somit ließ sich aus (F5) ableiten:

*F9: Der optimale BPM Ansatz kann die Sicht einzelner Rollen darstellen*

Anschließend fassten wir die Forderungen (F5) und (F6) sowie (F8) zusammen und leiteten daraus ab:

*F10: Der optimale BPM Ansatz liefert automatisch ausführbaren Code*

In Kombination mit der Notwendigkeit, die einzelne Rollensicht optimaler Weise in der ausführbaren Simulation des Prozesses beizubehalten, fügten wir hinzu:

*F11: Der ausführbare Code spiegelt das Verhalten der einzelnen Rollen wieder*

Da die Ermöglichung einzelner Rollensichten zwangsweise zu einem verteilten System führt, fassten wir die letzen beiden Forderungen (F10) und (F11) zusammen und leiteten daraus die zentrale Forderung F12 ab:

*F12: Ein BPM Ansatz ist optimal, wenn sich aus einem Prozessmodell dieses Ansatzes vollautomatisch ein ausführbares System aus Programmen generieren lässt, von denen jedes Programm einen Prozessbeteiligten repräsentiert, dessen Verhalten durch eben dieses Programm (im Sinne des Prozessmodells) vollständig und abgeschlossen definiert ist und welches System bei Ausführung den modellierten Prozess widerspiegelt.*

Im Laufe der weiteren Argumentation stellten wir unter Berücksichtigung von vier Prämissen drei Theoreme auf, welche notwendig sind, um diese zentrale Forderung zu erfüllen. Diese sind in Kurform:

| | |
|---|---|
| T1: | Code-Generierung benötigt ein formale Theorie als BPM Ansatz. |
| T2: | Diese Theorie muss den Kommunikationsaspekt formalisieren |
| T3: | Somit muss diese Theorie vom Wesen her dreistufig sein |

Anschließend haben wir die Konsequenzen aus den Theoremen gezogen haben und die Welt der BPM Ansätze in die zentralistische und die verteilte Sichtweise unterteilt haben, wovon letztere aus besagten Gründen vorzuziehen sind.

Weiterhin haben wir aus der Forderung F1 (Flexibilität) die Forderung F13 abgeleitet:

*F13: Der optimale BPM Ansatz besitzt eine hohe Mächtigkeit*

Dies führte uns aufgrund der direkten Proportionalität von Mächtigkeit und Kosten eines Modells zu einem Widerspruch mit F4 (Geringe Kosten). Die von uns angebotene Lösung zu diesem Dilemma ist die Verwendung der Standardsemantik (SPO) der natürlichen Sprache als grundlegende, formale Theorie eines BPM Ansatzes.

Zuletzt stellten wir den BPM Ansatz jPASS vor und demonstrierten damit eine optimale weil effiziente Transformation in ein BPM Modell als auch eine ebensolche Transformation in Richtung Implementierung.

**Literatur**:


[BecXX] K. Beck, [DADDY FRAGEN]

[Cha07] CHAOS Reports [DADDY FRAGEN]

[Dud95] P. Eisenberg, H. Gelhaus, H. Wellmann, H. Henne, H. Sitta: Der Duden: Die Grammatik, Bibliographisches Institut & F. A. Brockhaus, Mannheim, 1995

[Gart06] *Gartner, Dataquest Inside:* BPMS Software , Market Size and Forecast, Worldwide 2006-2011

[HaCh93] Hammer, Champy [DADDY FRAGEN]

[Hol99] *A. Holl:* Empirische Wirtschaftsinformatik und evolutionäre Erkenntnistheorie (Bde. Becker, Jörg et al. (ed.)).: Gabler, Wiesbaden 1999

[Hom02] *K. Homann:* Vorteile und Anreize, hrsg. Von Christoph Lütge, Tübingen 2002

[HoyXX] P. Hoyningen-Huene, Formale Logik, Eine philosophische Einführung, Reclams Unversalbibliothek 9692 W. Detel, Grundkurs Philosophie, Band 1 Logik, Reclams Universalbibliothek 18468 [ALBERT FRAGEN]

[Picot03] *A. Picot, R. Reichwald, R. Wigand:* Die grenzenlose Unternehmung, Gabler, Wiesbaden, 2003

[Rob99] *S. Robertson, J. Robertson:* Mastering the Requirement Process, ACM Press, New York/Oxford, 1999

[Rup07] *C. Rupp & die SOPHISTen*: Requirementsengineering und Management, Carl Hanser Verlag, München, 2007

[Sche07] *A.-W. Scheer,* [ALBERT FRAGEN]

[ScKr04] *B. Schwarzer, H. Krcmar:* Wirtschaftsinformatik, Schäffer / Poeschel Verlag, Stuttgart, 2004

[Sne07] *H.M. Sneed, M. Baumgartner, R. Seidl*: Der Systemtest, Carl Hanser Verlag, München, 2007

[ToAc03] *J.P. Thommen, A.-C. Achleitner*: Allgemeine Betriebswirtschaftslehre, Gabler / GWV Fachverlage, Wiesbaden, 2003