
# *Gegenüberstellung verschiedener Paradigmen zur Darstellung von Prozesseigenschaften unter Berücksichtigung von Zeit und Daten*

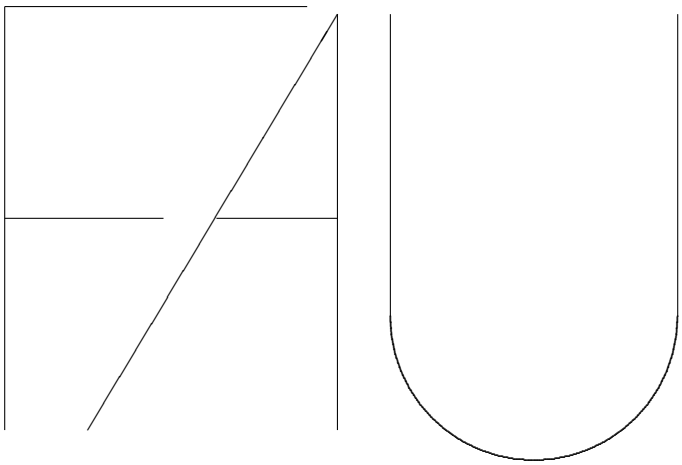
Bachelorarbeit

***Benedikt Lempetzeder***

Lehrstuhl für Informatik 6
(Datenmanagement)

Department Informatik
Technische Fakultät

Friedrich Alexander-
Universität
Erlangen-Nürnberg



# Gegenüberstellung verschiedener Paradigmen zur Darstellung von Prozesseigenschaften unter Berücksichtigung von Zeit und Daten

Bachelorarbeit im Fach Informatik

vorgelegt von

**Benedikt Lempetzeder**

geb. 28.04.1988 in Lichtenfels

angefertigt am

**Department Informatik**
**Lehrstuhl für Informatik 6 (Datenmanagement)**
**Friedrich-Alexander-Universität Erlangen-Nürnberg**

Betreuer: Univ.-Prof. Dr.-Ing. habil. Richard Lenz
Dipl.-Inf. Christoph P. Neumann

Beginn der Arbeit: 01.05.2011
Abgabe der Arbeit: 30.09.2011

# Erklärung zur Selbständigkeit

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass diese Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Der Universität Erlangen-Nürnberg, vertreten durch den Lehrstuhl für Informatik 6 (Datenmanagement), wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Bachelorarbeit einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt.

Erlangen, den 30.09.2011

(Benedikt Lempetzeder)

# Kurzfassung

## Gegenüberstellung verschiedener Paradigmen zur Darstellung von Prozesseigenschaften unter Berücksichtigung von Zeit und Daten

Die Modellierung von Prozessen nimmt in jedem Unternehmen eine tragende Rolle ein. Da die Anforderungen an diese Prozesse, je nach Einsatzzweck unterschiedlich sind, existieren auch verschiedene Ansätze diese Prozesse zu modellieren. Es gibt dazu im Allgemeinen zwei verbreitete Paradigmen: das inhaltsbasierte Paradigma und das aktivitätsbasierte Paradigma. Jene Paradigmen legen unterschiedliche Schwerpunkte in der Modellierung, sie kommen in den Modellierungssprachen zum Ausdruck. Diese Sprachen werden in dieser Arbeit auf Aussagekraft untersucht und gegenübergestellt. Dabei wird der Fokus auf bestimmte Modellierungskonstrukte gelegt, die in Verbindung mit Kontrollfluss und Datenfluss stehen. Diese Konstrukte sind elementare Modellierungsbausteine, wie zum Beispiel Sequenz. Aufbauend auf dieser Untersuchung wird ein Modell für den $\alpha$-Flow Ansatz aufgestellt. Dieses Modell beschreibt das Modellieren aller betrachteten Konstrukte auf Grund von Datenabhängigkeiten. Anschließend werden die vorher betrachteten Ansätze zur Prozessmodellierung mit dem entworfenen Modell gegenübergestellt, um einen Vergleich der Aussagekraft zu erhalten. Dazu werden etablierte Evaluierungsframeworks benutzt. Als Abschluss wird geprüft, ob das aufgestellte Modell, rein aufbauend auf Datenabhängigkeiten zum Modellieren der Konstrukte geeignet ist.

# Abstract


## Comparison of different paradigms for representation of process characteristics considering time and data

The modelling of processes takes over a significant role at all kinds of enterprises. As requirements of processes are different for each operation purpose, there are distinct approaches to model processes. In general there are two common paradigms: the content-oriented paradigm and the activity-oriented paradigm. Each paradigm has its distinct emphases in modelling. The emphases are reflected in modelling notations. The notations were under examination for their explanatory power and were opposed to each other. The comparison concentrates on specified constructs, which are related to controlflow and dataflow. Constitutive to this examination a model for the $\alpha$-Flow beginning was built. This model characterises the modelling of all compared constructs only with data dependencies. Subsequently the former described approaches were faced to this model, to get a relation of explanatory power. Thereto well-established evaluation frameworks were used. In the end the model is evalutated if the model is adequate for modelling only with data dependencies.

# Danksagung

Ich möchte mich hiermit bei Herrn Prof. Dr.-Ing. Richard Lenz und Herrn Christoph P. Neumann für ihre Unterstützung bedanken. Sie standen mir immer mit Rat und Tat zur Seite und ich hätte mir keine bessere Unterstützung vorstellen können. Ein weiterer Dank geht an alle, die mich in irgendeiner Form unterstützt haben, insbesondere Florian Weikert, Bert Riffelmacher und Michael Günter, die sich Zeit für die Korrektur der Arbeit genommen haben.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **ACM** | Adaptive Case Management |
| **AD** | Aktivitätsdiagramm |
| **ADEPT** | Application Development Based on Encapsulated Pre-modeled Process Templates |
| **BI-RADS** | Breast Imaging-Reporting and Data System |
| **BPEL4WS** | Business Process Execution Language for Web Services |
| **BPDM** | Business Process Definition Metamodel |
| **BPMI** | Business Process Management Initiative |
| **BPMN** | Business Process Modelling Notation |
| **BWW** | Bunge-Wand-Weber |
| **DAB** | Datenabhängigkeit |
| **CSCW** | Computer-Supported Cooperative Work |
| **EPK** | Ereignis gesteuerte Prozessketten |
| **eEPK** | erweiterte Ereignis gesteuerte Prozesskette |
| **FODA** | Feature-Oriented Domain Analysis |
| **IDEF3** | Integrated DEFinition for Process Description Capture Method |
| **IT** | Informationstechnologie |
| **OMG** | Object Management Group |
| **UML** | Unified Modelling Notation |
| **YAWL** | Yet Another Workflow Language |

# 1 Einleitung

Prozesse spielen in allen Unternehmen eine tragende Rolle. Sie können aus allen verschiedenen Geschäftszweigen stammen. Dort werden sie dafür eingesetzt, um Abläufe im Unternehmen ganz oder teilweise zu automatisieren und dadurch eine Steigerung der Effizienz zu erreichen. Ein Prozess im medizinischen Umfeld ist beispielsweise die Behandlung eines Patienten, der in Folge von Beschwerden den Arzt aufsucht und verschiedene Stationen der Behandlung und Pflege durchlaufen muss. Dort sind z.B. inter-institutionelle Prozesse zu beherrschen und zu modellieren. Im Bereich der Wirtschaft (z.B. Produktionsstätten) ist ein typischer Prozess der sogenannte Fließbandprozess, bei dem Schritt für Schritt, Aufgabe für Aufgabe erledigt werden muss, um am Ende ein fertiges Produkt zu erhalten. Zum Beispiel gibt es in der Industrieproduktion Prozesse, die dafür benutzt werden Produktionsketten effizient abzubilden. Aus dem der Bereich der Dienstleistung (z.B. Banken) könnte ein Prozess die Anwerbung und Einstellung von neuen Personal sein. Je nach Geschäftszweig des Unternehmens stellen sich unterschiedliche Herausforderungen für die Prozessmodellierung, die für die Informationstechnologie (IT)-Abteilung genauso gelten, wie für kaufmännische Abteilungen.
Für Modellierungsaufgaben stehen bestimmte Notationen zur Verfügung, die teilweise von unabhängigen Konsortien standardisiert sind oder teilweise unternehmensintern entwickelt sind. Allen gemein ist die Intention mit ihren Mitteln den Prozess auf bestmögliche Art und Weise abzubilden, um das Ziel der Automatisierung, Effizienzsteigerung und Dokumentation zu erreichen. Dabei wird unterschieden, ob Prozesse formal oder semi-formal spezifiziert werden sollen. Es kann auch unterschieden werden, ob ein Prozess nur zu Dokumentationszwecken oder zur Ausführung modelliert wird.
Ziel dieser Arbeit ist es, diese Prozessbeschreibungssprachen auf deren Darstellungsmöglichkeiten und Ausdrucksmächtigkeit hinsichtlich Daten- und Kontrollfluss zu untersuchen und gegenüber zu stellen. Diese Arbeit ist Teil des ProMed Forschungsprojekts, das ein lose gekoppeltes inter-institutionelles Prozessunterstützungssystem im medizinischen Kontext entwickelt.

## 1.1 Problemdefinition

Durch Vorhandensein der Prozesse in unterschiedlichen Bedarfsfeldern und Geschäftszweigen stellen sich auch unterschiedliche Anforderungen an die Mächtigkeit und Aussagekraft der Modellierungsnotationen. Beispielsweise werden im Gesundheitswesen andere Anforderungen gestellt als bei der Modellierung von Fließbandprozessen.
Es gibt deshalb unter den Modellierungsarten verschiedene Vorgehensweisen Prozesse abzubilden. Grundsätzlich lässt sich das Datengetriebene und das Aktivitätsgetriebene Paradigma benennen (vergleiche Kapitel 3.4). Der Einsatz eines dieser beiden Paradigmen führt deshalb zu unterschiedlichen Semantiken der Prozessausführung und Modellierung von Prozessen.
Deswegen wird eine Gegenüberstellung der verschiedenen Prozessmodellierungsnotationen durchgeführt. Anhand davon werden die Kontrollfluss- und Datenflussmöglichkeiten analysiert. Das Ergebnis dieser Analyse wird in Bezug zum ProMed Forschungsprojekt $\alpha$-Flow gestellt.
Bei der Gegenüberstellung wird die Herkunft und die Geschichte der verschiedenen Sprachen untersucht, genauso wie die Einsatzgebiete, bestimmte Prozessmuster und die generelle Leistungsfähigkeit in Bezug auf die Modellierung von Prozessen.
Bei medizinischen Prozessen beispielsweise ist es nötig, dass verschiedene Fachabteilungen effektiv und verteilt kooperieren können. Dabei müssen Informationen jederzeit für das medizinische Fachpersonal zugänglich sein, um darauf aufbauend Entscheidungen treffen zu können. Die Informationen sollen effektiv und zielgerichtet zwischen den einzelnen medizinischen Fachabteilungen ausgetauscht werden können. Dabei sollte die Effizienz der einzelnen Fachabteilungen durch die Prozesse gesteigert werden. Der Austausch bestimmter Prozessstrukturen und Nachrichten basiert auf Kommunikationsstandards und Kommunikationsinfrastrukturen. Die Kommunikationsstandards und Kommunikationsstrukturen unterliegen einer mangelhaften Standardisierung, weshalb die zuverlässige und integrierte Verwendung weitere Schwierigkeiten aufwirft [LR07],[LBM+05]. Die Prozesse sind von vorne herein nicht planbar, weil sie auf Entscheidungen des geschulten und spezialisierten medizinischen Personals setzten und diese Entscheidungen erst zur Laufzeit des eigentlichen Prozesses stattfinden [DRK97].

## 1.2 Ziel der Arbeit

Ziel dieser Arbeit ist die ganzheitliche Gegenüberstellung von Zeitflussmodellen und Datenflussmodellen im Kontext von Prozessmodellen. Die Gegenüberstellung findet in Hinsicht auf Herkunft und Geschichte, Einsatzgebiete, Prozessmuster und Leistungsfähigkeit statt. Es werden in Folge dieser Gegenüberstellung verschiedene, weit verbreitete Ansätze der Prozessmodellierung analysiert. Der medizinische Kontext steht hier im Vordergrund, weil das ProMed Forschungsprojekt zur Unterstützung von inter-institutionellen, verteilten Prozessen im Gesundheitswesen gedacht ist. Jedoch soll das Ergebnis der Arbeit auch Allgemeingültigkeit besitzen, da der medizinische Kontext nur eine Beispieldomäne für Prozesse darstellt [DRK97].
Die Arbeit ist Teil des ProMed Forschungsprojektes des Lehrstuhls 6 (Datenmanagement) der Friedrich-Alexander Universität Erlangen-Nürnberg. Ein Bereich des Forschungsprojektes ist $\alpha$-Flow. $\alpha$-Flow bietet eine Prozessunterstützung im medizinischen Umfeld an. $\alpha$-Flow soll in der Arbeit als Referenz dienen. Es sollen die Erkenntnisse, die aus der Gegenüberstellung der anderen Beschreibungsarten kritisch hinterfragt und auf $\alpha$-Flow übertragen werden.
Mit Hilfe dieser Arbeit soll Folgendes geklärt werden: Welche Arten der Modellierung von Abhängigkeiten der Daten werden benötigt? Benötigt man Vorwärtsverkettung und Rückwärtsverkettung oder genügt einer der beiden Ansätze zur Modellierung von Prozessen im medizinischen Umfeld aus? Vorwärtsverkettung bedeutet, dass der Prozessfortschritt durch das Beendigen einer Aktivität zur nächsten Aktivität getriggert wird. Rückwärtsverkettung bedeutet genau das Gegenteil, da es den Prozessfortschritt auf Grund von Datenabhängigkeiten definiert und freischaltet. Welche Modellierungskonstrukte schränken die Mächtigkeit der Notation ein bzw. welche sind unbedingt nötig, z.B. in Hinblick auf die Modellierung von zeitlichen Abfolgen? Die Erkenntnis aus den Antworten auf diese Fragen sollen dann auf das $\alpha$-Flow Projekt übertragen werden. Durch die Evaluation dieses Erkenntnisgewinnes wird die Ausdrucksmächtigkeit des $\alpha$-Flow Ansatzes, im Vergleich zu den weitverbreiteten Modellierungsnotationen untersucht. Denn als Grundmotivation für die Arbeit stellt sich die Frage, ob es möglich ist, rein aufbauend auf Datenabhängigkeiten die gleiche Ausdrucksmächtigkeit zu erreichen, wie die anderen Ansätze.

## 1.3 Abgrenzung

Es gibt viele Ansätze, die sich mit dem Vergleichen von Prozessunterstützungssystemen beschäftigen [AHKA03]. Der Schwerpunkt dieser Arbeit richtet sich jedoch gezielt auf grafische Notationsarten, die allgemein benutzbar und nicht nur in Bezug auf ein fertiges Prozessunterstützungssystem Gültigkeit besitzen. Anhand der in [AHKA03] aufgeführten Betrachtung, lässt sich jedoch auch eine guter Überblick über die Vergleichsmöglichkeiten von Prozessunterstützungssystemen gewinnen.
Des Weiteren ist Business Process Execution Language for Web Services (BPEL4WS) kein Gegestand der Betrachtung, weil in der vorliegenden Analyse nur grafische Modellierungsnotationen betrachtet werden und BPEL4WS blockorientiert und textuell (Extended Markup Language) modelliert wird [LLN11]. Es sei jedoch erwähnt, dass BPEL4WS trotzdem ein integraler Bestandteil der Prozessmodellierung ist, da es laut [LLN11] immer noch eine weit verbreitete Beschreibungssprache für die Prozessausführung darstellt. Deshalb gibt es eine Vielzahl von Ansätze, die Prozessmodellierungsnotationen auf BPEL4WS Prozessbeschreibungen abbilden (YAWL: [BP05], Petri-Netze:[HSS05], BPMN: [ODTHVDA06], EPK: [Sch07], UML: [Gar03]).
In [MRH07] wird auf komplexe Prozessstrukturen eingegangen, die mit Hilfe von datengetriebenen Verfahren koordiniert und abgewickelt werden sollen. Dies liegt jedoch nicht im Fokus dieser Arbeit, denn diese Prozessstrukturen stellen eine Ablaufbeschreibung von z.B. Betriebssystemen dar.
Diese Arbeit ist eine konzeptionell-analytische Arbeit die Implementierungsdetails vernachlässigt und sich auf die reine Aussagekraft und Semantik der Syntax von Prozessbeschreibungsnotationen konzentriert.

# 2 Methodik

Für die Arbeit wurde eine umfangreiche Literaturrecherche betrieben, die das Ziel hatte, bestehende Ansätze zu Prozessmodellierung aufzudecken, welche Methoden und Notationen weit verbreitet in Gebrauch sind und welche es dafür gibt. Des Weiteren wurden bestehende komparative Analysen von Prozessmodellierungsnotationen recherchiert, um sich einen Überblick über Unterschiede und Gemeinsamkeiten der bestehenden Notationen zu machen und Aufschluss über ihre Stärken und Schwächen zu erhalten.
Um den Vergleich in der Aussagemächtigkeit der verschiedenen Prozessnotationen und dem resultierenden Modell für $\alpha$-Flow zu evaluieren, war es nötig, sich ein Evaluationsframework auszusuchen (vergleiche Kapitel 3.5).
Die Arbeit ist folgendermaßen erarbeitet worden: Zuerst wurden die Grundlagen des Themengebietes erarbeitet (siehe Kapitel 3). Diese Grundlagen werden in Kapitel 3 beschrieben. Hier werden unter anderem die Grundlagen für Workflows bzw. Prozesse angegeben, die Grundlagen der verschiedenen Paradigmen und deren Abgrenzung untereinander. Die Einführung in zwei bewährte Evaluierungsframeworks wird ebenfalls hier gegeben. Zudem wird hier eine Kurzdarstellung des $\alpha$-Flow Ansatzes gegeben. Daraufhin wurde vergleichende und verwandte wissenschaftliche Literatur gesichtet. Ein Überblick darüber findet sich in Kapitel 4. Es wurde daraufhin eine Motivation erarbeitet, um auf die darauffolgende Gegenüberstellung hinzuführen (siehe Kapitel 5). Daraufhin wurde die konzeptionelle Darstellung der Gegenüberstellung der Prozessmodellierungsnotationen erarbeitet (siehe Kapitel 6). Dies erfolgt in einem Abschnitt je Prozessmodellierungsnotation. Zu Beginn der Analyse wurde ein Beispielprozess und Anforderungen an die Notation definiert, die je Notation als Grundlage der Analyse dienen soll und deren Stärken/Schwächen bzw. Unterschiede/Gemeinsamkeiten veranschaulichen soll. Aus den Erkenntnissen dieser Gegenüberstellung wurde ein Modell für $\alpha$-Flow entwickelt und in Beziehung zu den Anforderungen an Prozessmodellierungsnotationen gesetzt (siehe Kapitel 7). Das entstandene Modell wurde daraufhin auf Grundlage der Evaluierungsframeworks evaluiert (siehe Kapitel 8). Hier wird gezeigt, wie mächtig bzw. aussagekräftig $\alpha$-Flow im Vergleich zu den dargestellten Ansätzen ist. Abschließend wird

in Kapitel 9 eine Zusammenfassung und ein Ausblick gegeben.

# 3 Grundlagen

In diesem Kapitel werden Grundlagen erarbeitet, die für das weitere Verständnis der Arbeit wichtig sind. Zunächst werden im Abschnitt 3.1 Grundlagen der Prozessmodellierung beschrieben. Dies beinhaltet die Beschreibung des Tokenkonzeptes (3.1.1), Erläuterung der Begriffe Daten- und Kontrollfluss (3.1.2). Hierauf wird eine kurze Darstellung des $\alpha$-Flow Ansatzes (3.2) gegeben. Anschließend werden zwei besonders relevante Darstellungen zu $\alpha$-Flow verwandten Ansätzen dargestellt: Proclets und Application Development Based on Encapsulated Pre-modeled Process Templates (ADEPT). Darauf folgt eine kurze Gegenüberstellung der betrachteten Paradigmen der Prozessmodellierung (3.4). In der Sektion BWW-Modell (3.5.2) und Workflow Pattern (3.5.1) werden zwei Evaluierungsframeworks für Prozessmodellierungsnotationen vorgestellt, die in den folgenden Kapiteln zur Evaluierung benutzt werden.

## 3.1 Grundlagen der Prozessmodellierung

Modellieren bedeutet nach Jablonski et al. [JBS97], dass man von einem Gegenstand oder in diesem Fall von einem Prozess ein Modell erstellt; ein Modell wiederum ist eine Darstellung der realen Welt, die aber in ihren Funktionen und ihrer Ausprägung vereinfacht ist, dass eine leichtere Handhabung bestimmter Belange möglich wird.
Ein Prozess beschreibt einen Ablauf, der eine Folge von Aktionen beinhaltet. Diese sind in der Regel geordnet und möglicherweise parallel, synchron oder asynchron strukturiert [CS06]. Prozessmodellierung ist daraus folgend die Modellierung von Prozessen. Mit anderen Worten: die Abläufe aus der realen Welt in eine handhabbare, vereinfachte Form zu bringen. Dies geschieht z.B. weil man den Prozess automatisieren will [LLN11]. Um diese Modelle zu erstellen, gibt es verschiedene Grammatiken. Dies sind die Prozessmodellierungsnotationen, die in dieser Arbeit analysiert und gegenübergestellt werden.

### 3.1.1 Das Token-Konzept

Ein Token im Kontext von Prozessmodellierung ist ein theoretisches Konstrukt, mit dem sich der Ablauf von Prozessmodellen veranschaulichen lässt ([FRH10], [OMG11b]). In [FR10] wird das Token mit einem Auto verglichen, indem das Auto den Straßenverlauf verfolgt, wie das Token in gleicher Weise den Prozessverlauf (vergleiche Abbildung 3.1). So wie ein Auto sich an manchen Stellen auf der Straße entscheiden muss, in welche Richtung es fährt, muss auch das Token verschiedene Wege durch einen Prozess nehmen. So kann das Token von Aktivität zu Aktivität fließen und damit die Prozessausführung determinieren und triggern. Triggern bedeutet in diesem Kontext, dass das Token von einer Aktivität an die nächste Aktivität weitergegeben werden muss und damit die nächste Aktivität beginnen lässt. Das Token-Konzept findet Anwendung in verschiedenen aktivitätsbasierten Prozessmodellierungsnotationen (siehe Abschnitt 3.4.1) z.B. Petri-Netze [Aal98a], BPMN [OMG11a], UML [OMG11b].

### 3.1.2 Daten-und Kontrollfluss

**Datenfluss** ist der Austausch von Datenobjekttoken, der anzeigt, welche Aktivität welche Daten benötigt, produziert und verarbeitet bzw. verändert. Daten können in der Domäne der Prozessmodellierung von einfachen textuellen Nachrichten über Listen bis hin zu komplexen Datenobjekten reichen.

**Kontrollfluss** oder auch *Sequenzfluss* legt die Reihenfolge (sowohl temporal als auch kausal) der Abarbeitung des Prozesses fest (Kontrolltoken). Dadurch lässt sich auch die Nebenläufigkeit bzw. Bedingtheit des Prozesses ausdrücken [JBS97].

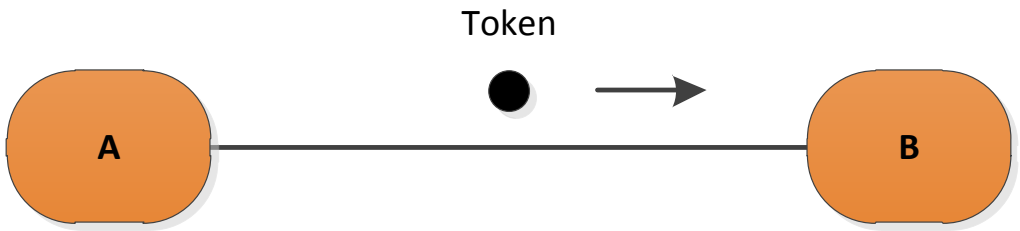



**Bild 3.1:** Token „fließt“ über Kante von A nach B (z.B. Aktivitäten)

## 3.2 Kurzdarstellung von α-Flow

Der Abschnitt basiert auf den bisher veröffentlichten Artikeln der Forschungsgruppe des Lehrstuhl 6 der Friedrich-Alexander Universität Erlangen-Nürnberg: [NL09], [NL10], [NL12] und [NSWL11]. α-Flow ist ein Prozessunterstützungssystem für interinstitutionelle Prozesse im medizinischen Umfeld, das auf dem Ansatz des Case-Handlings basiert (vergleiche Bild 3.3 und Abschnitt 6.7). Mit dem Case-Handling Ansatz besteht vor allem die Gemeinsamkeit über das Erreichen von Prozessfortschritt. Dieser wird in α-Flow auch durch das Bereitstellen von Daten erreicht und freigeschalten. Als Beispiel aus dem medizinischen Kontext ist das Bereitstellen von untersuchungsrelevanten Daten zu nennen. Erst wenn diese Daten bereit stehen, kann der Prozessfortschritt freigeschalten werden.

### Das α-Doc

Im Mittelpunkt von α-Flow steht das α-Doc. Das α-Doc ist ein aktives Dokument. Das α-Doc stellt die kleinste Einheit des Austausches dar. Der Inhalt eines α-Docs sind medizinische Befunde, Diagnosen usw., aber auch Metadaten bzw. Koordinierungsdokumente, die den technischen Ablauf steuern. Diese Metadaten bieten Informationen für Benutzerverwaltung, Zugriffssystem, Log-Dateien und Versionsmanagementsystem. In einem α-Doc sind Sichten auf den Inhalt integriert. Das Ziel des α-Docs ist die vollständige Entkopplung von anderen Applikationssystemen, um einen optimalen Austausch von Informationen in einer verteilten Umgebung (z.B. verschiedenen Abteilungen eines Krankenhauses) ohne Installation zu gewährleisten und die Gesamtheit der Daten zu jeder Zeit zur Verfügung zu haben. Damit wird eine vollständige Systemintegration gewährleistet.

### Die α-Card

Ein α-Doc kann aus mehreren α-Cards bestehen. Diese beinhalten medizinische Befunde oder Koordinierungsdaten (z.B. Zugriffsrechte). Die α-Card ist die kleinste Einheit der Validierung. Die α-Card hat viele Eigenschaften, die sich unter dem Begriff der Adornments zusammenfassen lassen. Es besteht unter anderem eine α-Card gültig (valid) oder ungülitg (invalid) und sichtbar (public) oder unsichtbar (private) zu kennzeichnen. Diese

beiden Eigenschaften (Sichtbarkeit und Gültigkeit) sind dabei voneinander unabhängig. Des Weiteren ist dort ein Versionenmodell, Variantenmodell, Authorisierungs- und Authentisierungsmodell und ein Verbindungsmodell ($\alpha$-Card $\leftrightarrow$ $\alpha$-Card) integriert.

#### Die $\alpha$-Episode

Den kompletten Behandlungsprozess beschreibt der Begriff der $\alpha$-Episode. Pro $\alpha$-Episode wird ein $\alpha$-Doc geführt. Zur Abarbeitung einer solchen $\alpha$-Episode wird eine Priorisierungsliste bereitgestellt. Diese wird bei Erstellung eines $\alpha$-Docs durch das Einfügen von $\alpha$-Card Deskriptoren aufgebaut. Jede dieser $\alpha$-Card Deskriptoren stellt einen Eintrag in der Priorisierungsliste dar. Jeder dieser Einträge ist von Beginn an der $\alpha$-Episode sichtbar, um ihn zu bearbeiten. Diese $\alpha$-Card Deskriptoren müssen noch mit prozessrelevanten Informationen befüllt werden, um Prozessfortschritt freizuschalten.
$\alpha$-Flow will somit Prozessen gerecht werden, die von vornherein nicht planbar, unstrukturiert und zur Laufzeit durch ad hoc Entscheidungen änderbar und umplanbar sind. Die Benutzer des $\alpha$-Flow Systems sollen damit in ihrer Ausführung des Prozesses nicht eingeschränkt, sondern unterstützt werden. Dies soll fundamental nur auf Grund von Datenabhängigkeiten passieren.

## 3.3 Besondere Ansätze

In den folgenden beiden Abschnitten werden zwei besonders relevante Ansätze zur Prozessmodellierung aufgezeigt: Proclets und ADEPT. Diese beiden Ansätze sind besonders relevant, da sie besonders innovativ sind und z.B. auch zu Laufzeitänderungen in der Lage sind.

### 3.3.1 Proclets

In diesem Abschnitt wird lediglich eine kurze Darstellung des Proclet-Ansatzes gezeigt, da er einige Besonderheiten in der Darstellung von Prozessen bietet, deshalb wird hier auch keine Forderung auf Vollständigkeit erhoben. Der Proclet-Ansatz findet auch Verwendung im Gesundheitswesen [MRVDA+10].
Proclets werden definiert als leichtgewichtige Workflow Prozesse [APEJ00]. Es werden fünf

Nutzen von Proclets gegenüber typischen Ansätzen zur Prozessmodellierung angegeben [APEJ01]:

- Die durch Proclets definierten Prozesse sind näher an der Realität, da nicht der komplette Prozess in ein Modell gezwängt werden muss
- Die statische Sicht eines Klassendiagramms passt gut zu der dynamischen Sicht von Proclets
- Proclets sind ausdrucksmächtiger als bisherige Verfahren
- Proclets sind wiederverwendbar
- Prozesselemente mit verschiedenen Aggregationsstufen lassen sich besser handhaben

Mit Proclets wird der Schwerpunkt der Modellierung von Kontrolle auf die Kommunikation zwischen Proclets verschoben [AMR09].
Die einzelnen Bestandteile eines Procletsprozesses sind [APEJ00]:

- Die Proclets an sich, die Objekte des Prozesses darstellen (Dokument, Patient etc.) spezifiziert durch YAWL-Modelle:
  - Performatives zur Kommunikation zwischen verschiedenen Proclets
  - Ports, die Proclets mit den Channels verbinden
  - Knowledge Base, in der die Performatives gespeichert werden können und die sich weiterentwickeln kann
- Kommunikations Channels über die die Perfomatives zwischen den Proclets ausgetauscht werden können
- Naming Service, in dem die erreichbaren Proclets erfragbar sind
- Actors, sie sind Besitzer von Proclets und sind auch als externe Proclets spezifiziert

Durch das Erzeugen eines Proclets wird der Prozess gestartet und ihr jeweiliges YAWL-Modell abgearbeitet. Wenn im Prozess eine Kommunikation zwischen zwei Prozessentities, wie zum Beispiel einem Blatt Papier und einem Bleistift spezifiziert ist, dann werden diese beiden Entities, die beide durch Proclets modelliert sind, eine Kommunikation miteinander haben. Dies wird durch Channels modelliert. Über diese Channels werden Performatives ausgetauscht, die prozessrelevante Informationen beinhalten.
So lassen sich Prozesse modellieren, die aus vielen Proclets bestehen, die miteinander kommunizieren.

### 3.3.2 Application Development Based on Encapsulated Pre-modeled Process Templates

In dieser Sektion soll eine kurze Darstellung von Application Development Based on Encapsulated Pre-modeled Process Templates (ADEPT) gegeben werden. Der Schwerpunkt dieser Darstellung liegt auf der Besonderheit des ADEPT-Ansatzes: der Veränderbarkeit eines Prozessmodelles zur Laufzeit [DRK97], [RD98], [LRD06] und [MGR04].
Ein Grund für die Entwicklung von ADEPT war, dass der Einsatz von Prozessmanagement in der Healthcare Domäne zu Problemen führte. Insbesondere die ad hoc Abweichungen vom spezifizierten Ablaufplan stellten unüberwindbare Hürden dar [DRK97]. Deshalb wurde begonnen mit ADEPT ein System zu entwickeln, dass es möglich macht auch während der Laufzeit von Prozessen Änderungen vorzunehmen. Diese Änderungen werden sowohl auf syntaktische, als auch auf semantische Korrektheit geprüft [RD98], [LRD06].
Dazu wurde eine eigene Prozessmodellierungsnotation entworfen, die es blockorientiert ermöglicht Prozesse zu definieren. Es werden Tasks, AND-/OR-Split/Join, Synchronisierungskanten zum Synchronisieren von Nebenläufigkeit, Schleifen, globale Datenobjekte und Fehlerknoten unterstützt [RD98].
Auf Grund dieser Notation werden Regeln definiert, die sowohl triviale[1] Änderungen, als auch Änderungen in Schleifendurchläufen ermöglichen sollen [DRK97].
Um semantische Korrektheit nach Änderungen zu gewährleisten, werden semantische Checks durchgeführt, die Abhängigkeiten und Mutual Exclusion als Einschränkungen nutzen [LRD06].
Das AGENTWORK- Framework, welches auf ADEPT basiert, geht noch einen Schritt weiter und will automatische Änderungen während der Laufzeit durch das System erlauben [MGR04]. Dabei werden während der Laufzeit Schätzungen über das Verhalten des Prozesses angestellt, um logische Fehler im Voraus zu entdecken und gegebenenfalls zu beseitigen [MGR04].

1 Eine triviale Änderung wäre zum Beispiel das Hinzufügen eines Tasks

## 3.4 Paradigmen der Prozessmodellierung

In diesen Abschnitt erfolgt eine kurze Einführung in zwei Paradigmen der Prozessmodellierung: Das aktivitätsbasierte (Abschnitt 3.4.1) und das inhaltsbasierte (3.4.2) Paradigma.

### 3.4.1 Aktivitätsbasiertes Paradigma der Prozessmodellierung

Das aktivitätsbasierte Paradigma der Prozessmodellierung stellt die Aktivität, als Kern eines jeden Prozesses in den Mittelpunkt (siehe Bild 3.2). Der Prozessfortschritt wird durch das geregelte Ausführen der Aktivitäten gemäß des Sequenzflusses definiert und determiniert (dies wird triggern genannt). Dabei werden die verschiedenen Aktivitäten durch den Kontrollfluss in eine temporale und kausale Ordnung gebracht. Die daraus resultierenden Prozessmodelle sind dahingehend komplett strukturiert. Sie bieten keine Flexibilität zur Laufzeit an (Ausnahme: ADEPT, siehe Abschnitt 3.3.2), da der Prozessverlauf gemäß der Reihenfolge der Aktivitäten im *happy path*[1] modelliert und vollständig strukturiert ist. Das Token wird im Kontext der aktivitätsbasierten Modellierung benutzt werden, um den Fortschritt im Modell aufzuzeigen. Diese Art der Modellierung wird auch als Vorwärtsverkettung bezeichnet, da sich die temporal und kausal nächste Aktivität durch den nach vorwärts gerichteten Pfeil bestimmen lässt.
Im aktivitätsbasierten Prozessmodellieren fehlt ein generelles Konzept, um die Verwendung von Daten zu beschreiben. Daten werden nur als Hilfswerkzeug und Betriebsmittel zur Erreichung des Prozessziels betrachtet. Daten stellen deshalb nur einen abgeleiteten Faktor, im Beschreiben von Prozessen dar. Sie wirken sich deshalb im Allgemeinen auch nicht auf den weiteren Prozessverlauf aus.

### 3.4.2 Inhaltsbasiertes Paradigma der Prozessmodellierung

Im inhaltsbasierten Paradigma der Prozessmodellierung stehen die Daten im Mittelpunkt (siehe Bild 3.2). So wird der Prozessfortschritt nur erreicht, wenn bestimmte Datenzustände bzw. Attribute vorliegen. Die Daten stellen somit einen Rohstoff bzw. Werkstoff dar, der von verschiedenen Zwischenzuständen hin zu einem Endzustand überführt wird. Diese Abhängigkeit zwischen den Zuständen wird Datenabhängigkeit

1 *happy path* ist die vorgesehene und störungsfreie Abarbeitung eines vorstrukturierten Ablaufplans, der durch das Kontrolltoken bis zum Ende durchlaufen wird.

genannt. Diese Datenabhängigkeiten stehen im Kontrast zu den temporalen und kausalen Verkettungen beim Kontrollfluss, deshalb werden sie Rückwärtsverkettung genannt. Datenmanipulation/-bereitstellung kann auf Grund von Aktivitäten auf den Daten erreicht werden. Somit wird hier ein Datenproduktionsprozess dargestellt, der die Daten des Prozesses von einem Zustand in den nächsten transferiert. Durch dieses Erstellen und Verändern der Daten werden neue Aktivitäten freigeschalten.
Auch im inhaltsbasierten Paradigma wird nur der *happy path*[1] modelliert. Jedoch bietet sich die Möglichkeit zur Laufzeit neue Daten bereitzustellen, auf die angemessen zu reagieren ist. Durch die Möglichkeit zur Laufzeit neue Daten bereitzustellen und darauf zu reagieren besteht die Möglichkeit von dem vorstrukturierten Prozess abzuweichen. Reaktionen auf neue Daten können auf Grund von ad hoc Entscheidungen getroffen werden.

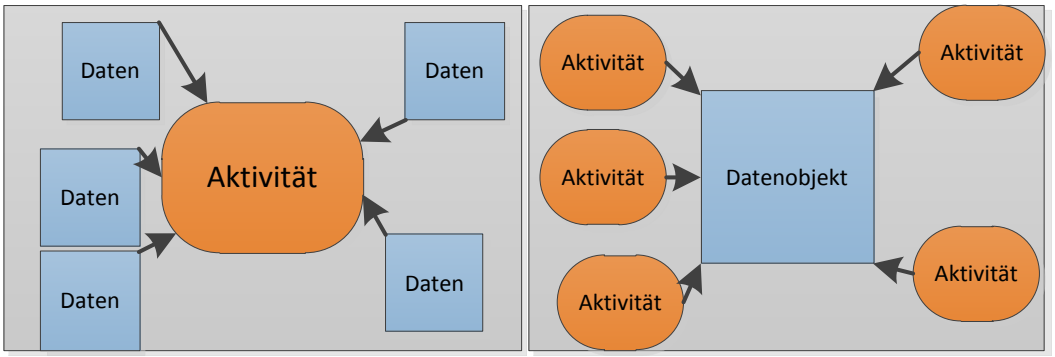



**Bild 3.2:** Rechts das inhaltsbasierte Paradigma, links das aktivitätsbasite Paradigma in Gegenüberstellung ihrer Schwerpunkte

### Abgrenzung der beiden Paradigmen

Zur Abgrenzung der beiden Paradigmen zeigt Abbildung 3.3, dass das aktivitätsbasierte Paradigma prozessgetrieben und strukturiert ist. Das inhaltsbasierte Paradigma umfasst,

1 *happy path* ist der vorgesehene und störungsfreie Entstehungsprozess eines Datenobjekts von der ersten Erstellung über Zwischenprodukte zu einem vollständigen, genauen und fehlerfreien Endprodukt des Datenobjekts.

nach 3.3 auch wenig strukturierte Prozesse, die datengetrieben sind. Im Unterschied dazu steht Groupware (siehe Abschnitt 9.1), welches völlig unstrukturiert und rein datengetrieben ist. Abbildung 3.3 zeigt somit, dass das inhaltsbasierte Paradigma mehr dazu in der Lage ist, mit unstrukturierten Prozessen umzugehen, als das aktivitätsbasierte Paradigma, welches rein prozessgetrieben und vorstrukturiert ist.

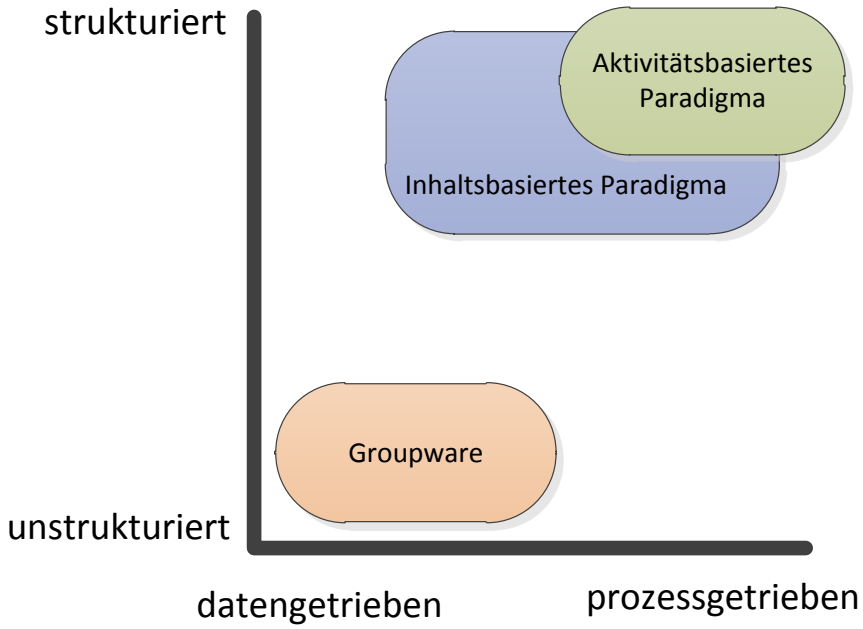



**Bild 3.3:** Abgrenzung von Groupware zu inhaltsbasiertem Paradigma und aktivitätsbasiertem Paradigma, aufbauend auf [AH05]

## 3.5 Evaluationframeworks von Notationen zur Prozessmodellierung

Dieser Abschnitt stellt zwei Evaluierungsframeworks vor. Mit Hilfe dieser Frameworks können Prozessmodellierungsnotationen analysiert und hinsichtlich ihrer Eignung beurteilt werden.

### 3.5.1 Die Workflow Patterns

Es besteht keine zentrale Referenz für Prozessmodellierungsnotationen. Daher kann man auch nicht bestimmen, welche Sprache für seine Zwecke besser bzw. schlechter geeignet ist. Deshalb werden Workflow Patterns definiert, um unabhängig von einer Notation semantische Schwierigkeiten, Passgenauigkeit von Prozessmodellnotationen und deren Ausdrucksmächtigkeit aufzuzeigen [ABHB00]. Diese Workflow Pattern repräsentieren Anforderungen in Prozessen, die durch die verschiedenen Notationen bewältigt werden sollten [ABHB00], [AHKA03].

#### Gruppierung der Muster

Der Schwerpunkt dieses Evaluierungsframeworks liegt auf der Kontrollflussperspektive, da alle anderen Perspektiven (Daten-, Operationale-, und Organisationsperspektive) auf dieser aufbauen [AWG05]. Innerhalb der Perspektiven werden die Muster in Untergruppen aufgeteilt. Zum Beispiel wird die Kontrollflussperspektive aufgeteilt in:

- Die Grundlegenden Muster (z.B. Sequenz)
- Erweiterte Synchronisierung und Verzweigung (z.B. Mehrfachauswahl)
- Muster mit mehreren Aktivitätsinstanzen (Mehrfach-Instanzmuster)
- Zustandsbasierte Muster (z.B. verzögerte Wahl)
- Abbruchsmuster
- Iterationsmuster
- Beendigungsmuster

Dieses Framework bietet im Moment über 40 Patterns an, wobei es zu Beginn nur 20 waren [RHAN11]. Die Anzahl der Muster wurde in mehreren Iterationen erweitert.

#### Gründe für das Framework

Es werden fünf Gründe angegeben, weshalb das Framework in Benutzung ist und für Analysen eingesetzt wird [RVDATHW06]:

1. weit verbreitete Benutzung
2. hohe Akzeptanz

3. verständlich für alle IT Anwender
4. ausreichend detailliert zum Beschreiben der Prozessanforderungen
5. das vollständigste und mächtigste Framework

Durch diese fünf Gründe wird das Workflow Pattern Framework in dieser Arbeit benutzt, um die weitverbreiteten Modellierungsnotationen zu evaluieren.

### Beispiele für Musterevaluationen

In diesem Abschnitt werden zwei Beispiele von Mustern betrachtet. Für eine komplette Übersicht der Muster und deren Evaluationskriterien sei auf [RHAN11] verwiesen. Diese beiden Muster seien hier erwähnt, da sie später von Relevanz sein werden. Als erstes Beispiel eines der Muster, wird eines der Mehrfach-Instanzmuster vorgestellt. Bei Mehrfach-Instanzmuster sollen mehrere Instanzen einer modellierten Aktivität parallel ablaufen können. Dabei wird unterschieden, wann bekannt sein muss, wie viele Instanzen ablaufen sollen: Vor der Laufzeit oder während der Laufzeit. Wir betrachten hier als Beispiel das Mehrfach-Instanzmuster *Mehrfach-Instanzen ohne vorherige Laufzeitinformationen*. Dazu darf vor der Laufzeit nicht bekannt sein, wieviele Instanzen einer Aktivität zur Laufzeit aktiv sein werden. Als Evaluationskriterium wird angegeben, dass es ein Kontrollflusskonstrukt geben muss, das dafür sorgt mit derartigem Verhalten umzugehen.
Als zweites Beispiel eines der Muster wird hier das Muster Sequenz, nach [RHAN11] (siehe Abbildung 3.4) gezeigt und erklärt. Sequenz lässt sich in die Kategorie der Kontrollmuster einordnen. Es wird folgendermaßen beschrieben: Eine Aufgabe in einem Prozess wird erlaubt, wenn die vorherige Aufgabe in dem selben Prozess abgearbeitet ist. In Abbildung 3.4 bedeutet dies, dass Aufgabe B erst bearbeitet werden darf, wenn Aufgabe A abgearbeitet ist und man in die Stelle p1 übergegangen ist. Des Weiteren wird für jedes Muster ein Erfüllungskriterium angegeben. Im Beispiel der Sequenz lautet es folgendermaßen: Volle Unterstützung für das Muster wird gezeigt, wenn eine explizite Repräsentation von Abhängigkeit (z.B. gerichteter Pfeil) zwischen zwei Aufgaben unterstützt wird, welche eine Abarbeitungsreihenfolge festlegt. Wenn dieses Erfüllungskriterium unterstützt wird, dann wird das Muster als unterstützt angegeben.

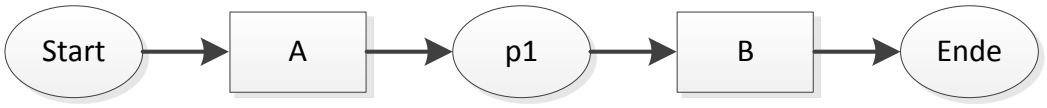



**Bild 3.4:** Darstellung des Sequenzmusters nach [RHAN11]

### 3.5.2 Das Bunge-Wand-Weber-Modell

Das Bunge-Wand-Weber-Modell (BWW) [WWDG+90] basiert auf einer Ontologie von Bunge aus den 50er Jahren, die Konstrukte zur Beschreibung von ontologischer Vollständigkeit anbietet. Ontologische Vollständigkeit bedeutet, dass die reale Welt vollständig abgebildet werden kann. Es soll demnach kein natursprachliches Realweltkonstrukt (z.B. Patient) geben, das nicht auch in der Modellierungsnotation abgebildet werden kann.
Wand und Weber haben diese Ontologie für ihre Sicht auf Informationssysteme benutzt. Sie haben mehrere Modelle aufgestellt, die für Informationssysteme gelten müssen: Das Repräsentationsmodell, das Zustandsverfolgungsmodell und das Zerlegungsmodell. Diese Modelle beschreiben das notwendige Verhalten und die Struktur von Informationssystemen, um geeignet für den Gebrauch zu sein [Web95]. In dieser Arbeit wird nur das Repräsentationsmodell betrachtet, da es für unsere Zwecke ausreicht. Die beiden anderen Modelle beziehen sich nicht auf die Modellierungsnotationen, sondern auf andere Ebenen eines Informationssystems, die in dieser Arbeit nicht von Relevanz ist.
Als Grundlage für das Repräsentationsmodell dient der iterative Entwicklungsprozess eines Informationssystems. Wand und Weber postulieren die Bedingung, dass in jeder Iteration die ontologische Vollständigkeit bis hin zur Maschinenebene, der Informationssystemkonstrukte gewahrt sein muss. Dies bedeutet, dass jedes natursprachliche Konstrukt, das als Anforderung an das Informationssystem festgehalten wurde, auch auf Maschinenebene eine eindeutige Bedeutung hat. Daraus folgt, dass die Sprachen, die dieses System beschreiben sollen, die statischen und dynamischen Eigenschaften der realen Welt eindeutig beschreiben können müssen.

#### Konstrukte des BWW-Modells und deren Abbildungsbeziehungen

In [WWDG+90] sind notwendige und hinreichende Konstrukte beschrieben, um die ontologische Vollständigkeit zu beweisen. Diese Konstrukte lassen sich gliedern in 4 Grup-

pen: Gegenstände und ihre Eigenschaften, Zustände und Ereignisse, Transformationen und Ereignisse von Gegenständen und Systemstruktur von Gegenständen. Gegenstände bilden die Grundlage einer Prozessmodellierungsnotation. Ein Gegenstand kann demnach z.B. eine Aufgabe, ein Datenobjekt sein. In Kapitel 8 werden alle Konstrukte benötigt, deshalb sei für eine komplette Aufstellung aller Konstrukte auf das Paper [WWDG+90] verwiesen. Der Beweis der vollständigen Ontologie geschieht durch Abbildung von den Konstrukten der Prozessmodellierungssprache auf die Konstrukte des BWW-Modells. Diese Abbildung erfolgt durch das Überprüfen der Konstrukte, ob sie von der zu evaluierenden Modellierungsnotation unterstützt werden oder nicht. Dies führt zu 1:n-, 1:1- ,n:1-, 0:1 und 1:0-Abbildungen (siehe Bild 3.5), die eine Rolle für die ontologische Klarheit der Notation spielen [RRIG09], [RRK07]. Ontologische Klarheit bedeutet, dass gewisse Redundanz vorherrscht, wenn z.B. das Konstrukt Gegenstand gegen mehrere Konstrukte des BWW-Modell abbildbar sind. Damit wird sozusagen gezeigt wie einfach und intuitiv die untersuchte Modellierungsnotation ist, denn umso mehr Konstrukte der Modellierungsnotation auf ein Konstrukt des BWW-Modells abbildbar sind, desto unklarer wird dem Benutzer der Notation, welches Konstrukt er benutzen soll bzw. was eine bestimmtes, verwendetes Konstrukt aussagen soll. Die 1:1 - Abbildung beschreibt die ontologische Vollständigkeit, die anderen (bis auf 1:0 und 0:1) die ontologische Klarheit.

#### Beziehung von BWW-Modell und den Workflow Pattern

Das BWW-Modell ist somit abstrakter als das Framework der Workflow Pattern, da es nicht auf einzelne detaillierte Konstrukte der Sprachen abzielt, sondern auf die generelle Darstellungsmächtigkeit der Notationen [RRIG09]. Laut [RRK07] gibt es Gemeinsamkeiten zwischen den Workflow Pattern und dem BWW-Modell. Es wurden verschiedene Abbildungen zwischen einer Modellierungsnotation und den Evaluierungsframeworks betrachtet. Diese Abbildung wurde mit Workflow Patterns durchgeführt, wie auch mit dem BWW-Modell. Als Ergebnis wurde festgehalten, dass bei der Beziehung Äquivalenz (1:1 Abbildung), die Workflow Patterns wegen der kleineren Granularität mehr Information bieten. In der Defizit- (kein Konstrukt des Evaluierungsframeworks unterstützt) und Unterscheidbarkeitsbeziehung (mehr Konstrukte unterstützt) stimmen Workflow Pattern und BWW-Modell überein. In der Beziehung Mehrdeutigkeit (je Konstrukt bietet die Modellierungsnotation mehr Abbildungen an) widersprechen sich die beiden Frameworks sogar. In der Beziehung Mehrwert (die Modellierungsnotation bietet Konstrukt an, dass von Evaluierungsframework nicht unterstützt wird) bietet das

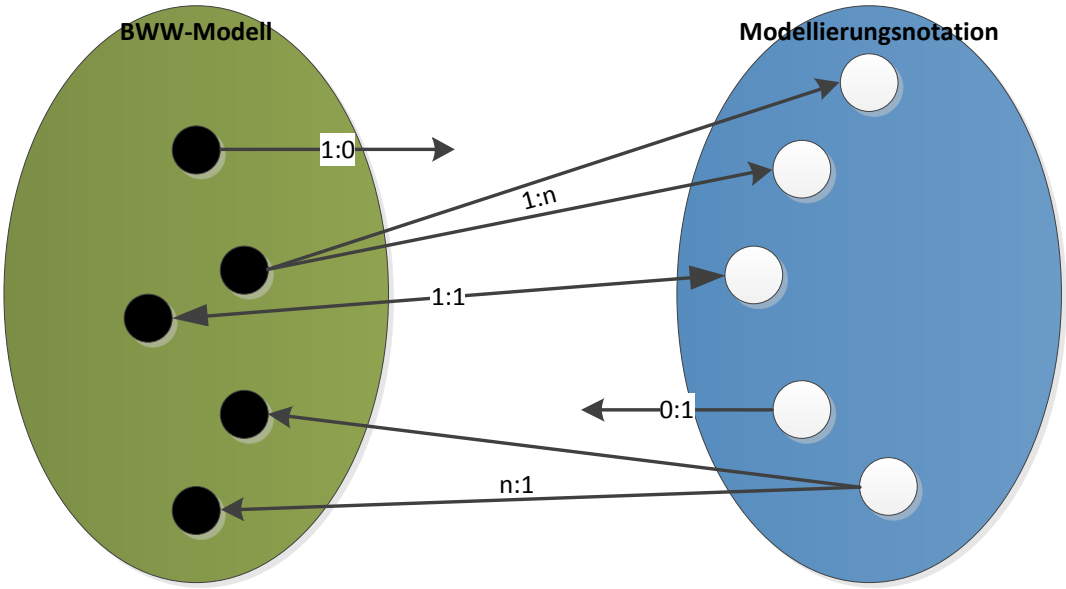



**Bild 3.5:** Möglichkeiten des Mappings von Framework- auf Notationskonstrukte nach [RRK07]

BWW-Modell Informationsmehrwert, durch die Bereitstellung von 1:n Abbildungsverhältnissen.

## 3.6 Zusammenfassung

In diesem Kapitel der Arbeit wurden kurz die Grundlagen der Prozessmodellierung erläutert (3.1) und in das Konzept des Tokens eingeführt (3.1.1), was zu Beschreibung des Verhaltens von Prozessmodellen benutzt wird. Genauso wurde gezeigt, was der Unterschied zwischen Kontrollfluss und Datenfluss ist (3.1.2). Darauf aufbauend wurden zwei Paradigmen der Prozessmodellierung vorgestellt (3.4). Eine Kurzdarstellung von $\alpha$-Flow wurde gegeben und anschließend auch eine Darstellung von zwei interessanten Ansätzen. Die Begrifflichkeiten des Kapitels werden in den folgenden Kapiteln immer wieder aufgegriffen. Ebenso werden in den folgenden Kapiteln die Evaluierungsframeworks auftauchen, die in diesem Kapitel vorgestellt wurden, um die Gegenüberstellung bzw. die Analyse der verschiedenen Prozessmodellierungsnotationen zu untermauern (3.5.1) und 3.5.2). In dieser Arbeit werden beide Frameworks benutzt. Sie treffen beide eine Aussage

bezüglich der Aussagekraft von Modellierungsnotationen. Da in Kapitel 8 jedoch nur die Äquivalenzperspektive betrachtet wird, widersprechen sich die beiden Rahmenwerke nicht.

# 4 Verwandte Arbeiten

In diesem Kapitel werden Arbeiten vorgestellt, die in ihrer Aufgabenstellung ähnlich zu der vorliegenden sind. Dabei wird vor allem auf die Gegenüberstellung und Evaluation von Prozessmodellierungsnotationen eingegangen.

## Recker at al.: Business Process Modeling- A Comparative Analysis

In [RRIG09] wird eine vergleichende Analyse vollzogen, die das BWW-Model benutzt. Hier werden grafischen Prozessmodellierungsnotationen gegenüber gestellt. Die Analyse in dieser Arbeit hat dabei das Ziel, festzustellen, wie hoch die ontologische Vollständigkeit bzw. die ontologische Klarheit der verschiedenen Notationen ist. Dabei wurden Sprachen beginnend von Petri Netzen aus dem Jahr 1962 über EPK's aus dem Jahr 1992 bis hin zu BPMN 1.0 aus dem Jahr 2004 analysiert und auf das BWW-Modell gemappt. Dabei wurde ein spezielles Verfahren[1] zur Mapping-Bildung angewandt, um eine hohe Objektivität zu gewährleisten. Es wurden Mappings von 1:1, 1:m und n:1 durchgeführt (siehe Bild 3.5). Das 1:1 Mapping dient dazu die ontologische Vollständigkeit zu testen. Die übrigen Abbildungsverhältnisse geben Aufschluss über die ontologische Klarheit der Ansätze. Es wird gezeigt, dass Petri-Netze trotz ihres Alters eine hohe ontologische Vollständigkeit aufweisen, trotz der wenigen Modellierungskonstrukte, die angeboten werden. Dem gegenüber steht BPMN, als neuester Vertreter in der Analyse, auch mit einer hohen ontologischen Vollständigkeit, aber auch mit 38 verschiedenen Konstrukten, was eine niedrige ontologische Klarheit zur Folge hat. Die niedrigste ontologische Redundanz, also die wenigsten redundanten Modellierungskonstrukte bieten die EPK's[2]. Da die Petri-Netze nur zwei Konstrukte zur Modellierung anbieten, haben sie auch

1 Es wurden mehrere Gruppen von Forschungsteilnehmern gebildet, die unabhängig von einander die Abbildung vollzogen haben.

2 Dies ist die 1:n Abbildung

keine Konstrukte, die mehr Modellierungsanforderungen erfüllen als das BWW-Modell erfordert[1], denn Petri-Netze bieten nur Stellen und Transitionen, welche keinen Mehrwert gegenüber den Konstrukten des BWW-Modells bieten. Dem gegenüber steht die höchste Überladung[2] von Konstrukten.
Im Allgemeinen wurde festgestellt, dass die Modellierungstechniken, die im Business Umfeld eingesetzt werden, weniger ontologisch vollständig sind, weil ihre Syntax restriktiver ist. Als Ergebnis der Analyse wurde überprüft ob das BWW-Modell um neue Konstrukte erweitert werden müsste[3] und ob man mit Hilfe dieser Analyse die Schwachstellen einer Notation begutachten kann.

## Recker et al.: Measuring method complexity: UML versus BPMN

Die Arbeit [RZMS+09] beschäftigt sich mit der Gegenüberstellung der beiden Modellierungsnotationen UML und BPMN in Hinsicht auf die Komplexität der jeweiligen Sprache. Zur Messung und Beurteilung der Komplexität wird ein Komplexitätsmaß benutzt. Anschließend wird die theoretische Komplexität (= Komplexität, der in der Spezifikation enthaltenen Konstrukte) und die konkrete Komplexität (= Komplexität, der wirklich benutzten Konstrukte) der beiden Modellierungsnotationen untersucht. Als Ergebnis wird herausgestellt, dass bei beiden die theoretische Komplexität um ein vielfaches höher ist, als die konkrete Komplexität, wobei BPMN, durch die Fülle von Konstrukten nochmals viel komplexer ist als UML.

## White: Process Modeling Notations and Workflow Patterns

In [Whi04b] werden ebenfalls BPMN und UML Aktivitätsdiagramme verglichen. Diesmal aber in Bezug auf deren Eignung hinsichtlich des Workflow Pattern Frameworks. Für diese Untersuchung wurden die beiden Notationen auf das Erfüllen der Anforderungen der einzelnen Muster überprüft. Als Ergebnis der Untersuchung wird aufgeführt, das beide Modellierungssprachen alle hier untersuchten Muster adäquat unterstützen, bis auf

1 Dies ist die 1:n Abbildung
2 Dies ist die n:1 Abbildung
3 Es wurde zum Beispiel festgestellt, dass Kontrollflusselemete (z.B. generell Kanten) nicht unterstützt werden und für diese deshalb ein eigenes Metaschema bereitgestellt werden muss.

das Interleaved Parallel Pattern, das von UML AD's nicht ausreichend dargestellt werden kann. Es wird auch angegeben, dass sich beide Sprachen von den grafischen Konstrukten, bis auf die Modellierung von Nebenläufigkeit kaum unterscheiden, deswegen zeigen sie auch grundsätzlich ähnliche Ansätze die Muster zu lösen. Der Grund für diese Ähnlichkeit wird darauf zurückgeführt, dass sowohl UML AD's als auch BPMN für den gleichen Zweck entworfen worden sind: Zum grafischen Modellieren von Geschäftsprozessen.

### List und Korherr: An Evaluation of Cenceptual Business Process Modelling Languages

Einen weiterer Ansatz der vergleichbar zu dieser Arbeit ist, stellt [LK06] dar. Hier werden Business Process Definition Metamodel (BPDM), BPMN, EPK, Integrated DEFinition for Process Description Capture Method (IDEF3), Petri-Netze und UML AD's verglichen. Dazu werden hier fünf verschiedene Metamodellperspektiven aufgestellt:

- Funktionale Perspektive (Welche Elemente führen die Prozesse aus?)
- Organisatorische Perspektive (Wo und von wem wird der Prozess ausgeführt?)
- Information Perspektive (Informationen, die am Prozess beteiligt sind)
- Geschäftsprozess Kontext Perspektive (Welche Anforderungen im Business Umfeld werden an die Prozesse gestellt?)
- Verhaltens Perspektive (Wie wird der Prozess ausgeführt?)

Zur Analyse der Prozessmodellierungsnotationen, die Gegenstand der Untersuchung in der Arbeit sind, wurden die Metamodelle, die identifiziert wurden, mit den Konstrukten der Notationen verglichen. Dabei wurde der Schwerpunkt auf die Metamodelle der Notationen gelegt und nicht auf einzelne Modellierungskonstrukte. Als Ergebnis wurde festgehalten, dass die funktionale und die Verhaltensperspektive gut dargestellt werden können, während die organisatorische und die Informationsperspektive nur teilweise unterstützt werden. Die Geschäftsprozesskontext Perspektive wird von keiner Sprache explizit supported.

## Rosemann et al.: A Study of the Evolution of the Representational Capabilities of Process Modeling Grammars

[RRIG06] bietet einen Vergleich der Ausdrucksmächtigkeit der verschiedenen Notationsarten an und wie sich diese im Verlauf der Zeit verändert haben. Als Grundsatzfrage wird aufgestellt, ob sich die Ausdrucksmächtigkeit im Laufe der Zeit verbessert hat, man also die Notationen immer besser gestaltet hat oder aber auch frühe Ansätze schon ausreichend waren. Für diese Untersuchung wurde das BWW-Modell benutzt und die verschiedenen Notationsarten dagegen gemappt (vergleiche Bild 3.5). Als älteste Notation finden die Petri-Netze (1962) Verwendung, als jüngste Modellierungsnotation wird BPMN (2004) benutzt. In der Untersuchung wird quantitativ gezählt, wie viele Konstrukte des BWW-Modells 1:1 auf die Konstrukte der Sprachen abgebildet werden können (siehe Bild 3.5). Das Ergebnis dieser Arbeit ist, dass Petri-Netze, trotz ihres Alters viele Konstrukte anbieten, aber trotzdem eine Tendenz zur Verbesserung der Ausdrucksmächtigkeit der Sprachen zu beobachten ist. Ein weiterer Befund der Arbeit ist, dass die einzelnen Konstrukte, die unterstützt werden entweder von vielen Notationen unterstützt werden oder von keiner unterstützt werden. Dadurch zeigt sich, dass einige BWW-Konstrukte wichtiger sind, andere weniger wichtig.

In [AWG05] werden 10 verschiedene Modellierungsnotationen bzw. Prozessunterstützungssysteme untersucht und ihre Eignung hinsichtlich der Workflow Patterns untersucht.

# 5 Ziele der Gegenüberstellung

Die Motivation für den Überblick und die Gegenüberstellung in Kapitel 6 und der anschließenden Modellbildung in Kapitel 7 ist, dass durch die Analyse der verschiedenen bewährten Prozessmodellierungsnotationen, deren Aussagekraft bzw. Stärken und Schwächen aufgezeigt werden. Dabei wird der Schwerpunkt auf bestimmte Konstrukte gelegt, auf die während der Betrachtung besonders eingegangen werden soll: Die zeitliche Abfolge von Aktivitäten, Entscheidungen, Datendarstellung, Kontrollfluss, Beendigung von Prozessen, Zustandsdarstellungsmöglichkeiten, Subprozessmodellierung und Möglichkeit zur Änderung von Prozessen zur Laufzeit. Damit erfolgt eine Abgrenzung zum Aufzeigen von Rollen und organisatorischer Modellierung, genauso wie Modellierung von Ausnahmen.
Mit Hilfe des Überblicks von Kapitel 6, wird im Kapitel 7 ein Modell zur Beschreibung der oben aufgezählten Konstrukte gegeben. Damit soll gezeigt werden, dass $\alpha$-Flow in der Lage ist, diese Konstrukte rein auf Grund von Rückwärtsverkettung zu modellieren. Das Modell soll demnach die gleiche Aussagemächtigkeit haben, wie die in Kapitel 6 betrachteten Ansätze. Das Modell soll demnach mehrere Konzepte bieten, die die Handhabung der betrachteten Konstrukte allein durch Datenabhängigkeiten ermöglichen soll. Diese Konzepte sollen auch zusammenarbeiten und einen kompletten und in sich verankerten Mechanismus zum Beschreiben eines Prozesses bilden. Es soll dabei in erster Linie auf Prozesse aus dem medizinischen Umfeld eingegangen werden. Dennoch soll das Modell Allgemeingültigkeit besitzen und auf andere Bedarfsfelder angewendet werden können.
In dem darauf folgenden Kapitel 8 werden diese Erkenntnisse aus den Kapiteln 6 und 7 mit Hilfe der Evaluierungsframeworks analysiert, um die jeweiligen Stärken und Schwächen zu formalisieren. Es müssen demnach sowohl die Abbildungen von $\alpha$-Flow, als auch die Abbildungen der anderen betrachteten Ansätze auf die beiden Evaluierungsframeworks veranschaulicht und analysiert werden, um anschließend die grundsätzlichen Unterschiede bzw. Gemeinsamkeiten in der Darstellungsmächtigkeit herauszustellen. Dabei wird auf die Notwendigkeit einiger Konstrukte für den $\alpha$-Flow

Ansatzes eingegangen.

# 6 Gegenüberstellung und Überblick der einzelnen Notationsarten

In diesem Kapitel werden die verschiedenen Prozessmodellierungsnotationen vorgestellt und anschließend in Hinblick auf ausgewählte Charakteristiken untersucht und gegenübergestellt. Diese Gegenüberstellung soll durch einen Beispielprozess illustriert werden, der in Sektion 6.1 vorgestellt werden soll. Diesem Abschnitt schließt sich die Darstellung der BPMN in Abschnitt 6.2 und im Weiteren UML (6.3), Petri-Netze (inklusive der YAWL) ( 6.4), EPK (6.5), PHILharmonicFlows als Beispiel objekt-orientierter Prozessmodellierung (6.6), ACM (6.7) an. Zum Abschluss des Kapitels wird ein zusammenfassendes Resümee der Darstellung in Abschnitt 6.8 gegeben.

## 6.1 Definition des Beispielprozesses

Der Beispielprozess entstammt dem Artikel [NL10]. Dabei wird ein typischer Ablauf einer Brustkrebs Untersuchung geschildert:
Die Patientin besucht die Praxis ihres niedergelassenen Gynäkologen, um dort eine klinische Untersuchung und eine Sonographie durchführen zu lassen. Aus den Ergebnissen entscheidet der Gynäkologe, ob die Befunde verdächtig sind. Wenn er sich für eine weitere Behandlung entscheidet, wird die Patientin zum Radiologen überwiesen, um dort eine Mammographie machen zu lassen, die Ergebnisse werden wieder zum Gynäkologen der Patientin geschickt, um dort den Breast Imaging-Reporting and Data System (BI-RADS) zu überprüfen. Falls dieser größer gleich vier ist, wird die Patientin zum Gynäkologen an einem Krankenhaus überwiesen, um dort eine Biopsie durchführen zu lassen. Die daraus entstandene Gewebeprobe wird zum Pathologen geschickt, um eine Histologie durchzuführen. Zeitgleich mit der Histologie fertigt der Gynäkologe im Krankenhaus bereits seinen Bericht an. Die Berichte der Histologie und der Biopsie werden dann wieder zum niedergelassenen Gynäkologen der Patientin geschickt, um festzustellen, ob Krebs vorliegt oder nicht. Danach wird die Patientin zu einem abschließenden Gespräch

zum Gynäkologen eingeladen.

## 6.2 Business Process Modelling Notation

Die Business Process Modelling Notation (BPMN) wurde im Mai 2004 in Version 1.0 von der Business Process Management Initiative (BPMI) Notation Working Group veröffentlicht [Whi04a], [DAH05]. Mittlerweile liegt die BPMN in Version 2.0 vor und wird von der Object Management Group (OMG) spezifiziert, einem Konsortium, das auch die UML spezifiziert [OMG11a]. Das Hauptziel der Sprache ist, dass sowohl Business Usern als auch IT-Fachkräften eine Sprache zur Beschreibung von Prozessen zur Verfügung gestellt werden kann, die beide Benutzergruppen in gleicher Weise verstehen und anwenden können [Whi04a]. Deshalb hat sich in weiten Teilen der Wirtschaft die BPMN als grafische Notationsart von Prozessen durchgesetzt [Bar06]. Eine Intention der BPMN ist weiterhin das einfache Überführen nach BPEL4WS zur Prozessausführung [LK06], [ODTHVDA06].

### Grundsätzliche Modellierungskonstrukte

In BPMN werden 50 Konstrukte plus Attribute definiert, die sich in fünf Kategorien einteilen lassen [LK06],[OMG11a]:

- Flussobjekte (z.B. Ereignisse, Aktivitäten und Gateways)
- Daten (z.B. Datenobjekte)
- Verbindungsobjekte (z.B. Sequenzfluss und Nachrichtenfluss)
- Swimlanes/Pools
- Artefakte (z.b. Annotationen)

Von diesen Konstrukten sind jedoch nur im Schnitt neun pro Modell im Einsatz [ZMR08]. Zum Beispiel existieren von einem Ereignis mehrere Fassungen, die in bestimmten Situationen zum Einsatz kommen. Dies ist besonders erwähnenswert, weil trotz der Fülle an Modellierungsmöglichkeiten nur neun Konstrukte für die meisten Modelle genügen. Dies ist ein Zeichen dafür, dass es viele Konstrukte gibt, die redundant sind bzw. die sehr spezifisch auf einen bestimmten Einsatzzweck ausgerichtet sind.

### Spezifische Betrachtung der Modellierungsmöglichkeiten - Kontrollfluss, Daten, Nebenläufigkeit und Entscheidungen

Der Schwerpunkt in der Modellierung von Prozessen liegt bei BPMN auf dem Kontrollfluss und z.B. der Datenfluss wird nur nebensächlich behandelt und betrachtet [FRH10]. Das Grundmodellierungskonstrukt stellt die Aktivität dar (Bild 6.1, hier wird der Prozess mit Hilfe von aufeinander folgenden Aktivitäten modelliert). BPMN lässt sich in das aktivitätsbasierte Paradigma einordnen. Dennoch wird der Wichtigkeit der Daten seit der Version 2.0 höheren Wert zugesprochen [FR10], da Daten nicht mehr in die Kategorie der Artefakte gezählt werden, sondern ihre eigene Konstruktkategorie erhalten haben. Aus der Schwerpunktsetzung auf den Kontrollfluss leitet sich auch ab, dass Daten keinen direkten Einfluss auf den Prozessfortschritt haben. Dies leitet sich aus der Tatsache ab, dass BPMN das Tokenkonzept (siehe Abschnitt 3.1.1) benutzt. Datenobjekte werden demzufolge über Assoziationen mit Aktivitäten verknüpft und über Assoziationen können keine Token fließen [OMG11a]. Die Token fließen demnach über Kontrollflusskanten von Aktivität zu Aktivität. So wird Prozessfortschritt erreicht, der komplett auf Vorwärtsverkettung beruht. Daten lassen sich darstellen als Datenobjekte, Dateninput bzw. -ouput des kompletten Prozesses, als Listendatenobjekt (Ansammlung von mehreren Datenobjekten), als Datenspeicher oder als Nachrichten zur Kommunikation zwischen zwei Prozessteilnehmern [FRH10]. Datenfluss ist aber nicht modellierbar.
BPMN bietet Möglichkeiten Nebenläufigkeit zu modellieren, dazu werden Gateways benutzt, die unterschiedliche Semantiken aufweisen. Es gibt AND-,OR-,XOR- und Complex-Gateways für diesen Zweck. Mit deren Hilfe lässt sich jedoch nicht nur Nebenläufigkeit (durch Einsatzes als Split und Joins der Gateways), sondern auch komplexe Entscheidungssituationen modellieren [WAD+06],[FR10].

### Spezifische Betrachtung der Modellierungsmöglichkeiten - Zustände, Subprozesse, Beendigung von Prozessen und Laufzeitänderungen

Zustände des Gesamtprozesses lassen sich in BPMN durch sogenannte Ereignisse modellieren. Diese Ereignisse können eingetretene, ausgelöste oder Zwischenereignisse sein. Durch sie wird ausgedrückt, dass dieses Ereignis eintreten muss bzw. ausgelöst wird, wenn ein bestimmter Prozessfortschritt erreicht wurde. Das Konzept der eingetretenen Ereignisse bzw. Zwischenereignis erfüllt die Definition des Zustandes, nachdem das System (Prozess) sich in einer bestimmten Situation (Ereignis) zu einem gewissen

Zeitpunkt befinden muss (Prozessfortschritt) [CS06]. Es werden eine Vielzahl verschiedener Variationen von Ereignissen angeboten [FRH10]. Zustände von Aktivitäten bzw. Datenobjekten werden durch eckige Klammern im entsprechenden Konstrukt dargestellt (z.B. [accepted])[OMG11a].
Um Subprozesse zu modellieren, wird in BPMN das Konstrukt der Teilprozesse angeboten. Man kann damit einen Subprozess entweder in einem eigenen Prozessdiagramm modellieren oder in den Oberprozess integrieren[1] [FRH10]. Die Subprozesse basieren ihrerseits auch wieder auf dem Tokenkonzept und verhalten sich wie der Oberprozess.
Zur Beendigung von Prozessen gibt es grundsätzlich zwei Möglichkeiten: Die Beendigung des kompletten Prozesses an einer Stelle oder die Beendigung eines Zweiges des Prozesses. Dafür gibt es das Terminierung-Endereignis bzw. das Ende-Blanko-Ereignis. Das Terminierung-Endereignis konsumiert alle Token, die sich noch im Prozess befinden und beendet den Prozess auf der Stelle. Das Ende-Blanko-Ereignis konsumiert nur einen Token und beeinflusst die anderen Zweige des Prozesses in keiner Weise [WAD+06], [FRH10]. Änderungen an dem Prozessmodell sind zur Laufzeit nicht mehr möglich [FR10].

### Probleme der Modellierungskonstrukte

Nicht eindeutig definiert ist das Verhalten des OR-Joins, der entweder lokal (lokales Verhalten bedeutet hier, dass an der Stelle des OR-Joins, das Gateway nicht weiß wie viele Tokens ankommen werden) oder nicht-lokal (nicht-lokales Verhalten bedeutet hier, dass es an der Stelle des OR-Joins bekannt ist wie viele Tokens ankommen werden) agieren kann. Die Semantik wird als nicht-lokal definiert, was zu unübersichtlichen Modellen führen kann [FR10]. Im Allgemeinen ist in der Spezifikation nicht formal beschrieben, wie die einzelnen Konstrukte wirken, da sie in natürlichsprachlichen Text geschrieben ist [OMG11a] und es dadurch zu Uneindeutigkeiten kommt.

### Modellierung des Beispielprozesses

In Abbildung 6.1 ist der Beispielprozess in BPMN realisiert. In dieser Abbildung sieht man deutlich, dass das Hauptmodellierungskonstrukt von BPMN die Aktivität ist (z.B. händische Untersuchung und Sonographie). Sie bilden die Grundstruktur des

1 aufgeklappt oder als zugeklappte Aktivität mit spezieller Markierung

Prozesses ab und legen Reihenfolge der Abarbeitung fest. Entscheidungen sind im Beispielprozess durch die XOR-Gateways dargestellt. Z.B. muss entschieden werden, ob der BI-RADS größer gleich 4 ist oder kleiner. Die Daten sind nur durch Assoziationen an die Kontrollflusskanten angebunden. Dies verdeutlicht, dass sie zwar abgebildet werden müssen, da sie ein Werkzeug zur Prozessabarbeitung darstellen, aber sie keinen Einfluss auf den Prozessverlauf ausüben. Parallelität wird erzeugt durch den AND-Split nach der Biopsie. Wie man auch an diesen Beispielprozess sieht, bietet BPMN eine Vielzahl von Modellierungskonstrukte an. Das Ende des Prozesses wird mit den Terminierungsendereignissen modelliert, wie es hier entweder nach vollständiger Abarbeitung des Prozesses oder einer negative Entscheidung modelliert ist.

### Eignung anhand der Evaluierungsframeworks

Zur Eignung von BPMN zur Prozessmodellierung gibt es Analysen mit den Workflow Patterns [WAD+06] und des BWW-Modells [RRIG09]. Dabei erfüllt BPMN die einfachen Muster wie Sequenz komplett, die erweiterten Synchronisierungsmuster werden auch unterstützt, genauso wie Strukturmuster. Das Modellieren von mehrfachen Instanzen wird unterstützt, bis auf die Möglichkeit während der Laufzeit die Anzahl der Instanzen zu verändern. Die zustandsbasierten Muster werden teilweise unterstützt, denn BPMN fehlt die Möglichkeit Meilensteine und Interleaved nebenläufiges Routing zu modellieren. Die Abbruchmuster werden vollständig unterstützt.
Die Abbildung auf das BWW-Modell zeigt, dass BPMN Gegenstände und Eigenschaften von Gegenständen vollständig modellieren kann. Dafür besteht ein großes Defizit (0/7 Konstrukten) beim Modellieren von Zuständen von Gegenständen. Ereignisse und Transformationen von Gegenständen werden sehr gut unterstützt (9/11 Konstrukten). In der Kategorie der Systeme die aus Gegenständen gebildet werden, schneidet BPMN dagegen gut ab (6/7 Konstrukten). Insgesamt wird BPMN mit einer Rate von 65,5% Vollständigkeit eine gute Eignung attestiert [RRIG09].

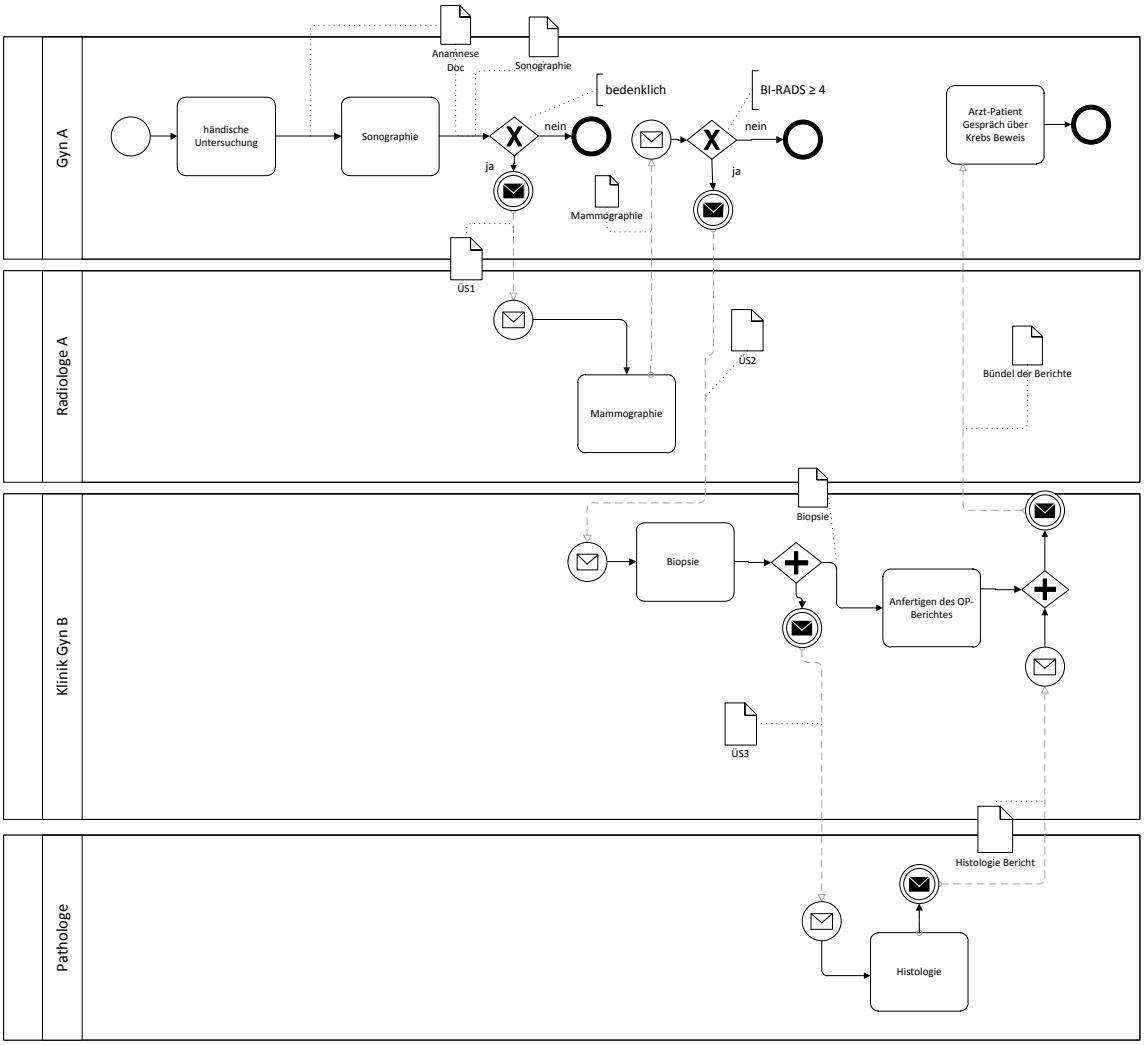



**Bild 6.1:** Umsetzung des Beispielprozesses mit BPMN

## 6.3 Unified Modelling Language (Aktivitätsdiagramme)

Die Unified Modelling Language UML Version 1.0 wurde im Januar 1997 eingereicht [HK99]. Im Jahr 2003 wurde die Version 2.0 verabschiedet [Boc03a]. Die UML versteht sich als eine Modellierungssprache zum Spezifizieren, Konstruieren, Visualisieren und Dokumentieren von Softwaresystemen [HK99]. Sie stellt dabei einen objektorientierten Ansatz dar [OHS02]. Dabei kommen verschiedene Diagrammarten zu Einsatz wie zum

Beispiel [Hol09]: Klassendiagramm, Aktivitätsdiagramm (AD), Sequenzdiagramm und Use Case Diagramm.

### Allgemeines zum Aktivitätsdiagramm

Da hierbei das Aktivitätsdiagramm zur Modellierung von Verhalten herangezogen wird ist es die typischste Darstellungsform von Prozessen mit der UML [Hol09], [HK99]. Deshalb beschränken wir uns auch auf das AD. Diese sind auch abbildbar auf die BPEL4WS [Gar03].
Das AD basierte in der Version 1.4 noch auf den StateCharts [LP99],[Boc03a]. Die Version 2.0 basiert nunmehr auf dem Tokenkonzept und damit auf dem Prinzip der Petri-Netze (siehe Abschnitt 6.4) [Boc03a]. Eine Abbildung auf Petri-Netze ist nach [Stö04], [Stö05] möglich, um die ADs zu verifizieren und zu analysieren. Die größten Unterschiede bei dieser Umstellung sind nach Bock [Boc03a]:

- nun basierend auf das Tokenkonzept
- echte Nebenläufigkeit war in StateCharts schwer möglich

### Grundsätzliche Modellierungskonstrukte

Das Aktivitätsdiagramm lässt sich auch in das aktivitätsbasierte Paradigma der Prozessmodellierung einordnen. Die Aktivität steht im Mittelpunkt der Prozessmodellierung [LK06]. Die Abfolge der Aktivitäten wird durch das Tokenkonzept beschrieben, die entweder über Objektflusskanten oder über Kontrollflusskanten fließen. Dadurch werden die einzelnen Aktivitäten vorwärtsverkettet. Es stehen folgende Konstrukte zur Modellierung zur Verfügung [Boc03b]:

- Aktivität
- Objekt (Daten)
- Kontrollknoten
- Fork/Join Balken
- Start- und Endzustände

### Spezifische Betrachtung der Modellierungsmöglichkeiten - Kontrollfluss, Daten, Nebenläufigkeit und Entscheidungen

Der Kontrollfluss wird dargestellt als eine gerichtete Kante zwischen zwei Aktivitäten über die, via Tokenkonzept, ein Token fließt und die einzelnen Aktivitäten zur Ausfüh-

rung triggert [Boc03b]. Im UML AD ist es des Weiteren möglich, Datenfluss explizit darzustellen. Dabei werden die Daten auch in das Tokenkonzept einbezogen, indem ein Objekt als Objekttoken über Objektflusskanten fließt. Zur Modellierung von Daten werden verschieden Möglichkeiten angegeben [Boc04]:

- Darstellung mit Pins am Ende der gerichteten Kante zwischen zwei Aktivitäten
- Darstellung über Objekt zwischen zwei Aktivitäten verbunden mit gerichteten Kanten
- Darstellung als Pin der an der gerichteten Kante zwischen zwei Aktivitäten klebt

Die Dualität von Kontrolltoken auf der einen und Objekttoken auf der anderen Seite wird gelöst, indem man dem Token einen theoretischen Charakter zuschreibt. Dadurch ist es unwichtig, ob ein Token ein Kontrolltoken oder ein Objekttoken ist, um Prozessfortschritt zu erreichen. Wenn z.B. bei einem Join sowohl Objekttoken und Kontrolltoken aufeinander treffen, werden die Kontrolltoken konsumiert und zerstört und nur die Objekttoken weitergegeben, weil die Kontrolltoken redundant sind [OMG11b].
Nebenläufigkeit wird im UML AD durch die Split Balken erreicht, diese teilen den Prozessverlauf auf und vervielfältigen Kontrolltoken bzw. legen Referenzkopien von Objekttoken an [OMG11b], [Boc03b]. Durch den Join Balken werden mehrere nebenläufige Zweige des Prozesses wieder synchronisiert. Komplexeres Verhalten des Joins müssen durch Annotationen bzw. Guards beschrieben werden [OMG11b], [DAH05].
Als Entscheidungsinstrument steht der Kontrollknoten zur Verfügung. Mit Hilfe dieses Knotens, lassen sich komplexe Entscheidungen durch Guards in eckigen Klammern modellieren [DAH05]. Es wird jedoch keine Nebenläufigkeit erzeugt, weshalb nur ein Token auf eine ausgehende Kante fließen darf [OMG11b]. Die eingehenden Kanten an diesen Knoten sind entweder Kontrollflusskanten oder Objektflusskanten, sonst ist das Verhalten nicht definiert [OMG11b]. Das Vereinigen mehrerer Prozesszweige durch einen Entscheidungsknoten hat die Semantik, dass alle eingehenden Token auch auf die ausgehende Kante weitergegeben wird [OMG11b]. Dieses Vereinigen der Prozesszweige ist nicht zum synchronisieren von parallelen Prozesszweigen gedacht.

#### Spezifische Betrachtung der Modellierungsmöglichkeiten - Zustände, Subprozesse, Beendigung von Prozessen und Laufzeitänderungen

Zustände des Gesamtprozesses sind in UML AD's nicht modellierbar. Es existiert allein die Notwendigkeit den Beginn und das Ende eines Prozesses zu kennzeichnen, dazu

stehen das Konstrukt des Startzustandes als auch das Konstrukt des Aktivitätendknoten und des Flussendknnoten zur Verfügung [RVDATHW06]. Mit dem Aktivitätsendknoten lässt die gesamte Aktivität beenden mit dem Flussendknoten nur der jeweilige Zweig der Aktivität [DAH05].
Subprozesse lassen sich durch Aktivitätsaufrufe lösen. Dadurch lassen sich in Aktivitäten andere Aktivitäten aufrufen. Diese Aktivitäten stellen auch wieder einen Start- und Endknoten zur Verfügung. Dadurch lässt sich die Komplexität vermindern, aber dennoch wird eine komplexe Modellierung ermöglicht [DAH05], [Boc05]. Änderungen am Prozessmodell sind nicht durchführbar.

### Probleme der Modellierungskonstrukte

Probleme der UML AD's sind wiederum die fehlende formale Semantik, weil auch diese in der Spezifikation nur in natürlichsprachlichen Text angegeben wird [OMG11b], [OMG11c], [SF07] und es dadurch zu Uneindeutigkeiten kommen kann. Außerdem wird bemängelt, dass die UML zu viele Konstrukte bereitstellt [SF07]. Diese liegt nur minimal unter der Komplexität der BPMN [RZMS+09], [IRGR08]. Beides hat zur Folge, dass der Benutzer sich mit der komplexen Semantik vertraut machen muss, um das Modell vollständig zu verstehen. Dadurch wird die Benutzung und das Verständnis erschwert.

### Modellierung des Beispielprozesses

In Abbildung 6.2 wird die Modellierung des Beispielprozesses mit UML AD's angegeben. Man sieht, dass die Aktivitäten im Beispielprozess das Hauptmodellierungskonstrukt darstellen, denn sie determinieren die Reihenfolge und die Struktur des Prozesses. Sie können durch den Kontrollfluss oder durch den Fluss von Daten verbunden sein. Diese Daten sind zum Beispiel die Überweisungsscheine. Hier wird der Fluss dieser Daten explizit modelliert. Und wie man anhand des Mammographie Berichts erkennen kann, haben sie direkten Einfluss auf den Prozessverlauf. Die Modellierung von Entscheidungen wird hier durch die Entscheidungsknoten getroffen. Sie unterliegen einer XOR-Semantik. Im Beispielprozess kann zum Beispiel der BI-RADS entweder größer gleich vier sein oder kleiner. Die Nebenläufigkeit wird erzeugt durch die Split Balken. Das hat den Vorteil gegenüber zum Beispiel BPMN, dass keine Verwechslung mit Entscheidungen auftritt. Man sieht im Beispiel, dass nach der Biopsie die Parallelität erzeugt wird. Am

Beispiel gut zu sehen ist der unterschiedliche Einsatz der Aktivitätsendknoten und der Flussendknoten.

### Eignung anhand der Evaluierungsframeworks

Die Eignung von UML AD's zur Prozessmodellierung wird auf Grund von den Workflow Patterns [RVDATHW06] und des BWW-Modells [OHS02] beschrieben. Die Analyse der Eingnung durch die Worklow Pattern erfolgt in [RVDATHW06], die Analyse der Eignung durch das BWW-Modells erfolgt in [OHS02]. UML AD's unterstützen alle einfachen Muster, wie Sequenz. Bei den erweiterten synchronisierenden Mustern unterstützen UML AD's nur das synchronisierende Mischen[1] nicht. Die Strukturmuster dagegen werden vollständig unterstützt. Genauso wie BPMN muss den UML AD's die Anzahl der Instanzen vor der Laufzeit bekannt sein, deswegen werden nicht alle mehrfach Instanzmuster unterstützt. Die Unterstützung für Meilensteinge und Interleaved nebenläufiges Routing ist dagegen auch nicht gegeben. Die Abbruchmuster werden vollständig unterstützt. Die Abbildung auf das BWW-Modell erlaubt das Modellieren von Gegenständen und deren Eigenschaften vollständig. Es ist auch möglich eine Abbildung auf die Kategorie der Ereignisse und Transformationen von Gegenständen vollständig zu definieren. Die zustandsbasierten Konstrukte können abgebildet werden, bis auf die Definition von Zustandsräumen. Eine Systemstruktur von Gegenständen kann in UML nur teilweise abgebildet werden (keine Subsysteme, Systemumgebungen, Systemstrukturen, keine Levelstrukturen).

1 Die Möglichkeit zwei oder mehrere eingehende Kanten zu synchronisieren

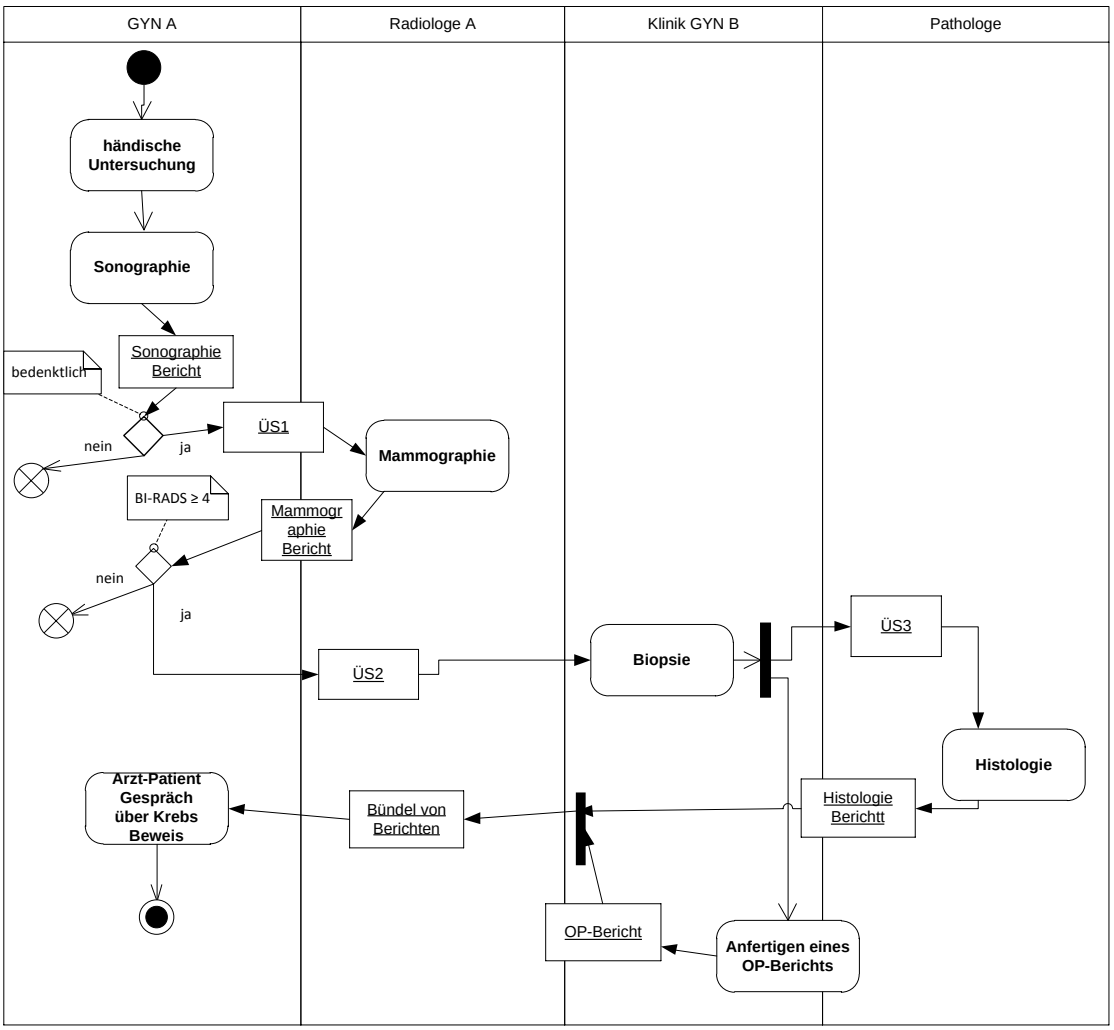



**Bild 6.2:** Umsetzung des Beispielprozesses mit UML AD

## 6.4 Petri Netze (inklusive Yet Another Workflow Language)

Petri Netze wurden 1962 von Carl Adam Petri am Institut für Instrumentelle Mathematik der Universität Bonn entwickelt. Er versuchte eine beliebig oft erweiterbare Rechnerarchitektur zu beschreiben. In seiner Dissertation mit dem Thema „Kommunikation mit Automaten“ zeigt er die Konstruierbarkeit dieser Rechnerarchitektur mit Hilfe lokal beschränkter Komponentenkommunikation. Diese lokal beschränkten Aktionen sind wiederum die Grundlage für Petri Netze [Rei10], [RRIG09]. Petri Netze dienen als mathe-

matische Theorie, als grafische Notationsart und als formale Sprache [DAH05]. Mithilfe von Petri Netzen lassen sich Prozesse modellieren. Nach [Jör03] können Petri Netze auch im Gesundheitswesen eingesetzt werden. Es werden drei Gründe angegeben, weshalb Petri Netze für die Prozessmodellierung herangezogen werden [Aal98b], [Aal98a]:

- Formale Semantik für die grafische Notation
- Zustandsbasiert und nicht ereignisbasiert
- Vielzahl von Analysetechniken

Das hat die Vorteile, dass Petri Netze eindeutig und Tool unabhängig sind. Außerdem wird umweltbasiertes Triggern erlaubt[1]. Deadlocks und Livelocks können entdeckt werden. Korrektheit und Konsistenz können verifiziert und Performance analysiert werden [Aal98b], [Aal98a].

#### Grundsätzliche Modellierungskonstrukte

Die Prozesse werden durch den Einsatz von drei verschiedenen Konstrukten modelliert [Rei10], [PW08]:

- Stellen
- Transitionen
- gerichtete Kanten

Durch die geringe Anzahl der Konstrukte ist die Modellierung uneindeutig, da so zum Beispiel Transitionen viele verschiedene Sachverhalte, wie zum Beispiel Nebenläufigkeit und Entscheidungen, modellieren [RRIG09]. Petri Netze basieren auf dem Tokenkonzept, das hier explizit durch das Zeichnen eines Punktes dargestellt wird. So fließt hier das Token über die Kanten von Stelle zu Transition und von Transition zu Stelle. Stelle und Transition müssen sich in Petri Netzen immer abwechseln. Petri Netze stellen demnach einen bipartiten Graphen dar. Wenn ein Token in einer Stelle ist, bedeutet das, dass die Transition „feuern“ kann aber nicht muss und ist damit gleichzusetzen mit dem Erlauben einer Aktivität im Prozess, aber noch nicht dem Ausführen [Aal98a].

1 Durch Zustandsbasiertes Modellieren wird umwelstbasiertes Triggern erlaubt, das durch rein event-basiertes Modellieren nicht möglich ist

### Spezifische Betrachtung der Modellierungskonstrukte

Durch den Tokenfluss wird der Kontrollfluss dargestellt und dadurch der Prozessfortschritt explizit modelliert, indem der Standort der Token in einem Petri Netz als Markierung bezeichnet wird. Die Markierung zeigt damit an, wie weit das Token im Petri Netz fortgeschritten ist. Das Petri Netz ist durch Vorwärtsverkettung modelliert [Rei10], [DAH05]. Datenfluss wird im Petri Netz nicht unterstützt, da es kein Konstrukt angeboten wird das Daten darstellen kann.
Nebenläufigkeit wird durch das Feuern einer Transition erreicht, die mehr als eine ausgehende Kante besitzt. Durch das Feuern einer Transition wird auf jede ausgehende Kante ein Token gebracht. Beim Join wird wieder eine Transition benötigt, die diesmal mehrere eingehende und eine ausgehende Kanten besitzt. Dies hat synchronisierenden Charakter [DAH05].
Entscheidungen werden durch Stellen, die mehrere ausgehende Kanten haben, modelliert. Dabei kann das Token nur genau zu einer Transition fließen. Die Auswahl, welche Transition feuert, erfolgt nichtdeterministisch, kann jedoch durch Wächter beschrieben werden [DAH05]. Beim Mergen von verschiedenen Zweigen durch eine Stelle mit mehreren eingehenden Kanten und einer ausgehender Kante, muss darauf geachtet werden, dass hier keine synchronisierende Semantik vorliegt [DAH05].
Beendigung von Prozessen wird erreicht, indem alle Token die in einem Petri Netz unterwegs sind, an einer Endstelle angekommen sind und dort konsumiert werden. Endstellen sind Stellen die keine ausgehenden Kanten besitzen.
Subprozesse lassen sich durch Petri Netze nicht modellieren, können jedoch in einem Prozess durch Nebenläufigkeit und anschließenden Join modelliert werden.
Zustände sind als ein Hauptmerkmal von Petri Netzen definiert. Zustände werden durch jede Stelle angegeben. Sie spiegeln dabei einen lokalen Zustand des Prozesses wieder (Markierung) [DAH05], [Aal98b]. Das Petri Netz kann zur Laufzeit nicht modifiziert werden.

### Modellierung des Beispielprozesses

Abbildung 6.3 zeigt die Modellierung des Beispielprozesses durch ein Petri Netz. Anhand dieses Beispielprozesses sieht man, dass die Petri Netze sich auf wenige Elemente der Modellierung beschränken. Denn der gesamte Beispielprozess ist aufgebaut aus Transitionen (z.B. Sonographie), Stellen (z.B. Überwiesen an Radiologen) und den Kontrollflusskanten

dazwischen. Das hat zur Folge, dass Entscheidungen und Parallelität nur mit diesen Mitteln abgebildet werden. Entscheidungen, wie bei „Analyse des Ergebnisses“ mit mehreren ausgehenden Kanten aus einer Transition und Parallelität wird synchronisiert durch mehrere eingehende Kanten in eine Stelle, wie bei „Krebs Beweis“. Das macht es für den Benutzer des Modells nicht einfach das Modell zu verstehen. Das Konzept der Daten fehlt vollständig. Und da auf eine Stelle immer eine Transition folgen muss, werden viele Pseudostellen und -transitionen erschaffen (z.B. Überwiesen an Radiologen). Das macht das Modell komplexer und umfangreicher.

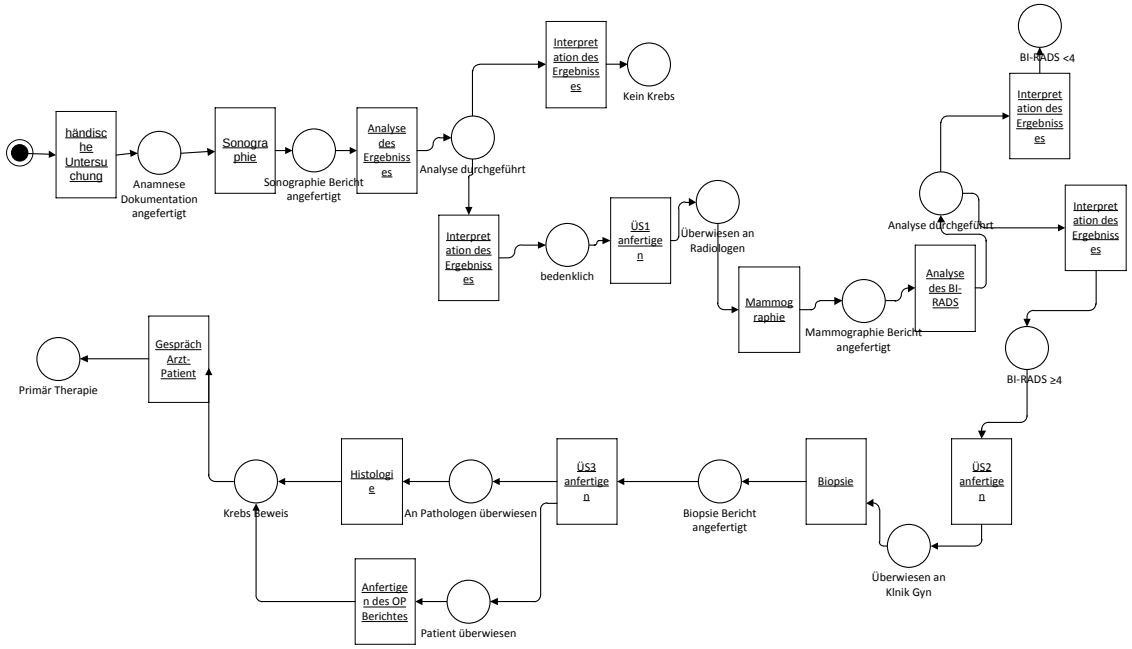



**Bild 6.3:** Umsetzung des Beispielprozesses mit Petri-Netz

### Eignung anhand der Evaluierungsframeworks

Die Eignung von Petri-Netzen wird mit Hilfe der Workflow Patterns [AHKA03] und dem BWW-Modell [RRIG09] gezeigt. Die Analyse mit den Workflow Patterns ist in [AHKA03] dargestellt, die Analyse mit dem BWW-Modell wird in [RRIG09] gezeigt. Petri Netze unterstützen die grundsätzlichen Kontrollmuster komplett. Bei erweiterten Synchronisierungsmuster werden, wie bei UML AD’s, das synchronisierende Mischen Muster unterstützt. Die Strukturmuster werden nur teilweise unterstützt (Diskriminator wird nicht unterstützt). Bei den Abbruchmustern wird das implizite Beendigungsmuster nicht unterstützt. Bei den mehrfach-Instanzmustern werden mehrere Muster nicht unter-

stützt. Zustandsbasierte Muster werden komplett unterstützt, im Gegensatz dazu werden keine Abbruchmuster unterstützt. In Bezug auf das BWW-Modell unterstützten die Petri-Netze in der Kategorie der Gegenstände und deren Eigenschaften, die Eigenschaften nicht. In der Kategorie Zustände der Gegenstände wird die Historie nicht unterstützt. Bei Ereignissen und Transformationen von Gegenständen wird z.B. Kopplung nicht unterstützt. Systeme von Gegenständen werden gar nicht unterstützt. Als Ergebnis, wird in [RRIG09] festgehalten, dass die Petri Netze damit 41,4% ontologisch vollständig sind.

#### Erweiterungen der Petri Netze

Es existieren diverse Erweiterungen der Petri Netze. Häufig verwendet werden die farbigen Petri Netze [Jen91], [Jen87]. Dabei werden die Tokens gefärbt. Die Farbe der Token gibt dabei den Typen des Tokens an. Dadurch lässt sich durch diese farbigen Petri Netze auch Datenfluss darstellen. Diese farbigen Petri Netze sind ein Beispiel für sogenannte High Level Petri Netze, die auch Erweiterungskontruke für Zeit und Hierarchie anbieten [Aal98a]. Diese Hierarchie kann benutzt werden um Subprozesse abzubilden.

### Yet Another Workflow Language

In [AHKA03] und [AWG05] wird eine Modifikation von Petri Netzen angegeben: Yet Another Workflow Language YAWL. YAWL versucht einige Schwächen von Petri Netzen zu umgehen und bietet deswegen neue Konstrukte zur Modellierung an. Diese neuen Konstrukte sind:

- Explizite Konstrukte für AND-, OR- ,XOR-Split/Join
- Kompositum Transitionen zur Modellierung von Subprozessen
- Abbruchregionen, um Prozesse abbrechbar zu machen
- Aufweichen der Bedingung, dass sich Stellen und Transitionen abwechseln müssen

Durch diese Erweiterung ist die Modellierung von Prozessen übersichtlicher und leichter zu verstehen, aber dennoch bleiben die Vorteile der Modellierung mit Petri Netzen erhalten.

### Modellierung des Beispielprozesses mit YAWL

In Abbildung 6.4 wird der Beispielprozess modelliert mit YAWL angegeben. Wie man an der Beispielmodellierung sieht, stellt YAWL eine vereinfachte und verständlichere Modellierung dar. Es werden XOR-und AND-Transitionen modelliert zum Entscheiden und zum Modellieren der Nebenläufigkeit.

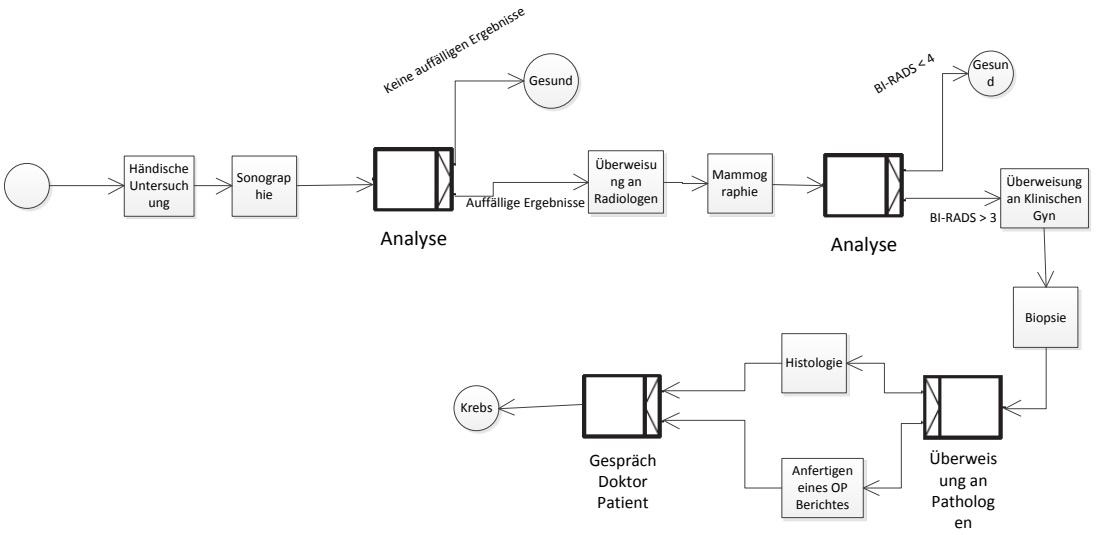



**Bild 6.4:** Umsetzung des Beispielprozesses mit YAWL

### Eignung anhand der Evaluierungsframeworks

Laut Aalst et al. [AHKA03], kann zur Eignung von YAWL zur Prozessmodellierung gesagt werden, dass die Klassifizierung durch die Workflow Patterns nur bedingt aussagekräftig ist, da YAWL gerade zu dem Zweck entwickelt wurde diese zur erfüllen. Deshalb erfüllt YAWL alle Muster, bis auf die implizite Beendigung, was von den Entwicklern auch nicht beabsichtigt ist [AHKA03], weil sie den Modelldesigner dazu bringen wollen über die Beendigungsbedingungen des Prozesses nachzudenken.

## 6.5 Ereignis gesteuerte Prozessketten

Die Ereignis gesteuerte Prozesskette (EPK) wurde am Institut für Wirtschaftsinformatik der Universität des Saarlandes 1992 entwickelt. Dabei bestand eine Zusammenarbeit mit SAP [NR02], [NR05]. EPK's haben sich als Standard zum Modellieren von Geschäftsprozessen etabliert [NZ98]. Deshalb werden von Mendling et al. und Nüttgens und Rump auch Möglichkeiten angegeben, die Ereignis gesteuerten Prozessketten in BPEL4WS Modelle zu überführen bzw. im Gesundheitswesen einzusetzen [MNN05], [NR05]. Die EPK's sind nur semi-formal definiert, was zur Folge hat, dass Prozessmodelle nicht eindeutig sind und falsch interpretiert werden können [Aal99]. Deshalb werden viele Ansätze angeboten, die EPK's zu formalisieren [Aal99], [Rit00], [NR02]. Auf der anderen Seite bedeuten semi-formale Semantiken, dass die Modellierungsnotation leichter erlernbar und anwendbar ist. Deshalb hat sich die Ereignis gesteuerte Prozesskette so schnell im betrieblichen Umfeld durchgesetzt [Rit00], [DAH05].

#### Grundsätzliche Modellierungskonstrukte

Zur Prozessmodellierung werden in der ursprünglichen Definition der Ereignis gesteuerten Prozessketten folgende Konstrukte angegeben:

- Funktionen
- Ereignis
- logische Konnektoren (AND, OR und XOR)
- gerichtete Kanten

EPK's modellieren den Kontrollfluss zwischen einem Ereignis und einer Funktion und zwischen einer Funktion und eines Ereignisses durch die gerichteten Kanten, die die Reihenfolge der Aktivitäten im Prozess bestimmen. Ereignisse und Funktionen müssen im Wechsel auftreten. Dabei stellen Ereignisse die Vor- bzw. Nachbedingungen von Funktionen dar. Funktionen spezifizieren die Durchführung von Aktionen [Rit00], [Aal99]. Dadurch lassen sich die Ereignis gesteuerten Prozessketten in das aktivitätsbasierte Paradigma der Prozessmodellierung einordnen. Die Modellierung ist durch Vorwärtsverkettung gekennzeichnet, es wird jedoch nicht das Tokenkonzept benutzt.
In dieser Arbeit wird die erweiterte Ereignis gesteuerte Prozesskette verwendet (erweiterte Ereignis gesteuerte Prozesskette (eEPK)). Dadurch werden weitere Konzepte zur Modellierung eingeführt: Die sogenannten Ressourcen [NZ98]. Diese Ressourcen beinhalten

die Abbildung von Datenflüssen, Organisationseinheiten oder Anwendungssystemen im Modell.
Im weiteren Verlauf der Arbeit wird der Begriff EPK und eEPK ununterscheidbar benutzt, da heutzutage eigentlich nur noch die eEPK's benutzt werden, da sie sinnvolle Erweiterungen anbieten, die den Grundbedarf an Modellierungskonstrukten abdecken.

**Spezifische Betrachtung der Modellierungskonstrukte**

Datenherkunft wird durch die Verknüpfung von Datenressourcen und Funktionen dargestellt. Dies erfolgt durch die Verbindung der Ressource und der Funktion mit Hilfe einer gerichteten Kante. Dies spezifiziert das Datenobjekt entweder als Input oder als Output [NZ98]. Die Datenressourcen stellen keine vollständige Sicht auf die Daten dar, sondern nur einen bereichsspezifischen Ausschnitt, der zur jeweiligen Funktion benötigt wird bzw. erzeugt wird. Die erzeugten Daten können aber daraufhin als transiente Daten als Input Daten einer späteren Funktion fungieren. Dabei haben die Datenobjekte keinen Einfluss auf den Prozessfortschritt.
Nebenläufigkeit wird durch die logischen Konnektoren erreicht. Dabei erzeugt der AND- bzw. OR-Konnektor Nebenläufigkeit. Ein Konnektor kann entweder als Split oder als Join modelliert werden, der Split erzeugt die Nebenläufigkeit, der Join synchronisiert diese wieder. Wobei hier Vorsicht geboten ist, da nur der AND-Join eindeutig synchronisierenden Charakter hat. Der XOR-Join erlaubt die ausgehende Funktion genau sooft, wie vorherige Ereignisse eingetreten sind, weshalb man damit unbeabsichtigte Situationen generieren kann [DAH05]. Wie sich der OR-Join verhält ist Gegenstand einer ausgiebigen wissenschaftlichen Diskussion, deshalb wird hier nicht weiter darauf eingegangen [Aal99], [Rit00], [MNN05].
Entscheidungen werden auch auf Grund der logischen Konnektoren getroffen. Der XOR-Konnektor bedeutet damit, dass nur eine ausgehende Kante verfolgt werden kann. Der OR-Konnektor spiegelt somit eine n aus m Auswahl wider. Das spezielle Verhalten der Entscheidungen kann durch Wächter an den ausgehenden Kanten spezifiziert werden [DAH05]. Die logischen Konnektoren können auch zur logischen Verknüpfung von Ereignissen benutzt werden, um zum Beispiel auszudrücken, dass zwei Ereignisse eintreten müssen. Dass die nachfolgende Funktion ausgeführt werden kann, wird der AND-Konnektor benutzt [DAH05]. Die Beendigung von Prozessen wird ausgedrückt, durch das Eintreffen aller Prozesszweige in einem Ereignis ohne ausgehende Kanten.
Zustände werden bei EPK's nicht modelliert. Die Ereignisse spiegeln nur die jeweiligen

Vor- bzw. Nachbedingungen der Funktion wider.
Für Subprozesse wird kein Konstrukt angeboten. Jedoch ist es auch hier wieder möglich, dies durch ein Hilfskonstrukt zu lösen, indem man Nebenläufigkeit erzeugt und dann wieder synchronisierend joined.

### Modellierung des Beispielprozesses

Eine Modellierung des Beispielprozesses ist in Abbildung 6.5 gegeben. Der Prozess wird durch Funktionen (wie z.B. händische Untersuchung) und Ereignisse (wie z.B. Normale Ergebnisse) modelliert. Dabei fällt auch auf, dass hier, wie bei den Petri Netzen, viele Pseudozustände modelliert werden müssen. Explizite Entscheidungen werden durch den XOR-Konnektor modelliert, wie die Entscheidung, ob der BI-RADS größer gleich 4 ist oder kleiner. Die Nebenläufigkeit wird durch AND-Konnektoren erzeugt, im Beispiel der AND-Split nach dem Ereignis „Überwiesen an Pathologen“. Das Synchronisieren wird durch den AND-Join vor „Arzt-Patient Gespräch“ angezeigt. Die Daten werden durch gerichtete Kanten an die Funktionen gebunden. Zum Beispiel sind die „Mammographie Ergebnisse“ Output der „Mammographie“ aber Input des Ereignisses „Analyse der Ergebnisse“. Die Daten haben aber dennoch keinen Einfluss auf die Entscheidung, sondern stellen nur allgemein dar, dass Daten benötigt werden, um diese Analyse durchzuführen.

### Eignung anhand der Evaluierungsframeworks

Zur Eignung von EPK's wird sowohl das Framework der Workflow Patterns [MNN05], als auch das BWW-Modell [RRIG09] benutzt. Die Standardmuster der Workflow Pattern werden durch die EPK's[1] unterstützt, die erweiterten Synchronisierungsmuster werden bis auf mehrfach Merge und Diskriminator unterstützt. Genauso werden die Strukturmuster unterstützt. Multiple Instanzen werden generell nicht unterstützt und somit diese Muster auch nicht. Konstrukte für zustandsbasierte Muster werden nicht bereitgestellt. Die Abbruchmuster sind auch nicht modellierbar. Bei der Abbildung auf das BWW-Modell sieht man, dass die EPK's Gegenstände nicht unterstützen, nur Eigenschaften werden unterstützt. In der Kategorie Zustände von Gegenständen werden instabile Zustände, Historien und Zustandsräume nicht unterstützt. In der Kategorie der Ereignisse und Transformationen von Gegenständen unterstützen die EPK's bis auf

1 Das einfache Mergemuster lässt sich nur mit lokaler Semantik verwirklichen

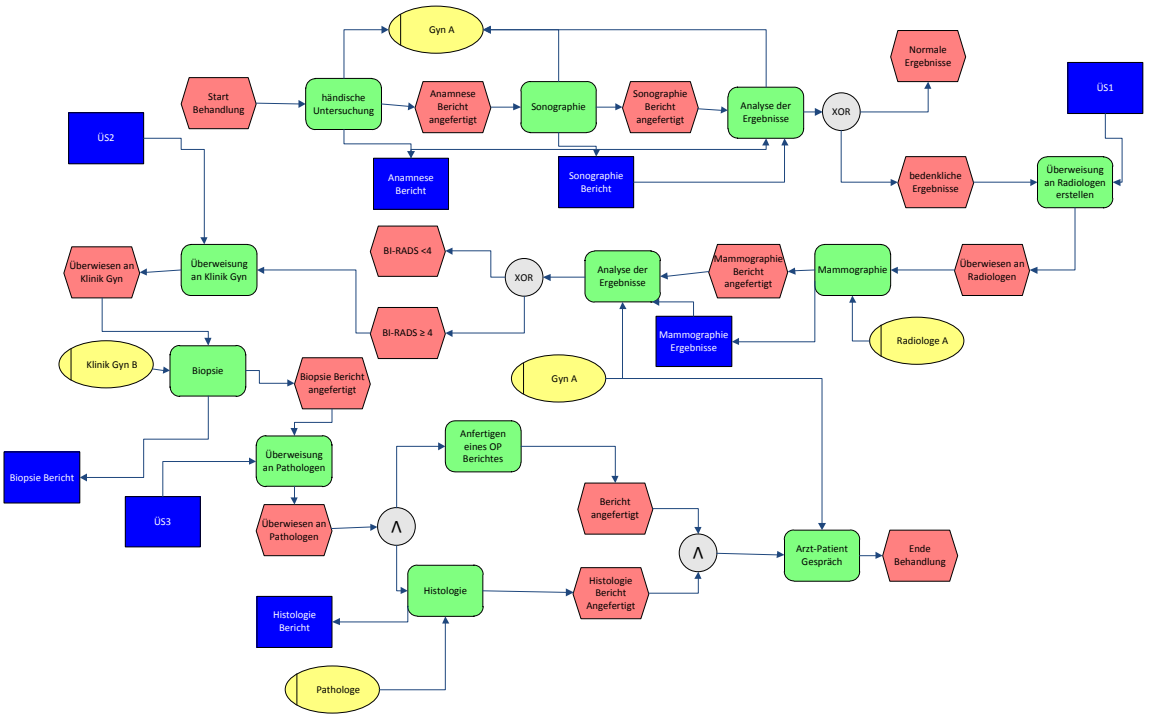



**Bild 6.5:** Umsetzung des Beispielprozesses mit EPK

Kopplung und Ereignisräume alle Konstrukte. EPK's unterstützen nur Levelstrukturen sonst keine Systemkonstrukte des BWW-Modells.

### Erweiterungen der Ereignis gesteuerten Prozessketten

Es gibt auch bei der Ereignis gesteuerten Prozesskette eine Vielzahl von Erweiterungen, die sich vor allem auf die Beseitigung von semantischen Uneindeutigkeiten beziehen [MNN05], [Rit00], [NZ98]. Ein Ansatz soll hier kurz erwähnt werden: Yet another Event-driven Process Chain. Dieser Ansatz orientiert sich sehr stark am Ansatz von YAWL [MNN05], da auch hier versucht wird, Schwächen der ursprünglichen Notation zu beseitigen. Deshalb werden auch hier Möglichkeiten definiert, Subprozesse, Abbruchregionen und neue Konnektoren darzustellen, um Probleme zu beseitigen und die Semantik der darstellbaren Modelle zu erweitern [MNN05].

## 6.6 PHILharmonicFlows (objektorientierte Prozessmodellierung)

Die folgende Darstellung des PHILharmonicFlows Ansatzes basiert auf [KR10], [KWR10] und [KR11]. PHILharmonicFlows stellt einen Vorschlag zur Prozessmodellierung mit Schwerpunkt auf der Datenperspektive dar. Er wird zur Zeit am Institut für Datenbanken und Informationssystem der Universität Ulm entwickelt. Des Weiteren ist der Ansatz als objekt-orientiert zu bezeichnen und ist spezialisiert auf die Benutzung von Formularen zur Prozessunterstützung. Dazu werden in dem Forschungsprojekt PHILharmonicFlows bestimmte Anforderungen deklariert und aufgestellt, die ein Prozessunterstützungssystem bieten muss, um als objekt-orientiert gelten zu können und gleichzeitig den Schwerpunkt auf die Datenperspektive zu setzen.

### Grundsätzliche Modellierungskonstrukte

Deshalb werden bei PHILharmonicFlows das Objektverhalten basierend auf Zuständen und die datengetriebene Prozessausführung kombiniert. Der Prozess wird dabei durch Objektinstanzen koordiniert. Diese Objekte sind wiederum prozessrelevante Objekte (z.B. Dokumente). Von diesen Prozessinstanzen können zur Laufzeit unterschiedlich viele Instanzen nebeneinander laufen. Bei PHILharmonicFlows wird ein integrierter Zugang zum Geschäftsprozess, seinen Funktionen und den Daten geboten. Der Prozessfortschritt wird getrieben durch Attribute in den Objekten. Deshalb lässt sich PHILharmonicFlows in das inhaltsbasierte Paradigma der Prozessmodellierung einordnen. Dies ist sowohl durch Vorwärtsverkettung als auch durch Rückwärtsverkettung gelöst.
Es gibt zwei Hauptmodellierungskonstrukte:

1. Microprozesse, zum modellieren von Objekt**verhalten**
2. Macroprozesse, zum modellieren von Objekt**interaktion**

### Microprozesse

Pro Objekttyp existiert ein Microprozess. Diese sind charakterisiert durch eine Menge von Zuständen und Transitionen zwischen den Zuständen (siehe Bild 6.6 z.B. ist Aufnahme hier ein Zustand des Microprozesses). In jedem Zustand des Micro Prozesses gibt es eine Menge von Attribute des Objekttyps zu setzen (siehe Bild 6.6 z.B PatientenID und

AuftragsgeberID in Zustand Aufnahme von Objekttyp Mammographie Bericht Diagnose). Diese sind verpflichtend zu setzen. Nur wenn alle einem Zustand zugeordneten Attribute gesetzt sind, wird der nächste Zustand zur Bearbeitung freigeschalten und die dort enthaltenen Attribute auf die verpflichtende Bearbeitung gesetzt. Dies kennzeichnet die datengetriebene Prozessausführung. Des Weiteren wird ein optionaler Zugriff auf Attribute späterer Zustände auf Grund von Authorisierungstabellen zugelassen. Diese Tabelle ist eine Matrix mit Attribut x Rolle + Zustand des Microprozesses in den Dimensionen und die Rechte als Eintrag. Innerhalb der Microprozesse ist der Prozess vorwärtsverkettet (Kontrollfluss), die Reihenfolge der Bearbeitung kann jedoch durch die optionale Bearbeitung variiert werden. Übergänge von Zuständen kann von purer Datenabhängigkeit abhängen, oder durch Bestätigung von Benutzer erreicht werden, dazu gibt es implizite Zustandstransitionen und explizite Zustandstransitionen (siehe Bild 6.6 z.B. implizit von Aufnahme nach Analyse, aber explizit von Analyse nach Beweis). Microprozessänderungen sind jedoch nicht mehr zur Laufzeit möglich.
Nebenläufigkeit ist implizit durchgängig gegeben. Nicht-Nebenläufigkeit ist durch das Spezifizieren von den verpflichtenden Attributen gegeben. Entscheidungen werden auf Attributebene dadurch getroffen, dass bestimmte Belegungen von Attributen zu einem anderen Prozessverlauf führen können, dies wird explizit im Modell gezeigt (siehe Bild 6.6 Entscheidung auf Grund von Wert des BI-RADS Attributs). Zustände sind explizit als Konstrukt vorhanden. Der Zustand des Micro Prozesses spiegelt den Zustand des Objekttyps im Prozess wieder. Die Modellierung eines beispiehaften Micro Prozesses, siehe Abbildung 6.6. Es sei darauf hingewiesen, dass dies nur einen Micro Prozess zu dem Mammographie Bericht Diagnose darstellt. Dabei hat dieses Objekt die links dargestellten Attribute, die durch die Zustände Aufnahme, Analyse und Beweis abgearbeitet werden.

### Beispielmodellierung eines Microprozesses

In Abbildung 6.6 ist eine Modellierung eines Microprozesses angegeben. Hier ist links der Objekttyp „Mammographie Bericht Diagnose“ angegeben. Dort werden die Attribute dargestellt die dieses Objekt enthält (z.B. PatientenID). Die Zustände des Microprozesses sind rechts daneben angegeben (z.B. Aufnahme, Analyse). Innerhalb der Zustände sollen die Attribute des Objekts belegt werden. Die Reihenfolge ist vormodelliert, kann jedoch wie beschrieben variiert werden. Die expliziten Transitionen werden als durchgezogene Pfeile modelliert, die impliziten als gestrichelte Pfeile. Attributsvorbelegungen werden

durch kleine Boxen innerhalb der Attribute modelliert (z.B. größer gleich vier innerhalb von BIRADS). Wenn man einmal einen Zustand erfolgreich abgearbeitet hat, geht man in den nächsten über und kann nicht mehr in den vorherigen zurückkehren.

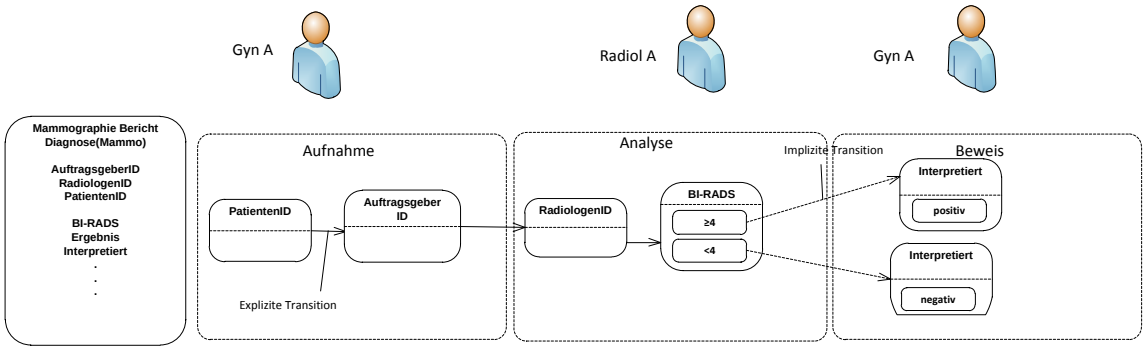



**Bild 6.6:** Umsetzung des Beispielprozesses (Ausschnitt) mit PHILharmonicFlows (Microprozess)

### Macroprozesse

Dieser Zustand des Microprozesses wird im Macroprozess zur Modellierung der Objekt-Interaktion benutzt. Hier werden die Interaktionen von verschiedenen Objektinstanzen angegeben. Diese Interaktion basiert auf der Interaktion der verschiedenen Objekttypen und ihrer Microprozessen und dessen Zuständen (siehe Abbildung 6.7). Im Macroprozess lassen sich AND und OR Semantiken modellieren, um notwendige Interaktionen zu beschreiben bzw. alternative Interaktionen zu beschreiben. Es ist möglich erforderliche Objektinstanzen mittels Kardinalitäten anzugeben, damit kann man auch angeben, wie viele Instanzen benötigt werden, um weiter im Prozess fortzuschreiten.

### Beziehungen von Objekttypen

Des Weiteren werden die Beziehungen[1] der einzelnen Objekttypen im Macroprozess explizit als verschiedenfarbige, gerichtete Kante abgegeben. Die Semantiken dieser Kanten ist, dass sie das Verhältnis der Objekte zu einander beschreiben. Top-Down beschreibt, das Verhältnis eines höher aggregierten Objektes zu einen niedriger aggregierten (zum Beispiel einer Stellenausschreibung zu einer dazu gehörigen Bewerbung), die Bottom Up

1 Beziehungen bedeutet hier, dass angegeben wird auf welcher Ebene die verschiedenen Objekttypen in Verbindung stehen: Dies ist zum Beispiel Bottom Up oder Top Down

Beziehung beschreibt die umgekehrte Richtung, von einem niedriger aggregierten Objekts zu einem höher aggregierten Objekts (z.B. von einem Bewerbungsgesprächteilnehmers zu einer Stellenausschreibung), die Transverse Beziehung beschreibt eine Beziehung auf der selben Aggregationsstufe.

### Prozessablauf im Macroprozess

Der Prozessfortschritt wird hier durch die Abarbeitung der einzelnen Microprozesse erreicht, dies ist im Modell des Macroprozesses durch Vorwärtsverkettung gelöst. Ein Abweichen vom Prozessmodell ist nicht möglich, da Macroprozesse auf den Microprozessen, der Objekte basieren und sich die Microprozesse nicht ändern lassen. Die Ababarbeitung der Macroprozesse ist somit determiniert durch die Zustandsreihenfolge der Microprozesse. Die einzige Abweichung ist, dass es möglich ist, zur Laufzeit neue Instanzen von Objekttypen zu erschaffen und dadurch mehrere nebenläufige Micro Prozesse zu kreieren.
Der Prozess ist beendet, indem der letzte Schritt im Macroprozess erreicht ist. Es ist im PHILharmonicFlows Rahmenwerk kein Konstrukt für Subprozesse vorhanden, jedoch können Subprozesse mit Hilfe der Top-Down Beziehung dargestellt werden. Dies ist jedoch nur bei direkten Beziehungen zwei aufeinanderfolgender Aggregationsebenen möglich. Zum Beispiel sind alle Bewerbungen als Subprozesse einer freien Stelle zu verstehen. Aber ein Bewerbungsgesprächteilnehmer stellt keinen Subprozess für eine Bewerbung dar.

### Beispielmodellierung eines Macroprozesses

In Abbildung 6.7 ist ein beispielhafter Macroprozess gezeigt, der den Microprozess aus Abbildung 6.6 benutzt. Es sei auch hier darauf hingewiesen, dass dies eine unvollständige Darstellung des Beispielprozesses ist, da nur die Zustände des oben angegebenen Microprozesses in die Darstellung eingeflossen sind. Der Macroprozess wird aus den Microprozessen aufgebaut (z.B. Sonographie). Die Zustände der Microprozesse werden innerhalb der Kästen für die Microprozesse angegeben (z.B. Aufnahme als Zustand des Micrprozesses Mammographie Bericht Diagnose).

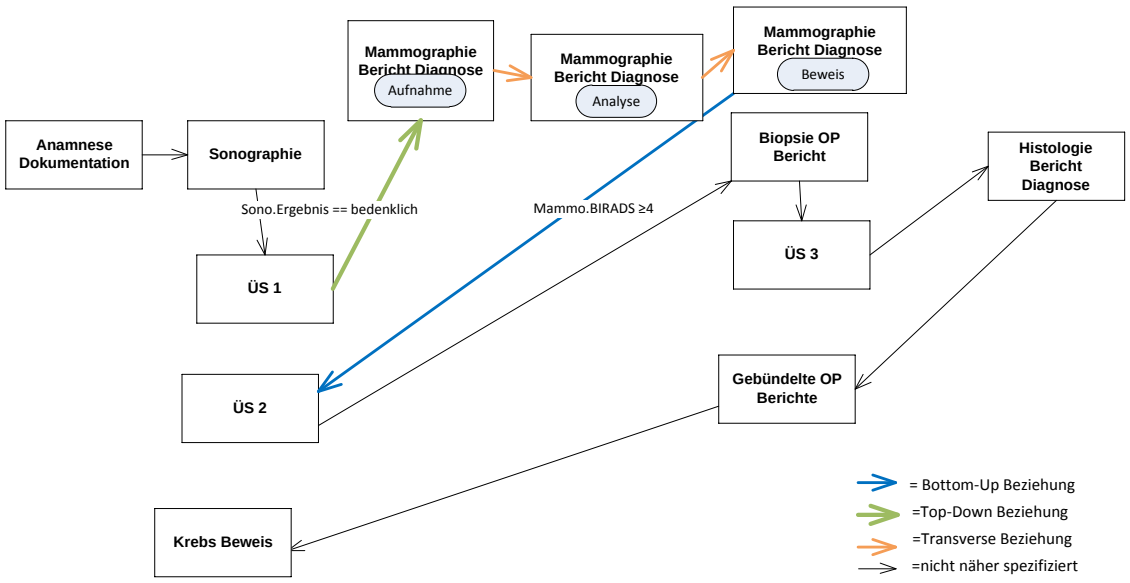



**Bild 6.7:** Umsetzung des Beispielprozesses (Ausschnitt) mit PHILharmonicFlows (Macroprozess)

### Eignung anhand der Evaluierungsframeworks

Zur Eignung von PHILharmonicFlows zur Prozessmodellierung wurde bisher keine Evaluierung durch die beiden Evaluierungsframeworks vorgenommen.

## 6.7 Adaptive Case Management

ACM basiert auf dem Buch „Mastering the Unpredictable - How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done“ von Keith D. Swenson [Swe10].

### Case Handling

Adaptive Case Management (ACM) basiert auf dem Paradigma des Case Handlings[1] [Swe10]. Dieses Paradigma wurde entwickelt, um Probleme bei der Benutzung von strukturierten aktivitätsbasierten Prozessmodellierungsparadigmen zu umgehen. Deshalb werden im Folgenden vier Ziele des Case Handling Ansatzes dargelegt, die auf die Probleme der aktivitätsbasierten Prozessmodellierung abzielen [AH05]:

- Bereitstellung der gesamten Informationen eines Cases zu jeder Zeit
- Entscheidung, welche Aktivitäten möglich sind, auf Grund von Datenabhängigkeiten (Rückwärtsverkettung)
- Unterschiedliche Arbeitsverteilung durch bessere Rollenverteilung
- Erlauben von Veränderungen an Daten, bevor sie benutzt werden

Als einer der Unterschiede zum aktivitätsbasierten Paradigma ist der Prozess datengetrieben, der Fokus des Prozesses liegt auf dem Case und der Prozess ist implizit modelliert [2][RRVDA03]. Der folgende Abschnitt gibt eine Kurzdarstellung eines Ansatzes des Case Handling Paradigmas. Die Darstellung ist jedoch nicht vollständig, soll aber einen Einblick in die Ziele von ACM geben.

### Kurzdarstellung von ACM

ACM ist ein verteiltes System zum Management von Prozessen. Diese Prozesse sind jedoch keine Prozesse wie sie von BPMN oder ähnlichem modelliert werden können. Denn die Prozesse die ACM modellieren will, sind nicht-planbare Prozesse: sogenannte Knowledge Work. Diese Prozesse sind durch ad hoc Entscheidungen geprägt, wodurch

1 Auch $\alpha$-Flow basiert auf dem Case Handling Paradigma
2 Nur bevorzugter Weg durch den Prozess ist definiert, nicht die genaue Kontrollflussvorgabe

sich der konkrete Ablauf eines solchen Prozesses bzw. Cases nicht vorhersagen lassen kann. Im Gegensatz dazu gibt es die strukturierten bzw. planbaren Prozesse - sogenannte Routine Work. Es wird beschrieben, dass ein Großteil der Prozesse, die uns im täglichen Leben begegnen nicht planbar sind und deshalb nicht mit strukturierten Prozessmodellierungsnotationen handhabbar sind. Als Beispiel wird hier der Behandlungspfad im Krankenhaus angegeben. Dennoch wird im ACM Wert auf eine Kopplung mit typischen Business Prozess Management gelegt, da dadurch die Handhabung von strukturierten Prozessen leichter ist.
Im Folgenden werden kurz einige Bestandteile des Konzepts für Adaptive Case Management aufgeführt:

- Es sollen Arbeitslisten für die sogenannten Knowledge Worker generiert werden können (Dies sind Arbeiter, die die wissensintensive Arbeit verrichten müssen)
- Es soll ein Rollensystem integriert werden können, nach dem verschiedene Rollen verschiedene Sichten und unterschiedlichen Zugriff auf das ACM-System bekommen können
- Es soll ein Repository für Entities implementiert werden können, in dem zentral Daten für den Case gespeichert werden können
- Das Hauptmerkmal für ACM soll sein, dass es extrem anpassbar und flexibel ist, wodurch man den Ablauf eines Cases benutzerdefiniert gestalten kann
- Es gibt keine Prozessmodelle im eigentlichen Sinne, sondern es sollen Templates verfügbar sein. Diese Templates sind eine Art best practices der Abarbeitung eines Cases. Sie sind erweiterbar, weiter entwickelbar und benutzerdefiniert und können anderen Benutzern zur Verfügung gestellt werden. Templates beinhalten Aktivitäten, Attribute und Dokumente, die für die Bearbeitung des Cases wichtig sind

### Modellierung des Beispielprozesses

In Abbildung 6.8 wird eine Notationsbemühung aus [Kre11] gezeigt. Damit soll die Struktur des Konzeptes für ACM veranschaulicht werden. Eine spezifizierte Modellierungsnotation ist nicht vorhanden, denn es handelt sich bei ACM nur um ein Konzept zur Umsetzung des Case Handling Paradigmas. Dies grenzt ACM von den anderen besprochenen Ansätzen ab, die eine definierte Modellierungsnotation unterstützen und anbieten.

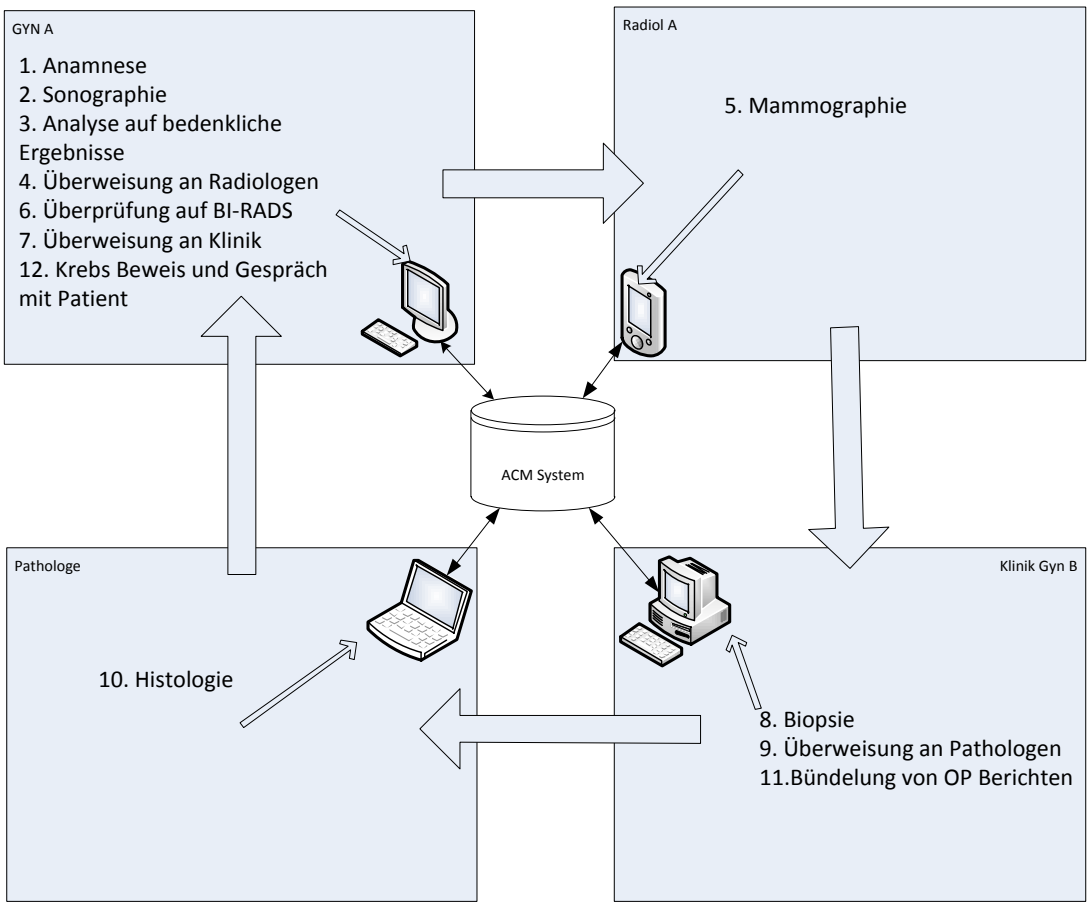



**Bild 6.8:** Umsetzung des Beispielprozesses (Schema) mit ACM

## 6.8 Zusammenfassung

In den vorherigen Abschnitten wurden die verschiedenen Modellierungsnotationen vorgestellt und einige Eigenschaften erläutert. In dieser Sektion soll darauf aufbauend eine tabellarische Zusammenfassung und Gegenüberstellung angegeben werden. Dabei werden nur diejenigen angegeben, die eine beispielhafte Modellierung des Beispielprozesses ermöglichten.
Tabelle 6.1 beschäftigt sich mit der Einteilung in verschieden Paradigmen, Tabelle 6.2 zeigt die Unterstützung der Nebenläufigkeit auf, Tabelle 6.3 zeigt die Unterstützung von Entscheidungen auf, Tabelle 6.4 zeigt die Darstellungsarten von Daten auf, Tabelle 6.5 zeigt das Vorhandensein von Kontrollfluss, Tabelle 6.6 listet die Möglichkeiten zur Beendigung eines Prozesses auf, Tabelle 6.7 zeigt die Art der unterstützten Zustände

auf, Tabelle 6.8 listet auf ob und wie Subprozesse unterstützt werden und Tabelle 6.9 beschreibt ob Änderungen zur Laufzeit möglich sind. Aufbauend auf dieser Zusammenfassung wird in Abschnitt 7 der Vergleich mit $\alpha$-Flow beschrieben und die Unterstützung dieser Konstrukte angegeben und besprochen. Dadurch wird das DAB-Modell entwickelt.

| *Notationsart* | *Paradigma* |
|---|---|
| BPMN | aktivitätsbasiertes Paradigma |
| UML AD | aktivitätsbasiertes Paradigma |
| Petri Netze | aktivitätsbasiertes Paradigma |
| EPK's | aktivitätsbasiertes Paradigma |
| PHILharmonicFlows | inhaltsbasiertes Paradigma |

**Tabelle 6.1:** Einteilung in Paradigmen

| *Notationsart* | *Unterstützung für Nebenläufigkeit* |
|---|---|
| BPMN | Durch Entscheidungsgateways auch Nebenläufigkeit erzeugt und gejoined |
| UML AD | Explizit durch Split und Join Balken |
| Petri Netze | Durch Transition mit mehreren Eingangs-bzw. Ausgangskanten |
| EPK's | Durch Entscheidungsknoten wird Nebenläufigkeit erzeugt und gejoined |
| PHILharmonicFlows | Implizit alles nebenläufig, Sequenzierung durch explizite Datenübergänge |

**Tabelle 6.2:** Unterstützung für Nebenläufigkeit

| *Notationsart* | *Entscheidungsunterstützung* |
|---|---|
| BPMN | Entscheidungsgateways |
| UML AD | Entscheidungsdiamond |
| Petri Netze | Stelle mit mehreren Ausgangskanten |
| EPK's | Entscheidungsknoten |
| PHILharmonicFlows | Entscheidungen an Hand von Attributsbelegungen |

**Tabelle 6.3:** Unterstützung für Entscheidungen

| *Notationsart* | *Datenunterstützung* |
|---|---|
| BPMN | Datenobjekt über Assoziation |
| UML AD | expliziter Datenfluss modellierbar |
| Petri Netze | keine Daten darstellbar (Erweiterungen siehe 6.4) |
| EPK's | Daten über gerichtete Kanten mit Funktionen verbindbar |
| PHILharmonicFlows | Prozesse werden auf Daten ausgeführt |

**Tabelle 6.4:** Unterstützung für Daten

| *Notationsart* | *Kontrollfluss* |
|---|---|
| BPMN | Durch Tokenkonzept unterstützt |
| UML AD | Durch Tokenkonzept unterstützt |
| Petri Netze | Durch Tokenkonzept unterstützt |
| EPK's | Unterstützt durch Vorwärtsverkettung der Funktionen |
| PHILharmonicFlows | Innerhalb des Micro/Macro-Prozesses Kontrollfluss |

**Tabelle 6.5:** Unterstützung für Kontrollfluss

| *Notationsart* | *Beendigung von Prozessen* |
|---|---|
| BPMN | Terminierung des kompletten Prozesses oder Beendigung eines Prozesszweiges |
| UML AD | Terminierung des kompletten Prozesses oder Beendigung eines Prozesszweiges |
| Petri Netze | Endstellen die Token konsumieren |
| EPK's | Endevent als Nachbedinung der letzten Funktion |
| PHILharmonicFlows | Endzustand des Micro Prozesses und Endzustand des Macro Prozesses |

**Tabelle 6.6:** Unterstützung für Beendigung von Prozesse

| *Notationsart* | *Unterstützung von Zuständen* |
|---|---|
| BPMN | teilweise Unterstützung durch Pseudozustände und Ereignisse(lokaler Zustand) |
| UML AD | keine Unterstützung |
| Petri Netze | explizite Unterstützung von lokalen Zustand als Stelle |
| EPK's | Unterstützung durch Vor-Nach-Bedingung von Funktionen durch Events |
| PHILharmonicFlows | Unterstützung als Zustand im Micro Prozess, als Zustand der Objektinstanz |

**Tabelle 6.7:** Unterstützung für Zustände

| *Notationsart* | *Unterstützung von Subprozessen* |
|---|---|
| BPMN | Unterstützung durch aufklappbare Subprozesse |
| UML AD | Unterstützung durch call-Activites |
| Petri Netze | keine Unterstützung (Workaround siehe 6.4) |
| EPK's | keine Unterstützung (Workaround siehe 6.5) |
| PHILharmonicFlows | keine Unterstützung |

**Tabelle 6.8:** Unterstützung für Subprozesse

| *Notationsart* | *Änderungen während der Laufzeit* |
|---|---|
| BPMN | nicht möglich |
| UML AD | nicht möglich |
| Petri Netze | nicht möglich |
| EPK's | nicht möglich |
| PHILharmonicFlows | möglich durch Veränderung der Anzahl laufender Objektinstanzen |

**Tabelle 6.9:** Unterstützung für Änderungen während der Laufzeit

# 7 Das Datenabhängigkeitsmodell von $\alpha$-Flow

In diesem Abschnitt werden die in Sektion 6.8 betrachteten Konstrukte in Verbindung mit $\alpha$-Flow gebracht und dabei erarbeitet, welche Möglichkeiten die $\alpha$-Flow Prozessmodellierung bietet, diese Konstrukte zu unterstützen. Hierbei wird auf die Konstrukte, Grobunterteilung in Paradigmen, Nebenläufigkeit, Entscheidungen, Datendarstellung, Kontrollfluss, Beendigung von Prozessen, Zuständen, Subprozessen und Änderungen zur Laufzeit eingegangen. Durch diese Betrachtung wird das Datenabhängigkeits (DAB)-Modell entwickelt, was als Ziel die Modellierung aller betrachteten Konstrukte durch Datenabhängigkeiten ermöglichen soll. Dabei soll keine Vorwärtsverkettung zum Einsatz kommen, sondern das Modell soll rein basierend auf Datenabhängigkeiten funktionieren. Abschließend soll in diesem Kapitel eine Gesamtübersicht über das DAB-Modell angegeben werden.

## 7.1 Einordnung in ein Paradigma

Wie in Tabelle 6.1 und in Abschnitt 6 gesehen, ist eine Unterteilung in bestimmte Paradigmen der Prozessmodellierung generell gut möglich und als sinnvoll zu erachten. Es gibt dabei das aktivitätsbasierte Paradigma und das inhaltsbasierte Paradigma. Das aktivitätsbasierte Paradigma stellt die Aktivität in den Mittelpunkt des Prozesses. Die Reihenfolge der Aktivitäten wird vorstrukturiert und ein Konzept für Daten ist nur teilweise spezifiziert. In dieses Paradigma lassen sich beispielsweise BPMN, Petri Netze, UML AD's und EPK's einordnen. Dem gegenüber steht das inhaltsbasierte Paradigma, das die Daten in den Mittelpunkt des Prozesses stellt und einen Datenproduktionsprozess beschreibt. Hier wird der Prozessfortschritt auf Grund von Daten freigeschalten. Somit ist Fortschritt im Prozess durch Aktivitäten auf den Daten erreichbar. $\alpha$-Flow speziell basiert auf dem inhaltsbasierte Paradigma. Die Motivation dafür ist, dass man Prozesse nicht vorstrukturieren kann und man deshalb flexibel auf Veränderungen reagieren

können muss. α-Flow basiert auf dem Ansatz des Case Handlings (siehe Abschnitt 6.7). Bei α-Flow wird ein Datenproduktionsprozess beschrieben. Durch das Freischalten von Daten wird Prozessfortschritt erreicht. Dieses Freischalten basiert auf Datenabhängigkeiten und somit Rückwärtsverkettung. Speziell in der Gesundheitswesendomäne ist der inhaltsbasierte Ansatz ein Vorteil, weil viele Daten benötigt und gesammelt werden, um den Patienten bestmöglich zu versorgen. Das inhaltsbasierte Paradigma hat es sich zur Aufgabe gemacht, Daten jederzeit und vollständig zur Verfügung zu stellen [AH05].

## 7.2 Datensicht im DAB-Modell

Wie in Tabelle 6.4 gesehen, verwenden die bisher betrachteten Notationsformen unterschiedliche Darstellungen für Daten. Dabei besitzen die bisher betrachteten Modellierungsformen nur lokale Sicht der Daten. Diese lokale Sicht auf die Daten bedeutet, dass man zu keiner Zeit im Prozess, Zugriff auf die Gesamtheit, der zu Verfügung stehenden Daten hat. Man besitzt nur den Zugriff auf das im Modell spezifizierte Datum. Dies vereinfacht das Prozessmodell, lässt aber ad hoc Entscheidungen auf Grund von Daten nicht zu, da nur Entscheidungen auf Grund von spezifizierten Daten nicht wohl fundiert sein können, da weitere Daten nötig sein können. Im Gegensatz dazu bietet α-Flow eine gesamte Übersicht über die Daten zu jeder Zeit. Das bedeutet, dass die Benutzer von α-Flow zu jeder Zeit im Prozessverlauf auf die Gesamtheit der Daten zugreifen können. Dies ist nötig, da die Motivation für α-Flow ist, dass ad hoc Entscheidungen auf Grund von Daten möglich sein müssen. Dies kann nur bewerkstelligt werden, wenn zu jeder Zeit ein Prozessteilnehmer auch die Sicht auf alle Daten hat. Des Weiteren wird im α-Flow Ansatz ein Datenproduktionsprozess beschrieben, weshalb während des Prozesses neue Daten zur Gesamtheit der Daten hinzugefügt werden können. Die Datenabhängigkeiten, die in α-Flow vorherrschen, werden in den nächsten Abschnitten ausführlicher betrachtet.

## 7.3 Notwendigkeit von Kontrollfluss

Wie in Tabelle 6.5 dargestellt, greifen alle betrachteten Ansätze der Prozessmodellierung auf Kontrollfluss zurück und strukturieren damit den Prozess und geben den Benutzer vor, was wie und wann zu tun ist. Ein Konzept des Kontrollflusses ist das

häufig angewandte Tokenkonzept, wodurch auch durch Abarbeitung von Aktivitäten der Prozessfortschritt erreicht wird. $\alpha$-Flow dagegen erreicht den Prozessfortschritt wie beschrieben durch Bereitstellen von Daten und ist somit rein durch Datenabhängigkeiten verkettet. $\alpha$-Flow benutzt somit keinen Kontrollfluss. Die Reihenfolge der Einträge in der Arbeitsliste stellt die Art der Verkettung dar, die $\alpha$-Flow zur Verfügung stellt. Diese Art der Verkettung wird in den nachfolgenden Abschnitten genauer erläutert.

## 7.4 Beherrschung der zeitlichen Abfolge mit Hilfe zweier Implikationen

Zu dem Konstrukt der Nebenläufigkeit wird in Abschnitt 6 und in Tabelle 6.2 gezeigt, dass die bestehenden Ansätze zur Prozessmodellierung verschiedene Konstrukte zur Modellierung von Nebenläufigkeit anbieten (z.B. Split-Konnektoren oder Split-Balken bzw. Join-Konnektoren oder Join-Balken). Diese unterscheiden sich vor allem nur syntaktisch für den Anwender untereinander. Des Weiteren besteht für diese Ansätze die allgemeine Notwendigkeit Nebenläufigkeit zu modellieren, da sonst alles sequenziell auszuführen ist. Als Gegensatz dazu ist im $\alpha$-Flow Ansatz das komplette Abarbeiten der Aufgabenliste implizit nebenläufig, deshalb wird für $\alpha$-Flow die Notwendigkeit für eine sequenzielle Abarbeitung hervorgehoben.

### Motivation

Da in $\alpha$-Flow alle Einträge in der Priorisierungsliste implizit nebenläufig sind, muss ein Konstrukt bereitgestellt werden, das es ermöglicht eine strikte zeitliche Reihenfolge zwischen zwei oder mehreren Priorisierungslisteneinträgen herzustellen. Sonst wäre es z.B. nicht möglich zu modellieren, dass man erst, wenn man einen Überweisungsschein hat auch zum Facharzt gehen darf. Dies muss also durch eine unterschiedliche Art der Datenabhängigkeit gelöst werden, da die implizit nebenläufige Verknüpfung keine strikte Reihenfolge vorgibt. Da alle Benutzer des Systems Zugriff auf alle Daten zu jeder Zeit besitzen, müssen alle in Kenntnis gesetzt werden über die strikte Abhängigkeit. Wenn z.B. Arzt A mit Arzt B kollaborieren, und Arzt A eine strikte Reihenfolge modelliert, dann soll dies Arzt B auch ersichtlich sein, dass er sich nicht darüber hinwegsetzen kann. Nun kann es jedoch sein, dass von einem Eintrag in der Priorisierungsliste der bereits in strikter zeitlicher Abhängigkeit steht, wieder eine strikte zeitliche Abhängigkeit besteht.

Dies kann auch wieder am Beispiel des Überweisungsscheins verdeutlicht werden: Denn erst nachdem man zum ersten Facharzt überwiesen werden kann, kann dieser einen auch behandeln. Somit ist eine Strukturierung in verschiedene Ebene notwendig. Auch bei der Implementierung muss dieser strikten zeitlichen Abfolge Rechnung getragen werden, indem die Reihenfolge explizit verankert sein muss.

### Diskussion des Lösungskonzepts

Dabei bestehen zwei verschiedene Varianten der Datenabhängigkeit unter den Einträgen in der Aufgabenliste:

- strikte zeitliche Abhängigkeiten (gekennzeichnet durch einfachen zurückgerichteten Pfeil mit doppelter Pfeilspitze)
- schwache zeitliche Abhängigkeit bzw. implizite Nebenläufigkeit (gekennzeichnet durch einfachen zurückgerichteten Pfeil mit einfacher Pfeilspitze)

### Schwache zeitliche Abhängigkeit

Die schwache zeitliche Abhängigkeit spiegelt sich in der Reihenfolge der Aktivitäten in der Arbeitsliste wider. Diese zeigt jedoch nur einen happy path des Cases, sind somit eine Abarbeitungsempfehlung. Das heißt, dass die Einträge in dieser Reihenfolge abgearbeitet werden sollten, jedoch nicht müssen, denn die Einträge können durch ad hoc Entscheidungen in anderer Reihenfolge abgearbeitet werden. Die implizite Nebenläufigkeit ist damit die Standardverkettung von Einträgen in der Arbeitsliste. Weil die Einträge normalerweise unsortiert vorliegen, ist die Arbeitsliste als Menge implementiert (siehe Abb. 7.1). In dieser Menge sind die Elemente mit ihren α-Card ID's gespeichert.

### Strikte zeitliche Abhängigkeit

Die strikte zeitliche Abhängigkeit bedeutet nun, dass erst wenn eine Aktivität bearbeitet wurde die nächste Aktivität zur Bearbeitung freigegeben wird (Erst wenn A, dann B!). Dabei wird ein Konzept zur farblichen Kennzeichnung verwendet, um zu verdeutlichen, dass z.B. die Aktivität[1] B, die Aktivität A voraussetzt, solange hervorgehoben dargestellt

1 Aktivität stellt hier einen Eintrag der Arbeitsliste dar

wird, bis Aktivität A vollendet ist. Was vollendet genau bedeutet, wird in den nächsten Abschnitten genauer behandelt. Sobald jedoch die Aktivität A bearbeitet ist, wird die Hervorhebung entfernt und normal gezeigt. Dies ermöglicht das Bearbeiten[1] der Einträge in der Arbeitsliste, da sie für jeden Prozessteilnehmer sichtbar ist. Jedoch soll durch die Hervorhebung verdeutlicht werden, dass eine andere Aktivität zwingend benötigt wird um diese Aktivität zu beginnen (Beispiel Überweisungsschein).
Der jetzige Stand im $\alpha$-Flow Ansatz ist, dass die Priorisierungsliste, nach Einfügereihenfolge aufgebaut wird. Deshalb finden weder schwache zeitliche Abhängigkeit, noch starke zeitliche Abhängigkeit Verwendung.

### Definition des Lösungskonzepts

Als Lösungskonzept zu dieser Problematik werden hier neue Begriffe eingeführt:

- *schwache Implikation* = die beschriebene schwache zeitliche Abhängigkeit, sie ist die standardmäßige Verknüpfung von Einträgen in der Arbeitsliste
- *strikte Implikation* = die beschriebene strikte zeitliche Abhängigkeit, mit ihr werden strikte zeitliche Reihenfolgen abgebildet.

Die schwache Implikation definiert, dass alle Einträge in dieser Reihenfolge abgearbeitet werden *sollen*. Die strikte Implikation definiert, dass alle Einträge in dieser strikten Reihenfolge abgearbeitet werden *müssen*. Die Hervorhebung der zeitlichen Abhängigkeit wird durch das *Konzept des Zurückstellens* gelöst. Damit werden die Elemente, die zeitlich von einem Eintrag abhängen ausgegraut dargestellt, bis der Eintrag erfüllt ist, solange sind die strikt zeitlich abhängigen Einträge *zurückgestellt*. Da in diesem Abschnitt eine Betrachtung der Verknüpfung von Einträgen einer Priorisierungsliste geführt wird, stellt sich die Frage warum man nicht von einer AND-Verknüpfung der Einträge ausgeht, wie das normalerweise für eine Arbeitsliste üblich ist. Doch im Falle von $\alpha$-Flow spielt die Reihenfolge der Einträge eine Rolle, bei einer AND-Verknüpfung würde sie dies aber nicht. Es können auch mehrere Einträge in Gruppen zusammengefasst werden, um die Abhängigkeit von einem anderen Eintrag darzustellen (Erst wenn A, dann B oder C oder D)[2].

1 Bearbeiten meint hier, z.B. bearbeiten von Metadaten
2 Die Einträge B, C und D sind hierbei wieder durch eine schwache Implikation verknüpft und können somit durch ad hoc Entscheidungen umsortiert werden

### Implementierungskonzept

Um die starke Implikation in der Implementierung abzubilden, wird zusätzlich zu der Mengenimplementierung (implizite Nebenläufigkeit, wegen fehlender Sortierung) noch eine Listenimplementierung (strikte Reihenfolge durch Sortierung) bereitgestellt, in der die α-Cards in Gruppen unterteilt sind. Diese Gruppen werden angelegt, je nachdem in welchem Implikationslevel sie sich befinden. Dieses Implikationslevel spiegelt die Strukturierung wider, die nötig ist, um eine Hierarchie von starken Implikationen zu bilden (Überweisungsschein → Behandlung Arzt). α-Cards müssen über die Extended ID zugreifbar sein. Schwache und starke Implikation basieren rein auf Datenabhängigkeiten, denn sie stehen im Gegensatz zum Tokenkonzept der aktivitätsbasierten Modellierungsnotationen. Es fließt kein Token von einem Eintrag der Priorisierungsliste zum nächsten, sondern durch die flexible Abarbeitung wird der Prozessverlauf vom Benutzer zur Laufzeit determiniert. In Abb.7.1 werden diese Konzepte verdeutlicht und grafisch veranschaulicht.

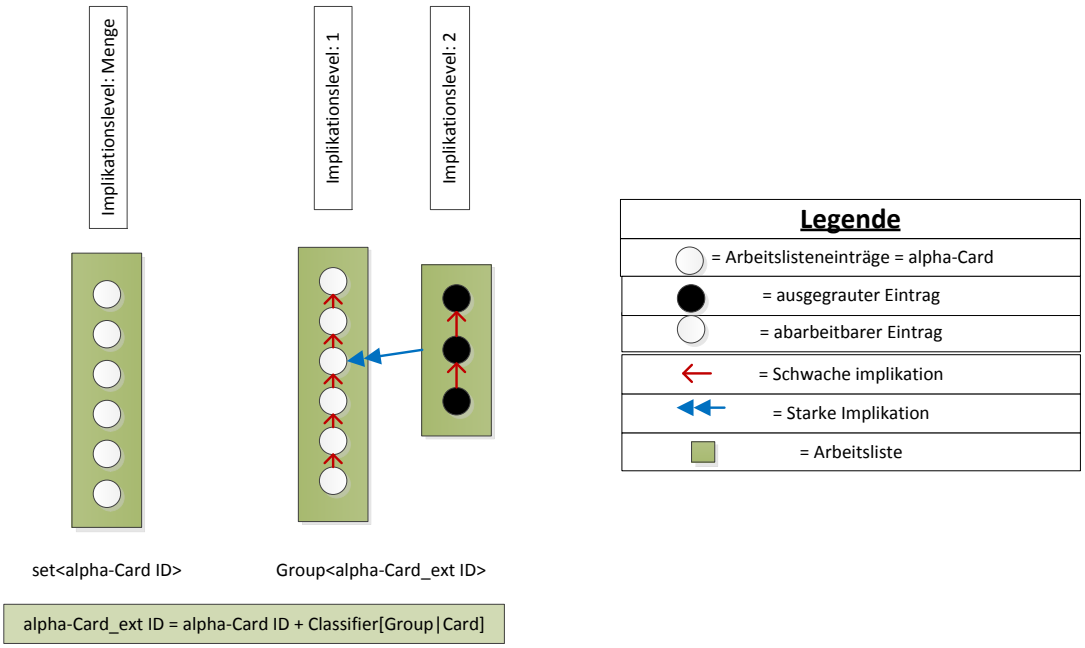



**Bild 7.1:** Darstellung der starken und schwachem Implikationen

## 7.5 Handhabung von Entscheidungen mit der PREANOR-Verknüfung

Entscheidungen werden bei den bisher betrachteten Ansätzen unterschiedlich modelliert. Diese Ansätze bieten verschiedene Konstrukte zur Modellierung von Entscheidungen. Es gibt XOR-Entscheidungskonnektoren, die eine exklusive Wahl zwischen Alternativen lassen. Des Weiteren gibt es OR-Entscheidungskonnektoren, die eine Auswahl von mehr als einer Alternative zulassen. Diese sind in Syntax ähnlich und die Semantik variiert nur marginal. Siehe dazu Tabelle 6.3. Bei $\alpha$-Flow dagegen sind die Entscheidungen nicht explizit enthalten, denn sie basieren auf ad hoc Entscheidungen zur Prozesslaufzeit.

### Motivation

Zur Motivation der Notwendigkeit zur Modellierung von Entscheidungen soll an dieser Stelle eine User Story angegeben werden. Diese User Story wird dem Artikel *The alpha-Flow Approach to Inter-Institutional Process Support in Healthcare* [NL12] entnommen. In der User Story geht es um eine Patientin, die eine Diagnose für Brustkrebs erhalten hat. Sie sitzt bei ihrem Gynäkologen, der gerade eine Sonographie durchgeführt hat. Das Ergebnis dieser Sonographie ist zweifelhaft, deshalb stellt der Gynäkologe den weiteren Behandlungsverlauf vor: Mammographie bei Radiologen, Biopsie und Histologie bei klinischem Gynäkologen, um zu bestimmen ob Brustkrebs vorliegt. Wenn ja, dann wird sie in das Universitätskrankenhaus für die Primärtherapie überwiesen. Dies stellt einen komplexen Behandlungsverlauf dar, der mit Hilfe des $\alpha$-Docs bearbeitet werden soll. Durch das Verteilen des Cases sind die Daten jederzeit verfügbar. Zu Beginn fügt der Gynäkologe die nötigen $\alpha$-Card Deskriptoren zur Verfügung, die seiner Meinung nach abgearbeitet werden müssen. So wird z.B. ein Deskriptor für die Mammographie erstellt, genauso für die Überweisungsscheine.
In dieser Situation zeigt sich die Bedeutung der Entscheidungen im $\alpha$-Flow Ansatz: Alle Untersuchungen sind von vorneherein notwendig (prospektiv verknüpft). Im Nachhinein kann jedoch eine Untersuchung gestrichen werden (positive Ergebnisse bei Untersuchung) (retrospektiv verknüpft). Die Reihenfolge der Untersuchungen muss nicht zwangsweise in dieser Reihenfolge geschehen (schwache Implikation), jedoch muss erst ein Überweisungsschein ausgestellt werden, bevor z.B. der Radiologe die Mammographie durchführen kann. Dies alles beruht auf einer Verknüpfung von den Einträgen der Priorisierungsliste,

muss aber vieles leisten können. Im Folgenden Abschnitt wird eine Diskussion zum Lösungskonzept angegeben.

### Diskussion des Lösungskonzepts

Die Arbeitsliste ist so modelliert, dass von Beginn an alle Schritte abzuarbeiten sind. Damit sind prospektiv alle Einträge der Arbeitsliste verpflichtend. Wenn jedoch in mitten einer Abarbeitung des Prozesses ad hoc sich gegen die Bearbeitung eines bestimmten Eintrags der Arbeitsliste entschieden wird, kann ein Eintrag auch gestrichen werden. Retrospektiv können demnach nicht alle Einträge der Arbeitsliste notwendig gewesen sein. Dies ist eine Frage des Standpunktes bzw. Beobachtungszeitpunkt. Aus der Beobachtung, dass es sowohl eine prospektive, als auch eine retrospektive Verknüpfung gibt, resultiert die Definition einer Verknüpfung die beide Prinzipien zusammenfasst.

### Diskussion der Verknüpfung

Diese Verknüpfung beinhaltet die Verknüpfung der Einträge in der Arbeitsliste, die durch Datenabhängigkeit gelöst wurde (Rückwärtsverkettung und schwache Implikation/-starke Implikation). Der Datenabhängigkeits-Teil der Definition schließt die schwache Implikation mit ein, denn die Einträge in der Arbeitsliste können in ihrer Reihenfolge variieren, genauso kann der Datenabhängigkeits-Teil die starke Implikation darstellen um strikte Abhängigkeiten zu definieren, die mit dem Konzept des Zurückstellens dargestellt werden. Der prospektive Teil der Verknüpfung resultiert aus der Beobachtung, dass von vorneherein alle Einträge notwendig sind. Der retrospektive Teil der Verknüpfung resultiert aus der Beobachtung, dass im Nachhinein durch ad hoc Entscheidungen nicht alle Einträge aus der Arbeitsliste abgearbeitet werden musstne.
Der jetzige Stand im α-Flow Ansatz ist, dass es kein spezifiziertes Konzept und Regelsystem für Streichungen von Arbeitslisteneinträgen gibt. Arbeitslisteneinträge können einfach unbearbeitet bleiben.

### Definition des Lösungskonzepts

Als Erklärungsmodell für die Verkettung der Einträge der Priorisierungsliste wird die *PREANOR-Verknüpfung* eingeführt. Sie besteht aus den Bestandteilen:

- „*PRE*“, dies ist der Datenabhängigkeitsteil der Verknüpfung
- „*AN*“, dies ist der prospektive Teil der Verknüpfung

- „*OR*“, dies ist der retrospektive Teil der Verknüpfung

Die PREANOR-Verknüpfung ist somit ein Erklärungsmodell für unsere Anforderungen an die Semantik der Arbeitsliste und dient als Begründung dafür, dass dieses Konstrukt zielführend ist. Dennoch stellt das Ergebnis der PREANOR-Verknüpfung eine primitive Arbeitsliste dar.
Die Datenabhängigkeitsbeziehung wird durch die starke bzw. die schwache Implikation beschrieben. Der prospektive Teil lässt sich aufassen als eine schwache AND-Verknüpfung, denn es sind prospektiv alle Einträge der Arbeitsliste nötig. Der retrospektive Teil lässt sich auffassen als eine retrospektive OR-Verknüpfung, da das streichen von Einträgen der Arbeitsliste durch ad hoc Entscheidungen möglich ist. Somit resultiert die PREANOR-Verknüpfung aus einer Trinität der schwachen/starken Implikation, des schwachen AND's und des retrospektivem OR's (siehe Abbildung 7.2).


**Bild 7.2:** Definition der PREANOR-Verknüpfung

### Abgrenzung der PREANOR-Verknüpfung

Abgrenzen lässt sich PREANOR von den anderen Kombinationen ANOR, PREOR und PREAND. PREAN würde bedeuten, dass das Streichen der einzelnen Einträge der Arbeitsliste nicht möglich ist. PREOR definiert keine notwendigen Einträge, da alle nur optional sind, dadurch wird der retrospektive Aspekt der retrospektiven OR-Verknüpfung ausgehebelt, denn man kann zu Beginn des Prozesses anfangen und Einträge in der Arbeitsliste streichen. Dies darf jedoch nicht der Fall sein, da dies keine ad-hoc Entscheidung repräsentiert. ANOR bezeichnet eine Verknüpfung, die nicht auf Datenabhängigkeiten beruht, deswegen scheidet sie für unsere Betrachtung aus. Diese Modellierung und Verknüpfung basiert rein auf Datenabhängigkeiten und ist damit rückwärtsverkettet, da als Verkettungsmechanismus der einzelnen Einträge auf die starke bzw. schwache Implikation zurückgegriffen wird. In Abbildung 7.3 ist eine Veranschaulichung der PREANOR-Verknüpfung dargestellt.

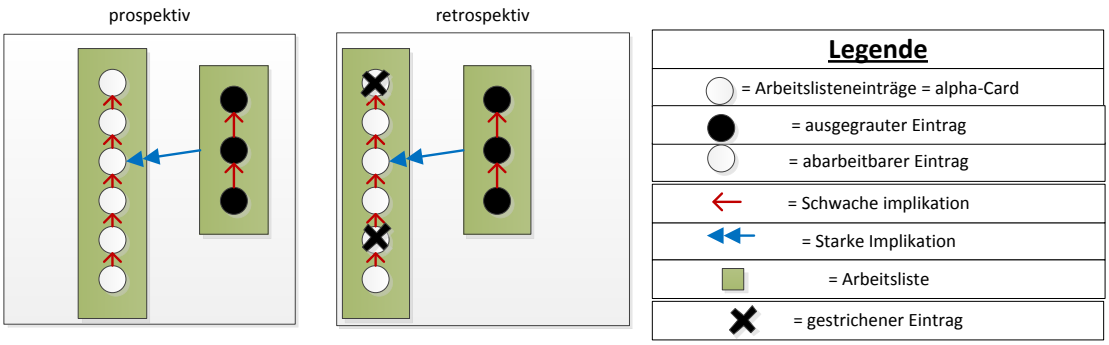



**Bild 7.3:** Veranschaulichung von PREANOR-Verknüpfung

## 7.6 Subprozesse im DAB-Modell

Subprozesse sind wichtig für die Prozessmodellierung, weil dadurch das Delegieren von Aufgaben und das Zerteilen von Aufgaben in kleinere Arbeitspakete bewerkstelligt werden kann. Wie in Tabelle 6.8 gezeigt, sind Subprozesse durch den Großteil der betrachteten Prozessmodellierungsnotationen darstellbar. Aber dennoch gibt es, trotz der Wichtigkeit des Konstrukts, Ansätze die Subprozesse nicht explizit abbilden können. Wenn sie nicht explizit darstellbar sind, bietet sich jedoch meistens die Möglichkeit durch ein Hilfskonstrukt das gleiche Ergebnis zu erzielen.

### Motivation

Zur Motivation der Vorteile des Konstrukts des Subprozesses wird hier auch eine User Story angegeben. Dazu wird die oben beschriebene User Story aufgegriffen. Der Gynäkologe in der Klinik hat demnach die Biopsie an der Patientin durchzuführen. Er greift dazu auf das Case File an seinem Desktop Computer in der Klinik zu. Dort bekommt er Einsicht in die vollständigen Daten und bisherigen Untersuchungsergebnisse der Patientin. Es ist ihm nun auch möglich den α-Card Deskriptor, der ihm zugeordnet ist (Biopsie), in kleinere Arbeitspakete zu unterteilen, um ihn für die Arbeit zu strukturieren und die Befunde in kleinere Teile aufzuteilen. Außerdem hat er die Möglichkeit, leichter durchzuführende Aufgaben an seine Studenten zu delegieren. Dies verdeutlicht zwei wichtige Anwendungsaspekte des Subprozesses in Arbeitslisten:

- Die Möglichkeit der Aufteilung einer Aufgabe in kleiner Aufgabe
- Die erleichterte Delegation kleinerer Aufgaben

### Diskussion des Lösungskonzepts und Definition einer Subprozessbeziehung

Bis jetzt bietet der $\alpha$-Flow Ansatz keine Möglichkeit zur Modellierung von Subprozessen. Doch durch die Wichtigkeit des Konzeptes, besteht im DAB-Modell die Möglichkeit der Modellierung eines Subprozesses. Dieser Subprozess wird durch ein Kompositum beschrieben. Das Kompositum ist ein Eintrag in der normalen Arbeitsliste. Dieser Eintrag wird in der Arbeitsliste als normale $\alpha$-Card dargestellt. Der Eintrag ist jedoch nur Platzhalter für die Komponenten des Eintrags, denn das Kompositum besteht aus der Summe aller seiner Komponenten. Diese Komponenten werden in der Liste durch einen Aggregationspfeil (Pfeil mit Raute in Richtung Kompositum) mit dem Kompositum verkettet, um die Aggregationsabhängigkeit (Sublistenabhängigkeit) zu verdeutlichen. Sie stellen eine Subliste bestehend aus $\alpha$-Cards dar.

### Verknüpfung innerhalb einer Subliste

Diese Komponenten sind untereinander durch die schwache Implikation verknüpft, wodurch die Reihenfolge der Abarbeitung wieder durch ad hoc Entscheidungen verändert werden kann. Sie können jedoch auch durch eine starke Implikation verknüpft sein, um zu verdeutlichen, dass unter den Komponenten der Subliste eine strikte zeitliche Reihenfolge einzuhalten ist. Dies stellt die PRE-Eigenschaft der-PREANOR Verknüpfung dar. Die Sublistenabhängigkeit zwischen Kompositum ($\alpha$-Card der Oberliste) und Komponenten ($\alpha$-Cards der Subliste) basiert somit rein auf Datenabhängigkeit.

### Erfüllung aller Komponenten

Erst wenn die Komponenten der Subliste in einem Zustand vorliegen, der ausreichend ist (zur genaueren Definition des Zustandes im Abschnitt 7.7), ist das Kompositum als erfolgreich abgearbeitet zu kennzeichnen. Dies verdeutlicht die PREANOR-Verknüpfung, denn es ist möglich durch nachträgliches Streichen einer Komponente, das Erfüllen des Kompositums zu erreichen. Was genau das Erfüllen des Kompositums bedeutet, wird in Abschnitt 7.8 bzw. 7.7 beschrieben. In Abbildung 7.4 wird das Konzept des Subprozesses veranschaulicht und grafisch definiert.

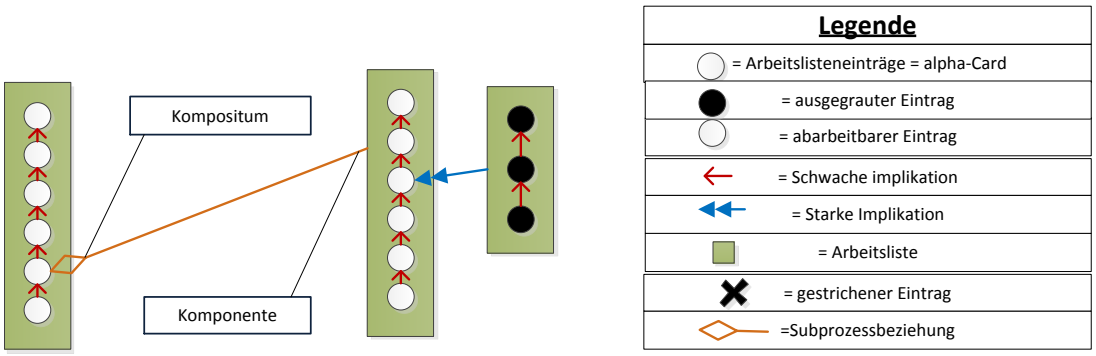



**Bild 7.4:** Darstellung der Subprozessbeziehung im α-Flow Ansatz

## 7.7 Prozessfortschrittszustand-Modell im DAB-Modell

Wie aus Tabelle 6.7 hervorgeht, bieten die bisher betrachteten Prozessmodellierungsnotationen verschiedene Ansätze an, um Zustände zu modellieren. Dabei ist der Artefaktlebenszyklus und der Aktivitätslebenszyklus zu unterscheiden. Der Erste spiegelt den Lebenszyklus des ganzen Prozesses wider und der Letztere den Zustand der einzelnen Aktivität. Bei den bisher betrachteten Ansätze wird der Zustand des gesamten Prozesses in den Vordergrund gestellt, da z.B. bei Petri Netzen durch die Stellen der Prozesszustand vor bzw. nach einer Transition abgebildet wird.

### Motivation

Zur Motivation des Prozessfortschrittszustandsmodelles wird folgende User Story angegeben: Wenn ein Arzt A mit einem Arzt B in einer komplexen Untersuchung kollaboriert, dann kann es sein, dass der Patient sich in Behandlung bei Arzt B befindet und Arzt A auf die Untersuchungsergebnisse von Arzt B wartet. Da beide Ärzte Sicht auf die gleichen Daten und Einträge in der Arbeitsliste haben, will Arzt A sehen, dass die Untersuchung von Arzt B bereits begonnen hat bzw. beendet ist. Auf der Gegenseite gibt Arzt B die Untersuchungergebnisse in seine α-Doc Kopie ein und setzt sie auf valid und public. Dadurch wird das Ergebnis für Arzt A sichtbar und er kann die Behandlung fortsetzen.

Des Weiteren interessiert beide Ärzte der Zustand des Gesamtcases, um zu sehen ob noch Einträge auf der Arbeitsliste zu erfüllen sind.

### Diskussion des Lösungskonzepts

Beim $\alpha$-Flow Ansatz werden diese beiden Lebenszyklen adäquat unterstützt. Es existiert ein Lebenszyklusmodell für den kompletten Case, der über ein regelbasiertes System Auskunft über den Zustand der $\alpha$-Episode gibt. Des Weiteren existieren durch Adornments auch Möglichkeiten zur Beschreibung von Zuständen für $\alpha$-Cards. Darauf wird im Folgenden genauer eingegangen.
Es besteht die Notwendigkeit drei Prozessfortschrittszustandsmodelle zu unterscheiden:

- Das Zustandsmodell des gesamten Cases
- Das Zustandsmodell einer Subliste von $\alpha$-Cards
- Das Zustandsmodell einer $\alpha$-Card

Die ersten beiden Prozessfortschrittszustandsmodelle basieren dabei auf dem Letzten, nur wird funktioniert dieses bisher nicht automatisch, sondern muss manuell bedient werden. So basiert der Zustand des gesamten Cases auf den Zuständen der einzelnen $\alpha$-Cards. Genauso ist der Zustand einer Subliste (Komponenten) mit dem Zustand der Ober-$\alpha$-Card (Kompositum) in Verbindung (Sublistenabhängigkeit) zu bringen. Für diese drei Prozessfortschrittszustandsmodelle ist es demnach nötig, zwei Marker einzuführen, die das Erfüllen eines Cases/einer $\alpha$-Card signalisieren. Der Marker für den Case ist SEALED und zeigt an, dass der Case schwach beendet ist. Schwach beendet bedeutet, dass es keine Garantie gibt, dass es weitergeht und der Case sich im Stillstand befindet. Der Marker für die $\alpha$-Card ist CLOSED. Er signalisiert, dass die $\alpha$-Card bearbeitet wurde und die starke Implikation schalten kann (es kann ent-ausgegraut werden, nicht mehr zurückgestellt). Ein Marker für Sublisten ist nicht nötig, denn er wirkt sich nur auf das Zustandsmodell der Ober-$\alpha$-Card aus.

### Definition des Lösungskonzepts

In diesem Abschnitt werden die drei Adornments definiert, die die jeweiligen Prozessfortschrittszustandsmodelle enthalten. Das Adornment für den gesamten Prozess ist im Process Structure Artifact verankern, da dort die Adornments des gesamten Prozesses angesiedelt sind. Die anderen beiden Adornments sind unabhängig für jede Subliste/$\alpha$-Card zu verwalten. Die Abbildung 7.5 zeigt das Zusammenspiel dieser drei Adornments.

### Adornment für den gesamten Case

Das Adornment für den gesamten Case wird *Case-Progression* genannt und beinhaltet vier Zustände:

- **open-ended**: Der Case ist offen und bereit zur Bearbeitung
- **ceased**: implizites Ende durch Befüllung aller α-Cards mit Payload (alle α-Cards in Zustand ceased). Drückt Stillstand aus. Case kann jederzeit weitergeführt werden.
- **sealed**: explizite Beendigung des Cases, kann jedoch wieder aufgenommen werden. Setzen des SEALED Marker.
- **closed**: Für immer beendet. Nicht rückgängig machbar. Explizit setzbar. Setzen des SEALED Marker.

### Adornment für eine Subliste

Das Adornment für die Subliste wird *Sublist-Progression* genannt und beinhaltet drei Zustände:

- **open-ended**: Die Subliste ist in Bearbeitung bzw. steht bereit zur Bearbeitung
- **ceased**: Die Subliste ist implizit beendet. Alle Container sind befüllt. Die Subliste ist im Stillstand. Signalisierung an Ober-α-Card Adornment, dass darauf in Zustand ceased übergeht.
- **finished**: explizit veranlasste Beendigung. Durch das Setzen aller α-Cards der Subliste auf valid/public. Signalisierung an Ober-α-Card Adornment, dass darauf in Zustand closed übergeht.

### Adornment für eine α-Card

Das Adornment für eine α-Card wird *Card-Progression* genannt und beinhaltet folgende vier Zustände:

- **open-ended**: α-Card bereit zur Bearbeitung und Befüllung.
- **started**: Dieser Zustand wenn α-Card Kompositum von Komponenten in einer Sublistenbeziehung. Mindestens eine α-Card der Subliste mit Payload befüllt.
- **ceased**: Stillstand. Implizites Ende. Alle α-Cards der Subliste mit Payload befüllt (Sublistenbeziehung) bzw. die α-Card selbst mit Payload befüllt (keine Sublistenbeziehung). Marker auf CLOSED setzen.

- **closed**: explizites Beenden der $\alpha$-Card durch finished aller $\alpha$-Cards der Subliste (Sublistenbeziehung) oder valid/public setzen des Payloads der $\alpha$-Card (keine Sublistenbeziehung)

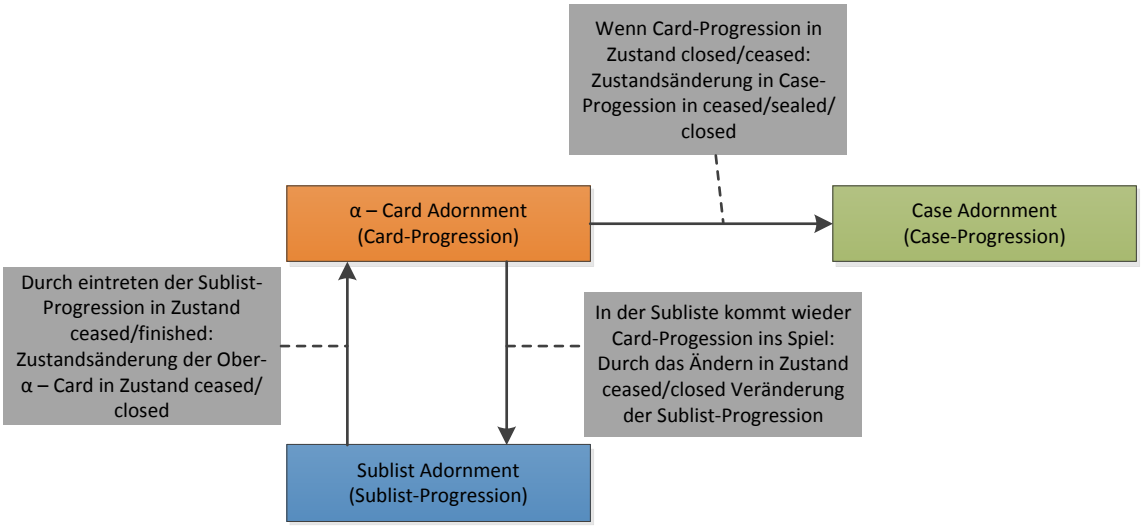



**Bild 7.5:** Veranschaulichung des Zusammenspiels der drei Prozessfortschrittszustandsmodelle

Eine genauere Betrachtung der Zustände und deren Notwendigkeit wird im nächsten Abschnitt gezeigt.

## 7.8 Beendigungen im DAB-Modell

Das Beendigen von Prozessen wird laut Tabelle 6.6 von allen betrachteten Notationen unterstützt. Es werden sogar oft mehrere Möglichkeiten der Beendigung bereit gestellt, es gibt die Möglichkeit einen Teilzweig eines Prozessmodells zu beenden oder es gibt die Möglichkeit den kompletten Prozess zu beenden. Dies wird mit Hilfe von Kontrollfluss modelliert, da man vorher wissen muss, wann der Prozess zu Ende sein wird, um dies explizit zu modellieren.

### Motivation

In $\alpha$-Flow gibt es drei Fragestellungen, die auf das Thema Beendigungen abzielen:

- Wann werden bei einer starken Implikation die zurückgestellten α-Cards ent-ausgegraut und nicht mehr zurückgestellt?
- Wann ist bei einer Sublistenbeziehung die Ober-α-Card erfüllt?
- Wann ist der gesamte Case als beendet anzusehen?

Zur Veranschaulichung dieser drei Fragestellungen sei folgende User Story angegeben: Der Arzt, der seine Aufgabe in mehrere Arbeitspakete aufgeteilt hat, läd die Befunde, die er bei der Biopsie gesammelt hat, nacheinander hoch. Dazu fügt er jedem α-Card Deskriptor Payload hinzu. Durch das Hinzufügen des Payloads verändert sich der Status der α-Cards. Nachdem er alle Befunde in den Case geladen hat, überprüft er diese noch einmal. Anschließend kann er seine Befunde als gültig und sichtbar markieren. Dadurch werden seine Befunde für alle anderen Prozessteilnehmer sichtbar. Des Weiteren wird durch die Abarbeitung seiner selbst definierten Arbeitspakete, das vom Gynäkologen definierte Arbeitspaket Biopsie als closed angezeigt und sind damit beendet. Damit ist das komplette Kompositum als beendet zu sehen. In dieser Situation wird das Zusammenspiel der Prozessfortschrittszustandsmodelle von Sublisten und Ober-α-Cards verdeutlicht.

**Diskussion des Lösungskonzepts und Definition der Lösung**

Die Antworten auf die obigen Fragestellungen können durch die vorgestellten Prozessfortschrittszustandsmodelle gegeben werden, denn bisher unterstützt α-Flow das Beenden eines Cases nicht. Dies soll nun aber ermöglicht werden, weshalb in diesem Abschnitt der Lösungsansatz beschrieben werden soll. Die starken Implikationen können schalten (die zurückgestellten α-Cards können ent-ausgegraut werden und nicht mehr zurückgestellt werden), wenn der CLOSED-Marker der α-Card gesetzt ist. Die Ober-α-Card kann als erfüllt angesehen werden wenn die Subliste in Zustand ceased bzw. finished ist. Ceased führt jedoch nur zu einer schwachen Beendigung. Finished führt zu einem explizitem closed. Dadurch wird für Ober-α-Card der Marker CLOSED gesetzt. Dies wiederum hat Einfluss auf die Beantwortung der letzen Fragestellung. Denn nur wenn für alle α-Cards der Marker CLOSED gesetzt ist, lässt sich auch der Marker CEASED für den gesamten Case setzen. Der Case lässt sich implizit und explizit beenden. Implizit bedeutet hier, dass der Case sich im Stillstand befindet, jedoch jederzeit weitergeführt werden kann. Explizit bedeutet, dass der Case explizit geschlossen wird. Diese explizite Beendigung wird jedoch in zwei Kategorien unterschieden: sealed und closed. Sealed lässt eine Aufnahme des Cases im weiteren Verlauf zu, wobei Closed eine unwiderrufliche Beendigung ist. Der Case lässt sich nur durch Beginnen einer neuen α-Episode von

Neuem beginnen. Durch dieses komplexe System lassen sich alle Kombinationen von Sublisten und strikten Reihenfolgen modellieren (Sublisten in starken Implikationen und starke Implikationen in Sublisten, siehe Abbildung 7.4). Dadurch basiert auch das Beenden von Cases/$\alpha$-Cards rein auf Datenabhängigkeiten.

## 7.9 Möglichkeiten für Laufzeitänderungen

Wie in Tabelle 6.9 gesehen, sind Änderungen während der Laufzeit des Prozesses bei bisher betrachteten Prozessmodellierungsformen nicht unterstützt. Die Ausnahme bildet hier ADEPT, siehe Abschnitt 3.3.2.

### Motivation

Zur Motivation für Laufzeitänderungen wird eine User Story angegeben: Wenn ein Arzt A einen Patienten behandelt und seinen Behandlungsverlauf durch seine Arbeitsliste spezifiziert, dann hat er durch die PREANOR-Verknüpfung implizit alle Einträge der Arbeitsliste als notwendig gekennzeichnet. Durch die PREANOR-Verknüpfung ist er auch in der Lage bestimmte Einträge wieder zu streichen, wenn er Untersuchungen durch ad hoc Entscheidungen nicht mehr wichtig empfindet. Wenn Arzt A jedoch im Behandlungsverlauf feststellt, dass er im Vornherein von einer falschen Krankheit ausgegangen ist und seinen Behandlungsverlauf teilweise umstellen will, benötigt er die Möglichkeit dies zur Laufzeit zu tun. Er will neue Untersuchungen und Behandlungen zu seiner Arbeitsliste hinzufügen. Dies kann er durch das Hinzufügen von $\alpha$-Cards.

### Diskussion des Lösungskonzept und Definition der Lösung

$\alpha$-Flow unterstützt die Änderung des Prozesses während der Laufzeit, diese Möglichkeit wird in diesem Abschnitt vorgestellt. Wie beschrieben wird dies durch die schwache Implikation und der daraus resultierenden Möglichkeit der Umstrukturierung der Abarbeitungsreihenfolge der Einträge in der Arbeitsliste modelliert. Des Weiteren wird durch die retrospektive OR-Verknüpfung erreicht, dass das Streichen einer Aktivität möglich ist, wieder getrieben durch ad hoc Entscheidungen der Prozessteilnehmer. Außerdem ist es möglich, während der Laufzeit zusätzliche Einträge in der Arbeitsliste hinzuzufügen. Dies geschieht durch Hinzufügen von vorher nicht spezifizierten Daten bzw. Befunden. Dabei ist es möglich zur Laufzeit neue $\alpha$-Card Deskriptoren anzulegen und damit entsprechend

mit den neuen Daten zu interagieren. Diese Interaktion besteht aus Befüllen der α-Card Deskriptoren. Hinzufügen von α-Card Deskriptoren bedeutet hinzufügen von neuen Einträgen in die Arbeitsliste, die sich wieder durch eine PREANOR-Verknüpfung auszeichnen.

## 7.10 Gesamtüberblick über das DAB-Modell

Das DAB-Modell lässt sich in das inhaltsbasierte Paradigma einordnen, weshalb die Daten in diesem Modell im Vordergrund stehen. Durch sie wird auch der Prozessfortschritt freigeschalten. Dies wird mit einer Arbeits- bzw. Priorisierungsliste veranschaulicht. Kontrollfluss wird im DAB-Modell nicht modelliert. Es gibt dennoch die Möglichkeit temporale Abfolgen zu modellieren. Dazu werden zwei verschieden Arten von Implikationen angeboten. Auf der einen Seite die schwache Implikation und auf der anderen Seite die starke Implikation. Mit Hilfe der schwachen Implikation kann ausgedrückt werden, dass implizit eine Reihenfolge in der Priorisierungsliste vorliegt, diese jedoch auf Grund von ad hoc Entscheidungen umgeändert werden kann. Die starke Implikation gibt nun strikte temporale Ordnung an, im Sinne von: Erst wenn A, dann B, wobei B auch wieder eine Liste sein kann. Dies wird dem Benutzer in der Priorisierungsliste durch das Konzept des Zurückstellens verdeutlicht. Entscheidungen werden im DAB-Modell durch die PREANOR-Verknüpfung gehandhabt. Diese ist eine Verbindung aus einem schwachen AND, einem retrospektiven OR und den beiden Implikationen. Das schwache AND bedeutet, dass vor dem Prozess alle Einträge in der Priorisierungsliste bearbeitet werden sollen. Das retrospektive OR zeigt, dass im Nachhinein aber doch nicht alle Schritte ausgeführt werden mussten (durch z.B. ad hoc Entscheidungen). Die beiden Implikationen spielen wieder die oben beschriebenen Rollen. Durch diese PREANOR-Verknüpfung wird demnach ein komplexes Zusammenspiel der Einträge in der Priorisierungsliste beschrieben. Subprozesse lassen sich im DAB-Modell modellieren, durch ein Kompositum in der Priorisierungsliste des Prozesses. Das Zustandsmodell des DAB-Modells regelt mittels Lebenszyklusmodellen die Beendigung und Fortschaltung des kompletten Cases, eines Subpozesses oder einer α-Card. Dies steht auch in Verbindung mit der starken Implikation und von Subprozessen, weil nur dadurch entschieden werden kann, wann die strikte zeitliche Abfolge eingehalten wurde bzw. ein Subprozess als beendigt angesehen werden kann. Diese Konzepte können kombiniert werden um komplexe Prozesssituationen zu modellieren. Zum Beispiel strikte zeitliche Abfolgen innerhalb von Subprozessen. Des

Weiteren werden im DAB-Modell Möglichkeiten zu Laufzeitänderungen erfasst, indem während der Laufzeit neue Arbeitslisteneinträge hinzugefügt werden können.

## 7.11 Erweiterungspotential des Modells

Der hier betrachtete Ansatz des DAB-Modells schließt die Rolle der Benutzer aus. Dieser Aspekt muss getrennt betrachtet werden, da die Verteilung von Rollen und $\alpha$-Card-Besitzern konkrete Auswirkungen auf das hier beschriebene Zustandsmodell haben kann. Außerdem wird bei der Betrachtung der Aspekt der Fehlerbehandlung außen vor gelassen. Denn auch dies hätte Einfluss auf das Zustandsmodell bzw. auf die Beendigung von Case/$\alpha$-Card. So wird hier stillschweigend davon ausgegangen, dass das Bearbeiten von $\alpha$-Cards immer erfolgreich abläuft. Würde dies nicht der Fall sein, müsste man sich zum Beispiel mit der Wiederholung von Arbeitslisteneinträgen beschäftigen, denn eine $\alpha$-Card könnte in den verschiedenen Zuständen der Bearbeitung sein (continuing, succeeded, failed).

## 7.12 Zusammenfassung

In diesem Kapitel wurde das DAB-Modell von $\alpha$-Flow vorgestellt. Dabei wurde auf die Einordnung in ein Paradigma, auf die Modellierung von Daten, auf die Beherrschung von Kontrollfluss, auf die Beherrschung von zeitlichen Abfolgen, auf die Handhabung von Entscheidungen, auf das Modellieren von Subprozessen, auf die Modellierung von Zuständen, auf das Beendigen von Prozessen und auf die Möglichkeit zur Laufzeitveränderung eingegangen. Abschließend wurde eine Gesamtübersicht des DAB-Modells, sowie das Erweiterungspotential des Modells angegeben. Mit Hilfe dieses Modells ist es möglich komplexe Modellierung nur auf Grund von Datenabhängigkeiten zu modellieren. Im Vergleich zu der Gegenüberstellung in Kapitel 6 lässt sich somit festhalten, dass das DAB-Modell in der Lage ist, die gleiche Aussagekraft wie die vorgestellten Prozessmodellierungsnotationen zu erreichen, alleine auf Grund von Datenabhängigkeiten. Dieser Befund wird im Kapitel 8 veranschaulicht und anhand der Abbildung auf die vorgestellten Evaluierungsframeworks untermauert.

# 8 Evaluation

In diesem Kapitel sollen die Ergebnisse, die in den Kapiteln 6 und 7 erarbeitet wurden evaluiert werden. Dies wird mit Hilfe der beiden in Abschnitt 3.5 vorgestellten Evaluierungsframeworks geschehen. Die zu Grunde liegenden Tabellen der Abbildungen sind im Anhangskapitel A zu finden. Dazu wurde der $\alpha$-Flow Ansatz sowohl gegen das Workflow Pattern Framework, als auch gegen das BWW-Modell abgebildet.

### Abbildung von $\alpha$-Flow auf die beiden Evaluationsframeworks

Die Abbildung auf das Workflow Pattern Framework wurde durchgeführt, indem die 20 ursprünglichen Muster ausgewählt wurden und die in [RHAN11] dargestellten Erfüllungskriterien überprüft wurden. Dies ist der Abbildung 8.1 zu entnehmen. Auffällig dabei ist, dass $\alpha$-Flow nur ein Muster der Workflow Pattern komplett unterstützt (Sequenz) und eines halb (verzögerte Entscheidung). Die Sequenz wird unterstützt, da das Erfüllungskriterium darin besteht, dass die Notation zwei Aufgaben in eine zeitliche Reihenfolge bringen kann, dies erfüllt die starke Implikation. Die verzögerte Entscheidung gilt als teilweise erfüllt, da es möglich ist, während der Prozesslaufzeit die Entscheidung für den weiteren Prozessverlauf zu treffen, aber es keinen Wettstreit zwischen mehreren Alternativen gibt. Das schlechte Abschneiden ist darin begründet, dass die Workflow Pattern ihren Schwerpunkt auf den Kontrollfluss legen. Der $\alpha$-Flow Ansatz jedoch, wie in Kapitel 7 beschrieben, nicht. Bei der Abbildung auf das BWW-Modell wurden die 29 Konstrukte, die in der verwandten Literatur Anwendung gefunden haben, verwendet und überprüft ob es eine Abbildung auf $\alpha$-Flow gibt. Dies ist der Abbildung 8.2 zu entnehmen. Zur Abbildung auf das BWW-Modell lässt sich sagen, dass $\alpha$-Flow viele Konstrukte unterstützt, aber da $\alpha$-Flow auf ad-hoc Entscheidungen setzt und die Prozesse nicht vorstrukturiert, ist eine Definition der BWW-Kriterien von erreichbaren Zustands/Ereignisräumen nicht möglich, dazu müsste $\alpha$-Flow die vollständige Definition aller Zustände und Ereignisse zulassen, die während der Benutzung erreichbar bzw. eintretbar sind. Dies ist wegen der Flexibilität und der Erweiterbarkeit nicht gegeben. Genauso wenig wird das BWW-Kriterium der Angabe von wohldefinierten Ereignissen unterstützt, denn dies würde bedeuten, dass

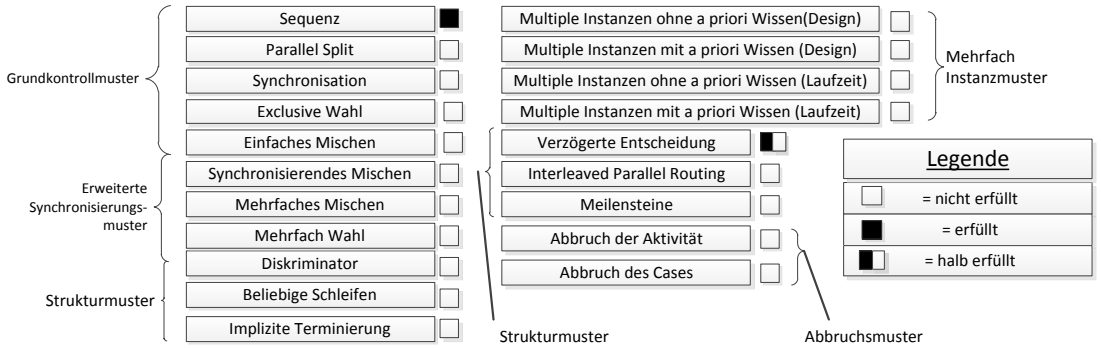



**Bild 8.1:** Evaluierung von $\alpha$-Flow mit Hilfe der Workflow Pattern

man wüsste was der nächste Prozesszustand ist. Dies würde einer Vorstrukturierung gleich kommen, was auf Grund der inhaltsbasierten Schwerpunktsetzung von $\alpha$-Flow nicht beabsichtigt ist. Systemzerlegung und -zusammensetzung sind in diesem Sinne auch nicht definiert, aber auch nicht nötig, da die Unterteilung in $\alpha$-Card und $\alpha$-Doc bereits verschiedene Granularitäten definiert. Um die BWW-Kriterien Systemzerlegung und -zusammensetzung zu erfüllen, müsste jedes Subsystem des Prozesses wieder ein eigener Prozess sein. Dies ist im Falle von $\alpha$-Flow aber nicht gegeben, da nur ein $\alpha$-Doc als Prozessinstanz fungiert, eine $\alpha$-Card jedoch nicht.

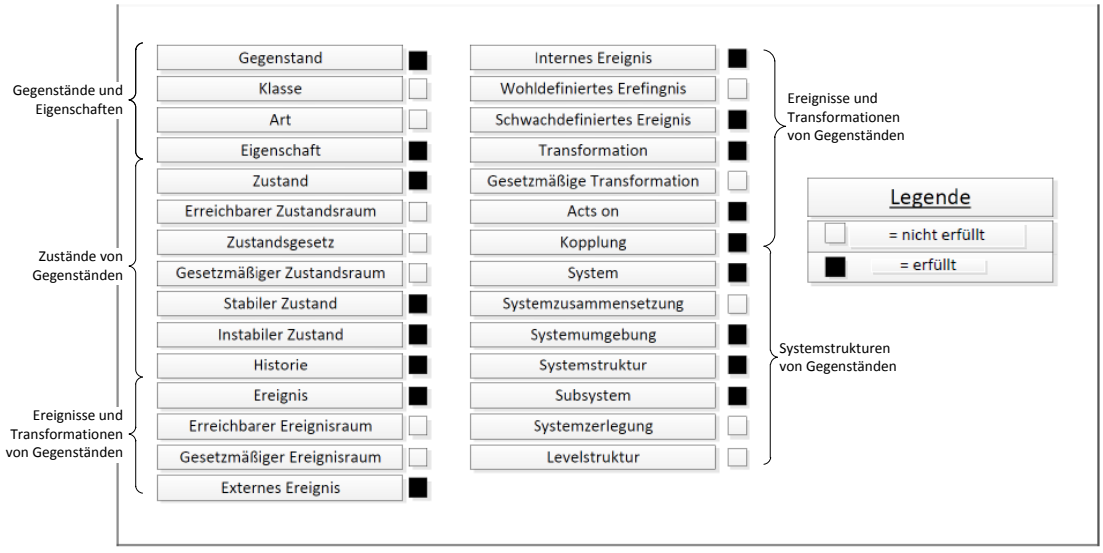



**Bild 8.2:** Evaluierung von $\alpha$-Flow mit Hilfe des BWW-Modells

### Abbildung aller Notationen auf das BWW-Modell

In Abbildung 8.3 sieht man, dass somit der $\alpha$-Flow Ansatz im Vergleich zu den anderen Notationsarten gut abschneidet. Man sieht damit, dass das BWW-Modell seinen Schwerpunkt nicht auf die Art der Verkettung legt, sondern nur auf die generelle Mächtigkeit der Darstellungsmöglichkeiten. Im Vergleich zu Petri-Netzen [RRIG09], EPKs [RRIG09] und UML ADs [OHS02] weist $\alpha$-Flow mit 18 von 28 unterstützten Konstrukten teils deutlich mehr als die Konkurrenz auf: 11 (EPK), 12 (Petri-Netze und UML ADs). Nur BPMN (19) [RRIG09] ist auf dem gleichen Niveau wie $\alpha$-Flow anzusiedeln.

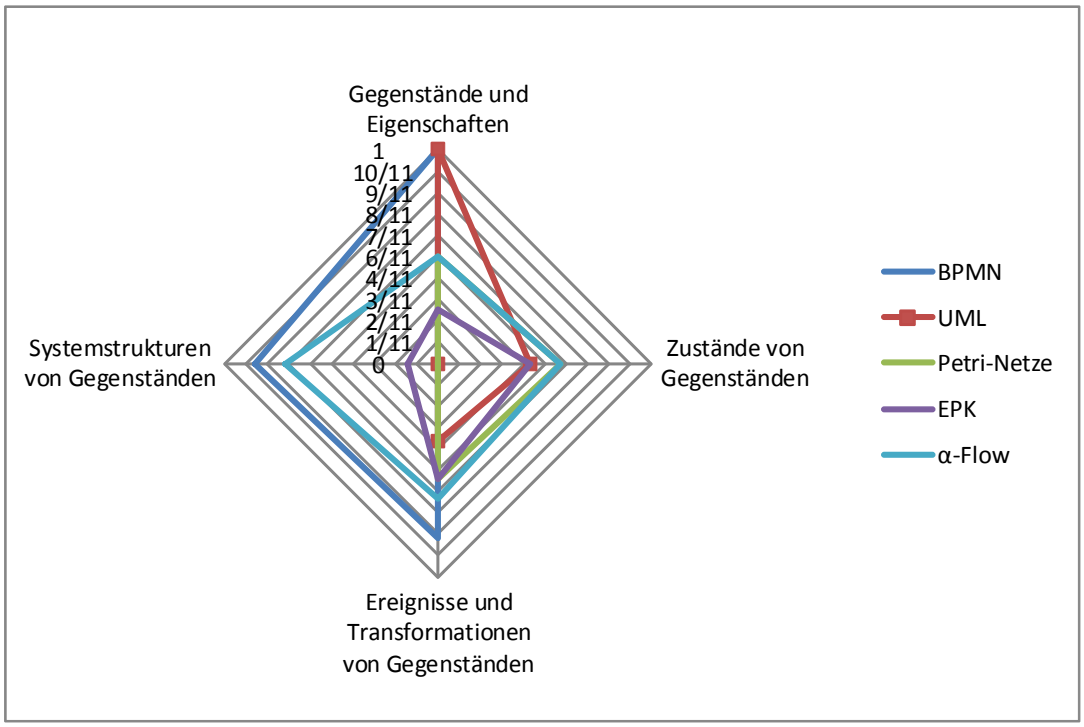



**Bild 8.3:** Gegenüberstellung aller betrachteten Notationen mit Hilfe des BWW-Modells

### Abbildung aller Notationen auf das Workflow Pattern Framework

In Abbildung 8.4 ist die Gegenüberstellung der betrachteten Notationen und $\alpha$-Flow mit Hilfe der Workflow Patterns angegeben. Wie man hier leicht sieht, schneidet hier $\alpha$-Flow mit weitem Abstand am schlechtesten ab. Alle anderen inhaltsbasierten Ansätze würden genauso schlecht abschneiden, denn inhaltsbasierte Notationen bieten meist keine Unterstützung für Kontrollflusskonstrukte an und die Workflow Pattern beschränken

sich nur auf die Analyse hinsichtlich von Kontrollflusskonstrukten. Deshalb ist es nicht verwunderlich, dass die aktivitätsbasierten Notationen diese Kriterien besser erfüllen. $\alpha$-Flow bietet Unterstützung für 1,5 von 20 Mustern, wobei die EPKs neun [MNN05], Petri Netze 14 [AHKA03], UML ADs 16 [WAD+06] und BPMN 17 [WAD+06] Muster unterstützen.

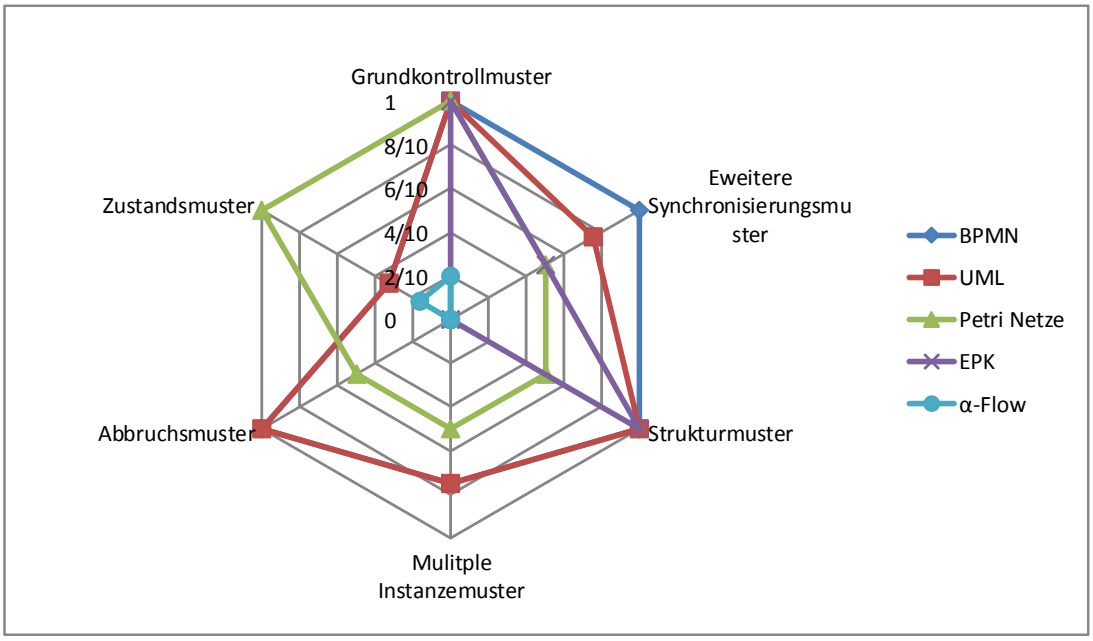



**Bild 8.4:** Evaluierung aller betrachteten Notationen mit Hilfe der Workflow Patterns

### Einführung der Datenperspektive

Aus dem Grund der Schwerpunktsetzung des Workflow Pattern Framworks wird eine neue Dimension eingeführt, um die Ausdrucksmächtigkeit sowohl nach aktivitätsbasierten als auch nach inhaltsbasierten Kriterien objektiv gegenüberzustellen: Die Datendimension. Diese Dimension erfasst neun typische Kriterien für inhaltsbasierte Ansätze:

1. Ändern der Reihenfolge bestimmter Aktivitäten ist möglich
2. Unterstützung von ad-hoc Entscheidungen ist gegeben, mit deren Hilfe man Aktivitäten streichen bzw. hinzufügen kann
3. Prozessfortschritt wird durch Daten erreicht
4. Datenproduktion wird beschrieben

5. Entscheidungen können auf Grund von Daten getroffen werden
6. integrierte Sicht auf alle Daten ist gegeben
7. open-end Prozess beschrieben kann werden[1]
8. datenbasiertes Zustandsmodell ist gegeben
9. Daten können zerlegt werden

Mit Hilfe dieser Kriterien lassen sich Modelle erstellen, die genauso viel Aussagekraft besitzen wie aktivitätsbasierte Notationen, nur durch Rückwärtsverkettung anstatt Vorwärtsverkettung. Eine Abbildung der betrachteten Notationen auf die neun Kriterien der Datenperspektive wurde durchgeführt, indem versucht wurde, zu jedem Kriterium eine Abbildung zu einem Konstrukt der jeweils betrachteten Notation zu finden. Diesbezüglich ist in Abbildung 8.5 die Abbildung der betrachteten Notationen auf diese Dimension angegeben. Wie man sieht erfüllt $\alpha$-Flow alle Kriterien, während die übrigen Ansätze wenige Konstrukte vollständig unterstützen, was nicht verwunderlich ist, da diese Perspektive, im Gegensatz zu den Workflow Pattern ihren Schwerpunkt auf Datenabhängigkeiten setzt und nicht auf Kontrollflusskonstrukte. $\alpha$-Flow besitzt durch das DAB-Modell alle Voraussetzungen, die Kriterien zu erfüllen. Die Zerlegung von Daten kann auf Grund der Subprozessbeziehung geschehen. Die Änderung der Reihenfolge und das ad hoc Streichen bzw. Hinzufügen wird durch die schwache Implikation und der Möglichkeit zu Prozessänderung zur Laufzeit ermöglicht. Der Prozessfortschritt lässt sich durch Daten erreichen und es wird ein Datenproduktionsprozess beschrieben, weil durch die Bearbeitung des Cases immer neue Daten entstehen und Fortschritt freischalten. Entscheidungen können auf Grund von Daten getroffen werden, dies ist sogar ein integraler Bestandteil des $\alpha$-Flow Ansatzes, da es sonst ad hoc Entscheidungen nicht möglich wären. Für diese ad hoc Entscheidungen ist eine Gesamtsicht auf die Daten genauso wichtig. Durch das Prozessfortschrittzustand Modell ist es möglich Open-end Prozesse zu definieren, dies ist auch eine wichtiger Ansatzpunkt des $\alpha$-Flow Ansatzes, denn es ermöglicht, dass die Daten für lange Zeit behalten werden können und der Case zu späterer Zeit wieder aufgenommen werden kann. Das Prozessfortschrittzustand Modell fungiert gleichzeitig als datenbasiertes Zustandsmodell.
Die einzige andere Notation, die drei Konstrukte vollständig unterstützt sind UML ADs. Dies ist darin begründet, dass sie als einzige Notation expliziten Datenfluss modellieren kann. Somit kann man Prozessfortschritt durch Daten erreichen, Datenproduktion

1 Es muss keine Garantie auf eine Beendigung des Cases gegeben werden.

beschreiben und Entscheidungen können auf Grund von Daten (Objekttoken) getroffen werden. Datenbasierte Zustandsmodelle werden von BPMN, EPK und den ADs teilweise unterstützt, weil sie rudimentäre Zustände á la [started], [finished] kennen. Datenproduktion wird von EPKs und BPMN teilweise unterstützt, da hier Daten als Output angegeben werden können, aber es unklar ist, ob diese Daten wirklich vorliegen **müssen**, wenn diese Aktivität abgearbeitet ist. Somit lässt sich bezüglich der Datenperspektive festhalten, dass in dieser Perspektive inhaltsbasierte Notationen große Stärken haben und diese Perspektive im starken Kontrast zu den Workflow Pattern steht. Denn die Mächtigkeit der Workflow Pattern wird durch die Datenperspektive genauso erreicht, nur aus aktivitätsbasierten Blickwinkel anstatt aus inhaltsbasierten Blickwinkel.

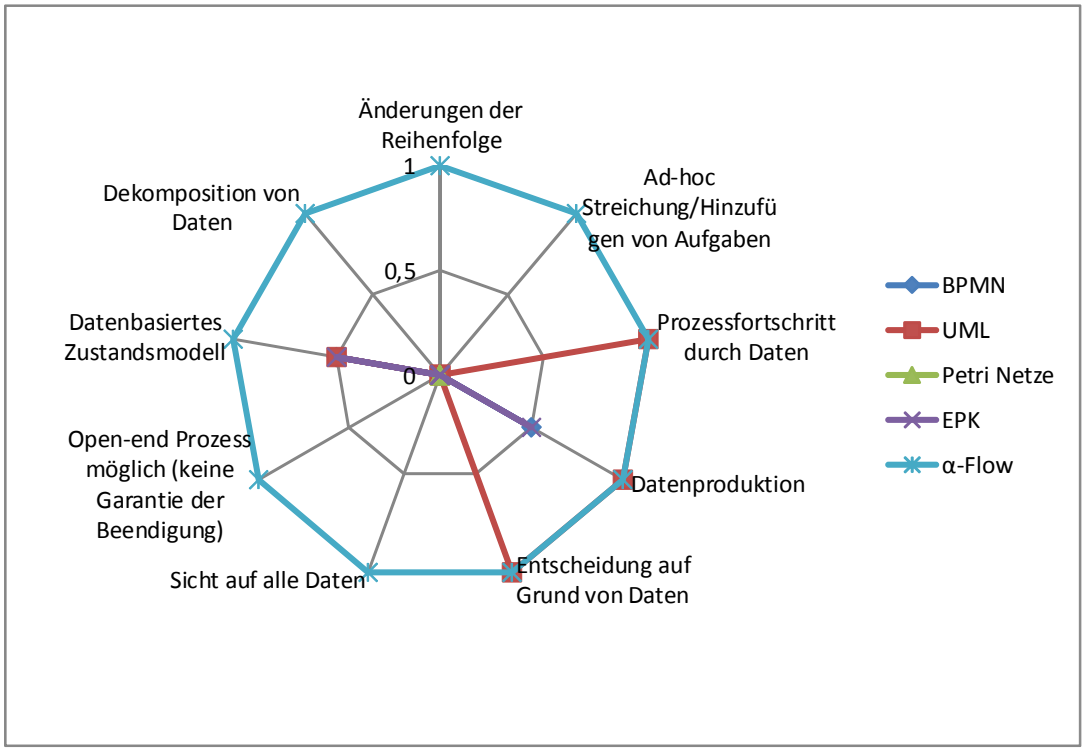



**Bild 8.5:** Evaluierung der betrachteten Ansätze mit Hilfe der Datenperspektive

### Gesamtdarstellung der Ausdrucksmächtigkeit

Um damit ein komplettes Bild der Situation zu generieren, zeigt Abbildung 8.6 die Gegenüberstellung der Workflow Pattern. Zusätzlich wurde in 8.6, als neue Perspektive die Datenperspektive eingezogen. Die Intention dieser Dimensionserweiterung ist, dass

damit veranschaulicht wird, dass es möglich ist allein durch Datenabhängigkeiten bzw. Rückwärtsverkettung die gleiche Ausdrucksmächtigkeit zu erreichen, wie dies durch aktivitätsbasierte bzw. vorwärtsverkettete Ansätze möglich ist. Damit ist eindeutig zu erkennen, dass $\alpha$-Flow im Speziellen und inhaltsbasierte Ansätze im Generellen in Hinsicht auf Kontrollfluss wenig bis keine Aussagekraft besitzen, diese Schwäche jedoch durch die Datenperspektive und die dadurch erreichte Aussagekraft ausgleichen kann. Als Ergebnis der Evaluation lässt sich deshalb festhalten, dass $\alpha$-Flow in der Lage ist die Aussagemächtigkeit der anderen betrachteten Notationen auf Grund des DAB-Modells bzw. $\alpha$-Flow im Allgemeinen durch Datenabhängigkeiten zu erreichen. Dass dies die gleiche Aussagekraft ist, die die anderen Notationen anbieten, lässt sich damit begründen, dass das DAB-Modell auf Grund von typischen Anforderungen und Konstrukten der aktivitätsbasierten Ansätze abgeleitet wurde. Damit ist sichergestellt, dass auch die gleiche Aussagekraft für die gleichen Anforderungen erreicht wird.

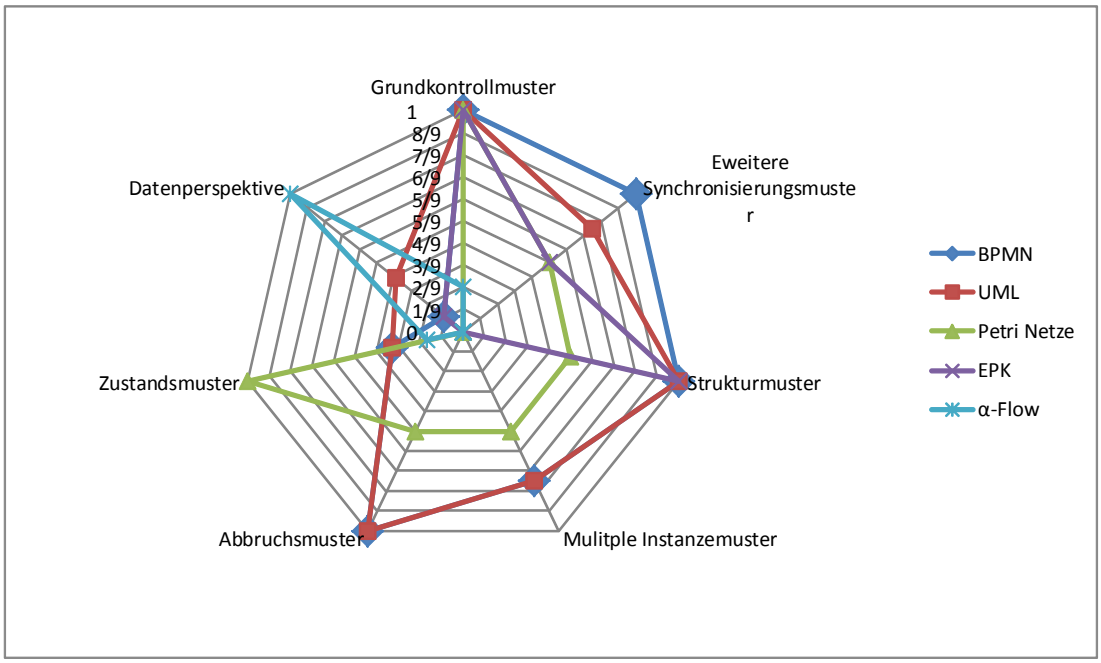



**Bild 8.6:** Evaluierung aller betrachteten Ansätze mit Hilfe der Workflow Patterns und der Datenperspektive

# 9 Ausblick und Zusammenfassung

In diesem Kapitel wird in Abschnitt 9.1 ein Ausblick auf die weiteren Möglichkeiten zur Forschung gegeben. In Abschnitt 9.2 wird eine Zusammenfassung der Arbeit angegeben.

## 9.1 Ausblick

In dieser Arbeit wurde das inhaltliche Paradigma, mit $\alpha$-Flow als Referenz, mit den aktivitätsbasierten Ansätzen verglichen. Diese beiden Ansätze grenzen sich ab von einem dritten Ansatz: Groupware. Durch die Analyse dieses Ansatzes, können weitere Schlüsse hinsichtlich der Modellierung von Prozessen im $\alpha$-Flow Ansatz ergeben. Im folgenden Abschnitt wird eine kurze Darstellung des Ansatzes der Groupware gegeben.

#### Grpupware als Paradigma der Prozessmodellierung

Groupware wird in einem eigenen Forschungsgebiet untersucht: *Computer-Supported Cooperative Work* (Computer-Supported Cooperative Work (CSCW)). [EGG91].
Die meisten Software Systeme unterstützen nur die Interaktion von Benutzern mit dem Computer. Aber die Interaktion mit anderen Benutzern ist genauso wichtig. Deshalb wird die Wichtigkeit von Kommunikation, Zusammenarbeit und Koordinierung von Benutzern betont [EGG91]. Daraus wird nun eine Definition von Groupware aus [EGG91]abgeleitet.

**Groupware** sind Computer basierte Systeme, die Gruppen von Personen unterstützen, die eine gemeinsame Aufgabe haben und die eine Schnittstelle für eine gemeinsame Umgebung anbieten.

Groupware ist datengetrieben, weil es sich rein auf den Austausch von Daten konzentriert. Groupware ist darüber hinaus auch unstrukturiert, da es keinerlei Vorgaben gibt, die den Austausch von Datenobjekten (z.B. Nachrichten) zwischen den Benutzern des Groupware Systems beschränkt [AH05] (siehe Bild 3.3).
Groupware als Prozessmodellierungsparadigma stellt somit die kooperative Arbeit verschiedener Prozessteilnehmer in den Vordergrund. Eine Reihenfolge der Aktivitäten,

die zu erledigen sind, wird nicht vorgegeben.

Es stellt sich nun die Frage, inwieweit sich Gemeinsamkeiten mit dem Groupware Paradigma und $\alpha$-Flow finden lassen. Eine Betrachtung hinsichtlich verschiedener Prozessunterstützungssysteme einschleißlich $\alpha$-Flow, Groupware und andere aktivitätsorientierte Ansätze findet sich in [Kre11].
Des Weiteren ist zu untersuchen, inwieweit sich die Konzepte der Feature-Oriented Domain Analysis (Feature-Oriented Domain Analysis (FODA)) auf den $\alpha$-Flow Ansatz beziehen [CHE04],[KCH$^+$90]. Der FODA-Ansatz strukturiert Komponenten in einer baumartigen Struktur und verknüpft dabei die verschiedenen Ebenen des Baumes mit Hilfe verschiedener Verknüpfungsoperatoren. Damit kann er Aufschluss über die Strukturierung der Priorisierungsliste geben. Dieser Ansatz steht in Verbindung mit dem DAB-Modell, weil Strukturierung der Arbeitsliste auf einer FODA ähnlichen Struktur basiert. Durch die Hierarchiebildung in der Arbeitsliste wird die Baumstruktur explizit gemacht. Die PREANOR-Verknüpfung steht demnach in eindeutiger Beziehung zu einem FODA-Baum, der eine OR-Verknüpfung anbietet. Wobei man hier unterscheiden muss, dass im FODA-Baum Beziehungen als mandatory gekennzeichnet werden können. Dadurch ist eine explizite Modellierung von einer PREANOR-Verknüpfung möglich, einzig die prospektive Komponente ist nicht gegeben, da man von vornherein die Wahl hat zu streichen.

## 9.2 Zusammenfassung

Es bestehen viele verschiedene Ansätze zur Modellierung von Prozessen. Diese Ansätze zielen auf verschiedene Anforderungen der Prozessmodellierung ab. Die zwei grundlegenden Paradigmen sind das aktivitätsbasierte Paradigma und das inhaltsbasierte Paradigma. Das aktivitätsbasierte Paradigma stellt die Aktivitäten in den Mittelpunkt, während das inhaltsbasierte Paradigma die Daten in den Mittelpunkt stellt. Daraus entstehen große semantische und syntaktische Unterschiede. Diese Unterschiede spiegeln sich auch in der Aussagekraft der einzelnen Ansätze bzw. der Eignung für bestimmte Anforderungen wieder.
Im medizinischen Umfeld basiert der Behandlungsprozess durch den Arzt auf ad-hoc Entscheidungen, die die Gesundheit des Patienten sicherstellen sollen. Deswegen stoßen hier die aktivitätsbasierten Ansätze an ihre Grenzen, denn sie strukturieren den Prozess

von Anfang an komplett.
$\alpha$-Flow ist ein Ansatz, das beruhend auf aktiven Dokumenten, eine Prozessunterstützung im medizinischen Umfeld darstellen soll. $\alpha$-Flow lässt sich in das inhaltsbasierte Paradigma einordnen.
Durch eine umfassende Literaturrecherche wurden in dieser Arbeit ausgewählte aktivitätsbasierte Ansätze mit $\alpha$-Flow als Referenz verglichen.Die Literaturrecherche sollte Aufschluss über die Möglichkeiten zur Modellierung für Kontrollfluss und Datenfluss geben. Die ausgewählten Notationen waren: BPMN, UML ADs, Petri Netze, EPKs. Außerdem wurden Ideen von PHILharmonicFLows und ACM behandelt. Außerdem sollte die Herkunft, die Einsatzbereiche und grundsätzliche Eigenschaften der Notationsarten vorgestellt werden. Darauf aufbauend wurde anhand eines Beispielprozesses gezeigt, welche Stärken und Schwächen die Notationen besitzen. Es wurden typische Anforderungen an aktivitätsbasierte Notationen herausgearbeiten und mit $\alpha$-Flow verglichen. Daraufhin wurden die Umsetzungsmöglichkeiten in $\alpha$-Flow diskutiert. Für diesen Zweck wurde eine Einteilung in die verschiedenen Paradigmen vorgenommen, es wurden die Darstellungsmöglichkeiten für Nebenläufigkeit bzw. temporale Abfolge von Aktivitäten untersucht, die Darstellungsarten von Entscheidungen verglichen. Außerdem wurden die Möglichkeiten zur Modellierung von Kontrollfluss und Daten erarbeitet, die Varianten der Beendigung von Prozessen dargestellt und die Bereitstellung von Zuständen durch das Modell aufgezeigt. Des Weiteren wurden die Möglichkeiten für Subprozesse dargestellt. Ob eine Möglichkeit zur Änderung am Prozess zur Laufzeit besteht, war das letzte Kriterium.
Das Ergebnis der Arbeit stellt das DAB-Modell dar. Dieses Modell beschreibt, wie der $\alpha$-Flow Ansatz zur Prozessmodellierung im medizinischen Kontext, basierend auf Datenabhängigkeiten bzw. Rückwärtsverkettung, möglich ist. Dazu wurden die Konzepte der PREANOR-Verknüpfung, des Zurückstellens, der Subprozesse und das Prozessfortschrittszustands Modell eingeführt. Die PREANOR-Verknüpfung wurde definiert als die Zusammensetzung von zwei Arten von Implikationen (schwache und starke) und dem schwachen AND, plus dem retrospektiven OR. Dadurch lassen sich die oben beschriebenen Kriterien, rein basierend auf Datenabhängigkeiten, modellieren. Dabei sind die Implikationen zur Unterstützung von temporaler Reihenfolge und zur Unterstützung von ad-hoc Entscheidungen zuständig. Die schwache AND Verknüpfung und die retrospektive Verknüpfung repräsentieren Entscheidungen. Die PREANOR-Verknüpfung ist damit zuständig für das Modellieren von Beziehungen zwischen Aktivitäten in der Priorisierungsliste. Das Zustandsmodell ist notwendig um das Beenden von Prozessen zu

modellieren und gibt darüber hinaus auch Aufschluss über den derzeitigen Case/$\alpha$-Card Zustand. Aufbauend auf diesen Überlegungen wurde das Subprozesskonzept eingeführt. Durch Evaluation basierend auf zwei weitverbreiteten Evaluierungs-Frameworks wurde gezeigt, dass $\alpha$-Flow damit in der Lage ist, die gleiche Aussagekraft wie die aktivitätsbasierten Notationen zu erreichen, jedoch mit anderem Schwerpunkt. Diese aktivitätsbasierten Notationen haben demnach Schwächen in der Modellierung von Datenabhängigkeiten, wenn sie überhaupt unterstützt werden. $\alpha$-Flow besitzt dagegen keine Unterstützung für Kontrollfluss, grenzt sich jedoch auch bewusst davon ab.
Demnach lässt sich konstatieren, dass $\alpha$-Flow, basierend auf dem DAB-Modell, am besten für die Modellierung von Prozessen aus dem medizinischen Umfeld geeignet ist, da es die höchste Ausdrucksmächtigkeit in dieser Domäne besitzt.

# Appendices

# A Tabellen der Evaluation

In diesem Kapitel werden die Tabellen, die den Grafiken aus Kapitel 8 zu Grunde liegen aufgeführt. In Tabelle A.1 wird die Einteilung aller betrachteten Notationen mit Hilfe des BWW-Modells gezeigt. In Tabelle A.2 wird die Einteilung aller betrachteten Notationen mit Hilfe der Workflow Pattern gezeigt. In Tabelle A.3 wird die Einteilung aller betrachteten Notationen mit Hilfe der Datenperspektive gezeigt. In der Tabelle A.4 ist die Einteilung aller betrachteten Notationen mit Hilfe der Datenperspektive und der Workflow Pattern angegeben.

| 1 | | BPMN | UML | Petri-Netze | EPK | α-Flow |
|---|---|---|---|---|---|---|
| 2 | Gegenstände und Eigenschaften | 1 | 1 | 1/2 | 1/4 | 1/2 |
| 3 | Zustände von Gegenständen | 0 | 3/7 | 4/7 | 3/7 | 4/7 |
| 4 | Ereignisse und Transformationen von Gegenständen | 5/6 | 1/3 | 1/2 | 1/2 | 2/3 |
| 5 | Systemstrukturen von Gegenständen | 6/7 | 0 | 0 | 1/7 | 5/7 |

**Tabelle A.1:** Tabelle mit Einteilung aller betrachteten Notationen mit Hilfe des BWW-Modells

| 1 | | BPMN | UML | Petri Netze | EPK | α-Flow |
|---|---|---|---|---|---|---|
| 2 | Grundkontrollmuster | 1 | 1 | 1 | 1 | 1/5 |
| 3 | Eweitere Synchronisierungsmuster | 1 | 3/4 | 1/2 | 1/2 | 0 |
| 4 | Strukturmuster | 1 | 1 | 1/2 | 1 | 0 |
| 5 | Mulitple Instanzemuster | 3/4 | 3/4 | 1/2 | 0 | 0 |
| 6 | Abbruchsmuster | 1 | 1 | 1/2 | 0 | 0 |
| 7 | Zustandsmuster | 1/3 | 1/3 | 1 | 0 | 1/6 |
| 8 | | | | | | |

**Tabelle A.2:** Tabelle mit Einteilung aller betrachteten Notationen mit Hilfe des Workflow Pattern Frameworks

| 1 | | BPMN | UML | Petri Netze | EPK | α-Flow |
|---|---|---|---|---|---|---|
| 2 | Änderungen der Reihenfolge | 0 | 0 | 0 | 0 | 1 |
| 3 | Ad-hoc Streichung/Hinzufügen von Aufgaben | 0 | 0 | 0 | 0 | 1 |
| 4 | Prozessfortschritt durch Daten | 0 | 1 | 0 | 0 | 1 |
| 5 | Datenproduktion | 0,5 | 1 | 0 | 0,5 | 1 |
| 6 | Entscheidung auf Grund von Daten | 0 | 1 | 0 | 0 | 1 |
| 7 | Sicht auf alle Daten | 0 | 0 | 0 | 0 | 1 |
| 8 | Open-end Prozess möglich (keine Garantie der Be | 0 | 0 | 0 | 0 | 1 |
| 9 | Datenbasiertes Zustandsmodell | 0,5 | 0,5 | 0 | 0,5 | 1 |
| 10 | Dekomposition von Daten | 0 | 0 | 0 | 0 | 1 |

**Tabelle A.3:** Tabelle mit Einteilung aller betrachteten Notationen mit Hilfe der Datenperspektive

| 1 | | BPMN | UML | Petri Netze | EPK | α-Flow |
|---|---|---|---|---|---|---|
| 2 | Grundkontrollmuster | 1 | 1 | 1 | 1 | 1/5 |
| 3 | Eweitere Synchronisierungsmuster | 1 | 3/4 | 1/2 | 1/2 | 0 |
| 4 | Strukturmuster | 1 | 1 | 1/2 | 1 | 0 |
| 5 | Mulitple Instanzemuster | 3/4 | 3/4 | 1/2 | 0 | 0 |
| 6 | Abbruchsmuster | 1 | 1 | 1/2 | 0 | 0 |
| 7 | Zustandsmuster | 1/3 | 1/3 | 1 | 0 | 1/6 |
| 8 | Datenperspektive | 1/9 | 2/5 | 0 | 1/9 | 1 |

**Tabelle A.4:** Tabelle mit Einteilung aller betrachteten Notationen mit Hilfe der Datenperspektive und den Workflow Pattern

# Literaturverzeichnis


[Aal98a] Aalst, W.M.P. van d.: The Application of Petri nets to workflow management. In: *Journal of Circuits Szstems and Computers* 8 (1998), S. 21–66

[Aal98b] Aalst, W.M.P. van d.: Three Good Reasons for Using a Petri-net-based Workflow Management System. In: *Information and Process Integration in Enterprises* Bd. 428, 1998 (The Kluwer International Series in Engineering and Computer Science), 161-182

[Aal99] Aalst, W.M.P. van d.: Formalization and verification of event-driven process chains. In: *Information and Software Technology* 41 (1999), July, Nr. 10, 639-650. http://dx.doi.org/10.1016/S0950-5849(99)00016-6

[ABHB00] Aalst, W.M.P. van d. ; Barros, A.P. ; Hofstede, A.H.M.ter ; B.Kiepuszewski: Advanced Workflow Patterns. In: *Cooperative Information Systems* Bd. 1901/2000, 2000 (Lecture Notes in Computer Science), 18-29

[AH05] Aalst, W.M.P. van d. ; Hofstede, A.H.M. ter: YAWL: Yet another Workflow Language. In: *Information Systems* 30 (2005), June, 245-275. http://dx.doi.org/10.1016/j.is.2004.02.002

[AHKA03] Aalst, W.M.P. van d. ; Hofstede, A.H.M. ter ; Kiepuszewski, B. ; A.P.Barros: Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages. In: *Distributed and Parallel Databases* 14 (2003), 5-51. http://dx.doi.org/10.1023/A:1022883727209

[AMR09] Aalst, W.M.P. van d. ; Mans, R.S. ; Russel, N.C.: Workflow Support Using Proclets: Divide, Interact and Conquer. In: *IEEE Data Engineering Bulletin* 32 (2009), Nr. 3, S. 1–7

[APEJ00] Aalst, W.M.P. van d. ; P.Barthelmess ; Ellis, C.A. ; J.Wainer: Workflow Modeling using Proclets. Version: 2000. http://dx.doi.org/10.1007/10722620_20. In: *Proc CoopIS00 LNCS 1901* Bd. 1901. Springer, 2000, 198-209

[APEJ01] Aalst, W.M.P. van d. ; P.Barthelmess ; Ellis, C.A. ; J.Wainer: Proclets: A Framework for Lightweight Interacting Workflow Processes. In: *International Journal of Cooperative Information Systems* 10 (2001), Nr. 4, 443-481. http://dx.doi.org/10.1.1.10.4032

[AWG05] Aalst, W.M.P. van d. ; Weske, Mathias ; Grünbauer, Dolf: Case handling:a new paradigm for business process support. In: *Data & Knowledge Engineering* 53 (2005), May, 129-162. http://dx.doi.org/10.1016/j.datak.2004.07.003

[Bar06] Bartonitz, Martin: *BPMS, BPML, BPEL, BPMN, WMS; XPDL,...* org-portal.org. www.org-portal.org. Version: 2006

[Boc03a] Bock, Conrad: UML 2 Activity and Action Models. In: *Journal of Object Technology* 2 (2003), 43-53. www.jot.fm

[Boc03b] Bock, Conrad: UML2 Activity and Action Models Part tII: Control Nodes. In: *Journal of Object Technology* 2 (2003), 7-23. www.jot.fm

[Boc04] Bock, Conrad: UML 2 Activity and Action Models Part IV: Obejct Nodes. In: *Journal of Object Technology* 3 (2004), 27-41. www.jot.fm

[Boc05] Bock, Conrad: UML 2 Activity and Action Models Part VI: Structured Activites. In: *Journal of Object Technology* 4 (2005), 43-66. www.jot.fm

[BP05] Brogi, Antonio ; Popescu, Razvan: From BPEL Processes to YAWL Workflows. In: *Web Services and formal Methods* Bd. 4184/2005, 2005 (Lecture Notes in Computer Science)

[CHE04] Czarnecki, Krzysztof ; Helsen, Simon ; Eisenecker, Ulrich: Staged Configuration Using Feature Models. In: *Software Product Lines* Bd. 3154/2004, 2004, 162-164

[CS06] Claus, Prof. Dr. V. ; Schwill, Prof.Dr.Andreas ; Bauer, Michael (Hrsg.): *Duden der Informatik A-Z, Fachlexikon für Sutudium. Ausbildung und Beruf*. Bd. 1. Bibliographisches Institut & F.A. Brockhaus AG ISBN:9783411052349, 2006. – 767 S. www.duden.de

[DAH05] Dumas, Marlon ; Aalst, W.M.P. van d. ; Hofstede, H.M.ter: *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Bd. 1. John Wiley & Sons, 2005. – 432 S. 10.1002/0471741442.fmatter

[DRK97] Dadam, Peter ; Reichert, Manfred ; Kuhn, Klaus: Clinical Workflow - The Killer Application for Process-oriented Information Systems? In: *Proc 4th Int Conf on Business Information Systems*, 1997, S. 36–59

[EGG91] Ellis, C.A. ; Gibbs, S.J. ; G.L.Rein: Groupware: Some Issues and some Experiences. In: *Communications of the ACM* 34 (1991), January, 38-58. http://dx.doi.org/10.1145/99977.99987

[FR10] Freund, Jakob ; Rücker, Bernd: *Praxishandbuch BPMN 2.0*. Bd. 2. Carl Hanser Verlag ISBN: 9783446424555, 2010. – 268 S. www.hanser.de/computer

[FRH10] Freund, Jakob ; Rücker, Bernd ; Henninger, Thomas: *Praxishandbuch BPMN*. Bd. 1. Carl Hanser Verlag ISBN: 9783446417687, 2010. – 266 S. www.hanser.de/computer

[Gar03] Gardner, Tracy: UML Modelling of Automated Business Process with a Mapping to BPEL4WS. In: *Business* 1 (2003), S. 1–5

[HK99] Hitz, Martin ; Kappel, Gerti: *UML@Work*. Bd. 1. dpunkt.verlag ISBN: 9783932588389, 1999. – 358 S. www.ifs.univie.ac.at/UML

[Hol09] Holt, Jan: *A pragmatic Guide to Business Process Modelling*. Bd. 2. British Computer Society ISBN: 9781906124120, 2009. – 215 S.

[HSS05] Hinz, Sebastian ; Schmidt, Karsten ; Stahl, Christian: Transforming BPEL to Petri Nets. In: *Proceedings of the Third International Conference on Business Process Management BPM 2005* 3649 (2005), Nr. Bpm, 220-235. http://dx.doi.org/10.1007/11538394_15

[IRGR08] Indulska, Marta ; Recker, Jan ; Green, Peter ; Rosemann, Michael: Representational Deficiency of Process Modelling Languages: Measures and Implications. In: *Proceedings of the 16th European Conference on Information Systems*, National University of Ireland, 2008, 1632–1643

[JBS97] Jablonski, Stefan ; Böhm, Markus ; Schulze, Wolfgang: *Workflow Management Entwicklung von Anwendungen und Systemen: Facetten einer neuen Technologie*. Bd. 1. dpunkt.verlag ISBN: 392099373X, 1997. – 537 S.

[Jen87] Jensen, Kurt: Coloured Petri Nets. In: *Petri nets central models and their properties* 254 (1987), 248-299. `http://www.springerlink.com/index/121708715w5177mu.pdf`

[Jen91] Jensen, Kurt: Coloured Petri Nets: A high Level Language for System Desing and Analysis. In: *Lecture Notes in Computer Science* 483 (1991), 342-416. `http://www.springerlink.com/index/y002446703r30355.pdf`

[Jör03] Jörgensen, Jens B.: *Coloured Petri Nets in Development of a pervasive Health Care System*. 2003

[KCH+90] Kang, Kyo C. ; Cohen, Sholom G. ; Hess, James A. ; E.Novak, William ; Peterson, A.Spencer: Feature-Oriented Domain Analysis (FODA) Feasibility Study. In: *Distribution* 17 (1990), Nr. November, 161. `http://dx.doi.org/10.1.1.124.8815`

[KR10] Künzle, Vera ; Reichert, Manfred: Integrating Users in Object Orientated Process Management Systems: Issues and Challenges. In: *BPD09 5th International Workshop on Business Process Design* Bd. 43, 2010, 29-41

[KR11] Künzle, Vera ; Reichert, Manfred: PHILharmonicFlows: Towards a Framework for Object-aware Process Management. In: *Journal of Software Maintenance and Evolution Research and Practice* 1 (2011), S. 1–29

[Kre11] Kreuter, Andre: *Recherche und vergleichende Evaluation von verfügbaren Ansätzen für "Content-oriented Workflows"*, FAU Erlangen Nürnberg, Diplomarbeit, 2011

[KWR10] Künzle, Vera ; Weber, Barbara ; Reichert, Manfred: Object-aware Business Processes: Properties,Requirements, Existing Approaches. In: *International Journal of Information System Modeling and Design* 1 (2010), S. 31

[LBM+05] Lenz, Richard ; Beyer, Mario ; Meiler, Christian ; Jablonski, Stefan ; A.Kuhn, Klaus: Informationsintegration in Gesundheitsversorgungsnetzen. In: *Spektrum Informatik* 4 (2005), 105-119. http://www.springerlink.com/index/GL0502134X466236.pdf

[LK06] List, Beate ; Korherr, Birgit: An Evaluation of Cenceptual Business Process Modelling Languages. In: *Proceedings of the 2006 ACM symposium on Applied computing SAC 06* 1 (2006), Nr. Section 3, 1532. http://dx.doi.org/10.1145/1141277.1141633

[LLN11] Lessen, Tammo van ; Lübke, Daniel ; Nitzsche, Jörg: *Geschäftsprozesse automatisieren mit BPEL*. Bd. 1. dpunkt.verlag ISBN: 9783898646703, 2011. – 260 S. www.bpelbuch.de

[LP99] Lilius, Johan ; Paltor, Ivan P.: The Semantics of UML State Machines. (1999), Nr. 273, 1–20. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.3110

[LR07] Lenz, R ; Reichert, M: IT support for healthcare processes – premises, challenges, perspectives. In: *Data & Knowledge Engineering* 61 (2007), Nr. 1, 39–58. http://linkinghub.elsevier.com/retrieve/pii/S0169023X06000784

[LRD06] Ly, Linh T. ; Rinderle, Stefanie ; Dadam, Peter: Semantic Correctness in Adaptive Process Management Systems. In: *Surger* 4102 (2006), 193–208. http://dx.doi.org/10.1007/11841760_14

[MGR04] Müller, R ; Greiner, U ; Rahm, E: AGENTWORK: A Workflow-System Supporting Rule-Based Workflow Adaptation. In: *Data and Knowledge Engineering* 51 (2004), Nr. 2, 223–256. http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6TYX-4CG0JDT-1&_user=129520&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000010758&_version=1&_urlVersion=0&_userid=129520&md5=49410470b516e494822c661ddb8005f2

[MNN05] Mendling, Jan ; Neumann, Gustaf ; Nüttgens, Markus: Yet Another Event-driven Process Chain - Modeling Workflow Patterns with yEPCs. In: *Enterprise Modelling and Information Systems Architectures* 1 (2005), Nr. 1, S. 3–13

[MRH07] In: Müller, Dominic ; Reichert, Manfred ; Herbst, Joachim: *Data-driven modeling and coordination of large process structures.* 2007, 1

[MRVDA+10] Mans, R S. ; Russell, N C. ; Van Der Aalst, W M P. ; Bakker, P J M. ; Moleman, A J. ; Jaspers, M W M.: Proclets in healthcare. In: *Journal of Biomedical Informatics* 43 (2010), Nr. 4, 632–649. `http://www.ncbi.nlm.nih.gov/pubmed/20359548`

[NL09] Neumann, Christoph P. ; Lenz, Richard: alpha-Flow: A Document-based Approach to Inter-Institutional Process Support in Healthcare. In: *Proc of the 3rd Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'09) in conjunction with the 7th Int'l Conf on Business Process Management (BPM'09).* Ulm, DE, September 2009

[NL10] Neumann, Christoph P. ; Lenz, Richard: The alpha-Flow Use-Case of Breast Cancer Treatment – Modeling Inter-Institutional Healthcare Workflows by Active Documents. In: *Proc of the 8th Int'l Workshop on Agent-based Computing for Enterprise Collaboration (ACEC) at the 19th Int'l Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2010).* Larissa, GR, Juni 2010

[NL12] Neumann, Christoph P. ; Lenz, Richard: The alpha-Flow Approach to Inter-Institutional Process Support in Healthcare. In: *International Journal of Knowledge-Based Organizations* 2 (2012). – Accepted for publication

[NR02] Nüttgens, Markus ; Rump, Frank J.: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: *Promise* P-21 (2002), Nr. 6, 64–77. `http://scholar.google.de/scholar?q=Ereignisgesteuerte+Prozesskette&hl=de&btnG=Suche&lr=#0`

[NR05] Nüttgens, Markus ; Rump, Frank J.: Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. In: *EPK 2005*, 2005

[NSWL11] Neumann, Christoph P. ; Schwab, Peter K. ; Wahl, Andreas M. ; Lenz, Richard: alpha-Adaptive: Evolutionary Workflow Metadata in Distributed Document-Oriented Process Management. In: *Proc of the 4th Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'11) in conjunction with the 9th Int'l Conf on Business Process Management (BPM'11)*. Clermont-Ferrand, FR, August 2011

[NZ98] In: Nüttgens, M ; Zimmerman, V: *Geschäftsprozeßmodellierung mit der objektorientierten Ereignisgesteuerten Prozeßkette (oEPK)*. 1998, S. 23–36

[ODTHVDA06] Ouyang, Chun ; Dumas, Marlon ; Ter Hofstede, Arthur H M. ; Van Der Aalst, Wil: From BPMN Process Models to BPEL Web Services. In: *2006 IEEE International Conference on Web Services ICWS06* 1 (2006), 285–292. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4032038

[OHS02] Opdahl, Andreas L. ; Henderson-Sellers, Brian: Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model. In: *Software and Systems Modeling* 1 (2002), Nr. 1, 43–67. http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s10270-002-0003-9

[OMG11a] OMG ; OMG (Hrsg.): *OMG Business Process Model and Notation*. OMG, January 2011. http://www.omg.org/spec/BPMN/2.0/

[OMG11b] OMG ; OMG (Hrsg.): *OMG Unfiefied Modeling Language Superstructure*. OMG, January 2011. http://www.omg.org/spec/UML/2.4/

[OMG11c] OMG ; OMG (Hrsg.): *OMG Unifiedd Modeling Language Infrastructure*. OMG, January 2011. http://www.omg.org/spec/UML/2.4/

[PW08] Priese, Lutz ; Wimmel, Harro: *Petri-Netze*. Bd. 2. Springer Verlag eXamen.press ISBN: 9783540769705, 2008. – 374 S.

[RD98] Reichert, Manfred ; Dadam, Peter: Adept - Supporting Synamic Changes of Workflows Without Loosing Control. In: *Journal of Intelligent Information Systems* 2 (1998), S. 93–129

[Rei10] Reisig, Wolfgang: *Petrinetze - Modellierungtechnik, Analysemethoden, Fallstudien*. Bd. 1. Vieweg+Teubner ISBN: 9783834812902, 2010. – 247 S.

[RHAN11] Russel, N. ; Hofstede, A.H.M. ter ; Aalst, W.M.P van d. ; N.Mulyar: *Workflow Patterns*. http://www.workflowpatterns.com/, 2011

[Rit00] Rittgen, Peter: Quo vadis EPK in ARIS ? : Ansätze zu syntaktischen Erweiterungen und einer formalen Semantik. In: *Wirtschaftsinformatik* 42 (2000), Nr. 1, 27–35. http://www.adm.hb.se/~pri/ZWI00.pdf

[RRIG06] Rosemann, Michael ; Recker, Jan ; Indulska, Marta ; Green, Peter: A Study of the Evolution of the Representational Capabilities of Process Modeling Grammars. In: *Advanced Information Systems Engineering* 4001 (2006), 447–461. http://dx.doi.org/10.1007/11767138_30

[RRIG09] Recker, Jan ; Rosemann, Michael ; Indulska, Marta ; Green, Peter: Journal of the Association for Information Systems Business Process Modeling- A Comparative Analysis * Business Process Modeling- A Comparative Analysis. In: *Journal of the Association for Information Systems* 10 (2009), Nr. 4, 333–363. http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1501&context=jais

[RRK07] Recker, Jan ; Rosemann, MIchael ; Krogstie, John: Ontology - Versus Pattern Based Evaluation of Process Modelling Languages: A Comparison. In: *Communications of the Association for Information Systems* 20 (2007), S. 774–799

[RRVDA03] Reijers, Hajo A. ; Rigter, Jaap ; Van Der Aalst, Wil M P.: The case handling case. In: *International Journal of Cooperative Information Systems* 12 (2003), Nr. 3, 365–391. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.4951&rep=rep1&type=pdf

[RVDATHW06] Russel, Nick ; Van Der Aalst, Wil M P. ; Ter Hofstede, Arthur H M. ; Wohed, Petia: On the Suitability of UML 2 . 0 Activity Diagrams for Business Process Modelling. In: *Reproduction* 53 (2006), 95–104. http://portal.acm.org/citation.cfm?id=1151866

[RZMS+09] In: Recker, Jan ; Zur Muehlen, M ; Siau, K ; Erickson, John ; Indulska, Marta: *Measuring method complexity: UML versus BPMN*. Association for Information Systems, 2009 (June), 6–9

[Sch07] Schmelzle, Oleg: *Transformation von annotierten Geschäftsprozessen nach BPEL*, Gottfired Wilhelm Leibniz Universität Hannover, Diplomarbeit, 2007. `static.se.uni-hannover.de`

[SF07] Schattkowsky, Tim ; Forster, Alexander: On the Pitfalls of UML 2 Activity Modeling. In: *International Workshop on Modeling in Software Engineering MISE07 ICSE Workshop 2007* 1 (2007), 8–8. `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4273248`

[Stö04] Störrle, Harald: Semantics of Control-Flow in UML 2.0 Activities. In: *2004 IEEE Symposium on Visual Languages Human Centric Computing* 1 (2004), 235–242. `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1372327`

[Stö05] Störrle, Harald: Semantics and Verification of Data Flow in UML 2.0 Activities. In: *Electronic Notes in Theoretical Computer Science* 127 (2005), Nr. 4, 35–52. `http://linkinghub.elsevier.com/retrieve/pii/S1571066105001775`

[Swe10] Swenson, Keith D.: *Mastering the Unpredictable - How Adaptive Case Management Will Revolutuionize the Way That Knowledge Workers Get Things Done*. Bd. 1. Meghan-Kiffer Press ISBN:9780929652122, 2010. – 338 S. `www.MasteringTheUnpredictable.com`

[WAD+06] Wohed, P ; Aalst, W M P Van D. ; Dumas, M ; Ter Hofstede, Arthur ; Russell, Nick: On the suitability of BPMN for business process modelling. In: *Business Process Management* 4102 (2006), 161–176. `http://www.springerlink.com/index/x75270954k172283.pdf`

[Web95] Weber, Ron: Toward a theory of the deep structure of information systems. In: *Information Systems Journal* 5 (1995), Nr. 3, 203–223. `http://www3.interscience.wiley.com/journal/119835681/articletext?DOI=10.1111/j.1365-2575.1995.tb00108.x`

[Whi04a] White, S A.: Introduction to BPMN. In: *Elements* 15 (2004), Nr. c, 1–11. http://www.zurich.ibm.com/~olz/teaching/ETH2011/White-BPMN-Intro.pdf

[Whi04b] White, Stephen A.: Process Modeling Notations and Workflow Patterns. In: *Business* March (2004), Nr. 1999, 1–25. www.bptrends.com

[WWDG+90] In: Wand, Yair ; Weber, Ron ; De Gross, Janice I. ; Alavi, Maryam ; Oppelland, Hans: *Towards a Theory of the Deep Structure of Information Systems*. ACM Press, 1990, S. 61–71

[ZMR08] In: Zer Muehlen, Michael ; Recker, Jan: *How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation*. Bd. 5074. Springer, 2008, 465–479