ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken- und Informationssysteme

Evaluierung bestehender Konzepte für datenorientiertes Prozessmanagement

Diplomarbeit an der Universität Ulm

Vorgelegt von:
Mark Alexander Popa
mark.popa@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Dr. Vera Künzle

2015

"Evaluierung bestehender Konzepte für datenorientiertes Prozessmanagement"
Fassung: 21. März 2016

Satz: PDF-LaTeX 2ε

Danksagung

Für die Unterstützung und den Rückhalt während meiner Diplomarbeit möchte ich mich hiermit von ganzem Herzen bei meiner Familie bedanken.

Außerdem bedanke ich mich bei Herrn Professor Reichert und meinen Betreuern Kevin Andrews und Dr. Vera Künzle.

Danke.

Inhaltsverzeichnis

1

Einleitung

Jede Unternehmensstrategie beinhaltet Konzepte zur Kostenreduktion, Effizienzsteigerung und auch zur Geschäftsprozessoptimierung. Denn auf einem globalisierten Markt ist es für jedes Unternehmen wichtig und notwendig, seine zukünftige Ausrichtung ständig zu überprüfen und zu überarbeiten. Nur eine regelmäßige Anpassung der Geschäftsprozesse garantiert, flexibel und zeitnah auf Marktveränderungen reagieren zu können. Deshalb hat sich die letzten Jahre auf Managementebene immer mehr das Thema Business Prozess Management (BPM) etabliert. Dieses unterstützt die Implementierung von Aufgaben und Maßnahmen, um Geschäftsprozesse effizienter und effektiver zu gestalten. [1][2]
BPM beinhaltet Konzepte, Methoden und Techniken, um Punkte wie die Gestaltung, die Konfiguration, die Verwaltung, die Ausführung, die Überwachung und die Analyse von Geschäftsprozessen zu unterstützen. Dabei können Personen, Organisationen, Programme, Dokumente oder andere Informationsquellen beteiligt sein. [3] Wird dies durch

IT-gestützte Softwaresysteme realisiert, spricht man von einem Process Management System (PrMS). Beispiele für Frameworks, die sich auf dem Markt befinden, sind die Oracle BPM Suite [4], das IBM WebSphere [5] oder die AristaFlow BPM Suite. [6] In Verknüpfung mit einer Prozessmodellierungssprache, wie beispielsweise Business Process Management and Notation (BPMN) [7] helfen PrMS, die genannten Eigenschaften von BPM umzusetzen.
Der Vorteil von heutigen PrMS, gegenüber hart verdrahteten Systemen, besteht darin, dass die Prozesse modelliert werden. Durch die Trennung der Anwendungen von der Ablauflogik, können Prozesse leicht angepasst und verändert werden. Dabei werden die Anwendungen, welche außerhalb des PrMS verwaltet werden, wiederverwendet oder ersetzt. Jedoch führt diese prozessorientierte Sicht zu dem Nachteil, dass die Prozesse getrennt von den Daten und Funktionen betrachtet werden. Die Daten werden von den Anwendungen autonom verwaltet. Dies wird als ein Grund gesehen, dass noch viele Customer Relationship Management (CRM)-Systeme und Enterprise Resource Planning (ERP)-Systeme in den Unternehmen im Einsatz sind. [8] Denn hier werden die Prozesse mit den Anwendungsdaten fest implementiert, so dass ein hart verdrahtetes System entsteht.
Trotz der Vorteile, die sich in einer integrierten Sicht auf Funktionen und Daten widerspiegeln, besteht auch ein weitreichender Nachteil. Prozessänderungen, oder die Einführung neuer Prozesse, sind in diesen Informationssystemen nur sehr schwer umsetzbar. Dies führt zu hohen Kosten und einem höchst unflexiblen System.
Dennoch haben die genannten Vorteile von BPM zu einem erhöhten Interesse und einer erweiterten Anwendung in Unternehmen geführt. Allerdings geht aus der Studie von BPM&0 und BearingPoint hervor, dass viele Punkte bei der Umsetzung von solchen Systemen nicht zufriedenstellend erreicht werden. [1] So wurden die Ziele, wie zum Beispiel die Steigerungen der Kundenzufriedenheit, oder auch die Kostensenkungen, in gerade einmal jedem sechsten Fall erreicht.
Neben vielseitigen Ursachen für diese Zielverfehlungen, können die Probleme auch auf die fehlende Integration der Geschäftsperspektiven Daten und Funktionen zurückgeführt werden. Einem Benutzer ist es, in den meist klassischen Systemen, nicht möglich, flexible Datenänderungen und Funktionsaufrufe außerhalb der obligatorischen Aktivitäten

vorzunehmen.
Doch steigt die Anzahl an wissensintensiven Arbeiten, die genau diese Flexibilität fordern. Dabei handelt es sich um die, immer öfter auftretenden, schwach strukturierten Prozesse, welche sich häufig ändern, sehr komplex sind und von den Benutzern mit vorhandenem Fachwissen ausgeführt werden müssen. [9] Diese sogenannten Wissensarbeiter wollen keine festen Vorgaben bezüglich der Ausführungsreihenfolge, sondern entscheiden selbst über den Ablauf und wie sie das Prozessziel erreichen. Dafür ist die Verfügbarkeit aller prozessrelevanten Daten notwendig. Deshalb gibt es eine große Anzahl unterschiedlicher Forschungsansätze, die sich mit der Enwicklung von datenzentrierten Systemen beschäftigt. Neben dem artefaktzentrierten Ansatz wird in dieser Arbeit auch ein fallzentrierter Ansatz erläutert. Durch die Integration der Geschäftsperspektiven Daten und Prozesse, unter der Einbeziehung von Funktionen und Benutzern, entsteht mit PHILharmonicFlows ein objektorientiertes (object-aware) Prozessmanagement-Framework, entwickelt von dem Institut für Datenbanken- und Informationssysteme (DBIS). Dieses basiert darauf, dass der Fortschritt eines Prozesses primär nicht mehr von den ausgeführten Aktivitäten und des damit verbundenen Kontrollflusses abhängt, sondern von Bedienungen und Werten der Datenobjekte.
Ein weiteres Problem bei datenzentrierten Ansätzen besteht in der Modellierung dieser Ansätze. Da die Abfolge einer Prozessinstanz bei einem solchen System oft variiert und von den Daten der Aktivitäten abhängt, ist eine Darstellung mit Modellierungssprachen wie BPMN nur schwer bis gar nicht möglich. Deshalb hat die Object Management Group (OMG) mit Case Mangement Model and Notation (CMMN) [10] einen neuen Notationsstandard zur Beschreibung wissensintensiver Prozesse herausgebracht.

1.1 Motivation

Ein sehr bekanntes Zitat des amerikanischen Physikers und Vorreiters im Bereich des Qualitätsmanagements, William Edwards Deming, lautet:

> „If you can’t describe what you are doing as a process, you don’t know what you are doing.“

Unter Betrachtung der bereits erwähnten klassischen prozessorientierten PrMS definiert dieses Zitat genau die Anforderungen und Erwartungen, die an ein solches System geknüpft sind. Abläufe in einem Unternehmen sollten im Vorfeld immer klar geregelt sein, um sie dann zu modellieren und festzusetzen. Genau so erhält man eine Übersicht über die Abläufe in seinem Unternehmen und kann diese überwachen und anpassen. Nur wenn man in Form eines Prozesses aufzeigen kann, was man umsetzen will, weiß man auch, was man tut.
Diese zunächst unbestreitbare Aussage rückt jedoch in ein anderes Licht, wenn man die bereits erwähnten Veränderungen miteinbezieht, die sich in Form von wissensintensiven oder schwach bis lose strukturierten Prozessen äußern. Dieses Zitat spiegelt aber nicht die Flexibilität wider, die Unternehmen für immer mehr Geschäftsprozesse im heutigen Arbeitsalltag fordern. Es ist nicht immer möglich, Arbeitsabläufe im Vorfeld festzulegen. Oft sind Informationen nicht bekannt und müssen während der Laufzeit eines Systems eingebunden werden. Wie können also auch Wissensarbeiter, die nicht in die starren Strukturen eines klassischen PrMS passen, bei ihrer Arbeit unterstützt werden, ohne sie dabei einzuschränken? Wie kann man auf ein sich veränderndes Benutzerverhalten bei der Bearbeitung von Prozessen eingehen? Inwieweit das Zitat von W. E. Deming heute noch aktuell ist, oder ob es als veraltet angesehen werden muss, wird in dieser Arbeit beleuchtet.

1.2 Zielsetzung

In dieser Arbeit werden, im Rahmen einer Literaturrecherche, die aktivitätenzentrierten Ansätze von den datenzentrierten Ansätzen abgegrenzt. Mit der Forschung an PHILharmonicFlows wurden Herausforderungen zusammengetragen, die nötig sind, um ein gesamtheitliches, datenorientiertes PrMS zu erhalten. Die datenzentrierten Konzepte werden dann gegen die Herausforderungen evaluiert, um zu zeigen, welche Eigenschaften in Bezug auf ein gesamtheitliches, datenorientiertes System umgesetzt werden. Dabei wird auch das, von der OMG entwickelte, CMMN untersucht werden. Es wird erforscht, inwieweit dieses den Ansprüchen eines datenorientieren Systems gerecht wird und wie es Wissensarbeiter bei ihrer Arbeit unterstützen kann. Außerdem wird

untersucht, in welchem Umfang CMMN bei der Modellierung strukturschwacher und datengetriebener Prozesse hilft und ob diese Prozesse auch mit anderen Notationssprachen, wie beispielsweise BPMN 2.0, modelliert werden können. Weiterhin wird erläutert, wann die Notationen an ihre Grenzen stoßen, welche Vor- und Nachteile sie mit sich bringen und inwiefern diese miteinander verknüpft werden können.

1.3 Aufbau der Arbeit

Kapitel 2:
In Kapitel 2 werden die theoretischen Grundlagen für diese Arbeit zusammengefasst, um einen Überblick über die bisherige Forschungsarbeit zu geben. Zunächst wird der aktivitätenzentrierte Ansatz definiert und in einem ersten Schritt evaluiert. Weiterhin werden wissensintensive Prozesse vorgestellt. Nachdem auch hier auf die Vor- und Nachteile eingegangen wurde, werden Ansätze der Prozessmodellierung erläutert. Insbesondere müssen hierbei der imperative, der deklarative sowie der regelzentrierte Ansatz genannt werden. Letztlich ist es notwendig, auf die datenzentrierten Ansätze einzugehen. Neben dem artefaktzentrierten Ansatz, welcher unter anderem das Guard-Stage-Milestone Modell beinhaltet, gehört dazu auch der fallzentrierte Ansatz. Als letztes wird das PHILharmonicFlows Framework vorgestellt. Dieses dient als Beispiel für ein objektorientiertes Prozessmanagement und wird als solches zunächst in seinen Grundbestandteilen erläutert

Kapitel 3:
Da die bisher untersuchten Ansätze, wie bereits erwähnt, einige Nachteile und Lücken aufweisen, wird in Kapitel 3 diskutiert, wie ein gesamtheitlicher Ansatz aussehen könnte. Hierfür ist es unabdingbar, die Herausforderungen der heutigen Zeit an einen solchen Ansatz zusammenzutragen. Um ein sogenanntes datenorientiertes Prozessmanagement entwickeln zu können, wird zunächst evaluiert, inwieweit die bisher behandelten Ansätze an die Herausforderungen angepasst sind. In einem letzten Schritt wird das Framework von PHILharmonicFlows untersucht. Dabei wird gezeigt, inwieweit die Herausforderungen in diesem objektorientierten Framework umgesetzt worden sind.

Kapitel 4:
Eine weitere Forschungsrichtung befasst sich mit den Modelliersprachen BPMN und CMMN, welche in Kapitel 4 in ihren Grundlagen erläutert werden. Insbesondere CMMN und dessen Metamodell werden dabei genauer untersucht. In einer Literaturstudie wird nachvollzogen, auf welchem Wege und vor welchem Hintergrund CMMN entstanden ist. In einem weiteren Schritt werden die Gemeinsamkeiten und Unterschiede der beiden Notationen herausgearbeitet.

Kapitel 5:
Im letzten Kapitel werden die Ergebnisse der Arbeit zusammengefasst und ein Fazit gezogen. Insbesondere wird über CMMN als eigenständige Modelliersprache diskutiert.

2

Grundlagen zu aktivitäten- und datenzentriertem Prozessmanagement

Für ein näheres Verständnis von PHILharmonicFlows und einer objektorientierten Ansicht werden in diesem Kapitel zunächst die Anforderungen besprochen, die an ein datenorientiertes PrMS gestellt werden. In einem ersten Schritt werden prozessorientierte Systeme vorgestellt und Gründe für eine fehlende Integration von Daten und Funktionen aufgezeigt. Dann werden die Herausforderungen an ein datenorientertes PrMS ausgearbeitet. Im Anschluss daran werden Konzepte vorgestellt, die einzelne Anforderungen eines gesamtheitlichen Ansatzes bereits berücksichtigen. Dazu gehören das Case Handling und der artefaktzentrierte Prozessmanagementansatz. Im Folgenden wird zunächst der aktivitätenzentrierte Ansatz erläutert sowie Vor- und Nachteile herausgearbeitet.

2.1 Aktivitätenzentrierter Ansatz (activity-centric)

Die rasante Geschwindigkeit bei der Entwicklung neuer Produkte und die immer kürzeren Produktzyklen machen eine automatisierte Unterstützung von Geschäftsprozessen im Unternehmen unausweichlich, um mit einem vernetzten und dynamischen Markt Schritt zu halten. Ein Geschäftsprozess ist ein aus mehreren Aktivitäten bestehender Arbeitsablauf zur Realisierung eines Geschäftsziels. Verknüpft mit einer angepassten Softwarelösung erhält man ein PrMS, welches die Ausführung der Geschäftsprozesse unterstützt. [1]

Viele der Frameworks, die heute auf dem Markt eingesetzt werden, basieren auf einem aktivitätenzentrierten Ansatz. In dieser Arbeit werden diese als klassisches Prozessmanagement oder als klassisches PrMS bezeichnet.
Bei der Modellierung wird hierbei ein Prozess erstellt, welcher aus Aktivitäten besteht, die jeweils mit einer Anwendungsfunktion verknüpft werden.
Die dazugehörige Ablauflogik wird über einen Kontrollfluss gesteuert und hat eine festgelegte Reihenfolge.
Die benötigten Daten werden von den darunterliegenden Anwendungen selbst verwaltet und befinden sich somit außerhalb der Kontrolle des PrMS. Somit ist es nur möglich, über Ein- und Ausgabeparameter einzelne atomare Datenelemente mit dem System zu verknüpfen.
Das PrMS erstellt während der Laufzeit eine Prozessinstanz und übernimmt die Ablaufsteuerung der einzelnen Aktivitäten. Dazu stellt das System alle benötigten Ressourcen zur Verfügung und verteilt die manuell zu bearbeitenden Aufgaben über Arbeitslisten an die Benutzer. Die Anwender erhalten somit eine sogenannte prozessorientierte Sicht auf das PrMS, welches die Prozesse getrennt von den darunterliegenden Funktionen und Daten betrachtet (siehe Abb. 2.1).
Bei der Ausführung hängt die erfolgreiche Beendigung eines Prozesses von der erfolgreichen Bearbeitung jeder einzelnen Aktivität innerhalb des Kontrollflusses ab.

Für sehr strukturierte oder sich wiederholende Prozesse ist dieses klassische PrMS das richtige System, um Geschäftsabläufe passend zu modellieren und bei der Ausführung zu unterstützen. [11]

Die feste Reihenfolge führt jedoch dazu, dass Benutzer Daten während der Laufzeit nicht flexibel bearbeiten können. Deshalb ist es nicht möglich, außerhalb der aktivierten Aktivität Daten einzusehen oder zu bearbeiten. Genauso wenig ist es möglich, eine Aktivität zu überspringen, weil die Ausführung nicht mehr benötigt wird. Denn die Aktivitäten hängen vom Status vorhergehender Aktivitäten ab.
Durch die bereits erwähnte steigende Zahl an wissensintensiven Arbeiten, wird in einem PrMS ein hoher Grad an Flexibilität erwartet, der in einem klassischen System nicht erfüllt wird. Die aktivitätenzentrierte Sicht bietet nicht die Möglichkeit, strukturschwache, flexible, objektzentrierte Prozesse adäquat zu unterstützen, so dass daten- und funktionsorientierte Anwendungen oft durch fest kodierte Prozesslogik erweitert werden. Dadurch wird der Anwendungscode unnötig komplex und schwer wartbar. [8] Die eigentlichen Vorteile eines PrMS, nämlich die Trennung der Prozesse von den Funktionen zur leichteren Anpassung von sich verändernden Geschäftsabläufen, werden damit wieder eingeschränkt.

Abbildung 2.1: Aufbau eines klassischen PrMS, vergleichbar [12]

Da die aktivitätenzentrierten Ansätze einige Nachteile aufweisen und somit nicht in allen Bereichen sinnvoll einzusetzen sind, sollen im Folgenden die wissensintensiven Prozesse vorgestellt werden.

2.2 Wissensintensive Prozesse (knowledge-intensive)

In vielen Arbeitsbereichen hat man festgestellt, dass man mit aktivitätenzentrierten Modellansätzen an die Grenzen der Umsetzbarkeit stößt.
Das kann sehr gut an dem praktischen Beispiel eines Prozessablaufs in einer Notaufnahme verdeutlicht werden. Jede Patientenbehandlung ist unterschiedlich und muss ganz individuell auf die individuellen Bedürfnisse angepasst werden, um dem Patienten bestmöglich helfen zu können. Hier wird deutlich, dass man mit vordefinierten, festen Strukturen bei der Prozessmodellierung nicht weit kommt. Es handelt sich hier um eine Tätigkeit, die im Vorfeld schwer oder gar nicht festgelegt werden kann und oft auch im Team, also kollaborativ ausgeführt wird. Mit der Diskussion um schwache oder unstrukturierte Prozesse und mehr Flexibilität, fällt immer auch der Begriff von wissensintensiven Prozessen.
Wissensarbeit ist, im Gegensatz zu routinierter wiederholbarer Arbeit, dadurch gekennzeichnet, dass diese oft nicht vorhersehbar und von der jeweiligen Situation abhängig ist. [13]
Wissensintensive Prozesse (Knowledge-intensive Processes (KiP)) sind Prozesse, deren Leitung und Ausführung stark von Wissensarbeitern geprägt ist. Diese Bearbeiter von wissensintensiven Aufgaben benötigen ein hohes Fachwissen, welches zur selbstständigen Entscheidungsfindung eingesetzt werden muss.

KiPs sind wissens-, informations- und datenzentriert und benötigen deshalb einen hohen Grad an Flexibiltät, sowohl zur Modellier- als auch zur Ausführungszeit. [14] Somit können diese Prozesse durch Benutzerentscheidungen vom Referenzmodell abweichen, zum Beispiel durch das Eintreten eines unvorhergesehenen Events.
KiPs zeigen, wie wichtig das Wissen in einigen Prozessen geworden ist. Die Wissensarbeiter übernehmen immer mehr Verantwortung bei der Ausführung sogenannter menschenzentrierter (human-centric) Prozesse. Dazu bedarf es einer Erweiterung der prozessorientierten Sicht für diese Benutzer. Durch ein datenzentriertes Modell ist es möglich, die Benutzer, die Daten, die Funktionen und die Prozesse wieder miteinander zu verknüpfen (siehe Abb. 2.2). Dadurch entsteht ein direkter Zugriff der Benutzer und

damit eine Grundlage, die Benutzer von wissensintensiven Arbeiten in ein solches PrMS zu integrieren.

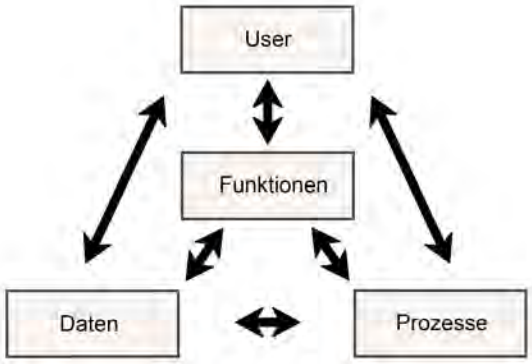


Abbildung 2.2: Aufbau eines objekt-zentrierten PrMS, vergleichbar [12]

Das Fachmagazin BPTrend hat schon 2009 festgestellt, dass bis zu 40 Prozent der Arbeit in einem Unternehmen von Wissensarbeitern ausgeführt wird, mit steigender Tendenz. Weiterhin wurde auch die Wichtigkeit von Wissensarbeitern für den Erfolg und das Wachstum eines Unternehmens hervorgehoben, denn die wichtigsten Prozesse in einem Unternehmen bestehen aus Wissensarbeit. [15]
Die Integration von KiPs in ein imperatives BPM-Framework ist schwer bis unmöglich zu bewerkstelligen, da es durch die Verwendung einer festen Abfolge von Aktivitäten an Flexibilität fehlt. Dabei ist es gerade im Bereich der strukturschwachen Prozesse umso wichtiger, den Benutzer oder auch Wissensarbeiter bei der Bearbeitung eines Prozesses zu unterstützen und einen Überblick über den Ablauf der Geschäftsprozesse zu behalten.
Nachdem der aktivitätenzentrierte Ansatz und der wissensintensive Ansatz näher erläutert, sowie Vor- und Nachteile herausgearbeitet wurden, werden im Folgenden verschiedene Ansätze der Prozessmodellierung vorgestellt. Hierzu zählen der imperative Ansatz und der deklarative Ansatz.

2.3 Ansätze der Prozessmodellierung

Die Unterscheidung von imperativen und deklarativen Sprachen hat ihren Ursprung in der Programmierung und kann auf den Bereich der Prozessmodellierung übertragen werden. [16] Die imperative Programmierung beschreibt den Ansatz, *wie etwas ausgeführt werden soll*. Im Unterschied dazu verfolgt die deklarative Programmierung den Ansatz, *was ausgeführt werden soll, und wie man es erreichen kann*. [17] Dies wird im Folgenden auf die Prozessmodellierung übertragen.

2.3.1 Imperativer Ansatz

Das Ziel einer Prozessmodellierung ist die Beschreibung des Prozesses. Dabei benutzen klassische PrMS zur Modellierung einen imperativen Ansatz. Beispiele für imperative Modelliersprachen sind BPMN oder Petri-Netze. Diese imperativen Modelle gebrauchen einen *inside to outside* Ansatz. Ableitend von der Programmierung bedeutet dies, dass alle Ausführungsalternativen eindeutig zur Entwurfszeit modelliert sein müssen.
Es wird zur Modellierzeit spezifiziert, in welcher Reihenfolge und unter welchen Bedingungen die Aktivitäten ausgeführt werden und nur hier können neue Ausführungspfade hinzugefügt werden. Dabei sind die Aktivitäten die Knoten, während der Kontrollfluss über die Kanten visualisiert wird. Es kann zwischen sequentiellen, alternativen und parallelen Pfaden unterschieden werden. Weiterhin wird auch die Modellierung eines Schleifenkonstrukts unterstützt. [18] Das Hauptaugenmerk liegt auf der Kontrollflussperspektive eines Prozesses (siehe Abb. 2.3). Die Darstellung der Prozesse erfolgt über einen gerichteten Graphen. [19] Die Datenobjekte werden dabei über einen Datenfluss geregelt und dienen nur zur Ablaufsteuerung des Prozesses. Die Verzweigung stellt eine exklusive Fallunterscheidung. Entweder wird Aktivität 3 oder Aktivität 4 ausgeführt. Danach wird die Verzweigung wieder zusammengeführt.

Imperative Prozessmodelle eignen sich am besten für die Ausführung von sich wiederholenden Abläufen, die sich wenig ändern. [20]
Ein Anwendungsbeispiel dafür wäre ein Bestellvorgang in einem Online-Shop. Dieser ist von der Struktur immer gleich, so dass sich dieser Prozess sehr gut mit einer imperativen

Modellierung umsetzen lässt.
Hier erkennt man auch die typische Herangehensweise imperativer Ansätze. Die technische Ausführung der Arbeitsabläufe, das *Wie* steht hier klar im Mittelpunkt.
Durch die Ausrichtung an stark strukturierten Prozessen, sind imperative Ansätze nicht flexibel genug, um schwach strukturierte oder lose Prozesse zu modellieren. [21]

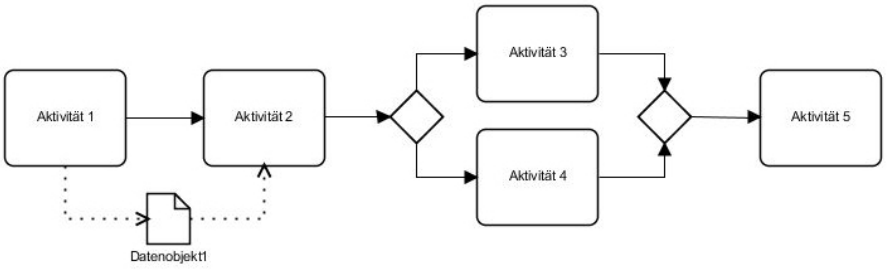


Abbildung 2.3: Abfolge der Aktivitäten bei einem imperativen Ansatz, vergleichbar [8]

2.3.2 Deklarativer Ansatz

Ein neuerer Ansatz versucht den Grenzen der imperativen Modellierung gerecht zu werden. Dementsprechend fokussiert sich das deklarative Prozessmodell darauf, *was getan werden muss*. Im Gegensatz zum imperativen Ansatz steht bei der deklarativen Modellierung nicht der Ablauf im Vordergrund. Vielmehr rückt die Analyse und Gestaltung von Arbeitsabläufen in den Mittelpunkt, um vorgegebene geschäftliche Ziele zu erreichen. Somit werden lediglich die notwendigen Charakteristika beschrieben, die den Rahmen eines Prozesses darstellen. Das Modell fokussiert die Eigenschaften, die das Zusammenspiel der Aktivitäten und Objekte in einem Prozess regeln. [20]
Es wird ein *outside to inside* Ansatz verwendet. Es ist zu Beginn eines Prozesses jeder erdenkliche Ausführungspfad erlaubt. Erst durch das Aufstellen von Restriktionen werden dann die Grenzen festgesetzt. Diese Regeln (constraints) stellen Richtlinien oder Geschäftsregeln dar. [22]
Das Beispiel (siehe Abb. 2.4) zeigt drei Aktivitäten, wobei die Abfolge durch Regeln beeinflusst ist. So können Aktivität 2 und 3 jeweils nur ausgeführt werden, wenn davor

Aktivität 1 bearbeitet wurde. Wird Aktivität 2 ausgeführt, ist das Bearbeiten von Aktivität 3 nicht zulässig und umgekehrt.

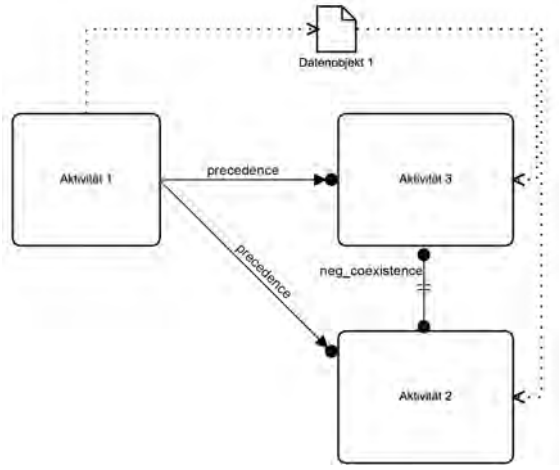


Abbildung 2.4: Aufbau eines deklarativen Ansatzes, vergleichbar [8]

Zur Ausführung ist also jede Ausführungsvariante erlaubt, die nicht gegen diese Regeln verstößt. Durch das Hinzufügen neuer Regeln zum Modell können die Ausführungspfade, je nach Anforderung, Schritt für Schritt eingeschränkt werden. [16] Ein Beispiel für die Benutzung eines deklarativen Ansatzes ist Declare [23] und auch die Modelliersprache CMMN basiert auf diesem Ansatz. Durch die deklarative Sicht bieten diese Modelle den Benutzern mehr Flexibilität bei der Modellierung loser oder schwach strukturierter Prozesse. Dabei wird auch auf eine weitreichende Unterstützung bei der Entscheidungsfindung zur Laufzeit geachtet. [24]

Wenn die Ausführung von Aktivitäten in einem Prozess über einen Kontrollfluss gesteuert wird, steht der Rahmen, in dem sich der Prozess bewegt, bereits im Vorfeld fest. Diese eingeschränkte Sicht bei der Ausführung eines Prozesses soll mit den regelzentrierten Ansätzen erweitert werden. Diese werden im Folgenden näher erläutert.

2.3.3 Regelzentrierter Ansatz (constraints-based Approach)

Bei einem regelzentrierten Ansatz findet die Steuerung mit Hilfe von Regeln statt. Dabei kann man drei verschiedenen Szenarien betrachten, die in einem Geschäftsprozess

existieren können.
Verbotene Szenarien, welche bei der Ausführung niemals vorkommen dürfen.
Optionale Szenarien, die nicht Teil der Hauptausführung sind, aber falls nötig benutzt werden können.
Erlaubte Szenarien, welche ohne Einschränkungen verwendet werden dürfen. [25]
Damit erweitern sich die Ausführungsmöglichkeiten gegenüber einem klassischen Prozess (siehe Abb. 2.5). Benutzer können selber entscheiden, wie sie einen Prozess ausführen, solange sie keine verbotenen Ausführungspfade verwenden.

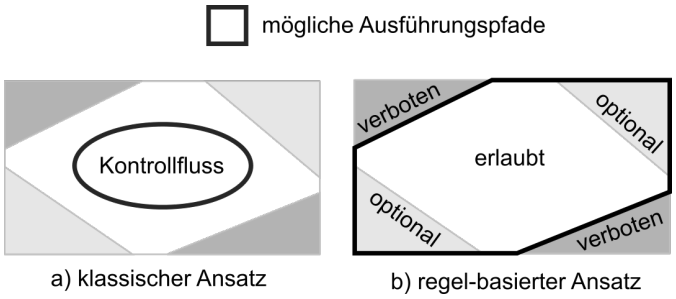


Abbildung 2.5: Ausführungspfade durch regel-basierten Ansatz, vergleichbar [8]

In diesem Modell wird auch die Möglichkeit von optionalen Aktivitäten in einen Ansatz mit integriert. Dies stellt auch eine Herausforderungen für die Entwicklung eines objektzentrierten Ansatzes dar. Durch die Konzentration auf die Werte und Beziehungen von Daten zur Regelung eines Prozesses, spielt auch die Differenzierung von optionalen und obligatorischen Aktivitäten eine wichtige Rolle.
Durch optionale Aktivitäten entsteht die Möglichkeit, Daten jederzeit einzusehen und zu bearbeiten, unabhängig von den Prozessen. Jedoch gilt es hierbei auch immer zu beachten, dass es zu keiner fehlerhaften Beeinflussung der Prozessinstanzen kommt, wohingegen obligatorische Aktivitäten ausgeführt werden müssen. [26] Mit der deklarativen und regelbasierten Ansicht entsteht ein Schritt in die Richtung hin zu datenzentrierter und wissensintensiver Prozessunterstützung.

2.4 Datenzentrierte Ansätze

Im folgenden Abschnitt werden datenzentrierte Ansätze untersucht. Insbesondere stehen hierbei der artefaktzentrierte Ansatz sowie der fallzentrierte Ansatz im Fokus.

2.4.1 Artefaktzentrierter Ansatz (artefact-centric Approach)

Eine datenzentrierte Sicht für Prozessmodelle bietet der artefaktzentrierte Ansatz. [27, 28] Eine artefaktzentrierte Modellierung fokussiert sich auf erweiterte Datensätze, sogenannte Geschäftsartefakte, welche mit geschäftsrelevanten Objekten kommunizieren. Damit werden in einem Geschäftsartefakt, kurz Artefakt, Daten und Prozesslogik miteinander verknüpft.
Artefakte sind ein guter Ansatz, um das Verständnis und auch die Überwachung von Geschäftsoperationen zu erleichtern. Denn es handelt sich um geschäftsrelevante Objekte, die erstellt, entwickelt und archiviert werden können, während sie durch einen Geschäftsvorgang leiten. [29]

Ein Artefakt besteht einerseits aus einem Informationsmodell, welches alle geschäftsrelevanten Daten beinhaltet und damit die Ausführung eines Geschäftsprozesses unterstützt. Diese Daten werden als Objekttyp abgebildet und beinhalten dessen Attribute und Relationen. Dabei können sowohl atomare als auch strukturierte Attribute enthalten sein. Deshalb kann es auch als Datenschema bezeichnet werden.
Zum anderen besteht ein Artefakt aus einem Lebenszyklusmodell, das beschreibt, was während der Ausführung eines Prozesses erlaubt ist. Weiterhin beschreibt es die möglichen Änderungen am Informationsmodell und die Beziehungen zu anderen Artefakten.

Das Lebenszyklusmodell wird über dynamische Beschränkungen definiert und kann auf zwei Arten ausgeführt werden. Es kann imperativ definiert werden, basierend auf einer endlichen Zustandsmaschine (finite state machine) (FSM). Dieses Modell beinhaltet die wichtigsten Stages und deren Übergänge. Damit können in jedem Status die weiteren möglichen Pfade eindeutig abgelesen werden. [30]
Der imperative Lebenszyklus besteht aus drei Teilen, dem *Status*, den *Event-Condition-Action-Regeln* und der *Transistionsbeziehung*:

Beim *Status* handelt es sich um einen festen Zustand, der sich im Moment nicht verändert, beispielsweise nach dem Beenden einer Aktivität.
Event-Condition-Action (ECA)-Regeln entscheiden, ob ein Übergang ausgeführt werden muss. Sie definieren, wie ein Service mit einem Artefakt verbunden ist. Durch die Ausführung dieser Services wird das Artefakt durch seinen Lebenszyklus gelenkt.
Die *Tranistionsbeziehung* beschreibt die Beziehung zwischen einem Status und einer ECA-Regel und stellt den Kontrollfluss des gesamten Prozesses dar. Es überwacht weiterhin die Statusänderungen. [31]
Die FSM hat den Vorteil eines leicht verständlichen und einfach lesbaren Modells. Jedoch ist sie sehr aufwendig zu modellieren und zu ändern. Deshalb wurde 2011 ein deklarativer, regelbasierter Ansatz für das Artefakt-Lebenszyklusmodell vorgestellt, das Guard-Stage-Milestone (GSM)-Metamodell, welches im folgenden Kapitel näher erläutert wird. [32]

Die Abbildung 2.6 zeigt die vier Ebenen eines artefaktzentrierten Ansatzes. Die oberste Ebene ist das Informationsmodell und zeigt die Artefakte und die verbundenen Relationen. Darunter liegt das Lebenszyklusmodell, auch Makro-Lebenszyklus genannt. Dieses zeigt den Status und die Transitionen. Diese beiden Ebenen stellen die Datenmodellierung dar. Darunter liegt die Ebene der Prozessmodellierung. Innerhalb dieser befinden sich die Services. Die vierte Ebene eines artefaktzentrierten Ansatzes stellt die Verknüpfung der Datenmodellierung mit der Prozessmodellierung dar. Hierfür werden mit Hilfe der ECA-Regeln das Lebenszyklusmodell mit den Services verbunden.

2.4.2 Guard-Stage-Milestone-Modell

Das GSM-Modell für Artefakte wurde entwickelt, um einen hierarchischen, modularen und deklarativen Ansatz für die Spezifizierung von Artefakt-Lebenszyklen zu erhalten. Es basiert auf vier Hauptkomponenten, nämlich auf dem *Informationsmodell*, den *Stages*, den *Milestones* und den *Guards*. Diese werden im Folgenden näher erläutert.

Informationsmodell
Die erste Komponente stellt das *Informationsmodell* dar. Es unterstützt, wie bereits erwähnt, eine integrierte Sicht auf alle geschäftsrelevanten Informationen über Artefakt-

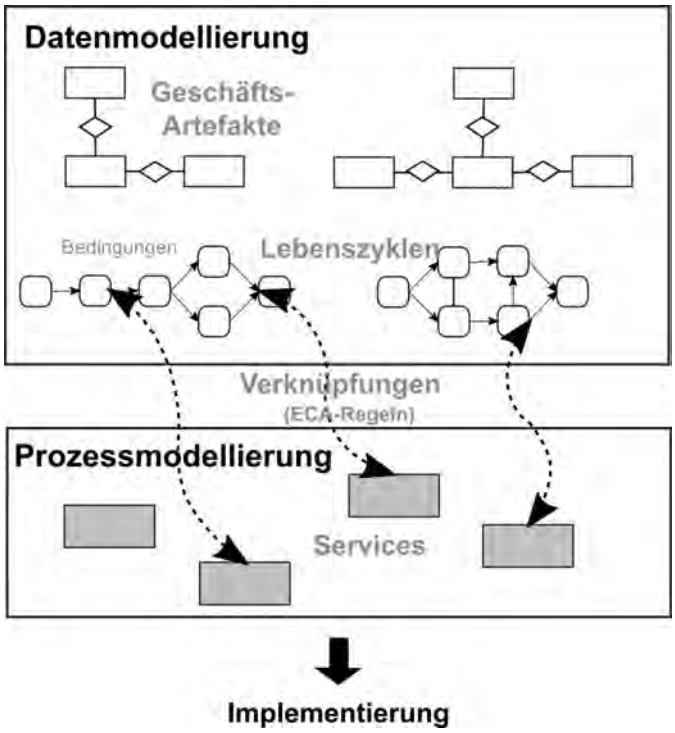


Abbildung 2.6: Aufbau eines artefaktzentrierten Ansatzes, vergleichbar [28]

instanzen. Es wird ähnlich wie ein relationales Schema beschrieben und die Attribute der Objekttypen werden in Daten- und Statusattribute unterteilt.
Der Lebenszyklus wird definiert über die Benutzung von *Stages*, verbunden mit *Guards (Wärter)* und *Milestones (Meilensteinen)*. [33]

Stages
Stages gruppieren individuelle atomare Aufgaben und können auch in anderen *Stages* liegen. Sie sollen die Daten des Informationsmodells erweitern und aktualisieren. Eine Aufgabe ist der Aufruf eines atomaren externen Service, welcher durch Vor- und Nachbedingungen genauer beschrieben wird. Artefakt-Lebenszyklen werden durch *Stages* beschrieben. Diese wiederum sind verknüpft mit einem oder mehreren *Milestones* und *Guards*.

Milestones
Stages sind mit *Milestones* verbunden und stellen operationale Objekte dar, die abgeschlossen sind, wenn die verknüpften Bedingungen erfüllt sind. *Milestones* zeigen somit Ziele an, die es zu erreichen gilt. Wird ein Ziel erreicht, wird die *Stage* geschlossen und die Aufgaben innerhalb der *Stage* können nicht mehr ausgeführt werden.

Guards
Guards steuern die Aktivierung einer *Stage*. Es müssen bestimmte Voraussetzungen erfüllt sein, damit die *Stages* ausgeführt werden können. *Guards* beinhalten somit Eingangskriterien zu einer *Stage*.
Guards und Meilensteine werden als datengesteuerte Ausdrücke beschrieben. Diese werden Sentries (Wächter) genannt. Beide zusammen sind die Hauptelemente, die den Prozessablauf der Artefaktinstanz festsetzen.

Der Vorteil eines deklarativen GSM-basierten Lebenszyklus Modells ist, dass es sehr leicht zu erstellen und zu bearbeiten ist. Gleichzeitig entsteht dadurch jedoch der Nachteil, dass es sehr kompliziert zu lesen und zu verstehen ist. [34]

Nachdem nun der artefaktzentrierte Ansatz erläutert wurde, wird im Folgenden mit dem fallzentrierten Ansatz ein weiterer datenzentrierter Ansatz dargestellt und diskutiert.

2.4.3 Fallzentrierter Ansatz (case-handling Approach)

Das Case Handling (CH) ist aufgekommen, um Wissensarbeiter in Applikationen zu unterstützen, die mehr Flexibilität brauchen, als der Prozessfluss von klassischen BPM-Systemen bietet. [30]
Die Prozessüberwachung im CH-Konzept ist eine Kombination aus den aktivitätenzentrierten und den artefaktzentrierten Kontrollflussprinzipien. Es existiert immer noch ein logischer Fluss, jedoch können berechtigte Benutzer eines Falls Aktivitäten erneut ausführen, oder auch überspringen. [35] Die Entwicklung von fallbasierten Ansätzen entstand durch die Grenzen der klassischen PrMS. Hierbei sind für die Benutzer nur atomare Daten sichtbar, die bei der Ausführung einer Aktivität benötigt werden. Diese beschränkte Sicht verhindert einen Zugriff auf Daten, die außerhalb des Kontrollflusses liegen. Die fehlende Flexibilität führte zu der Entwicklung eines datenzentrierteren Ansatzes, der wissensintensive Geschäftsprozesse unterstützen kann. [36]

Der Unterschied zwischen einem Fall und einem Prozess zeigt sich in der Herangehensweise, wie ein Ziel erreicht wird. Der Prozess behandelt die Aufgaben Schritt für Schritt, bis der Prozess abgeschlossen ist. Dagegen zeigt der Fall nur die Arbeit an, die erledigt werden soll. [37] Der Benutzer steuert den Ablauf selbst, da kein Sequenzfluss vorhanden ist.

Das zentrale Konzept von CH richtet sich auf den Geschäftsfall (case) und die Daten, die zu jedem Zeitpunkt von einem berechtigen Benutzer eingesehen werden können. [38] Weiterhin beinhaltet ein Fall neben den Daten auch alle Dokumente, verknüpfte Artefakte, Vorgaben, Regeln, Analysen und andere Informationen, die notwendig sind, um den Fall zu bearbeiten. Die Ausführung einer Aktivität basiert auf den Daten und deren Werten, welche mit einem Fall verknüpft sind. Somit sind die Daten verantwortlich für den Fortschritt des Prozesses und nicht, wie im klassischen PrMS, der Kontrollfluss eines Prozesses.

Es werden drei Datenelemente unterschieden, nämlich freie, obligatorische und beschränkte Datenelemente.
Freie Datenelemente sind nicht direkt mit einer bestimmten Aktivität verbunden, sondern werden dem Fall selbst zugeordnet und können zu jeder Zeit bearbeitet werden.

Obligatorische Datenelemente sind mit einer oder mehreren Aktivitäten verknüpft. Wenn das Datenelement zwingend innerhalb einer Aktivität bearbeitet werden muss, muss ein Wert gesetzt werden, um die Aktivitäten zu beenden.
Als dritte Form gibt es beschränkte Datenelemente. Diese sind auch mit einer oder mehreren Aktivitäten verknüpft. In diesem Fall muss eine Aktivität aktiviert sein, damit ein Wert in das Datenelement eingetragen werden kann.

Die Erweiterung von CH erlaubt drei verschiedene Rollen für die Ausführung von Aktivitäten. Die Rollen können für jede Aktivität einzeln gesetzt werden:
Eine ausführende Rolle (execution role), die das Ausführen einer Aktivität erlaubt. Des Weiteren bietet CH eine wiederholende Rolle (redo role), wodurch die Möglichkeit entsteht, ausgeführte Aktivitäten erneut auszuführen.
Ein wichtiges Element für datenzentrierte Ansätze stellt auch CH eine überspringende Rolle (skip role) dar. Dadurch entsteht eine erweiterte Flexibilität in der Ausführung einer Aktivität, da es dem Benutzer ermöglicht wird Aktivitäten wegzulassen oder zu überspringen. Ist ein Datenobjekt beispielsweise schon mit einem passenden Wert belegt, wird die Ausführung einer Aufgabe überflüssig und kann übersprungen werden.

Um eine kontextspezifische Sicht auf Falldaten anbieten zu können, werden mehrere Aufgaben mit einem Formular verbunden. Somit kann flexibel auf mehrere Aktivitäten gleichzeitig Einfluss genommen werden. Außerdem kann ein Formular auch direkt mit einem Fall selbst verbunden sein. (siehe Abb. 2.7).
Ein Benutzer erhält zwar mehr Freiheit bei der Ausführung eines Falls, muss aber auch den ganzen Fall im Auge behalten. Weiterhin besitzt jeder Benutzer lesende Rechte auf alle Fälle, die er bearbeitet.

Der Ansatz des CH wird bei der Umsetzung unterschiedlicher Frameworks integriert und findet auch Anwendung in der Forschung zu PHILharmonicFlows, welches im nächsten Abschnitt genauer betrachtet werden soll.

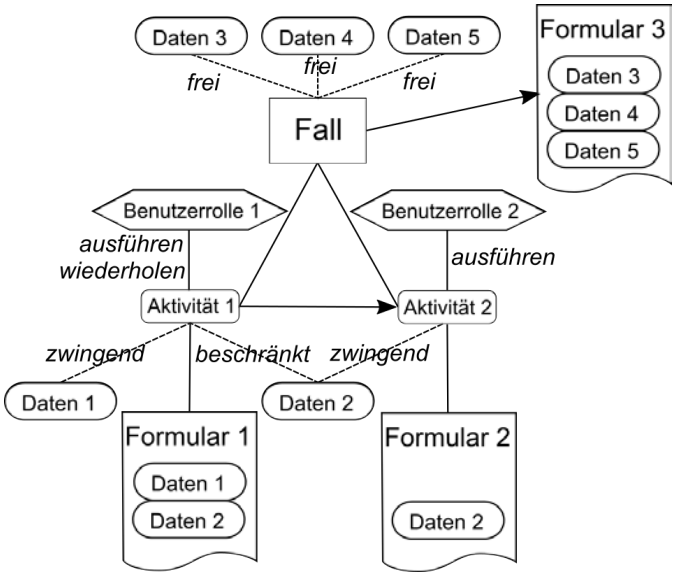


Abbildung 2.7: Aufbau eines fallzentrierten Ansatzes, vergleichbar [38]

2.5 Objektorientierter Ansatz

Ein objektorientierter Ansatz bietet eine Verknüpfung von Prozessen und Daten. Dabei werden auch die Funktionen und Benutzer berücksichtigt. Wie bereits erwähnt, ermöglicht dies eine enge Integration aller vier Hauptgeschäftsperspektiven und führt zu einem datenzentrierten Prozessmanagementansatz, indem der Prozessfortschritt über die Daten gesteuert wird. Der Unterschied in einem objektorientierten Konzept besteht darin, dass die Daten als Objekttypen organisiert werden und ein bestimmtes Verhalten aufweisen. Anhand dieses Verhaltens wird aufgezeigt, welche Daten zum Fortführen eines Prozesses benötigt werden. Weiterhin kann damit die Beziehung zwischen verschiedenen Objekten beschrieben werden. [39] Dieser Ansatz wird mit Hilfe des PHILharmonicFlows Frameworks im Folgenden genauer erläutert.

2.5.1 PHILharmonicFlows

Das Projekt PHILharmonicFlows [40] ist ein objektorientiertes PrMS Framework, welches sich aus einer Laufzeitumgebung und einer Entwurfsebene zusammensetzt. Die Lauf-

zeitumgebung wird dabei automatisch anhand von zuvor definierten Modellen generiert und enthält zwei Sichtweisen. Bei der ersten handelt es sich um eine datenorientierte Sicht, welche Übersichtstabellen beinhaltet und damit einen Überblick über alle Datenobjekte und deren Attribute auflistet. Diese wird vom System automatisch generiert. Die zweite ist eine prozessorientierte Sicht. Diese wird ebenfalls automatisch erstellt und hält die Arbeitslisten für die Bearbeiter des Systems bereit. Die prozessorientierte Sicht beinhaltet weiterhin die Überwachungselemente für das Framework. Die Manipulation der Daten erfolgt über formularbasierende Aktivitäten. Dabei werden alle benötigten Formulare dynamisch zur Laufzeit generiert und für den jeweiligen Bearbeiter angepasst zur Verfügung gestellt. In einem objektorientierten System können Datenänderung zur Laufzeit zu jedem beliebigen Zeitpunkt vorgenommen werden. Damit können Daten auch außerhalb der Prozesse manipuliert werden. Somit kann eine Verknüpfung eines klassischen Prozessmanagements mit Elementen des CH-Ansatzes festgestellt werden, die in PHILharmonicFlows integriert werden. [41]
Die Entwurfsebene unterteilt sich in vier Hauptkomponenten. Die ersten beiden sind für die Modellierung der *Datenstruktur* und der *Prozessstruktur* verantwortlich. Diese zwei Strukturen beinhalten das Grundkonzept des artefaktzentrierten Ansatzes, wobei die Artefakte in PHILharmonicFlows als Objekttypen beschrieben werden. [42] Daneben stehen die Komponenten der *Benutzerintegration* und die *Koordinationskomponenten*. Zur Entwurfszeit können weiterhin Black-Box-Aktivitäten implementiert werden, um externe Services oder Systeme in das System zu integrieren. Damit liegen die Zuständigkeiten und Daten jedoch außerhalb der Funktionslogik des Systems. Im Folgenden werden die vier Hauptkomponenten erläutert:

Datenstruktur

In PHILharmonicFlows wird zunächst das Datenmodell festgelegt, da es sich um ein datenzentriertes PrMS handelt, indem die Ausführung über Objektattributänderungen gesteuert wird. Dafür wird eine *Datenstruktur* definiert, die aus *Objekttypen*, den dazugehörigen *Attributen* und *Relationen* gebildet wird. Die Daten werden in einer relationalen Datenbank verwaltet und sind dadurch eindeutig identifizierbar.
Der Objekttyp stellt einen Container dar, welcher eine Menge von Attributwerten enthält

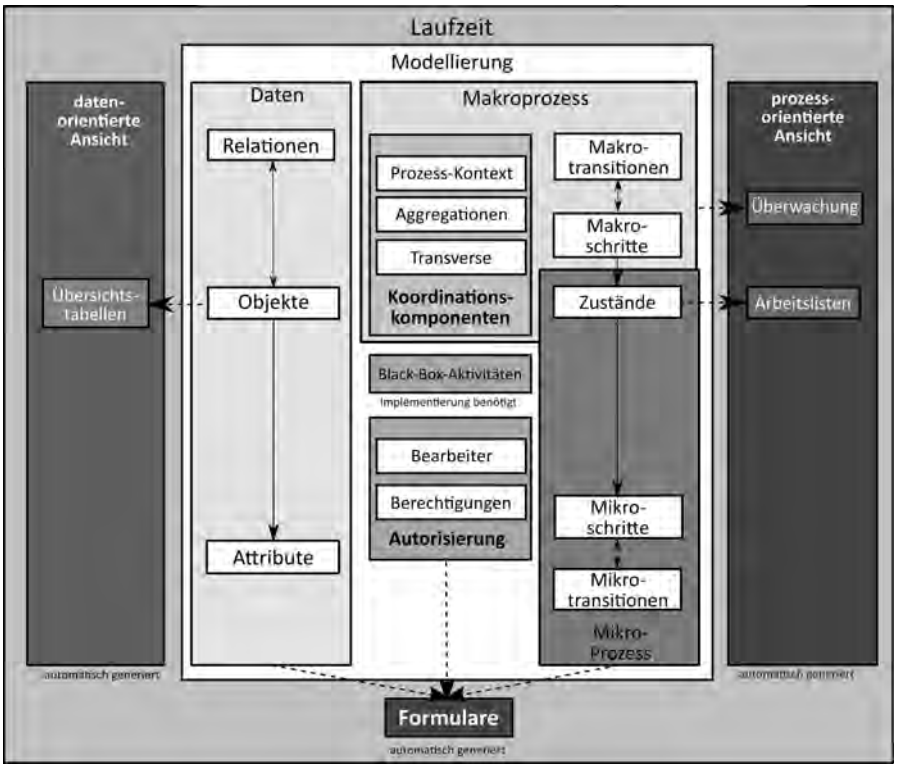


Abbildung 2.8: Modell des PHILharmonicFlows Frameworks, vergleichbar [43]

und aus atomaren Werten besteht. Die Datenstruktur entsteht dadurch, dass mit Hilfe von Relationen Beziehungen zwischen den verschiedenen Objekttypen beschrieben werden. Die Objekttypen lassen sich hierarchisch in unterschiedlichen Ebenen einordnen, wodurch verschiedene Beziehungsarten abgeleitet werden können. Ein Zyklus durch eine Relationen ist nicht zulässig und muss vor der Einordnung eines Objekttypen aufgelöst werden. Weiterhin können keine Relationen zwischen Objekttypen der gleichen Ebene gebildet werden.
Ein spezieller Objekttyp ist der Wertetyp. Dieser kann in keine Datenebene eingeordnet werden, da er keine Relation zu anderen Objekttypen enthält. Damit werden Wertebereiche für Attribute definiert, die zur Laufzeit eingesetzt werden.

Prozessstruktur

Das Prozessmodell besteht aus zwei unterschiedlichen Stufen der Granularität. Diese werden durch den *Makroprozess* und den *Mikroprozess* dargestellt.
Der *Mikroprozess* besteht aus *Mikroschritten*, *Mikrotransitionen* und *Zuständen*. Dieser beschreibt dabei das Verhalten eines bestimmten Objekttyps und kann somit auch als Lebenszyklus definiert werden. Jedem Objekttyp ist genau ein *Mikroprozess* zugeordnet.
Der *Mikroschritt* definiert Bedingungen, die für ein bestimmtes Attribut eines Objekttyps gelten. Erst wenn diese Bedingungen erfüllt sind, kann der Prozess weiter fortschreiten. Dabei gibt es drei Arten von Mikroschritten:
Leere Mikroschritte, welche keine Bedingung enthalten und somit immer sofort erfüllt sind.
Atomare Mikroschritte, bei denen ein Attribut mit einem beliebigen Wert angegeben werden muss, um die Bedingung zu erfüllen. Die letzte Art sind die *wertespezifischen Mikroschritte*. Diese benötigen wie atomare Mikroschritte ein Attribut, sind jedoch erst erfüllt, wenn dieses Attribut einen bestimmten Wert annimmt.
Die *Mikrotransitionen* modellieren die Übergänge zwischen den einzelnen Mikroschritten. Daraus entsteht ein zyklusfreier, gerichteter Graph, in dem die Knoten durch Mikro-Schritte und die Kanten durch die Transitionen dargestellt werden. Dabei kann sowohl zwischen impliziter und expliziter, als auch zwischen interner und externer Transition unterschieden werden.

In einem Mikroprozess sind Objekttypen häufig durch Beziehungen von anderen Objekttypen abhängig. Deshalb werden die Objekttypen in Ebenen unterteilt. Somit zeigt ein Mikroschritt einer niedrigeren Ebene immer auf die darüber liegenden Ebenen. Daher muss das System eine Möglichkeit besitzen, dass Abhängigkeiten zur Laufzeit gegebenenfalls synchronisiert werden können.
Dazu dient der *Makroprozess*. Dieser ist für die Objektinteraktionen und damit für die Synchronisation verschiedener Objekttypen untereinander verantwortlich. Der Makroprozess fasst Mikroprozesse zusammen und ermöglicht dem Modellierer somit eine übersichtliche Darstellung in Form eines abstrakten Prozesses. Der Makroprozess besteht wie der Mikroprozess aus *Makroschritten* und *Makrotransitionen*.
Ein Makroschritt wird genau einem Objekttyp und einem dazugehörigen Zustand zugeordnet.
Die Makrotransition stellen die Verbindung zwischen zwei Makroschritten her. Auch bei den Makrotransitionen werden dadurch Abhängigkeiten aufgezeigt. Je nachdem auf welchen Objekttyp eine Transition verweist und in welcher Datenebene sich dieser Objekttyp befinde, werden verschiedene Transitionsarten definiert. Diese werden in Top-Down, Bottom-Up, Transverse und Self unterteilt.

Benutzerintegration

Zur Verwaltung von Benutzerzugriffen auf Objektinstanzen werden *rollenbasierte Autorisierungstypen* in PHILharmonicFlows definiert. Dazu werden *Benutzertypen* in Form von Objekttypen angelegt und mit einer oder mehrerer Rollen verbunden. Damit werden Rechte nicht an Benutzertypen sondern an Rollentypen gebunden. Über diese werden dann Rechte für das Lesen, Schreiben, Erstellen und Löschen auf Objektinstanzen und anderen Attributen definiert. Dabei können auch obligatorische und optionale Rechte vergeben werden. So können Zugriffsberechtigungen abhängig von den Attributwerten im Datenmodell vergeben werden. Während der Laufzeit kann mit Hilfe einer Rechtetabelle für einen Objekttyp festgelegt werden, ob eine Benutzerrolle Zugriff auf bestimmte Attribute erhält oder sogar verpflichtet wird bestimmte Attribute zu schreiben, um eine erfolgreiche Bearbeitung eines Mikroprozesses zu erreichen. Zusätzlich wird die Rechtetabelle zur Laufzeit eingesetzt, um automatisierte Arbeitslisten und Formulare

zu erstellen. So können Aufgaben den richtigen Benutzern zugeordnet und Formulare abhängig von den Rechten eines Bearbeiters angepasst werden.

Koordinationskomponenten
Für die Herausforderungen an ein gesamtheitliches Prozessmanagement, welche in Kapitel (3) genauer beschrieben werden, spielen auch die Koordinationskomponenten eine wichtige Rolle. Diese ermöglichen eine variable Anzahl an Objektinstanzen. Weiterhin kann die Kardinalität auch zur Laufzeit angepasst werden.

Mit diesen vier Komponenten werden in PHILharmonicFlows die Abhängigkeiten zwischen den Daten und Prozessen bereits auf Entwurfsebene eingebunden und bieten somit eine eine Unterstützung für datenzentrierte und wissensintensive Geschäftsprozesse. Mit diesem Framework lassen sich somit auch sehr komplexe Funktionen direkt innerhalb des Systems umsetzen, ohne eine harte Verdrahtung der Prozesslogik zu erhalten. Damit wird eine allgemeine, anpassbare Prozessunterstützung erreicht, wie sie innerhalb eines klassischen PrMS nicht umgesetzt werden könnte. Dies zeigt sich auch in der Integration der Bearbeiter von PHILharmonicFlows. Diese erhalten ebenfalls einen integrierten Zugang auf die Daten, Prozesse und Funktionen des Systems. Dabei kann mit Hilfe der Benutzerautorisierung der Zugriff auf Attribute über Listen und Formulare dynamische zur Laufzeit angepasst werden. Die Herausforderungen eines gesamtheitlichen, datenzentrierten Systems werden im nächsten Kapitel detailliert untersucht und auch anhand von PHILharmonicFlows evaluiert.

2.6 Zusammenfassung

Die automatisierte Unterstützung von Geschäftsprozessen ist in der heutigen Zeit unabdingbar für Unternehmen und wird in der Praxis auch häufig eingesetzt. In diesem Kapitel wurden die wichtigsten Ansätze und damit die Grundlagen für PrMS erläutert. Viele Frameworks bauen auf diesen Ansätzen auf, erweitern sie oder fassen mehrere Ansätze zusammen, was in der Einführung des PHILharmonicFlows Frameworks deutlich sichtbar ist.
Es wurden bereits die ersten Schwächen eines rein aktivitätenbasierten Prozessma-

nagement Ansatzes aufgezeigt. Weiterhin wurde die zunehmende Bedeutung von wissensintensiven und schwach strukturierten Prozessen herausgearbeitet. Neben den wissensintensiven Prozessen wurden auch verschiedene Ansätze der Prozessmodellierung dargestellt. Der deklarative Ansatz und der regelzentrierte Ansatz stellen dabei erste Schritte zu datenzentrierter und wissensintensiver Prozessunterstützung dar. Es existieren weitere Ansätze, die in dieser Arbeit nicht näher erläutert werden. Diese untersuchen beispielsweise eine entscheidungsbasierte Prozessmodellierung, bei der ebenfalls durch Wissensarbeiter gesteuerte Prinzipien im Mittelpunkt stehen. Ebenso hat die OMG dieses Jahr den Decision Model and Notation (DMN)-Standard auch für den entscheidungsbasierten Ansatz herausgegeben, was die Bedeutung von Wissensarbeitern und die Wichtigkeit für die Modellierung loser Strukturen nochmals hervorhebt.

Durch den Anstieg an KiPs in Unternehmen steigt auch die Nachfrage nach einem datenorientierten beziehungsweise objektzentrierten Prozessmanagement. Jedoch sind für die Umsetzung eines solchen Systems viele verschiedenen Faktoren zu berücksichtigen. Im folgenden Kapitel werden daher zunächst die vorgestellten Ansätze mit den Herausforderungen an ein objektzentriertes System abgeglichen. Diese Herausforderungen zeigen die Verknüpfung der Daten mit den Prozessen, Funktionen und Benutzern und stellen die Grundlagen für ein flexibles und datenzentriertes System dar.
Durch die Ermittlung der datenzentrierten Eigenschaften vieler vorhandener Forschungsansätze und Frameworks, konnte mit PHILharmonicFlows ein objektzentriertes Framework entwickelt werden. [43] Dieses vereint klassische PrMS-Ansätze und datenzentrierte Ansätze [41] und versucht so die Anforderungen an heutige PrMS bestmöglich umzusetzen. Durch die Wissensarbeiter und deren Forderungen nach flexibleren Arbeitsbedingungen innerhalb der Systeme, wurde der CMMN-Standard eingeführt. Er besteht aus mehreren Ansätzen, wie dem CH und den artefaktbasierten Sichten und wird in Kapitel 4 genauer untersucht.

3

Objektzentriertes Prozessmanagement

Im letzten Kapitel wurden bereits die Grenzen von klassischen PrMS aufgezeigt. Deshalb streben viele Forschungsansätze ein neues Ziel an. Dieses beinhaltet die Integration der vier Hauptgeschäftsperspektiven in ein PrMS. Daten, Prozesse, Funktionen und Benutzer sollen in ein System eingebunden und verknüpft werden, um ein objektzentriertes und damit gesamtheitliches, datenorientiertes Prozessmanagement zu erhalten. Die Abbildung 2.2 verbildlicht dieses Ziel.
Jedoch werden nicht alle Phasen dieser integrierten Sicht umgesetzt, so dass sich die Ansätze nur auf bestimmte Eigenschaften einer objektzentrierten Idee beschränken.

Dieses Kapitel zeigt die Eigenschaften, die nötig sind, um ein engeres Zusammenwirken der vier Perspektiven zu erlangen. Dabei wird die Frage gestellt, wie eine Verknüpfung der Daten mit den Benutzern erzielt werden kann und wie die Flexibilität eines Systems gesteigert wird. Dazu werden die bisher vorgestellten Ansätze gegen die Anforderungen

und Eigenschaften eines objektzentrierten PrMS abgeglichen. Zunächst werden hierfür der imperative und der deklarative Ansatz näher untersucht. Danach werden die beiden datenzentrierten Systeme, also der artefaktzentrierte und der fallzentrierte Ansatz, mit den Anforderungen verglichen, die für ein objektzentriertes Modell erfüllt sein müssen. Diese beiden Ansätze werden evaluiert, weil sie Einfluss auf das PHILharmonicFlows Framework nehmen. Dieses Framework zeigt im letzen Absatz, was nötig ist, damit alle Herausforderungen in ein System integriert werden können.

3.1 Herausforderungen an ein gesamtheitliches Prozessmanagementsystem

Um objektzentrierte Prozesse zu unterstützen, werden Voraussetzungen benötigt, die in dem Paper [44] mit Hilfe von Fallstudien zu verschiedenen Bereichen zusammengetragen wurden. Dabei wurden der medizinische Bereich, das Personalwesen, die Auftragsabwicklung und weitere Bereiche untersucht. Weiterhin wird eine Literaturstudie herangezogen [40]. Diese wertet Forschungsansätze im Bereich datenzentriertes Prozessmanagement aus, die ebenfalls die Wichtigkeit der Prozess- und Datenintegration thematisieren. Auch die Aspekte der Modellierungs- und Laufzeitumgebung [45], sowie der Benutzerintegration für ein objektzentriertes PrMS werden berücksichtigt. [8] Die Anforderungen für ein objektorientiertes PrMS werden anhand dieser Paper im nächsten Abschnitt zusammengefasst. Dabei orientiert sich der Abschnitt an den vier Hauptgeschäftsperspektiven.

3.1.1 Daten

Der Begriff *objektzentrierter Prozess* lässt sich am besten anhand der Geschäftsperspektive *Daten* nachvollziehen. Denn die Verwaltung der Daten basiert auf Objekttypen, welche mit anderen Objekttypen in Beziehung stehen.
Des Weiteren beschreibt die Datensicht die Möglichkeit für Benutzer, zu jeder Zeit auf die Daten zugreifen zu können. Auch optionale Aktivitäten werden berücksichtigt. Zur

Laufzeit kann die Anzahl der Objektinstanzen durch verschiedene Kardinalitätsbeschränkungen begrenzt werden.

H1: Datenintegration
Daten werden als Objekttypen verwaltet. Diese besitzen eine Menge von Attributen und stehen in Relation zu anderen Objekttypen (siehe Abb. 3.1).

H2: Datenzugriff
Der Zugriff auf die Daten ist zu jedem Zeitpunkt möglich, nicht nur bei der Ausführung der jeweiligen Aktivität. So kann der Benutzer auch auf Datenelemente außerhalb seiner Arbeitsliste zugreifen. Dies setzt natürlich vorhandene Nutzungsrechte voraus.

H3: Kardinalität
Zur Laufzeit kann für jeden Objekttyp eine unterschiedliche Anzahl an Objektinstanzen auftreten. Objektinstanzen können sich dadurch in der Anzahl der gegenseitigen Beziehungen unterscheiden. Diese sollten durch Kardinalitätsbeschränkungen auch begrenzt werden können.

H4: Obligatorische Informationen
Um von einem Status zum nächsten Objektinstanzstatus zu gelangen, müssen bestimmte Attributwerte zwingend gesetzt sein. Eine Aktivität enthält ein Formular mit obligatorischen Eingabefeldern. Dieses Formular wird Benutzern, die über die Bearbeitungsrechte verfügen, über eine Arbeitsliste zugeteilt. Bestimmte Felder des Formulars beinhalten obligatorisch auszufüllende Attributwerte, daneben können jedoch auch obligatorisch auszufüllende Felder vorkommen, die nicht zwingend für den Fortlauf des Prozesses notwendig sind.

3.1.2 Aktivitäten

Bei den Aktivitäten können *Formularbasierende Aktivitäten* und *Black-Box-Aktivitäten* unterschieden werden. In objektzentrierten Prozessen wird während der Ausführung der Aktivitäten normalerweise der Wert der Objektattribute benutzt, um mit dem Prozess fortzufahren. Dabei benötigen manche Aktivitäten zur Ausführung die Eingabe des Benutzers. Dies wird mit Hilfe eines Formulars (user-forms) umgesetzt. Andere Aktivitäten

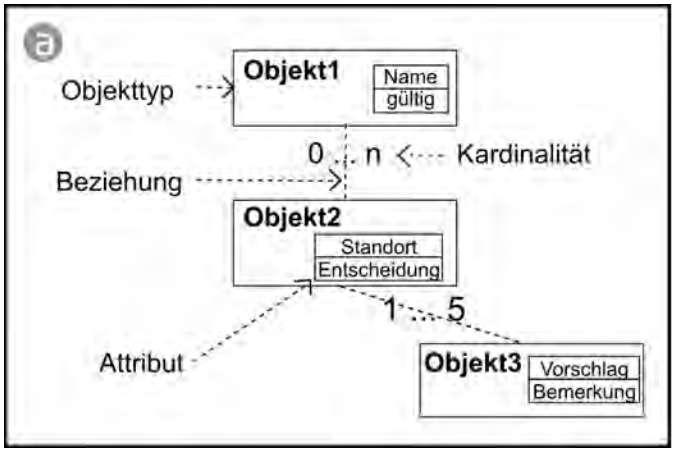


Abbildung 3.1: Datenstruktur zur Modellierzeit, vergleichbar [44]

werden durch Black-Box-Aktivitäten integriert. Diese sind an entsprechende Prozesse gebunden, welche zur Laufzeit ausgeführt werden.
Bei der Bearbeitung sollte es dem Benutzer ermöglicht werden, zwischen unterschiedlichen Aktivitätenarten frei zu entscheiden, um das geforderte Ziel bestmöglich zu erreichen. Ein PrMS kann den Benutzer dabei unterstützen, indem es anzeigt, ob und wann welche Aktivitäten bearbeitet werden müssen und welche optional bearbeitet werden können. Dies kann sich während der Ausführung durch die Bearbeitung von Attributwerten auch ändern.

H5: Formularbasierende Aktivitäten

Diese Aktivitäten beinhalten eine Menge an atomaren Handlungen. Jede Aktivität ist entweder mit einem Eingabefeld verknüpft, um Werte eines Objektattributs zu schreiben, oder mit einem Datenfeld, um Werte zu lesen. Welche Attribute in einer formularbasierenden Aktivität gelesen oder geschrieben werden dürfen, hängt sowohl vom beteiligten Benutzer ab, als auch vom Status der Objektinstanz. So entsteht eine große Anzahl an unterschiedlichen Formularen. Diese formularbasierenden Aktivitäten sollten zur Laufzeit automatisiert erzeugt werden, um hohe Kosten zu vermeiden, die durch die Implementierung aller Formulare entstehen würden.

H6: Black-box-Aktivitäten

Black-box-Aktivitäten setzen komplexe Berechnungen in Gang oder erlauben die Inte-

gration von erweiterten Funktionalitäten. Um eine korrekte Ausführung zu gewährleisten, werden Attributwerte mit Vorbedingungen verknüpft. Im Gegensatz zu *formularbasierenden Aktivitäten*, welche zur Laufzeit automatisiert erzeugt werden, ist für jede Black-box-Aktivität eine Implementierung notwendig.

H7: Unterschiedliche Aktivitätengranularität
Bei der Granularität werden drei Arten unterschieden, die instanzbestimmte, die kontextbestimmte sowie die Batch-Aktivitäten.

- Instanzbestimmte Aktivitäten (instance-specific): Formularbasierende Aktivitäten gehören zu genau einer Objektinstanz. Bei der Ausführung einer solchen Aktivität können Attribute dieser Objektinstanz in einem Formular gelesen, geschrieben oder aktualisiert werden. Black-Box-Aktivitäten benötigen nur Eingabeparameter, welche sich auf Attribute einer bestimmten Objektinstanz beziehen.
- Kontextbestimmte Aktivitäten (context-sensitive): Formularbasierende Aktivitäten enthalten zusätzliche Eingabefelder, welche darüber oder darunter liegende Ebenen miteinbinden. Black-box-Aktivitäten benötigen Attributwerte von verschiedenen Objektinstanzen als Inputparameter.
- Batch-Aktivitäten (batch acticities): Batch-Aktivitäten bieten dem Benutzer die Möglichkeit, eine Auswahl an Objektinstanzen auf einmal auszuführen. Ein Formular ändert die Attributwerte aller ausgewählten Objektinstanzen. Die Ausführung mehrerer Black-Box-Aktivitäten zur gleichen Zeit sollte auch möglich sein.

Dem Benutzer sollte es überlassen werden, welchen Aktivitätentyp er auswählt, um sein Ziel zu erreichen. Dabei gehören bei instanzbestimmten Aktivitäten alle Aktionen zu Attributen einer bestimmten Objektinstanz. Kontextbestimmte Aktivitäten verbinden Aktionen, welche zu verschiedenen Objektinstanzen gehören, die jedoch miteinander in Verbindung stehen. Diese können vom gleichen oder von einem unterschiedlichen Typ sein. Bei Batch-Aktivitäten sind mehrere Objektinstanzen eingebunden, jedoch vom gleichen Typ.

H8: Obligatorische und optionale Aktivitäten
Während der Ausführung einer Objektinstanz sind manche Attributwerte zwingend notwendig, bevor mit der Prozessausführung fortgefahren werden kann. Es sollte jedoch

auch möglich sein, zusätzliche Aktivitäten optional auszuführen. Dadurch können bestimmte Attributwerte auch dann geschrieben werden, wenn dies nicht obligatorisch ist.

H9: Kontrollfluss innerhalb eines Benutzerformulars
Ob bestimmte Objektattribute bei der Ausführung einer Aktivitäten gesetzt werden müssen, hängt unter Umständen auch von anderen Objekt-Attributwerten ab. Somit kann es beim Ausfüllen eines Formulars on the fly dazu kommen, dass auch andere Attribute obligatorisch auszufüllen sind.

3.1.3 Prozesse

Zur Laufzeit soll es möglich sein, die Prozessausführung abhängig von den Daten zu gestalten und dynamisch auf Attributwertänderungen zu reagieren. Deshalb ist es unabdingbar, den Status des Prozesses mit Attributwerten zu verknüpfen. Das Objektverhalten bestimmt, in welcher Reihenfolge Objektattribute geschrieben werden und welchen Wert diese Attribute annehmen dürfen. Weiterhin wird festgelegt, welche Benutzer die Attribute beschreiben dürfen. Zusätzlich muss sichergestellt werden, dass obligatorische Daten während der Prozessausführung zur Verfügung stehen. Deshalb sollte das Objektverhalten an Datenbedingungen gekoppelt werden, und nicht an Black-Box-Aktivitäten. [46]
Für jeden Objekttyp können mehrere Objektinstanzen existieren. Objektinstanzen werden zu beliebigen Zeitpunkten erstellt und gelöscht, so dass eine komplexe Datenstruktur entsteht. Dies geschieht dynamisch zur Laufzeit. Die Objektinteraktionen regeln dann den Ablauf zwischen den Instanzen. Dabei müssen sowohl die Beziehungen der Instanzen untereinander, als auch deren asynchrone Ausführung berücksichtigt werden. Um das Ziel des Prozesses zu erreichen, müssen, wie bereits erwähnt, einige Aktivitäten obligatorisch ausgeführt werden, wohingegen andere optional sind. Dabei sollte es möglich sein, eine Aktivität mehrmals ausführen zu können, falls dies notwendig sein sollte. Der Benutzer hat weiterhin auch die Möglichkeit, auf die Anzahl der Objektinstanzen einzuwirken.

H10: Objektverhalten (object-behavior)
Beim Objektverhalten handelt es sich um die erste Granularitätsebene. Hier wird dem Prozess eine individuelle Objektinstanz zugeschrieben. Für jeden Objekttyp existiert eine dazugehörige Prozessdefinition. Wird zur Laufzeit eine Prozessinstanz erstellt, ist diese an die Objektinstanz gekoppelt. Für jeden Objekttyp kann es zur Laufzeit mehrere Objektinstanzen geben. Das Objektverhalten bestimmt, in welcher Reihenfolge Objektattribute geschrieben werden (siehe Abb. 3.2).

H11: Objektinteraktionen (object-interactions)
Bei der Objektinteraktion handelt es sich um die zweite Granularitätsebene. Diese betrachtet das Zusammenspiel, das zwischen den Instanzen der verschiedenen Objekt-Typen stattfindet. Einzelne Prozessinstanzen können gleichzeitig und unabhängig voneinander ausgeführt werden, unter Beachtung der Objektbeziehungen und der Kardinalitätsbeschränkungen.
Diese müssen jedoch bei Bedarf synchronisiert werden. Dabei soll die Synchronisation nicht an bestimmten Punkten im Prozess festgesetzt werden, um den Prozess asynchroner und flexibler gestalten zu können. Ob eine Objektinstanz in einen bestimmten Status wechselt, hängt vom Status anderer Objektinstanzen ab. Dafür müssen sowohl *Top-Down*- und *Bottom-Up*-Abhängigkeiten, als auch quergerichtete und transitive Abhängigkeiten berücksichtigt werden.

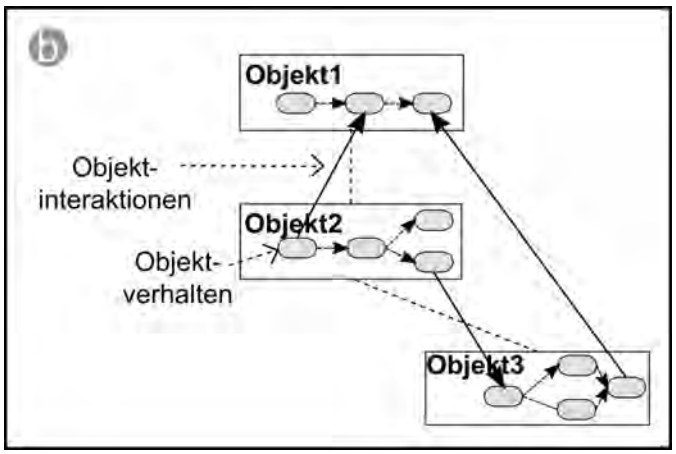


Abbildung 3.2: Prozessstruktur zur Modellierzeit, vergleichbar [44]

H12: Prozessorientierte Sicht
Während der Prozessausführung müssen einige Aktivitäten obligatorisch ausgeführt werden, andere nur optional. Die obligatorischen Aktivitäten werden den berechtigten Benutzern über Arbeitslisten zugeteilt, um sicherzustellen, dass die Aktivitäten zur richtigen Zeit ausgeführt werden.

H13: Flexible Prozessausführung
Obligatorische Aktivitäten sind notwendig für die Prozessausführung. Sie entscheiden über die Notwendigkeit, ob ein Attributwert für die Weiterführung des Prozesses geschrieben werden muss. Wird der Attributwert über eine optionale Aktivität geschrieben, bevor er für die Prozessausführung relevant ist, kann die obligatorische Aktivität im späteren Verlauf automatisch übersprungen werden, wenn sie aktiviert wird.

H14: Wiederholte Ausführung von Aktivitäten
Benutzer sollten die Möglichkeit besitzen, eine bestimmte Aktivität nochmals auszuführen, beispielsweise um Attribute zu aktualisieren. Dies sollte selbst dann möglich sein, wenn die obligatorischen Objektattribute bereits geschrieben wurden.

H15:Benutzerentscheidungen
Es sollte mehrere Wege geben, um ein Prozessziel zu erreichen. Ein Grund für die Auswahl eines alternativen Ausführungswegs könnte eine bestimmte Benutzerentscheidung sein. Im Rahmen der vorgegebenen Kardinalitäten sollte der Benutzer auch über die Zahl der Objektinstanzen entscheiden, die zur Laufzeit erzeugt werden.

3.1.4 Benutzerintegration

In klassischen PrMS werden vom Benutzer auszuführende Aktivitäten zur Modellierzeit über *Benutzerrollen (user roles)* festgelegt. Somit ist es möglich, dass ein Benutzer auch Attributwerte von Aktivitäten einsehen kann, die er nicht bearbeitet. Deshalb sollte auch die Objektinstanz, die von einer Aktivität ausgeführt wird, bei der Rechtevergabe miteinbezogen werden. Weiterhin spielt der Fortschritt eines Prozesses eine Rolle bei der Vergabe von Rechten. Außerdem muss bei der der Rechtevergabe auch die Unterscheidung von optionalen und obligatorischen Aktivitäten berücksichtigt werden.

H16: Datenautorisierung

Um einen Zugriff auf Daten zu jedem beliebigen Zeitpunkt bereitzustellen, müssen Berechtigungen erteilt werden. Diese regeln das Erstellen und Löschen von Objektinstanzen sowie das Lesen und Schreiben der Attribute.
Jedoch muss verhindert werden, dass die Änderungen an Attributen dem Objektverhalten widersprechen. Deshalb muss der Fortschritt eines Prozesses in die Betrachtung miteinbezogen werden, wenn Berechtigungen zur Bearbeitung von Objektattributen ausgegeben werden. Damit wird verhindert, dass bereits bestätigte Attributwerte im Nachhinein geändert werden und es zu Inkonsistenzen kommt.
Dies stellt sowohl für kontextbestimmte Aktivitäten, als auch für Batch-Aktivitäten eine große Herausforderung dar, da die Änderungen an den Attributwerten für alle ausgewählten Instanzen zulässig sein müssen.

H17: Prozessautorisierung

Jeder obligatorischen Aktivität sollte zur Laufzeit mindestens ein Benutzer, oder eine Benutzerrolle zugeordnet sein. In Bezug auf formularbasierende Aktivitäten muss jeder ausführende Benutzer die passenden Berechtigungen zum Lesen und Schreiben der entsprechenden Attributwerte besitzen.

H18: Differenzierung zwischen Autorisierung und Bearbeiterzuordnung

Bei der Bearbeiterzuordnung für obligatorische Aktivitäten muss differenziert werden, ob Benutzer eine Attributänderung durchführen dürfen, oder sie innerhalb des Prozesses zwingend durchführen müssen. Bei der Ausführung von obligatorischen Aktivitäten müssen bestimmte Objektattribute ausgefüllt werden. Dabei dürfen nur die Benutzer diese Aktivitäten ausführen, die Schreibrechte besitzen. Einige Benutzer müssen Aktivitäten obligatorisch ausführen, andere Benutzer können berechtigt sein, solche Aktivitäten optional auszuführen. Daher sollte zwischen obligatorischen und optionalen Berechtigungen unterschiedenen werden können. Bei dem Wechsel in einen weiterführenden Prozessschritt sollte differenziert werden können, ob dieser Übergang explizit oder implizit erfolgen darf. Explizit heißt, durch die Ausführung der dazugehörigen obligatorischen Aktivität, implizit meint die Ausführung einer optionalen Aktivität.

H19: Vertikale Autorisierung und Bearbeiterzuordnung
In klassischen PrMS wird jeder Aktivität, die eine Benutzeraktion erfordert, ein Bearbeiterausdruck zugeordnet. Dabei spricht man von einer horizontalen Autorisierung. In einem objektzentrierten PrMS ist die Auswahl möglicher Bearbeiter nicht nur von der Aktivität abhängig, sondern auch von den Objektinstanzen, die innerhalb der Aktivität benötigt werden. Dies wird als vertikale Autorisierung bezeichnet.

3.1.5 Überwachung

Der Fortschritt der Prozesse sollte transparent und nachvollziehbar sein, auch wenn er wie bei objektzentrierten PrMS durch Beziehungen zwischen Objektinstanzen festgelegt wird. Denn nur durch eine gesamtheitliche Sicht kann das PrMS die Fortschritte eines Geschäftsprozesses überwachen und auswerten. Nur so können Verbesserungen in den Abläufen vorgenommen werden.

H20: Gesamtheitliche Sicht
Die Prozessüberwachung sollte eine umfassende Sicht beinhalten. Sowohl auf alle Objektinstanzen, die in einem Prozess beteiligt sind, als auch auf deren Abhängigkeiten.

Über die vier Geschäftsperspektiven Daten, Aktivitäten, Prozesse und Benutzer wurden zwanzig Herausforderungen erarbeitet, die als Grundlage für einen objektzentrierten Prozessmanagementansatz herangezogen werden können.
Dabei spielt nicht nur die Integration von Daten und Prozessen eine wichtige Rolle, auch die Integration der Benutzer ist für eine gesamtheitliche Sicht von Bedeutung. Denn durch die Trennung von obligatorischen und optionalen Aktivitäten steigt die Verantwortung der Benutzer in einem Prozess. Benutzer bekommen mehr Entscheidungsgewalt über die Ausführung. Zusätzlich steigt, durch die Integration der Daten, auch die Bedeutung einer adäquaten Vergabe und Beschränkung von Rechten. Inwieweit diese Anforderungen in den bekannten Ansätzen umgesetzt werden, wird im Folgenden diskutiert.

3.2 Evaluation der Herausforderungen gegen bereits vorgestellte Ansätze

Die Integration von Daten und Prozessen steht schon länger im Fokus verschiedener Forschungsarbeiten. Diese orientieren sich häufig an den Prozessmanagement-Ansätzen, die bereits vorgestellt wurden. Deshalb ist es sinnvoll, zu evaluieren, welche Stärken und Schwächen die Ansätze aufweisen. Weiterhin wird untersucht, inwiefern diese Ansätze die Herausforderungen für objektzentrierte Ansprüche bereits umsetzen. Zunächst wird hierfür der imperative Ansatz herangezogen.

3.2.1 Imperativer Ansatz

Die Eigenschaften eines imperativen Systems wurden bereits in Kapitel 2.3.1 vorgestellt. Mit BPMN wird der imperative Ansatz in einer Prozessmodellierungssprache verwendet, die auch in der praktischen Anwendung weit verbreitet ist. Imperative Ansätze beinhalten nur einige der Anforderungen, die für die Unterstützung von objektzentrierten Prozessen nötig sind. [44] Im Folgenden wird untersucht, inwieweit der imperative Ansatz den jeweiligen Herausforderungen der vier Hauptgeschäftsperspektiven gerecht wird.

Daten:
Die *Datenintegration* H1 wird nicht unterstützt, da diese auf Objekttypen, Attributen und Relationen basiert. Der imperative Ansatz hat jedoch keine Kenntnis darüber, welche Objektinstanzen während der Ausführung eines Prozesses bearbeitet werden. Der Zugriff auf Daten ist nur möglich, wenn eine Aktivität ausgeführt wird. Das bedeutet, eine Aktivität muss innerhalb des Kontrollflusses erst aktiviert werden, bevor auf die Daten zugegriffen werden kann.
Weiterhin ist ein Datenzugriff außerhalb des Kontrollflusses nicht vorgesehen und kann zu Inkonsistenzen führen. *Datenzugriff* H2 wird somit nur sehr eingeschränkt unterstützt.
Um die *Kardinalität* H3 zu beurteilen, muss zunächst auf die frei wählbare Prozessgranularität eingegangen werden. Der imperative Ansatz kann nicht zwischen dem Objektverhalten einer einzelnen Objektinstanz und dem dazugehörigen Prozess unterscheiden. Es existiert keine Methode, die regelt, wie viele Objekttypen in einer Prozessbeschrei-

bung vorkommen. Dadurch kann ein Prozess entweder fein- oder grobkörnig modelliert werden.
Feinkörniges Prozessmodell:
Hier wird die Ausführung einer Aktivität mit genau einer Prozessinstanz verknüpft.
Grobkörniges Prozessmodell:
Hier werden die Aktivitäten mit mehreren Objekttypen verbunden. Es existiert jedoch keine Methode, durch die *Kardinalität* H3 auf die Anzahl der zu erstellenden Objektinstanzen bei der Modellierung eines Prozess einzuwirken. Der imperative Ansatz kann auch nicht überprüfen, welche Datenattribute bereits erfolgreich geändert wurden, da dies innerhalb der Applikation geschieht und somit nicht eingesehen werden kann. Dies gilt somit auch für die *obligatorischen Informationen* H4. Diese Anforderungen werden daher auch nicht umgesetzt.

Aktivitäten:
Die Abläufe in imperativen PrMS sind abhängig von der jeweils vorausgehenden Aktivität. Damit muss auch darauf vertraut werden, dass benötigte Daten für die nächste Aktivität von der vorherigen Aktivität korrekt geschrieben werden. Dies kann jedoch nicht überprüft werden. Bei falschen oder nicht vorhandenen Werten kann es zu einem Fehler in der Ausführung oder zu einem inkorrekten Datenfluss kommen.
Mit *formularbasierenden Aktivitäten* H5 können fehlende Daten vom Benutzer nachgefordert werden. Dies ist hier nicht möglich, da die Formulare nicht automatisch und unabhängig vom Kontrollfluss aufgerufen werden können.
Black-Box-Aktivitäten H6 werden unterstützt, jedoch mit der Einschränkung, dass die Beziehungen zwischen den Attributen nicht eingesehen werden können.
Die *unterschiedliche Aktivitätengranularität* H7 wird ebenfalls nicht unterstützt, da verschiedene Ausführungsmöglichkeiten, wie die instanzspezifische, kontextsensitive, oder auch die Batch-Aktivitäten in einem imperativen System nicht unterstützt werden. Die Aktivität wird zur Modellierzeit mit einer Geschäftsfunktion verknüpft und hat deshalb eine unveränderbare Granularität.
Die Ausführung von *optionalen Aktivitäten* H8, und damit verbunden der Zugriff auf Daten zu jeder Zeit des Prozessablaufs, wird in einem imperativen PrMS nicht unterstützt.
Der *Kontrollfluss innerhalb eines Benutzerformulars* H9 wird ebenfalls nicht unterstützt.

Prozesse:

Durch eine feinkörnige Modellierungsart ist gewährleistet, dass miteinander verknüpfte Prozessinstanzen eine bestimmte Objektinstanz zur Laufzeit bearbeiten. Aus diesem Grund wird dem Prozessmodell entweder ein Datenelement zur Steuerung der Objekt-ID hinzugefügt, oder mehrere Datenelemente, welche zu den Objektattributen gehören. Der imperative Ansatz besteht aus einem aktivitätenzentrierten Ansatz, der jedoch das *Objektverhalten* H10 nur teilweise unterstützt. Prozessinstanzen werden unabhängig voneinander ausgeführt. Deshalb können keine Abhängigkeiten modelliert werden. Das Problem kann nur durch den Einsatz von untergeordneten Prozessen modelliert werden. Jedoch ist dieser Subprozess sehr eng mit der Synchronisation des darüberliegenden Prozesses verknüpft. Somit ist dieser solange blockiert, bis der Subprozess beendet wird. Durch eine fehlende weitreichende Synchronisationsverwaltung werden *Objektinteraktionen* H11 nur teilweise unterstützt.

Von Benutzern ausgeführte Aktivitäten werden mit Benutzerrollen verknüpft. So werden zur Laufzeit eines Prozesses die Aktivitäten nur von berechtigten Benutzern ausgeführt. Dafür werden auch im imperativen PrMS die Aktivitäten, die bereit zur Ausführung sind, über Arbeitslisten (worklists) verteilt. Dadurch können die Aktivitäten von den richtigen Benutzern zur richtigen Zeit ausgeführt werden. Dies nennt man *prozessorientierte Sicht* H12, welche bei diesem Ansatz auch unterstützt wird.

In einem imperativen System ist die Aktivierung einer Aktivität abhängig vom Beenden der vorhergehenden Aktivitäten. Dies macht eine *flexible Prozessausführung* H13 unmöglich.

Es besteht keine Möglichkeit, eine Aktivität zu überspringen, wenn beispielsweise die benötigten Attributwerte schon vorhanden sind. Solange eine Aktivität nicht von einem Benutzer beendet wurde, ist keine *wiederholte Ausführung der Aktivität* H14 möglich.

Dadurch, dass die Aktivierung einer Aktivität von der Fertigstellung anderer Aktivitäten abhängt, gibt es keine Unterstützung in einem imperativen PrMS, die es ermöglicht, durch *Benutzerentscheidungen* H15 einen anderen Weg zum Prozessziel einzuschlagen.

Benutzerintegration:

Die *Datenautorisierung* H16 wird nicht ausreichend unterstützt, da die Daten von Be-

nutzern nur bearbeitet werden können, wenn eine Aktivität aktiv ausgeführt wird. Die Unterstützung für eine erneute Ausführung der Daten oder bereits ausgeführte Prozesse ist nicht vorgesehen. So wird auch keine *Differenzierung zwischen Autorisierung und Bearbeiterzuordnung* H18 berücksichtigt.
Die *Prozessautorisierung* H17 wird nur auf Aktivitätenebene unterstützt. Es kann bei einem imperativen PrMS nicht sichergestellt werden, dass die Berechtigung für das Lesen und Schreiben einer Aktivität gleichzeitig die Berechtigung für das Lesen und Schreiben von Attributen beinhaltet, die von einer Aktivität ausgeführt werden. Es ist nicht möglich, unterschiedliche Rechte abhängig vom Status der auszuführenden Prozessinstanz zu vergeben. Damit wird die *vertikale Autorisierung und Bearbeiterzuordnung* H19 nicht unterstützt.

Monitoring:
Eine *gesamtheitliche Sicht* H20 der Überwachung ist nicht möglich, aufgrund der Trennung der Aktivitäten in Objektinstanzen und Attribute.

Die Auswertung zeigt, dass der imperative Ansatz nur sehr wenige der Herausforderungen erfüllt, die für ein objektzentriertes Prozessmanagement aufgestellt wurden.
Aufgrund der Grenzen der Ausführbarkeit, wurde mit dem deklarativen Ansatz ein anderer Weg zur Beschreibung von Geschäftsprozessen eingeführt. Diese werden hierbei durch Regeln, und nicht durch einen Kontrollfluss begrenzt. Die Grundlagen zu dem deklarativen Ansatz wurden bereits in Kapitel 2.3.2 beschrieben.

3.2.2 Deklarativer Ansatz

Die wesentliche Unterscheidung zum imperativen Ansatz liegt in erster Linie in der regelbasierten Steuerung der Aktivitäten, anstelle der kontrollflussbasierten Steuerung. Aus diesem Grund entsprechen die meisten Auswertungen dem imperativen Ansatz. Deshalb werden hier nicht nochmals alle Herausforderungen erwähnt, sondern nur einige Punkte des deklarativen Ansatzes überprüft. [44]

Die Regeln können grob in drei Klassen unterteilt werden. [25] Es gibt Regeln, welche die Ausführung einer Aktivität verbieten. Außerdem existieren Regeln, die eine optionale

Ausführung unter bestimmten Bedingungen erlauben. Alle anderen Aktivitäten dürfen frei ausgeführt werden. Dadurch werden Bedingungen erstellt, die ein unerwünschtes Ausführungsverhalten verbieten.

Deklarative Ansätze unterstützen keine mehrfach Instanziierung, so dass *Kardinalitäten* H3 zu übergeordneten Prozessen nicht unterstützt werden. *Black-Box-Aktivitäten* H6 werden unterstützt, jedoch wie beim imperativen Ansatz mit der Einschränkung, dass Beziehungen zwischen den Attributen nicht eingesehen werden können. Durch die Regeln ist es bei einem deklarativen Ansatz möglich, *optionale Aktivitäten* H8 auszuführen, solange die Regeln die Ausführung nicht erzwingen.

Wie auch beim imperativen Ansatz können *Objektverhalten* H10 und *Objektinteraktionen* H11 nur teilweise unterstützt werden. Auch hier findet eine Integration der Prozessinstanzen über Subprozesse statt.
Die *Prozessorientierte Sicht* H12 wird auch vom deklarativen Ansatz unterstützt, da es obligatorische Aktivitäten gibt, die ausgeführt werden müssen, um das Prozessziel zu erreichen.
Doch auch der deklarative Ansatz lässt noch viele Anforderungen zur Steigerung der Flexibiltät unberücksichtigt. Zum Beispiel ist es nicht möglich, die Ausführung von Aktivitäten zu überspringen, auch wenn die Attributwerte schon vorhanden sind. Es wird also keine *flexible Prozessausführung* H13 unterstützt. Auch wenn dem Benutzer durch den regelbasierten Ansatz mehrere Alternativen zur Prozessausführung zur Verfügung stehen, so ist eine *wiederholte Ausführung von Aktivitäten* H14 dennoch nicht möglich.

Der deklarative Ansatz unterstützt ebenfalls nur sehr eingeschränkt die Anforderungen, die an ein objektorientiertes PrMS gestellt werden. Im Gegensatz zum imperativen Ansatz bringt der regelbasierte Ansatz jedoch eine Unterstützung von optionalen Aktivitäten mit.

3.2.3 Datenzentrierte Ansätze

Im folgenden Abschnitt werden zwei Forschungsansätze gegen die Herausforderungen eines objektorientierten PrMS verglichen, die bereits eine datenzentrierte Sicht für

einen Prozessablauf integrieren. Dafür wird eine artefaktzentrierte und eine fallbasierte Sichtweise untersucht.

3.2.3.1 Artefaktzentrierter Ansatz

Der bereits in Kapitel 2.4.1 erläuterte artefaktzentrierte Ansatz stellt durch seine Ausrichtung auf Datenobjekte, sogenannte Artefakte, einen datenzentrierten Ansatz für Prozessmanagement dar. Welche Aspekte eines objektzentrierten Prozessmanagements unterstützt werden, wird in diesem Abschnitt erläutert. [43, 47]

Daten:
Artefakte besitzen, wie Objekte, verschiedene Attributwerte und stehen mit anderen Artefakten in Beziehung. Die *integrierte Datenstruktur* H1 wird unterstützt. Jedoch können die Attribute und Beziehungen mehrmals in verschiedenen Artefakten auftreten, da die Artefakte unabhängig voneinander betrachtet werden. Die entstehende Datenstruktur muss deshalb mehrfach über die verschiedenen Datenmodelle verteilt werden und wird dadurch schwer verständlich und kompliziert zu bearbeiten. Die *Kardinalität* H3 ist möglich, da die Anzahl der Beziehungen beschränkt werden kann. [34] Für jede Transition eines Artefakt-Lebenszyklus wird eine Bedingung festgelegt, welche aus *obligatorischen Informationen* H4 besteht. Damit werden diese unterstützt.

Aktivitäten:
Formularbasierende Aktivitäten H5 werden nicht unterstützt, da diese nicht automatisiert während der Laufzeit angepasst werden können. Services werden über ECA-Regeln mit einem Artefakt verknüpft und innerhalb einer *Black-Box-Aktivität* H6 definiert. Deshalb wird der *Kontrollfluss innerhalb eines Benutzerformulars* H9 nicht unterstützt. Denn es ist weder möglich, die Reihenfolge zu bestimmen, in der die Objektattribute geschrieben werden, noch ist es möglich, gültige Attributwerte zu definieren.

Jeder Service wird separat implementiert, so dass die Granularität der Aktivitäten zur Modellierzeit bereits feststeht. Eine *unterschiedliche Aktivitätengranularität* H7 zur Laufzeit ist somit nicht möglich. In der Arbeitsliste eines Benutzers findet keine Unterscheidung

von *optionalen oder obligatorischen Aktivitäten* H8 statt, so dass diese nicht unterstützt werden. Weiterhin werden auch keine *Benutzerentscheidungen* berücksichtigt.

Prozesse:
Ein Artefakt besteht aus einem Informationsmodell und einem Lebenszyklusmodell. Dadurch wird das *Objektverhalten* H10 unterstützt, denn das Lebenszyklusmodell beschreibt genau das Verhalten eines Artefakts. Während der Ausführung eines Prozesses werden die Artefakte bearbeitet und wechseln zu verschiedenen Stages eines Lebenszyklus. Wird eine bestimmte Aktivität durchgeführt, ist es nicht klar ersichtlich, welche Artefakte tatsächlich ausgeführt werden. Jeder Service bearbeitet ein oder mehrere Artefakte. ECA-Regeln werden zur Steuerung des Ablaufs eines Artefakt-Lebenszyklus benutzt. Dazu werden Quantifikatoren eingesetzt. Es besteht keine klare Trennung zwischen dem Objektverhalten und der *Objektinteraktion* H11, so dass die Interaktion nur teilweise in einem objektzentrierten Prozess unterstützt wird. Weiterhin werden keine transitiven oder quergerichteten Beziehungen zwischen Artefakten unterstützt. ECA-Regeln definieren hauptsächlich Vorbedingungen, so dass es nicht möglich ist, während der Laufzeit einen Service zu überspringen. Damit ist eine *Flexible Prozessausführung* H13 ebenfalls nicht möglich.

Weder die **Benutzerintegration** noch die **Überwachung** werden unterstützt. Es wird wie beim klassischen PrMS eine rollenbasierte Rechteverteilung verwendet, um den Zugriff einzuschränken. Weiterhin bietet der artefaktzentrierte Ansatz keine gesamtheitliche Sicht auf den Prozess.

Der artefaktzentrierte Ansatz bietet eine weitreichende objektorientierte Sichtweise im Bereich der Daten. Jedoch werden die meisten Herausforderungen der anderen Bereiche nicht unterstützt, so dass nur von einer sehr eingeschränkten objektorientierten Unterstützung dieses Ansatzes gesprochen werden kann.

3.2.3.2 Fallzentrierter Ansatz

Einen vielversprechenden Ansatz stellt der fallzentrierte Prozessmanagementansatz dar. Er erfüllt die meisten Anforderungen gegenüber objektzentrierten Prozessen, verglichen

mit den bereits vorgestellten Ansätzen. Durch die Integration von Daten und Prozessen ist die Ausführung eines Falls über die Veränderung der Datenelemente möglich. Dadurch entsteht mehr Flexibilität, die in einem klassischen PrMS, durch die strikte Trennung der Prozesse von den Daten, nur sehr eingeschränkt umsetzbar ist. Somit entsteht für den Benutzer die Möglichkeit, die Ausführungsreihenfolge seiner Arbeit individuell fest zu legen. Aktivitäten können flexibler ausgeführt werden, ohne den gesamten Prozess zu blockieren. Die Ausführung wird weiterhin nicht mehr an ein vordefiniertes Prozessmodell gebunden. Außerdem ist es möglich, mehrere verknüpfte Aktivitäten auf einmal auszuführen. Die Darstellung eines CH-Systems erfolgt über Benutzerformulare, welche eine Reihe von Eingabefeldern enthalten
Anhand der vier Hauptgeschäftsperspektiven werden im Folgenden alle Herausforderungen für ein objektzentriertes Prozessmanagment mit den Eigenschaften von CH verglichen. [44, 48, 47]

Daten:
Ein Fall kann größtenteils als ein Objekt angesehen werden. Datenelemente, die mit einem Fall verknüpft sind, können als Objektattribute betrachtet werden. Ein Fall wird hierarchisch mit untergeordneten Fällen verbunden, die mit Objektbeziehungen verglichen werden können. Jedoch sind nur direkte Beziehungen möglich. Relationen zwischen zwei Instanzen desselben Falls werden nicht unterstützt, das heißt, dass keine quergerichteten oder transitiven Beziehungen zugelassen sind. Somit wird die *Datenintergration* H1 teilweise unterstützt. In CH liegt ein Schwerpunkt darin, dass der ganze Fall betrachtet wird. Alle Benutzer erhalten lesende Rechte für den gesamten Fall, wenn diese eine Aktivität ausführen. Es gibt keine eingeschränkte Sicht für einzelne Benutzer, wodurch eine erhöhte Flexibiltät erreicht werden soll. So erhält der Benutzer zu jeder Zeit *Datenzugriff* H2 auf alle benötigten Daten.

Die *Kardinalität* H3 wird in CH nicht unterstützt, da zur Laufzeit nicht überprüft werden kann, ob die richtige Anzahl an untergeordneten Prozessen erstellt wurde. Dadurch, dass einige Datenelemte als obligatorisch definiert werden, werden auch *obligatorische Informationen* H4 unterstützt.

Aktivitäten:
Formularbasierende Aktivitäten sind mit einer Anzahl von obligatorischen oder beschränkten atomaren Datenelementen verbunden. Eine Aktivität wird als beendet definiert, wenn alle obligatorischen Datenelemente beschrieben wurden. Jedoch können die Formularfelder nicht auf den beteiligten Benutzer angepasst werden. Damit werden *Formularbasierende Aktivitäten* H5 nur teilweise unterstützt. *Black-Box-Aktivitäten* H6 werden unterstützt, jedoch nur teilweise, da die Beziehungen zwischen verschiedenen Objektinstanzen nicht überprüft werden können. Die *unterschiedliche Aktivitätengranularität* H7 wird von CH nicht unterstützt, da eine Aufgabeninstanz mit genau einer Prozessinstanz verbunden ist. Somit werden nur instanzbestimmte Aktivitäten umgesetzt. Jedoch beinhaltet CH weder kontextbestimmte Aktivitäten noch Batch-Aktivitäten, da das Zusammenfassen verschiedener Instanzen desselben Falltyps nicht unterstützt wird. Eine Aufgabe in CH kann Datenelemente enthalten, die obligatorisch ausgeführt werden müssen, bevor der Fall beendet wird. Das Bearbeiten eines obligatorischen Datenelements in einem Fall wird als obligatorische Aktivität betrachtet. Weiterhin gibt es in einem Fall freie Datenelemente. Diese können optional von jedem User eines Falls zu jedem beliebigen Zeitpunkt ausgeführt werden, solange der Fall nicht beendet ist. Damit werden *obligatorische und optionale Aktivitäten* H8 unterstützt. Ein Formularfeld kann nicht dynamisch als obligatorisch gekennzeichnet werden und auch die Abhängigkeiten können nicht beschrieben werden. Damit wird das Ziel eines *Kontrollflusses innerhalb eines Benutzerformulars* H9 nicht erreicht.

Prozesse:
CH unterstützt keine komplexen Objekte und Relationen zwischen Objekten. Dennoch hat ein Fall, wie bereits erwähnt, viele Gemeinsamkeiten mit einem Objekt, so dass sie in weiten Teilen gleichgestellt werden können. Somit wird *Objektverhalten* H10 weitestgehend unterstützt, da gesteuert werden kann, welcher Benutzer welches Attribut zu welchem Zeitpunkt und in welcher Reihenfolge bearbeitet.
Die *Objektinteraktion* H11 wird nicht unterstützt, da es in CH nicht möglich ist, eine Interaktion der verschiedenen Objektinstanzen festzulegen. Ein Benutzer kann einer vom Benutzer zu bearbeitenden Aktivität zugeteilt werden, wenn er Schreibrechte für die Aktivität besitzt. Der Benutzer wählt sich alle zu bearbeitenden Aktivitäten aus den

Fällen aus und generiert so eine *prozessorientierte Sicht* H12 für sich.
Eine Aufgabe wird aktiviert, sobald die Daten als verfügbar gekennzeichnet werden. Sind in einer bestimmten Aktivität alle dazugehörigen obligatorischen Datenelemente bearbeitet worden, kann die nachfolgende Aktivität als verfügbar markiert und ebenfalls bearbeitet werden. Es findet eine *flexible Prozessausführung* H13 statt, da es möglich ist, eine Aktivität zu überspringen, falls bereits alle obligatorischen Datenelemente durch vorhergehende Aufgaben bearbeitet wurden. Die *wiederholte Ausführung von Aktivitäten* H14 ist im CH-Ansatz möglich.
Über Rollen kann auch festgelegt werden, welcher Benutzer eine Aktivität nochmals ausführen darf. Diese Rolle erlaubt dem Benutzer eine Aktivität so lange mehrmals auszuführen, bis der Fall beendet ist. Danach sind keine Änderungen an den Datenelemente mehr möglich. Jedoch kann der Wert eines Datenelements während der Bearbeitung eines Falls nicht fixiert werden, um eine Aufgabe als beendet zu markieren und eine weitere Bearbeitung dieses Elements zu verhindern. Weiterhin werden auch *Benutzerentscheidungen* H15 unterstützt.

Benutzerintegration:
Die *Datenautorisierung* H16 wird in CH nicht unterstützt. Das liegt an mehreren Punkten. Zum einen können, wie gerade beschrieben, Werte nicht fixiert werden. Daher kann es zu Inkonsistenzen kommen. Des Weiteren ist das Prinzip von CH, dass jeder Benutzer eines Falles alle relevanten Daten lesen darf. Dies widerspricht der Idee eines objektzentrierten Ansatzes, in der ein Bearbeiter nur so viele Rechte erhält, die zur Bearbeitung eines Prozesses benötigt werden. Es gibt zwar eine Rollenunterteilung, die es ermöglicht, zu unterscheiden, wer ein Datenelement schreiben darf. Jedoch ist die Rolle fest mit einer Aufgabe verknüpft, so dass diese Aufgabe anderen Benutzern nicht zur Verfügung gestellt werden kann. Weiterhin kann die Rolle nicht abhängig vom Fortschritt des Prozesses vergeben werden. Sind die Datenelemente während der Ausführung einer Aktivität bekannt, kann sichergestellt werden, dass alle Benutzer mit einer execution role auch die dazugehörigen Datenrechte auf eine Aktivität besitzt. Damit ist eine *Prozessautorisierung* H17 teilweise möglich. Besitzt ein Benutzer eine execution role für eine Aktivität, muss er eine obligatorische Aktivität zwingend ausführen. Somit ist eine *Differenzierung zwischen Autorisierung und Bearbeiterzuordnung* H18 nicht möglich

und auch eine *vertikale Autorisierung und Bearbeiterzuordnung* H19 wird in CH nicht unterstützt.

Überwachung:
Es werden weder Objekttypen noch Objektinstanzen und deren Beziehungen betrachtet. So ist eine *gesamtheitliche Sicht* H20 nicht möglich.

Der Case Handling-Ansatz integriert, bis auf die Objektinteraktionen, alle Eigenschaften eines objektorientierten PrMS im Prozessbereich. Weiterhin werden auch optionale Aktivitäten und Informationen unterstützt. Ebenfalls sind Ansätze bei der Umsetzung einer Prozessautorisierung vorhanden, die für ein objektorientiertes System notwendig sind. Diese Herausforderungen stellen die Grundlage für die Entwicklung des PHILharmonicFlows Frameworks, welches eine vollständige Umsetzung aller Eigenschaften implementiert und somit ein objektorientiertes System darstellt.

3.2.4 PHILharmonicFlow

Die Grundlagen zu PHILharmonicFlows wurden bereits in Kapitel 2.5.1 angesprochen. Welche Komponenten des Frameworks für die Umsetzung der Eigenschaften einer ganzheitlichen objektorientierten Sicht verantwortlich sind, wird im folgenden Abschnitt aufgezeigt.

Daten:
Mit Hilfe des Datenmodells wird die *Datenintegration* H1 unterstützt. Der *Datenzugriff* H2 ist zu jedem Zeitpunkt möglich. Weiterhin sind minimale und maximale *Kardinalitätsbeschränkungen* H3 möglich, so dass zur Laufzeit eine dynamische Anzahl an Objektinstanzen berücksichtigt wird. Mit Hilfe des Makroprozesses und der Kardinalitäten werden notwendige Objektinstanzen definiert. Der Mikroprozess definiert notwendige Attributwerte. Damit ist es möglich, *obligatorische Informationen* H4 zu definieren.

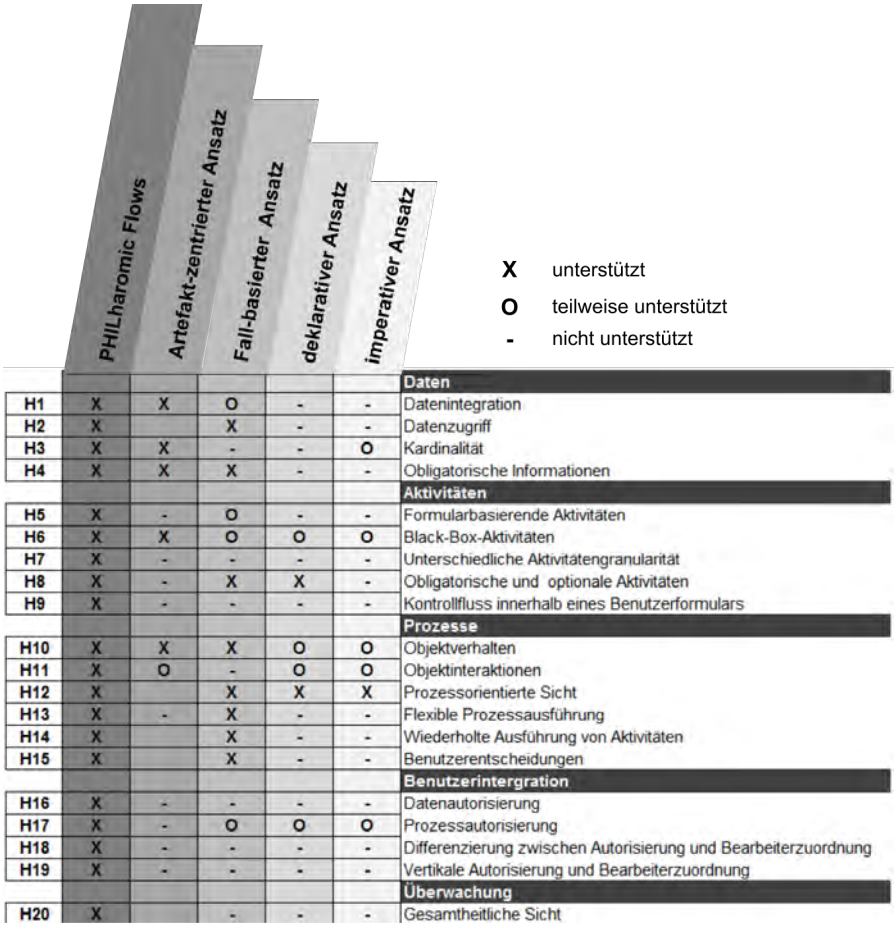
Aktivitäten:
Formularbasierende Aktivitäten H5 werden zur Laufzeit automatisch anhand der Attribute und Berechtigungen generiert. Auch der Status des Prozesses wird dabei berücksichtigt. Externe Services können mit Hilfe von *Black-Box-Aktivitäten* H6 in das Framework

eingebunden werden. Zwischen *obligatorischen und optionalen Aktivitäten* H8 wird anhand der Rechtevergabe unterschieden. Weiterhin ist durch die automatische Erstellung von formbasierenden Aktivitäten auch eine Bearbeitung von mehreren Objektinstanzen *unterschiedliche Aktivitätengranularität* H7 implementiert. Über Mikroschritte und interne Mikrotransitionen wird ein *Kontrollfluss innerhalb eines Benutzerformulars* H9 modelliert.

Prozess:
Das Prozessmodell in PHILharmonicFlows unterteilt sich in einen Mikroprozess, welcher das *Objektverhalten* H10 darstellt, und einen *Makroprozess* H11, welcher die Objektinteraktionen definiert. Neben der datenorientierten Sicht wird zur Laufzeit auch eine *prozessorientierte Sicht* H12 unterstützt, welche automatisiert Arbeitslisten erstellt und die Überwachung verwaltet. Durch die datengesteuerte Ausführung ist eine *flexible Prozessausführung* H13 möglich. In Verbindung mit expliziten Mikrotransitionen wird auch eine *wiederholte Ausführung der Aktivitäten* H14 realisiert.

Benutzerintegration:
Der Benutzer kann über die Anzahl der Objektinstanzen frei entscheiden, solange die Kardinalitätsgrenzen berücksichtigt werden. *Benutzerentscheidungen* H16 sind somit möglich. Über Rechtetabellen können obligatorische Schreibvorgänge auf Attribute definiert werden. Weiterhin wird eine Mikrotransition mit einer Benutzerrolle verknüpft, um Benutzerrechte für eine Aktivität zu erteilen. Damit wird eine *Prozessautorisierung* H17 unterstützt. Daneben gibt es eine Unterscheidung zwischen einer optionalen und obligatorischen Rechtervergabe. Somit wird auch die *Unterscheidung zwischen Autorisierung und Bearbeiterzuordnung* H18 in PHILharmonicFlows umgesetzt. Das System erlaubt die Rechtebegrenzung einer Objektinstanz eines betimmten Objekttyps und integriert damit eine *vertikale Autorisierung und Benutzerzuordnung* H19.

Überwachung:
Zur Laufzeit unterstützt das Framework eine *gesamtheitliche Überwachung* H20 aller Objektinstanzen und deren Abhängigkeiten.



	PHILharomic Flows	Artefakt-zentrierter Ansatz	Fall-basierter Ansatz	deklarativer Ansatz	imperativer Ansatz	
						Daten
H1	X	X	O	-	-	Datenintegration
H2	X		X	-	-	Datenzugriff
H3	X	X	-	-	O	Kardinalität
H4	X	X	X	-	-	Obligatorische Informationen
						Aktivitäten
H5	X	-	O	-	-	Formularbasierende Aktivitäten
H6	X	X	O	O	O	Black-Box-Aktivitäten
H7	X	-	-	-	-	Unterschiedliche Aktivitätengranularität
H8	X	-	X	X	-	Obligatorische und optionale Aktivitäten
H9	X	-	-	-	-	Kontrollfluss innerhalb eines Benutzerformulars
						Prozesse
H10	X	X	X	O	O	Objektverhalten
H11	X	O	-	O	O	Objektinteraktionen
H12	X		X	X	X	Prozessorientierte Sicht
H13	X	-	X	-	-	Flexible Prozessausführung
H14	X		X	-	-	Wiederholte Ausführung von Aktivitäten
H15	X		X	-	-	Benutzerentscheidungen
						Benutzerintergration
H16	X	-	-	-	-	Datenautorisierung
H17	X	-	O	O	O	Prozessautorisierung
H18	X	-	-	-	-	Differenzierung zwischen Autorisierung und Bearbeiterzuordnung
H19	X	-	-	-	-	Vertikale Autorisierung und Bearbeiterzuordnung
						Überwachung
H20	X		-	-	-	Gesamtheitliche Sicht

X unterstützt
O teilweise unterstützt
- nicht unterstützt

Abbildung 3.3: Zusammenfassung aller vorgestellten Ansätze, vergleichbar [44, 48, 40]

Mit dem PHILharmonicFlows wurde ein objektorientiertes Framework entwickelt, welches alle im Vorfeld definierten Herausforderungen einbezieht und damit ein gesamtheitliches datenzentriertes System umsetzt.

3.3 Zusammenfassung

In diesem Kapitel wurden mehrere Konzepte gegen die Anforderungen eines objektorientierten PrMS evaluiert. Dabei können diese Ansätze als Grundlage für die Entwicklung des PHILharmonicFlows Frameworks angesehen werden. Das Kapitel zeigt auf, welche Anforderungen entwickelt werden mussten, um in einem System eine hohe Flexibilität und eine gesamtheitliche, datenzentrierte Sicht umzusetzen . Dabei stand die Integration der Daten, Prozesse und Funktionen im Vordergrund. Weiterhin wurde eine enge Integration der Benutzer implementiert.
PHILharmonicFlows unterstützt dadurch auch wissensintensive Prozesse, die eine weitreichende Flexibilität voraussetzen und durch Wissensarbeiter ausgeführt werden. Das Framework kann zur Laufzeit auf unerwartete Ereignisse reagieren und ermöglicht dem Bearbeiter auch außerhalb des Prozessablaufs Datenveränderungen vorzunehmen. Im Gegensatz zu fallorientierten Prozessen können auch sehr komplexe Prozesse mit PHILharmonicFlows umgesetzt werden.

Im nächsten Kapitel wird untersucht, inwieweit Modelliersprachen eine datenzentrierte Sicht integrieren und ein System bei der Modellierung unterstützen können. Dabei wird die fallbasierte Beschreibungssprache CMMN mit der prozessorientierten Beschreibungssprache BPMN verglichen.

4

Prozess- und datenorientierte Notationen

Zur grafischen Modellierung und Darstellung eines Geschäftsprozesses werden in einem PrMS häufig standardisierte Modelliersprachen eingesetzt. Mit Hilfe dieser Notationen kann Prozesswissen in einem Modell schematisch und verständlich visualisiert werden und stellt somit eine Schnittstelle zwischen der betriebswirtschaftlichen Anforderung und der Umsetzung der Prozesse dar. [49] Diese Art der Darstellung von Prozessmodellen unterstützt auch die Neugestaltung und Einführung von Arbeitsabläufen in einem Unternehmen und besitzt eine hohe Übersichtlichkeit, die auch zur Dokumentation von Geschäftsprozessen verwendet werden kann. So können komplexe Prozesse mit Hilfe eines festgelegten Metamodells für verschiedene Anwendergruppen veranschaulicht werden. Darüber hinaus steigt durch den Einsatz einer einheitlichen Modelliersprache die Kontrolle über sämtliche Prozesse im Unternehmen, wodurch notwendige Anpassungen schneller erkannt und umgesetzt werden können. Einige PrMS, wie beispielsweise die Oracle BPM Suite, bieten dafür eine grafische Oberfläche an, über welche Änderungen

am Modell mit einer *Drag and Drop*-Funktionalität umgesetzt werden können. [4]
Dieses Kapitel zeigt zunächst die Voraussetzungen, die nötig sind, um einen Prozess modellieren zu können und wie flexibel diese Modelle gestaltet werden können. Danach werden zunächst die viel verwendete Modelliersprache BPMN und die neu veröffentlichte Notation CMMN vorgestellt, bevor diese im Anschluss daran miteinander verglichen werden. Dabei wird insbesondere herausgearbeitet, inwiefern die Flexibilität umgesetzt wird und wie diese Sprachen in einem daten-orientierten Prozessmanagementsystem eingesetzt werden.

4.1 Strukturierungsgrad

Als Voraussetzung, um einen Prozess mit Hilfe einer Modellierungssprache darstellen zu können, muss ein bestimmter Strukturierungsgrad vorhanden sein. [50] Zunächst gibt es nicht strukturierbare Arbeitsabläufe. Diese führen dazu, dass Benutzer eine vollständige Entscheidungsfreiheit darüber besitzen, wie ein Prozessablauf auszusehen hat. Jedoch lässt sich solch ein Prozess nicht modellieren, da alle Entscheidungen spontan der Situation angepasst werden, so dass keine Struktur erkennbar ist.
Daneben gibt es strukturschwache und strukturstarke Arbeitsabläufe. Diese lassen sich modellieren, da sich Arbeitsschritte ganz oder teilweise im Voraus definieren lassen.
Vollständig strukturierbare und damit strukturstarke Prozessabläufe bilden überwiegend lineare Prozesse ab und können wegen ihrer strikten Abfolge zur Entwurfszeit vormodelliert werden. Dies wird, wie bei den prozessorientierten Prozessen bereits beschrieben (siehe 2.1), durch einen Kontrollfluss dargestellt und gestaltet sich bei der Ausführung als sehr unflexibel. Daher besitzen Benutzer keinen Freiheitsgrad bei der Bearbeitung solcher Prozesse. Ein System, das eine klassische Geschäftsprozessmodellierung unterstützt, ist auf Prozesse beschränkt, die ablaufgesteuert sind und explizit angegeben werden können. [51] Dabei besteht ein sehr hoher Anteil aus sich wiederholenden Abläufen. Zur Modellierung von Prozessen in solchen Bereichen eignen sich imperative Prozessbeschreibungssprachen, zu denen auch BPMN gehört.

Benötigen Benutzer mehr Flexibilität bei der Bearbeitung eines Prozesses oder wiederholen sich Elemente eines Arbeitsablaufes sehr wenig, spricht man von strukturschwachen Prozessabläufen. Dabei werden Arbeitsschritte nicht mehr vollständig im Vorfeld modelliert, da sich Teile des Prozesses erst zur Laufzeit entwickeln. Diese Prozesse treten oft in fallbasierten und wissensintensiven Arbeitsfeldern auf und sind häufig nicht mehr aktivitätengesteuert, sondern stellen die Daten in den Mittelpunkt. Daher benötigen solche Prozess auch andere Herangehensweisen, um übersichtlich und verständlich modelliert zu werden. Um auf diese Entwicklung einzugehen, hat die OMG mit dem CMMN-Standard eine deklarative Sprache veröffentlicht, welche versucht die Veränderungen und Anforderungen in vielen Geschäftsbereichen aufzugreifen, in denen mehr Freiheit bei der Bearbeitung von Aufgaben benötigt wird. So erhält der Benutzer mit CMMN die Möglichkeit, Prozessabläufe zur Laufzeit individuell an seine Bedürfnisse anzupassen. Die Notwendigkeit für mehr Flexibilität und damit verbundenen strukturschwachen Arbeitsabläufen ist kein neues Phänomen. Mit der Einführung von automatisierten Arbeitsabläufen hat man sehr schnell erkannt, welche Hürden bei der Umsetzung entstehen, wenn man Arbeitsabläufe vollständig vorgibt. Somit existieren viele verschiedene Prozessbeschreibungssprachen, welche alle Vor- und Nachteile besitzen und bestimmte Arbeitsabläufe nur unzureichend oder kompliziert ausdrücken können. Die Effizienz der Modellierung leidet, wenn eine Sprache nicht auf unterschiedliche Anforderungen eingehen kann. [52] Daraus können Nachteile bei der Umsetzung von BPM entstehen, indem beispielsweise mehrere verschiedene PrMS innerhalb eines Unternehmens zum Einsatz kommen. Ein weiterer Nachteil ist, dass nicht modellierbare Lösungen am PrMS vorbei umgesetzt werden müssen. In beiden Fällen wird der gesamtheitliche Geschäftsprozess nicht mehr einheitlich ausgeführt und überwacht. Die bereits genannten Vorteile, welche durch eine Einführung von BPM im Unternehmen entstehen, gerade auch im Bereich der Kostenreduktion und der Effizienzsteigerung, werden damit geschmälert oder gehen ganz verloren. Dies zeigt, wie wichtig es ist, vor der Einführung eines PrMS anhand des Einsatzzwecks, eine passende Modelliersprache auszuwählen. In den nächsten Abschnitten werden mit BPMN und CMMN eine imperative und eine deklarative Sprache untersucht, um die verschiedenen Charakteristika der Sprachen und die Gegensätze zwischen schwachstrukturierten und starkstrukturierten Prozessen

aufzuzeigen. Die unterschiedlichen Stukturierungsgrade der beiden Sprachen stehen dabei für die Grenzen, innerhalb derer Prozesse modelliert werden können (siehe Abb. 4.1). Für die meisten Prozesse dürfte der benötigte Grad an Flexibilität irgendwo dazwischen liegen. [53] So können starkstrukturierte Prozesse beispielsweise flexibler gestaltet werden, indem Ausnahmebehandlungen zur Laufzeit ermöglicht werden oder schwachstrukturierte Prozesse durch vordefinierte Elemente mehr Struktur erhalten.

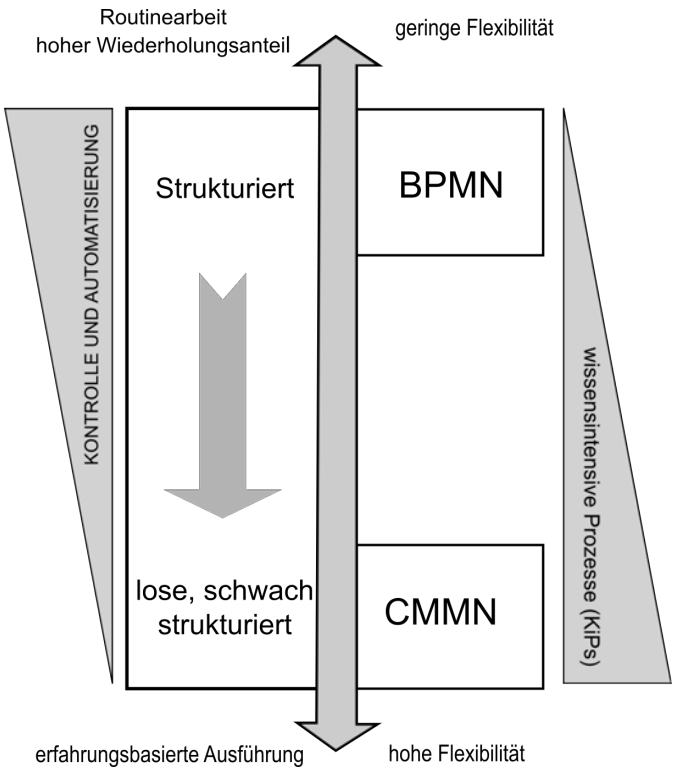


Abbildung 4.1: Strukturierungsgrad im Prozessmanagement, vergleichbar [9]

4.2 Flexibilität

Durch die Abbildung 4.1 wird verdeutlicht, wie mit einem abnehmenden Strukturierungsgrad eines Prozesses die wissensintensiven Prozesse zunehmen. Dadurch steigt die Flexibilität innerhalb eines Prozesses auch für den Benutzer. Wird im Zuge von KiPs von Flexibilität gesprochen, wird für den Wissensarbeiter mehr Freiraum bei der Bearbeitung

von Prozessen gefordert. Es handelt sich damit um mehr Flexibilität zur Laufzeit eines Prozesses. Dabei unterscheidet man zwei Arten von Flexibilität. So ist es ebenfalls möglich, Prozesse flexibler zu gestalten, indem Änderungen zur Modellierzeit vorgenommen werden. [54]
Bei der Flexibiltät zur Modellierzeit sollen Prozesse, durch leicht umsetzbare und verständliche Änderungsmöglichkeiten, zur Entwurfszeit rasch angepasst werden. Dabei setzt sich der Zeitraum bis zur Umsetzung einer solchen Prozessänderung nicht nur aus der Modellierung zusammen. Der aufwändigere Teil besteht darin, geänderte Modelle dahin gehend zu überprüfen, ob der Prozessablauf technisch stabil und robust ausgeführt werden kann. Mit kurzen Intervallen bei der Erneuerung eines Prozessmodells steigt die Flexibilität eines Systems und durch eine regelmäßige Anpassung und Verbesserung der Prozessabläufe wird sichergestellt, dass veränderte Abläufe innerhalb des PrMS umgesetzt werden und somit die Kontrolle des Systems nicht umgangen wird.
Das gilt auch für die zweite Art der Anpassungsmöglichkeit. Bei der Flexibilität zur Ausführungszeit ist es ebenfalls wichtig, Möglichkeiten zur Verfügung zu stellen, damit Änderungen zur Laufzeit innerhalb einer Prozessinstanz und damit unter der Kontrolle des PrMS vorgenommen werden können. So kann der Ablauf individuell angepasst werden, indem einzelne Bearbeitungsschritte übersprungen, miteinander vertauscht oder zu einem anderen Zeitpunkt ausgeführt werden. Während der Ausführung ist die Veränderung eines Prozesses mit strikter Abfolge im Rahmen einer Ausnahmebehandlung allerdings nur von Prozessexperten durchführbar. Denn durch eine falsche Anpassung der Prozessinstanz kann schnell die Robustheit der weiteren Ausführung gefährdet werden. Die Notwendigkeit einer Prozessanpassung besteht jedoch in jedem Unternehmen, so dass ein Prozessmodell regelmäßig angepasst werden muss. Wird in einem PrMS ein neues Prozessmodell eingeführt, wird eine Prozessschemaevolution eingesetzt. Diese besitzt die Fähigkeit, laufende Prozessinstanzen ohne Fehler in das neue Prozessmodell einzubinden. Somit ist es nicht notwendig, jede Prozessinstanz einzeln mit Hilfe einer Ausnahmebehandlung anzupassen.

Die beiden Arten der Flexibilität wurden für strukturstarke und prozessorientierte Prozesse definiert. Diese lassen sich gleichwohl auch auf fallbasierte und damit struktur-

schwache Prozesse übertragen. So muss auch bei der Bearbeitung eines Falls, bei welchem der Ablauf eines Prozesses nicht komplett zur Entwurfszeit festgelegt wird, den Benutzern die Möglichkeit gegeben werden, Aufgaben zur Laufzeit zu ergänzen. Damit erhält gerade der Benutzer ohne tiefgreifende technische Kenntnisse mehr Flexibilität bei der Bearbeitung eines Falls, da der Prozessablauf den persönlichen Bedürfnissen angepasst und zur Laufzeit auf unerwartete Situationen und Ausnahmen reagiert werden kann. [55]

Mit Hilfe der zwei Arten der Flexibilität zur Entwurfszeit und zur Laufzeit lassen sich vier Klassen der Flexibilität beschreiben, mit denen auf Veränderungen in einem PrMS reagiert werden kann, ohne dass eine komplette Neugestaltung der Prozessmodelle notwendig wird. [56]

Flexibility by design ist nur zur Entwurfszeit konfigurierbar. Das bedeutet, dass die Anzahl der möglichen Pfade schon zur Modellierzeit festgelegt wird. Dabei kann es sich um parallele Ausführungspfade oder Entscheidungen handeln, die der Benutzer zur Laufzeit auswählen kann.

Flexibility by deviation und **Flexibility by change** werden ausschließlich zur Laufzeit konfiguriert. Die "Flexibility by deviation"bietet für Benutzer die Möglichkeit, vom vorgegebenen Prozessablauf abzuweichen, indem Prozesschritte übersprungen oder erneut ausgeführt werden. Dabei wird im Gegensatz zur "Flexibility by change"der Prozesstyp nicht angepasst und somit auch die Prozessdefinition nicht dauerhaft geändert.

Flexibility by underspecification wird eingesetzt, wenn bei der Modellierung eines Prozesses bestimmte Ausführungselemente nicht definiert werden können. Dabei ist die Prozessdefinition noch nicht vollständig und wird durch einen Platzhalter ersetzt. Diese Klasse kann sowohl zur Entwurfs- als auch zur Laufzeit konfiguriert werden. Wird der Platzhalter während der Ausführung mit vordefinierten Fragmenten gefüllt, wird dies als Konfiguration zur Entwurfszeit mit Hilfe des *late bindings* definiert. Wird der Platzhalter mit neu erstellten Prozessfragementen zur Laufzeit gefüllt, findet die Konfiguration mit Hilfe des *late modeling* zur Laufzeit statt.

Die Wichtigkeit von Flexibilität in einem Prozessablauf ist nicht neu. Gründe und Lösungsansätze für Anpassungen im Prozessmanagement wurden, ebenso wie die dabei

entstehenden Probleme, früh erkannt und beschrieben. [57]. Damit wird verdeutlicht, dass Änderungen ein wichtiger Teil des Prozessmanagements sind. Ein PrMS muss daher passend, je nach Anwendungsdomäne und Einsatzzweck ausgewählt und auf benötigte Anforderungen bei der Bearbeitung eines Geschäftsfalls angepasst werden. Dabei stellt sich für die Nutzung des Systems die grundlegende Frage, inwieweit ein Benutzer überhaupt vom vordefinierten Prozess abweichen darf oder soll. Muss ein Prozess strikt ausgeführt werden oder wird einem Benutzer mehr Entscheidungsfreiheit eingeräumt, damit dieser auch auf unvorhergesehene Situationen oder persönliche Entscheidungen reagieren kann. Weiterhin muss das PrMS auch die Rechte und Beschränkungen der Benutzer mit passenden Autorisierungsmöglichkeiten berücksichtigen. Folglich muss der Grad an Flexibilität der Anwendung angepasst werden, um auch bei den Anwendern eine hohe Akzeptanz zu erhalten.
In welchem Umfang eine Prozessbeschreibungssprache die Flexibilität in einem PrMS zur Entwurfs- und Laufzeit unterstützen kann und welche Klassen der Flexibilität mit Hilfe einer Sprache modelliert werden können, soll anhand zweier Spezifikationen der OMG ermittelt werden. Der BPMN-Standard steht für eine imperative Sprache, welche sehr gut zur Modellierung von strukturstarken, prozessorientierten Abläufen geeignet ist. Der CMMN-Standard dagegen ist eine deklarative Sprache, welche für die Modellierung von strukturschwachen, fallbasierten Prozessen entwickelt wurde und damit eine datenorientierte Sicht bei der Modellierung umsetzt. Weiterhin erhält der Wissensarbeiter mit CMMN die Möglichkeit, die Ablaufplanung und andere Entscheidungen zur Laufzeit zu treffen. [58]
Auch wenn die vorrangigen Einsatzgebiete der beiden Sprachen gerade beschrieben wurden, ist mit der Einführung von CMMN eine rege Diskussion über die Notwendigkeit und den Unterschied zu BPMN entstanden. Daher werden diese beiden Sprachen im Folgenden vorgestellt, um diese dann im letzten Abschnitt miteinander zu vergleichen und die Gemeinsamkeiten und Unterschiede herauszustellen.

4.3 Grundlagen BPMN 2.0

BPMN hat sich die letzten Jahre als eine der am meisten eingesetzten Modellierspra-chen durchgesetzt. Dies liegt daran, dass die Sprache für viele Arbeitsbereiche und über viele Branchen hinweg eingesetzt werden kann. [42] Ein Grund dafür liegt darin, dass die Notation so konzipiert ist, dass sie von den vielen unterschiedlichen Anwendergrup-pen verstanden und verwendet werden kann. Dies beinhaltet sowohl Fuhrungskrafte, Fachexperten und Softwareentwickler, als auch die Benutzer, welche die Prozesse ausführen und überwachen. BPMN wurde 2004 veröffentlicht und erschien 2006 als offizielle OMG-Spezifikation in der Version 1.0. Diese wurde über die Jahre mehrmals überarbeitet und seit 2011 liegt der Standard in der Version 2.0 vor. BPMN ist als Be-schreibungssprache auf die grafische Darstellung von Prozessen ausgerichtet und kann seit der Version 2.0 in vier verschiedenen Diagrammtypen modelliert werden:

Prozessdiagramm
Dieses stellt die Abläufe dar, die zur Ausführung einer Aufgabe und aller dazugehörigen Schritte benötigt werden.

Choreographiediagramm
Dieses betrachtet vorrangig den Nachrichtenaustausch von Prozessbeteiligten. Dieses Diagramm zeigt die Kommunikation zwischen verschiedenen Beteiligten, die durch se-parate Pools dargestellt werden.

Kollaborationsdiagramm
Dieses stellt wie das Choreographiediagramm den Austausch von Nachrichten zwischen verschiedenen Kommunikationspartnern dar. Es fokussiert sich jedoch stärker auf die Aufgaben der Prozesspartner und damit mehrerer Prozesse.

Konversationsdiagramm. Der vierte Diagrammtyp stellt eine Art Überblickdiagramm dar. Hier wird zum Erhalt der Übersichtlichkeit das Hauptaugenmerk auf das Prozessdia-gramm gerichtet, das die Struktur eines Prozesses am besten darstellt.

Prozessdiagramme werden auch Orchestrierungsdiagramme genannt und bestehen aus fünf verschiedenen Basiselementen (siehe Abb. 4.2). [7] Mit diesen werden die Aktionen beschrieben, die innerhalb eines Geschäftsprozesses auftreten können. Die

Daten werden erst seit BPMN 2.0 als eigenständiges Element betrachtet und erhalten so einen höheren Stellenwert als noch in den vorhergehenden Spezifikationen. [59]
Die Elemente werden im Folgenden kurz vorgestellt.

Flussobjekte (flow objects)

Diese sind Hauptbestandteil der Sprache und bestehen aus den Elementen *Aktivitäten (activity)*, *Entscheidungen (gateway)*, und *Ereignissen(events)*.
Aktivitäten werden entweder als atomare *Aufgabe (task)* oder als *untergeordneter Prozess (sub-process)* dargestellt. Dieser *Teilprozess* kann wiederum ein eigenständiges Prozessdiagramm enthalten.
Gateways erlauben das Aufspalten und Zusammenführen eines Kontrollflusses. Dafür stehen verschiedene logische Bedingungen zur Verfügung und auch mit Hilfe eines Events ist es möglich ein *Gateway* zu steuern.
In einem Prozess können verschiedene Formen von *Ereignissen* auftreten, welche die Prozessausführung beeinflussen. Es wird zwischen *Startereignissen*, *zwischenliegenden Ereignissen* und *Endereignissen* unterschieden.

Verbindende Objekte (connecting objects)

Es gibt drei unterschiedliche Wege, um verschiedene Flussobjekte miteinander zu verbinden.
Einen *Sequenzfluss (sequence flow)*, einen *Nachrichtenfluss (message flow)* und *Assoziationen (association)*. Der *Sequenzfluss* definiert die Reihenfolge, in der Flussobjekte ausgeführt werden.
Der *Nachrichtenfluss* verbindet verschiedene Pools miteinander. Somit werden externe Kommunikationspartner miteingebunden.
Assoziationen werden gebraucht, um Artefakte oder Datenobjekte ohne weitere Semantik im Modell zu verknüpfen.

Schwimmbahnen (swimlanes)

Zur Steigerung der Übersichtlichkeit werden organisatorische Aspekte durch die Be-

schreibung der Elemente *Pool (Becken)* und *Lane (Bahn)* beschrieben. In einem *Pool* können die Zuständigkeiten an einem Prozess dargestellt werden. Dies kann sowohl ein Bearbeiter, als auch eine Abteilung oder ein System sein. Ein *Sequenzfluss* darf nur innerhalb eines solchen *Pools* fließen.
Für eine genauere Struktiereung besteht die Möglichkeit einen *Pool* durch mehrere *Lanes* zu untergliedern. So können beispielsweise mehrere Rollen grafisch unterteilt werden.

Artefakte (artifacts)
Zur Darstellung zusätzlicher Informationen über den Prozess können in BPMN die Elemente *Gruppe (group)* oder *Textannotationen* verwendet werden.
Durch eine *Gruppe* können mehrere Elemente gruppiert und eine Zusammengehörigkeit modelliert werden.
Mit *Annotationen* können Elemente kommentiert und durch Text ergänzt werden.

Daten (data)
Diese beinhalten die Elemente *Dateneingabe (data input)*, *Datenausgabe (data output)*, *Datenspeicher* und *Datenobjekt*. Die bereits erwähnte prozessorientierte Sicht ist hier in BPMN gut zu veranschaulichen. Es exisitert ein atomares *Datenobjekt*, das über Ein- und Ausgabeparameter beschrieben wird und über den *Datenfluss* mit *Aktivitäten* verknüpft ist. Des Weiteren können in BPMN 2.0 *Dateneingabe* Elemente und *Datenausgabe* Elemente modelliert werden.
Dateneingabe Elemente enthält einen Wert, der für die Ausführung eines *Tasks* benötigt wird. Auf der anderen Seite können *Tasks* einen Wert in *Datenausgabe*-Elemente schreiben. *Dateneingabe*- und *Datenausgabe*-Elemente sind nicht Teil des *Datenflusses*.
All diese Daten können nur innerhalb eines Prozesses und nicht über *Pool*-Grenzen hinweg verwendet werden.

Diese Übersicht beinhaltet nicht alle Elemente von BPMN, sondern soll nur als Grundlage für das Verständnis der Sprache dienen. Eine komplette Ausführung aller Elemente

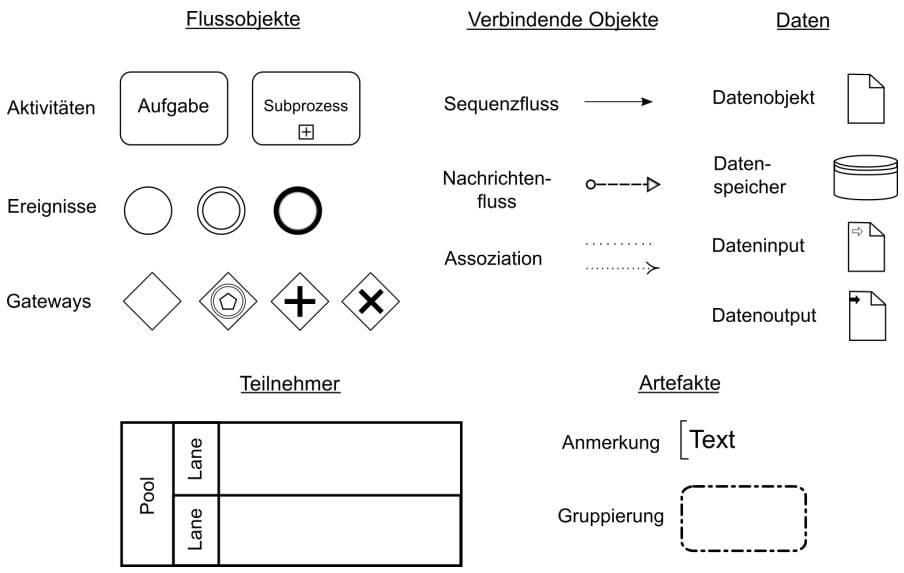


Abbildung 4.2: Basiselemente in BPMN 2.0, vergleichbar [60]

wird in der Spezifikation von der OMG beschrieben. [7] In einem nun folgenden Beispiel werden die Grundlagen von BPMN verdeutlicht. Dabei wird ein Ablauf in einer Personalabteilung dargestellt.

4.3.1 Modellierungsbeispiel aus dem Personalmanagement

Als Anwendungsbeispiel für die Modellierung eines Prozesses in BPMN und CMMN wird der Ablauf einer Stellenausschreibung in einem Unternehmen aus dem Bereich des Personalmanagements herangezogen.
Dies stellt das gleiche Beispiel wie in der Forschung zur PHILharmonicFlows dar und soll eine bessere Vergleichbarkeit bei der anschließenden Evaluation von CMMN und PHILharmonicFlows ermöglichen. [45] Weiterhin können mit Hilfe dieses Beispiels die Vor- und Nachteile sowie die Grenzen bei der Modellierung mit einer prozessorientierten und einer fallorientierten Sprache Modellierungssprache erläutert werden.

Das Beispiel beschreibt eine Bewerbung bei einer Personalabteilung eines Unternehmens und wird über ein Online-Formular erstellt. Dies wird mittels eines PrMS bearbeitet und überwacht.
Der Ablauf der Stellenausschreibung beginnt mit der Einstellung einer Ausschreibung in das Online-System. Darüber kann sich der Bewerber auf die Stelle bewerben. Dabei füllt er alle notwendigen Informationen aus und schickt diese ab. Die Bewerbung wird in der Personalabteilung bearbeitet und geprüft. Wird die Bewerbung abgelehnt, bekommt der Bewerber eine Absage. Ansonsten kann der Personalbearbeiter die Bewerbung zu einer genaueren Prüfung in die Fachabteilungen weiterleiten. Dort werden Gutachten zu einer Bewerbung erstellt, wobei die Anzahl der Gutachten in jedem Fall variieren kann. Die Bearbeitung in den einzelnen Abteilungen muss innerhalb einer zeitlich festgesetzten Frist abgeschlossen werden, wobei die Mitarbeiter die Erstellung eines Gutachtens auch ablehnen können. Nach Fertigstellung der Gutachten, wird der Personalbearbeiter anhand aller Gutachten den Bewerber zu einem Vorstellungsgespräch einladen, eine Absage formulieren oder bei Bedarf die Bewerbung mit anderen Stellenausschreibungen im Unternehmen abgleichen. Bei einem Gespräch wird entschieden, ob der Bewerber für die Stelle geeignet ist oder die Stelle weiter ausgeschrieben bleibt. Somit wird der Prozess entweder durch eine Absage des Bewerbers oder dessen Einstellung im Unternehmen abgeschlossen.

4.3.1.1 Grafische Umsetzung mit BPMN 2.0

Das Beispiel zeigt den Prozessablauf einer Bewerbung, modelliert mit der Prozessbeschreibungssprache BPMN (siehe Abb. 4.3). Dabei wird ein abstraktes Geschäftsprozessdiagramm gezeigt, welches ähnlich wie ein Kollaborationsdiagramm auch die Nachrichtenflüsse zwischen dem Pool der *Bewerber* und dem Pool des *Unternehmens* darstellt und so eine neutrale Sichtweise auf den Prozess darstellt. Jedoch wird der Pool *Bewerber* nur in Form einer Black-Box-Darstellung modelliert, da dieser hier nicht näher untersucht wird. Der Pool *Unternehmen* wird unterteilt in die Lanes *Personalabteilung* und in drei verschiedene *Fachabteilungen*. In den Fachabteilungen werden gegebenenfalls die Gutachten zu einer Bewerbung erstellt. Der restliche Prozess wird komplett in

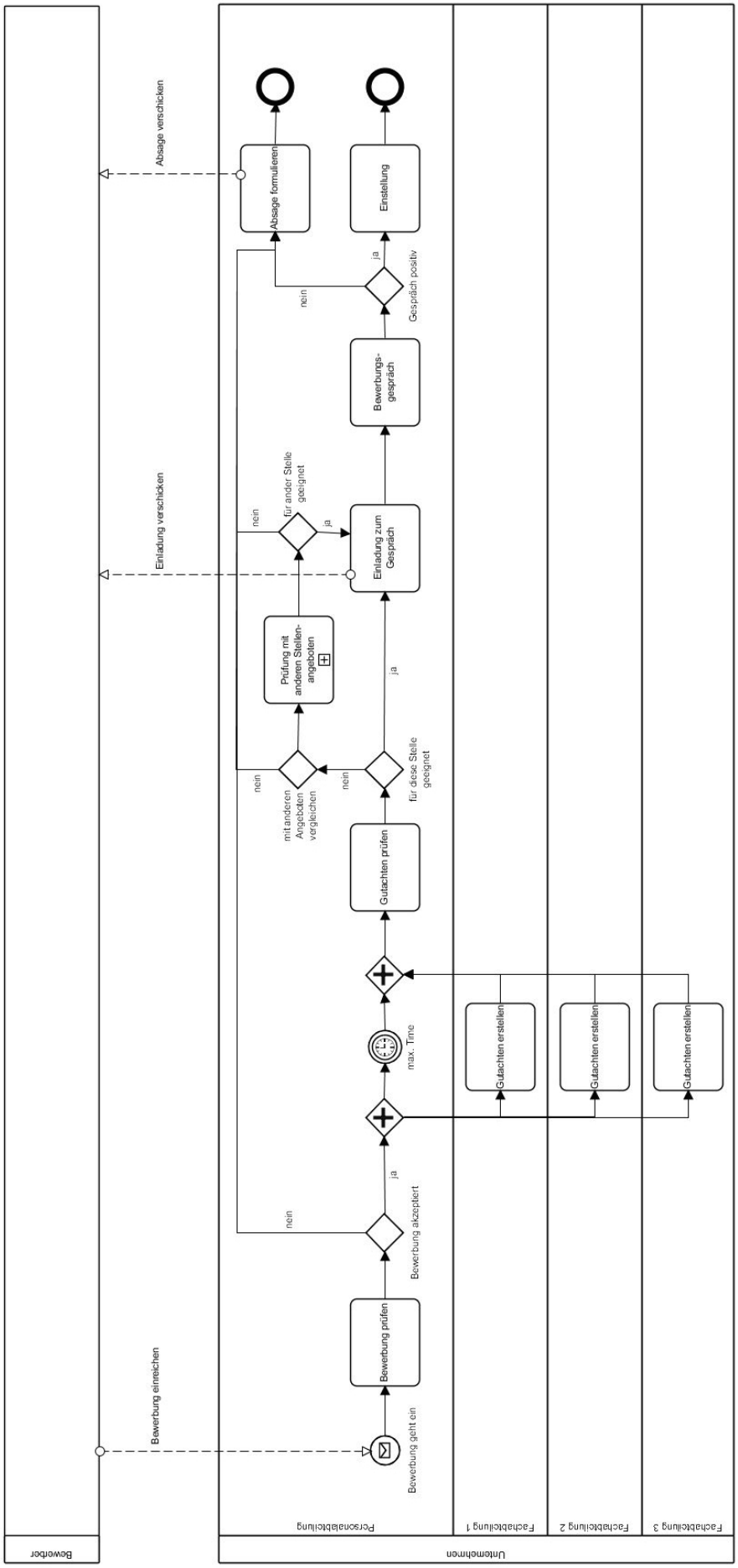


Abbildung 4.3: Stellenausschreibung mit BPMN 2.0, erstellt mit [61]

der Personalabteilung bearbeitet und stellt den Ablauf einer Bewerbung vom Eintreffen der Bewerbungsunterlagen bis zur Einstellung oder Absage des Bewerbers dar. Mit diesem Beispiel lassen sich sowohl viele positive Aspekte der Sprache als auch die Grenzen von BPMN aufzeigen.

4.3.1.2 Vor- und Nachteile bei der Modellierung mit BPMN 2.0

Durch den Kontrollfluss in dem Bewerbungsbeispiel (siehe Abb. 4.3) ist eine genau Ablaufreihenfolge vorgegeben, welche einen standardisierten Ablauf ermöglicht und durch die Trennung der verschiedenen Abteilungen eines Unternehmens mit einzelnen Lanes eine übersichtliche grafische Darstellung des Prozesses aufzeigt. Auch der modellierte Nachrichtenaustausch bringt einen Gewinn an Übersichtlichkeit. Zudem ist das Modell auch für den normalen Benutzer leicht verständlich. Weiterhin werden die Aktivitäten für die *Prüfung der Bewerbung mit anderen Stellenangeboten* in einem untergeordneten Prozess zusammengefasst, womit ebenfalls die Lesbarkeit des Modells erhöht wird. Alle genannten Punkte führen zu einer besseren Bearbeitung des Prozesses und helfen somit sowohl bei dessen Dokumentation, als auch bei der Entwicklung von neuen Optimierungsansätzen. Durch die prozessorientierte Ausrichtung von BPMN zeigen sich bei der Modellierung dieses Beispiels allerdings auch einige Schwachstellen dieser Beschreibungssprache.
So müssen für jede Bewerbung alle Fachabteilungen in das Modell miteinbezogen werden, da das Modell für jede Bewerbung im Vorfeld standardisiert modelliert wird. Für den Benutzer der Personalabteilung ist es daher nicht möglich, die Prozessinstanz durch eine gezielte Auswahl der Fachabteilungen anzupassen. Bei einem großen Unternehmen würde das zu einem sehr ineffizienten Modellierung führen, welches alle Fachabteilungen auflistet. Auch der Versuch einer Modellierung durch eine Reihe von Auswahlentscheidungen, mit Hilfe einer Vielzahl von Gateways, würde an einem viel zu komplexen Diagramm scheitern. Daneben ist es auch nicht möglich die Dauer für die Bearbeitung der Gutachten individuell und fallbezogen auszuwählen. Damit ist dieser Zeitrahmen fest vorgegeben und kann nur durch eine Modelländerung angepasst werden. Falls in einer Instanz kein Gutachten für eine Bewerbung erstellt wird, ist es auf

Grund der Abhängigkeit einer Aktivität von der Beendigung der vorhergehenden Aktivität in einem BPMN-Modell nicht möglich den Task *Gutachten prüfen* zu überspringen, auch wenn dieser auf Grund von Laufzeitentscheidungen nicht mehr ausgeführt werden muss. Dieses Beispiel zeigt, dass sich mit dem imperativen Ansatz der Sprache nicht alle Bedürfnissen des Bearbeiters modellieren lassen. Wird einem Benutzer mehr Flexibiltät für die Bearbeitung seiner Aufgaben überlassen, ist es mit einer deklarativen Beschreibungssprache über Geschäftsregeln möglich, den Ablauf des Prozesses auch zur Laufzeit individueller zu gestalten. Daher wird im folgenden Abschnitt CMMN vorgestellt, welche die Modellierung von Arbeitsabläufen ermöglicht, die nicht vollständig zur Entwurfszeit strukturiert werden können. Damit wird dem Wissensarbeiter eine daten- oder fallzentrierte Sicht auf einen Prozess ermöglicht.

4.4 Dynamische Case Management

BPMN als Modelliersprache für die Unterstützung von klassischem BPM kann als sehr weit entwickelt angesehen werden. Es wird vielseitig eingesetzt und bietet in vielen Bereichen eine optimale Unterstützung bei der Prozesserstellung. Jedoch steigt die Unzufriedenheit auch bei einfachen Anwendern, wenn es um die Erstellung von datenzentrierten Aufgaben geht. [42]
Klassische PrMS bilden einen Lebenszyklus ab, der aus der Modellierung, der Implementierung, der Ausführung und der Kontrolle eines Prozesses besteht. Dadurch müssen alle erdenklichen Ausführungsmöglichkeiten eines Prozesses bereits in der Modellierungsphase berücksichtigt und umgesetzt werden. Alle benötigten Aspekte, die nicht modelliert werden, können erst bei der nächsten Modellüberarbeitung integriert werden, da eine Änderungen zur Laufzeit, während der Bearbeitung einer Prozessinstanz nicht vorgenommen werden kann. [62] Werden Aufgaben modelliert, die auf Erfahrungen und Entscheidungen der Benutzer basieren, muss eine alternative Modellierungsmöglichkeit gesucht werden, da klassische Systeme dies nicht oder nur unzureichend unterstützen. (siehe Abb. 4.1)

Durch die Erkenntnis, dass der Bereich der wissensintensiven Arbeiten immer weiter zunimmt und die Systeme sich mehr an den Benutzer anpassen sollen anstatt umgekehrt, ist das Case Management entstanden. Der Begriff selbst ist nicht neu. So wurde bereits in den 80er Jahren versucht fallbasierte Probleme durch Geschäftsregeln zu implementieren, [63] und bereits 1990 wurde die *Case Management Society of America* [64] gegründet.
Der Begriff kommt ursprünglich aus den Bereichen, in denen typischerweise mit Fällen gearbeitet wird. Dazu gehören das Gesundheitswesen oder das Rechts- und Justizwesen. Jedoch waren diese Systeme in den genannten Geschäftsfeldern lange Zeit fest implementiert. [65] Erst mit der zunehmenden Entwicklung von BPM in Unternehmen stieß das Thema, welches sich mit der Integration erweiterter Flexibilität in klassischen PrMS beschäftigt, in die letzten Jahre wieder auf erhöhtes Interesse. Daraus entstand eine neue dynamische BPM-Lösung, das Case Management.[66]
Unter dynamischem BPM versteht man ein System, das durch Entscheidungen von Wissensarbeitern zur Laufzeit angepasst werden kann. Dazu zählen optionale Aktivitäten oder auch eine individuelle Anpassung der Prozessabfolge. In Bezug auf Case Management (CM) bedeutet das, dass ein Fall sehr dynamisch sein kann, sich häufig ändert und durch neue Informationen oder Ereignisse während der Ausführung beeinflusst wird. Darauf muss das System reagieren können. [37] Dabei sollte ein dynamische System auch in der Lage sein, aus abgeschlossenen Fällen zu lernen, um Vorschläge für die Bearbeitung neuer Fälle bereitzustellen.

Als Abgrenzung eines CM-Konzepts gegenüber einem klassischen BPM-Ansatz wurden aus verschiedenen Forschungsarbeiten folgende Eigenschaften zusammengetragen. Ein dynamisches BPM-System ist damit: [38, 37, 62, 67]

wissensintensiv
Wie bereits in Kapitel 2.2 beschrieben, sind damit Prozesse gemeint, die sich selten wiederholen und schwer vorhersehbar sind. Dadurch wird ein tiefes fachliches Wissen der Mitarbeiter benötigt. Ein CM-System ist in der Lage solche Prozesse zu unterstützt.

entscheidungsunterstützend
Wissensarbeiter brauchen bei individuellen Entscheidungen ein System, welches sie bei dieser Art der Arbeit unterstützt. So kann im System die Aktivierung optionaler Aktivitäten gesteuert und verwaltet werden. Weiterhin kann ein Bearbeiter auf Einflüsse und unvorhergesehene Ereignisse reagieren und die Reihenfolge eines Prozesses anpassen. So ist auch das Löschen oder Überspringen von Aktivitäten möglich.

langlaufend
Ein Fall kann auch über einen sehr langen Zeitraum aktiv sein. Ein Rechtsfall kann sich auch über Jahre hinwegziehen. Solche Aspekte werden von einem CM-System berücksichtigt.

datenzentriert, context tunneling
Die Bearbeitung und der Fortschritt eines Falles wird anhand der Datenwerte im Prozess ermittelt. Der Benutzer erhält in einem Fall lesenden Zugriff auf alle relevanten Daten. Diese Sicht ist somit viel weitreichender und beschränkt sich nicht nur auf einzelne Aktivitäten. Das System gewährleistet so einen inhaltlichen Überblick über den gesamten Fall.

kollaborativ
Oft sind mehrere Personen an der Bearbeitung eines Falles beteiligt. Ein CM-Sytem muss daher Werkzeuge für die Koordinierung und Kommunikation aller beteiligten Benutzer sicherstellen.

transparent
Um eine effiziente Ausführung in einem System zu gewährleisten, müssen Benutzer in einem Fall immer erkennen, welche Aufgaben sie zu bearbeiten haben, welche Fristen einzuhalten sind, welche Informationen sie benötigen und welche schon vorhanden sind. Somit entsteht ein Überblick darüber, wie weit ein Prozess fortgeschritten ist und wie und wann das Ziel des Prozesses erreicht wird. Zusätzlich kann durch ein transparentes System eine doppelte Ausführung von Aufgaben verhindert werden.

regelbasiert
Die Ausführungsreihenfolge eines Prozesses in einem CM-System wird über Beschränkungen geregelt. So wird festgelegt, welche Aufgaben ausgeführt werden müssen,

welche optional ausgeführt werden können und bei welchen Aufgaben die Ausführung untersagt ist. Damit handelt es sich um einen deklarativen Ansatz zur flexibleren Gestaltung von Prozessen.

ereignisbasiert
Der Ablauf eines Falls wird nicht nur durch den Benutzer gesteuert. Auch Ereignisse können den Verlauf des Falls beeinflussen, oder die Bearbeitung zusätzlicher Aufgaben anstoßen.

zielorientiert
Das System gibt keinen Weg vor, wie ein Ziel erreicht wird. Sondern das Ziel ist vorgegeben und der Bearbeiter kann selbst entscheiden, welchen Ausführungspfad er benutzt, um dieses Ziel zu erreichen.

Inwieweit diese Eigenschaften durch eine Modellierungssprache unterstützen werden können, wird im folgenden Abschnitt mit CMMN gezeigt.

4.5 CMMN

Die bessere Integration von wissensintensiven Prozessen, datenzentrierten Ansätzen und auch von fallbasierten Konzepten in ein PrMS ist kein neues Phänomen. Daher gehen der Veröffentlichung des neuen Modells und Notationsstandards CMMN durch die OMG im Mai 2014, viele Forschungen und Entwicklungen in diesen Gebieten voraus, die in den letzten zwei Dekaden im Bereich des Prozessmanagements stattgefunden haben Die Entwicklung der Notation selbst, startete mit einem Vorschlag der OMG 2009, [68], der sich für die Entwicklung eines Prozessmodell-Standards für fallbasierte Prozesse aussprach und endete letzten Jahres mit der Veröffentlichung von CMMN 1.0. In den Entwicklungsprozess des The Case Management Process Modeling (CMPM)-Vorschlags waren auch zehn Unternehmen mit eingebunden, darunter auch SAP, Oracle, Pallas Athena oder IBM.
Diese haben den CMMN-Standard mit Regeln, Bezeichnungen und Abläufen für fallbasiertes Prozessmanagement spezifiziert und dabei auch Elemente aus anderen Forschungsansätzen und eigenen Frameworks integriert.

Ein Konzept, das die Ausgestaltung von CMMN mit beeinflusste, ist der bereits vorgestellte artefaktzentrierte Ansatz, welcher auf einem datenzentrierten Ansatz beruht. [30] (siehe 2.4.1) Dieser besteht aus zwei Modellteilen. Einem Informationsmodell, das alle relevanten Informationen zu einem Prozess enthält sowie einem Lebenszyklusmodell, welches das Verhalten zur Laufzeit beschreibt. Dies wurde in die Beschreibungssprache übernommen, so dass CMMN ebenfalls aus einer verhaltensbezogenen Sicht und einem Informationsmodell besteht. Geschäftsartefakte wurden schon 2003 eingeführt [27] und seitdem laufend weiterentwickelt. Der artefaktzentrierte Ansatz beinhaltete zuerst nur einen imperativen Ansatz. [29] Dieser wurde durch den GSM-Ansatz (siehe 2.4.2) mit einem deklarativen Ansatz erweitert und kann als Grundlage dieser Notation angesehen werden. [69] Die datenzentrierte, deklarative Ausrichtung erweitert CMMN um eine regelbasierte Ausführungsgestaltung (siehe: 2.3.2) und integriert damit Forschungsansätze, die bereits in einem Vortex System 1999 Anwendung fanden. [30, 70]
In CMMN wurden große Teile des GSM-Ansatzes übernommen und durch einige Eigenschaften erweitert. So ist es in CMMN möglich, einen Milestone auch außerhalb einer bestimmten Stage zu benutzen.
Im Gegensatz zu GSM gibt es weiterhin Wiederholungsstrategien, die auf Stages und Tasks angewendet werden können.
Die letzte Erweiterung betrifft die Implementierung benutzerbestimmbarer Aufgaben (discretionary tasks). Damit werden Aktivitäten modelliert, über dessen Einsatz ein Bearbeiter zur Laufzeit selbst entscheiden kann. Damit entsteht eine hohe Flexibilität, sowie eine individuellere Prozessausführung. [9] Diese Flexibilität ähnelt der *Flexibility by underspecification*. Dabei werden die Konzepte des *late modeling* und des *late binding* miteinander verknüpft. Somit ist es erlaubt, unstrukturierte Anteile eines Geschaftsprozesses in das Modell zu integrieren. Jedoch müssen die Aktivitäten bereits zur Entwurfszeit modelliert werden. Trotzdem erhält der Benutzer damit weitreichende Gestaltungsmöglichkeiten bei den Arbeitsabläufen, da CMMN eine Modellierung ermöglicht, ohne dabei die Anzahl an Ausführungsschritten im Vorfeld zu definieren. [35]
Der zweite großer Forschungsschwerpunkt, der die Grundlage für die Entwicklung von CMMN vorgibt, beschäftigt sich mit der Sicht auf den Fall selbst. Ein Fall stellt einen wissensintensiven Geschäftsprozess dar, welcher von Wissensarbeitern mit Hilfe von

Informationen individuell bearbeitet werden muss. Da die Bearbeitung mit starren Strukturen eines prozessorientierten PrMS nicht möglich ist, wurde Anfang 2000 anhand Charakteristika von Wissensarbeitern [71] ein fallbasiertes Konzept für die Unterstützung solcher Prozesse entwickelt. [72, 73, 74, 38] Dieser CH-Ansatz, der bereits in Kapitel 2.4.3 vorgestellt wurde, unterstützt Wissensarbeiter während eines Prozesses und bei der Bearbeitung von unvorhergesehenen Ereignissen.
Das ist auch das Ziel der Prozessbeschreibungssprache CMMN. Eine übersichtliche Modellierung eines wissensintensiven Prozesses, unter Einbeziehung der Bearbeiter. Daher soll im folgende Abschnitt der CMMN-Standard vorgestellt und anhand von grundlegenden Eigenschaften, Struktur und Elementen genauer untersucht werden.

4.5.1 Struktur von CMMN

Mit Hilfe der wichtigsten Metamodelle von CMMN soll der Aufbau der fallbasierten Sprache erläutert werden. In diesem Metamodell werden die Anforderungen und Regeln für eine formale Beschreibung von CMMN erläutert, um das Verständnis für die spätere grafische Dastellung durch die Notation zu erleichtern. Hierfür wird das Kernmodell, das Informationsmodell, das Fallmodell und das Planungsmodell näher beschrieben, um einen Überblick über die Struktur der Notation zu erhalten. Dabei wird auf die Klassen der einzelnen Diagramme näher eingegangen. Anschließend werden die Modellierungselemente zur Beschreibung eines Falles besprochen und auf den Lebenszyklus eines Falles bei der Ausführung eingegangen. [10]

Kernaufbau der Sprache (core infrastructure)
An äußerster Stelle des CMMN-Metamodells steht die *Definitionsklasse*. (siehe Abb. 4.4 Diese beinhaltet Objekte für alle *CMMN-Elemente*. Die *Definitions*klasse erbt von der *CMMN-Elemente* und verknüpft so alle Objekte von CMMN. Der Austausch von CMMN-Daten wird immer über eine oder mehrere *Definitionen* geleitet. Die Klasse ist mit vier weiteren Klassen verknüpft. Die *Import*klasse ist für den Import extern definierter Typen. Welche Typen im speziellen zugelassen sind, wird vom Notationsstandard nicht beschrieben. Jedoch werden XML-Schemadefinitionen als Beispiel genannt und empfohlen.

Eine weitere Klasse ist das *Process* Element. Damit entsteht die Möglichkei,t andere Notationsspezifikationen in CMMN mit zu integrieren. Dazu gehört auch BPMN. Die anderen Klassen *CaseFileItemDefinition* und *Case* verknüpfen deren jeweilige Strukturen mit dem Kernmodell und werden im Folgenden näher untersucht.

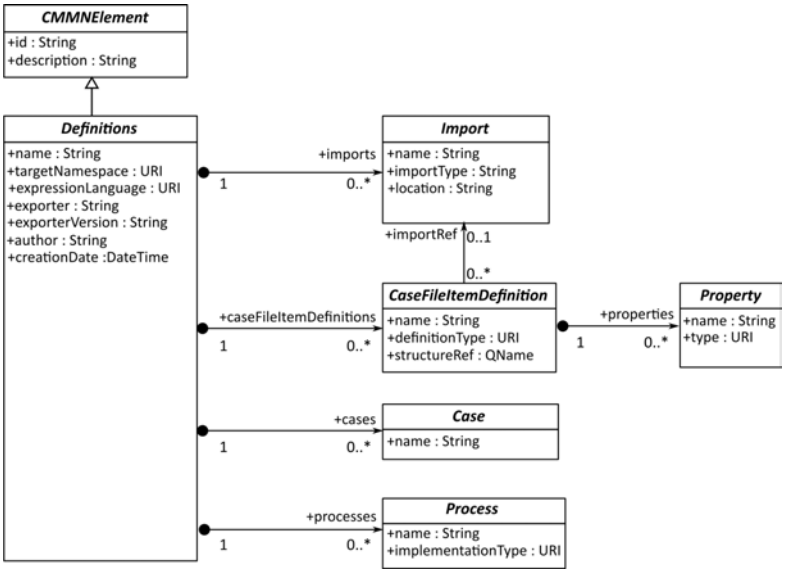


Abbildung 4.4: Diagramm des Kernaufbaus von CMMN, vergleichbar [10]

Elemente des Fallmodells

In diesem Metamodell (siehe Abb. 4.5 wird der eigentliche Fall angelegt und durch verknüpfte Klassen näher definiert und beschrieben. Der Fall besteht somit aus einen *CasePlanModel*, einem *CaseFileModel* und mehreren *caseRoles*. Weiterhin können Ein- und Ausgabe *CaseParameter* mit dem Fall verknüpft werden, um Informationen mit anderen Fällen auszutauschen. Der *caseFile* wird somit zu einem Container, der die Daten für andere Systeme und Benutzer außerhalb des Systems zugänglich macht. Für die Ausführung und Planung von manuellen Aufgaben durch Bearbeiter oder Bearbeitergruppen werden über die *CaseRole* Klasse Benutzer-Rollen definiert. So kann die Autorisierung für einen Fall sichergestellt werden. Ebenfalls können mit den vorhanden Rechten benutzergesteuerte Events ausgelöst werden. Zur Übersichtlichkeit können in einem Fall Aufgaben in *stages* zusammengefasst werden. Dabei ist die äußerste *stage*

der Fall selbst, also das *casePlanModel*. Jeder Fall ist mit genau einem *CasePlanModel* und genau einem *CaseFileModel* verknüpft, welches jetzt genauer beschrieben wird.

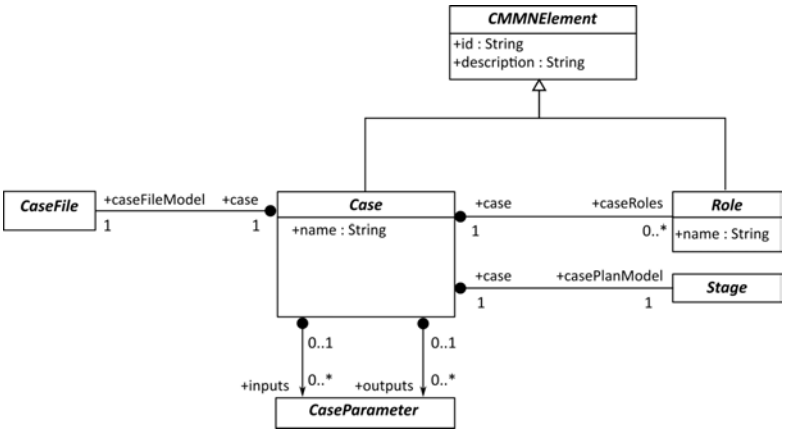


Abbildung 4.5: Diagramm des Aufbaus eines Falls in CMMN, vergleichbar [10]

Elemente des Informationsmodells

Alle Informationen und Daten eines Falls werden im Informationsmodell eines Falls zusammengefasst. Alle Informationen oder Referenzierungen, die für die Bearbeitung des Falles wichtig sind, werden in einem *CaseFile* im Metamodell (siehe Abb. 4.6 dargestellt. Dieses besteht aus sogenannten *CaseFileItems*. In dieser Klasse wird jede erdenkliche Art von Information definiert. Somit können sowohl Ordner, Dokumente und Ordnerstrukturen, die andere *caseFileItems* beinhalten, als auch komplexe oder unstrukturierte Informationen eingebunden werden. Auch Informationen, wie beispielsweise XML-Dokumente werden unterstützt. Die Eigenschaften und Metadaten zu den Sprachen oder Strukturen der verschiedenen Elemente, und damit zu den verschiedenen *CaseFileItems*, werden durch die *caseFileItemDefinition* definiert. So entsteht in CMMN eine sehr flexible Einbindung von Daten, was einen Aspekt für eine datenzentrierte Ausrichtung des CMMN-Standards aufzeigt.

Elemente des Planmodells

Das *CasePlanModel* beinhaltet sowohl alle Elemente, die für den Anfangsplan (initial

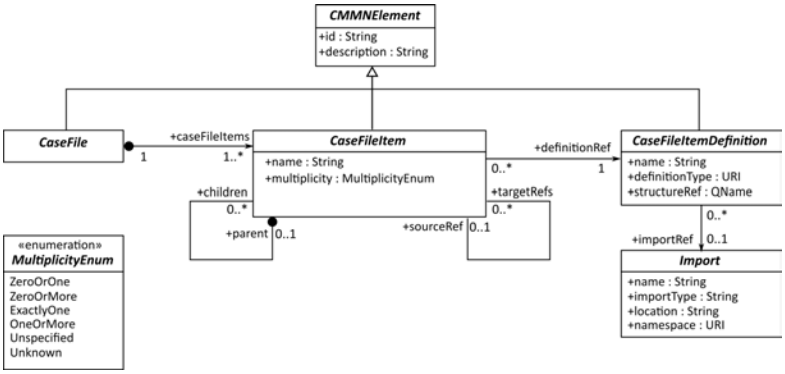


Abbildung 4.6: Diagramm mit den Elemente und Verknüpfungen des Informationsmodells [10]

plan) eines Falles von Bedeutung sind, als auch alle Elemente, die für die Unterstützung zur Laufzeit nötig sind. Diese weiterführende Unterstützung benötigt der Bearbeiter für die Planung des Ablaufs. In einem *CasePlanModel* wird somit das Verhaltensmodell eines Falls dargestellt. Zusammen mit den Elementen des Informationsmodells lässt sich gut die Integration des artefaktzentrierten Ansatzes bei der Entwicklung des CMMN-Standards erkennen. Dieser besteht ebenfalls aus einem Informationsmodell und einem Lebenszyklusmodell zur Ablaufsteuerung. Das Metamodell (siehe Abb. 4.7) *PlanItemDefiniton* zeigt die Möglichkeiten zum Aufbau eines Falls *PlanItemDefiniton*. Dabei handelt es sich bei *PlanItemDefiniton* um eine abstrakte Klasse, die von *CMMNElement* vererbt wird. Eine weitere Klasse dieses Metamodells ist *PlanItemControl*. Diese definiert Vorgaben für die Steuerung und Überwachung der *PlanItemDefinitions*. Die weiteren Konzepte dieses Metamodells sind das *PlanFragements* (und *Stages*), der *Milestones*, der *EventListener* und der *Tasks* Diese werden zusammen mit der Notation von CMMN im nächsten Abschnitt genauer untersucht.

4.5.2 Modellierungselemente von CMMN

Dieser Abschnitt beschreibt die wichtigsten grafischen Elemente des CMMN-Standards, die zur Modellierung eines Falls notwendig sind. Ein Fall beschreibt einen Ablauf von Aktivitäten in einer bestimmten Situation, mit dessen Hilfe ein bestimmtes Ergebnis

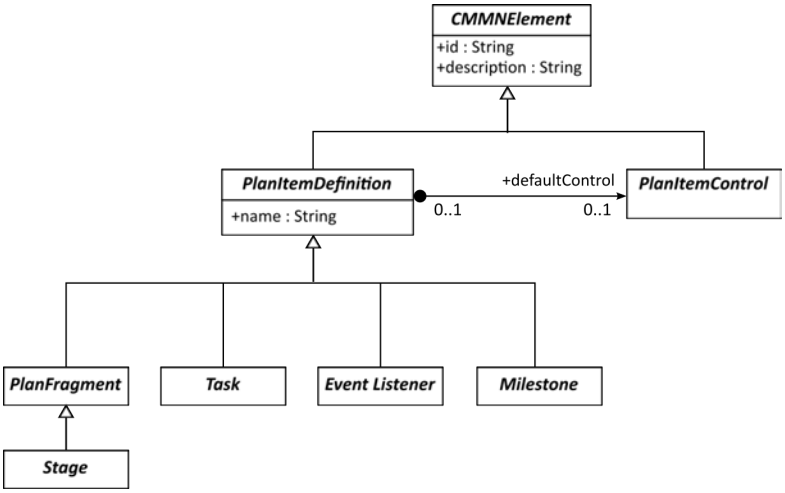


Abbildung 4.7: Diagramm des Planmodells [10]

erreicht werden soll. Durch die Modellelemente wird das Lebenszyklusmodell der Sprache beschrieben. Wird dieses mit dem Verhalten des Falls verknüpft, wird auch das Informationsmodell sichtbar. Die Symbole der Sprache lehnen sich grafisch an den BPMN-Standard an, werden jedoch teilweise unterschiedlich verwendet oder haben eine andere Bedeutung. Eine vollständige Übersicht aller Notationselemente findet sich in der CMMN-Spezifikation der OMG. [10]

Zur Modellierung eines Falls steht an äu/sserster Stelle das **CasePlanModel**. Es wird durch einen Ordner repräsentiert und enthält den Namen des Falls in einem Reiter. Alle weiteren Elemente des Falls werden innerhalb dieses Ordners modelliert.
Das **CaseFileItem** wird durch ein Dateienbild dargestellt und enthält relavante Daten eines Falls. Diese werden im *CaseFile* abgelegt. Für jeden Fall existiert genau ein CaseFile.
Der **Task** ist eine atomare Aufgabeneinheit und wird als Rechteck mit abgerundeten Ecken dargestellt. Dabei kann mit Hilfe eines Symbols zwischen verschiedenen Tasktypen unterschieden werden, um unterschiedliche Anforderungen zu modellieren.
Human Task: Dies beschreibt eine Aufgabe, welche durch einen Fallbearbeiter ausgeführt wird. Solange diese Aufgabe bearbeitet wird, darf keine andere Aufgabe bearbeitet werden. Dies wird als *Blocking Task* bezeichnet. Ist ein sofortiges Fortfahren der Aufga-

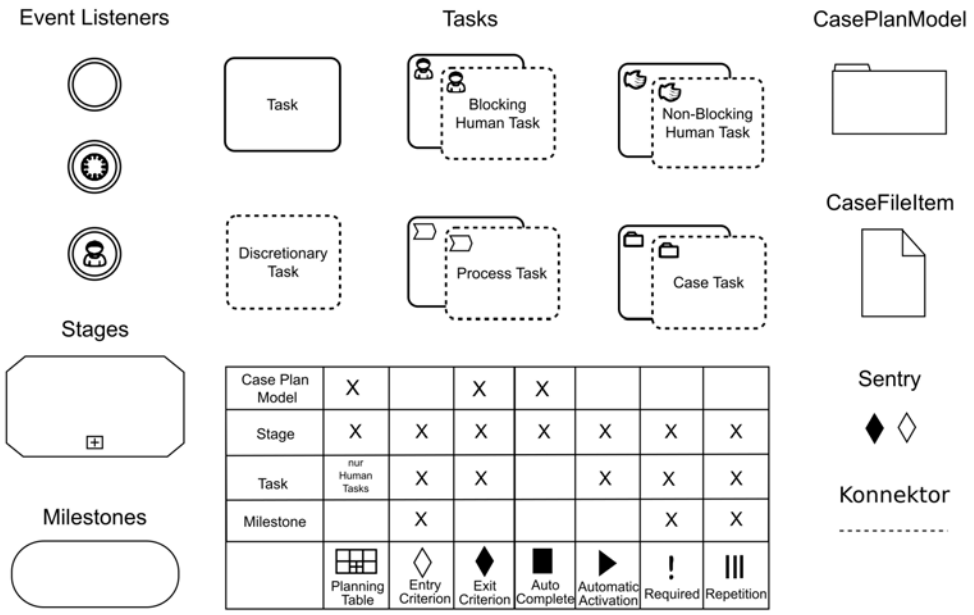


Abbildung 4.8: Basiselemente in CMMN

ben durch andere Bearbeiter erwünscht, kann dies mit einem *Non-blocking Task* erreicht werden.

Process Task: Er wird benutzt, um einen anderen Geschäftsprozess aufzurufen. Damit kann beispielsweise auch ein BPMN-Modell in den Fall eingebunden werden.

Case Task: Dieser ermöglicht das Aufrufen eines anderen Falls.

Discretionary Task: Der Task wird durch ein gestricheltes Rechteck dargestellt und modelliert die bereits vorgestellten Taskarten, ohne dass diese Teil des vorgegebenen Fallablaufs sind. Besitzt ein User zur Laufzeit die nötigen Rechte und benötigt einen Discretionary Task zur Bearbeitung seines Falls, kann dieser flexibel mit in die Prozssabfolge eingebunden werden.

Eine **Stage** wird durch ein Rechteck mit abgeflachten Ecken dargestellt, und stellt eine Art Episode eines Falls dar. In einer Stage können mehrere Tasks oder auch andere Stages zusammengefasst werden. Die äußerste Stage eines Falls ist das *CasePlanModel*.

Tasks und Stages werden von **Sentries** überwacht . Diese bewachen den Lebenszyklus verschiedener Bestandteile eines Falls und reagieren auch auf den Lebenszyklus anderer Elemente. Ein Sentry ist eine Kombination aus Events und Zuständen. Das Event

innerhalb des Sentrys wir durch einen OnPart, der Zustand durch einen IfPart geregelt. Diese definieren Eintritts- und Abbruchbedingungen. Dabei entspricht ein Sentry einer ECA-Regel (Event-Condition-Action) und nehmen eine der folgenden Formen an:
Entweder besteht das Sentry aus einem Eventteil und einen Zustandsteil mit der Form *on <event> if <condition>* oder das Sentry besthet nur aus einem Eventteil *<event>* oder nur aus einem Zustandsteil in der Form *<condition>*
Ein **PlanFragment** kann als ein Container von PlanItems und Sentries gesehen werden. Beispielsweise kann die Verbindung von zwei Tasks über einen Konnektor und einem Eintritts-Sentry als ein PlanFragment bezeichnet werden.
Durch den **Konnektor** können somit verschiedene Abhängigkeiten zwischen Elementen ausgedrückt werden. Konnektoren können dabei den OnPart eines Sentries darstellen, um Abhängigkeiten zu modellieren, oder AND- und OR-Verknüpfungen darstellen.
Milestones sind dazu da, ein erreichbares Ziel aufzuzeigen. Weiterhin kann anhand der bereits erreichten Milestones der Fortschritt eines Prozesses nachvollzogen werden. Mit einem Milestone werden keine Arbeiten verknüpft.
EventListener ist ein Ereignis, welches während der Ausführung des Falles auftritt. Dadurch wird eine Stage, ein Task oder ein Milestone beeinflusst und führt zur Aktivierung oder Terminierung dieser Elemente. Somit können Ereignisse modelliert werden, die durch ein PlanItem nicht abgebildet werden können. Es existierten unterschiedliche EventListerner. Ein TimeEventListener löst vordefinierte Zeitangaben aus. Ein UserEventListener gibt dagegen die direkte Interaktion zwischen einem Benutzer und einer Aufgabe frei.

Zum besseren Verständnis der einzelnen Elemente, wird nun das Bewerbungsbeispiel mit Hilfe der CMMN-Sprache modelliert.

4.5.3 Grafische Umsetzung mit CMMN

Die einzelnen Schritte des Prozessablaufs einer Stellenausschreibung in einer Personalabteilung wurden bereits in Abschnitt 4.3.1 erläutert und mit BPMN umgesetzt (siehe 4.3.1.1. Im Folgenden wird der Ablauf mit CMMN modelliert.

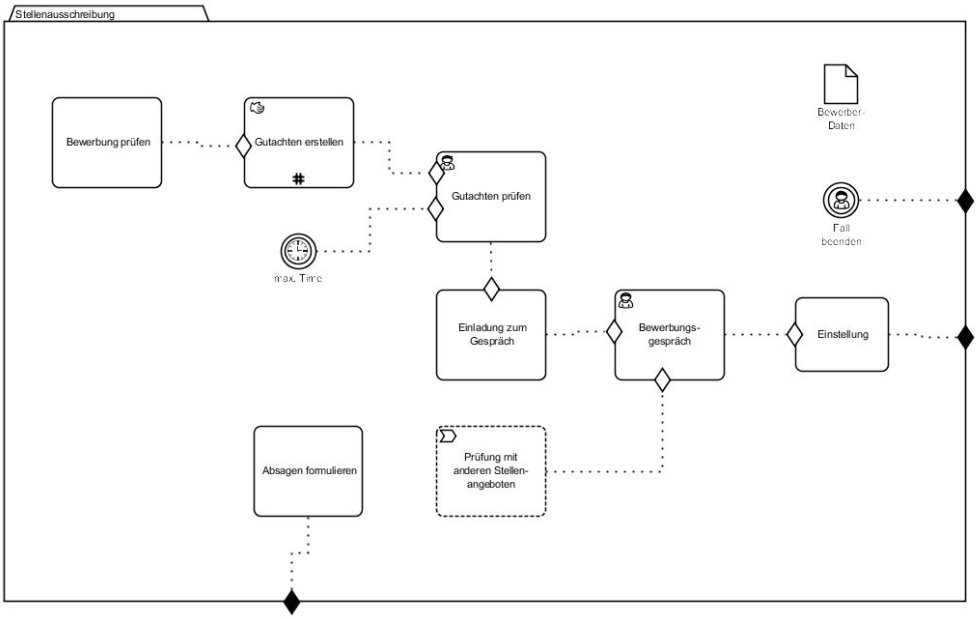


Abbildung 4.9: Stellenausschreibung in CMMN 1.0, erstellt mit [61]

Mit dem Eingang einer Bewerbung wird der Fall geöffnet. Darin befinden sich zwei Aktivitäten, die zur Ausführung bereit sind und bearbeitet werden können. Somit kann direkt mit der Aktivität *Bewerbung prüfen* begonnen werden. Weiterhin ist es während des gesamten Falles möglich, den Bewerbungsprozess auf verschiedene Arten zu beenden. So kann der Fall jederzeit von einem berechtigen Benutzer, mittels des *Fall beenden*-EventListeners, beendet werden oder der Bewerbungsvorgang kann mit der Aktivität *Absage formulieren* beendet werden. Die beiden Aktivitäten können sowohl automatisiert als auch von einem Benutzer bearbeitet werden und unterliegen keinen genaueren Einschränkungen.
Des Weiteren ist dem Fall das CaseFileItem *Bewerberdaten* angehängt. In diesem werden alle bewerberrelevanten Daten gespeichert. Auf dieses Objekt kann jeder Bearbeiter lesend zugreifen, der Rechte für diesen Fall besitzt. Hat ein Bearbeiter Schreibrechte, ist es zu jedem Zeitpunkt im Prozess möglich, das CaseFileItem zu bearbeiten. So kann beispielsweise bei einem Umzug des Bewerbers die Adresse angepasst werden oder weitere Bewerbungsunterlagen hinzugefügt werden.
Bei den Verbindungen der Aktivitäten handelt es sich nicht um einen Kontrollfluss, son-

dern es werden Abhängigkeiten zwischen den Aktivitäten verdeutlicht. Somit ist der Punkt *Gutachten erstellen* erst möglich, wenn die Bewerbung erfolgreich geprüft wurde. Bei dieser Aktivität handelt es sich um einen *non-blocking Human Task*, der mehrmals ausgeführt werden darf. Somit ist es asynchron möglich, mehrere Gutachten gleichzeitig zu erstellen. Dabei kann die Aktivität wiederholt werden.
Nach einer vom Benutzer festgelegten Zeit werden die Gutachten von einem Bearbeiter geprüft (*Gutachten prüfen*) und entweder eine Absage erteilt oder eine *Einladung zu einem Bewerbungsgespräch* verschickt. Weiterhin kann jedoch von einem Bearbeiter entschieden werden, die Bewerbung mit anderen offenen Stellen im Unternehmen zu vergleichen. Dann wird zur Laufzeit der Discretionary Task *Prüfung mit anderen Stellenangeboten* in den Prozessablauf miteingebunden und kann ausgeführt werden. Bei dieser Aktivität handelt es sich um einen Process Task. Somit könnte hier ein BPMN Prozess eingebunden werden, der diese Aufgabe, mit Hilfe eines prozessorientierten Ablaufs, bearbeitet. Kommt es zu einem positiv verlaufenden Bewerbungsgespräch, wird der Bewerber eingestellt und der Prozess kann geschlossen werden.

Der Bewerbungsprozess einer Personalabteilung ist sehr gut geeignet, um die Vorteile der verschiedenen Beschreibungssprachen aufzuzeigen. So sind Teile des Ablaufs einfacher und übersichtlicher unter BPMN modellierbar. Jedoch gibt es Bereiche, in denen eine flexiblere Ausrichtung mit CMMN den Personalbearbeiter besser unterstützt und mehr Möglichkeiten bei der Bearbeitung des Falls bietet. Ein genauer Vergleich wird im nächsten Abschnitt vorgenommen.

4.5.4 Vergleich von BPMN und CMMN

Ein direkter Vergleich der beiden Prozessbeschreibungssprachen ist ein sehr ungleicher Vergleich. BPMN ist in der ersten Version bereits im Jahr 2004 entstanden, womit bis zu Veröffentlichung von CMMN im Jahr 2014 ein langer Zeiraum liegt, indem viele Forschungsansätze auf diesem Gebiet erzieht wurden. Damit soll verdeutlicht werden, dass die Ausrichtung von BPMN durch die Modellierung von starkstrukturierten sich wiederholenden Prozessen eine ganz andere darstellt, als CMMN. Das Ziel von CMMN war dagegen die Entwicklung einer Sprache, welche wissensintensive und struktur-

schwache, sowie eventgesteuerte Prozesse unterstützt. [58] Trotzdem gibt es seit der aktualisierten Versionen BPMN 2.0 einige Weiterentwicklungen, mit denen versucht wird, auf die heutigen Anforderungen vieler Geschäftsprozesse einzugehen.
Ein Beispiele hierfür sind die neu eingeführten Adhoc Subprozesse in BPMN um mehr Flexibilität in einem prozessorientierten PrMS zu erhalten. Diese werden in einem Subprozesscontainer modelliert, welches ein Tilden Symbol enthält. In diesem untergeordneten Prozess dürfen Tasks und Subprozesse frei angeordnet werden. Das bedeutet, dass kein Kontrollfluss modelliert wird, welcher eine Ausführungsreihenfolge vorgibt. Somit ist eine freie Ausführung der Aktivitäten, basierend auf den Entscheidungen eines Bearbeiters, möglich. Jedoch können dabei keine Regeln integriert werden, so dass dieser Subprozess komplett von einem Bearbeiter ausgeführt werden muss. CMMN integriert durch seine deklarative Ausrichtung Regeln, so dass Tasks modelliert werden können, die auch automatisiert ausgeführt werden können.
Ein weiteres Beispiel stellt der Eventsubprozess dar. Diese können innerhalb eines Subprozesse modelliert werden und dienen der Verarbeitung von auftretenden Events. Auch dieser Prozess liegt außerhalb des Kontrollflusses. Mit dessen Hilfe ist es möglich den Kontrollfluss des eigentlichen untergeordneten Prozesses zu unterbrechen oder den Subprozess parallel weiter auszuführen. Zusätzlich zu diesen beiden Beispielen ist es in einigen Fällen auch möglich, durch eine komplexere Modellierung gewisse Aufgaben in BPMN umzusetzen. Allgemein ist es mit BPMN sehr gut möglich große und komplexe Aufgabenstellungen zu modellieren. Was einen Grund für den weit verbreiteten Einsatz von BPMN in Unternehmen darstellt. Inwieweit dies mit CMMN möglich ist wird sich in der praktischen Anwendung mit fallbasierenden PrMS zeigen müssen. Jedoch wird ein CMMN-Modell bei einer großen Prozessumsetzung sehr schnell an Übersichtlich verlieren. Durch eine klare Struktur ist ein kontrollflussgesteuerter Prozess hinsichtlich der Lesbarkeit deutlich überlegen.
In Bezug auf das Einsatzgebiet der beiden Modelliersprachen wurde der unterscheidliche Strukturierungsgrad bereits erwähnt. Ein Wissensarbeiter ist selbst für die Ausführung eines Prozesses verantwortlich, wohingegen der Ablauf eines Bearbeiter in strukturierten, fest vormodellierten Prozesses komplett kontrolliert werden kann. Dieser muss auch nicht mit unvorhergesehen Anforderungen umgehen, sondern bearbeitet einen

weitestgehend gleichbleibenden Prozess. Hierdurch wird der Unterschied zwischen einer imperativen und deklarativen Sprache am besten deutlich. Damit werden die vorhanden Unterschiede von BPMN und CMMN deutlich.

Ein Unterschied wird auch bei der Untersuchung der Komplexität der beiden Sprachen deutlich. [75] Dabei wurde der CMMN-Standard mit der BPMN-Beschreibungssprache verglichen. Dabei wird gezeigt, dass BPMN komplexer ist als CMMN, damit auch schwerer zu lernen und benutzerunfreundlicher.
Ein weiterer Punkt ist dem Alter der BPMN-Spezifikation geschuldet. In der CMMN Spezifikation wird eine Integration von BPMN definiert. Anders herum sollte das genaus möglich sein, ist bis zum jetztigen Zeitpunkt jedoch noch nicht offiziell von der OMG spezifiziert.

Untersucht man die beiden unterschiedlichen Modelle des vorgestellten Bewerbungsbeispiels, stellt man eine unterschiedliche Ausrichtung der beiden Sprachen fest. In CMMN wird der Wissensarbeiter in den Vordergrund gestellt und der Prozess so modelliert, dass eine höchst mögliche Flexibilität gewährleistet ist. Wobei der Bearbeiter trotzdem durch das Systems unterstützt wird. So modelliert das System die Abhängigkeiten der Aktivitäten und zeigt, welche Aufgaben bearbeitet werden können und müssen. Weiterhin kann der Bearbeiter eigenständig auf die Daten zugreifen und diese, genauso wie die Anzahl der Gutachten, jederzeit anpassen. Zusätzlich ist zu jedem Zeitpunkt möglich den Fall abzubrechen, wenn dazu ein Grund besteht, so dass der Prozess nicht bis zum Ende ausgeführt werden muss.
Das mit BPMN modellierte Beispiel hingegen orientiert sich an einem festen Kontrollfluss. Hier ist keine flexible Anpassung des Prozessablaufes möglich, da dieser komplett zur Entwurfszeit fest modelliert wurde. Jedoch handelt es sich bei einer Stellenauschreibung von Grund auf, um einen sehr fest vorgegebenen Prozess. Dies sieht man auch im CMMN-Beispiel, bei welchem die Aktivitäten mit vielen nacheinander aufgereihten Abhängigkeiten modelliert werden müssen. So wird der Bearbeiter durch die feste Routine entlastet und muss sich nicht für jeden Fall eine neue Ausführungsreihenfolge zusam-

menstellen. Jedoch kann die starre Reihenfolge zu längeren Ausführungszeiten führen, worunter die Effizienz leidet.

Ein Fall sollte daher nicht nur zur Entwurfszeit definiert werden, sondern eben auch die Möglichkeit besitzen, dass auch ein Benutzer ohne tiefes technisches Verständnis zur Laufzeit Anpassungen vornehmen kann. So definiert das CM die Idee, um während der Ausführung auf unvorhergesehene Situationen einzugehen. Jedoch bietet CMMN keine so weitreichende Unterstützung von Ad-hoc Prozessen. So können wie bereits erwähnt bestimmte Aktivitäten zur Laufzeit angepasst oder hinzugefügt werden, jedoch müssen diese zur Modellierzeit bekannt gewesen und vormodelliert worden sein. Somit ist auch in einem CMMN modellierten Prozess der Grad der Flexibilität, der umgesetzt werden kann beschränkt.

4.6 Diskussion

Mit der Einführung von CMMN ist eine rege Diskussion über den Einfluss von CM entstanden. Diese ging bis zu der Frage, ob man BPMN oder prozessorientierte Systeme überhaupt noch benötigt in einer Zeit, in der die Zahl an wissensintensiven Prozessen immer mehr ansteigt. [76]
Ein weiterer Beitrag beschreibt den Einsatz von mehreren Systemen in Unternehmen, um fallbasierte Prozesse umzusetzen. Hierbei wird ein erweitertes CM-System gefordert, um den veränderten Bedürfnissen heutiger moderner Unternehmen gerecht zu werden. [37]
Dabei werden Wirtschaftsbereiche, wie beispielsweise Produktion, Fertigung und auch andere Industriezweige ignoriert, in denen dieser feste Ablauf nicht als Nachteil bewertet werden darf. Die bereits diskutierten Probleme und Grenzen bei starren Strukturen können in klassischen BPM-Systemen optimal zur Modellierung, Ausführung, Optimierung und Überwachung von Geschäftsprozessen eingesetzt werden.
Die Hauptfaktoren, mit denen man über die Modellierungssprache CMMN oder BPMN entscheidet, sollten daran festgemacht werden, wie viele unvorhergesehene Prozesse bei der Bearbeitung eines Geschäftsprozesses auftreten und wie viele Wissensarbeiter

in einem Unternehmen beschäftigt oder benötigt werden.
Bruce Silver diskutiert in seinem Blog die Gemeinsamkeiten von CMMN und BPMN. Dabei schildert er, dass die Grundlagen von CMMN bereits in BPMN 2.0 umgesetzt sind und plädiert für eine Erweiterung des BPMN-Standards.
Hierbei handelt es sich nur um eine Meinung, in der eine Verknüpfung der beiden Standards als der richtige Schritt dargestellt wird. Jedoch dürfte ein Grund für die separate Veröffentlichung eines neuen Standards auch daran gelegen haben, dass die Einführung einer neuen Spezifikation leichter umgesetzt werden kann, als die Anpassung eines bereits existierenden Standards.
Unabhängig davon, können bei einer Kombination der beiden Sprachen die Vorteile beider Standards in einem Prozessmanagementsystem implementiert werden. Dies hat die *BPM Plattform camunda* versucht. In der neusten Version wurde die CMMN-Spezifikation zusätzlich zum vorhanden BPMN-Standard mit in das Framework implementiert. [77]

5

Zusammenfassung und Aussicht

Aus der Studie von BPM&0 und BearingPoint geht hervor, dass ein großer Teil der befragten Unternehmen davon ausgeht, dass die Wichtigkeit von BPM in den nächsten Jahren zunehmen wird. [1]

Dies zeigt auch die Vielzahl an Forschungen in diesem Bereich. Dabei wurde in dieser Arbeit anhand einer Literaturrecherche die Entwicklung vom datenzentrierten Prozessmanagement hin zum objektorientierten Framework PHILharmonicFlows aufgezeigt. Dazu wurden der artefaktzentrierte und der fallbasierte Ansatz, sowie der deklarative und imperative Ansatz beschrieben und gegen Herausforderungen evaluiert, die für ein gesamtheitliches datenzentriertes System erforderlich sind. Weiterhin wurde in der Arbeit der CMMN-Standard vorgestellt und dabei Eigenschaften herausgearbeitet, die für eine datenzentrierte Notation entscheidend sind.
Dafür wurde die Notation mit der prozessorientierte Sprache BPMN verglichen, um die

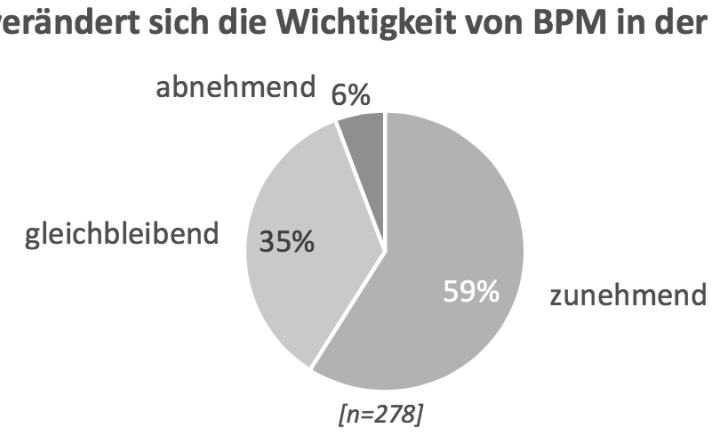


Abbildung 5.1: Studie von BPM&0 und BearingPoint, [1]

Unterschiede zwischen einer starkstrukturierten prozessgesteuerten und einer schwachstrukturierten datengesteuerten Notation zu verdeutlichen. Dabei zeigten sich die Vorteile von CMMN bei der Modellierung von wissens-intensiven Prozessen. Mit CMMN lässt sich zur Laufzeit sehr flexibel die Ausführungsreihenfolge bestimmen und es ist möglich, diese sehr individuell den Bedürfnissen der Benutzer anzupassen.

Literaturverzeichnis

[1] BPM&OGmbH, BearingPoint: Business process management- studie 2015: Messbare verbesserung der leistungsfähigkeit durch prozessmanagement. (2015)

[2] Müller, S.: Situationsanalyse: Bpm in deutschland. (2014)

[3] Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer-Verlag Berlin Heidelberg (2007)

[4] Oracle: (Oracle business process management suite - http://www.oracle.com/de/-technologies/bpm/overview/index.html) Zuletzt geöffnet am 19.11.2015.

[5] IBM: (Ibm websphere - http://www.ibm.com/websphere) Zuletzt geöffnet am 19.11.2015.

[6] AristaFlow: (Arsitaflow bpm suite - http://www.aristaflow.com/de/bpm-suite/ueberblick.html) Zuletzt geöffnet am 19.11.2015.

[7] OMG: (Business process model and notation (bpmn), version 2.0 - http://www.omg.org/spec/bpmn/2.0/) Zuletzt geöffnet am 19.11.2015.

[8] Künzle, V., Reichert, M.: Integrating users in object-aware process management systems: Issues and challenges. In Rinderle-Ma, S., Sadiq, S., Leymann, F., eds.: Business Process Management Workshops. Volume 43 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2010) 29–41

[9] Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches. Journal on Data Semantics **4** (2015) 29–57

[10] OMG: (Case management model and notation (cmmn), version 1.0 - http://www.omg.org/spec/cmmn/1.0) Zuletzt geöffnet am 29.11.2015.

[11] Georgakopoulos, D., Hornick, M., Sheth, A.: An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and Parallel Databases **3** (1995) 119–153

[12] Künzle, V., Reichert, M.: Philharmonicflows: Research and design methodology. Technical Report UIB-2011-05, University of Ulm, University of Ulm (2011)

[13] Scheithauer, G., Hellmann, S.: Analysis and documentation of knowledge-intensive processes. In: Business Process Management Workshops, Springer (2013) 3–11

[14] Vaculin, R., Hull, R., Heath, T., Cochran, C., Nigam, A., Sukaviriya, P.: Declarative business artifact centric modeling of decision and knowledge intensive business processes. In: Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International, IEEE (2011) 151–160

[15] White, M.: Case management: Combining knowledge with process. (2009)

[16] Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.A.: Imperative versus declarative process modeling languages: An empirical investigation. In: Business Process Management Workshops, Springer (2012) 383–394

[17] Lloyd, J.W.: Practical advtanages of declarative programming. In: 1994 Joint Conference on Declarative Programming, GULP-PRODE'94. (1994)

[18] van Der Aalst, W.M., Ter Hofstede, A.H., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and parallel databases **14** (2003) 5–51

[19] Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. Data and Knowledge Engineering **66** (2008) 438–466

[20] Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S.: Declarative versus imperative process modeling languages: The issue of understandability. In Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R., eds.: Enterprise, Business-Process and Information Systems

Modeling. Volume 29 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2009) 353–366

[21] Wang, Y., Huang, L., Guo, Y.: Integrating declarative and imperative approach to model logistics service processes. Journal of Industrial Engineering and Management **6** (2013) 237–248

[22] Pesic, M., van der Aalst, W.: A declarative approach for flexible business processes management. In Eder, J., Dustdar, S., eds.: Business Process Management Workshops. Volume 4103 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) 169–180

[23] Pesic, M., Schonenberg, H., Van der Aalst, W.M.: Declare: Full support for loosely-structured processes. In: Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International, IEEE (2007) 287–287

[24] Westergaard, M., Maggi, F.M.: Declare: A tool suite for declarative workflow modeling and enactment. BPM (Demos) **820** (2011)

[25] Pesic, M.: Constraint-based workflow management systems: Shifting control to users. (2008)

[26] Künzle, V., Reichert, M.: Herausforderungen bei der integration von benutzern in datenorientierten prozess-management-systemen. EMISA Forum **30** (2010) 11–28

[27] Nigam, A., Caswell, N.: Business artifacts: An approach to operational specification. IBM Systems Journal **42** (2003) 428–445

[28] Hull, R.: Artifact-centric business process models: Brief survey of research results and challenges. In Meersman, R., Tari, Z., eds.: On the Move to Meaningful Internet Systems: OTM 2008. Volume 5332 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2008) 1152–1163

[29] Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering **32** (2009) 3–9

[30] Marin, M., Hull, R., Vaculín, R.: Data centric bpm and the emerging case management standard: A short survey. In La Rosa, M., Soffer, P., eds.: Business Process Management Workshops. Volume 132 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2013) 24–30

[31] Ngamakeur, K., Yongchareon, S., Liu, C.: A framework for realizing artifact-centric business processes in service-oriented architecture. In Lee, S.g., Peng, Z., Zhou, X., Moon, Y.S., Unland, R., Yoo, J., eds.: Database Systems for Advanced Applications. Volume 7238 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 63–78

[32] Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath III, F.T., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P., et al.: Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In: Web services and formal methods. Springer (2011) 1–24

[33] Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of gsm-based artifact-centric systems through finite abstraction. In Liu, C., Ludwig, H., Toumani, F., Yu, Q., eds.: Service-Oriented Computing. Volume 7636 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 17–31

[34] Solomakhin, D., Montali, M., Tessaris, S., De Masellis, R.: Verification of artifact-centric systems: Decidability and modeling issues. In: Service-Oriented Computing. Springer (2013) 252–266

[35] de Man, H.: Case management: A review of modeling approaches. (2009)

[36] Mutschler, B., Weber, B., Reichert, M.: Workflow management versus case handling: results from a controlled software experiment. In: Proceedings of the 2008 ACM symposium on Applied computing, ACM (2008) 82–89

[37] Burns, E.V.: Case management 101: 10 things you must know about case management. In: Taming the Unpredictable, Real World Adaptive Case Management: Case Studies and Practical Guidance, Future Strategies Inc (2011) 17–25

[38] van der Aalst, W.M., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowledge Engineering **53** (2005) 129 – 162

[39] Andrews, K., Steinau, S., Reichert, M.: A runtime environment for object-aware processes. In: Proceedings of the BPM Demo Session 2015 (BPMD 2015), co-located with the 13th International Conference on Business Process Management (BPM 2015). Number 1418 in CEUR Workshop Proceedings, CEUR-WS.org (2015) 6–10

[40] Künzle, V., Reichert, M.: Philharmonicflows: towards a framework for object-aware process management. Journal of Software Maintenance and Evolution: Research and Practice **23** (2011) 205–244

[41] Kreher, U.: Konzepte, Architektur und Implementierung adaptiver Prozessmanagementsysteme. PhD thesis, University of Ulm (2014)

[42] Marrella, A., Mecella, M., Russo, A., Steinau, S., Andrews, K., Reichert, M.: A survey on handling data in business process models (discussion paper). In: 23rd Italian Symposium on Advanced Database Systems (SEBD). (2015)

[43] Künzle, V.: Object-Aware Process Management. PhD thesis, University of Ulm (2013)

[44] Künzle, V., Weber, B., Reichert, M.: Object-aware business processes: Fundamental requirements and their support in existing approaches. International Journal of Information System Modeling and Design (IJISMD) **2** (2011) 19–46

[45] Künzle, V., Reichert, M.: Towards object-aware process management systems: Issues, challenges, benefits. In: Proc. 10th Int'l Workshop on Business Process Modeling, Development, and Support (BPMDS'09). Number 29 in LNBIP, Springer (2009) 197–210

[46] Reichert, M., Weber, B.: User- and data-driven processes. In: Enabling Flexibility in Process-Aware Information Systems. Springer Berlin Heidelberg (2012) 377–403

[47] Künzle, V., Reichert, M.: Striving for object-aware process support: How existing approaches fit together. In: 1st Int'l Symposium on Data-driven Process Discovery and Analysis (SIMPDA'11). Number 116 in LNBIP, Springer (2012) 169–188

[48] Chiao, C.M., Künzle, V., Reichert, M.: Enhancing the case handling paradigm to support object-aware processes. In: 3rd Int'l Symposium on Data-Driven Pro-

cess Discovery and Analysis (SIMPDA'13). Number 1027 in CEUR Workshop Proceedings, CEUR-WS.org (2013) 89–103

[49] Becker, J., Probandt, W., Vering, O.: Modellierungssprachen. In: Grundsätze ordnungsmäSSiger Modellierung. BPM kompetent. Springer Berlin Heidelberg (2012) 4–30

[50] Gadatsch, A.: Grundlegende begriffe. In: Grundkurs Geschäftsprozess-Management. Methoden und Werkzeuge für die IT-Praxis: eine Einführung für Studenten und Praktiker. 5 edn. (2008) 1–72

[51] van der Aalst, W., ter Hofstede, A., Weske, M.: Business process management: A survey. In van der Aalst, W., Weske, M., eds.: Business Process Management. Volume 2678 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2003) 1–12

[52] Frank, U., van Laak Bodo: Anforderungen an sprachen zur modellierung von geschäftsprozessen. Arbeitsberichte des Instituts für Wirtschaftsinformatik 34 (2003)

[53] Rosenfeld, A.: Bpm: Structured vs. unstructured. (2011)

[54] Dadam, P., Reichert, M., Rinderle-Ma, S.: Prozessmanagementsysteme: Nur ein wenig flexibilität wird nicht reichen. Informatik-Spektrum **34** (2011) 364–376

[55] Hauder, M., Munch, D., Michel, F., Utz, A., Matthes, F.: Examining adaptive case management to support processes for enterprise architecture management. In: Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International, IEEE (2014) 23–32

[56] Schonenberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W.M.: Towards a taxonomy of process flexibility. In: CAiSE forum. Volume 344. (2008) 81–84

[57] van der Aalst, W.M., Jablonski, S.: Dealing with workflow change: identification of issues and solutions. Computer systems science and engineering **15** (2000) 267–276

[58] Breitenmoser, R., Keller, T.: Case management model and notation-a showcase. European Scientific Journal, ESJ **11** (2015)

[59] Chinosi, M., Trombetta, A.: Bpmn: An introduction to the standard. Computer Standards Interfaces **34** (2012) 124 – 134

[60] Freund, J., Rücker, B.: Praxishandbuch BPMN 2.0. 4. aktualisierte auflage edn. Carl Hanser Verlag GmbH Co KG (2014)

[61] Trisotech: (Trisotech, digital enterprise suite - http://www.trisotech.com/) Zuletzt geöffnet am 19.12.2015.

[62] Herrmann, C., Kurz, M.: Adaptive case management: Supporting knowledge intensive processes with it systems. In Schmidt, W., ed.: S-BPM ONE - Learning by Doing - Doing by Learning. Volume 213 of Communications in Computer and Information Science. Springer Berlin Heidelberg (2011) 80–97

[63] Buchanan, B.G., Shortliffe, E.H.: Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley Series in Artificial Intelligence). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1984)

[64] CMSA: (Case managment society of america - http://www.cmsa.org/) Zuletzt geöffnet am 19.12.2015.

[65] Motahari-Nezhad, H., Swenson, K.: Adaptive case management: Overview and research challenges. In: Business Informatics (CBI), 2013 IEEE 15th Conference on. (2013) 264–269

[66] Koehler, J.: The role of bpmn in a modeling methodology for dynamic process solutions. In Mendling, J., Weidlich, M., Weske, M., eds.: Business Process Modeling Notation. Volume 67 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2010) 46–62

[67] White, M.: Case management: Combining knowledge with process. (2009)

[68] Group, O.M.: (Case management process modeling (cmpm) - http://www.omg.org/cgi-bin/doc?bmi/2009-09-23) Zuletzt geöffnet am 19.12.2015.

[69] Hull, R., Damaggio, E., De Masellis, R., Fournier, F., Gupta, M., Heath, III, F.T., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P.N., Vaculin, R.: Business artifacts with guard-stage-milestone lifecycles: Managing artifact interactions with conditions and events. In: Proceedings of the 5th ACM International Conference on Distributed Event-based System. DEBS '11, New York, NY, USA, ACM (2011) 51–62

[70] Hull, R., Llirbat, F., Siman, E., Su, J., Dong, G., Kumar, B., Zhou, G.: Declarative workflows that support easy modification and dynamic browsing. In: ACM SIGSOFT Software Engineering Notes. Volume 24., ACM (1999) 69–78

[71] Kidd, A.: The marks are on the knowledge worker. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '94, New York, NY, USA, ACM (1994) 186–191

[72] van der Aalst, W.M.P., Berens, P.J.S.: Beyond workflow management: Product-driven case handling. In: Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work. GROUP '01, New York, NY, USA, ACM (2001) 42–51

[73] Reijers, H.A., Rigter, J.H.M., van der Aalst, W.M.P.: The case handling case. International Journal of Cooperative Information Systems **12** (2003) 365–391

[74] van der Aalst, W., Stoffele, M., Wamelink, J.: Case handling in construction. Automation in Construction **12** (2003) 303–320

[75] Marin, M.A., Lotriet, H., Van Der Poll, J.A.: Measuring method complexity of the case management modeling and notation (cmmn). In: Proceedings of the Southern African Institute for Computer Scientist and Information Technologists Annual Conference 2014 on SAICSIT 2014 Empowered by Technology, ACM (2014) 209

[76] Swenson, K.: Position: Bpmn is incompatible with acm. In La Rosa, M., Soffer, P., eds.: Business Process Management Workshops. Volume 132 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2013) 55–58

[77] Silver, B.: (Business process watch, - http://brsilver.com/bpmn-cmmn-compared/) Zuletzt geöffnet am 19.12.2015.

Abbildungsverzeichnis

CMMN Case Management Model and Notation

BPMN Business Process Management and Notation

BPM Business Prozess Management

PrMS Process Management System

GSM Guard-Stage-Milestone

IT Informations-Technologie

CRM Customer Relationship Management

ERP Enterprise Resource Planning

CM Case Management

OMG Object Management Group

KiP Knowledge-intensive Processes

CH Case Handling

DBIS Institut für Datenbanken- und Informationssysteme

on the fly während der Ausführung

ECA Event-Condition-Action

FSM endlichen Zustandsmaschine (finite state machine)

DMN Decision Model and Notation

CMPM The Case Management Process Modeling

Name: Mark Alexander Popa Matrikelnummer: 631535

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Mark Alexander Popa