Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Evaluation der Modellierungskonzepte eines objektzentrierten Prozessmanagementsystems

Masterarbeit an der Universität Ulm

Vorgelegt von:
David Rothmaier
david.rothmaier@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert
Dr. Vera Künzle

Betreuer:
Sebastian Steinau

2017

Fassung 12. Juni 2017

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Die Bedeutung des Bereichs Prozessmanagement, einem Teilgebiet der Informatik, ist bereits sehr groß und wird in den kommenden Jahren in der Wissenschaft, wie auch in der Industrie, weiter an Bedeutung gewinnen. Dies kommt unter anderem daher, dass immer mehr Daten erfasst und aufbereitet werden müssen. Aufbereitete Informationen sind heutzutage ein wichtiger Schlüsselfaktor für unternehmerischen Erfolg. Prozesse in Unternehmen sind auf korrekt ausgewertete und passgenau aufbereitete Informationen angewiesen. Hierdurch steigen die Anforderungen an Systeme, welche Prozesse digital unterstützen, in Bezug auf Funktionalität und Qualität der Datenverarbeitung stetig an. Somit müssen Prozesse immer mehr auf eine flexible Art mit Daten umgehen können.

Um den wachsenden Anforderungen in der Prozessunterstützung gerecht zu werden, wurden in der Vergangenheit die klassischen Prozessmanagementsysteme entwickelt, welche inzwischen zur Geschäftsprozessunterstützung eingesetzt werden. Diese Systeme arbeiten großteils aktivitätenzentriert. Dies bedeutet, dass das Fortschreiten des Prozesses von auszuführenden Aktivitäten abhängig ist. Daher orientiert sich die Prozesslogik sehr an den im Prozess stattfindenden Aktivitäten. Es existiert aber keine ausreichende Datenperspektive für viele realen Prozesse, datenintensive Prozesse können nur unbefriedigend umgesetzt werden. Auf diesem Hintergrund entstanden immer mehr Ansätze, datenzentrierte Prozessmanagementsysteme zu entwickeln, welche von Aktivitäten abstrahieren und die Daten des Prozesses in den Fokus stellen, um somit einen flexiblen Umgang mit Daten zu ermöglichen.

Einer der Ansätze ist der objektzentrierte Ansatz, welcher von der Implementierung PHILharmonicFlows welche von der Universität Ulm entwickelt wird, umgesetzt wird. Ein objektzentrierter Ansatz stellt eine spezielle Form des datenzentrierten Paradigmas dar. Hierbei werden Daten in Objekten organisiert, mit welchen dann Prozesse realisiert werden.

Diese Arbeit evaluiert die Fähigkeiten der Modellierungsumgebung von PHILharmonicFlows anhand von Prozessen aus einer realen Anwendung. Als Maßstab diente ein Anforderungsprofil der Persis GmbH zu einer Umsetzung eines neuen Moduls in der

Persis Unternehmenssoftware. Das System wurde unter den Aspekten der Umsetzbarkeit der Anforderungen der Persis GmbH evaluiert. Auf Basis dieser Auswertung wurden Schwächen und Stärken von PHILharmonicFlows aufgezeigt sowie Weiterentwicklungsvorschläge für das System ausgearbeitet.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Erstellung dieser Masterarbeit unterstützt und motiviert haben.

Ein besonderer Dank gilt an dieser Stelle Sebastian Steinau, der meine Arbeit und somit auch mich betreut hat. Er stand mir stets für Fragen zur Verfügung und gab mir einige wertvolle Hinweise. Weiter fand er und Kevin Andrews immer sofort Zeit, Softwarefehler der Modellierungsumgebung von Philharmonic Flows zu beheben. Auch wenn diese am Wochenende bekannt wurden, erhielt ich eine schnelle Antwort. Vielen Dank für die Geduld und Mühen.

Daneben gilt mein Dank Herrn Ulrich Riegger und meinem Vater Reinhold Rothmaier welche in zahlreichen Stunden Korrektur gelesen haben. Sie wiesen auf Schwächen hin und konnten als Fachfremde immer wieder zeigen, an welchen Stellen noch Erklärungsbedarf bestand.

Nicht zuletzt gebührt meinen Eltern Dank, da Sie mich während des Studiums nicht nur finanziell, sondern vor allem auch emotional immer unterstützt haben!

Inhaltsverzeichnis

1 Einleitung

Dieses Kapitel legt die Problemstellung und Zielsetzung dieser Arbeit fest. Weiter wird der Aufbau der Arbeit erläutert.

1.1 Motivation

Unternehmen setzen zunehmend Prozessmanagementsysteme (PrMS) ein, die vielversprechende Perspektiven für eine flexiblere und effizientere Prozessdurchführung bieten [4]. Für viele prozessbezogene Anwendungssysteme (z.B. ERP- oder CRM-Systeme) ist die zugrunde liegende Prozesslogik jedoch noch manuell in Programmcode umgesetzt. Infolgedessen sind diese Anwendungen sowohl komplex zu erstellen als auch teuer zu warten. Es sind somit lange Entwicklungszyklen nötig und sogar einfache Prozeßänderungen können zu kostspieligen Codeanpassungen und hohem Aufwand beim Testen führen [26], [20]. Dieses Problem wird noch massiv verstärkt, wenn eine Anwendung älter wird und sich verwendete Programmiersprachen oder Frameworks gewandelt haben [15].

Ein Hauptproblem hierbei ist, dass klassische Prozessmanagementsysteme primär zur Unterstützung von strukturierten, sich wiederholenden Geschäftsprozessen entwickelt wurden. Diese Eigenschaft trifft jedoch nur auf eine Teilmenge von realen Prozessen in Unternehmen zu. Reale Prozesse sind oftmals nur teilweise strukturiert. Ein weiterer Teil realer Prozesse ist durch Benutzerentscheidungen gesteuert [19]. Somit können Geschäftsfunktionen dieser Prozesse oft nicht direkt in Aktivitäten einbezogen werden. [15].

Aus diesen Gründen ist das aktivitätsorientierte Paradigma generell zu unflexibel für die Realisierung fortgeschrittener Geschäftsanwendungen. Dieser Mangel ergibt sich vor allem aus der unbefriedigenden Integration von Daten in bestehende Prozessmanagementsysteme [15].

Mit genau diesen Herausforderungen ist auch die Firma Persis bei der Weiter-, bzw. Neuentwicklung ihrer Human Ressource Software konfrontiert. Persis unterstützt daher das Projekt PHILharmonicFlows, welches versucht einen datenzentrierten Ansatz eines Prozessmanagementsystemes umzusetzen, in der Hoffnung mit PHILharmonicFlows zukünftig ein System zu erhalten, welches die Datensicht in das Prozessmanagementsystem integriert und ein bessere Wartbarkeit von Benutzerformularen ermöglicht.

Diese Arbeit beschäftigt sich mit dem Versuch, Prozesse mit PHILharmonicFlows zu modellieren welche den Anforderungen der Persis GmBH entsprechen. Anhand dieser Umsetzungen wird das Konzept von PHILharmonic Flows anhand eines konkreten Bewertungsschemas evaluiert. Hierdurch werden konkrete Stärken und Schwächen von PhilharmonicFlows aufgezeigt. Aus den aufgezeigten Defiziten enstehen konkrete Weiterentwicklungsvorschläge, welche durch das Entwicklerteam von PHILharmonicFlows teilweise direkt in PHILharmonicFlows eingearbeitet wurden oder in Zukunft umgesetzt werden können.

1.2 Zielsetzung

Die Persis GmbH ist momentan mit einer Erweiterung der Module ihrer Unternehmenssoftware beschäftigt. Im Zuge dieser Erweiterung wurden von der Persis GmbH Anforderungen für ein neues Modul Employee-Self-Service mit Fehlzeitenverwaltung ausgearbeitet. Da die Persis GmbH interessiert daran ist, zukünftig Neuentwicklungen mit der Technologie von PHILharmonicFlows durchzuführen soll untersucht werden, inwieweit sich das Konzept von PHILharmonikFlows zur Umsetzung von Prozessen, welche den Anforderungen der Persis GmbH entsprechen, eignet. Hierzu sollen die Prozesse Employee-Self-Service (ESS) und Fehlzeitenverwaltung (FZ) nach Spezifikationen der Firma Persis mit PHILharmonicFlows abgebildet werden. Um diese Umsetzung

auszuwerten wurde ein Bewertungsschema entwickelt, welches eine Beurteilung der Fähigkeiten zur Umsetzung der Anforderungen der Persis GmbH mit PHILharmonicFlows zulässt.

Die Evaluation soll zeigen, inwieweit sich die Anforderungen des ESS und der FZ umsetzen lassen, die Stärken und Schwächen von PHILharmonicFlows bei der Umsetzung aufzeigen und welche Ergänzungen zu PHILharmonicFlows zu einer anforderungsgerechten Umsetzung benötigt werden.

1.3 Aufbau der Arbeit

Kapitel 2 führt die zum Verständnis dieser Arbeit benötigten Grundlagen ein. In Kapitel 3 werden Anforderungen an die zu modellierenden Systeme präsentiert, anhand deren später die Modellierung erfolgen soll. Weiter dienen diese Anforderungen später in Kapitel 5 als Basis für die Konzeption des Bewertungsschemas der Modellierung. Kapitel 4 stellt die Modellierung des Employee-Self-Service (ESS) und der Fehlzeitenverwaltung (FZ) mit PHILharmonicFlows vor. Im Anschluss daran werden die Ergebnisse der Evaluation in Kapitel 5 diskutiert. In Kapitel 6 wird die Evaluation neu bewertet indem Ergänzungen zu PHILharmonikFlows welcher parallel zu dieser Arbeit entstanden sind, berücksichtigt werden. In Kapitel 7 wird ein Bezug zu verwandten Arbeiten hergestellt. Das abschließende Kapitel 8 enthält das Fazit und einen Ausblick.

2

Grundlagen

Dieses Kapitel soll eine kurze Einführung in die verwendete Technologie PHILharmonicFlows geben, sowie den Bezug von PHILharmonicFlows zur Firma Persis erläutern.

2.1 Prozessmanagementsysteme

Daten in Unternehmen werden zur Zeit meist mit spezifischen Werkzeugen, welche direkt in Programmcode realisiert werden, verarbeitet. Die durch diese Systeme unterstützten Prozesse sind häufig stark wissensintensiv und von Benutzereingaben abhängig. Die Datenspeicherung dieser Prozesse erfolgt in den meisten Fällen über eine relationale Datenbank. Ein großer Nachteil solcher Systeme ist jedoch die feste Verdrahtung von Prozesslogik im Anwendungscode, was zu langen Entwicklungszeiten und zu hohen Kosten bei Prozessänderungen führt. Weiterhin ist oftmals die Prozesslogik mehrfach redundant im Code enthalten. Dies führt zu einer hohen Fehleranfälligkeit und somit auch zu hohem Testaufwand bei Änderungen. Um dieser Problematik entgegenzusteuern, entstanden die so genannten Prozessmanagementsysteme (PrMS). Prozessmanagementsysteme führen bei der Entwicklung von Informationssystemen eine Abstraktionsebene ein, welche ein weitgehend vom Programmcode unabhängiges Modellieren von Prozessen erlaubt, indem eine Umsetzung in Prozesslogik stattfindet aus der automatisiert Programmcode generiert wird [8]. Prozess-Management-Systeme bieten somit generische Dienste, welche eine schnellere Entwicklung und eine bessere Wartung von Informationssystemen ermöglichen [18].

Die verbreiteten Prozessmanagementsystem arbeiten nach einem aktivitätenzentrierten Paradigma. Dies bedeutet, dass Geschäftsprozesse aus einer großen Menge an Aktivi-

täten bestehen, welche durch Kontrollflüsse gesteuert werden. Das zentrale Element eines aktivitätenzentrierten Prozessmodells ist somit die Aktivität. Dies veranschaulicht auch das Beispiel in der Abbildung 2.1. Zur Laufzeit wird für jeden Geschäftsvorfall eine eigene Instanz generiert. Jeder registrierte Benutzer erhält seine eigene Arbeitsliste in der aktivierte, vom Benutzer auszuführende Aktivitäten automatisch hinzugefügt werden. Über eine Auswahl in der Arbeitsliste wird die verknüpfte Geschäftsfunktion der Anwendung gestartet. Somit hilft ein PrMS bei der Trennung von Prozess- und Funktionslogik was die Kosten einer Anwendung senkt und für eine einfachere Wartung sorgt [15].

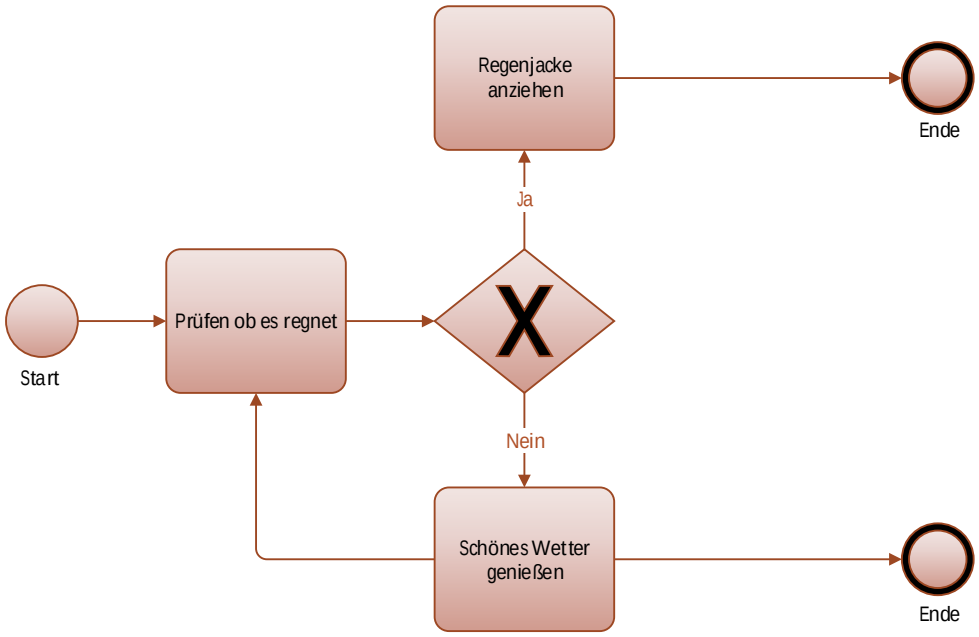

Abbildung 2.1: Beispiel eines aktivitätenzentrierten Prozessmodells

Jedoch konnten sich die aktivitätenzentrierten PrMS lange nicht in allen Bereichen durchsetzen [14]. Die Gründe hierfür sind, dass aktivitätenzentrierte Systeme für strukturierte, sich wiederholende Geschäftsprozesse entwickelt wurden, was aber oft nicht der Realität entspricht. Viele in der Praxis gefundene Prozesse sind eher unstrukturiert

oder halbstrukturiert, das heißt, sie sind wissensintensiv und durch Benutzerentscheidungen getrieben. Dies führt dazu, dass die zu integrierenden Geschäftsfunktionen in der Regel nicht direkt in die Aktivitäten umgesetzt werden können. Aus diesen Gründen ist das aktivitätsorientierte Paradigma, welches in zeitgenössischen PrMS verwendet wird, zu unflexibel für die Realisierung fortgeschrittener Geschäftsanwendungen [21]. Dies kommt vorallem aus der unbefriedigenden Integration von Prozessen und Daten in bestehende Prozessmanagementsysteme [15].

Genau diesen Herausforderungen ist auch die Firma Persis aus Heidenheim ausgesetzt, welche Human Resource Software (HRM) entwickelt, die stark wissensintensiv ist und von Formulareingaben durch Benutzer geprägt wird.

2.2 Persis

Die Firma Persis aus Heidenheim existiert seit dem Jahr 1987 und ist eines der führenden Unternehmen im Bereich High-Level Human Resource Software. Die Software von Persis unterstützt Prozesse im HRM Bereich in kleinen bis großen Unternehmen [1]. Persis bietet zwei unterschiedliche Arten, die Software zu beziehen, erstens Software as a Service, für Kunden ohne eigene Serverinfrastruktur. Die Software wird dann über eine Onlineschnittstelle verwendet. Die zweite Möglichkeit ist, die Software zu kaufen und über eine eigene Serverinfrastruktur zu betreiben. Die Persis Unternehmenssoftware besteht aus verschiedenen Modulen. Diese sind im einzelnen Bewerbermanagement, Personalmanager, digitale Personalakte, Personalentwicklung, Weiterbildung und ein Ausbildungsmanager. Die große Stärke der Persis Unternehmenssoftware ist, dass Persis Features bietet, die kein anderes vergleichbares Produkt in dem Umfang bieten kann. So kann fast alles an der Anwendung modifiziert und an Kundenwünsche angepasst werden. So können beispielsweise Formulare abgeändert werden (Hinzufügen von Feldern, Ausblenden von Feldern). Weiterhin ist es möglich, die Prozesse an den Kunden anzupassen oder der Anwendung ein komplett neues Layout zu geben, was der Persis GmbH ein Wettbewerbsvorteil gegenüber der Konkurrenz einräumt.

Die Persis Unternehmenssoftware besteht seit über 20 Jahren. Die Techniken hinter der Software gehen bis auf Java 1.1 zurück, die Prozesslogik der Software ist direkt im Code abgebildet. Diese direkte Umsetzung in Code und der Fakt, dass die Software seit über 20 Jahren weiterentwickelt wurde führen dazu, dass Änderungen am System immer aufwendiger werden. Die Schwierigkeiten bei Anpassungen sind somit, dass Technologien verwendet wurden, welche heute nicht mehr aktuell sind und daher ein großes Fachwissen an älteren Technologien benötigt wird. Ein noch größeres Problem ist aber die feste Kodierung von Prozesslogik in Programmcode. So ist bereits eine einfache Aufgabe wie z.B. das Hinzufügen eines Formularfeldes ein höchst komplexes Unterfangen, da hierzu erst eine neue Spalte in einer Datenbank zur Speicherung des Wertes generiert werden muss. Weiter muss nun das Feld im Formular angelegt werden. Dies führt dazu, dass jede Stelle der Anwendung, an der das Formular verwendet wird, angepasst werden muss. Dies ist ein höchst komplexes Unterfangen, da es durchaus vorkommen kann, dass Formulare mehrfach redundant verwaltet werden. Diese Abfolge von Abhängigkeiten zieht sich durch die ganze Software. Wird irgendwo bei einer Änderung eine Kleinigkeit übersehen, entsteht ein Fehler, welcher mühsam händisch im Programmcode gesucht werden muss. Aufgrund dieser Problematik muss jede Änderung durch aufwendige Tests überprüft werden.

Die Wartung und Anpassung ist somit zeit- und kostenintensiv für Persis. Da sich aktivitätenzentrierte PrMS aber auch nicht für eine Neuentwicklung der Persis Unternehmenssoftware eignen, ist Persis auf der Suche nach einer neuen Technologie, welche eine Lösung dieser Probleme verspricht. Die Firma Persis wurde somit auf einen neuen Ansatz von Prozessmanagementsystemen aufmerksam, welche nicht aktivitätenzentriert, sondern objektzentriert arbeiten. Die Persis GmbH entwickelt nun in Kooperation mit der Universität Ulm ein auf dem objektzentrierten Ansatz beruhendes System der Bezeichnung PHILharmonicFlows. Persis erhofft sich mit PHILharmonicFlows ein System zu erhalten, welches als generische Komponente die Persis Unternehmenssoftware bei zukünftigen Neuentwicklungen im Hintergrund unterstützt und generisch das Definieren von Formularen ermöglicht, sodass zukünftig benutzerdefinierte Formularanpassungen mit geringem Aufwand umsetzbar sind.

Persis arbeitet zurzeit an einer Neuentwicklung des Employee-Self-Services (ESS) sowie einer Fehlzeitenverwaltung (FZ) für die Persis HRM Software. Der Employee-Self-Service soll es Unternehmen ermöglichen, ihre Mitarbeiterdaten durch ihre Angestellten selbst verwalten zu lassen. Die Fehlzeitenverwaltung soll Fehlzeitenanträge von Mitarbeitern von der Antragsstellung bis zur genommenen Fehlzeit (Urlaub oder Abwesenheit im Unternehmen) unterstützen. Zur Neuentwicklung dieser Systeme wurden von Persis Anforderungen ausgearbeitet. Anhand dieser Anforderungen wird in dieser Arbeit versucht, die Systeme ESS und FZ mithilfe von PHILharmonicFlows abzubilden um bewerten zu können inwieweit sich das System zukünftig als generische Komponente für Neuentwicklungen dieser Art eignet.

2.3 Objektzentrierte Prozessmanagementsysteme

Das Objektzentriertes Paradigma bedeutet, dass die zentralen Elemente des Modells Objekte sind. Ein mögliches Objekt wird in Abbildung 2.2 dargestellt.

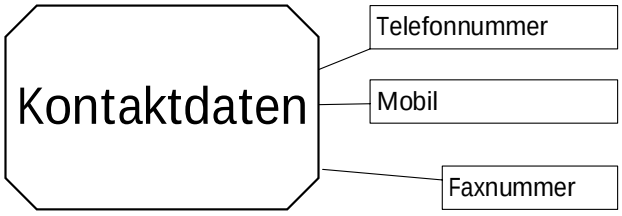

Abbildung 2.2: Beispiel eines Datenobjektes mit zugehörigen Attributen

Um Datenobjekte zu erhalten werden die Prozessdaten auf Attribute (z.B. Telefonnummer) heruntergebrochen. Diese Attribute werden dann strukturiert in Objekten (z.B. Kontaktdaten) gekapselt. Jedes dieser Objekte besitzt ein zugehöriges Objektverhalten, welches den Lebenszyklus eines Objektes als Prozess darstellt. Objekte modellieren direkt ausführbare Prozesse. Da ein einzelnes Datenobjekt wenig Sinn ergibt, kann es in einem Modell mehre Objekte geben. Diese müssen miteinander mithilfe von Koordinationsprozessen koordiniert werden.

Diese objektzentrierte Sicht auf Prozesse bietet viele Vorteile wenn viele Daten verarbeitet werden. Eine positive Eigenschaft objektzentrierter Prozesse wird in Abbildung 2.3 dargestellt. Hier ist ein Szenario mit zwei Formularen, drei Aktivitäten und drei Objekten abgebildet. Bei traditionellen aktivitätenorientierten Systemen müsste, um Aktivität A abzuschließen, X und Y vom Benutzer geschrieben werden. Im Falle eines objektzentrierten Prozesses ist nur das Schreiben von X notwendig, da nur das Objekt X von Aktivität A benötigt wird. Für Y ist es ausreichend, wenn dies zur Aktivität B geschrieben wird. Wenn jedoch gleichzeitig in Aktivität A, X und Y geschrieben werden, so kann Aktivität B automatisch übersprungen werden.

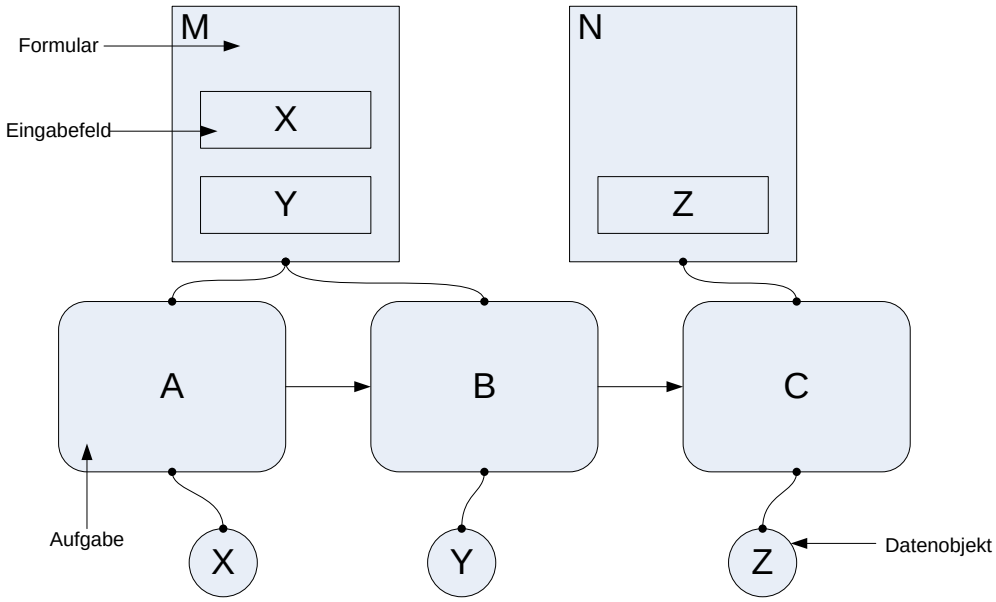

Abbildung 2.3: Beispiel Ablauf PAIS [23]

Ein konkreter Ansatz eines objektzentrierten Prozessmanagementsystemes wurde an der Universität Ulme mit Unterstützung der Firma Persis ausgearbeitet. Dieses System heißt PHILharmonicFlows und wird im kommenden Abschnitt vorgestellt.

2.4 PHILharmonicFlows: Eine Einführung

Ziel der Entwicklung von PHILharmonicFlows an der Universität Ulm ist es, ein objektzentriertes Prozess-Management-System zu schaffen, welches somit im Vergleich zu konventionellen aktivitätenbasierten Prozessmanagementsystemen von einem nach Aktivitäten orientiertem Ablauf abstrahiert und die Verarbeitung der Daten in den Mittelpunkt stellt. Um dies zu erreichen, wurde ein vierteiliges Konzept ausgearbeitet. Abbildung 2.4 zeigt den Aufbau des Konzeptes bestehend aus der Datenstruktur, den Mikroprozessen, den Makroprozessen sowie einem umfangreichen Rechtesystem. Diese Komponenten kombinieren eine datenorientierte Sicht bestehend aus Übersichtstabellen und eine prozessorientierte Sicht bestehend aus Arbeitslisten und Prozessüberwachung miteinander.

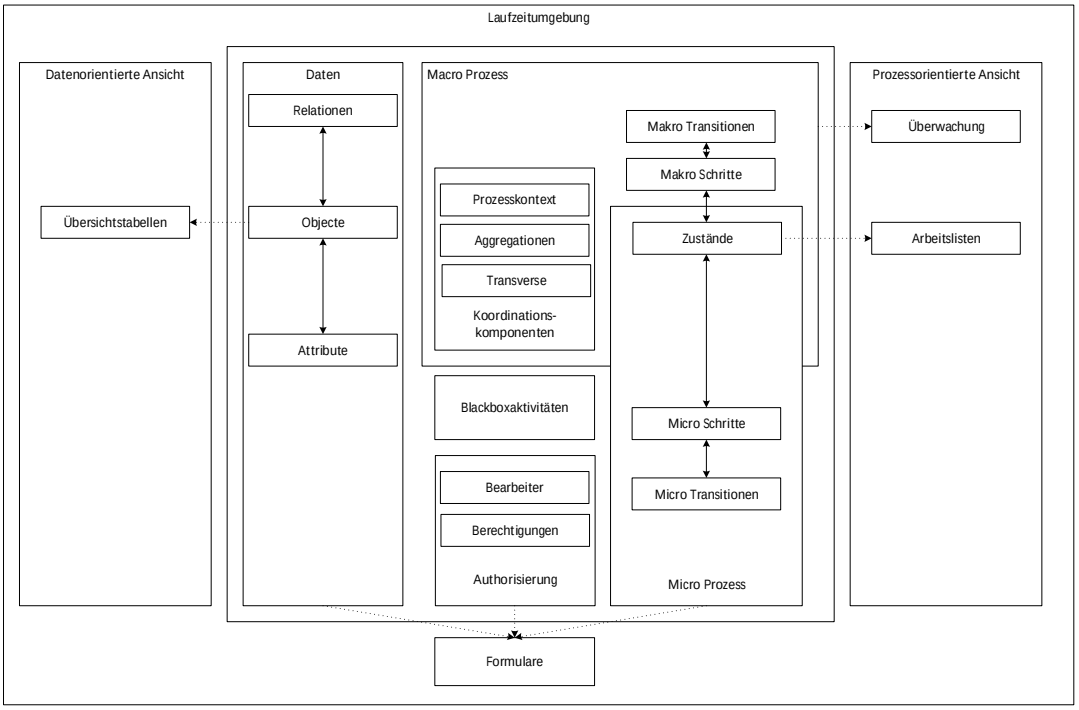

Abbildung 2.4: Modell des PHILharmonicFlows Frameworks, [22]

2.4.1 Objekttypen

Der Kern des PHILharmonicFlow Systemes wird durch Objekttypen gebildet. Ein Objekttyp kapselt eine Menge von Attributen, welche die Prozessdaten des modellierten Prozesses darstellen. In Abbildung 2.5 wird ein konkretes Beispiel eines Objekttyps dargestellt.

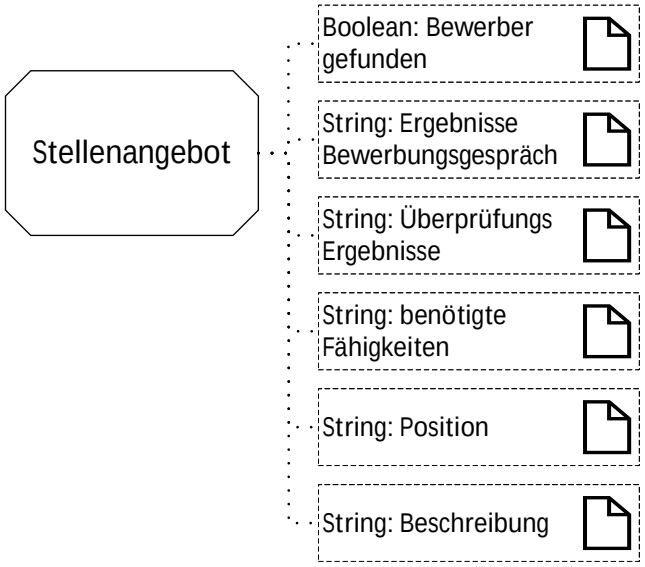

Abbildung 2.5: Darstellung eines Objekttyp mit zugehörigen Attributen,

Attribute enthalten Werte, welche einen Datentyp besitzen. Ein Attribut wird somit durch den Attributname und den Datentyp eindeutig spezifiziert. Alle Attribute können entweder als einzelner Wert oder als Liste von mehreren Werten desselben Datentyps angelegt werden. In PHILharmonicFlows stehen für Attribute folgende Datentypen zur Verfügung: Um Freitexte darzustellen wird der Datentyp String verwendet. Der String Typ enthält weiterhin die Möglichkeit seinen Inhalt auf vordefinierte Texte zu beschränken. Ein Beispiel hierfür wäre ein Attribut mit dem Namen Schultyp, wobei der Inhalt auf Hauptschule, Gymnasium oder Realschule beschränkt wird. Der Datentyp um Zahlen in Attributen darzustellen ist Number. Der Number Typ kann auf ganzzahlige Zahlen

und auch im Wertebereich eingeschränkt werden. Mit dem Datentyp Boolean können Wahrheitswerte mit dem Inhalt True oder False dargestellt werden. Für die Verwaltung von Datumswerten steht der Datentyp Date zur Verfügung. Dateianhänge können mit dem File Datentyp realisiert werden. Die Datentypen Reference und Relation stellen einen Bezug zu anderen Objekttypen dar. Mit Reference können beliebige Typen im Modell referenziert werden. Ein Relationenattribut verlangt eine direkte Relation zu einem anderen Objekttyp. Relationen zwischen Objekttypen werden im Datenmodell dargestellt auf welches im Folgenden näher eingegangen wird [16].

Von den Objekttypen gibt es eine speziellere Variante, den sogenannten Benutzertyp. Benutzertypen repräsentieren Benutzer in einem System, welche zunächst alle Eigenschaften von Objekttypen besitzen. Ein Benutzertyp muss zusätzlich verpflichtend die Attribute Benutzername und Password besitzen und kann im Gegensatz zu normalen Objekttypen Rollen zugewiesen bekommen. Dies wird später unter Benutzerintegration 2.4.6 näher erläutert wird. Die graphische Darstellung ist in der Abbildung 2.6 anhand des Objektes Mitarbeiter zu erkennen.

2.4.2 Datenmodell:

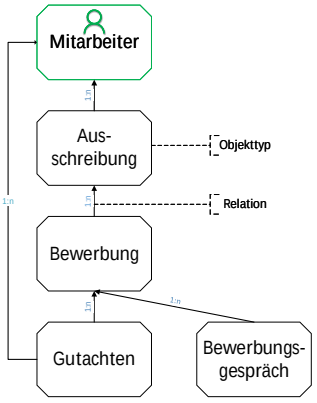

Abbildung 2.6: Beispiel eines einfachen Datenmodells.

Das Datenmodell legt fest, wie Objekttypen miteinander in Beziehung stehen. Das Datenmodell besteht somit aus den Komponenten: Objekttypen inklusive der zugehörigen Attribute, sowie Relationen, welche die Beziehungen der Objekttypen untereinander modellieren. Ein Beispiel eines Datenmodells zu einem Bewerbungsvorgang ist in der Abbildung 2.6 dargestellt.

Relation

Objekttypen stehen in einem Verhältnis zueinander. Dieses Verhältnis wird von Relationen beschrieben. Relationen bilden eine Mengenbeziehung zwischen einer Quell- und einer Zielobjektinstanz. Diese Beziehung wird mithilfe von gerichteten Kanten dargestellt. Relationen besitzen Kardinalitäten, welche an der Relation notiert werden. Diese sind standardmäßig immer 1:n, dies Bedeutet einem Objekt können beliebig viele andere Objekte eines Typs zugeordnet werden. Weiter kann der n Wert über Minimal- und Maximalwerte eingeschränkt werden. Durch die Relationen werden die einzelnen Objekte hierarchisch angeordnet so, dass verschiedene Datenebenen entstehen. Auf der obersten Ebene 0 befinden sich Objekttypen, die keine anderen Objekttypen referenzieren. In der darunterliegenden Ebene 1 diese, die Objekte aus der Ebene 0 referenzieren. Nach diesem Schema entstehen beliebig viele Ebenen. Diese Datenebenen sind wichtig, um später verschiedene semantische Beziehungen, welche im Abschnitt Makroprozesse beschrieben werden, zwischen den Objekten ableiten zu können [16].

2.4.3 Expressions

Expressions werden immer dann verwendet, wenn es darum geht vorhandene Werte auszuwerten. Eine Expression besteht aus Operatoren und Operanden. Die Operanden sind wiederum Expressions. Somit kann durch Schachtelung eine komplexe Bedingung geprüft werden. Mögliche Operatoren sind z.B. Gleichheit, kleiner, größer, boolsches und, boolsches oder sowie viele weitere. Mit den Operatoren einer Expression können Werte, welche entweder Attributen eines Objekttyps stammen oder Konstanten welche in der Expression vom Modellierer eingesetzt werden ausgewertet werden. Jede Ex-

pression kann über Klammern strukturiert werden um eine Auswertungsreihenfolge der Operatoren festzulegen

2.4.4 Mikroprozesse

Jeder Objekttyp besitzt ein Objektverhalten, welches den Lebenszyklus des Objekttyps repräsentiert. Dieses Objektverhalten wird in PHILharmonicFlows durch Mikroprozesse modelliert. Somit gehört zwangsweise zu jedem existierenden Objekttyp ein dazugehörender Mikroprozess [16]. Ein Beispiel Mikroprozess wird in Abbildung **??** dargestellt. Der abgebildete Mikroprozess gehört zu dem in Abbildung 2.5 dargestellten Objekttyp **Stellenangebot**.

Ein Mikroprozess ist ein gerichteter Graph, in dem die Knoten sogenannte Mikroschritte sind und die Kanten sogenannte Mikrotransitionen. Diese Mikroschritt-Typen werden zu Zuständen gruppiert. Dabei muss jeder Mikoschritt genau einem Zustand zugewiesen sein. Ein Mikroprozess besteht somit aus Schritten, Mikrotransitionen und Zuständen [16], welche im Folgenden erklärt werden.

Schritte

Schritte definieren eine Aktion in PHILharmonicFlows. Es existieren verschiedene Arten von Schritten:

Die meist verwendete Variante von Schritten sind die Attributschritte. Jeder Attributschritt referenziert ein Attribut des zugehörigen Objekttyps. Wird ein Attribut von einem Schritt referenziert und der Schritt im Prozess erreicht, so muss dieses verpflichtend mit einem Wert geschrieben werden, dass der Prozess weiter fortschreiten kann.

Etwas komplexer sind die Entscheidungsschritte, welche ebenfalls ein Attribut referenzieren. Zu einem Entscheidungsschritt gehören weiterhin Prädikatsschritte, welche eine Expression beinhalten, welche erfüllt werden muss, um diesen Schritt im Prozessverlauf zu passieren. Das Beispiel 2.7 besitzt ebenfalls einen Entscheidungsschritt in dem das Attribut **Bewerber gefunden** geschrieben und anhand zweier Prädikatschritte mit einer einfachen Expression ausgewertet wird [16].

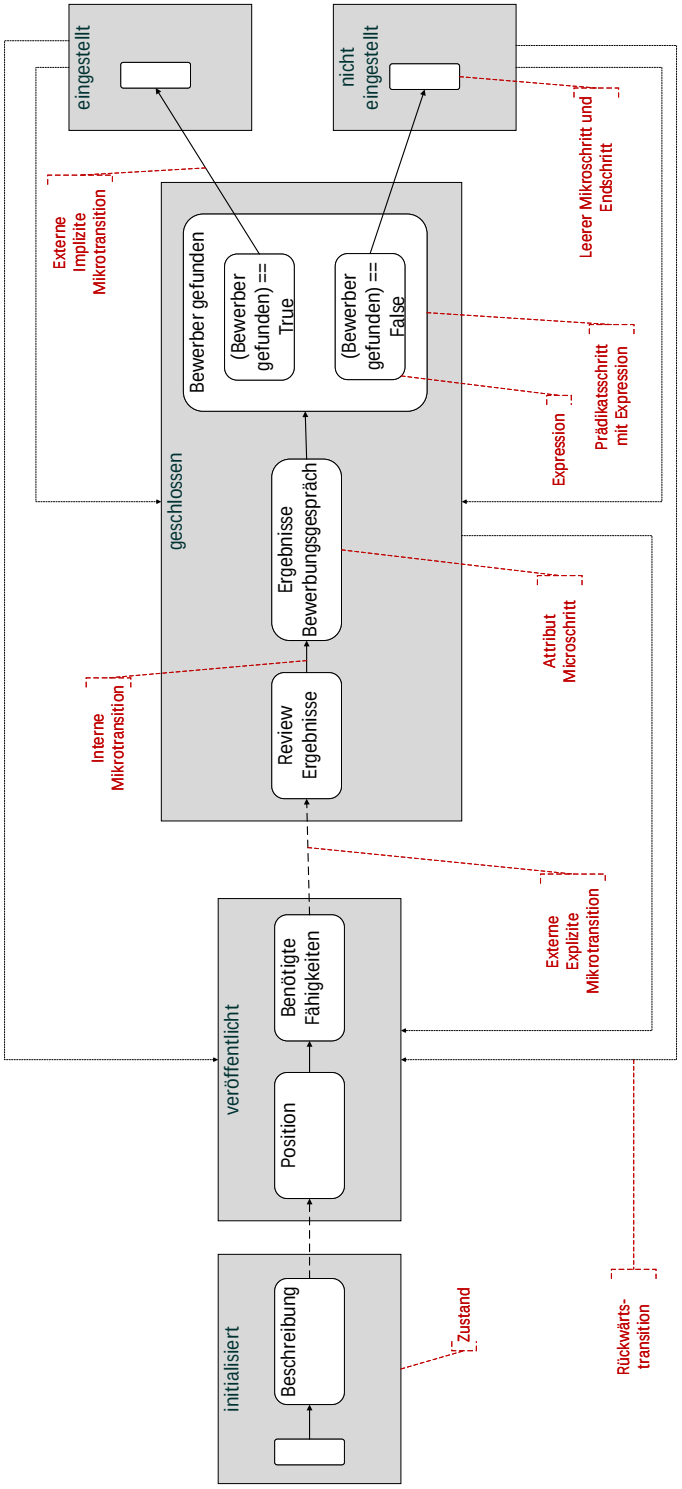

Abbildung 2.7: Mikroprozess mit Erklärungen zum Objekttyp Stellenangebot,

Eine weitere Variante sind leere Schritte. Leere Schritte haben die Besonderheit, dass hier kein Wert in ein Attribut geschrieben wird. Das häufigste Beispiel eines leeren Schrittes ist der Spezialfall eines Start- oder Endschrittes. Ein Mikroprozess muss immer mit einem leeren Startschritt beginnen und mit einem oder mehreren leeren Endschritten terminieren. Dies ist auch im Beispiel 2.7 zu erkennen.

Weiter gibt es noch das Konzept der Berechnungsschritte und der Blackboxschritte, beide Konzepte befinden sich noch in der Entwicklungsphase und existierten zum Zeitpunkt der Evaluation noch nicht in der Modellierungsumgebung von PHILharmonicFlows, daher werden bei der Modellierung Platzhalter verwendet.

Ein Berechnungsschritt repräsentiert einen Schritt, welcher in der Lage ist, einfache Berechnungen mit Attributen durchzuführen um das Ergebnis der Berechnung in einem Attribut abzuspeichern. Als Notation wird in dieser Arbeit die in Abbildung 2.8 gezeigte Symbolik verwendet.

Blackboxschritte werden immer dann verwendet wenn eine Funktionalität gefordert wird welche innerhalb des Konzeptes von PHILharmonicFlows nicht generisch realisierbar ist. Ein häufiger Anwendungsfall ist z.B. das Versenden einer Mail oder das Einlesen von Daten von einem anderen System. Als Notation wird in dieser Arbeit die in Abbildung 2.8 gezeigte Symbolik verwendet.

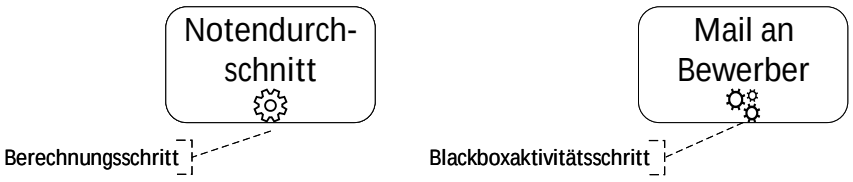

Abbildung 2.8: Notation für Berechnungs- und Blackboxaktivitätsschritte.

Zustände

Schritte werden in Zuständen gruppiert. Dies ermöglicht eine fachliche Gruppierung von zusammengehörigen Attributen. Dies wird auch zur Strukturierung bei der Formulargene-

rierung benötigt. Weiter werden Zustände für die später beschriebene Koordination von verschiedenen Prozessinstanzen mithilfe von Makroprozessen verwendet. So besitzt der Beispielmikroprozess 2.7 fünf verschiedene Zustände, wobei der Zustand **veröffentlicht** die Attribut Mikroschritte **Position** und **benötigte Fähigkeiten** gruppiert. Dies hat folgende Auswirkung: Um den Zustand **veröffentlicht** im Prozesverlauf passieren zu dürfen, müssen zwangsweise die Attribute **Position** und **benötigte Fähigkeiten** einen Wert besitzen. In welcher Reihenfolge diese Werte geschrieben werden ist jedoch unwichtig. So können die Werte auch schon in einem vorangegangenem Zustand geschrieben worden sein. In welcher Reihenfolge Zustände in einem Prozessablauf durchlaufen werden können legen Transitionen fest. Jeder Mikroprozess bildet einen zyklenfreien gerichteten Graph in dem die einzelnen Mikroschritte mit Transitionen untereinander verbunden werden. Dabei gibt es immer einen Startzustand und beliebig viele Endzustände.

Transitionen

Transitionen verbinden Schritte miteinander, hierbei gibt es verschiedene Arten. Interne Mikrotransitionen verbinden zwei Mikroschritte innerhalb eines Zustandes. Externe Mikrotransitionen verbinden zwei Mikroschritte, die in verschiedenen Zuständen liegen. Externe Mikrotransitionen können als implizite oder als explizite Transitionen modelliert werden. Bei impliziten externen Transitionen, findet automatisch ein Übergang zum nächsten Zustand statt, sobald der vorangegangene Mikroschritt vollständig ist. Bei expliziten Transitionen muss zusätzlich vom Benutzer der Übergang bestätigt werden. Im Beispiel 2.7 werden die verschiedenen Arten dargestellt.

Rückwärtstransitionen

Mit Rückwärtsransitionen ist es möglich, auf frühere Zustände des Prozesses zurückzuspringen um in dem vorangegangenen Zustand Daten korrigieren zu können. Eine Rückwärtstransition verbindet hierfür zwei Zustände miteinander, zwischen denen ein Rücksprung erlaubt ist. Im Beispiel 2.7 ist auch eine Rückwärtstransition verdeutlicht.

Ein Rücksprung auf frühere Prozesszustände ist somit nur möglich, wenn dies vom Modellierer explizit mittels einer Rücksprungtransition vorgesehen wurde.

2.4.5 Makroprozesse

In den vorangegangenen Kapiteln wurden Objekttypen und deren zugehörige Mikroprozesse erläutert. Ein einzelner Objekttyp mit seinem Mikroprozess stellt jedoch in aller Regel keinen vollständigen Geschäftsprozess dar. Geschäftsprozesse sind wesentlich komplexer aufgebaut und benötigen somit im Allgemeinen mehrere Objekttypen welche jeder für sich, einen Teil des Gesamtprozesses darstellen. Da zwischen den einzelnen Teilprozessen, welche einen Gesamtprozess ergeben sollen, Abhängigkeiten bestehen, müssen Objekttypen untereinander koordiniert werden. Dies bedeutet, die einzelnen Mikroprozesse können zwar in vielen Fällen parallel abgearbeitet werden, aber immer dann wenn Abhängigkeiten bestehen, muss für eine Koordination gesorgt werden, wobei aber die parallele Ausführung so wenig wie möglich eingeschränkt werden soll. Diese Koordination erfordert eine Kommunikation zwischen den einzelnen Prozessen. Diese angestrebte Koordination der Objekte untereinander beruht auf semantischen Beziehungen der Objekte. Daher werden im folgenden zunächst semantische Beziehungen zwischen verschiedenen Objekten erläutert.

Semantische Beziehungen

Semantische Beziehungen sind ein Konzept, welches Beziehungen zwischen Objekten spezifiziert. Eine semantische Beziehung zwischen zwei Objekten kann verschiedene Eigenschaften besitzen Abhängig von diesen Eigenschaften können die Beziehungen in die Kategorien Top-Down, Bottom-Up, Transverse, Self und Self-Transverse eingeteilt werden. Die einzelnen Kategorien von semantischen Abhängigkeiten werden im folgenden erläutert.

Von einer Top-Down Beziehung wird gesprochen, wenn die Ausführung von einem oder mehreren untergeordneten Prozessen von einem Ausführungsstatus eines übergeordneten Prozess abhängig ist. Schematisch ist dies in Abbildung 2.9 dargestellt.

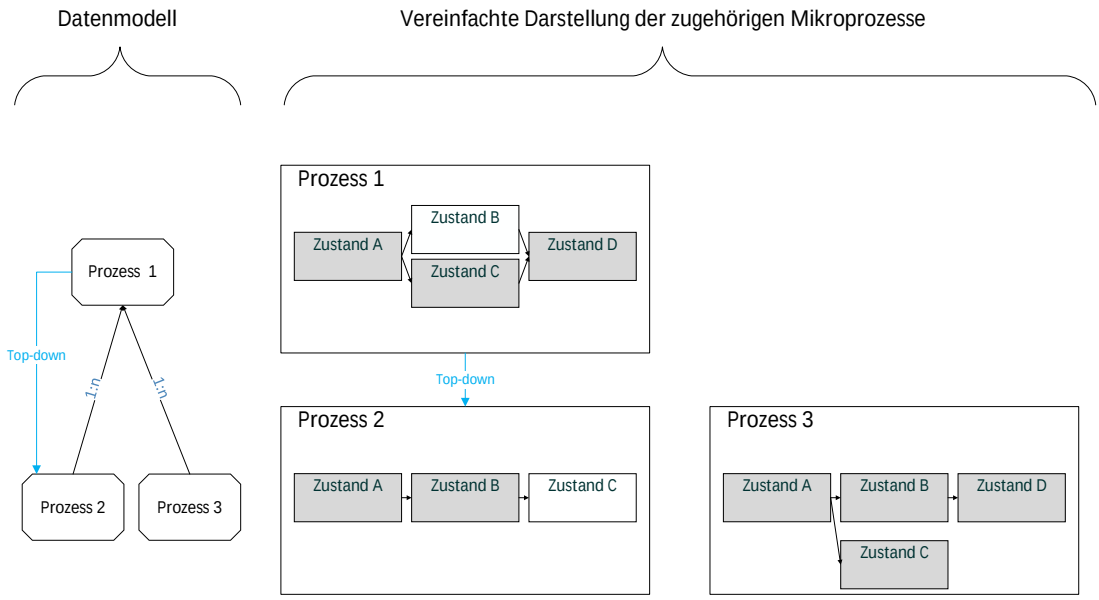

Abbildung 2.9: Semantische Top-down Beziehung.

Hierbei ist der **Prozess 2** abhängig von der Ausführung des **Prozesses 1**. Das heißt, der **Prozess 1** muss einen bestimmen Ausführungsstand erreichen (Im Beispiel **Zustand B**), dass der **Prozess 2** starten, bzw. weiter fortschreiten darf (Im Beispiel fortschreiten zu **Zustand C**). Weiter muss bei einer Top-Down Beziehung spezifiziert werden, wie lange die Beziehung gültig ist. Im Beispiel ist die Beziehung erfüllt, sobald sich der **Prozess 1** im **Zustand B** befindet. Sobald aber der **Zustand B** wieder verlassen wird und der **Prozess 1** in den **Zustand D** übergeht, ist die Bedingung nicht mehr erfüllt. Soll sie weiterhin erfüllt bleiben, so können Folgezustände des **Zustand B** einzeln als auch erlaubt definiert werden. Eine Top Down Beziehung liegt z.B. dann vor, wenn der Bewerber zu einer Bewerbung vollständig angelegt sein muss, damit eine Bewerbung abgegeben werden kann.

Um Bottom-Up Beziehungen handelt es sich, wenn die Ausführung eines höherliegenden Prozesses vom Ausführungsstand der Prozessinstanzen eines tieferliegenden Prozesses abhängen. Diese Abhängigkeit wird durch Grafik 2.10 illustriert. So muss sich der **Prozess 2** in einem bestimmten Ausführungsstand befinden (Im Beispiel im

Zustand B), damit der **Prozess 1** starten, bzw. weiter fortschreiten darf (Im Beispiel starten). Bottom-Up Beziehungen sind jedoch komplexer, da es von dem untergeordneten Objekttyp mehrere Instanzen geben kann. Somit sind für korrekt formulierte Bottom-Up Beziehungen Angaben über die Menge der untergeordneten Prozessinstanzen unabdingbar. Somit gehört zur jeder Bottom-up Beziehung eine Expression welche diese Menge auswertet. Anhand dieser Expression kann definiert werden, wieviele der untergeordneten Prozesse die Bedingung erfüllen müssen. Denkbar ist hier jede Variante von nur einer Prozessinstanz (z.B. es reicht ein positives Ergebnis von mehreren Beurteilungen) bis zu allen Instanzen (z.B. Alle Beurteilungen müssen positiv sein). Eine Bottom-Up Beziehung liegt z.B. dann vor wenn mindestens zwei positive Gutachten zu einer Bewerbung vorliegen müssen, dass die Bewerbung im Prozess voranschreiten darf.

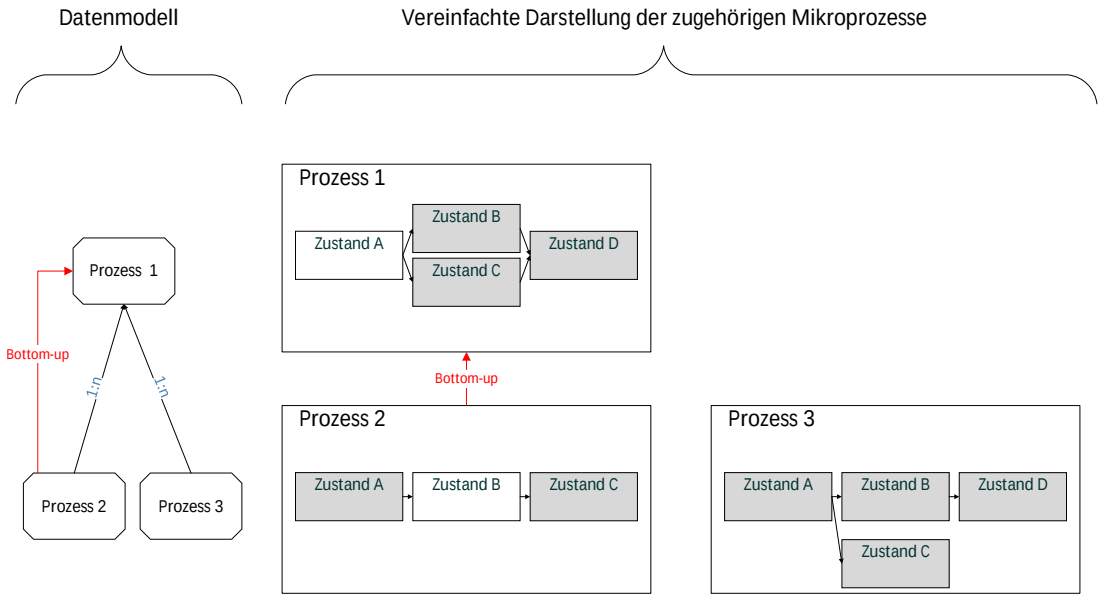

Abbildung 2.10: Semantische Bottom-Up Beziehung.

Besitzen zwei abhängige Prozesse einen gemeinsamen übergeordneten Prozess, so wird von einer transversen Abhängigkeit gesprochen. Dies wird in der Abbildung 2.11 verdeutlicht. So hängt im Beispiel der Ausführungsstand des **Prozesses 3**, vom Ausfüh-

rungsstand des **Prozess 2** ab. Im Beispiel verlangt der **Prozess 3** um in den **Zustand D** übergehen zu können, dass der **Prozess 2** sich im **Zustand B** befindet. Da vom **Prozess 2** wieder, wie auch bei den Bottom-Up Beziehungen mehrere Instanzen existieren können, muss hier wieder eine Mengenbeziehung wie auch bei der Bottom-Up Abhängigkeit mittels einer Expression spezifiziert werden. Eine transverse Beziehung liegt z.B. dann vor, wenn zu einer Bewerbung nur dann ein Bewerbungsgespräch stattfinden kann, wenn mindestens 4 positive Beurteilungen zu der Bewerbung abgegeben wurden.

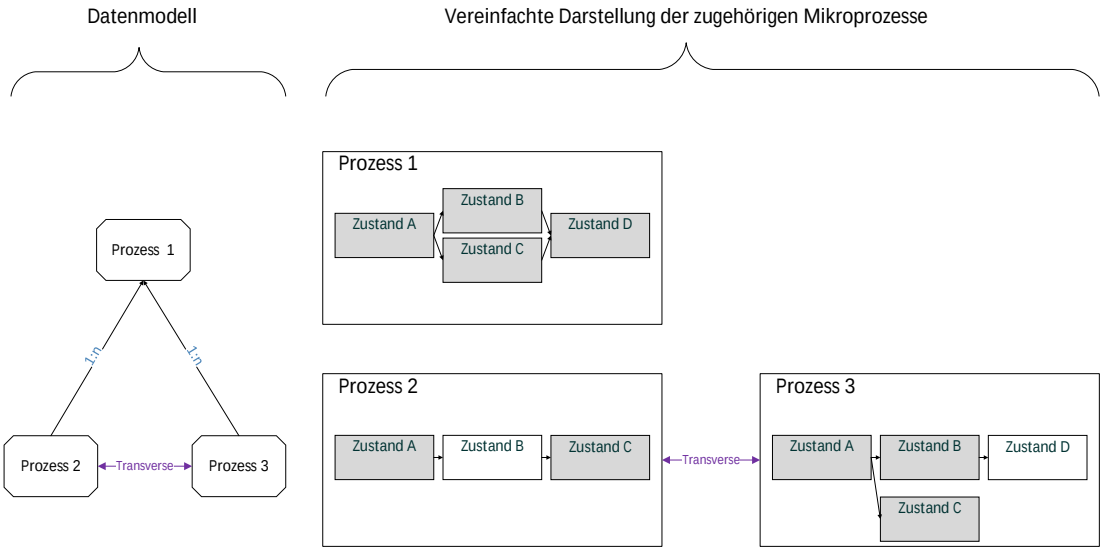

Abbildung 2.11: Semantische transverse Beziehung.

Die Self Beziehung ist die einfachste semantische Abhängigkeit. Die Self Beziehung erwartet keine weiteren Bedingungen und die Abhängigkeiten sind bereits durch den Mikroprozess definiert. In Abbildung 2.12 wird dies verdeutlicht. Im Beispiel kann der **Zustand D** des **Prozesses 1** nur erreicht werden, wenn vorab der **Zustand A** desselben Prozesses erfüllt wurde. Wird die Abhängigkeit auf andere Prozessinstanzen desselben Objekttyps bezogen, wird von einer Self-Transversen Beziehung gesprochen. Dies wird in Abbildung 2.12 Anhand des **Prozesses 3** dargestellt. Der **Prozess 3** darf nur dann in den **Zustand B** übergehen, wenn es andere Prozessinstanzen des **Prozess 3** gibt, welche sich im **Zustand C** befinden. Diese mengenmäßige Abhängigkeit wird

wie bei einer transversen Abhängigkeit auch mittels einer Expression definiert. Eine Self-Transverse Beziehung herrscht z.B. dann, wenn nur ein Bewerber auf eine Stelle angenommen werden kann.

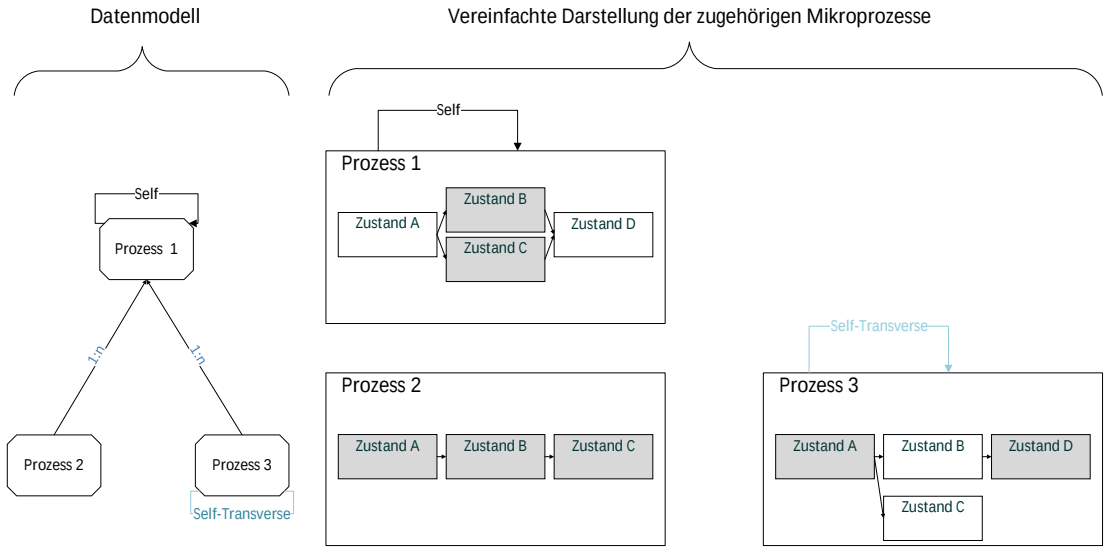

Abbildung 2.12: Semantische Self und Self-Transverse Beziehung.

Expressions für semantische Beziehungen

Bei der Beschreibung der semantischen Beziehungen wurden an verschiedenen Stellen Expressions erwähnt. Die Expressions besitzen denselben Aufbau wie im Kapitel Mikroprozess beschrieben. Jedoch gibt es noch spezifische Erweiterungen in Form von Funktionen, welche die Anzahl der Prozessinstanzen berechnet die eine Bedingung in Abhängigkeit von einem gegebenem Zustand erfüllen. Um dies zu verdeutlichen wird das Beispiel in Abbildung 2.13 verwendet. Das Beispiel zeigt vier Prozessinstanzen in verschiedenen Zuständen.

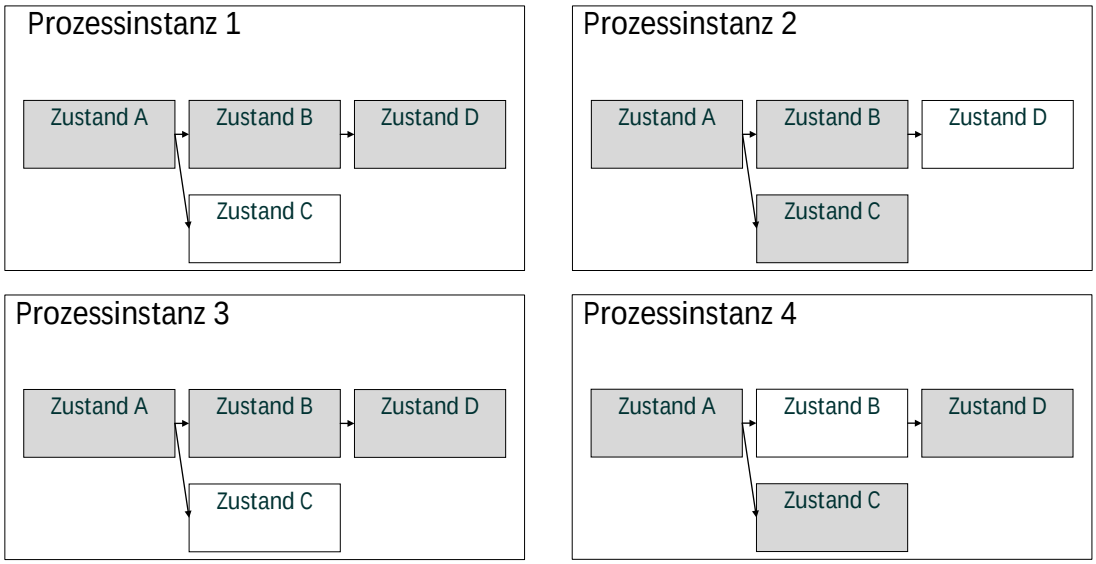

Abbildung 2.13: Beispiel zu Prädikaten in Expressions.

Das Prädikat **ALL** gibt die Anzahl aller Prozessinstanzen zurück. Im Beispiel wären das alle 4 Stück. Das Prädikat **IN** gibt die Anzahl aller Prozessinstanzen welche sich im gewählten Zustand befinden zurück. Wird die Bedingung **IN=Zustand C** verwendet, so lautet der Wert des Prädikates 2 (Instanz 1 und 3). Die Prädikate **AFTER** und **BEFORE** die jeweilige Anzahl an Prozessinstanzen welche sich vor, bzw. nach dem aktiven Zustand befinden. So ergibt **AFTER=Zustand B** den Wert 1 (Instanz 2) und **BEFORE=Zustand B** den Wert 0. Mit **SKIPPED** können Prozessinstanzen, welche nicht mehr erreichbar sind gezählt werden. So ergibt das Prädikat **SKIPPED= Zustand C** den Wert 2.(Instanz 2 und 4).

Um solche Abhängigkeiten zwischen verschiedenen Mikroprozessen in PHILHarmonicFlows modellieren zu können, existiert das Konzept der Makroprozesse [16]. Den zweiten Teil der Prozessstruktur bei PHILharmonicFlows stellen somit die Makroprozesse dar. Diese modellieren die Abhängigkeiten verschiedener Zustände von verschiedenen Objekttypen zueinander. Hierzu werden Makrotransitionen verwendet. Der Makroprozess ist somit kein Prozess der nur abgearbeitet wird, sondern auch ein Koordinationsmechanismus für die Mikroprozess-Instanzen.

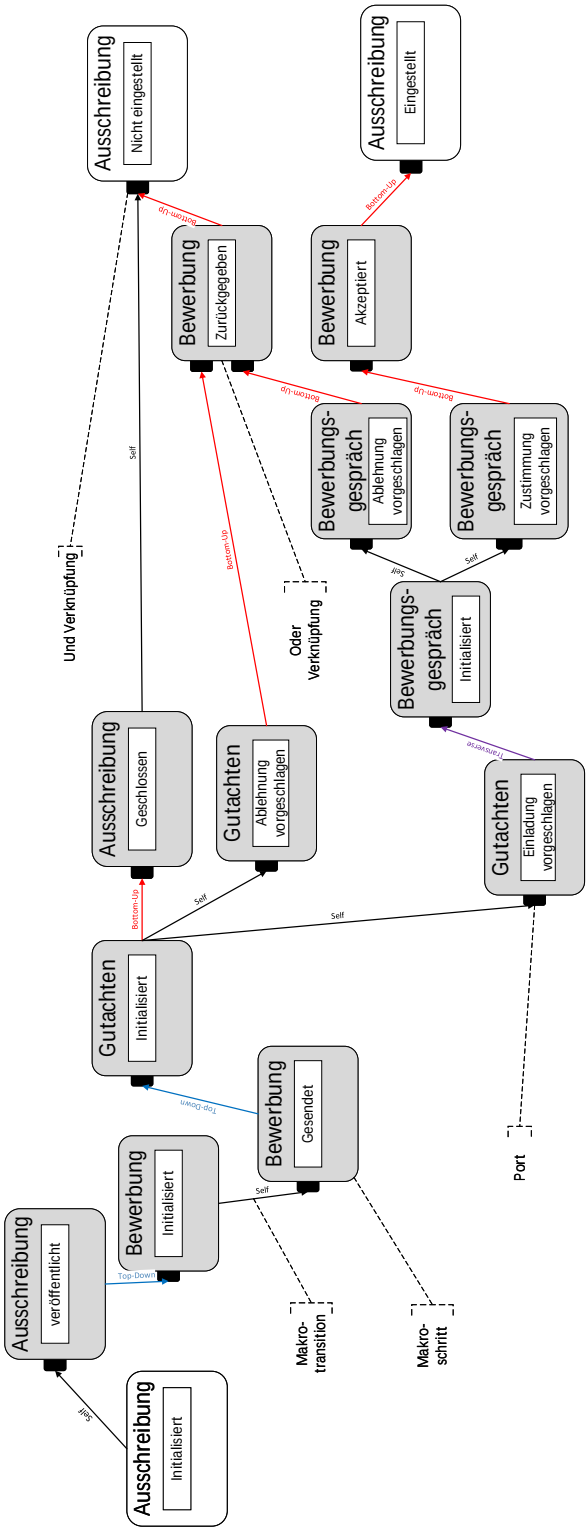

Abbildung 2.14: Darstellung des Makroprozesses zum Objekttyp Job Angebot.

Der Makroprozess besteht aus Makroschritten und Makrotransitionen. In einem PHILharmonicFlows Prozessmodell kann es einen oder auch mehrere Makroprozesse geben. Ein Makroprozess wird immer einem Objekttyp zugeordnet. Dieser Objekttyp wird dann koordinierender Objekttyp genannt. Im folgenden wird ein Makroprozesses welcher dem Objekttyp **Ausschreibung** aus Abbildung 2.5 zugeordnet wurde und in Abbildung 2.14 dargestellt wird als Beispiel verwendet. Der koordinierende Objekttyp ist somit der Objekttyp **Ausschreibung**. Ein Makroprozess welcher einem Objekttyp zugeordnet wurde ist in der Lage, Koordination von Objekttypen welche in derselben oder in einer tiefer liegenden Datenebene liegen und durch Relationen verknüpft sind, vorzunehmen.

Makroschritt

Die grundlegende Komponente eines Makroprozesses sind die Makroschritte. Die Komponenten eines Makroschrittes werden in der Abbildung 2.15 dargestelllt.

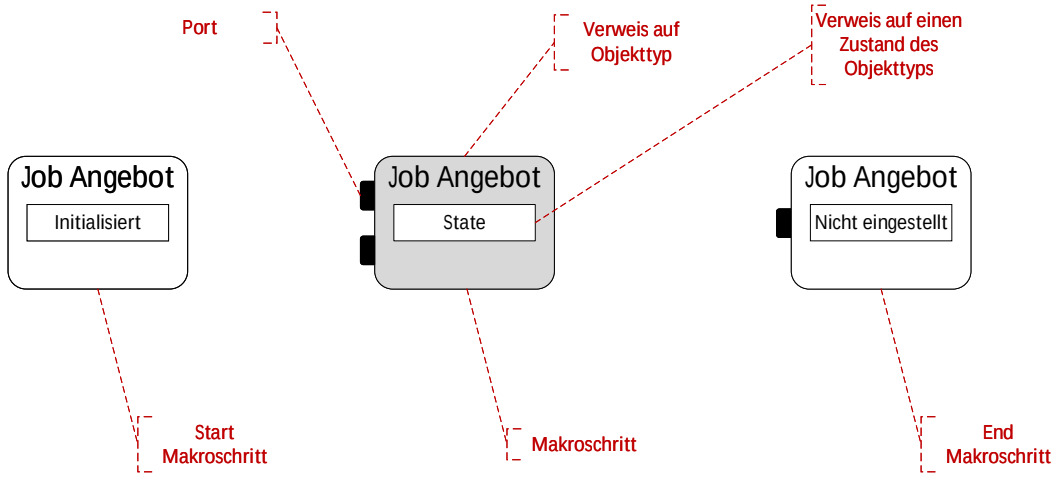

Abbildung 2.15: Darstellung eines Makro Schritt Typ.

Makroschritte bestehen erstens aus einem Verweis auf einen Objekt-Typ und einem Zustand von dem Objekt [24]. Die Start- und Endmakroschritte eines Makroprozesses müssen immer auf die entsprechenden Start- und Endzustände des zum koordinieren-

den Objekttyp gehörenden Mikroprozesses verweisen. Zwischen Start- und Endstep dürfen beliebige Objekt-Typ-Zustandskombinationen verwendet werden. Einschränkungen hierzu können in der Arbeit von Vera Künzle [16] nachgelesen werden. Um Makroschritte mit anderen Makroschritten verknüpfen zu können, besitzt jeder Makroschritt mindestens einen Port (Ausnahme sind Startschritte).

Ports

Mithilfe der Ports kann die Semantik festgelegt werden, wann ein Makroschitt aktiviert wird. Soll ein Makroschritt aktiviert werden, muss mindestens ein Port aktiviert sein, was einer Oder-Semantik entspricht. Dass ein Port erfüllt wird, müssen alle an ihm anliegenden Transitionen erfüllt werden, was einer Und-Semantik entspricht und somit direkt zum nächsten Element eines Makroprozesses, den sogenannten Transitionen führt, welche Makroschitte miteinander verbinden. Dies wird in Abbildung 2.16 dargestellt.

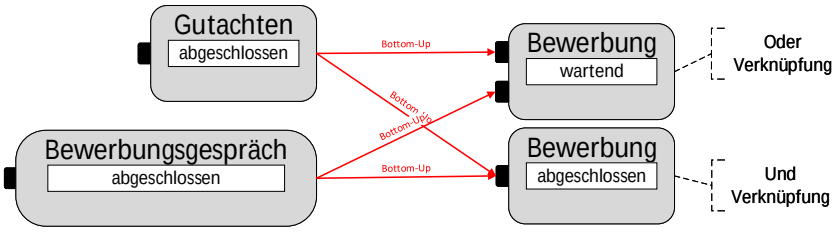

Abbildung 2.16: Beispiel zur Darstellung von Und und Oder

Makrotransitionen

Makrotransitionen verbinden zwei einen Schritt und einen Port miteinander. Diese Transitionen stellen eine semantische Beziehung dar, welche für das Aktivieren des Zustandes im Zielschritt erfüllt sein muss.Die Art der semantischen Beziehung ergibt sich unmittelbar aus den beteiligten Objekten und deren Anordnung im Datenmodell. Erst wenn die Bedingungen des Makroprozesses erfüllt sind, dürfen die von den Zuständen abhän-

gigen Mikroprozesse weiter fortfahren. Die Abbildung 2.17 stellt diese Semantischen Beziehungen dar.

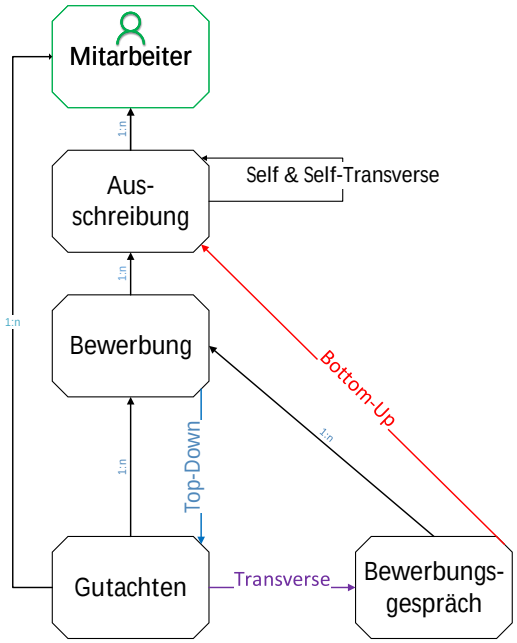

Abbildung 2.17: Beispiel zu den Arten zur Koordination im Makroprozess, abhängig von der Anordnung im Datenmodell.

2.4.6 Benutzerintegration

Da nicht jeder Benutzer des Systems alle Daten lesen oder schreiben können soll, sondern nur die Daten für welche er auch zuständig ist, existiert in PHILharmonicFlows ein umfangreiches Authorisierungssystem. Bei PHILharmonicFlows werden die Benutzer durch die Benutzertypen mit in das Modell integriert. Somit sind Benutzer direkt Teil des Datenmodells und stehen in einer sehr engen Verbindung zum Prozessablauf. Daher werden Benutzer wie in 2.4.1 beschrieben als Benutzertyp im Datenmodell integriert. Benutzertypen können dann verschiedene Arten an Rechten erhalten, jedoch werden die Rechte nicht direkt an die Benutzertypen vergeben, sondern an Rollen gebunden, welche ein Benutzertyp als Kontextrolle oder als Relationenrolle erhalten kann [24].

Rollen

Um Benutzertypen mit Rechten zu versehen werden Rollen verwendet. Zu jeder Rolle wird dann definiert, auf welchen Attributen die Rolle Lese- oder Schreibrechte hat. Eine Rolle besteht somit aus einem Rollennamen, den zugehörigen Rechten, sowie einer Expression welche die Rollenvergabe an eine Bedingung binden kann. Es gibt im Konzept von PHILharmonicFlows zwei verschiedene Arten von Rollen. Der Unterschied hierbei ist, wie diese Rollen vergeben werden.

Kontextrollen erhält ein Benutzertyp dann, wenn die entsprechende Rolle bei seinem Benutzertyp definiert wurde und die bei der Definition hinterlegte Expression, welche auch Bezug auf die Attribute des Benutzertyps nehmen kann, zur Laufzeit zutrifft. Somit ist es möglich, einem Benutzertyp abhängig von Attributwerten verschiedene Rollen zu vergeben. Ein Benutzertyp kann auch zeitgleich mehrere Rollen besitzen. Besitzt ein Benutzertyp mehrere Rollen, so erhält er die Vereinigungsmenge aller Rechte der Rollen, die ihm zugeteilt wurden.

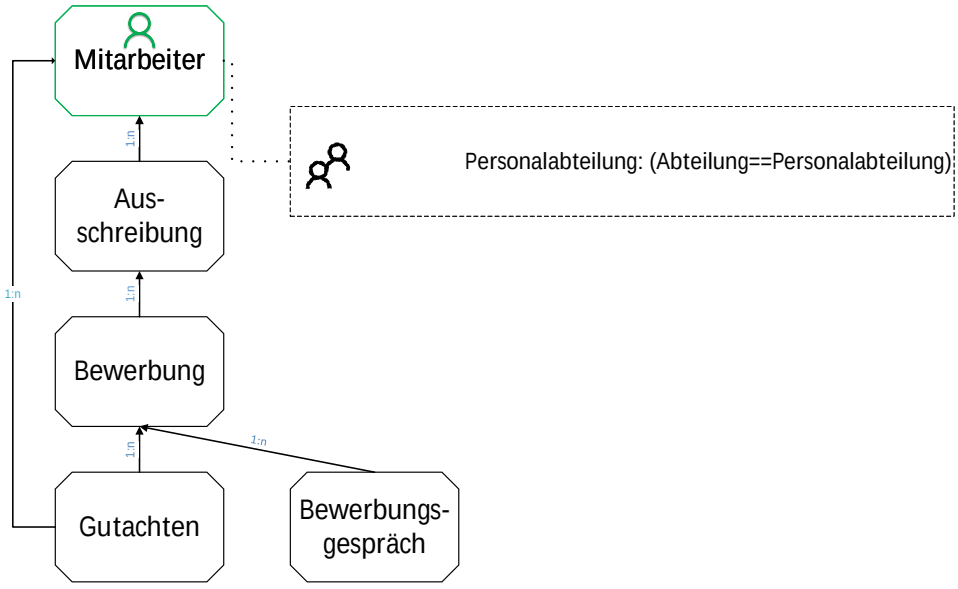

Abbildung 2.18: Vergabe einer Kontextrolle

Das Beispiel in Abbildung 2.18 definiert eine Kontextrolle **Personalabteilung** zu dem Benutzertyp **Mitarbeiter**. Bei der Definition der Kontextrolle wird die Expression **Abteilung==Personalabteilung** hinterlegt. **Abteilung** ist ein String Attribut des Benutzertyps **Mitarbeiter**. Steht in dem Attribut in der Instanz zur Laufzeit "Personalabteilung", so wird die entsprechende Rolle an den Benutzertyp vergeben.

Relationenrollen erhält ein Benutzertyp dann, wenn auf einer Relation im Datenmodell eine Relation Role definiert wurde und wenn die Userinstanz zur Laufzeit über eine Instanz dieses Relation Types mit einer anderen Objektinstanz verknüpft wird. Optional kann zusätzlich noch eine Expression hinterlegt werden, welche von der Benutzerinstanz erfüllt werden muss um zusätzliche Einschränkungen zur Vergabe hinterlegen zu können. Somit besteht hierdurch die Möglichkeit, die Relationenrollenvergabe noch abhängig von Attributwerten zu gestalten.

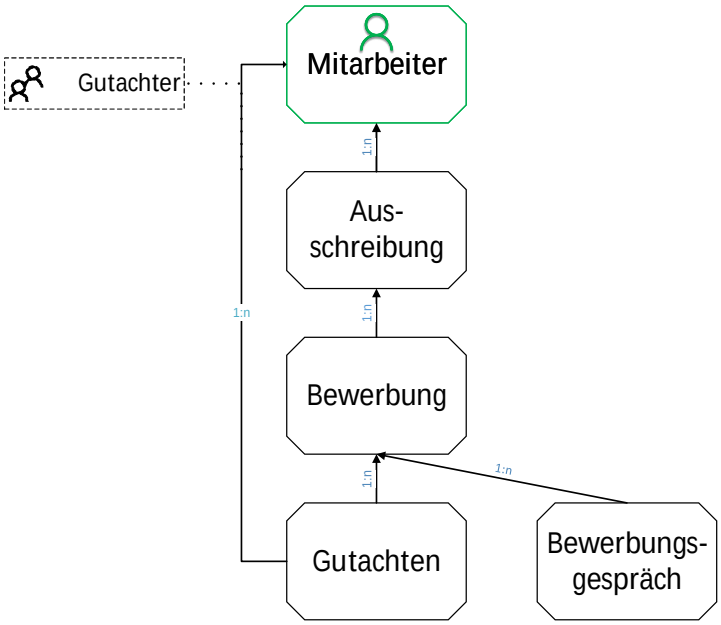

Abbildung 2.19: Vergabe einer Relationenrolle

Das Beispiel in Abbildung 2.19 demonstriert die Vergabe einer Relationenrolle **Gutachter**, welche an den **Mitarbeiter** vergeben wird sobald zur Laufzeit eine Relation

zwischen einer **Überprüfung** und einem **Mitarbeiter** angelegt wird. Hierbei wurde keine weitere Einschränkung mittels einer Expression vorgenommen.

Rechte

Zu jeder Rolle gehört die Vergabe von Rechten an die Rolle. Rechte erlauben den Rollen die Attributetypen in der Datenstruktur zu lesen oder auch zu schreiben. Hierbei können Rechte in Lese- und Schreibrechte unterteilt werden. Weiterhin können Rechte instanzspezifisch sein und einen sogenannten Datenkontext haben. Was dies im Einzelnen bedeutet, wird im Folgenden erklärt.

Mithilfe des Datenkontext kann ein Recht eingeschränkt werden. Hierzu wird wieder eine Expression verwendet, welche die Bedingung der Einschränkung definiert. Zum Beispiel kann dadurch definiert werden, dass ein Recht nur gilt, wenn eines der betroffenen Objekte das Attribut **Name** mit dem Wert **Maier** enthält.

Instanzspezifische Rechte sind Rechte einer Rolle zu einem bestimmten Objekttyp. Bestimmt wird dieser Objekttyp über einen Pfad von Relationen. Somit können die Instanzen zur Laufzeit in zwei Gruppen eingeteilt werden. Die erste Gruppe sind die abhängigen Objektinstanzen welche über den Pfad der Relationen von der Benutzerinstanz aus erreichbar sind. Für diese gelten dann die instanzspezifischen Rechte. Die zweite Gruppe sind die unabhängigen welche nicht über einen Pfad erreichbar sind und somit nur über die allgemeinen Rechte verfügen.

Beispiele zu den Rechten

Im folgenden einige Beispiele zu dem Rechtesystem. Die Beispiele zeigt die Kombinationsmöglichkeiten der einzelnen Rechtearten. Hierzu verwenden wir zwei Objekttypen: **Ausschreibung** und **Bewerbung** sowie die definierten Relationen (**Bewerbung ->Ausschreibung**) und (**Ausschreibung ->Mitarbeiter**). Das dem Beispiel zugrunde liegende Datenmodell wird in Abbildung 2.20 dargestellt.

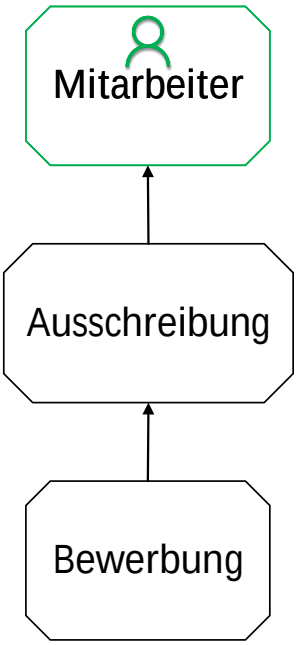

Abbildung 2.20: Datenmodell für das Rechtebeispiel.

In den folgenden Beispielen existieren Von den Objekten Mitarbeiter und Ausschreibung jeweils eine und von dem Objekt Bewerbung jeweils 5 Instanzen. Von diesen 5 Bewerbungsinstanzen sind 2 mittels einer Relation mit der Instanz des Ausschreibungsobjektes verbunden. Attribute mit Schreibrechten werden hell, Attribute ohne werden dunkel eingefärbt dargestellt.

Beispiel 1 zeigt eine Kontextrolle mit einem Schreibrecht auf das Attribut **Z** des Objekttyp **Bewerbung**. Das Recht ist nicht instanzspezifisch und besitzt keinen Datenkontext. Der Benutzer erhält Schreibrechte auf das Attribut **Z** der Objektinstanzen **Bewerbung 1**, **Bewerbung 2**, **Bewerbung 3**, **Bewerbung 4** und **Bewerbung 5**. Das Beispiel wird in Abbildung 2.21 illustriert.

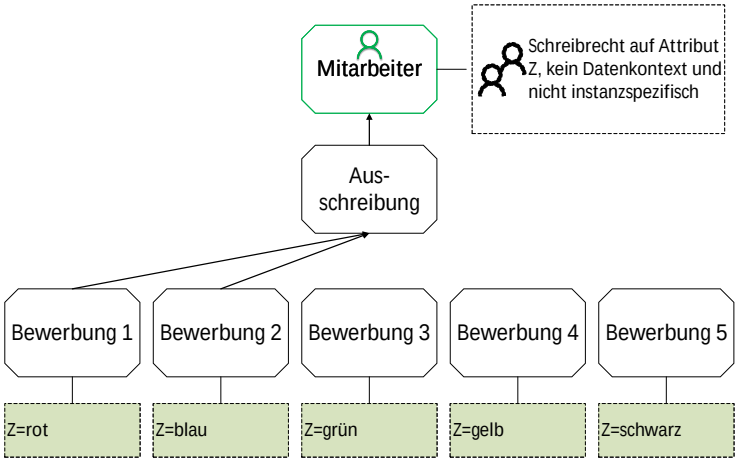

Abbildung 2.21: Abbildung zu Beispiel 1.

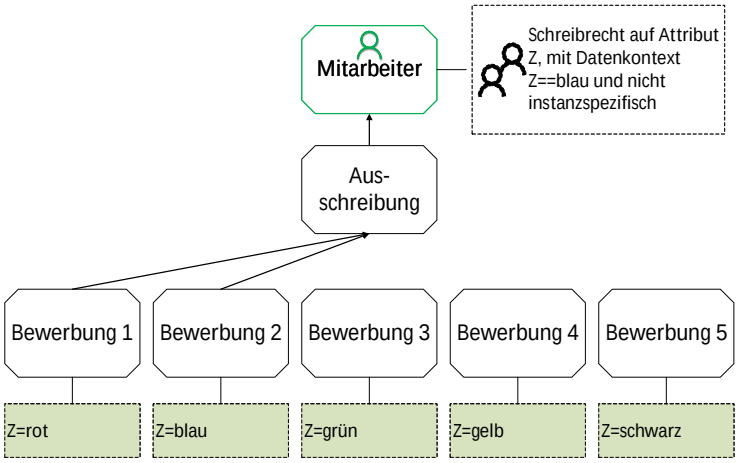

Abbildung 2.22: Abbildung zu Beispiel 2.

Beispiel 2 zeigt eine Kontextrolle mit einem Schreibrecht auf das Attribut **Z** des Objekttyp **Bewerbung**. Das Recht ist nicht instanzspezifisch und besitzt den Datenkontext **Z==blau**. Der Benutzer erhält Schreibrechte auf das Attribut **Z** der Objektinstanzen

Bewerbung 1, **Bewerbung 2**, **Bewerbung 3**, **Bewerbung 4** und **Bewerbung 5**. Da das Recht nicht instanzspezifisch ist reicht es aus, dass eine Objektinstanz ein Attribut **Z** mit Wert blau besitzt, um das Recht für alle Instanzen zu gewähren. Das Beispiel wird in Abbildung 2.22 illustriert.

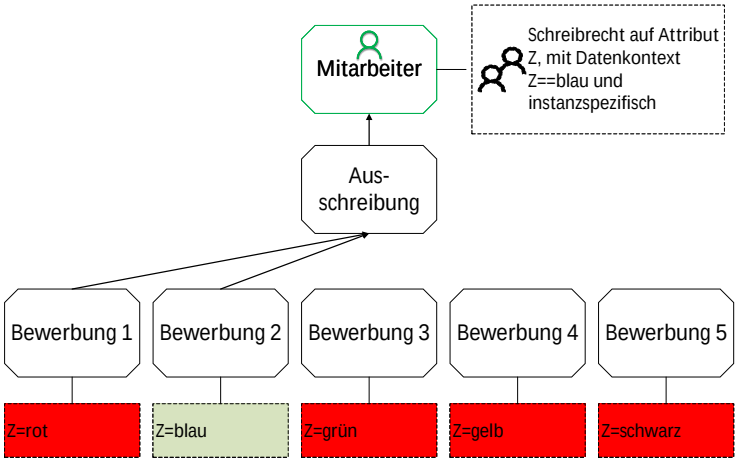

Abbildung 2.23: Abbildung zu Beispiel 3.

Beispiel 3 zeigt eine Kontextrolle mit einem Schreibrecht auf das Attribut **Z** des Objekttyp **Bewerbung**. Das Recht ist instanzspezifisch und besitzt den Datenkontext **Z==blau**. Der Benutzer erhält Schreibrechte auf das Attribut **Z** der Objektinstanz **Bewerbung 2**. Da das Recht hier instanzspezifisch gewählt wurde gilt das Recht nur für die Instanz Bewerbung 2 welche die Bedingung erfüllt. Das Beispiel wird in Abbildung 2.23 illustriert.

Beispiel 4 zeigt eine Relationrolle auf der Relation **Ausschreibung–>Mitarbeiter** mit einem Schreibrecht auf das Attribut **Z** des Objekttyp **Bewerbung**. Das Recht ist nicht instanzspezifisch und besitzt keinen Datenkontext. Der Benutzer erhält Schreibrechte auf das Attribut **Z** der Objektinstanzen **Bewerbung1**, **Bewerbung 2**, **Bewerbung 3**, **Bewerbung 4** und **Bewerbung 5**. Das Beispiel wird in Abbildung 2.24 illustriert.

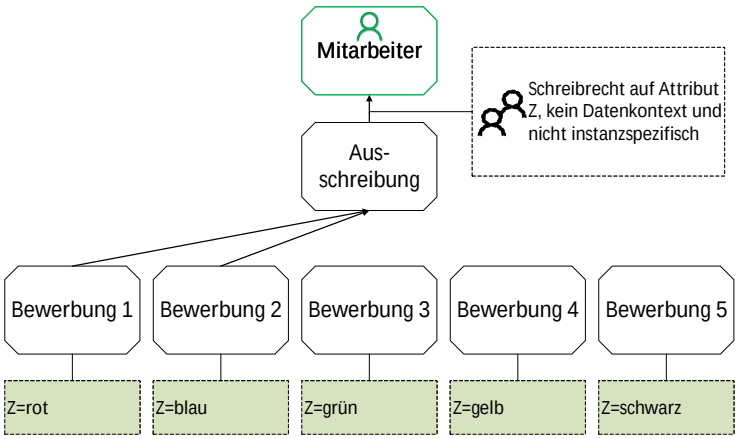

Abbildung 2.24: Abbildung zu Beispiel 4.

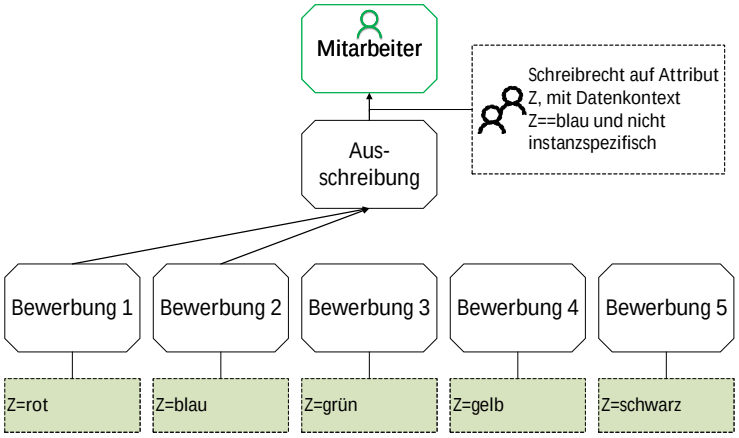

Abbildung 2.25: Abbildung zu Beispiel 5.

Beispiel 5 zeigt eine Relationrolle auf der Relation **Ausschreibung–>Mitarbeiter** mit einem Schreibrecht auf das Attribut **Z** des Objekttyp **Bewerbung**. Das Recht ist nicht instanzspezifisch und besitzt einen Datenkontext **Z==blau**. Der Benutzer erhält Schreibrechte auf das Attribut **Z** der Objektinstanzen **Bewerbung 1**, **Bewerbung 2**,

Bewerbung 3, **Bewerbung 4** und **Bewerbung 5**. Da das Recht nicht instanzspezifisch ist reicht es aus, dass eine Objektinstanz ein Attribut **Z** mit Wert blau besitzt um das Recht für alle Instanzen zu gewähren. Das Beispiel wird in Abbildung 2.25 illustriert.

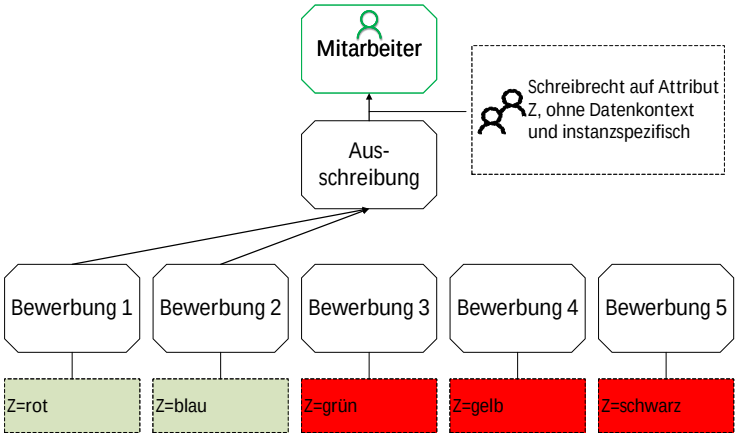

Abbildung 2.26: Abbildung zu Beispiel 6.

Beispiel 6 zeigt eine Relationrolle auf der Relation **Ausschreibung–>Mitarbeiter** mit einem Schreibrecht auf das Attribut **Z** des Objekttyp **Bewerbung**. Das Recht ist instanzspezifisch und besitzt keinen Datenkontext. Der Benutzer erhält Schreibrechte auf das Attribut **Z** der Objektinstanzen **Bewerbung 1** und **Bewerbung 2**. Bei Relationenrollen heißt instanzspezifisch, dass das Recht nur auf Objektinstanzen wirkt, die transitiv oder direkt mit der Userinstanz verbunden sind. Das Beispiel wird in Abbildung 2.26 illustriert.

Beispiel 7 zeigt eine Relationrolle auf der Relation **Ausschreibung–>Mitarbeiter** mit einem Schreibrecht auf das Attribut **Z** des Objekttyp **Bewerbung**. Das Recht ist instanzspezifisch und besitzt einen Datenkontext **Z==blau**. Der Benutzer erhält Schreibrechte auf das Attribut **Z** der Objektinstanzen **Bewerbung 2**. Damit muss jetzt der Datenkontext erfüllt sein und es muss eine Verbindung zum User Type bestehen. Das Beispiel wird in Abbildung 2.27 illustriert.

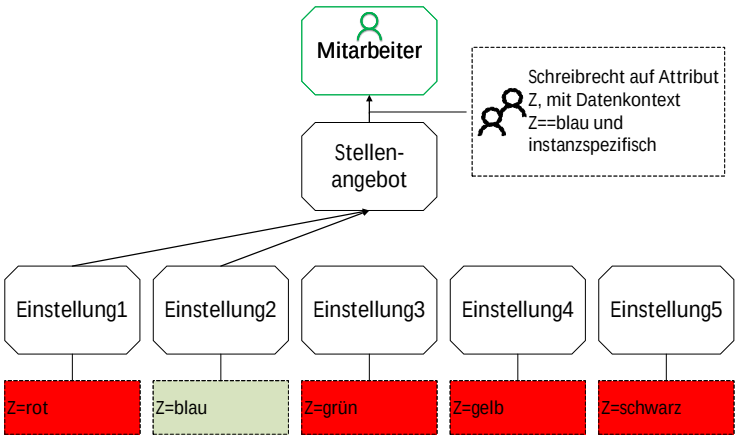

Abbildung 2.27: Abbildung zu Beispiel 7.

3

Anforderungen

In diesem Kapitel werden die zu erfüllenden Anforderungen definiert, welches die Prozessmodelle erfüllen müssen. Anhand dieser Anforderungen erfolgt im Kapitel 4 die Modellierung des Employee-Self-Services und der Fehlzeitenverwaltung. Weiterhin werden die hier definierten Anforderungen als Grundlage der Evaluation, der Fähigkeiten von PHILharmonicFlows herangezogen.

3.1 Employee Self Service und Fehlzeitenverwaltung

In Unternehmen, welche mehrere Mitarbeiter beschäftigen, fällt ein gewisser Aufwand an, diese Mitarbeiter und die entsprechenden Stammdaten zu verwalten. Der Employee-Self-Service von Persis stellt ein Modul der Persis Unternehmenssoftware dar, das diese Aufgabe vereinfacht.

Mithilfe des ESS-Modules soll es möglich sein, zu jedem Mitarbeiter persönliche Daten, Bankverbindungen, Ansprechpartner, private sowie betriebliche Kontaktdaten, Bildungsweg, Berufsweg und ein Mitarbeiterfoto zu verwalten. Da das aktuelle Modul, welches in der Persis Unternehmenssoftware verwendet wird, bereits seit längerer Zeit existiert und die vom Kunden gewünschte Funktionalität nicht in vollem Umfang bietet, entwickelt die Firma Persis den Employee-Self-Services (ESS) neu und hat dazu Anforderungen zu dessen Funktionalität definiert.

Ein weitere Aspekt in Unternehmen, welcher Daten von jedem Mitarbeiter betrifft, ist, wie Fehlzeiten und Urlaubsansprüche von Mitarbeitern verwaltet werden. Diese Verwaltung

ist fachlich sehr an die Verwaltung von Mitarbeiterdaten gebunden, so dass es sich anbietet, diese Fehlzeitenverwaltung (FZ) zusammen mit dem ESS neu zu gestalten.

Für eine Neuentwicklung der Module ESS und FZ wurden von der Persis GmbH interne Anforderungsdokumente ausgearbeitet. Auf diesen Dokumenten beruhen die im Folgenden beschriebenen Anforderungen zur Modellierung des ESS-Systemes mit Fehlzeitenverwaltung mittels PHILharmonicFlows.

Die hier definierten Anforderungen werden weiterhin verwendet um in Kapitel 5 die Mächtigkeit der Modelierungskonzepte von PHILharmonicFlows in Bezug auf die Umsetzbarkeit der Anforderungen zu evaluieren.

3.2 Employee Self Service (ESS)

Nachfolgend werden die zum ESS aus den internen Dokumenten der Persis GmbH herausgearbeiteten Anforderungen unterteilt in die Kategorien Anforderungen an das ESS aus Datensicht, Anforderungen an das ESS Rollendefinition und Anforderungen an das ESS aus Prozesssicht dargestellt

3.2.1 Anforderungen an das ESS aus Datensicht

In diesem Abschnitt werden die Anforderungen zum ESS aus Datensicht dargestellt. diese Anforderungen beschreiben welche Daten das System verarbeiten soll. Weiter wird auf die Repräsentation, Eigenschaften sowie Zusammenhänge dieser Daten eingegangen.

Anf. 1.1.1 Antrag auf Datenänderung [2]: Ein Antrag auf Datenänderung besteht aus den Feldern: Persönliche Daten, Bankverbindung, Bildungsweg, Berufsweg, Seminare, Kenntnisse, Kontaktpersonen, Passbild. Weiter wird zu jedem Antrag der Antragsteller sowie ein Status abgespeichert, welcher wie folgt lauten kann: Offen, eingereicht, abgelehnt, genehmigt, zurückgezogen.

Anf. 1.1.2 Mitarbeiterdaten [2]: Die persönlichen Daten eines Mitarbeiters bestehen aus: Titel, Betriebstitel, Name, Vorname, Namenszusatz, Vorsatzwort, Geburtsname, Geburtsnamenvorsatz, Geburtsnamenzusatz, Geburtstag, Geburtsort, Geburtsland, Staatsangehörigkeit, Staatsangehörigkeit 2, Konfession, Familienstand, Familienstand seit/bis, Anzahl Kinder sowie die Anzahl Kinder unter 18.

Anf. 1.1.3 Privatanschrift [2]: Die private Anschrift eines Mitarbeiters beinhaltet: Straße, Hausnummer, Postfach, Region, Land, Bundesland, PLZ sowie den Wohnort. Weiter gibt es die Möglichkeit, eine zweite Privatanschrift mit denselben Attributtypen zu hinterlegen. Diese wird dann Privatanschrift 2 genannt.

Anf. 1.1.4 Persönliche Kontaktdaten des Mitarbeiters [2]: Die privaten Kontaktdaten beinhalten folgende Attribute: Telefon, Kurzwahl, Telefax, Handy sowie eine E-Mail. Weiterhin gibt es die Möglichkeit, zusätzliche persönliche Kontaktdaten mit denselben Attributtypen zu hinterlegen. Diese werden dann persönliche Kontaktdaten 2 genannt.

Anf. 1.1.5 Dienstliche Kontaktdaten des Mitarbeiters [2]: Die dienstlichen Kontaktdaten bestehen aus den folgenden Attributen: Telefon, Kurzwahl, Telefax, Handy, E-Mail, Gebäude, Büro, Ausweis sowie einem Zutrittsprofil.

Anf. 1.1.6 Bankverbindungen [2]: Jeder Mitarbeiter kann mehrere Bankverbindungen haben. Zu jeder der vorhandenen Bankverbindungen sollen folgende Werte verwaltet werden: Gültig von, Gültig bis, BLZ, Konto-Nr., Bankname, IBAN welche automatisch aus Konto Nr. und BLZ ermittelt wird, BIC welche ebenfalls automatisch ermittelt wird, Kontoinhaber, Konto-Art, Anschrift, PLZ, Ort, Postfach PLZ sowie einem Ansprechpartner.

Anf. 1.1.7 Berufsweg [2]: Für jeden Mitarbeiter können mehrere Berufswege erfasst werden. Zum Berufsweg werden jeweils folgende Werte abgelegt: Datum von, Datum

bis, Jahre, Art, Studienrichtung, Berufsbezeichnung, Fachrichtung, Schlüssel, Firma, Bereich, Ort sowie eine Bemerkung.

Anf. 1.1.8 Bildungsweg [2]: Für jeden Mitarbeiter können mehrere Bildungswege erfasst werden: Zu jedem Bildungsweg gehören Angaben über: Datum von, Datum bis, Jahre, Art, Abschluss, Schlüssel, Zusatz, Bemerkung, Studienrichtung, Studiengang, Abschlussthema, Abschlussklasse, Note sowie einem Ort.

Anf. 1.1.9 Kenntnisse [2]: Für jeden Mitarbeiter können mehrere Kenntnisse erfasst werden, zu jeder Kenntnis gehören Angaben über: Gruppe, Kenntnis, Einschätzung sowie eine Bewertung.

Anf. 1.1.10 Kontaktpersonen [2]: Für jeden Mitarbeiter ist es möglich mehrere Kontaktpersonen zu hinterlegen. Zu diesen sollten jeweils die im folgenden genannten Attribute verwaltet werden: Status, Geschlecht, Titel, Name, Vorname, Namenszusatz, Ort, PLZ, Telefon, Kurzwahl, Telefax, Handy, E-Mail, Mitarbeiterverknüpfung und eine Bemerkung.

Anf. 1.1.11 Passbild [2]: Jedem Mitarbeiter muss ein Bild zugeordnet werden können. Zusätzlich sollte es möglich sein, das Bild zu löschen oder zu sperren.

Anf. 1.1.12 Seminare [2]: Für jeden Mitarbeiter können mehrere Seminare erfasst werden. Zu einem Seminar gehören die folgenden Daten: Beginn, Ende, Gruppe, Seminar, Titel der Weiterbildungsmaßnahme, Kosten, Veranstalter, Veranstalter Name /Anschrift, Veranstalter PLZ, Ort, Bemerkung sowie ein Dokument.

Anf. 1.1.13 Darstellung von Änderungen [2]: Jegliche Datenänderung muss nachvollziehbar sein. Bei mehreren Datensätzen müssen gelöschte Datensätze und neu eingefügte Datensätze ersichtlich sein.

3.2.2 Anforderungen an das ESS Rollendefinitionen

Der folgende Abschnitt beschreibt Anforderungen zur ESS Rollendefinition. Das heißt, es wird darauf eingegangen, welche Benutzer es geben soll und welche Rechte die einzelnen Benutzer besitzen.

Anf. 1.2.1 Rolle MA [2]: Mitarbeiter mit der Rolle MA sind alle Mitarbeiter die im Unternehmen angestellt sind. Im ESS haben diese Mitarbeiter keine weiteren Rechte. Den Bezug zu den anderen Rollen im System illustriert Diagramm 3.1.

Anf. 1.2.2 Rolle MA-ESS [2]: Mitarbeiter mit der Rolle MA-ESS sollen zusätzlich zu den Rechten der Rolle MA berechtigt sein, folgende Rubriken zu editieren: Mitarbeiterdaten, Bankverbindungen, Bildungsweg, Berufsweg, Seminare, Kenntnisse, Kontaktpersonen, Passbild. Für jede dieser Rubriken existiert zukünftig ein eigener Antrag auf Datenänderung. Ist ein Antrag erfolgt, wird er jeweils vom zuständigen Personalsachbearbeiter geprüft und die Änderung von diesem übernommen. Sowohl der Mitarbeiter selbst als auch der zuständige Sachbearbeiter müssen zu jedem Zeitpunkt einen Überblick und Einblick in die Datenänderungen haben. Den Bezug zu den anderen Rollen im System illustriert Diagramm 3.1.

Anf. 1.2.3 Rolle MA-Verwaltung [2]: Mitarbeiter mit der Rolle MA-Verwaltung dürfen zusätzlich zur Rolle MA-ESS Anträge prüfen und diese genehmigen, ablehnen oder dem Mitarbeiter zur Überarbeitung zurückgeben. Hierbei ist noch zu unterscheiden, ob er auch seine eigenen Anträge genehmigen darf, (Rolle MA-Verwaltung-Chef) oder nur Anträge anderer Mitarbeiter (Rolle MA-Verwaltung). Den Bezug zu den anderen Rollen im System illustriert Diagramm 3.1.

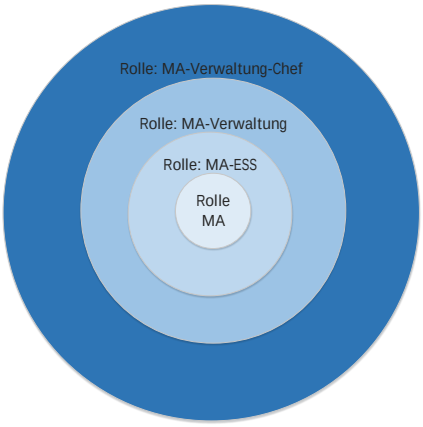

Abbildung 3.1: Darstellung der Überlappung der Rollen im ESS.

3.2.3 Anforderungen an das ESS aus Prozesssicht

Der folgende Abschnitt beschreibt Anforderungen an das ESS aus Prozesssicht. Das heißt, es wird darauf eingegangen, was für Bearbeitungsabläufe es im System geben soll.

Anf. 1.3.1 Zustände von Anträgen auf Datenänderung [2]: Die im folgenden abgebildete Graphik 3.2 verdeutlicht die Möglichkeiten der Zustandsfolge eines Antrages.

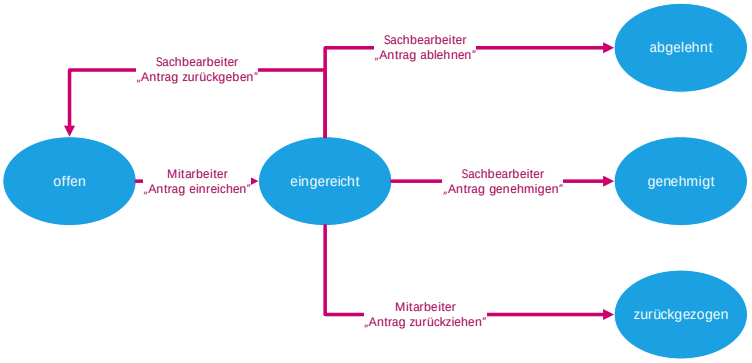

Abbildung 3.2: Zustandsdiagramm des ESS

Die Tabelle 3.1 zeigt im Folgenden die im System vorhandenen Zustände von Anträgen sowie deren Zustandseigenschaften auf.

Tabelle 3.1: Zustände des ESS und deren Eigenschaften

Zustand	**Beschreibung**
offen	-der Antrag liegt in der Zuständigkeit des Mitarbeiters -die Änderungen müssen vervollständigt und der Antrag eingereicht werden (Folgezustand: eingereicht) -diesen Zustand bekommt ein Antrag auch nach Rückgabe des Antrags vom Sachbearbeiter an den Mitarbeiter
eingereicht	-der Antrag liegt in der Zuständigkeit des Personalsachbearbeiters -die Änderungen müssen geprüft und übernommen werden (Folgezustand: genehmigt) -der Sachbearbeiter kann die Änderungen auch ablehnen (Folgezustand: abgelehnt) -der Mitarbeiter hat in diesem Zustand die Möglichkeit seinen Antrag zurückzuziehen (Folgezustand: zurückgezogen) -der Sachbearbeiter kann den Antrag zur Überarbeitung (beliebig oft) an den Mitarbeiter zurückgeben (Folgezustand: offen)
genehmigt	-der Antrag ist abgeschlossen und die Änderungen wurden übernommen -Zustand dient der Archivierung
abgelehnt	-der Antrag ist abgeschlossen und die Änderungen wurden verworfen (vom Personalsachbearbeiter) -Zustand dient der Archivierung
zurückgezogen	-der Antrag ist abgeschlossen ohne dass die Änderungen übernommen wurden (vom Mitarbeiter) -Zustand dient der Archivierung

3.3 Fehlzeitenverwaltung

Nachfolgend werden die zur FZ aus den internen Dokumenten der Persis GmbH herausgearbeiteten Anforderungen unterteilt in die Kategorien Anforderungen an die FZ aus Datensicht, Anforderungen an die FZ Rollendefinition und Anforderungen an die FZ aus Prozesssicht dargestellt

3.3.1 Anforderungen zur Fehlzeitenverwaltung aus Datensicht

In diesem Abschnitt werden die Anforderungen an die FZ aus Datensicht dargestellt. Diese Anforderungen beschreiben, welche Daten das FZ-System verarbeiten soll. Weiter wird auf die Repräsentation, Eigenschaften sowie Zusammenhänge dieser Daten eingegangen.

Anf. 2.1.1 Fehlzeitenantrag [3]: Je Fehlzeitantrag soll der Name des Antragstellers, die Antragsart und der Antragszeitraum bestehend aus **Datum von** sowie **Datum bis** angegeben werden. Im Anschluss auf die Eingabe des Zeitraumes soll für jeden Arbeitstag im angegebenen Zeitraum gewählt werden, ob der Tag aus dem Antrag gestrichen werden soll oder ob die Fehlzeit nur den Vormittag, nur Nachmittag oder den ganzen Tag betrifft. Weiter soll automatisch das Datum der Antragstellung mit abgespeichert werden. Eine Bemerkung des Antragstellers ist optional.

Anf. 2.1.2 Urlaubskonto [3]): Für jedes Jahr und für jeden Mitarbeiter existiert ein separates Urlaubskonto. In diesem Urlaubskonto wird der jeweils verbleibende Resturlaub gespeichert. Auf diesen Resturlaubsanspruch werden die neuen genehmigten Anträge abgezogen und Stornoanträge addiert, so dass im Attribut Urlaubsanspruch immer der verfügbare Resturlaub des Mitarbeiters verwaltet wird. Als Einheit für den Urlaubsanspruch werden Tage mit einer Nachkommastelle auf halbe Arbeitstage beschränkt verwendet.

Anf. 2.1.3 Vertreter [3]): Im Fehlzeitenantrag kann ein Vertreter benannt werden. Dieser Vertreter muss zustimmen. Lehnt er ab, so muss er zwangsweise eine Bemerkung als Begründung verfassen. Stimmt er jedoch zu, so ist die Bemerkung optional. Das Entscheidungsdatum des Vertreters soll automatisch mit gespeichert werden.

Anf. 2.1.4 Entscheidung [3]): Jede Fehlzeit muss genehmigt werden. Je nach Position des Mitarbeiters muss die Fehlzeit von einem oder von mehreren berechtigten Mitarbeitern genehmigt werden. Zu jeder Entscheidung soll automatisch das Entscheidungsdatum mit der aktuellen Urzeit gespeichert werden. Es steht auch hier ein Bemerkungsfeld zur Verfügung, welches im Falle einer Ablehnung verpflichtend ist, bei einer Zustimmung jedoch optional.

Anf. 2.1.5 Mitarbeiterkalender [3]): Der Mitarbeiterkalender repräsentiert alle genehmigten Fehlzeiten aller Mitarbeiter im Unternehmen. Auf Wunsch lassen sich offene Anträge mit einblenden, wobei Überschneidungen mit anderen offenen Anträgen eines Antragstellers graphisch hervorgehoben werden können.

Anf. 2.1.6 Stornierung [3]): Eine Stornierung steht immer im Bezug zum ursprünglichen Antrag und besitzt alle Attribute des ursprünglichen Antrages. Weiterhin soll für den genehmigten Urlaubszeitraum je halbem Tag eingegeben werden können, ob dieser storniert oder als Fehlzeit beibehalten werden soll. Optional kann eine Bemerkung verfasst werden.

Anf. 2.1.7 Mitarbeiter [3]): Zu jedem Mitarbeiter soll gespeichert werden, welche anderen Mitarbeiter entscheidungsberechtigt für Fehlzeiten sind.

3.3.2 Anforderungen zur Fehlzeitenverwaltung Rollendefinitionen

Der folgende Abschnitt beschreibt Anforderungen zur FZ Rollendefinition. Das heißt es wird darauf eingegangen welche Benutzer es im System geben soll und welche Rechte die einzelnen Benutzer besitzen.

Anf. 2.2.1 Rolle MA, Rolle MA-ESS, Rolle MA-Verwaltung, Rolle MA-Verwaltung-Chef [3]: Mitarbeiter mit den Rollen MA, MA-ESS, MA-Verwaltung und der Rolle MA-Verwaltung-Chef sind berechtigt, Fehlzeitenanträge zu stellen und eine Vertretung anzunehmen, wenn sie als Vertretung benannt wurden. Mitarbeiter mit mindestens der Rolle MA-Verwaltung können zudem entscheiden, welche Mitarbeiter aus der Liste an Entscheidern, die zu jedem Mitarbeiter angelegt ist, einem konkreten Fehlzeitantrag zustimmen müssen.

Anf. 2.2.2 Rolle Entscheider [3]: Jeder Mitarbeiter mit einer der Rollen MA, MA-ESS, MA-Verwaltung und der Rolle MA-Verwaltung-Chef kann zusätzlich die Rolle Entscheider erlangen. Bedingung hierfür ist, dass er als Entscheider für Fehlzeitenanträge zur antragsstellenden Person definiert wurde und dass er zu einem aktuellen Antrag verlinkt wurde. Jedoch darf kein Mitarbeiter seinen eigenen Antrag genehmigen.

3.3.3 Anforderungen zur Fehlzeitenverwaltung aus Prozesssicht

Der folgende Abschnitt beschreibt Anforderungen zur FZ aus Prozesssicht. Das heißt, es wird darauf eingegangen, welche Bearbeitungsabläufe es im FZ-System geben soll.

Anf. 2.3.1 Fehlzeitenantrag [3]: Ein Fehlzeitenantrag kann von jedem Mitarbeiter im Unternehmen gestellt werden. Zu Beginn eines Antrages muss die Art der Fehlzeit ausgewählt werden. Nach Eingabe der Fehlzeitdaten wird vom System geprüft ob, das Datum des Antrages über das Jahresende geht. Ist dies der Fall, geht das System in einen Fehlerzustand über und meldet dem Benutzer, dass der Antrag gesplittet für beide Jahre separat einzugeben ist. Der Mitarbeiter kann seine Antragsdaten korrigieren. Ist der Zeitraum in Ordnung, kann für jeden Arbeitstag im Antragszeitraum separat angegeben werden, ob die Fehlzeit nur den Vormittag, nur den Nachmittag betrifft oder ob der Tag von der Fehlzeit nicht betroffen ist. Sind diese Angaben vorhanden, prüft das System ob der vorhandene Urlaubsanspruch ausreichend ist. Ist dies der Fall, schreitet der Antrag weiter fort und ist nun im Zustand 'Vertreter gewünscht'. Ist dies nicht der

Fall, so wird eine Fehlermeldung ausgegeben und der Mitarbeiter kann seinen Antrag korrigieren.

Anf. 2.3.2 Vertreterregelung [3]: Ist ein Antrag im Zustand **Vertreter gewünscht**, muss der Antragsteller entscheiden, ob er für seine Abwesenheit einen Vertreter benennen will. Entscheidet sich der Antragsteller für einen Vertreter, kann dieser aus der Liste aller Mitarbeiter ausgewählt werden. Ist ein Vertreter ausgewählt, wird dieser per Mail benachrichtigt und muss seine Zustimmung abgeben. Stimmt der Vertreter zu, geht das System in den Zustand 'Entscheidung' über, stimmt der Vertreter nicht zu, muss dieser eine Bemerkung verfassen und der Antragssteller kann eine andere Person als Vertreter auswählen.

Anf. 2.3.3 Entscheidung [3]: Anträge, welche keinen Vertreter benötigen, und Anträge, bei denen der Vertreter zugestimmt hat, gehen in den Zustand Entscheidung über. Hier kann ein Mitarbeiter mit der Rolle MA-Verwaltung alle benötigten Entscheider aus einer Liste an potentiellen Entscheidern, welche zu jedem Mitarbeiter verwaltet wird, wählen. All diese Entscheider müssen eine Zustimmung abgeben. Haben alle gewählten Entscheider zugestimmt, ist der Antrag 'genehmigt' und das Urlaubskonto wird aktualisiert. Hat mindestens ein Entscheider nicht zugestimmt, ist der Antrag 'abgelehnt'. Der Antragsteller wird über den neuen Status per Mail informiert. Im Falle einer Ablehnung wird weiterhin ein eventuell involvierter Vertreter per Mail benachrichtigt.

Anf. 2.3.4 Stornierung [3]: Anträge, welche genehmigt wurden, können ganz- oder halbtagesweise storniert werden. Eine Stornierung muss durch einen Mitarbeiter mit der Rolle MA-Verwaltung genehmigt werden. Wird eine Stornierung genehmigt, werden, falls vorhanden, der Vertreter sowie alle involvierten Entscheider per Mail benachrichtigt. Das Urlaubskonto wird angepasst.

Anf. 2.3.5 Zurückziehen eines Antrages [3]: Anträge, welche noch nicht genehmigt wurden, können vom Antragsteller zurückgezogen werden. Involvierte Vertreter werden per Mail benachrichtigt.

Anf. 2.3.6 Mitarbeiterkalender [3]: Der Mitarbeiterkalender zeigt immer den aktuellen Stand an genehmigten Fehlzeiten an. Optional können beantragte Fehlzeiten eingeblendet werden.

4

Darstellung der Modellierung

In diesem Kapitel wird die Modellierung des Employee-Self-Service sowie des Fehlzeitenverwaltung mittels PHILharmonicFlows dargestellt.

4.1 Darstellung der Modellierung

Dieses Kapitel stellt die Modellierung der Persis Module und Fehlzeitenverwaltung mit PHILharmonicFlows dar. Anhand dieser Modellierungsergebnisse wird im Zusammenspiel mit den Anforderungen aus Kapitel 3 im Kapitel 5 die Evaluation der Modellierungskonzepte von PHILharmonicFlows durchgeführt.

4.1.1 Objekttypen und Datenmodell

Die Module ESS und FZ wurden in einem gemeinsamen Modell modelliert. Abbildung 4.1 zeigt das Datenmodell zu den Prozessen. Das Datenmodell besteht aus drei Datenebenen. In der obersten Ebene ist ein Benutzertyp, welcher die Mitarbeiter eines Unternehmens repräsentiert. Dieser Mitarbeiterobjekttyp wird sowohl für die Funktionalitäten des ESS wie auch der FZ verwendet, da beide Systeme Mitarbeiterdaten aus demselben Mitarbeiterstamm im Unternehmen verarbeiten. In der Ebene darunter liegen Objekttypen, welche die aktuellen Daten zu jedem Mitarbeiter repräsentieren. Jeder Mitarbeiter besitzt genau einmal **persönliche Daten** (1:1 Beziehung) und jeweils beliebig viele **Bankverbindungen**, **Bildungswege**, **Berufswege**, **Seminare**, **Kenntnisse**, **Kontaktpersonen** und **Passbilder** (1:n Beziehungen).

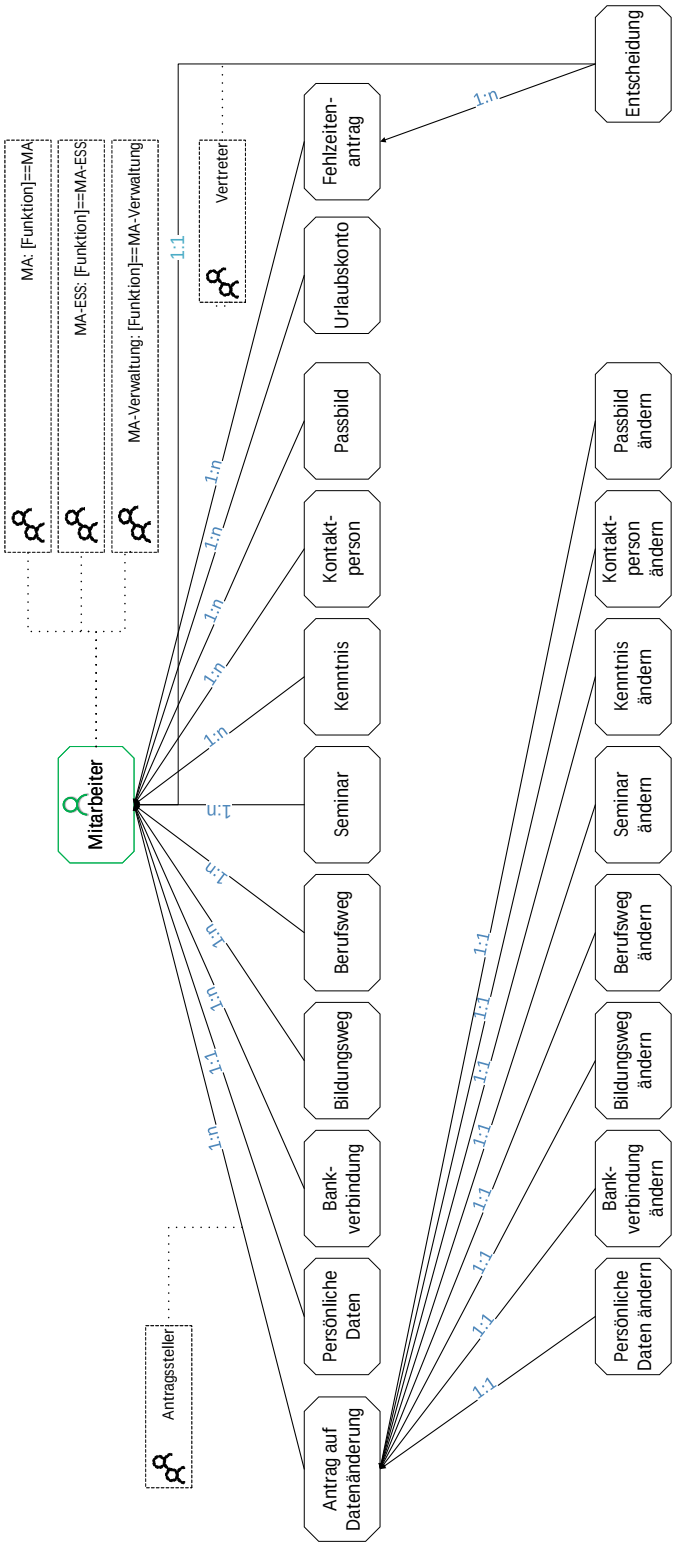

Abbildung 4.1: Datenmodell zum Employee-Self-Service und Fehlzeitenverwaltung

Im Objekt **Urlaubsanspruch** wird für jeden Mitarbeiter verwaltet, wie sich sein Urlaubsanspruch für ein bestimmtes Jahr darstellt. Jeder Mitarbeiter besitzt somit mehrere Urlaubsanspruchsobjekte (1:n Beziehung). Über das Objekt **Fehlzeitenantrag** kann ein Fehlzeitenantrag des FZ Modules und über das Objekt **Antrag auf Datenänderung** ein Datenänderungsantrag des ESS Modules gestartet werden. In der dritten Ebene liegen Objekte, die Komponenten zu dem Basisantrag verwalten. Das sind die Objekte **persönliche Daten ändern**, **Bankverbindung ändern**, **Bildungsweg ändern**, **Berufsweg ändern**, **Seminar ändern**, **Kenntnis ändern** und **Kontaktperson ändern**. Dies Objekte können einem Datenänderungsantrag des ESS-Modules zugeordnet werden. Da ein Antrag auf Datenänderung immer aus genau einer Änderungsversion besteht, wird hier eine 1:1 Beziehung verwendet. Weiter repräsentiert das Objekt **Entscheidung**, eine Entscheidung eines Entscheiders zu einem Fehlzeitenantrag des FZ- Modules. Da es sein kann, dass zu einem Antrag mehrere Entscheider eine Enscheidung abgeben, besteht hier eine 1:n Beziehung. Zu jeder Entscheidung gehört aber genau ein Entscheider, somit besteht hier eine 1:1 Beziehung. Weiterhin wurden im Datenmodell 3 Kontextrollen (MA, MA-ESS und MA-Verwaltung) und zwei Relationenrollen definiert. Welche Rechte diese Rollen erhalten wird später erläutert.

Das Datenmodell besitzt einige Objekttypen, welche sich sehr ähnlich sind. So unterscheiden sich die Objekttypen **Bankverbindung**, **Bildungsweg**, **Berufsweg**, **Seminar**, **Kenntnis** und **Kontaktperson** nur durch ihre fachliche Ausrichtung und ihren Attributnamen. Die Prozesslogik, welche zu den genannten Objekttypen gehört, ist jedoch immer dieselbe. Genauso verhält es sich auch bei den Objekttypen **Bankverbindung ändern**, **Bildungsweg ändern**, **Berufsweg ändern**, **Seminar ändern**, **Kenntnis ändern** und **Kontaktperson ändern**. Wir beschränken uns daher zukünftig auf die Verwendung von **Bankverbindung** und **Bankverbindung ändern**, da mit diesen beiden Typen alle Problematiken, welche bei der Modellierung aufgetreten sind, erläutert werden können.

Zu jedem Objekttyp gehören eine Vielzahl an Attributen. Da diese Attributliste sehr umfangreich ist, wurde diese hier aus Darstellungsgründen nicht graphisch mit im Datenmodell dargestellt. Stattdessen wurde eine tabellarische Darstellung gewählt, welche als Anhang zu dieser Arbeit angefügt wurde. Auf die Darstellung der Attribute zu den Objekttypen **Bildungsweg**, **Berufsweg**, **Seminar**, **Kenntnis**, **Kontaktperson**,

Bildungsweg ändern, **Berufsweg ändern**, **Seminar ändern**, **Kenntnis ändern** und **Kontaktperson ändern** wurde aus den genannten Gründen verzichtet. Im Anhang zeigt die Tabelle 1 die Attribute der einzelnen Objekttypen auf.

4.1.2 Mikroprozesse

In diesem Abschnitt werden die modellierten Mikroprozesse des ESS und der FZ erläutert. Es wird im folgenden auf alle Mikroprozesse eingegangen mit Ausnahme der zu den im vorigen Abschnitt ausgeschlossenen Objekttypen.

Mikroprozess Mitarbeiter

Ein erster und zugleich sehr einfacher Mikroprozess gehört zum Benutzertyp **Mitarbeiter**. Dargestellt wird dieser in Abbildung 4.2. Mit Ausnahme des Start- und Endzustandes gibt es nur einen weiteren Zustand welcher dafür sorgt, dass alle Attribute, die ein Mitarbeiter zwangsläufig besitzen muss, geschrieben werden. Dies sind als erstes der **Benutzername** und das **Passwort** um sich am System anmelden zu können. Das verpflichtende Attribut **Funktion** gibt die Stellung des Mitarbeiters im Unternehmen an und bestimmt damit direkt welche Kontextrolle der Mitarbeiter erhält. In der Referenzliste **Entscheider Fehlzeit** werden andere Mitarbeiter referenziert, welche als Entscheider bei einem Fehlzeitenantrag herangezogen werden können.

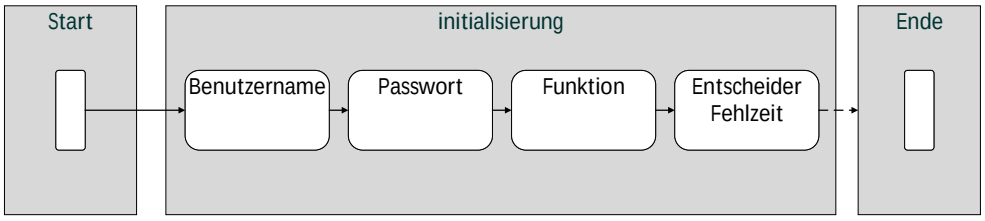

Abbildung 4.2: Mikroprozess zum Objekttyp Mitarbeiter

Mikroprozess Persönliche Daten

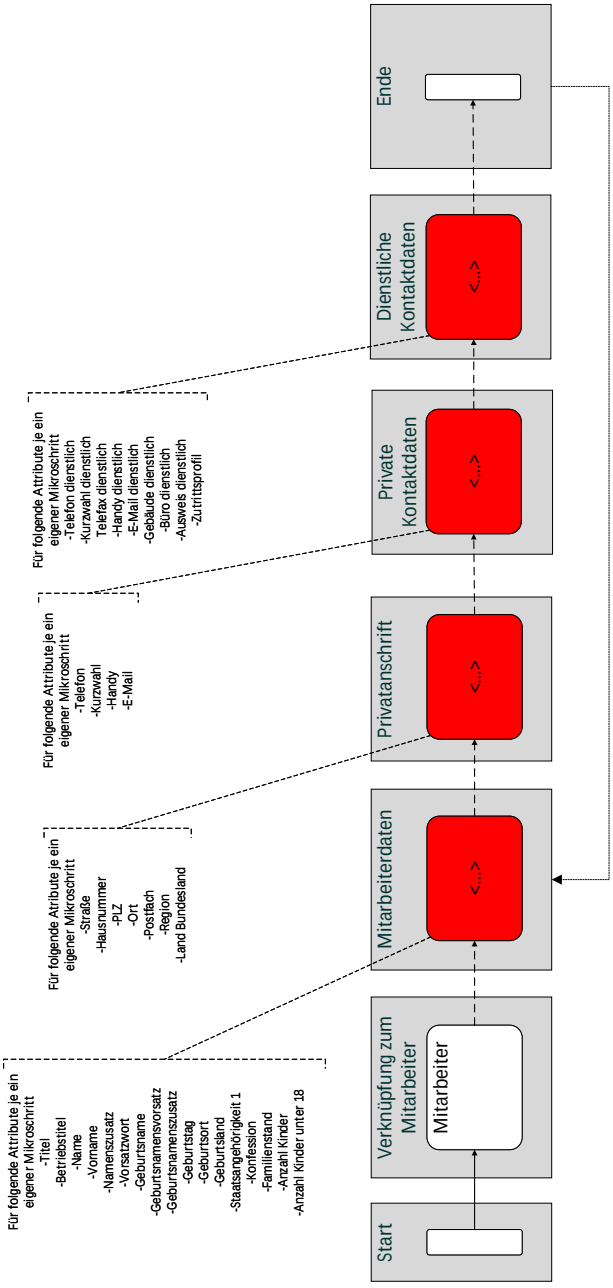

Abbildung 4.3: Mikroprozess zum Objekttyp persönliche Daten

Der in 4.3 abgebildete Mikroprozess gehört zum Objekttyp persönliche Daten. In einem ersten Zustand **Verknüpfung zum Mitarbeiter**, wird durch das Schreiben eines Relattionenatributes eine Relation zu dem Mitarbeiter Objekttyp angelegt, zu dem die persönlichen Daten gehören. Die Zustände **Mitarbeiterdaten**, **Privatanschrift**, **private Kontaktdaten** und **dienstliche Kontaktdaten** gliedern jeweils die Eingabe von verpflichtenden Datensätzen nach inhaltlichen Gesichtspunkten. Da je Zustand extrem viele Mikroschritte vorkommen welche in rein linearer Abfolge das Schreiben von Attributen abhandeln, wurde hierfür aus Darstellungsgründen ein Platzhalter modelliert. Alle Attribute, die in keinem Mikroschritt referenziert werden sind optional für den Prozessablauf. Als Beispiel kann hier die **Staatsangehörigkeit 2** angeführt werden. Die Rückwärtstransition aus dem Endzustand, zurück zu dem Zustand Mitarbeiterdaten, dient dazu, dass die eingegebenen Daten korrigiert werden können.

Mikroprozess Bankverbindung

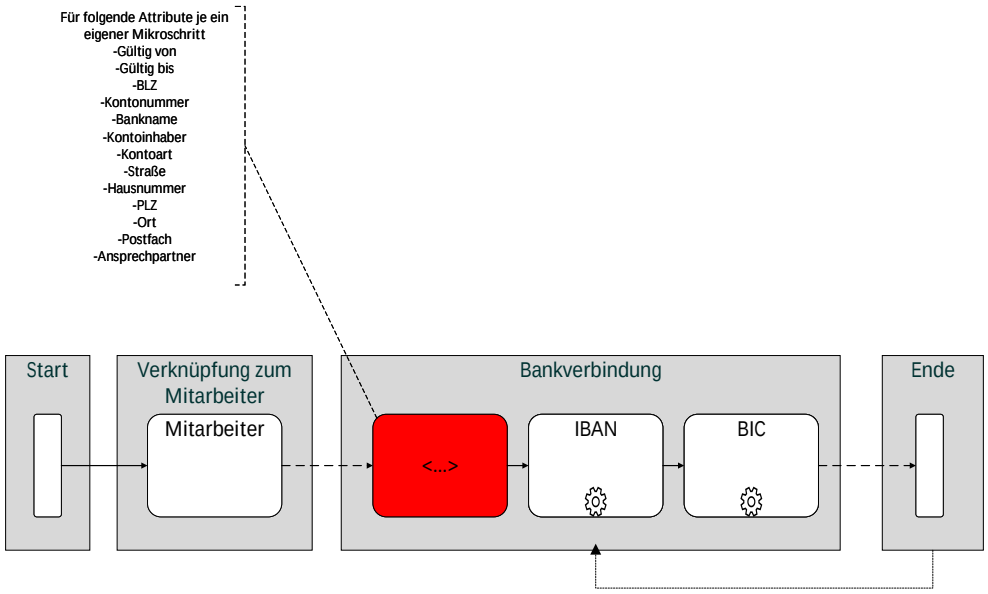

Abbildung 4.4: Mikroprozess zum Objekttyp Bankverbindung

Der Mikroprozess, der eine große Ähnlichkeit zum Mikroprozess **Persönliche Daten** besitzt, gehört zum Objekttyp **Bankverbindung** und wird in Abbildung 4.4 gezeigt. Die Berechnungsschritte **IBAN** und **BIC** sollen einen Mikroschritt darstellen, der seine Werte nicht durch eine Benutzereingabe, sondern durch eine vom System durchgeführte Berechnung aus der eingegebenen **Kontonummer**, **Bankleitzahl** sowie des **Landes** erhält.

Mikroprozess Passbild

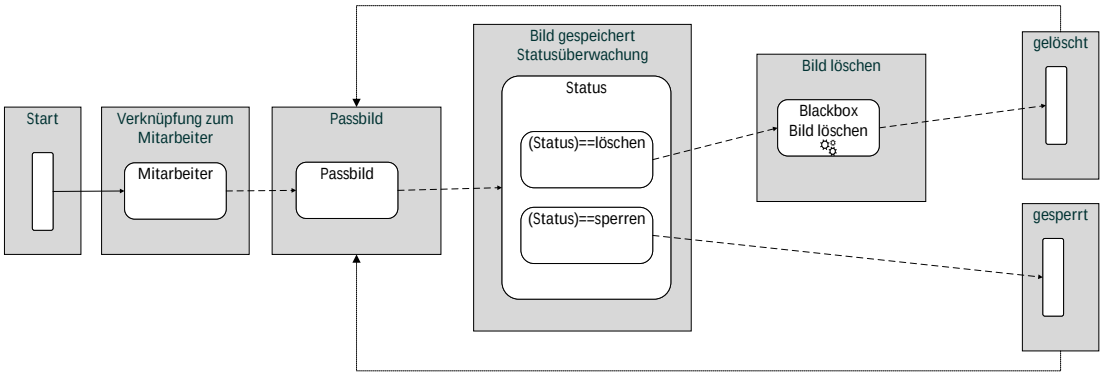

Abbildung 4.5: Mikroprozess zum Objekttyp Passbild

Der Objekttyp, dessen Mikroprozess in Abbildung 4.5 gezeigt wird, unterscheidet sich von den Objekttypen **Persönliche Daten** und **Bankverbindung**. Hier geht es zwar ebenfalls darum, Daten eines Mitarbeiters zu verwalten, jedoch gibt es zu Bilddaten noch erweiterte Funktionalitäten, die im Prozess abzubilden sind. Es wird zuerst im Zustand **Verknüpfung zum Mitarbeiter**, ein Bezug über ein Relationenattribute zu einem Mitarbeiter des Unternehmens hergestellt. Weiter wird im Zustand **Passbild**, ein Bild als Datei in ein File-Attribut eingelesen. Jedoch terminiert der Prozess nicht, wenn die Daten vorhanden sind, da es jederzeit möglich sein soll, ein Bild komplett aus dem System zu entfernen oder zu sperren. Um dies umzusetzen, verbleibt der Prozess nach dem erfolgreichen ablegen eines Bildes im Zustand **Bild gespeichert Statusüberwachung**. Hier besteht die Möglichkeit durch Schreiben des Attributes **Status**, das Bild zu

sperren oder zu löschen. Wird das Attribut **Status** mit dem Wert sperren geschrieben, geht der Objektzustand in den Status **gesperrt** über, das Bild bleibt jedoch im Passbild Attribut gespeichert. Soll das Bild hingegen gelöscht werden, geht das System in den Zustand **Bild löschen** über. Die Blackboxaktivität soll das Passbild löschen, das Attribut **Passwort** mit einem Nullwert befüllen und das Bild auch in allen Änderungsanträgen in denen ebenfalls das Bild abgelegt ist, entfernen. Die beiden Rücksprungtransitionen aus den Zuständen **gelöscht** und **gesperrt** heraus, zurück zum Zustand **Passbild**, dienen dazu eine Sperrung aufzuheben, bzw. nach einer Löschung ein neues Bild einfügen zu können.

Mikroprozess Antrag auf Datenänderung

Der Objekttyp Antrag auf Datenänderung, welcher in Abbildung 4.6 abgebildet ist, verwaltet das Herzstück des ESS Systemes. Im Antrag auf Datenänderung wird die Zustandsfolge eines realen Antrages abgebildet. Für jeden definierten Zustand eines realen Antrages besitzt der Mikroprozess einen eigenen Zustand. Zu Beginn des Mikroprozesses wird die Eingabe auf das Attribut **Art der Änderung** ausgewertet. Durch das Schreiben dieses Attributes wird bestimmt, welche Daten des Mitarbeiters geändert werden sollen. Je nach Wert des Attributes wird in einen eigenen Zustand **spezifische Daten** übergegangen, in dem dann eine Relation mittels Relationenattribut zu einem der Objekte **persönliche Daten ändern**, **Bankverbindung ändern**, **Bildungsweg ändern**, **Berufsweg ändern**, **Seminar ändern**, **Kenntnis ändern** oder **Kontaktperson ändern** hergestellt wird, in Abhängigkeit davon, in welchem Zustand sich der Mikroprozess befindet. Ist die entsprechende Relation angelegt, so kann der Zustand in den Zustand **Status offen** wechseln. Die Zustände **Status offen** und **genehmigt** enthalten jeweils nur einen leeren Mikroschritt, da in den Zuständen keine Attribute geschrieben werden müssen. Diese Zustände dienen später der Koordination durch die Makroprozesse und der expliziten Weiterschaltung durch die Bearbeiter. Ist ein Antrag eingereicht, kann dieser zurückgezogen werden. Wird ein Antrag durch Schreiben des Attributes **zurückziehen?** zurückgezogen, verbleibt der Prozess im Zustand **zurückgezogen** zur Dokumentation des Vorganges.

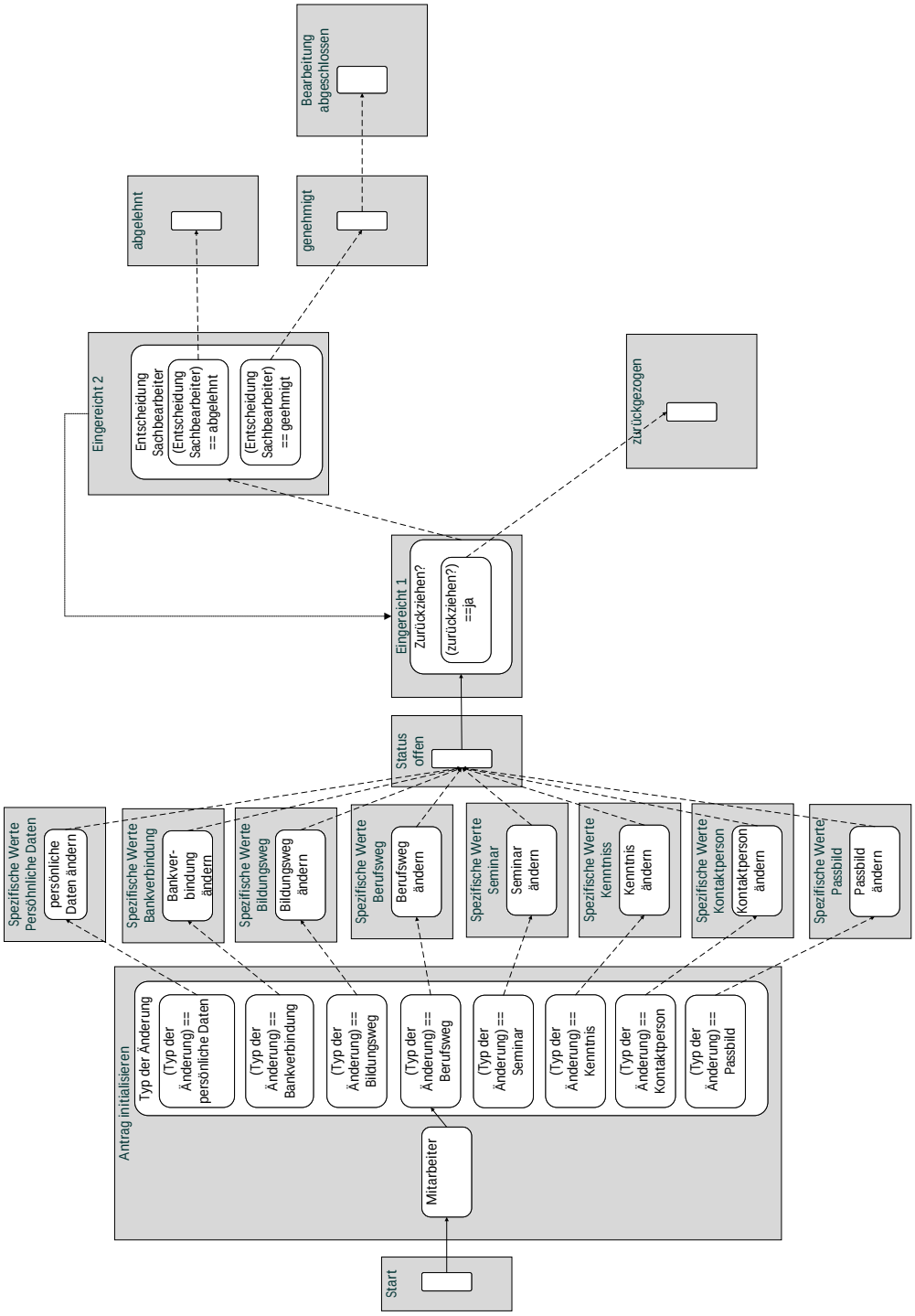

Abbildung 4.6: Mikroprozess zum Objekttyp Antrag auf Datenänderung

Wird der Antrag nicht zurückgezogen, wird auf die Genehmigung durch einen Mitarbeiter der Verwaltung gewartet. Ein Antrag kann nur zurückgezogen werden, solange er nicht genehmigt ist. Dieser Mitarbeiter gibt seine Zustimmung, indem er das Attribut **Entscheidung Sachbearbeiter** mit dem Wert genehmigt schreibt und seine Ablehnung mit dem Wert abgelehnt. Die Rücksprungtransitionen aus **eingereicht 2** zu **eingereicht 1** dient dazu, dass jederzeit das Zurückziehen des Antrages durch den Antragssteller möglich ist, solange dieser nicht genehmigt wurde.

Mikroprozess Persönliche Daten ändern

Der Mikroprozess zum Objekt **Persönliche Daten ändern**, welcher in Abbildung 4.7 dargestellt ist, ähnelt dem Mikroprozess des Objekttypes **persönliche Daten**. Dies kommt daher, dass beide sehr ähnliche Daten verwalten. Im Gegensatz zum Mikroprozess **persönliche Daten** geht es beim Mikroprozess **persönliche Daten ändern** aber darum, Anträge zu verwalten und geänderte Daten einzulesen, welche dann in den Objekttyp **persönliche Daten** übernommen werden sollen, sobald die Änderungen freigegeben wurden. Da PHILharmonicFlows zur Modellierungszeit keine Möglichkeit geboten hat, auf andere Objekttypen und deren Werte zuzugreifen, verwendet der Zustand **persönliche Daten einlesen** einen Platzhalter in Form einer Backboxaktivität. Diese Blackboxaktivität soll die Eigenschaft besitzen, das zum Mitarbeiter gehörende Datenobjekt **persönliche Daten auszulesen** und diese Werte in die Attribute des **persönliche Daten ändern** Objektes zu übernehmen um diese dort in den Folgezuständen editieren zu können. Der Zustand **Warten auf Freigabe**, dient der Koordination mit dem Objekttyp **Antrag auf Datenänderung** über einen Makroprozess, welcher später beschrieben wird. Der Zustand **Update durchführen**, stellt die Umkehrung des Zustandes **persönliche Daten auslesen** dar. Hier sollen nach genehmigter Änderung die neuen Werte in das Objekt persönliche Daten übernommen werden. Dies geschieht wieder über eine Blackboxaktivität. Ist das Update der Daten erfolgreich durchgeführt, verbleibt der Mikroprozess in dem Zustand **Ende** zur Dokumentation der Änderung.

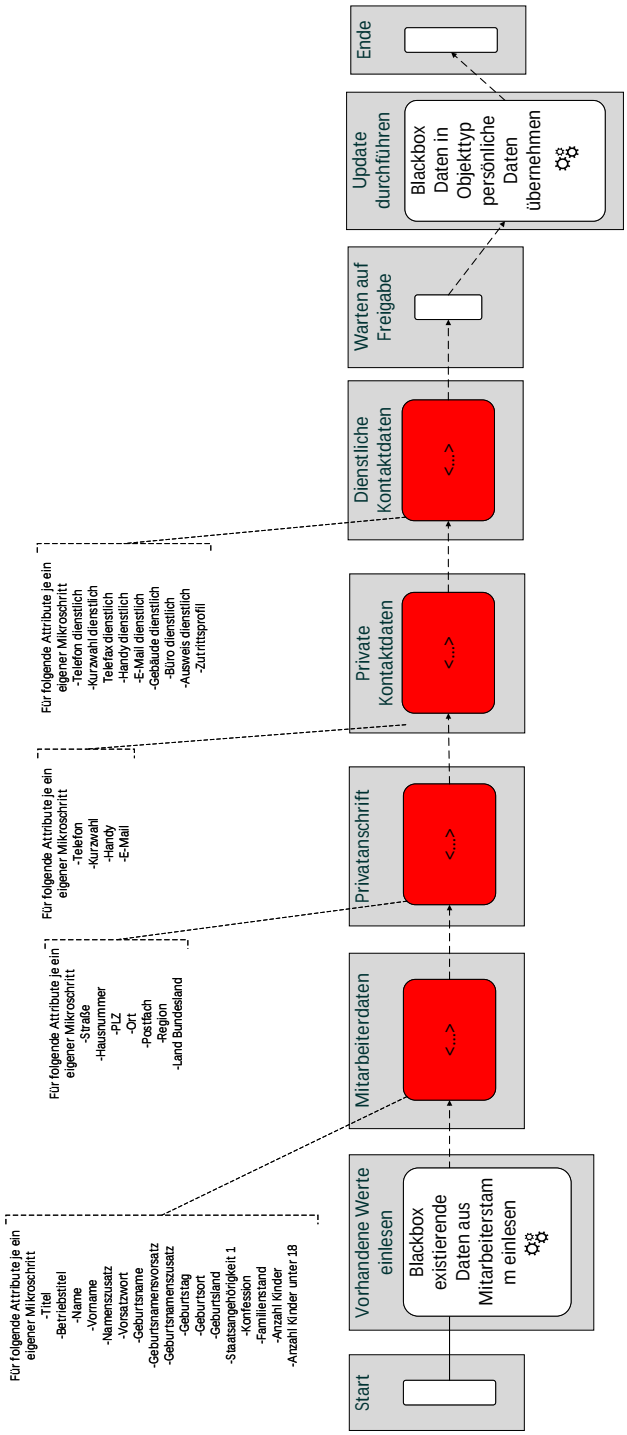

Abbildung 4.7: Mikroprozess zum Objekttyp persönliche Daten ändern

Mikroprozess Bankverbindung ändern

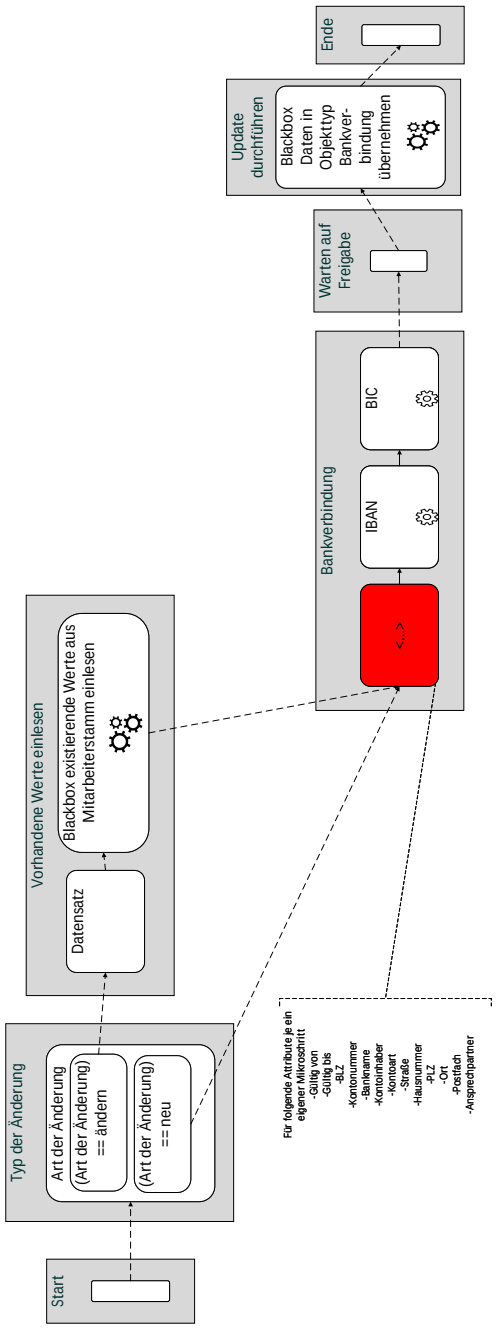

Abbildung 4.8: Mikroprozess zum Objekttyp Bankverbindung ändern

Der Mikroprozess Bankverbindung ändern wird in Abbildung 4.8 illustriert. Dieser Prozess besitzt aus Prozesssicht große Ähnlichkeiten zu dem Prozess **persönliche Daten ändern**. Die Unterschiede im Prozessablauf entstehen dadurch, dass es bei Bankverbindungen je Mitarbeiter mehrere Datenobjekte geben kann, im Gegensatz zu den persönlichen Daten, bei denen jedem Mitarbeiter ein Datensatz zugeordnet ist. Dies hat folgende Konsequenzen: Bei einem Antrag muss zuerst unterschieden werden, ob eine weitere Bankverbindung hinzugefügt werden soll, oder ob eine bestehende Bankverbindung geändert werden soll. Diese Entscheidung wird im Zustand **Typ der Änderung** abgefragt. Im Falle des Hinzufügens einer Bankverbindung wird direkt in den Zustand **Bankverbindung** gewechselt, welcher die Eingabe neuer Werte zulässt. Im Falle einer Änderung gelangt der Prozess in den Zustand **vorhandene Werte einlesen**. In diesem Zustand muss zuerst der konkrete Datensatz einer Bankverbindung, die editiert werden soll, gewählt werden. Dies geschieht durch das Schreiben eines Referenzattributes mit Referenz auf das entsprechende Objekt. Im Folgenden werden, wie bei persönliche Daten ändern beschrieben, die vorhandenen Werte ausgelesen und der Antrag weiterverarbeitet. Ein kleiner Unterschied bei der Weiterverarbeitung besteht noch im Zustand **Update durchführen**. Hier muss die Backboxaktivität zusätzlich unterscheiden, ob ein Datensatz zu editieren oder neu anzulegen ist, wenn die Werte des Antrages in den Objekttyp **Bankdaten** übernommen werden.

Mikroprozess Passbild ändern

Der Mikroprozess, welcher in Abbildung 4.9 gezeigt wird und zum Objekttyp **Passbild ändern** gehört, besitzt zu großen Teilen dieselbe Logik wie auch der Mikroprozess des Objekttyps **Bankverbindung ändern**. Jedoch muss zusätzlich noch mit der Möglichkeit umgegangen werden, dass ein Bild gelöscht oder gesperrt werden soll. Daher ist der Wertebereich des wertespezifischen Mikroschrittes im Zustand **Typ der Änderung** um die Werte löschen und sperren erweitert. Auch die Blackboxaktivität, welche dann das Update in der entsprechenden Instanz des Objekttyps durchführt, muss somit um diese Unterscheidung und Ausführung ergänzt werden.

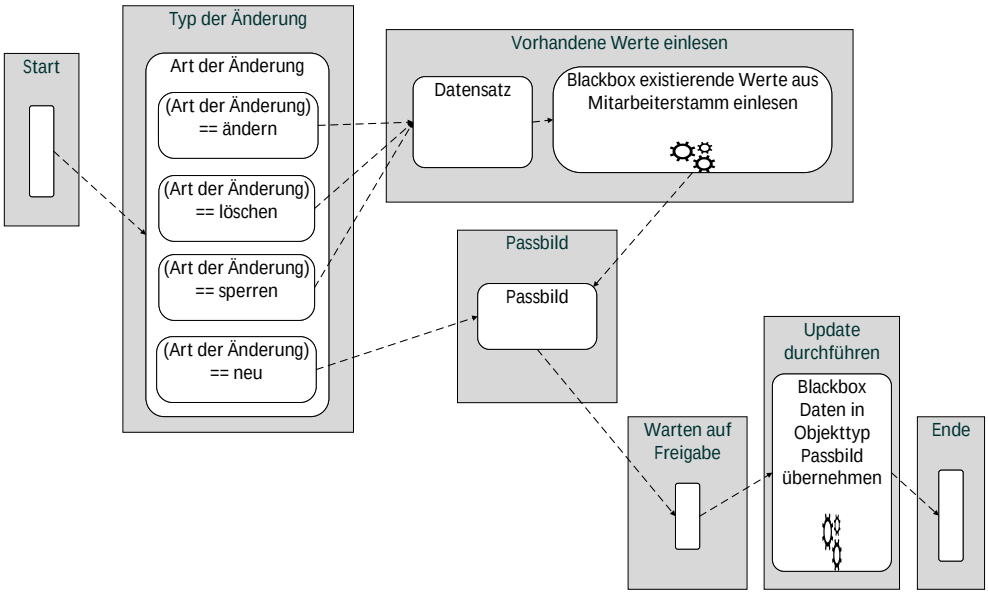

Abbildung 4.9: Mikroprozess zum Objekttyp Passbild ändern

Mikroprozess Urlaubskonto

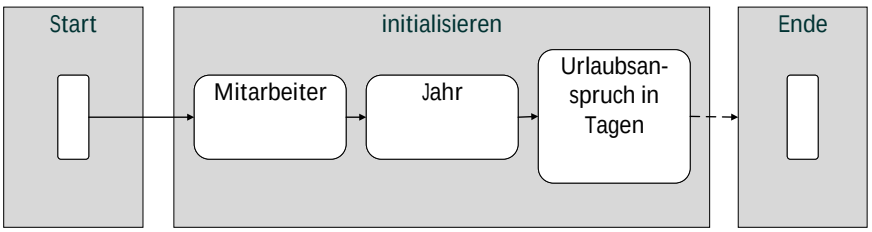

Abbildung 4.10: Mikroprozess zum Objekttyp Urlaubskonto

Die bisher aufgezeigten Mikroprozesse wurden mit Ausnahme des Mitarbeiter Mikroprozesses alle für die Funktionen des ESS benötigt. Im Folgenden werden die Mikroprozesse vorgestellt, welche die Funktionen der Fehlzeitenverwaltung implementieren. Ein erster Prozess gehört zu dem Objekttyp **Urlaubskonto**. Dieser Prozess ist in Abbildung

4.10 dargestellt. Er ist sehr simpel aufgebaut. Der Mikroschritt welcher das Attribut **Mitarbeiter** schreibt, sorgt dafür, dass zwischen dem **Urlaubskonto** und dem **Mitarbeiter** eine Relation entsteht. Durch das verpflichtende Schreiben des Attributes **Jahr** und **Urlaubsanspruch in Tagen** wird dafür gesorgt, dass der Urlaubsanspruch einer eindeutigen Mitarbeiter-Arbeitsjahr-Kombination zugeordnet werden kann und dass ein Wert als Urlaubsanspruch für diese Kombination vorhanden ist.

Mikroprozess Fehlzeitenantrag

Der Mikroprozess in den Abbildungen 4.11 und 4.12 welcher zum Objekttyp **Fehlzeitenantrag** gehört, ist der komplexeste in der gesamten Umsetzung. Er koordiniert den Ablauf des Genehmigungsverfahrens eines Fehlzeitenantrages. Der Mikroprozess besitzt folgende Eigenschaften: Im Zustand **Antrag stellen** werden die Eckdaten eines Antrages wie die Dauer und der Antragsteller erfasst, weiter wird das Antragsdatum geschrieben, was jedoch nicht durch eine Benutzereingabe, sondern automatisch erfolgen soll. Hierfür wurde ein Berechnungsschritt modelliert. Sind die Eckdaten eines Antrages eingegeben, müssen die eingegebenen Werte auf Korrektheit geprüft werden. Dies soll automatisch durch das System erfolgen. Deshalb wird im Zustand **Zeitraum prüfen** eine Blackboxaktivität modelliert, welche diese Prüfung vornimmt und je nach Prüfergebnis Werte zurückliefert, die von einem Entscheidungsschritt ausgewertet werden. Geht der Zeitraum des Antrages über den Jahreswechsel hinaus, geht das System in einen Fehlerzustand über mit der Bezeichnung **Fehler Zeitraum über Jahreswechsel**. Von diesem Zustand aus kann der Antragsteller zurückspringen um seine Angaben zu korrigieren. Reicht der vorhandene Urlaubsanspruch des Mitarbeiters nicht aus, wird in den Zustand **Fehler Urlaubsanspruch nicht ausreichend** übergegangen. Hier kann der Mitarbeiter ebenfalls zurückspringen um seine Werte zu korrigieren. Ist jedoch alles in Ordnung schreitet der Prozess im Zustand **Vertreter gewünscht** fort.

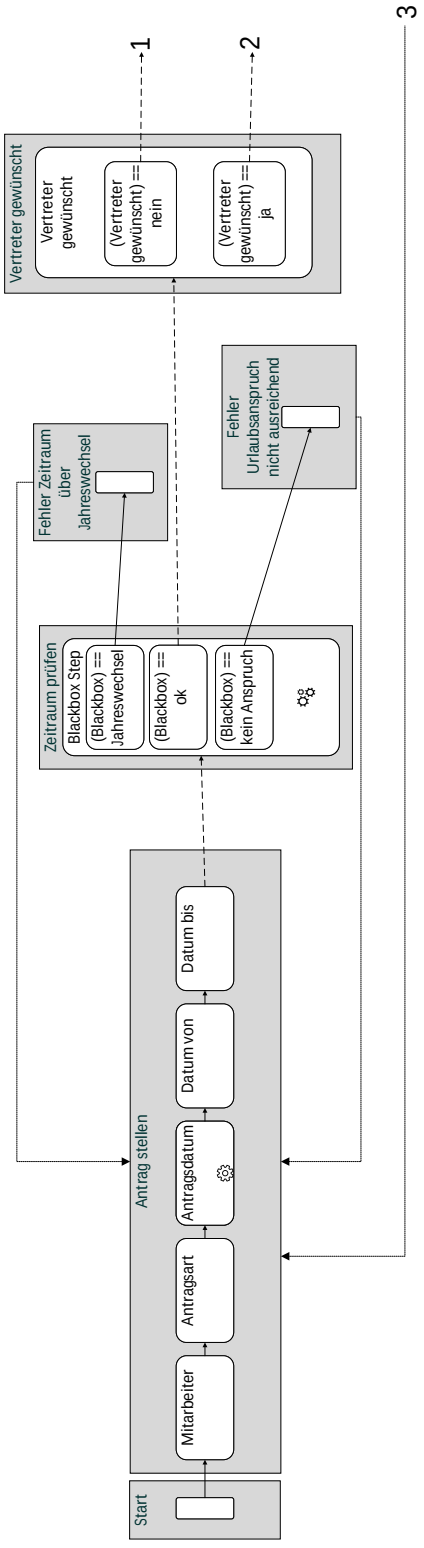

Abbildung 4.11: Mikroprozess zum Objekttyp Fehlzeitenantrag, Teil 1

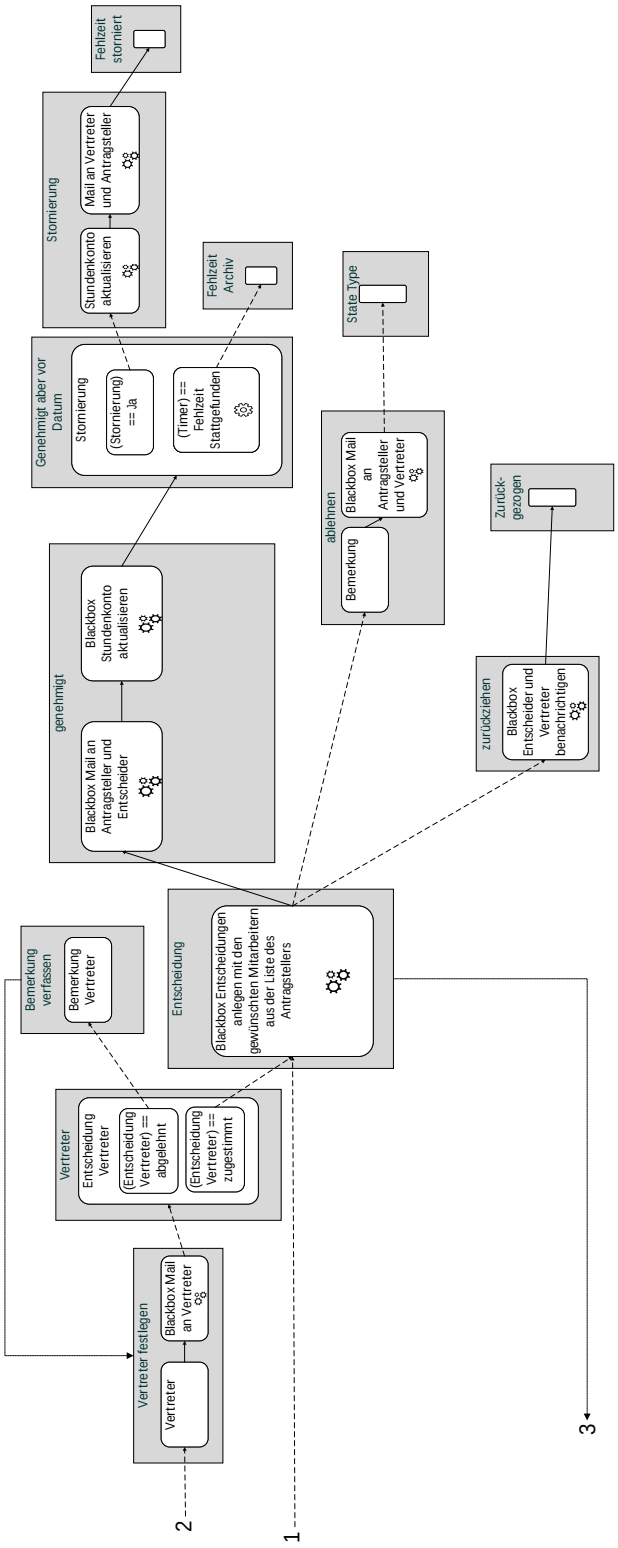

Abbildung 4.12: Mikroprozess zum Objekttyp Fehlzeitenantrag, Teil 2

Wünscht der Mitarbeiter einen Vertreter, kann er einen aus der Liste aller Mitarbeiter wählen, welcher dann vom System über eine Blackboxaktivität per Mail benachrichtigt wird. Dieser Vertreter muss nun im System seine Zustimmung abgeben. Stimmt der Vertreter nicht zu, muss er eine Bemerkung verfassen und kann dann im Prozess zurückspringen, so dass der Antragsteller einen neuen Vertreter auswählen kann. Stimmt er zu oder wurde kein Vertreter gewünscht, geht der Mikroprozess in den Zustand **Entscheidung**. Die hier gewünschte Funktionalität konnte wiederum mit Bordmitteln nicht abgebildet werden. Daher wurde wieder Mithilfe einer Blackboxaktivität ein Platzhalter geschaffen. In der Blackboxaktivität soll die Liste an Entscheidern geladen werden, welche beim Antragsteller im Mitarbeiterobjekttyp hinterlegt ist. Aus dieser Liste soll ein Mitarbeiter der Verwaltung bestimmen, wer dem Antrag zustimmen muss. Für jeden dieser Entscheider soll ein Entscheidungsobjekt angelegt werden, welches über ein Relationenattribut mit dem Mitarbeiter in Beziehung gebracht wird. Weiter soll im Objekttyp **Fehlzeitenantrag** eine Relationenliste erstellt werden, welche eine Relation zu jedem Entscheidungsobjekt abspeichert. Der Mikroprozess wartet bis die Entscheidungen gefallen sind. Diese Koordination übernimmt der Makroprozess, welcher später erläutert wird. Solange keine Entscheidung gefallen ist, kann der Antragsteller jederzeit seinen Antrag zurückziehen. Der Mikroprozess geht dann in den Zustand **zurückziehen** über, in dem per Blackboxaktivität alle Entscheider und der Vertreter per Mail benachrichtigt werden. Sind diese Benachrichtigungen ausgeführt, verbleibt das System zur Dokumentation des Vorganges im Zustand **zurückgezogen**. Lehnt mindestens ein Entscheider den Antrag ab, kann der Prozess in den Zustand **ablehnen** übergehen. Hier werden ebenfalls alle Entscheider und der Vertreter per Mail über eine Blackboxaktivität benachrichtigt und das System verbleibt im Zustand **abgelehnt** zur Dokumentation des Vorganges. Stimmen alle Entscheider zu, geht das System in den Zustand **genehmigt** über in dem per Blackboxaktivität das entsprechende Objekt **Stundenkonto** mit dem korrigierten Urlaubsanspruch des Mitarbeiters aktualisiert und alle Entscheider sowie der Antragsteller per Mail benachrichtigt werden. In Folge dazu wechselt der Mikroprozess in den Zustand **genehmigt aber vor Datum**. In diesem Zustand kann der Antrag storniert werden. Wird eine Stornierung vorgenommen, wird per Blackboxaktivität das Urlaubskonto korrigiert und der Antragsteller sowie Vertreter über eine Blackboxaktivität per Mail benachrichtigt.

Der Prozess verbleibt anschließend im Zustand storniert zur Dokumentation. Wird keine Stornierung vorgenommen und die Fehlzeit hat begonnen, gibt ein Berechnungsschritt, welcher als Platzhalter für ein Zeitereignis modelliert wurde, den Übergang zum Zustand **Fehlzeit Archiv** frei in dem das System dann auch verbleibt.

Mikroprozess Entscheidung

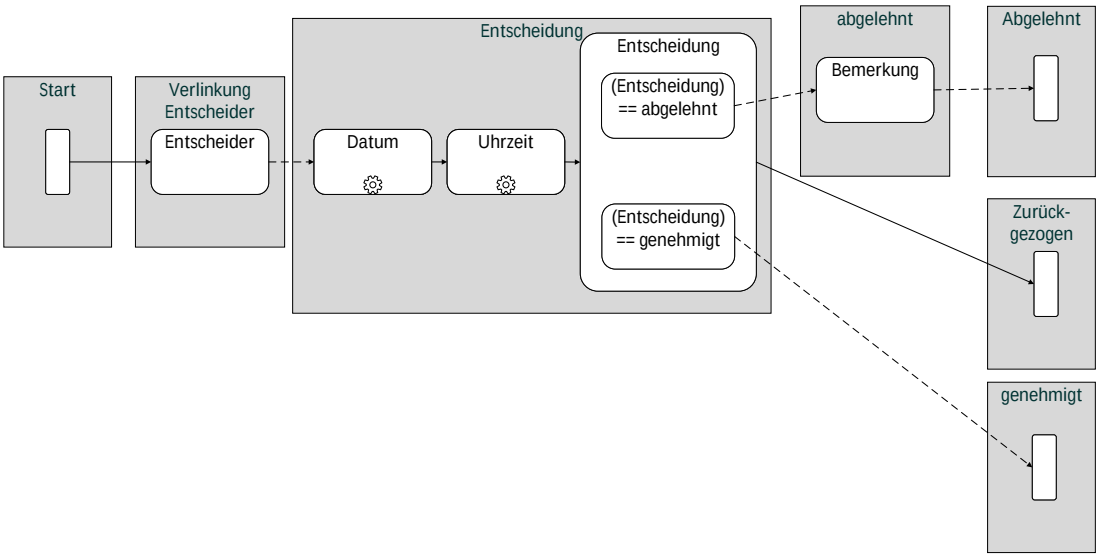

Abbildung 4.13: Mikroprozess zum Objekttyp Entscheidung

Der in Abbildung 4.13 dargestellte Mikroprozess ist dem Objekttyp Entscheidung zugeordnet, welches eine Entscheidung eines Entscheiders zu einem Fehlzeiteneintrag repräsentiert. Im Zustand **Verlinkung Entscheider** wird mittels schreiben eines Relationattributes eine Relation zu einem Mitarbeiter hergestellt, welcher als Entscheider tätig wird. Im Folgezustand wird mittels des Platzhalters Berechnungsschritt das aktuelle Datum und die aktuelle Uhrzeit vom System generiert und abgespeichert. Nun kann der Entscheider seine Entscheidung vornehmen. Stimmt er zu, geht der Prozess in den Zustand **genehmigt** über. Lehnt er ab, muss eine Bemerkung verfasst werden und der Prozess geht in den Zustand **abgelehnt** über. Wird der Antrag zurückgezogen, kann

aus den Zuständen **Entscheidung genehmigt** und **abgelehnt** heraus, in den Zustand **zurückgezogen** übergegangen werden. Da dies aber von anderen Objekttypen abhängig ist, wird diese Berechtigung in den Zustand zurückgezogen überzugehen, widerum von einem Makroprozess freigegeben, welcher später beschrieben wird.

4.1.3 Makroprozesse

In diesem Abschnitt werden die modellierten Makroprozesse zum Bewerberprozess erläutert. Der Gesamtprozess ESS und Fehlzeitenverwaltung besitzt zwei Makroprozesse, Die Koordinierenden Objekttypen sind der **Fehlzeitenantrag** und der **Antrag auf Datenänderung**. Der an das Objekt **Fehlzeitenantrag** angehängte Makroprozess koordiniert die Teile des FZ-Modules. Der an das Objekt **Antrag auf Datenänderung** angehängte die Teile des ESS-Modules.

Makroprozess zur Koordination der ESS-Objekte

Der an das Objekt **Antrag auf Datenänderung** angehängte Makroprozess wird in Abbildung 4.14 dargestellt. Er koordiniert die Abhängigkeiten zwischen dem Objekttyp **Antrag auf Datenänderung** und den einzelnen Änderungsanträgen.

Wird ein Antrag auf Datenänderung angelegt, befindet sich dessen Mikroprozess im Startzustand. Als Voraussetzung, dass ein abhängiges Objekt **persönliche Daten ändern** in den Zustand **vorhandene Werte einlesen** und die Objekte **Bankverbindung ändern**, **Passbild ändern** in den Zustand **Typ der Änderung** übergehen dürfen ist, dass das Objekt **Antrag auf Datenänderung** sich im Zustand **spezifische Werte persönliche Daten** bzw. **spezifische Werte Bankverbindung** bzw. **spezifische Werte Passbild** befindet. Dies stellt jeweils eine Top Down Beziehung dar, in der nur der Zustand **spezifische Werte persönliche Daten** bzw. **spezifische Werte Bankverbindung** bzw. **spezifische Werte Passbild** des Objektes **Antrag auf Datenänderung** einen positiven Wert für die Transition ergibt.

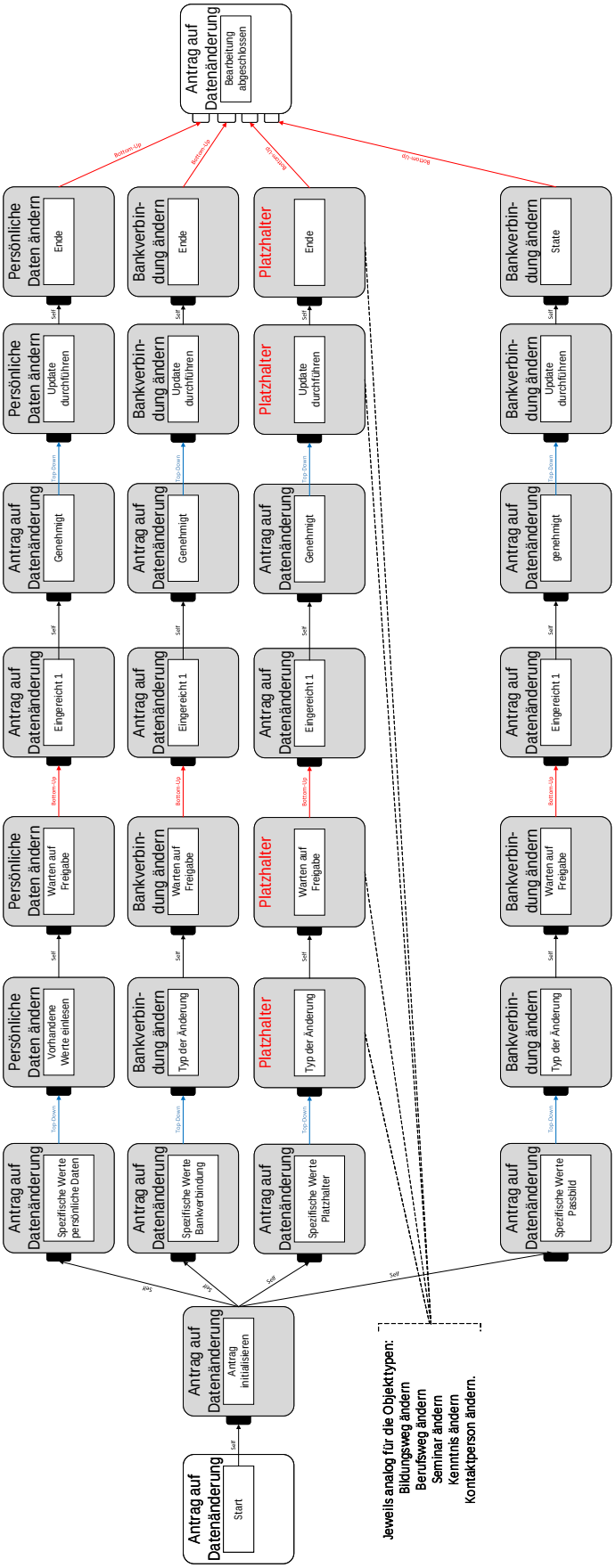

Abbildung 4.14: Makroprozess zur Koordination der ESS-Objekte

Der Antrag auf Datenänderung verlangt um in den Zustand **eingereicht 1** überzugehen, dass das Objekt **persönliche Daten ändern**, bzw. **Bankverbindung ändern**, bzw. **Passbild ändern** sich im Zustand **Warten auf Freigabe** befindet. Dies stellt eine Bottom Up Beziehung dar. Die Bedingung dieser Bottum Up Beziehung lautet: (SourceIn)==1, da genau ein verlinktes Änderungsobjekt existiert, welches dann im genannten Zustand sein muss. Um in den Zustand **Update durchführen** der Objekte **persönliche Daten ändern**, bzw. **Bankverbindung ändern**, bzw. **Passbild ändern** überzugehen, muss sich das Objekt **Antrag auf Datenänderung** im Zustand **genehmigt** befinden. Dies stellt wiederum eine Top Down Beziehung dar, bei der nur der Zustand **genehmigt** des Objektes **Antrag auf Datenänderung** einen positiven Wert für die Transition ergibt. Um einen Antrag auf Datenänderung abzuschließen, müssen die Änderungen erfolgreich durchgeführt sein. Daher wird für den Zustand **Bearbeitung abgeschlossen**, des Objektes **Antrag auf Datenänderung** vorausgesetzt, dass sich die Objekte **persönliche Daten ändern**, **Bankverbindung ändern** bzw. **Passbild ändern** im Zustand **Ende** befinden. Da je Antrag nur ein Objekt verlinkt ist, lautet die Bedingung der Bottom Up (SourceIn)==1. Die restlichen Transitionen im Prozess sind alle vom Typ Self und somit ohne weitere Bedingung.

Makroprozess zur Koordination der FZ-Objekte

An das Objekt **Fehlzeitenantrag** ist ebenfalls ein Makroprozess angehängt, welcher in Abbildung 4.15 dargestellt wird. Dieser koordiniert die Abhängigkeiten zwischen den Objekten **Fehlzeitenantrag**, **Entscheidung** und **Urlaubskonto**.

Der Prozess startet im Startzustand des Fehlzeitenantrages um der Semantik der Makroprozesse zu entsprechen, welche immer mit dem Startstep des Objektes starten müssen an den Sie angehängt wurden. Als Voraussetzung dafür, dass eine Entscheidung in den Zustand **zurückgezogen** übergehen darf ist, dass der abhängige Fehlzeitenantrag mindestens im Zustand **zurückziehen** ist. Dies stellt eine Top Down Beziehung dar. In der Bedingung der Top Down Beziehung werden die Zustände **zurückziehen** und **Zurückgezogen** erlaubt. Damit ein Fehlzeitenantrag in den Zustand zurückgezogen übergehen darf, müssen alle verlinkten Entscheidungen im Zustand **zurückgezogen** sein Dies wird

mittels einer Bottom Up Beziehung mit der Bedingung (SourceIn)==(SourceAll) erreicht. Da ein Urlaubskonto aber jederzeit angelegt werden kann, hat die Transverse Beziehung vor dem Urlaubskonto den Ausdruck (SourceIn)>=0 was immer erfüllt ist. Damit eine Entscheidung im Fehlzeitenantrag stattfinden kann, muss für den antragstellenden Mitarbeiter mindestens für ein konkretes Jahr ein Urlaubskonto angelegt werden. Dies wird mit einer transversen Transition mit der Bedingung (SourceIn)>0 realisiert. Um über den Ausgang eines Antrages zu entscheiden, sind Entscheidungen notwendig. Eine Ablehnung benötigt mindestens eine negative Entscheidung. Dies realisiert eine BottomUp mit der Bedingung (SourceIn)>0. Eine Genehmigung fordert, dass alle verlinkten Entscheidungen positiv sind. Dies realisiert eine BottumUp mit der Bedingung (SourceIn)==(SourceAll). Da ein Makroprozess in den Endzuständen des angehängten Objektes enden muss, existieren noch Self Beziehungen ohne Bedingung.

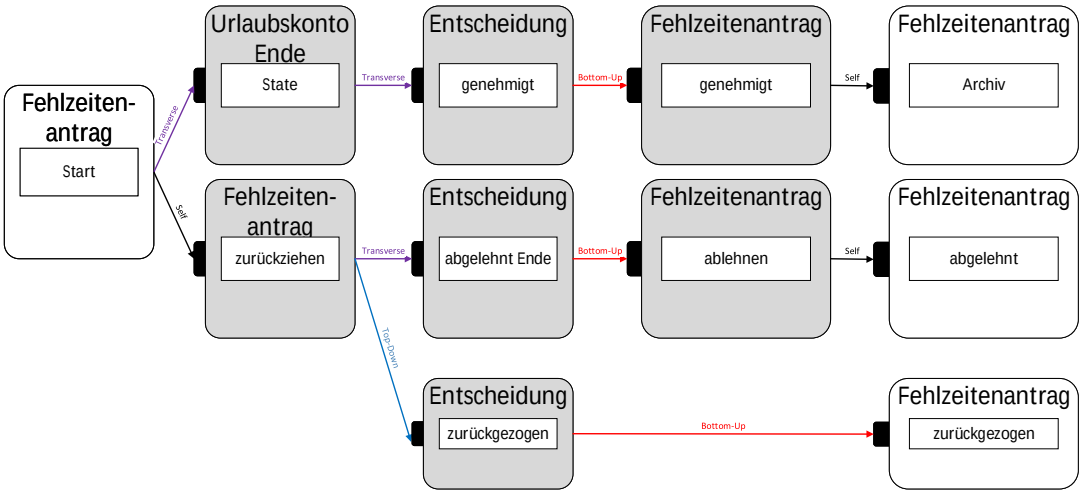

Abbildung 4.15: Makroprozess zur Koordination der FZ-Objekte

4.1.4 Benutzerintegration

In diesem Abschnitt wird die Benutzerintegration zum Modell erläutert. Das Modell beinhaltet wie bereits in Abbildung 4.1 gezeigt 3 Kontextrollen und zwei Relationenrollen.

Jeder Mitarbeiter bekommt im Attribut Funktion einen der Werte MA, MA-ESS oder MA-ESS-Verwaltung zugewiesen. Anhand dieses Attributes werden die Kontextrollen MA, MA-ESS oder auch MA Verwaltung vergeben.

Eine genaue Auflistung der Rechtevergabe je Attribut ist im Anhang 2 zu finden. Im Folgenden wird diese nur grob erläutert. Die Rolle MA erhält nur Rechte im Bereich eines Fehlzeitenantrages, welche benötigt werden um einen Fehlzeitenantrag stellen zu können. Die Rolle MA ESS erhält alle Rechte, die auch die Rolle MA erhalten hat und zusätzlich die Rechte, welche benötigt werden um einen Antrag auf Datenänderung zu stellen. Die Rolle MA Verwaltung erhält auf alles mit der Ausnahme einer Entscheidung zum Fehlzeitenantrag Lese und Schreibrechte, da diese Rechte über eine Relationenrolle der Bezeichnung Entscheider vergeben werden. Diese Relationenrolle erhält ein Mitarbeiter, wenn er zu einer Entscheidung verknüpft wird. Eine weitere Relationenrolle der Bezeichnung Antragssteller besitzt die Rechte welche zum Editieren von Daten im Antrag auf Datenänderung benötigt werden. Der Mitarbeiter erhält diese Rolle, sobald ein Antrag auf Datenänderung und der Mitarbeiter verlinkt wird.

5

Evaluation

In diesem Kapitel werden die Ergebnisse der Modellierung der Anforderungen aus Kapitel 3 evaluiert. Hierzu wird zuerst ein Bewertungsschema erarbeitet, um zu untersuchen, wie weit die Anforderungen aus Kapitel 3 zum ESS bzw. der Fehlzeitenverwaltung mit Philharmonic Flows umsetzbar waren. Hierbei wird ein Score errechnet. Weiter werden die Stärken und Schwächen von PHILharmonicFlows aufgelistet.

5.1 Evaluation

Um die Evaluation durchzuführen, ist ein Evaluationsschema entwickelt worden, welches eine objektive, nachvollziehbare und replizierbare Bewertung der Modellierungskonzepte von PHILharmonicFlows auf Basis der Anforderungen zulässt.

5.1.1 Klassifikation von Evaluationsmethoden

Eine Evaluation eines Softwareprojektes lässt sich mit verschiedenen Methoden durchführen. Eine erste Möglichkeit der Evaluation sind subjektive Evaluationsmethoden, welche unmittelbar an die Beurteilung durch den Benutzer anknüpfen. Eine subjektive Beurteilung kann z.B. mithilfe von Interviews oder mit Fragebögen geschehen. Die Beantwortung der Fragen erfolgt auf der Basis von Erfahrungen im Umgang mit dem System. Dies hat den Nachteil, dass die persönliche Meinung des Nutzers einen hohen Stellenwert in der Bewertung einnimmt und dass für belastbare Ergebnisse eine große Personenanzahl befragt werden muss. Somit werden bei subjektiven Evaluationsmethoden weiche Daten gewonnen, wie z.B. ob die Benutzung des Systems bequem,

angenehm, klar, einsichtig ist [11]. Da aber objektive, nachvollziebare und replizierbare Fakten zum Entwicklungsstand des Leistungsumfanges gewonnen werden sollen und da es nicht praktikabel ist, das System von einer breiten Menge an Personen bewerten zu lassen, sind rein subjektive Methoden nicht geeignet um das gewünschte Ergebnis zu erzielen. Eine weitere Möglichkeit sind objektive Methoden, welche sich auf objektiv messbare Größen konzentrieren. Hierbei wird versucht, subjektive Einflüsse weitgehend auszuschalten. Ohne technische Unterstützung erfolgt die Beobachtung mit Hilfe von Beobachtungsprotokollen. Eine wichtige Rolle bei der Evaluierung von Benutzungsschnittstellen spielt auch die technisch unterstützte Beobachtung, wobei ein Computer die Beobachtung übernimmt und alle Interaktionsschritte und Handlungen des Benutzers an den Eingabegeräten aufzeichnet. Durch die technisch unterstützte Beobachtung können Beobachtungs- und Beurteilungsfehler minimiert werden. Für die Systemevaluation bedeutet die Verwendung harter Methoden die Erhebung quantitativer, statistisch abgesicherter Daten, wie beispielsweise Ausführungs- und Lernzeiten sowie Fehlerraten [11]. Nachdem aber die Durchführung objektiver Methoden eine sehr aufwändige Testumgebung vorraussetzt und auf Laboruntersuchungen beschränkt ist, ist diese Variante ebenfalls nicht geeignet um PHILharmonicFlows zu bewerten, da dies den Anforderungen von Persis widerspricht.

Weiter gibt es die Techniken der leitfadenorientierten Evaluationsmethoden mit Expertenurteil. Hiervon gibt es verschiedene Varianten. Eine mögliche Variante prüft das System mit einen Experten, welcher sich an den Anforderungen zu einem umzusetzenden System orientiert. Dieses Verfahren kombiniert somit subjektive und objektive Methoden, da der Experte seine subjektive Meinung abgibt sich aber an objektiven Kriterien (den Anforderungen) orientiert. Dieses Verfahren wird nach Oppermann und Reiterer (1994) ein methodengeleitetes Expertenurteil genannt [11]. Nachteilig ist die fehlende Berücksichtigung realer Benutzer der Anwendung, da diese unter Umständen auf andere Nutzungsprobleme stoßen als die Experten, denn Experten und Benutzer gehören verschiedenen Nutzungskontexten an. Dieser Nachteil kann sich bereits bei der Auswahl der Bewertungs-Items eines Leitfadens bemerkbar machen, weil nicht validiert wurde, ob und inwieweit die Items für den Nutzungskontext des zu evaluierenden Produkts überhaupt relevant sind. Dieser Einfluss wird jedoch wieder relativiert, wenn

als Leitfaden reale Prozesse sowie deren Anforderungen, welche durch reale Benutzer aufgestellt wurden, benutzt werden.

Um PHILharmonic Flows zu evaluieren bietet sich dieses Verfahren an, da kein großer Personenkreis befragt werden muss und bereits Anforderungen an ein reales System existieren. Es wird somit eine leitfadenorientierte Evaluationsmethode mit Expertenurteil verwendet. Hierbei wird das System PHILharmonicFlows durch einen Experten geprüft, der sich an den Anforderungen zu den Modulneuentwicklungen der Persis GmbH orientiert. In einem methodengeleiteten Expertenurteil sind in einem Prüfleitfaden die anzuwendenden Methoden und Prüfkriterien festgehalten. Als Prüfkriterium wird die Umsetzbarkeit der Anforderungen aus Kapitel 3 definiert. Die Methodik der Überprüfung wird wie im folgenden beschrieben definiert: Für die Auswertung des Prüfkriteriums wird ein Schema verwendet, welches die Auswertung in drei Bereiche unterteilt, welche jeweils zu gleichen Teilen für ein Gesamtergebnis gewichtet werden. Die einzelnen Teile sind wie von der Persis GmbH vorgegeben: Umsetzbarkeit der Anforderungen zur Neuentwicklung des Employee Self Services von Persis. Umsetzbarkeit der Anforderungen zur Neuentwicklung der Fehlzeitenverwaltung von Persis, sowie allgemeinen Kriterien in denen zusätzliche Anforderungen ausgewertet werden, welche im Laufe der Modellierung als wichtig in Erscheinung getreten sind. Diese aber nicht auf ein spezielles Projekt zutreffen. Als Ergebnis dieser Auswertung soll ein Score entstehen, welcher den Entwicklungsstand der Modellierungsumgebung in einem Zahlenwert bewertet. Dazu werden für jedes Kriterium der Anforderungen individuelle Scores ermittelt und gewichtet zu einem Score für jeden Teilberreich verrechnet. Aus den drei Teilbereichen wird ein Gesamtscore per Mittelwertbildung gebildet, wobei jeder drei Bereiche gleich stark gewichtet wird.

Um die einzelnen Evaluationsbereiche zu einem Gesamtergebnis zu kombinieren wird der Mittelwert verwendet. Da jeder Bereich die gleiche Bedeutung zur Beurteilung des Gesamtergebnisses hat. Die Entscheidung für den Mittelwert wurde getroffen, weil dieser einfache und aussagekräftige Ergebnisse liefert. Alternativ wäre auch der Median vorstellbar gewesen, welcher robuster gegen statistische Ausreißer ist. Jedoch sollen statistische Ausreißer bewusst stark gewichtet werden, da ein nicht umsetzbares Kriterium bereits das komplette Konzept in Frage stellen kann.

Der Gesamtscore ist folgendermaßen zu interpretieren: Ein Score von 5 Würde bedeuten dass alles genau den Anforderungen entsprechend umsetzbar ist. Ein Score von 4 dass es umsetzbar war. Ein Score von 3 sagt dass Erweiterungen benötigt werden. Ein Score von 2 Dass es nicht vollständig umsetzbar war. Ein Score von 1 Dass der Konzeptionelle Gedanke nicht für die Problemstellung geeignet ist. Ein Score von 0 würde bedeuten dass absolut gar nichts umsetzbar war.

Weiter haben wir uns entschieden, die einzelnen Bewertungskriterien in den einzelnen Teilbereichen zu gewichten. Dies hat folgende Begründungen. Anforderungen welche gestellt sind, für deren Umsetzung das Konzept von PHILharmonicFlows aber niemals gedacht war, werden mit der Gewichtung 0 bewertet, so dass diese dann aus dem Gesamtergebnis herausfallen. Ein Beispiel hierfür wäre wie eine Benutzeroberfläche im Endsystem dargestellt wird, da PHILharmonicFlows lediglich Darstellungsinformationen zur Verfügung stellt, jedoch nie für die eigendliche Darstellung entwickelt wurde. Gewichtung 1 erhalten alle Punkte, welche durch das System umgesetzt werden sollten, aber keine besonders hervorzuhebende Bedeutung haben. Für besonders wichtige Anforderungen, welche für das System grundlegende Bedeutung haben, wird die Gewichtung 2 vergeben um diesen Anforderungen im Gesamtergebnis einen höheren Stellenwert zu geben. Diese Eingruppierung erfolgt subjektiv durch einen Experten der Persis GmbH. Für die Bewertung der einzelnen Bewertungskriterien wird ein Punktesystem mit Punkten 0 bis 5 verwendet, da dieses genügend Freiraum für eine Abstufung lässt, welche sich an bestimmte Kriterien zur Einordnung binden lässt, ohne so detailliert zu werden, dass eine Einordnung nur noch willkürlich erfolgen könnte, wie dies beispielsweise bei der Vergabe von Prozentwerten der Fall wäre.

Die einzelnen Bewertungen haben folgende Bedeutung:

- **perfekt umsetzbar** (5 Punkte): Die Anforderung lässt sich komplett und direkt umsetzen.
- **umsetzbar** (4 Punkte): Die Anforderung konnte umgesetzt werden, jedoch gibt es im Konzept noch Optimierungsmöglichkeiten. Ein Beispiel hierfür wäre die Abbildbarkeit von Mailadressen in einem Datentyp Text. Welcher zwar das abbilden

einer Mailadresse erlaubt, jedoch spezifische Eigenschaften dieser nicht separat berücksichtigt.

- **Workarounds benötigt** (3 Punkte): Die Anforderung konnte momentan nicht komplett umgesetzt werden, jedoch wäre dies mit entsprechenden Erweiterungen des Konzeptes möglich. Diese Bewertung wurde auch dann verwendet, Blackboxaktivitäten verwendet wurden, da hier für die Funktionalität Implementierungsaufwand nötig ist.
- **mit geänderten Anforderungen** (2 Punkte): Eine Umsetzung war nicht den Anforderungen entsprechend möglich. Es können jedoch die Anforderungen so abgeändert werden, dass sich eine semantisch äquivalente Alternative ergab. Welche weniger Nachteile bietet wie eine Umfangreiche Erweiterung des Konzeptes mit sich bringen würde.
- **umfangreiche Erweiterungen** (1 Punkt): Eine 100 Prozent den Anforderungen entsprechende Umsetzung sowie eine Anpassung der Anforderungen ist nicht möglich. Jedoch könnte mit umfangreichen Erweiterungen, welche grundlegend andere Funktionalitäten wie bisher im Konzept vorgesehen möglich machen, eine Umsetzbarkeit geschaffen werden.
- **nicht Umsetzbar** (0 Punkte): Eine Umsetzung ist nicht möglich auch nicht mit Erweiterungen von PHILharmonicFlows, da diese Erweiterungen das Kernkonzept von PHILharmonicFlows verletzen würden.

5.2 Teil 1 Umsetzbarkeit der Anforderungen zum ESS

Die Auswertung erfolgt strikt anhand der im Anforderungsdokument hinterlegten Kriterien, welche im Kapitel 3.2 detailliert beschrieben wurden.

ESS aus Datensicht

Anf. 1.1.1 Antrag auf Datenänderung (3.2.1): Die einzelnen Bereiche (**Persönliche Daten**, **Bankverbindung**, **Bildungsweg**, **Berufsweg**, **Seminare**, **Kenntnisse**, **Kon-**

taktpersonen, **Passbild**) eines Antrags auf Datenänderung wurde per Drop-Down Feld abgebildet. So kann der Antragsteller auswählen, welchen Bereich sein Antrag betrifft. (Siehe auch Abbildung 4.6). Die Zustände eines Antrages werden per Mikroprozesszustände abgebildet. Somit kann die Anforderung komplett umgesetzt werden. Diese Anforderung war **perfekt umsetzbar** (5 Punkte) und wird mit dem **Gewicht 1** bewertet.

Anf. 1.1.2 Mitarbeiterdaten (3.2.1): Im Objekt Persönliche Daten existiert ein Zustand **Mitarbeiterdaten**. Dieser Zustand sorgt dafür, dass alle verpflichtenden Werte als Attribute abgebildet werden. Die Anforderung konnte somit **perfekt umgesetzt** (5 Punkte) werden und wird mit dem **Gewicht 1** bewertet.

Anf. 1.1.3 Privatanschrift (3.2.1): Um dafür Sorge zu tragen, dass im Objekt **Persönliche Daten** alle verpflichtenden Attribute abgebildet werden, existiert ein Mikroprozesszustand **Privatanschrift** mit entsprechenden Mikroschritten. Die zweite Privatanschrift wird mit optionalen Attributen abgebildet. Die Anforderung war somit **perfekt umsetzbar** (5 Punkte) und wird mit dem **Gewicht 1** bewertet.

Anf. 1.1.4 Persönliche Kontaktdaten des Mitarbeiters (3.2.1): Im Objekt **Persönliche Daten** existiert ein Zustand **persönliche Kontaktdaten**. Dieser Mirkoprozesszustand sorgt dafür, dass alle gewünschten Werte als Attribute abgebildet werden. Die zweiten persönlichen Kontaktdaten wurden mit optionalen Attributen abgebildet. Die Anforderung konnte somit komplett umgesetzt werden, jedoch wäre für Mail Adressen ein eigener Datentyp gut, welcher nur plausible Werte als Mail Adresse zulässt. Daher ist diese Anforderung **umsetzbar**(4 Punkte) und wird ebenfalls mit dem **Gewicht 1** bewertet.

Anf. 1.1.5 Dienstliche Kontaktdaten des Mitarbeiters (3.2.1): Im Objekt **Persönliche Daten** wurde ein Zustand **dienstliche Kontaktdaten** hinzugefügt. Dieser Zustand sorgt dafür dass alle verpflichtenden Werte als Attribute abgebildet werden. Die Anforderung konnte somit komplett umgesetzt werden, jedoch würde es auch hier Sinn ergeben,

wenn es für Mail Adressen einen eigenen Datentyp geben würde, welcher nur plausible Werte als Mail Adresse zulässt. Daher ist diese Anforderung **umsetzbar** (4 Punkte) und wird ebenfalls mit dem **Gewicht 1** bewertet.

Anf. 1.1.6 Bankverbindungen (3.2.1): Es existiert ein Objekt **Bankverbindung** (4.4). In diesem Objekt konnten alle gewünschten Werte als Attribute abgebildet werden. Jedoch soll die **IBAN** und die **BIC** automatisch aus der **BLZ** und der **Kontonummer** generiert werden. Hierfür wurde ein Berechnungsschritt verwendet. Sollen mehrere Bankverbindungen verwaltet werden, so werden hierfür mehrere Objekte angelegt. Dies führt zu der Bewertung **Workarounds benötigt** (3 Punkte) und wird mit **Gewicht 1** bewertet.

Anf. 1.1.7 Berufsweg (3.2.1): Auch für den Berufsweg wurde ein eigenes Objekt **Berufsweg** geschaffen. In diesem Objekt konnten alle gewünschten Werte als Attribute abgebildet werden. Für jeden Abschnitt des Berufsweges muss ein separates Objekt angelegt werden. Die Anforderung war somit **perfekt umsetzbar** (5 Punkte) und wird mit dem **Gewicht 1** bewertet.

Anf. 1.1.8 Bildungsweg (3.2.1): Es gibt ein Objekt **Bildungsweg**. In diesem Objekt konnten alle gewünschten Werte als Attribute abgebildet werden. Für mehrere Abschnitte eines Bildungsweges müssen mehrere Objekte angelegt werden. Die Anforderung konnte komplett erfüllt werden, war somit **perfekt umsetzbar** (5 Punkte) und wird mit dem **Gewicht 1** bewertet.

Anf. 1.1.9 Kenntnisse (3.2.1): Um die Kenntnisse abbilden zu können, wurde ein Objekt **Kenntnisse** geschaffen. In diesem Objekt wurden alle gewünschten Werte als Attribute abgebildet. Für mehrere Kenntnisse müssen mehrere Objekte angelegt werden. Die Anforderung war somit **perfekt umsetzbar** (5 Punkte) und wird mit dem **Gewicht 1** bewertet.

Anf. 1.1.10 Kontaktpersonen (3.2.1): Es existiert ein Objekt **Kontaktperson**. In diesem Objekt konnten alle gewünschten Werte als Attribute abgebildet werden. Für mehrere Kontaktpersonen müssen mehrere Objekte angelegt werden.Die Anforderung konnte komplett umgesetzt werden. Die Anforderung war somit **perfekt umsetzbar** (5 Punkte) und wird mit dem **Gewicht 1** bewertet.

Anf. 1.1.11 Passbild (3.2.1): Es existiert ein Objekt **Passbild**. In diesem Objekt kann ein Passbild als File angelegt werden. Ist ein Passbild hinterlegt ist es möglich, dieses sperren oder löschen zu lassen. Wird das Bild gesperrt, geht das Objekt in den Zustand **gesperrt** über, das Bild bleibt jedoch gespeichert. Soll das Bild gelöscht werden, so soll das Bild auch wirklich überall entfernt werden. Auch in einem evtl. vorhandenen **Antrag auf Datenänderung** der dieses Bild evtl mal bereitgestellt hat. Daher wurde als Workaround das Konzept der Blackbox Aktivität verwendet um das Bild zu löschen. Wurde ein Bild entfernt geht der Zustand des Mikroprozesses in den Zustand **gelöscht** über. Dies führt zu der Bewertung **Workarounds benötigt** (3 Punkte) und wird mit **Gewicht 1** bewertet.

Anf. 1.1.12 Seminare (3.2.1): Seminare werden über das Objekt **Seminar** verwaltet. In diesem Objekt wurden alle gewünschten Werte als Attribute abgebildet. Für mehrere Seminare müssen mehrere Objekte angelegt werden. Die Anforderung war somit **perfekt umsetzbar** (5 Punkte) und wird mit dem **Gewicht 1** bewertet.

Anf. 1.1.13 Darstellung von Änderungen (3.2.1): Um Daten zu ändern, muss ein **Antrag auf Datenänderung** ausgefüllt werden. Wird dieser Antrag freigegeben, so werden die Daten im eigentlichem Datenobjekt abgeändert. Jeder freigegebene Antrag bleibt im System. Da eine Datenänderung somit nicht direkt auf dem Datenobjekt erfolgt, sondern für jede Änderung ein Antrag abgegeben werden muss, welcher dann auch im System verbleibt, kann jede Änderung nachvollzogen werden. Jedoch gibt es noch kein Konzept, wie die Daten eines genehmigten Antrages automatisch in die eigentliche Datenhaltung mit den aktuellen Werten übertragen werden können. Daher wurde zur

Übertragung der Daten in die eigentlichen Datensätze eines Mitarbeiters eine Blackboxaktivität verwendet. Diese führt ein Update eines bestehenden Datensatzes durch. bzw. legt bei einer Ergänzung um einen neuen Datensatz, z.B. wenn eine zusätzliche Bankverbindung angefügt wird, ein neues Objekt des entsprechenden Datensatzes an. Wie diese Datenanträge jedoch dargestellt werden können ist dann Sache einer Benutzeroberfläche und nicht der Datenhaltung durch PHILharmonicFlows. Da dies ein essentieller Vorgang im neuen Persismodul ist, wird dies mit dem **Gewicht 2** bewertet. Da Blackboxaktivitäten verwendet werden wird dies mit **Workarounds benötigt** (3 Punkte) bewertet.

ESS Rollendefinitionen

Anf. 1.2.1 Rolle MA (3.2.2): Es wurde eine Kontextrolle **MA** angelegt. Mitarbeiter mit der Rolle **MA** haben im ESS keine Schreib- und Leserechte. Da die Rollenvergabe im System essentielle Bedeutung hat wird das **Gewicht 2** verwendet. Bewertet wird die Anforderung mit **perfekt umsetzbar** (5 Punkte).

Anf. 1.2.2 Rolle MA-ESS (3.2.2): Weiter wurde eine Rolle **MA-ESS** definiert. Auch hier war es möglich alle Rechte so zu vergeben, dass nur die erlaubten Aufgaben ausgeführt werden dürfen. Die Rolle **MA-ESS** wird als Kontextrolle vergeben. Jedoch reicht die Rolle **MA-ESS** nicht aus, um definieren zu können, welche Mitarbeiter einen **Antrag auf Datenänderung** weiter ausfüllen dürfen. Hierfür wurde weiter eine Relationenrolle **Antragssteller** verwendet, welche ein Mitarbeiter erhält, wenn er als Antragssteller auf einen Datenänderungsantrag verknüpft wird. Diese Funktionalitäten sind essentiell für das ESS-Modul und werden mit dem **Gewicht 2** versehen, und waren **perfekt umsetzbar** (5 Punkte).

Anf. 1.2.3 Rolle MA-Verwaltung (3.2.2): Es wurde eine Rolle **MA-Verwaltung** definiert, welche ebenfalls als Kontextrolle erlangt wird. Es war nicht möglich, die Rechte zu dieser Rolle so einzuschränken, dass keine eigenen Anträge bearbeitet werden dürfen. Hierzu wäre es nötig, den aktuell angemeldeten Bearbeiter am System auslesen zu

können, um über einen Datenkontext den aktuell angemeldeter User mit dem Antragssteller vergleichen zu können. Es soll ein Schreibrecht nur vergeben werden wenn der angemeldete Benutzer ungleich dem Antragssteller ist. Eine Rolle **MA-Verwaltung-Chef** wurde dann gar nicht mehr definiert, da die Rolle **MA-Verwaltung** bereits alles darf was eigentlich nur die **Rolle MA-Verwaltung-Chef** können sollte. Es wird das **Gewicht 2** verwendet. Da es leider nicht möglich war, zwischen den Rollen MA-Verwaltung und MA-Verwaltung-Chef zu unterscheiden, was aber durch eine minimale Erweiterung machbar wäre, wird die Bewertung **umsetzbar** (4 Punkte) benötigt vergeben.

ESS aus Prozesssicht

Anf. 1.3.1 Zustände von Anträgen (3.2.3): Für jede Datenänderung muss ein Antrag auf Datenänderung gestellt werden. Dieser wird vom Objekt **Antrag auf Datenänderung** dargestellt. Die einzelnen Zustände eines Antrages werden über Mikroprozesszustände abgebildet. Die Übergänge zwischen den Zuständen werden durch den Mikroprozess und den Makroprozess (sofern der Übergang auch von einem weiteren Objekt abhängt) koordiniert. Es war möglich, alle Zustände und alle geforderten Übergänge entsprechend abzubilden. Da dies der grundlegende Prozess ist, welcher für das Modul die Grundlage darstellt, wird das **Gewicht 2** verwendet. Nachdem diese Zustandsabfolge komplett abgebildet wurde, wird die Bewertung **perfekt umsetzbar** (5 Punkte) vergeben.

Zwischenergebniss ESS

Die Summation der gewichteten Scores ergibt den Wert 98. Wird aus diesem der gewichtete Mittelwert der einzelnen Kriterien gebildet, so ergibt sich der Score 4,45 für das ESS.

5.3 Teil 2 Umsetzbarkeit der Anforderungen zur Fehlzeitenverwaltung

Die Auswertung erfolgt strikt anhand der im Anforderungsdokument hinterlegten Kriterien, welche im Kapitel 3.3 detailliert beschrieben wurden.

Fehlzeitenverwaltung aus Datensicht

Anf. 2.1.1 Fehlzeitenantrag (3.3.1): Eine Angabe der Werte: **Name des Antragstellers**, die **Antragsart** und der Antragszeitraum bestehend aus **Datum von** sowie **Datum bis** konnte als Attribut umgesetzt werden. Jedoch ist es nicht möglich, nach Eingabe des Zeitraumes der Fehlzeit automatisch ein Formular zu generieren, welches für jeden Tag des Zeitraums weitere Eingaben zulässt, da es hier nicht möglich ist, diese Formulare bereits zur Modellierungszeit zu generieren und dann nur abhängig von einer begrenzten Zahl an Eingabemöglichkeiten aufzurufen. Um Formulare abhängig von Eingaben generieren zu können, müsste das Konzept von PHILharmonicFlows erweitert werden. Um Attribute automatisch mit Werten zu befüllen wie dies z.B. notwendig ist, wenn automatisiert das Datum der Bearbeitung abgespeichert werden soll, wurden Berechnungsschritte verwendet. Nachdem das Szenario der Formulargenerierung von grundlegender Bedeutung für die gewünschte Funktion des Moduls ist, wird dies mit **Gewicht 2** gewichtet. Da ein Formular in dieser Hinsicht aber nur mit **umfangreichen Erweiterungen** (1 Punkt) umsetzbar ist, wird dies auch so bewertet.

Anf. 2.1.2 Urlaubskonto (3.3.1): Es wurde ein Objekt **Urlaubsanspruch** mit den Attributen **Jahr** und **Mitarbeiter** sowie **Urlaubsanspruch in Tagen** generiert. Somit kann der Urlaubsanspruch wie gefordert, je Jahr und Mitarbeiter abgebildet werden. Jedoch existieren momentan keine Konzepte, wie der Resturlaubsanspruch aus einem anderen Objekt heraus angepasst werden kann. Dies wäre nötig, wenn z.B. ein neuer Fehlzeitenantrag genehmigt wurde, welcher Auswirkungen auf den Resturlaubsanspruch hat. Die Modellierung behilft sich hier mit einer Blackboxaktivität im anderen Objekt, welche das Attribut des Objektes **Urlaubsanspruch** entsprechend manipuliert. Weiterhin

ist es nicht möglich, Kommazahlen auf eine bestimmte Anzahl an Nachkommastellen einzuschränken. Hierzu wäre eine Erweiterung der Attribute nötig. Das Verwalten des Urlaubsanspruches ist eine zentrale Anforderung, daher wird **Gewicht 2** verwendet. Da dies jedoch nur mit einer Blackboxaktivität möglich war, wird die Bewertung mit **Workarounds benötigt** (3 Punkte) abgegeben.

Anf. 2.1.3 Vertreter (3.3.1): Über den Mikroprozess wird bestimmt, ob ein Vertreter gewünscht ist. Ist ein Vertreter gewünscht, kann ein Vertreter aus allen Mitarbeitern ausgewählt werden. Dieser Mitarbeiter muss der Vertretung zustimmen. Es ist nicht möglich, die Schreibrechte auf die Zustimmung so einzuschränken, dass nur der Vertreter zustimmen kann. Hierfür müsste es möglich sein, abzuprüfen, ob der aktuell am System angemeldete User gleich dem Vertreter ist, was momentan nicht möglich ist. Zwar wäre es möglich, die Schreibrechte zur Zustimmung an eine Relationenrolle zu binden, welche nur vergeben wird, wenn eine Relation des Mitarbeiters zum Fehlzeitenantrag besteht. Jedoch auch dann ist diese Rolle nicht ausreichend definiert, da der Antragssteller diese Rolle ebenfalls erhält, da er als Antragssteller ebenfalls eine Relation zwischen sich und dem Fehlzeitenantragsobjekt besitzt. Lehnt der Vertreter ab, muss er zwangsweise eine Bemerkung als Begründung verfassen. Stimmt er jedoch zu, ist die Bemerkung optional. Das automatische Speichern des Entscheidungsdatums wird mittels Berechnungsschritt realisiert. Diese Anforderung wird mit dem **Gewicht 2** versehen und da nur kleinste Anpassungen bei der Prüfung des Logins nötig sind, mit **umsetzbar** (4 Punkten) bewertet.

Anf. 2.1.4 Entscheidung (3.3.1): Alle Personen, welche die Berechtigung haben, einen Fehlzeitenantrag eines Mitarbeiters zu genehmigen, sind in einer Liste im Objekt, welches den Antragssteller repräsentiert, abgespeichert. Aus diesen Personen soll nun ein Mitarbeiter mit der Rolle **MA-Verwaltung** eine Menge an Mitarbeitern auswählen, welche dem Antrag zustimmen müssen. PHILharmonicFlows ermöglicht es momentan nicht, auf Attribute eines anderen Objektes zuzugreifen. Daher wurde eine Blackboxaktivität verwendet, welche die Liste des Mitarbeiter Objektes besorgt und den ausführenden Mitarbeiter mit der Rolle **MA-Verwaltung** eine Menge an Mitarbeitern wählen lässt. Für

jeden dieser Mitarbeiter wird ein Entscheidungsobjekt, sowie eine Relation zu diesem angelegt, welche dem Mitarbeiter eine Relationenrolle Entscheider vergibt. Weiter ist es auch hier nicht möglich, das Datum der Entscheidung automatisch abzuspeichern. Es wird **Gewicht 2** vergeben und die Umsetzbarkeit mit **Workarounds benötigt** (3 Punkte) vergeben.

Anf. 2.1.5 Mitarbeiterkalender (3.3.1): Im System werden alle Daten erfasst, um eine ausreichende Datenbasis eines Mitarbeiterkalenders anbieten zu können. Jedoch ist es nicht Aufgabe von PHILharmonicFlows, diese im Kalender dazustellen. Dies ist Aufgabe einer Benutzeroberfläche. Da dies nicht Aufgabe von PHILharmonicFlows ist, ergibt sich das **Gewicht 0** und die Bewertung **nicht umsetzbar** (0 Punkte).

Anf. 2.1.6 Stornierung (3.3.1): Mögliche Stornierungen werden vom Fehlzeitantragsobjekt mit verwaltet. Somit steht die Stornierung immer im Kontext zum ursprünglichen Antrag. Jedoch ist es auch wie beim Fehlzeitantrag nicht möglich , automatisch ein Formular zu generieren, welches automatisiert Bezug auf die einzelnen Tage des eingegebenen Zeitraumes nimmt. Somit kann in der Modellierung ein Antrag nur als Ganzes storniert werden. Es wird **Gewicht 2** vergeben und die Bewertung **mit geänderten Anforderungen** (2 Punkte) vergeben.

Anf. 2.1.7 Mitarbeiter (3.3.1): Jedes Mitarbeiterobjekt besitzt eine Attributsliste mit allen Mitarbeitern, welche über seine Fehlzeiten entscheiden dürfen. Dies wird mit **Gewicht 1** bewertet. Aie Anforderung ist **perfekt umsetzbar** (5 Punkte).

Fehlzeitenverwaltung Rollendefinitionen

Anf. 2.2.1 Rolle MA, Rolle MA-ESS, Rolle MA-Verwaltung, Rolle MA-Verwaltung-Chef (3.3.2): Für alle Rollen, welche bereits beim ESS definiert wurden, konnten entsprechende Schreib- und Leserechte vergeben werden, um einen Antrag auf Fehlzeiten stellen zu können. Um das Schreibrecht zu einer Entscheidung zu erteilen, wurde

eine weitere Relationenrolle **Entscheider** vergeben, welche an die Relation von einer Entscheidung zu einem Mitarbeiter gebunden ist. Da die Rechte instanzspezifisch vergeben werden, darf auch nur die Entscheidung, welche per Relation mit dem Nutzer verbunden ist, geschrieben werden. Somit ist diese Anforderung erfüllt, was zur Bewertung **perfekt umsetzbar** (5 Punkte) mit dem **Gewicht 2** führt.

Anf. 2.2.2 Rolle Entscheider (3.3.2): Die Rolle **Entscheider** wird als Relationrolle vergeben, sobald eine entsprechende Relation eines Mitarbeiters als Entscheider zu einer Entscheidung angelegt wird. Da die Rechte, welche mit der Relationenrolle erteilt werden, instanzspezifisch sind, werden entsprechende Schreibrechte auch nur für die zu bearbeitende Entscheidung vergeben. Dies ist somit **perfekt umsetzbar** (5 Punkte) mit dem **Gewicht 2**

Fehlzeitenverwaltung aus Prozesssicht

Anf. 2.3.1 Fehlzeitenantrag (3.3.3): Im Mikroprozess **Fehlzeitenantrag** prüft eine Blackboxaktivität den eingegebenen Zeitraum und gibt eine entprechende Fehlermeldung aus, wenn der Zeitraum über das Jahresende geht. In PHILharmonicFlows ist es nicht möglich, ein Formular basierend auf Runtime Daten dynamisch zu modifizieren. Somit ist es nicht möglich, im Bezug auf den eingebenen Zeitraum spezifische Werte für jeden einzelnen Arbeitstag abzufragen. Eine Prüfung ob der Urlaubsanspruch ausreichend ist, kann auch nur mithilfe einer Blackboxaktivität durchgeführt werden, da momentan kein Zugriff auf Werte von Attributen von anderen Objekten in PHILharmonic Flows möglich ist. Dies führt zur Bewertung nur mit **umfangreichen Erweiterungen** (1 Punkt) realisierbar, da eine Blackboxaktivität nicht dazu verwendet werden sollte, um solche Funktionalitäten bereitzustellen. Es wird **Gewicht 2** verwendet da diese Funktion zwingend für die Funktionalität eines Antrages ist.

Anf. 2.3.2 Vertreterregelung (3.3.3): In der Modellierung kann der Antragssteller entscheiden, ob ein Vertreter gewünscht ist und im Falle, dass einer gewünscht ist, einen Vertreter wählen. Die Benachrichtigung des Vertreters wird per Blackboxaktivität

ausgeführt. Aufgrund dessen, dass eine Blackboxaktivität benötigt wird, ergibt sich die Bewertung **Workarounds benötigt** (3 Punkte). Da dies alles Standardfunktionen eines Systemes sind, wird **Gewicht 1** verwendet.

Anf. 2.3.3 Entscheidung (3.3.3): Das Auswählen aller benötigten Entscheider zu einem Antrag durch einen Mitarbeiter mit der Rolle **MA-Verwaltung** ist problematisch, da es keine Konzepte gibt, um auf die Attribute eines anderen Objektes zuzugreifen. Daher wurde diese Einschränkung mit einer Blackboxaktivität umgangen, welche die Werte aus der Liste des Mitarbeiters ausliest. Der ausführende Mitarbeiter in der Rolle **MA-Verwaltung** kann jetzt eine Auswahl treffen. Anschließend wird zu jedem gewählten Entscheider ein entsprechendes Entscheidungsobjekt anlegt und eine Relation herstellt. Der Makroprozess sorgt nun dafür, dass für eine Genehmigung des Antrages alle angelegten Entscheidungsobjekte im Zustand **genehmigt** sein müssen. Die entsprechenden Benachrichtigungen werden wiederum über eine Blackboxaktivität ausgeführt. Dies wird mit **Gewicht 2** gewichtet und mit der Bewertung **Workarounds benötigt** (3 Punkte) versehen.

Anf. 2.3.4 Stornierung (3.3.3): Aufgrund der Restriktion, dass keine Formulare basierend auf Daten der Runtime dynamisch erzeugt werden können, ist eine tageweise Stornierung nicht zu modellieren, sondern nur eine Stornierung des Auftrages im Gesamten. Die Benachrichtigungen erfolgen in den entsprechenden Zuständen in einer Blackboxaktivität. Nachdem das tageweise Stornieren eine essentielle Funktion des Modules ist ergibt sich **Gewicht 2** mit der Bewertung **umfangreiche Erweiterungen** (1 Punkt) benötigt.

Anf. 2.3.5 Zurückziehen eines Antrags (3.3.3): Das Zurückziehen eines Antrages ist möglich. Jedoch kann leider das Recht hierzu nicht auf den Antragssteller eingegrenzt werden, da hierfür die Bedingung definiert werden müsste, dass der Antragssteller gleich dem angemeldeten ausführenden Benutzer ist. Da dies momentan nicht möglich ist jedoch das nur das hinzufügen einer weiteren Expression bedeuten würde dass es umsetzbar ist,wird die Bewertung **umsetzbar** (4 Punkte) vergeben. Das Definieren von

Rechten passgenauen Rechten zu der Anforderung ist von essentieller Bedeutung, was zum **Gewicht 2** führt.

Anf. 2.3.6 Mitarbeiterkalender (3.3.3): Das Anzeigen eines Mitarbeiterkalenders ist nicht die Aufgabe von PHILharmonicFlows. Es werden zwar alle benötigten Daten durch das System bereit gestellt. Die Anzeige übernimmt aber die Benutzeroberfläche. Daher wird diese Anforderung mit dem **Gewicht 0** versehen und mit **nicht umsetzbar** (0 Punkte) bewertet.

Zwischenergebniss FZ

Die gewichteten Scores ergibt den Wert 68. Wird aus diesem der gewichtete Mittelwert der einzelnen Kriterien gebildet, so ergibt sich der Score 2,83 für die FZ.

5.4 Teil 3 Allgemeines

5.4.1 Evaluation Allgemeiner Kriterien

In dieser Sektion werden allgemeine Kriterien ausgewertet, welche beim Modellieren als besonders interessant in Erscheinung getreten sind. Daher sind hier auch nur Bewertungen zu finden, welche einen hohen Stellenwert einnehmen. Daher werden alle diese Bewertungen mit dem Gewicht 2 bewertet.

Koordination der Abläufe Es war weitestgehend möglich, die Abläufe so zu koordinieren, wie es von der Anwendung her Sinn ergibt. Lediglich das Instanziieren von Objekten konnte nicht eingeschränkt werden. So kann ein beliebiges Objekt zur Laufzeit jederzeit instanziiert werden. Ein Einschränken von Instanziierungen wird dann über die Benutzeroberfläche geregelt. Somit konnte die Koordination der Abläufe **perfekt umgesetzt** (5 Punkte) werden.

Einbindung bereits vorhandener Systeme wie z.B. Datenbanken Um PHILharmonicFlows sinnvoll in einem Unternehmen einsetzen zu können, benötigt es Schnittstellen zu anderen Systemen. Dies ist im Moment mit dem Konzept der Blackboxaktivitäten umsetzbar. Da eine Blackboxaktivität benötigt wird, ergibt sich die Bewertung **Workarounds benötigt** (3 Punkte).

Rechtemanagement von Usern im System Das Rechtemanagement von Philharmonic Flows ist sehr feingranular, was eine sehr genaue Vergabe von Rechten ermöglicht hat. Jedoch ist es noch nicht ausreichend, um alles abbilden zu können. Es kann z.B. nicht eingeschränkt werden, welche Objekte mittels eines Relationattributs verlinkt werden können. Dies ist zum Beispiel der Fall, wenn ein Datensatz eines Mitarbeiters ausgewählt werden soll, da dann grundsätzlich das Auswählen eines beliebigen Datensatzes möglich ist, unabhängig davon ob dieser zum Mitarbeiter gehört welcher dem aktuellen Vorgang zugeordnet ist. Somit sind hierbei im Moment immer alle vom Typ her passenden Objekte möglich und es müsste möglich sein, Expressions zu definieren welche dies einschränken.

Auch ist es nicht möglich den aktuell angemeldeten Benutzer am System abzufragen um bei der Rechtevergabe diesen in der Expression verwenden zu können um z.B. zu verbieten, dass ein Mitarbeiter seinen eigenen Antrag genehmigt. Weiter ist die Definition der Rechte noch nicht vollständig ausgereift. Hierzu würde es anwenderfreundlichere Lösungen geben, was jedoch den Funktionsumfang nicht schmälert. Daher ergibt sich die Bewertung **umsetzbar** (4 Punkte).

Kommunikation mit Personen außerhalb des Systems Um PHILharmonicFlows sinnvoll in einem realen Szenario einsetzen zu können, braucht es Schnittstellen zur Kommunikation (z.B. Mailversand, Export als XML, Anbindung an Datenbanken usw.). Dies wird durch das Konzept der Blackboxaktivitäten ermöglicht und somit mit **Workarounds benötigt** (3 Punkte) bewertet und da dies eine grundlegende Funktion ist mit **Gewicht 2** versehen.

Beziehungen von Objekten untereinander Beziehungen zueinander können mittels Relationen jederzeit hergestellt werden, jedoch gibt es keine Möglichkeit semantisch falsche Beziehungen zu unterbinden, so kann z.B. ein Bewerber beim Ausfüllen des Bewerberformulares das Formular nur mit seinem eigenen Benutzertyp, sondern auch mit beliebigen anderen Benutzertypen welche sich im System befinden, verlinken. Dies wird auch durch Abbildung 5.1 illustriert, welche ein Dropdownfeld zeigt, durch dessen Auswahl die Relation zum passenden Benutzertyp hergestellt wird. Daher wird die Bewertung **umsetzbar** (4 Punkte) vergeben.

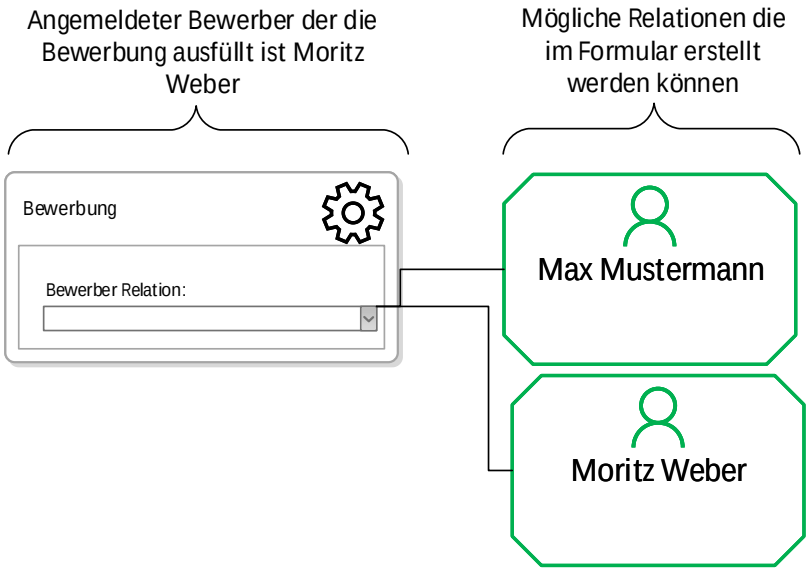

Abbildung 5.1: Beispiel zu Relationenvergabe.

Konsistenz des Systems bei der Modellierung Die Konsistenzprüfung der Modellierungsoberfläche ist im Moment noch nicht ausreichend, da z.B. Strings als Entscheidungsgrundlage in werteabhängigen Mikroschritten genutzt werden können, jedoch sich beim Ändern eines Stringattributes (z.B. es wird der erlaubte Wert des Strings von Mail auf E-Mail geändert) die im werteabhängigen Mikroschritt liegende Expression

nicht automatisch mitändert, was dann zu Verklemmungen in der Ausführung des Prozesses führen kann, da der String gar keinen gültigen Wert zum Weiterfortschreiben im Prozess mehr annehmen kann. Daher ist zwar alles modellierbar, aber es können Modellierungsfehler entstehen was die Bewertung **umsetzbar** (4 Punkte) ergibt.

Datentypen Mit den definierten Datentypen lassen sich 90 Prozent der Fälle abdecken, es wären jedoch weitere speziellere Datentypen wie Uhrzeit oder Mailadresse wünschenwert. In der Modellierung konnten sich diese als Workaround mit den vorhandenen Datentypen ersetzen lassen. Daher ergibt sich die Bewertung **Umsetzbar** (4 Punkte) benötigt.

Modellierung zeitlicher Aspekte Zeitliche Aspekte werden zurzeit nicht unterstützt, jedoch wären Timer, welche regelmäßig etwas ausführen, oder ein maximales Zeitlimit zur Bearbeitung eines Formulares wünschenswert. In der Modellierung wäre es z.B. gut gewesen, wenn es sich hätte darstellen lassen können, dass ein Entscheider innerhalb eines gewissen Zeitraumes seine Entscheidung abgegeben haben muss. Da dies noch nicht existiert, ergibt sich die Bewertung **umfangreiche Erweiterungen** (1 Punkt) benötigt.

Zwischenergebniss Allgemeines

Die Summation der gewichteten Scores ergibt den Wert 54. Wird aus diesem der gewichtete Mittelwert der einzelnen Kriterien gebildet, so ergibt sich der Score 3,38 für die Allgemeinen Kriterien.

5.5 Zusammenfassung der Auswertungen Teil 1- Teil 3

In der Tabelle 6.1 werden die Punktezahlen der Auswertungen Teil 1- Teil 3 zu einer Gesamtpunktezahl verrechnet, welche im Anschluss daran interpretiert wird.

Tabelle 5.1: Zusammenfassung der Evaluationsergebnisse

Kategorie	**Anforderung**	**Score**	**Gewicht**	**Score gewichtet**
ESS Datensicht	Antrag auf Datenänderung	5	1	5
ESS Datensicht	Mitarbeiterdaten	5	1	5
ESS Datensicht	Privatanschrift	5	1	5
ESS Datensicht	Persönliche Kontaktdaten des Mitarbeiters	4	1	4
ESS Datensicht	Dienstliche Kontaktdaten des Mitarbeiters	4	1	4
ESS Datensicht	Bankverbindungen	3	1	3
ESS Datensicht	Berufsweg	5	1	5
ESS Datensicht	Bildungsweg	5	1	5
ESS Datensicht	Kenntnisse	5	1	5
ESS Datensicht	Kontaktpersonen	5	1	5
ESS Datensicht	Passbild	3	1	3
ESS Datensicht	Seminare	5	1	5
ESS Datensicht	Darstellung von Änderungen	3	2	6
	Summe:		**14**	**60**
	Mittelwert:			**4,29**
ESS Rollendefinition	Rolle MA	5	2	10
ESS Rollendefinition	Rolle MA-ESS	5	2	10
ESS Rollendefinition	Rolle MA-Verwaltung	4	2	8
	Summe:		**6**	**28**
	Mittelwert:			**4,67**

Tabelle 5.1: Zusammenfassung der Evaluationsergebnisse Fortsetzung

Kategorie	**Anforderung**	**Score**	**Gewicht**	**Score gewichtet**
ESS Prozesssicht	Zustände von Anträgen	5	2	10
	Summe:		**2**	**10**
	Mittelwert:			**5**
	Summe ESS:		**22**	**98**
	Score ESS:			**4,45**
FZ Datensicht	Fehlzeitenantrag	1	2	2
FZ Datensicht	Urlaubskonto	3	2	6
FZ Datensicht	Vertreter	1	2	2
FZ Datensicht	Entscheidung	3	2	6
FZ Datensicht	Mitarbeiterkalender	0	0	0
FZ Datensicht	Stornierung	2	2	4
FZ Datensicht	Mitarbeiter	5	1	5
	Summe:		**11**	**31**
	Mittelwert:			**2,82**
FZ Rollendefinition	Rollen	5	2	10
FZ Rollendefinition	Rolle Entscheider	5	2	10
	Summe:		**4**	**20**
	Mittelwert:			**5**
FZ Prozesssicht	Fehlzeitenantrag	2	2	4
FZ Prozesssicht	Vertreterregelung	3	1	3
FZ Prozesssicht	Entscheidung	3	2	6
FZ Prozesssicht	Stornierung	1	2	2
FZ Prozesssicht	Zurückziehen eines Antrags	4	2	8

Tabelle 5.1: Zusammenfassung der Evaluationsergebnisse Fortsetzung

Kategorie	**Anforderung**	**Score**	**Gewicht**	**Score gewichtet**
FZ Prozesssicht	Mitarbeiterkalender	0	0	0
	Summe:		**9**	**21**
	Mittelwert:			**2,33**
	Summe FZ:		**24**	**72**
	Score FZ:			**3,00**
Allgemeines	Koordination der Abläufe	5	2	10
Allgemeines	Einbindung vorhanderner Systeme	3	2	6
Allgemeines	Rechtemanagement	4	2	8
Allgemeines	Kommunikation mit	3	2	6
Allgemeines	Personen von Extern			
Allgemeines	Beziehungen von Objekten	4	2	8
Allgemeines	Konsistenz bei der Modellierung	3	2	6
Allgemeines	Datentypen	4	2	8
Allgemeines	Modellierung zeitlicher Aspekte	1	2	2
	Summe:		**16**	**56**
	Mittelwert:			**3,5**
	Summe Allgemein:		**16**	**56**
	Score Allgemein:			**3,50**

Tabelle 5.1: Zusammenfassung der Evaluationsergebnisse Fortsetzung

Kategorie	Anforderung	Score	Gewicht	Score gewichtet
	Summe ESS:		**22**	**98**
	Score ESS:			**4,45**
	Summe FZ:		**24**	**72**
	Score FZ:			**3,00**
	Summe Allgemein:		**16**	**56**
	Score Allgemein:			**3,50**
	Score Gesamt:			**3,65**

Ein Durchschnitt von 3,65 sagt laut Bewertungsschema aus, dass der Prozess mit PHILharmonicFlows grundlegend umsetzbar ist, jedoch Erweiterungen benötigt werden welche momentan durch Workarounds abgebildet wurden. Betrachten wir in diesem Zusammenhang das Kreisdiagramm in Abbildung 5.2, welches die Verteilung der einzelnen Bewertungen zeigt, so fällt zunächst auf, dass es keine einzige für PHILharmonicFlows relevante Bewertung mit 0 Punkten gegeben hat. Dies bedeutet, dass es keine Anforderung gibt, welche gar nicht umsetzbar ist auch wenn es umfangreiche Erweiterungen geben würde. Weiter fällt auf dass sehr oft die Bewertung 3 vergeben wurde. Dies sind in der Regel Bewertungen, bei denen zur Umsetzbarkeit eine Blackboxaktivität oder eine einfache Berechnung zur Umsetzbarkeit benötigt wurde.

Es wurde jedoch auch oft die Bewertung 1 vergeben, welche sagt, dass die Anforderung so nicht umgesetzt werden konnte und dass das Konzept für eine Umsetzbarkeit umfangreiche Erweiterungen benötigen würde, welche aber durchaus entwickelbar sind. Auf diese Probleme werden wir noch detailliert zu sprechen kommen. Hierzu wird zuerst das vorhandene Problem beschrieben werden, um im Anschluss daran Vorschläge auszuarbeiten, wie eine Erweiterung des PHILharmonicFlows Konzeptes aussehen könnte, um die genannte Problematik zu beseitigen.

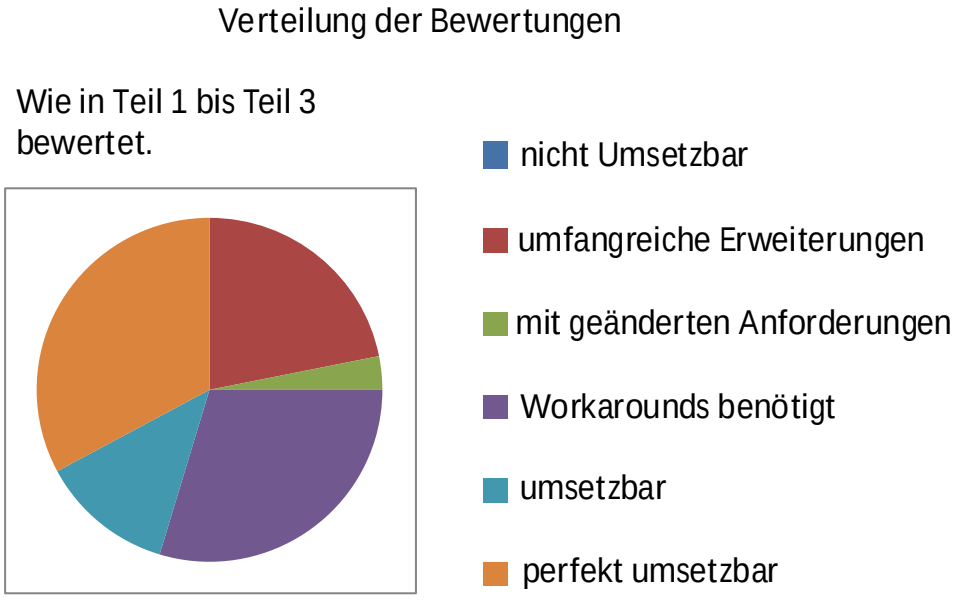

Abbildung 5.2: Verteilung der Punktevergabe

Betrachtet man die zwei Module ESS und FZ, so bemerkt man einen deutlichen Unterschied in den individuellen Scores, ESS 4,45 und FZ 2,83. Dies kommt daher, dass die Anforderungen der beiden Module auf verschiedene Arten definiert wurden. Bei der Definition der Anforderungen zum ESS wurden die Daten sehr in den Vordergrund gestellt, bei der FZ eher die Prozessabläufe. Die Art der Definition der ESS Anforderungen ist optimal für eine Umsetzung mit PHILharmonicFlows geeignet gewesen. Würde man die Anforderungen zur FZ auf dieselbe Art und Weise definieren, so ist zu vermuten dass sich das auch positiv auf den Score auswirken würde. Der Allgemeine Score liegt zwischen dem ESS und der FZ. Dies ist auch realistisch, da Einflüsse aus beiden Modulen eingeflossen sind.

5.6 Zusammenfassung der Probleme und Herausforderungen von PHILharmonicFlows

In dieser Sektion werden die Probleme, welche bei der Evaluation aufgetreten sind, detailiert beschrieben. Einge dieser Herausforderungen sind nicht gänzlich unerwartet, da PHILharmonicFlows erst seit 2 Jahren entwickelt wird und anfangs das Konzept einen engen Fokus auf Formularen hatte.

5.6.1 Probleme im Datenmodell und bei den Attributen

Die in PHILharmonicFlows definierten Attributtypen sind noch nicht ausreichend für eine Modellierung eines umfangreichen Systemes. Diese Problematik gliedert sich ein mehrere kleine Teilberreiche welche im Folgenden erläutert werden.

Der Number Typ

In der Modellierung des FZ Systemes soll der Urlaubsanspruch in Tagen mit einer Nachkommastelle verwaltet werden, Der Numbertyp kann bisher nur auf ganze Zahlen eingeschränkt werden, es wäre hierbei nützlich, den Typ zukünftig auch auf eine gewisse Anzahl an Nachkommastellen beschränken zu können.

Uhrzeiten

Bei der Entscheidung eines Entscheiders zu einem Fehlzeitenantrag soll das Datum sowie die Urzeit der Entscheidung abgespeichert werden. Für das Datum existiert bereits ein Datumstyp, für die Uhrzeit ist noch kein spezialisierter Typ vorhanden. Zwar ist ein Workaround über den Numbertyp möglich, aber nicht optimal, Daher sollte ein zusätzliches Attribut Uhrzeit eingeführt werden, um die Verwaltung und Kennzeichnung von Urzeiten zu verbessern.

Mail Addresen

Es müssen immer wieder Mailadressen verwaltet werden, eine Mailadresse ist wenn Formulare verarbeitet werden ein häufig auftretender Wertetyp, welcher spezielle Eigenschaften aufweist. Daher wäre es sinnvoll für Mailadressen einen speziellen Datentyp zu schaffen, welcher dann auch nur plausible Mailadressen zulässt. Alternativ könnte auch eine Einschränkung zum Textattribut definiert werden, welche Mailadressen behandelt.

Attribute mit vom System generierter Wertebelegung

Es gibt momentan keine Möglichkeit, Attribute bereits zur Modellierungszeit mit vordefinierten Werten zu befüllen. Für das durch das System automatische Befüllen von Attributen gibt es zwei verschiedene Varianten: Erstens das Befüllen von mit zur Modellierungszeit festgelegten Werten. Hier könnte man sich zum Bsp. ein Attribut der Bezeichnung Land bei der Abfrage eines der privaten Adresse eines Mitarbeiters vorstellen. Da das System in einem Land verwendet wird, ist das Land des Wohnsitzs eines Mitarbeiters vermutlich auch das Land in dem die Firma ihren Sitz hat. Hier würde es zur Laufzeit Arbeitsaufwand sparen, wenn bereits das entsprechende Land in Attribut vordefiniert ist. Der zweite Fall ist das Befüllen mit vom System stammenden Werten. Ein sehr klassisches Beispiel hierfür wäre der Mikroprozess einer Entscheidung zu einem Fehlzeiteneintrag. Es soll dabei das Datum und die Uhrzeit automatisch festgehalten werden an der die Entscheidung getroffen wurde. Diese Werte vom Benutzer händisch ausfüllen zu lassen ist unsinnig, da dies auch automatisch erledigt werden kann.

File Typ

Zu jedem Mitarbeiter soll ein Passbild abgelegt werden zur Ablage eines Passbildes existiert momentan ein Attribut mit dem Typ File. Dieses Attribut könnte jedoch viel tiefergehende Eigenschaften definiert haben. So fehlt eine Einschränkung auf Dateitypen um zu verhindern dass z.B. anstatt einer Bilddatei ein ausführbare Programmdatei hochgeladen wird. Daher sollte der Typ um Einschränkungen zum erlaubten Dateityp ergänzt werden.

5.6.2 Probleme im Mikroprozess

In dieser Sektion wird auf Probleme eingegangen, die in direktem Bezug zu den Mikroprozessen stehen.

Automatische Formulargenerierung anhand von Laufzeitdaten

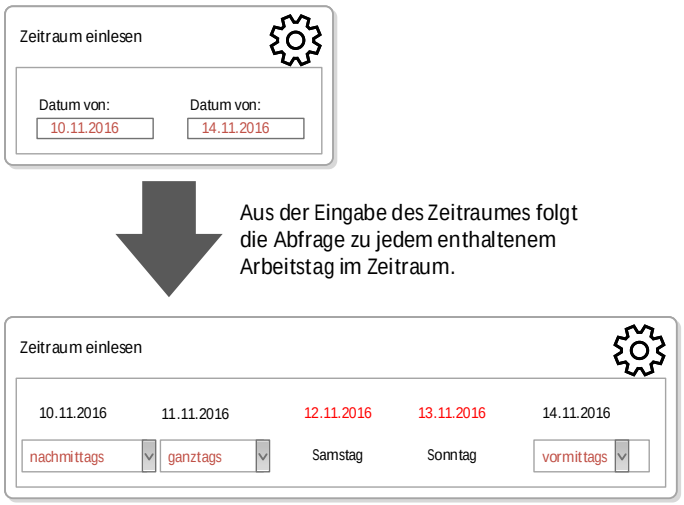

Abbildung 5.3: Darstellung Der Zeitraumabfrage eines Fehlzeitenantrages.

In einem Fehlzeitenantrag soll der Zeitraum der Fehlzeit eingegeben werden. Aus diesem Zeitraum heraus soll automatisch eine Abfrage für jeden Arbeitstag im Fehlzeitraum erstellt werden. Die gewünschte Vorgehensweise wird durch Abbildung 5.3 verdeutlicht.

Leider hat PHILharmonicFlows momentan keine Möglichkeiten, entsprechend auf Runtimedaten dynamisch zu reagieren. Das System müsste somit um eine Komponente erweitert werden, welche eine automatische Formulargenerierung anhand von Laufzeitdaten unterstützt.

Durchführen von Berechnungen auf Attributen

Zu jeder Bankverbindung soll die BLZ und die Kontonummer eingegeben werden. Da jedoch heute im Normalfall die IBAN verwendet wird, welche sich aus der BLZ und der Kontonummer generieren lässt, soll diese ebenfalls mit verwaltet werden. Ein automatisches Generieren und durchführen von Berechnungen auf Attributen ist jedoch zurzeit nicht möglich.

Konsistenzprobleme

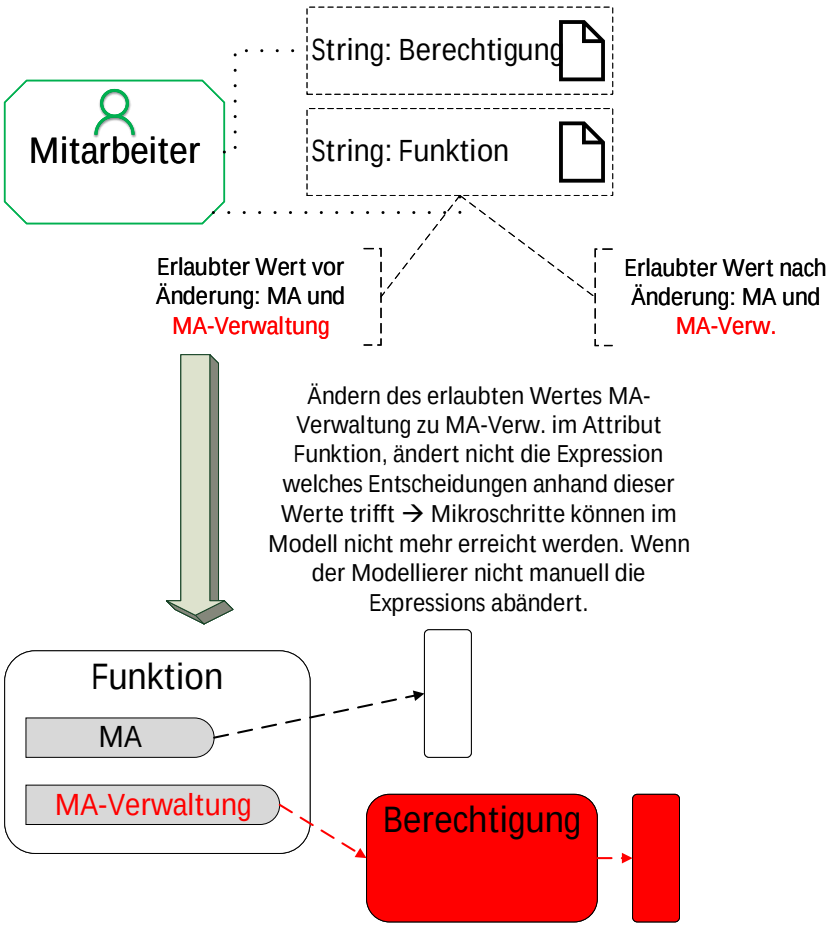

Abbildung 5.4: Konsistenzprobleme der Modellierungsumgebung.

Prädikatsschritte können sich in der Expression auf Attributwerte vom Typ Text beziehen. Bei der Attributdefinition kann der Wertebereich eines Textattributes auf feststehende Werte eingeschränkt werden. Wird nun ein Textattribut mit einem Prädikatsschritt ausgewertet, so wird oftmals als Vergleichswert zur Auswertung einer der Vorab definierten Werte des Wertebereiches des Attributes verwendet. Ändert man nun in der Modellierung einen erlaubten Wert des Attributes, so ändert sich eine Expression, welche Bezug auf den Wert nimmt, nicht und die Bedingung kann somit nicht mehr erfüllt werden. Hier sollte ein Warnmechanismus eingebaut werden, welcher abprüft, ob ein Attribut den Wert der Expression überhaupt annehmen kann. Dies wird auch in Abbildung 5.4 dargestellt.

Zugriff auf andere Objekttypen und deren Attribute

Im ESS sollen die jeweils aktuellen Daten zu einem Mitarbeiter in einem Objekt verwaltet werden. Ein konkretes Beispiel stellen z.B. die persönlichen Daten des Mitarbeiters dar. Änderungen auf diese Daten sollen per Antrag gestellt werden. Wird ein Antrag auf Datenänderung genehmigt, so soll dieser Antrag zur Dokumentation in der Datenhaltung verbleiben und die aktuellen Mitarbeiterdaten aktualisiert werden. Der Zusammenhang wird durch Abbildung 5.5 verdeutlicht. Eine Aktualisierung der Daten könnte natürlich händisch im Objekttyp persönliche Daten durch einen Mitarbeiter erfolgen. Jedoch ist es nicht Sinn und Zweck, manuell Daten zu kopieren. Daher müsste eine Möglichkeit geschaffen werden, welche es erlaubt, Werte eines anderen Objekttypen zu manipulieren, wenn dieser Objekttyp sich in einem Zustand befindet, welcher externe Manipulationen zulässt.

Ein weiterer Fall der Zugriff auf Attribute von anderen Objekten erfordert, sind Prädikatsschritte, um eine Entscheidung auch von einem Attributwert eines anderen Objektes abhängig machen zu können. Hierbei sollte abhängig davon was in einem Ausschreibungsobjekt an Qualifikationen gefordert wird eine Abfrage dieser Qualifikationen in einem Bewerbungsformular erfolgen. Diese Qualifikationen werden aber durch ein separates Objekt dargestellt.

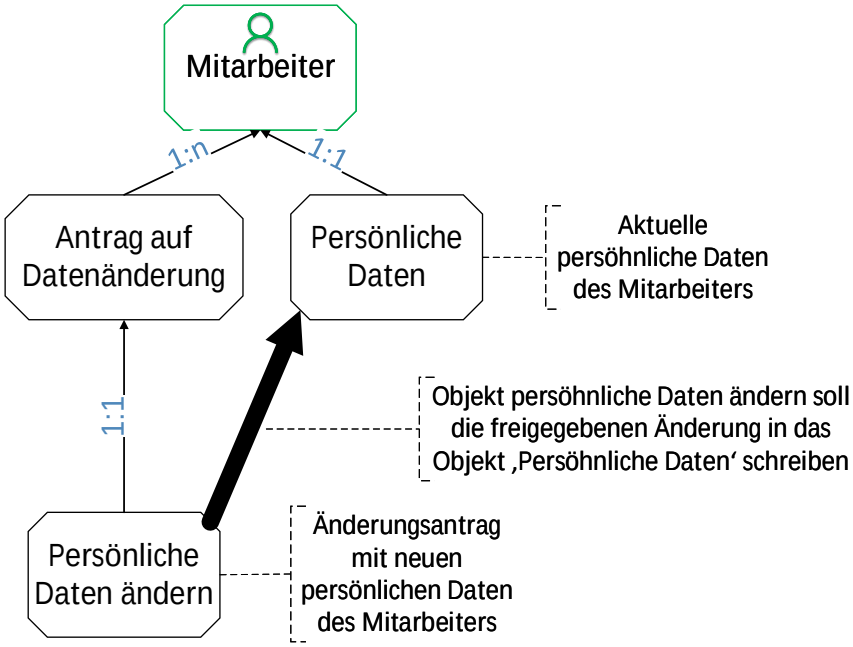

Abbildung 5.5: Zugriff auf Attribute eines anderen Objektes.

Interaktion mit anderen Systemen

Prozesse fordern immer wieder eine Interaktion mit Systemen außerhalb von PHILharmonicFlows. So sollte es z.B. möglich sein, Mails zu versenden. Hierzu muss mit einem Mailserver kommuniziert werden. Weitere vorstellbare Szenarien sind z.B. dass Prozessdaten durch Fremdsysteme wie z.B. eine SQL Datenbank bereitgestellt werden. Auch denkbar ist die Umkehrung dessen, z.B. dass PHILharmonicFlows Werte für andere Systeme bereitstellen sollte wie um z.B. einen Kalender mit allen Fehlzeiten zu erstellen. Für diese Aufgaben sollten feste Workflows definiert werden.

Einschränken von semantisch passenden Objekten im Relationenattribut

Im ESS Prozess werden immer wieder Relationenattribute verwendet, um den Bezug zu anderen Objekttypen herzustellen. So muss z.B. bei einem Antrag auf Datenänderung

ein Relationenattribut Mitarbeiter geschrieben werden, welches dann die Relation zwischen dem Mitarbeiter und dem Antrag auf Datenänderung herstellt. Es stehen für da Attribut jedoch alle Mitarbeiterobjekte zur Auswahl, welche sich im System befinden. Aber der Mitarbeiter sollte sich nur selbst, bzw. evtl. noch Mitarbeiter auswählen können, für welche er autorisiert ist einen Antrag zu stellen. Um dies kontrollieren zu können, muss ein erweitertes Berechtigungskonzept für Relationenattribute ausgearbeitet werden, welches es erlaubt, das <Erstellen von Relationen einzuschränken.

Temporale Aspekte

Wird eine Fehlzeit genehmigt, so kann diese wieder storniert werden, solange diese nicht stattgefunden hat. Der zeitliche Zusammenhang lässt sich mit PHILharmonicFlows nicht modellieren, da bisher keinerlei zeitlichen Aspekte konzeptionell vorgesehen wurden. Ein weiteres Beispiel für zeitliche Abhängigkeiten wäre ein Vertreter, der für eine beantragte Fehlzeit seine Entscheidung innerhalb von 48 Stunden abgeben muss. Gibt er in dieser Zeit keine Entscheidung ab, wird automatisch der Antrag abgelehnt.

5.6.3 Probleme im Makroprozesse

Der kommende Abschnitt behandelt Probleme, welche im Zusammenhang mit Makroprozessen stehen.

Anlegen von Objekten

Mithilfe der Makroprozesse können Objektinstanzen untereinander koordiniert werden. Jedoch kann ein Makroprozess nicht das Erstellen einer Objektinstanz zu verbieten, wie dies z.B. in der FZ sinnvoll wäre. Hier würde es Sinn ergeben, dass eine Entscheidung erst instanziiert werden kann, wenn es einen Antrag auf Fehlzeit gibt, zu dem noch keine Entscheidungen verlinkt wurden.

5.6.4 Probleme bei der Benutzerintegration

Abfragen des am System angemeldeten Benutzers im Datenkontext

Wird ein Fehlzeitenvertreter gewünscht, dann muss dieser vom Antragssteller ausgewählt werden. Ist der Vertreter ausgewählt, so muss dieser der Vertretung zustimmen. Dieser Vorgang wird mithilfe eines vereinfachten Datenmodells in Abbildung 5.6 und eines vereinfachten Mikroprozesses in Abbildung 5.7 dargestellt. Die Rollen MA, MA-ESS und MA-Verwaltung haben jeweils auf den Attributen Mitarbeiter, Vertreter gewünscht, Vertreter und Entscheidung Vertreter ein Schreibrecht, welches instanzspezifisch ist.

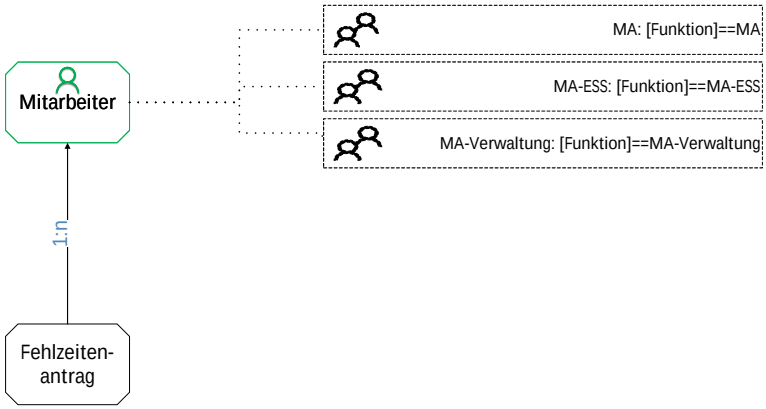

Abbildung 5.6: Vereinfachtes Datenmodel zur Vertreterregelung

Als Beispiel dient folgender Ablauf, welcher auch in Abbildung 5.7 als Mikroprozess verdeutlicht wird: Der Mitarbeiter Meier stellt einen Antrag auf Fehlzeit, das bedeutet, dass das Attribut Mitarbeiter eine Relation vom Objekt Fehlzeitantrag zum Benutzertyp (Instanz Meier) herstellt. Herr Meier gibt nun an, dass er einen Vertreter auswählen möchte und wählt im Anschluss Mitarbeiter Weber als Vertreter aus. Das Attribut Vertreter stellt nun eine Relation vom Objekt Fehlzeitenantrag zum Benutzertyp Mitarbeier (Instanz Weber) her. Somit haben im System sowohl der Mitarbeiter Meier als auch der Mitarbeiter Weber eine Relation zwischen dem Mitarbeiterobjekt und dem Objekt Fehlzeitenantrag und somit auch beide Mitarbeiter das instanzspezifische Recht, die Zustimmung zur

Vertretung zu schreiben, was aber eig. nur Mitarbeiter Weber tun dürfte. Auch eine Vergabe des Rechtes über eine Relationenrolle würde keine Abhilfe schaffen, da beide Mitarbeiter dieselbe Relation besitzen.

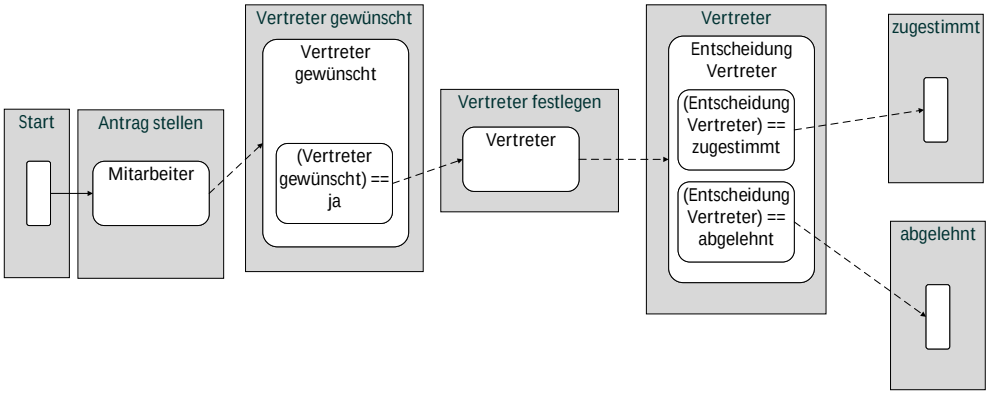

Abbildung 5.7: Vereinfachter Mikroprozess zur Vertreterregelung

PHILharmonicFlows stellt momentan für dieses Problem keine Lösung bereit. Es wäre jedoch vorstellbar, die Problematik mithilfe eines Datenkontextes bei der Rechtevergabe zu lösen. Allerdings sind die hierfür existierenden Expressions nicht mächtig genug, da nicht abgefragt werden kann, wer der am System angemeldete Benutzer ist. Würde man die Expressions um diese Möglichkeit erweitern, könnte man mit dem Ausdruck (angemeldeter Benutzer) == (Vertreter) die Rechtevergabe entsprechend einschränken.

Vererbung bei Rollen

In der Modellierung des ESS und der FZ werden drei Kontextrollen verwendet. Die Rolle MA ist die Rolle mit den wenigsten Rechten, die Rolle MA-ESS beinhaltet alle Rechte der Rolle MA und ergänzt diese um weitere Rechte. Die Rolle MA-Verwaltung hat wiederum alle Rechte, welche auch die Rolle MA-ES besitzt. Im Moment müssen die gemeinsamen Rechte für jede Rolle redundant definiert werden. Sehr viel praktischer und auch weniger fehleranfällig wäre ein Vererbungskonzept, mit dem Rechte von anderen Rollen geerbt werden können Siehe auch Abbildung 3.1.

6

Neue Evaluation unter Berücksichtung von Weiterentwicklungen

Im Laufe der Masterarbeit wurde PHILharmonicFlows auf Basis der Kritikpunkte, welche in dieser Arbeit aufgedeckt wurden weiterentwickelt. In diesem Kapitel werden die Ergebnisse der Modellierung der Anforderungen aus Kapitel 3 erneut evaluiert und ein neuer Score aufgestellt, welcher die Erweiterungen berücksichtigt.

Die maßgeblichen Neuerungen, welche berücksichtigt werden, sind Berechnungsschritte, welche das Ausführen von einfachen Berechnungen im Schritt erlauben. Weiter gibt es nun Proxyattribute, welche es erlauben, auf Attribute eines anderen Objekttypen zuzugreifen. Es ist jetzt möglich, Attribute mit vordefinierten Werten zu versehen, sowie Expressions zu definieren, welche das Erstellen von Relationen zur Laufzeit einschränken.

6.1 Zusammenfassung der Auswertungen Teil 1- Teil 3 unter Berücksichtigung der Erweiterungen

In der Tabelle 6.1 werden die Punktezahlen der Auswertungen Teil 1- Teil 3 unter Berücksichtigung der Neuerungen neu definiert und zu einer Gesamtpunktezahl verrechnet, welche im Anschluss daran interpretiert wird. Scores welche sich verbessert haben, sind in der Tabelle hervorgehoben notiert.

Tabelle 6.1: Zusammenfassung der Evaluationsergebnisse

Kategorie	**Anforderung**	**Score**	**Gewicht**	**Score gewichtet**
ESS Datensicht	Antrag auf Datenänderung	5	1	5
ESS Datensicht	Mitarbeiterdaten	5	1	5
ESS Datensicht	Privatanschrift	5	1	5
ESS Datensicht	Persönliche Kontaktdaten des Mitarbeiters	4	1	4
ESS Datensicht	Dienstliche Kontaktdaten des Mitarbeiters	4	1	4
ESS Datensicht	Bankverbindungen	**5 (3)**	1	**5**
ESS Datensicht	Berufsweg	5	1	5
ESS Datensicht	Bildungsweg	5	1	5
ESS Datensicht	Kenntnisse	5	1	5
ESS Datensicht	Kontaktpersonen	5	1	5
ESS Datensicht	Passbild	**5 (3)**	1	**5**
ESS Datensicht	Seminare	5	1	5
ESS Datensicht	Darstellung von Änderungen	**5 (3)**	2	**10**
	Summe:		**14**	**68**
	Mittelwert:			**4,86**
ESS Rollendefinition	Rolle MA	5	2	10
ESS Rollendefinition	Rolle MA-ESS	5	2	10
ESS Rollendefinition	Rolle MA-Verwaltung	4	2	8
	Summe:		**6**	**28**
	Mittelwert:			**4,67**

Tabelle 6.1: Zusammenfassung der Evaluationsergebnisse Fortsetzung

Kategorie	**Anforderung**	**Score**	**Gewicht**	**Score gewichtet**
ESS Prozesssicht	Zustände von Anträgen	5	2	10
	Summe:		**2**	**10**
	Mittelwert:			**5**
FZ Datensicht	Fehlzeitenantrag	1	2	2
FZ Datensicht	Urlaubskonto	**5 (2)**	2	**10**
FZ Datensicht	Vertreter	1	2	2
FZ Datensicht	Entscheidung	**5 (3)**	2	**10**
FZ Datensicht	Mitarbeiterkalender	0	0	0
FZ Datensicht	Stornierung	2	2	4
FZ Datensicht	Mitarbeiter	5	1	5
	Summe:		**11**	**39**
	Mittelwert:			**3,55**
FZ Rollendefinition	Rollen	5	2	10
FZ Rollendefinition	Rolle Entscheider	5	2	10
	Summe:		**4**	**20**
	Mittelwert:			**5**
FZ Prozesssicht	Fehlzeitenantrag	2	2	4
FZ Prozesssicht	Vertreterregelung	**5 (3)**	1	**5**
FZ Prozesssicht	Entscheidung	**5 (3)**	2	**10**
FZ Prozesssicht	Stornierung	1	2	2
FZ Prozesssicht	Zurückziehen eines Antrags	4	2	8
FZ Prozesssicht	Mitarbeiterkalender	0	0	0
	Summe:		**9**	**27**
	Mittelwert:			**3,00**

Tabelle 6.1: Zusammenfassung der Evaluationsergebnisse Fortsetzung

Kategorie	**Anforderung**	**Score**	**Gewicht**	**Score gewichtet**
Allgemeines	Koordination der Abläufe	5	2	10
Allgemeines	Einbindung vorhanderner Systeme	**5 (3)**	2	**10**
Allgemeines	Rechtemanagement	**5 (4)**	2	**10**
Allgemeines	Kommunikation mit Personen von Extern	**5 (3)**	2	**10**
Allgemeines	Beziehungen von Objekten	**5 (4)**	2	**10**
Allgemeines	Konsistenz bei der Modellierung	3	2	6
Allgemeines	Datentypen	4	2	8
Allgemeines	Modellierung zeitlicher Aspekte	1	2	2
	Summe:		**16**	**68**
	Mittelwert:			**4,25**
	Summe ESS:		**22**	**106**
	Score ESS:			**4,82**
	Summe FZ:		**24**	**86**
	Score FZ:			**3,58**
	Summe Allgemein:		**16**	**68**
	Score Allgemein:			**4,25**
	Score Gesamt:			**4,22**

Werden die Ergebnisse betrachtet so fällt auf, dass sich fast alle Ergebnisse deutlich verbessern. ein Gesamtscore von 4,22 liegt nun im ganz oberen Bereiches des aufgestellten Schemas und drückt aus, dass die Module mit PHIlharmonicFlows weitestgehend den Anforderungen entsprechend umsetzbar sind. Auch die Fehlzeitenverwaltung verbessert

sich deutlich, bleibt aber aus den bereits bei der ersten Bewertung genannten Gründen an letzter Stelle.

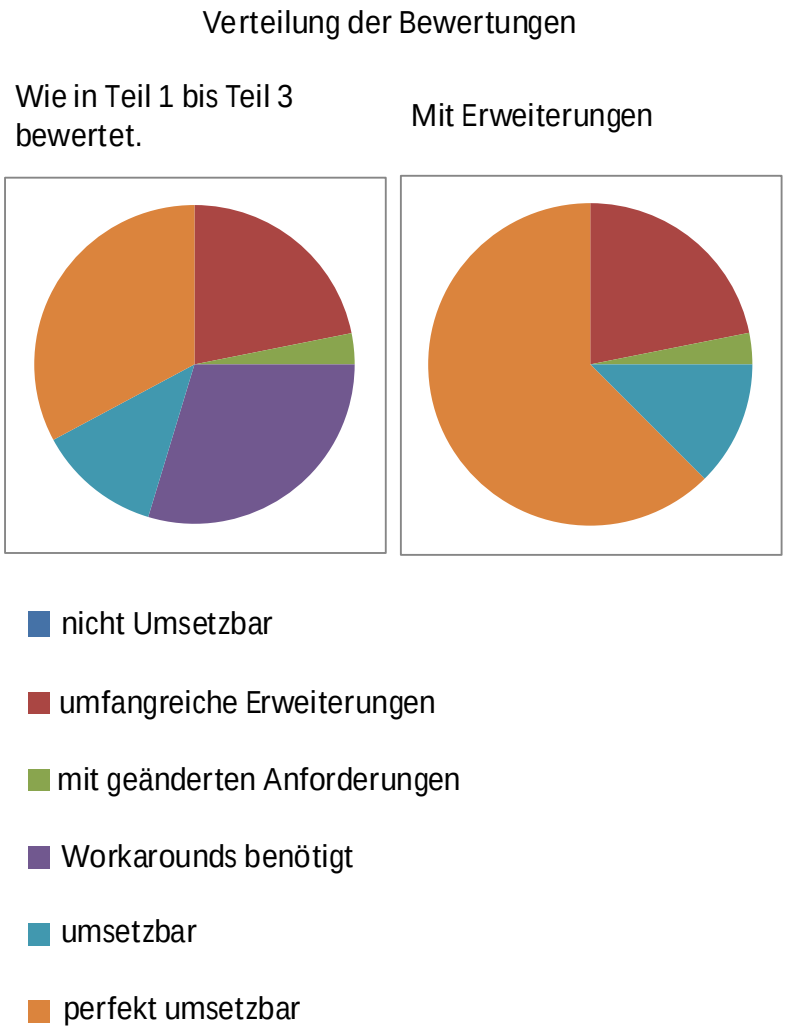

Abbildung 6.1: Darstellung Der Punkteverteilung mit und ohne Erweiterungen.

Die Verteilung der einzelnen Bewertungen im Vergleich zur ersten Evaluation wird im Kreisdiagramm in der Abbildung 6.1 dargestellt. Hierbei fällt auf, dass fast alle Bewertungen welche im ersten Durchgang mit Workarounds benötigt (3 Punkte) bewertet

wurden, inzwischen mit perfekt umsetzbar (5 Punkte) bewertet sind. Dies resultiert daraus, dass die fehlenden Workarounds in der Zwischenzeit zu PHILharmonicFlows als Konzept hinzugefügt wurden

7

Verwandte Arbeiten

In diesem Kapitel wird ein Vergleich dieser Masterarbeit mit anderen Arbeiten gezogen, welche thematische Ähnlichkeiten besitzen. Weiter werden diese anderen Arbeiten diskutiert.

7.1 Verwandte Arbeiten

Dies ist die erste Arbeit, welche PHILharmonicFlows evaluiert. Jedoch gibt es viele Arbeiten, welche sich mit datengetriebenen Ansätzen wie z.B. auch dem artifaktzentrierten Ansatz oder objektorientierten Ansätzen sowie mit PHILharmonicFlows befassen.

Vera Künzle analysiert in ihrer Dissertation Object-Aware-Process-Management [15]bestehende Prozesse, die von existierendem PrMS nicht ausreichend unterstützt werden. Als wesentliche Erkenntnis wurde deutlich, dass in vielen Anwendungsszenarien eine umfassende Prozessunterstützung sowohl das Objekt- als auch das Prozessbewusstsein erfordert. Dies bedeutet, dass Geschäftsprozesse und Geschäftsdaten nicht unabhängig voneinander behandelt werden dürfen. Basierend auf den identifizierten Eigenschaften erarbeitet diese Arbeit wesentliche Anforderungen, um objektzentrierte Prozessmanagementsysteme zu ermöglichen. Das Ergebnis dieser Arbeit ist das PHILharmonicFlows-Framework, das diese Anforderungen umsetzt und ein objektzentriertes Prozessmanagement ermöglicht.

Carolina Chiao, Vera Künzle und Manfred Reichert beschäftigen sich in dem Paper Object-aware Process Support in Healthcare Information Systems: Requirements, Conceptual Framework and Examples [6] mit Prozessen im Gesundheitswesen, welche

von aktivitätenzentrierten Prozessmanagementsystemen aufgrund der fehlenden Integration von Daten nur unzureichend unterstützt werden. Auf diesem Hintergrund wird ein Brustkrebs Diagnose-Szenario mithilfe von PHILharmonic Flows abgebildet. Anhand dieser Ergebnisse wird nun die Mächtigkeit der Modellierungskomponente von PHILharmonicFlows evaluiert. Weiter werden Einblicke in die operative Semantik der Prozessausführung in PHILharmonicFlows gegeben.

Vera Künzle analysiert in ihrem Paper PHILharmonicFlows: towards a framework for object-aware process management [17] bestehende Prozesse, welche von aktivitätenzentrieten Prozessmanagementsystemen nur unbefriedigend umgesetzt werden. Weiter stellt die Arbeit die grundlegenden Eigenschaften objektbewusster Prozesse sowie grundlegende Anforderungen für ihre operative Unterstützung dar. Darüber hinaus wird das PHILharmonicFlows-Framework vorgestellt, welches diese Anforderungen adressiert und ein objektbewusstes Prozessmanagement ermöglicht.

Thomas Spindler beschäftigt sich in seiner Masterarbeit Integration der Modellierungs- und Laufzeitumgebung eines datenorientierten Prozess-Management-Systems [24] mit der Entwicklung eines Rahmenwerkes innerhalb des PHILharmonicFlows Projektes, welches eine flexible Modellierung und Ausführung von objektzentrierten Prozessen erlaubt und eine integrierte Sicht auf Prozesse und Daten zur Verfügung stellt. Hierzu wurde in der Arbeit ein Prototyp der Modellierungs- und der Laufzeitumgebung für PHILharmonicFlows entwickelt.

Mark Popa beschäftigt sich in seiner Diplomarbeit Evaluierung bestehender Konzepte für datenorientiertes Prozessmanagement [22] im Rahmen einer Literaturrecherche mit einer Abgrenzung von aktivitätenzentrierten Ansätzen zu datenzentrierten Ansätzen. Weiter wurden aus der Forschung zu PHILharmonicFlows Herausforderungen für datenzentrierte Prozessmanagementsysteme gewonnen. Diese Herausforderungen werden in der Arbeit gegen die datenzentrierten Konzepte evaluiert. Weiter wird noch das von der OMG entwickelte, CMMN untersucht. Dabei wird erforscht, inwieweit dieses den Ansprüchen eines objektzentrierten Systems gerecht wird und wie es Wissensarbeiter bei ihrer Arbeit unterstützen kann. Weiterhin wird untersucht, in welchem Umfang CMMN bei der Modellierung strukturschwacher und datengetriebener Prozesse hilft und ob

diese Prozesse auch mit anderen Notationen, wie beispielsweise BPMN 2.0, modelliert werden könnten.

Angelina Henschel beschäftigt sich in ihrer Bachelorarbeit Konzept und Design einer grafischen Benutzeroberfläche eines datenzentrierten Prozess-Management-Systems [12] mit der Entwicklung eines Usability-Konzept für die Laufzeitumgebung des PHILharmonicFlows Frameworks. Dies beinhaltet auch die Gestaltung einer geeigneten Navigationsstruktur, eine effiziente Aufteilung der Benutzeroberfläche und eine harmonische Farbwahl sowohl für die Benutzeroberfläche als auch den Mikro-Prozess Graphen. Hierbei wurde Wert auf die Beachtung gängiger Normen und Vorgehensweisen für die Gestaltung von Benutzeroberflächen gelegt.

Sebastian Steinau beschäftigt sich in seiner Masterarbeit Design and Implementation of a Runtime Environment of an Object-Aware Process Management System [25] mit Konzepten und Implementierungen hinsichtlich Prozessregeln, welche die Durchführung von Mikro- und Makroprozessen im PHILharmonicFlows Projekt steuern.

Thomas Haselbeck beschäftigt sich in seiner Masterarbeit Evaluierung und Ausarbeitung eines Berechtigungskonzepts für Human Resource Management [10] mit einem Konzept zur Implementierung eines Berechtigungskonzepts. Hierfür klärt er zunächst die Grundlagen der Zugriffskontrolle, so dass daraus aufbauend notwendige Komponenten eines Berechtigungskonzepts definiert werden können. Weiter werden basierend auf den Grundlagen von datenzentrierten Prozessmanagement Anforderungen definiert. Als Ergebnis entsteht ein theoretisches Berechtigungskonzept für datenorientiertes Prozessmanagement welches auch Herausforderungen einer Umsetzung aufzeigt.

Richard Hull beschäftigt sich in der Arbeit Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events [13] mit Artefakten, einem anderen Ansatz für datenorientierter Prozessmanagementsysteme, welche Somit einen alternativen Ansatz für zu PHILharmonicFlows darstellen würden. In diesem Kontext wird das Guard-Stage-Milestone (GSM) Metamodell untersucht, welches den Lebenszyklus eines Artifakts beschreibt. Im speziellen wird darauf eingegangen, wie die formale operative Semantik von GSM Interaktion zwischen Artefakt-Instanzen unterstützt.

Ebenfalls mit GSM beschäftigen sich Damaggio in der Arbeit On the equivalence of incremental and fixpoint semantics for business artifacts with Guard–Stage–Milestone lifecycles [7]. Das GSM Konzept ist von Natur aus deklarativ und ermöglicht eine gleichzeitige Ausführung von langwierigen (möglicherweise von Menschen ausgeführten) Aktivitäten. In dieser Ausarbeitung wird nachgewiesen dass es drei verschiedene Formulierungen der GSM-Operations-Semantik gibt, welche gleichwertig sind.

Das Paper Artifact-based transformation of IBM global financing [5] behandelt eine Artefaktbasierte Prozessmodellierung, welche als eine Alternative zu PHILharmonicFlows betrachtet werden kann. Diese Prozessmodellierung betrachtet wichtige Business-Artifakte in einem Unternehmen, identifiziert diese und verfolgt ihre Lebenszyklen so wie sie im Unternehmen auftreten. Aus diesen realen Artefakten wird ein Business-Operationen-Modell aufgestellt, welches zu einem voll funktionsfähiger Rapid Prototyp führt. Das daraus resultierende Geschäftsmodell wurde als Grundlage für alle Aspekte der Geschäftsumwandlung in IBM Global Financing genutzt.

Mit einer Erweiterung des GSM Modells befasst sich Bagheri-Hariri in dem Paper Verification of semantically-enhanced artifact systems [9]. In dieser Arbeit wird das Modell um eine semantische Schicht ergänzt, welche eine konzeptionelle Sicht auf die durch das Modell zu untersuchende Domäne bietet und es somit erlaubt, die Entwicklung des Artefaktsystems auf einer höheren Abstraktionsstufe zu sehen. Konkret wird eine Technik vorgestellt um temporale Eigenschaften zu spezifizieren. Diese Technologie wurde in einem Tool implementiert, das modernste ontologiebasierte Datenzugriffstechnologien nutzt um die zeitlichen Eigenschaften nach der Ontologie und den Zuordnungen zu manipulieren.

8

Fazit und Ausblick

In diesem Kapitel wird ein Fazit aus der Zielsetzung und den gewonnenen Ergebnissen gezogen. Abgeschlossen wird diese Arbeit mit einem Ausblick auf zukünftige Entwicklungsmöglichkeiten von PHILharmonicFlows.

8.1 Fazit

Ziel dieser Arbeit war die Evaluation des Modellierungskonzeptes des objektzentrierten Prozessmanagementsystemes PHILharmonicFlows. Hierzu werden reale Anforderungen einer Modulneuentwicklung der Persis GmbH verwendet. Zu diesem Zweck wurden in Kapitel 2 die Grundlagen der Modellierung von Prozessen mit PHILharmonicFlows, sowie der Hintergrund der Persis Unternehmenssoftware Software erläutert. In Kapitel 3 wurden Anforderungen an die neuen Module der Persis Unternehmenssoftware spezifiziert. Diese Grundlagen dienen als Ausgangspunkt für die Ergebnisse dieser Ausarbeitung. Hierbei lassen sich die Ergebnisse in verschiedene Bereiche aufteilen.

Ein erstes Ergebnis ist eine Umsetzung der spezifizierten Persis Unternehmenssoftwaremodule mit PHILharmonicFlows wie sie in Kapitel vier abgebildet wurden. Diese Umsetzung stellt das momentan umfassendste Prozessmodell dar, welches mit dem System PHILharmonicFlows bisher modelliert wurde. Das Ergebnis zeigt dass PHILharmonicFlows für ein Projekt dieser Größenordnung geeignet ist.

Ein weiteres Ergebnis ist das Bewertungsschema in Kapitel 5 welches die Umsetzung mit PHILharmonicFlows anhand eines leitfadenorientierten Expertenurteils evaluiert. Hier zeigt sich, dass sich PHILharmonicFlows für Projekte wie die Neuentwicklung der

Persismodule sehr gut eignen würde, wenn es auch noch stellensweise an Funktionalität fehlt.

Als letztes Ergebnis entsteht ebenfalls in Kapitel 4 eine Auflistung der Problematiken bei der Umsetzung der Anforderungen. Weiter werden hier die Probleme detailliert ausgearbeitet und Lösungsansätze vorgeschlagen, welche bei einer zukünftigen Weiterentwicklung von PHILharmonikFlows beachtet werden können.

Abschließend lässt sich sagen, dass sich der Ansatz von PHILharmonicFlows sehr gut für die Umsetzung von datenintensiven Softwaresystemen, wie Sie auch die Firma Persis entwickelt, eignet. Es sollten jedoch die Konzepte von PHILharmonicFlows noch weiter verfeinert und um weitere Komponenten ergänzt werden um reale Anforderungen komplett abdecken zu können. So ist es für einen realen Einsatz des Systemes unabdingbar, dass Konzepte für die Kommunikation zu anderen Systemen, Konzepte zum Zugriff auf Attribute anderer Objekte sowie Ergänzungen im Rechtemanagement hinzugefügt werden.

8.2 Ausblick

Diese Arbeit zeigt den Stand der Modellierungskomponeten des objektzentrierten Prozessmanagementsystemes PHILharmonicFlows. So bietet das System großes Potenzial, welches weiter ausgebaut werden sollte. Es wird für die Zukunft sehr wichtig sein, die aufgezeigten Probleme weiter zu präzisieren und verschiedene detaillierte Lösungskonzepte auszuarbeiten um einen Praxiseinsatz zu ermöglichen. Weiterhin sollte in Zukunft auch die Laufzeit der modellierten Systeme einer näheren Betrachtung unterzogen werden, um z.B. die Frage, wie das modellierte Konzept sinnvoll in eine Benutzeroberfläche eingebunden werden kann, zu betrachten. Wichtig wäre auch die Frage: Was ist zu beachten wenn zwei unabhängig voneinander modellierte Modelle in ein gemeinsames Modell überführt werden sollen.

Diese Arbeit stellt somit den Ausgangspunkt für eine Weiterentwicklung der Modellierungskonzepte von PHILharmonicFlows dar. Die aufgetretenen Probleme werden weiterhin adressiert und mit Erweiterungen gelöst (Siehe auch Kapitel 7) Basierend auf

diesen Erweiterungen können künftig weitere praxisnahe Einsatzmöglichkeiten evaluiert werden.

Literaturverzeichnis

[1] Persis GmbH. Homepage (15 April 2016)

[2] Persis GmbH. Anforderungsdokument Persis ESS (2017)

[3] Persis GmbH. Anforderungsdokument Persis Fehlzeitenverwaltung (2017)

[4] Belkin, N.J., Croft, W.B.: Information Filtering and Information Retrieval: Two Sides of the Same Coin? Communications of the ACM 35(12), 29–38 (1992)

[5] Chao, T., Cohn, D., Flatgard, A., Hahn, S., Linehan, M., Nandi, P., Nigam, A., Pinel, F., Vergo, J., y Wu, F.: Artifact-based Transformation of IBM Global Financing. In: International Conference on Business Process Management. pp. 261–277. Springer (2009)

[6] Chiao, C.M., Künzle, V., Reichert, M.: Object-aware Process Support in Healthcare Information Systems: Requirements, Conceptual Framework and Examples. No. 1 & 2, International Journal on Advances in Life Sciences (2013)

[7] Damaggio, E., Hull, R., Vaculín, R.: On the Equivalence of Incremental and Fixpoint Semantics for Business Artifacts with Guard–Stage–Milestone Lifecycles. Information Systems 38(4), 561–584 (2013)

[8] Dumas, M., van der Aalst, W., ter Hofstede, A.: Process-aware Information Systems: Bridging People and Software Through Process Technology. John Wiley & Sons (2005)

[9] Hariri, B., Calvanese, D., Montali, M., Santoso, A., Solomakhin, D.: Verification of Semantically-enhanced Artifact Systems. In: International Conference on Service-Oriented Computing. pp. 600–607. Springer (2013)

[10] Haselbeck, T.: Evaluierung und Ausarbeitung eines Berechtigungskonzepts für Human Resource Management. Master's thesis, University of Ulm (2014)

[11] Hegner, M.: Methoden zur Evaluation von Software. IZ, InformationsZentrum Sozialwissenschaften (2003)

[12] Henschel, A.: Konzept und Design einer grafischen Benutzeroberfläche eines datenzentrierten Prozess-Management-Systems. Bachelor thesis, University of Ulm (2015)

[13] Hull, R., Damaggio, E., De Masellis, R., Fournier, F., Gupta, M., Heath III, F.T., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., et al.: Business Artifacts with Guard-Stage-Milestone Lifecycles: Managing Artifact Interactions with Conditions and Events. In: 5th ACM International Conference on Distributed Event-based System. pp. 51–62. ACM (2011)

[14] 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: International Workshop on Web Services and Formal Methods. pp. 1–24. Springer (2010)

[15] Künzle, V.: Object-aware Process Management. Ph.D. thesis, University of Ulm (2013)

[16] Künzle, V., Reichert, M.: Towards Object-aware Process Management Systems: Issues, Challenges, Benefits. In: Enterprise, Business-Process and Information Systems Modeling, pp. 197–210. Springer (2009)

[17] Künzle, V., Reichert, M.: PHILharmonicFlows: Towards a Framework for Object-aware Process Management, vol. 23. Wiley Online Library (2011)

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

[19] Müller, D., Reichert, M., Herbst, J.: A New Paradigm for the Enactment and Dynamic Adaptation of Data-driven Process Structures. In: International Conference on Advanced Information Systems Engineering. pp. 48–63. Springer (2008)

[20] Mutschler, B., Reichert, M., Bumiller, J.: Unleashing the Effectiveness of Process-oriented Information Systems: Problem Analysis, Critical Success Factors,

and Implications. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 38(3), 280–291 (2008)

[21] Nandi, P., Kumaran, S.: Adaptive Business Objects - A New Component Model for Business Integration. In: ICEIS (3). pp. 179–188 (2005)

[22] Popa, M.: Evaluierung bestehender Konzepte für datenorientiertes Prozessmanagement. Diploma thesis, University of Ulm (2015)

[23] Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies. Springer (2012)

[24] Spindler, T.: Integration der Modellierungs- und Laufzeitumgebung eines datenorientierten Prozess-Management-Systems. Master's thesis, University of Ulm (2013)

[25] Steinau, S.: Design and Implementation of a Runtime Environment of an Object-aware Process Management System. Master's thesis, University of Ulm (February 2015)

[26] Weske, M.: Business Process Management Architectures. In: Business Process Management, pp. 333–371. Springer (2012)

Abbildungsverzeichnis

Tabellenverzeichnis

Anhänge

1 Überblick über die Attribute aller Objekttypen

Tabelle 1: Attribute zu den Objekttypen des ESS und der FZ

Attribut Name	Attributtyp	Eigenschaften
Mitarbeiter		
Benutzername	String	
Password	String	
Funktion	String	Werte: MA, MA-ESS, MA-Verwaltung
Entscheider Fehlzeit	Referenzliste	Referenz zu Mitarbeiter
Antrag auf Datenänderung		
zurückziehen	String	Werte: Ja
Entscheidung Sachbearbeiter	String	Werte: Genehmigt, abgelehnt
Typ der Änderung	String	Werte: Persöhnliche Daten, Bankverbindung, Bildungsweg, Berufsweg, Seminar, Kenntnis, Kontaktperson, Passbild
persönliche Daten ändern	Relation	persönliche Daten ändern zu Antrag auf Datenänderung
Bankverbindung ändern	Relation	Bankverbindung ändern zu Antrag auf Datenänderung
Bildungsweg ändern	Relation	Bildungsweg ändern zu Antrag auf Datenänderung
Berufsweg ändern	Relation	Berufsweg ändern zu Antrag auf Datenänderung
Seminar ändern	Relation	Seminar ändern zu Antrag auf Datenänderung
Kenntnis ändern	Relation	Kenntnis ändern zu Antrag auf

Tabelle 1: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Attributtyp	Eigenschaften
		Datenänderung
Kontaktperson ändern	Relation	Kontaktperson ändern zu Antrag auf Datenänderung
Passbild ändern	Relation	Passbild ändern zu Antrag auf Datenänderung
Mitarbeiter	Relation	Antrag auf Datenänderung zu Mitarbeiter
Persönliche Daten		
Titel	String	
Betriebstitel	String	
Name	String	
Vorname	String	
Namenszusatz	String	
Vorsatzwort	String	
Geburtsname	String	
Geburtsnamensvorsatz	String	
Geburtsnamenszusatz	String	
Geburtsort	String	
Geburtsland	String	
Staatsangehörigkeit 1	String	
Staatsangehörigkeit 2	String	
Konfession	String	
Familienstand	String	
Straße	String	
Ort	String	
Postfach	String	
Region	String	
Land	String	
Bundesland	String	

Tabelle 1: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Attributtyp	Eigenschaften
Straße 2	String	
Ort 2	String	
Postfach 2	String	
Region 2	String	
Land 2	String	
Bundesland 2	String	
E-Mail	String	
E-Mail 2	String	
E-Mail dienstlich	String	
Gebäude dienstlich	String	
Büro dienstlich	String	
Ausweis dienstlich	String	
Zutrittsprofil	String	
Geburtstag	Date	
Anzahl Kinder	Number	ganzzahlig
Anzahl Kinder unter 18	Number	ganzzahlig
Hausnummer	Number	ganzzahlig
PLZ	Number	ganzzahlig
Hausnummer 2	Number	ganzzahlig
PLZ 2	Number	ganzzahlig
Telefon	Number	ganzzahlig
Kurzwahl	Number	ganzzahlig
Telefax	Number	ganzzahlig
Handy	Number	ganzzahlig
Telefon 2	Number	ganzzahlig
Kurzwahl 2	Number	ganzzahlig
Telefax 2	Number	ganzzahlig
Handy 2	Number	ganzzahlig
Telefon dienstlich	Number	ganzzahlig

Tabelle 1: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Attributtyp	Eigenschaften
Telefax dienstlich	Number	ganzzahlig
Kurzwahl dienstlich	Number	ganzzahlig
Handy dienstlich	Number	ganzzahlig
Mitarbeiter	Relation	persönliche Daten zu Mitarbeiter
Bankverbindung		
IBAN	String	
BIC	String	
Bankname	String	
Kontoinhaber	String	
Kontoart	String	
Straße	String	
Ort	String	
Postfach	String	
Ansprechpartner	String	
Gültig von	Date	
Gültig bis	Date	
BLZ	Number	ganzzahlig
Kontonummer	Number	ganzzahlig
Hausnummer	Number	ganzzahlig
PLZ	Number	ganzzahlig
Mitarbeiter	Relation	Bankverbindung zu Mitarbeiter
Passbild		
Status	String	Werte: Gelöscht, gesperrt
Mitarbeiter	Relation	Passbild zu Mitarbeiter
Passbild	File	

Tabelle 1: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Attributtyp	Eigenschaften
persönliche Daten ändern		
Titel	String	
Betriebstitel	String	
Name	String	
Vorname	String	
Namenszusatz	String	
Vorsatzwort	String	
Geburtsname	String	
Geburtsnamensvorsatz	String	
Geburtsnamenszusatz	String	
Geburtsort	String	
Geburtsland	String	
Staatsangehörigkeit 1	String	
Staatsangehörigkeit 2	String	
Konfession	String	
Familienstand	String	
Straße	String	
Ort	String	
Postfach	String	
Region	String	
Land	String	
Bundesland	String	
Straße 2	String	
Ort 2	String	
Postfach 2	String	
Region 2	String	
Land 2	String	
Bundesland 2	String	
E-Mail	String	

Tabelle 1: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Attributtyp	Eigenschaften
E-Mail 2	String	
E-Mail dienstlich	String	
Gebäude dienstlich	String	
Büro dienstlich	String	
Ausweis dienstlich	String	
Zutrittsprofil	String	
Geburtstag	Date	
Anzahl Kinder	Number	ganzzahlig
Anzahl Kinder unter 18	Number	ganzzahlig
Hausnummer	Number	ganzzahlig
PLZ	Number	ganzzahlig
Hausnummer 2	Number	ganzzahlig
PLZ 2	Number	ganzzahlig
Telefon	Number	ganzzahlig
Kurzwahl	Number	ganzzahlig
Telefax	Number	ganzzahlig
Handy	Number	ganzzahlig
Telefon 2	Number	ganzzahlig
Kurzwahl 2	Number	ganzzahlig
Telefax 2	Number	ganzzahlig
Handy 2	Number	ganzzahlig
Telefon dienstlich	Number	ganzzahlig
Telefax dienstlich	Number	ganzzahlig
Kurzwahl dienstlich	Number	ganzzahlig
Handy dienstlich	Number	ganzzahlig
Bankverbindung ändern		
Art der Änderung	String	Werte: Ändern, neu
IBAN	String	

Tabelle 1: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Attributtyp	Eigenschaften
BIC	String	
Bankname	String	
Kontoinhaber	String	
Kontoart	String	
Straße	String	
Ort	String	
Postfach	String	
Ansprechpartner	String	
Gültig von	Date	
Gültig bis	Date	
BLZ	Number	ganzzahlig
Kontonummer	Number	ganzzahlig
Hausnummer	Number	ganzzahlig
PLZ	Number	ganzzahlig
Datensatz	Reference	Referenz zu Bankverbindung
Passbild ändern		
Art der Änderung	String	Werte: Neu, ändern, sperren, löschen
Passbild	File	
Datensatz	Reference	Referenz zu Passbild
Fehlzeitenantrag		
Antragsart	String	
Bemerkung Vertreter	String	
Vertreter gewünscht	String	Werte: Ja, Nein
Entscheidung Vertreter	String	Werte: Zugestimmt, abgelehnt
Stornierung	String	Werte: Ja, Fehlzeit hat stattgefunden
Bemerkung	String	
Antragsdatum	Date	

Tabelle 1: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Attributtyp	Eigenschaften
Datum von	Date	
Datum bis	Date	
Mitarbeiter	Relation	Fehlzeitenantrag zu Mitarbeiter
Vertreter	Relation	Fehlzeitenantrag zu Mitarbeiter
Entscheidungen	Relationenliste	Fehlzeitenantrag zu Entscheidung
Urlaubskonto		
Jahr	Number	ganzzahlig
Urlaubsanspruch in Tagen	Number	
Mitarbeiter	Relation	Urlaubskonto zu Mitarbeiter
Entscheidung		
Bemerkung	String	
Entscheidung	String	Werte: Genehmigt, abgelehnt
Uhrzeit	String	
Entscheidungsdatum	Date	
Entscheider	Relation	Entscheidung zu Mitarbeiter

2 Überblick über die Rechtevergabe

Die einzelnen Rollen erhalten in der folgenden Darstellung die Zahlen 1-4 als Kürzel. 1 wird der Rolle MA zugeordnet, 2 der Rolle MA-ESS, 3 der Rolle Ma-Verwaltung, 4 der Rolle Entscheider und 5 der Rolle Antragsteller. Alle Rechte werden wenn nicht anders genannt, als Lese- und Schreibrecht und instanzspezifisch ohne Datenkontext vergeben.

Tabelle 2: Rechtevergabe

Attribut Name	Zustand	Recht an Rolle
Mitarbeiter		
Benutzername	Initialisierung	1
Password	Initialisierung	1
Funktion	Initialisierung	1
Entscheider Fehlzeit	Initialisierung	1
Antrag auf Datenänderung		
zurückziehen	eingereicht 1	2,3
Entscheidung Sachbearbeiter	eingereicht 3	
Typ der Änderung	Antrag initialisieren	5
persönliche Daten ändern	spezifische Werte persönliche Daten ändern	5
Bankverbindung ändern	spezifische Werte Bankverbindung ändern	5
Bildungsweg ändern	spezifische Werte Bildungsweg ändern	5
Berufsweg ändern	spezifische Werte Berufsweg ändern	5
Seminar ändern	spezifische Werte Seminar ändern	5
Kenntnis ändern	spezifische Werte Kenntnis ändern	5
Kontaktperson ändern	spezifische Werte Kontaktperson ändern	5
Passbild ändern	Spezifische Werte Passbild ändern	5
Mitarbeiter	Antrag initialisieren	2,3

Tabelle 2: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Zustand	Recht an Rolle
Persönliche Daten		
Titel	Mitarbeiterdaten	3
Betriebstitel	Mitarbeiterdaten	3
Name	Mitarbeiterdaten	3
Vorname	Mitarbeiterdaten	3
Namenszusatz	Mitarbeiterdaten	3
Vorsatzwort	Mitarbeiterdaten	3
Geburtsname	Mitarbeiterdaten	3
Geburtsnamensvorsatz	Mitarbeiterdaten	3
Geburtsnamenszusatz	Mitarbeiterdaten	3
Geburtsort	Mitarbeiterdaten	3
Geburtsland	Mitarbeiterdaten	3
Staatsangehörigkeit 1	Mitarbeiterdaten	3
Staatsangehörigkeit 2	Mitarbeiterdaten	3
Konfession	Mitarbeiterdaten	3
Familienstand	Mitarbeiterdaten	3
Straße	Privatanschrift	3
Ort	Privatanschrift	3
Postfach	Privatanschrift	3
Region	Privatanschrift	3
Land	Privatanschrift	3
Bundesland	Privatanschrift	3
Straße 2	Privatanschrift	3
Ort 2	Privatanschrift	3
Postfach 2	Privatanschrift	3
Region 2	Privatanschrift	3
Land 2	Privatanschrift	3
Bundesland 2	Privatanschrift	3
E-Mail	private Kontaktdaten	3

Tabelle 2: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Zustand	Recht an Rolle
E-Mail	private Kontaktdaten	3
E-Mail dienstlich	dienstliche Kontaktdaten	3
Gebäude dienstlich	dienstliche Kontaktdaten	3
Büro dienstlich	dienstliche Kontaktdaten	3
Ausweis dienstlich	dienstliche Kontaktdaten	3
Zutrittsprofil	dienstliche Kontaktdaten	3
Geburtstag	Mitarbeiterdaten	3
Anzahl Kinder	Mitarbeiterdaten	3
Anzahl Kinder	Mitarbeiterdaten	3
Hausnummer	Privatanschrift	3
PLZ	Privatanschrift	3
Hausnummer 2	Privatanschrift	3
PLZ 2	Privatanschrift	3
Telefon	private Kontaktdaten	3
Kurzwahl	private Kontaktdaten	3
Telefax	private Kontaktdaten	3
Handy	private Kontaktdaten	3
Telefon 2	private Kontaktdaten	3
Kurzwahl 2	private Kontaktdaten	3
Telefax 2	private Kontaktdaten	3
Handy 2	private Kontaktdaten	3
Telefon dienstlich	dienstliche Kontaktdaten	3
Telefax dienstlich	dienstliche Kontaktdaten	3
Kurzwahl dienstlich	dienstliche Kontaktdaten	3
Handy dienstlich	dienstliche Kontaktdaten	3
Mitarbeiter	Verknüpfung zum Mitarbeiter	3
Bankverbindung		
IBAN	Bankverbindung	3

Tabelle 2: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Zustand	Recht an Rolle
BIC	Bankverbindung	3
Bankname	Bankverbindung	3
Kontoinhaber	Bankverbindung	3
Kontoart	Bankverbindung	3
Straße	Bankverbindung	3
Ort	Bankverbindung	3
Postfach	Bankverbindung	3
Ansprechpartner	Bankverbindung	3
Gültig von	Bankverbindung	3
Gültig bis	Bankverbindung	3
BLZ	Bankverbindung	3
Kontonummer	Bankverbindung	3
Hausnummer	Bankverbindung	3
PLZ	Bankverbindung	3
Mitarbeiter	Verknüpfung zum Mitarbeiter	3
Passbild		
Status	Bild gespeichert	3
	statusüberwachung	
Mitarbeiter	Verknüpfung zum Mitarbeiter	3
Passbild	Passbild	3
Persönliche Daten ändern		
Titel	Mitarbeiterdaten	5, nur lesen: 3
Betriebstitel	Mitarbeiterdaten	5, nur lesen: 3
Name	Mitarbeiterdaten	5, nur lesen: 3
Vorname	Mitarbeiterdaten	5, nur lesen: 3
Namenszusatz	Mitarbeiterdaten	5, nur lesen: 3
Vorsatzwort	Mitarbeiterdaten	5, nur lesen: 3

Tabelle 2: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Zustand	Recht an Rolle
Geburtsname	Mitarbeiterdaten	5, nur lesen: 3
Geburtsnamensvorsatz	Mitarbeiterdaten	5, nur lesen: 3
Geburtsnamenszusatz	Mitarbeiterdaten	5, nur lesen: 3
Geburtsort	Mitarbeiterdaten	5, nur lesen: 3
Geburtsland	Mitarbeiterdaten	5, nur lesen: 3
Staatsangehörigkeit 1	Mitarbeiterdaten	5, nur lesen: 3
Staatsangehörigkeit 2	Mitarbeiterdaten	5, nur lesen: 3
Konfession	Mitarbeiterdaten	5, nur lesen: 3
Familienstand	Mitarbeiterdaten	5, nur lesen: 3
Straße	Privatanschrift	5, nur lesen: 3
Ort	Privatanschrift	5, nur lesen: 3
Postfach	Privatanschrift	5, nur lesen: 3
Region	Privatanschrift	5, nur lesen: 3
Land	Privatanschrift	5, nur lesen: 3
Bundesland	Privatanschrift	5, nur lesen: 3
Straße 2	Privatanschrift	5, nur lesen: 3
Ort 2	Privatanschrift	5, nur lesen: 3
Postfach 2	Privatanschrift	5, nur lesen: 3
Region 2	Privatanschrift	5, nur lesen: 3
Land 2	Privatanschrift	5, nur lesen: 3
Bundesland 2	Privatanschrift	5, nur lesen: 3
E-Mail	Private Kontaktdaten	5, nur lesen: 3
E-Mail 2	Private Kontaktdaten	5, nur lesen: 3
E-Mail dienstlich	dienstliche Kontaktdaten	5, nur lesen: 3
Gebäude dienstlich	dienstliche Kontaktdaten	5, nur lesen: 3
Büro dienstlich	dienstliche Kontaktdaten	5, nur lesen: 3
Ausweis dienstlich	dienstliche Kontaktdaten	5, nur lesen: 3
Zutrittsprofil	dienstliche Kontaktdaten	5, nur lesen: 3
Geburtstag	Mitarbeiterdaten	5, nur lesen: 3

Tabelle 2: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Zustand	Recht an Rolle
Anzahl Kinder	Mitarbeiterdaten	5, nur lesen: 3
Anzahl Kinder unter 18	Mitarbeiterdaten	5, nur lesen: 3
Hausnummer		5, nur lesen: 3
PLZ	Privatanschrift	5, nur lesen: 3
Hausnummer 2	Privatanschrift	5, nur lesen: 3
PLZ 2	Privatanschrift	5, nur lesen: 3
Telefon	Private Kontaktdaten	5, nur lesen: 3
Kurzwahl	Private Kontaktdaten	5, nur lesen: 3
Telefax	Private Kontaktdaten	5, nur lesen: 3
Handy	Private Kontaktdaten	5, nur lesen: 3
Telefon 2	Private Kontaktdaten	5, nur lesen: 3
Kurzwahl 2	Private Kontaktdaten	5, nur lesen: 3
Telefax 2	Private Kontaktdaten	5, nur lesen: 3
Handy 2	Private Kontaktdaten	5, nur lesen: 3
Telefon dienstlich	dienstliche Kontaktdaten	5, nur lesen: 3
Telefax dienstlich	dienstliche Kontaktdaten	5, nur lesen: 3
Kurzwahl dienstlich	dienstliche Kontaktdaten	5, nur lesen: 3
Handy dienstlich	dienstliche Kontaktdaten	5, nur lesen: 3
Bankverbindung ändern		
Art der Änderung	Typ der Änderung	5, nur lesen: 3
IBAN	Bankverbindung	5, nur lesen: 3
BIC	Bankverbindung	5, nur lesen: 3
Bankname	Bankverbindung	5, nur lesen: 3
Kontoinhaber	Bankverbindung	5, nur lesen: 3
Kontoart	Bankverbindung	5, nur lesen: 3
Straße	Bankverbindung	5, nur lesen: 3
Ort	Bankverbindung	5, nur lesen: 3
Postfach	Bankverbindung	5, nur lesen: 3

Tabelle 2: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Zustand	Recht an Rolle
Ansprechpartner	Bankverbindung	5, nur lesen: 3
Gültig von	Bankverbindung	5, nur lesen: 3
Gültig bis	Bankverbindung	5, nur lesen: 3
BLZ	Bankverbindung	5, nur lesen: 3
Kontonummer	Bankverbindung	5, nur lesen: 3
Hausnummer	Bankverbindung	5, nur lesen: 3
PLZ	Bankverbindung	5, nur lesen: 3
Datensatz	vorhandene Werte einlesen	5, nur lesen: 3
Passbild ändern		
Art der Änderung	Typ der Änderung	5, nur lesen: 3
Passbild	Passbild	5, nur lesen: 3
Datensatz	Vorhandene Werte einlesen:	5, nur lesen: 3
Fehlzeitenantrag		
Antragsart	Antrag stellen	1,2,3
Bemerkung Vertreter	Bemerkung verfassen	1,2,3
Vertreter gewünscht	Vertreter gewünscht	1,2,3
Entscheidung Vertreter	Vertreter	1,2,3
Stornierung	genehmigt aber vor Datum	1,2,3
Bemerkung	ablehnen	3
Antragsdatum	Antrag stellen	1,2,3
Datum von	Antrag stellen	1,2,3
Datum bis	Antrag stellen	1,2,3
Mitarbeiter	Antrag stellen	1,2,3
Vertreter	Vertreter festlegen	1,2,3
Entscheidungen	Entscheidung	3

Tabelle 2: Attribute zu den Objekttypen des ESS und der FZ Fortsetzung

Attribut Name	Zustand	Recht an Rolle
Urlaubskonto		
Jahr	initialisieren	3, nur lesen: 1,2,3,4
Urlaubsanspruch	initialisieren	3, nur lesen: 1,2,3,4
Mitarbeiter	initialisieren	3, nur lesen: 1,2,3,4
Entscheidung		
Bemerkung	abgelehnt	3
Entscheidung	Entscheidung	4
Uhrzeit	Entscheidung	4
Entscheidungsdatum	Entscheidung	4
Entscheider	Verlinkung Entscheider	3

Name: David Rothmaier Matrikelnummer: 747355

Erklärung

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

Ulm, den .....

David Rothmaier