Institut für Architektur von Anwendungssystemen
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3339

Modellierung von Sichten zur zielgruppenspezifischen Visualisierung von Prozesskennwerten

Christian Sachers

Studiengang:	Softwaretechnik
Prüfer:	Jun.-Prof. Dr.-Ing. Dimka Karastoyanova
Betreuer:	M. Sc. Monika Kochanowski M. Sc. Wirt.-Inf. Alexander Nowak
begonnen am:	14. Mai 2012
beendet am:	13. November 2012
CR-Klassifikation:	H.4.1, H.5.3

Zusammenfassung

Workflowmanagement wird in der heutigen Geschäftswelt immer wichtiger. Mit der Nutzung von Prozessmodellen und insbesondere deren automatisierter Ausführung auf Workflowengines entstand die Notwendigkeit, sowohl die einzelnen Instanzen des Prozesses als auch den Prozess als Ganzes zu überwachen. Die überwachten Kenwerte eines Prozesses sollten ebenfalls in einer spezialisierten Sprache beschrieben werden, die sich in die verwendete Prozessmodellierungssprache einfügt.

Um dies zu ermöglichen finden sich in der Literatur verschiedene Ansätze. Allerdings fehlt es, nach unserem besten Wissen, allen diesen Ansätzen an der Möglichkeit, nicht nur die „Was"-Dimension zu modellieren, sondern auch die „Wer"-Dimension. In dieser Arbeit wird daher, basierend auf einem Ansatz zur Prozesskennwertmodellierung, ein Konzept zur Modellierung der „Wer"-Dimension entwickelt. Das Konzept erlaubt es Unternehmen, nicht nur zu spezifizieren, was überwacht wird, sondern auch wer welche Teile des Prozesses überwachen darf.

Ein Prototyp des Konzepts wird basierend auf dem freien, webbasierten Prozessmodelleditor Oryx implementiert. Der Prototyp basiert außerdem auf dem bereits existierenden Prototyp von aPro. Das Konzept wurde darauf aufbauend anhand zweier Beispiele geprüft. Abschließend werden die Ergebnisse zusammengefasst und mögliche Erweiterungen des Konzepts beschrieben.

Abstract

In today's companies, workflow management is getting more and more important. With the use of process models and especially their automatic execution on workflow engines comes the need for monitoring the individual instances of processes, as well as the process as a whole. The performance indicators monitored for a process should also be modeled in a specialised language that integrates with languages used for modeling the process.

To enable this, there are various concepts available in literature. However, to the best of our knowledge, all of these concepts lack the possibility to model not only the "what"-dimension but also the "who"-dimension. In this thesis, a concept is developed based on an existing approach for modeling process performance indicators, that allows companies to model not only what is monitored, but also who is allowed to monitor parts of the process.

A prototype of this concept is then implemented based on the free and open web-based process model editor Oryx. The prototype integrates with the existing prototype of the aPro architecture. The concept is then evaluated by means of two examples. To conclude, a summary and an outlook on possible extensions of the concept are given.

Inhaltsverzeichnis

1. Einleitung **11**
1.1. Problemstellung 12
1.2. Anforderungen 12
1.3. Gliederung der Arbeit 13

2. Verwandte Arbeiten **15**
2.1. Vorhandene Konzepte für Kennwertmodellierung und Monitoring 15
2.2. Vorhandene Sichtenmodellierungkonzepte 23

3. Prüfung vorhandener Ansätze **27**
3.1. Prüfung der Konzepte für Kennwertmodellierung und Monitoring 27
3.2. Prüfung der Sichtenmodellierungskonzepte 30
3.3. Erläuterung der Auswahlkriterien 31
3.4. Auswahl eines Ansatzes 33

4. Sichtenkonzept **37**
4.1. Definition wichtiger Begriffe 37
4.2. Formale Definition des Sichtenkonzepts 38
4.3. XML-Format 40
4.4. Grafische Sprache 47
4.5. Rollenvererbung 51
4.6. Custom Alerts 54
4.7. Einbettung der Sichtengenerierung in bestehende Arbeitsabläufe 55

5. Prototyp **59**
5.1. Verwendete Software 60
5.2. Funktionsbeschreibung 63

6. Evaluation an zwei Beispielen **77**
6.1. Beispiel Automobiltest 77
6.2. Beispiel Versicherung 82
6.3. Ergebnis der Evaluation 90

7. Zusammenfassung und Ausblick **93**
7.1. Zusammenfassung 93

7.2. Ausblick 94

Anhang **99**

A. XML-Schema Definitionen 99

Abbildungsverzeichnis

2.1. Architektur des Monitorings auf Basis von RTML. Abbildung entnommen aus [Bar+06]. 16
2.2. Beispiel eines Prozesses in URN. Abbildung entnommen aus [Che07]. 17
2.3. Goal-Modell zum Prozess in Abbildung 2.2. Abbildung entnommen aus [Che07]. 18
2.4. BPMN-Prozess angereichert mit Elementen der BEMN. Abbildung entnommen aus [DGB07]. 19
2.5. aPro-Architektur. Abbildung entnommen aus [KK12b]. 19
2.6. BPMN-Prozess angereichert mit Elementen aus ProGoalML. Abbildung entnommen aus [KK12b]. 20
2.7. Zusammenhang zwischen verschiedenen Artefakten der aPro-Architektur. Abbildung entnommen aus [KK12b]. 21
2.8. Zielmodell in KAOS. Abbildung entnommen aus [KG06]. 22
2.9. Prozesskennwert in der Ontologie von [RORRC10]. Abbildung entnommen aus [RORRC10] 22
2.10. Beispiel für die Anwendung des Appereance-Patterns auf ein Prozessmodell. Abbildung entnommen aus [SLS10]. 23
2.11. Task in zwei unterschiedlich detaillierten Sichten. Abbildung entnommen aus [Lat10]. 24

4.1. Grafische Darstellung des XML Schemas der Benutzer- und Rollendefinitionssprache. 43
4.2. Grafische Darstellung des XML Schemas der Sichtdefinitionssprache. 45
4.3. Rollen an KPIs unter Verwendung des Annotation-Elements von BPMN. . . . 48
4.4. Rollen an KPIs unter Verwendung einer eigenen Spracherweiterung. 49
4.5. Auftreten derselben Rolle in unterschiedlichen Elementen. 50
4.6. Auftreten derselben Rolle an unterschiedlichen KPIs. 51
4.7. Situationen, in welchen Rollenvererbung zum Einsatz kommen könnte. 52
4.8. Beispiel mit Vorschau für Rolle Manager bei aufwärts gerichteter Rollenvererbung. 53
4.9. Beispiel mit Vorschau für Rolle Manager bei abwärts gerichteter Rollenvererbung mit einem Vorgänger. 53
4.10. Beispiel mit Vorschau für Rolle Manager bei abwärts gerichteter Rollenvererbung mit allen Vorgängern. 54
4.11. Prozess zur Erstellung eines Dashboards in aPro 55

4.12. Prozess zur Erstellung mehrerer Dashboards in aPro durch Aufspaltung der ProGoalML-Datei. . . . 56
4.13. Prozess zur Erstellung mehrerer Dashboards in aPro mit angepasstem Umwandlungsalgorithmus. . . . 56
4.14. Prozess zur Erstellung mehrerer Dashboards in aPro durch Aufspaltung der VisML-Datei. . . . 57

5.1. FMC-Diagramm der wichtigsten Komponenten. In blau Komponenten des Prototypen, in gelb Komponenten des Prototyps von aPro. . . . 59
5.2. aPro-Architektur. Abbildung entnommen aus [KK12b]. . . . 60
5.3. ProGoalML-Elemente und ihre Verknüpfungsmöglichkeiten. Abbildung entnommen aus [KK12b]. . . . 61
5.4. Beispiel der Verwendung des „Associated Roles“ Stencils. . . . 64
5.5. Ausschnitt der Oryx Toolbar mit den drei Menüs des Prototyps. . . . 64
5.6. GUI zum Pflegen der Rollen. . . . 65
5.7. GUI zum Pflegen der Benutzer. . . . 66
5.8. GUI zum Laden einer Datei mit Rollen und Benutzerdaten. . . . 67
5.9. GUI zur Angabe von Meta-Daten. . . . 67
5.10. GUI der Generierungs-Einstellungen. . . . 68
5.11. GUI der Validierungs-Einstellungen. . . . 69
5.12. Diagramm mit einer Vorschau für die Rolle SHE-Manager. . . . 70
5.13. GUI zur Auswahl der Rolle, für die eine Vorschau angezeigt werden soll. . . . 71
5.14. GUI zur Auswahl des Benutzers, für den eine Vorschau angezeigt werden soll. 71
5.15. Beispielhaftes Ergebnis einer Validierung. . . . 72
5.16. GUI zum Generieren von Sichten aus einer VisML-Datei. . . . 74
5.17. GUI zum Generieren von Sichten für ein deployes Dashboard. . . . 75

6.1. Erstellen eines Ziel-Modells. . . . 78
6.2. Fertiges BPMN-Modell. . . . 78
6.3. Mit aPro erweitertes BPMN-Modell. . . . 79
6.4. Fertiges Modell mit Sichtbarkeitsdefinitionen. . . . 80
6.5. Ausschnitt des Dashboards für den Automobiltest vor der Zerlegung in einzelne Sichten. . . . 81
6.6. Generator- und Validatoreinstellungen. . . . 82
6.7. Ausschnitt des Dashboards für den Automobiltest für die Rolle Manager. . . . 83
6.8. Beispiel aus der Versicherungsbranche. . . . 84
6.9. Beispiel aus der Versicherungsbranche mit Kennwerten. . . . 85
6.10. Beispiel aus der Versicherungsbranche mit Kennwerten und Sichten. . . . 87
6.11. Beispiel aus der Versicherungsbranche mit einzeln definierten Sichtbarkeiten. . 88
6.12. Vereinfachung der Sichtendefinition der Rolle „Controlling“ mit abwärts gerichteter Rollenvererbung. . . . 89

6.13. Unerwünschte Veränderung der Sichtbarkeiten der Rolle „Sachbearbeiter“ durch Rollenvererbung. 89
6.14. Dashboard der Rolle Sachbearbeiter des Beispiels aus der Versicherungsbranche. 90

7.1. Für Sichten auf Prozesse nicht eindeutige Situation. 96

1. Einleitung

In der heutigen Welt werden Geschäftsprozesse und insbesondere deren Überwachung, Anpassung und Verbesserung immer wichtiger [GG03]. Die Modellierung von Prozessen und deren automatisierte Ausführung auf Workflowengines bringen viele Vorteile. So kann zum Beispiel die fortschreitende Automatisierung Personalkosten einsparen. Je komplexer ein Prozess ist und je höher der Automatisierungsgrad, umso wichtiger wird auch die Überwachung der Prozesse. Der Grund liegt darin, dass Prozessmodelle weit weniger statisch sind als eigenständige Programme. Beispielsweise werden Tasks erst zur Laufzeit an ein bestimmtes ausführendes System gefunden, oder Prozessmodelle werden zur Laufzeit geändert um Optimierungspotentiale auszuschöpfen. Um jedoch die Ausführung dieser Prozesse zu überwachen werden geeignete Werkzeuge benötigt.

Erschwerend kommt hinzu, dass Änderungen an Prozessen zunehmend komplexer werden und nicht nur von IT-Spezialisten durchgeführt werden [Gar]. Als Folge davon müssen Monitoringlösungen auch für Anwender geeignet sein, welche keine IT-Spezialisten sind. Neben der Verwendung einer geeigneten Monitoringlösung können spezifisch zugeschnittene Sichten fachfremden Anwendern die Arbeit stark erleichtern.

Um die Performanz eines Prozesses zu messen sind Key Performance Indicators (KPI) weit verbreitet. Durch sie werden die maßgebenden Performanzkriterien festgelegt um sie daraufhin überwachen zu können. Zur Überwachung wurden Monitoringwerkzeuge entwickelt, welche die laufenden Prozesse überwachen und die Daten den Nutzern zur Verfügung stellen. Die Daten können dabei je nach System unterschiedlich stark aufbereitet werden und die Darstellung kann je nach System ebenfalls die ganze Bandbreite der Datendarstellungsmöglichkeiten ausnutzen, angefangen bei einfachen Tabellen bis hin zum komplexen Dashboard. Monitoringsysteme ermöglichen mitunter die Überwachung sehr vieler Messwerte eines Prozessmodells. Gepaart mit der Größe realer Prozesse und den unterschiedlichen Spezialgebieten der Nutzer, kann sich eine Überforderung des Nutzers ergeben.

Um den Nutzer optimal bei der Überwachung der Prozesse zu unterstützen, muss er also Zugriff auf möglichst alle relevanten Daten haben. Andererseits soll der Benutzer jedoch nicht mit Daten verwirrt werden, welche für ihn nicht relevant sind, für einen anderen Nutzer aber durchaus von hoher Relevanz sein könnten. Um dies zu erreichen, muss die Darstellung der Daten vom Nutzer und seinen Rollen innerhalb des Unternehmens abhängig gemacht werden können.

1.1. Problemstellung

Ziel dieser Arbeit ist es, ein Sichtenkonzept für das Prozessmonitoring auf Basis von KPIs zu erarbeiten. Die Definition der Sichten soll dabei völlig frei sein um eine Anwendbarkeit in unterschiedlichsten Einsatzbereichen zu gewährleisten.

Als Grundlage werden in dieser Arbeit zwei Beispiele verwendet. Das erste Beispiel entstammt der Versicherungsbranche. Ein weiteres Beispiel ist im Bereich Green-IT angesiedelt. Durch diese sehr unterschiedlichen Anwendungsbereiche mit unterschiedlichen Rollen ist eine gute Überprüfung der Anwendbarkeit auf unterschiedlichste Einsatzbereiche gesichert.

Die in dieser Arbeit gefundene Lösung darf dabei nicht auf Monitoringfunktionalitäten konkreter Implementierungen von Workflowengines basieren, sondern soll auf beliebige Workflowengines anwendbar sein. Die Allgemeingültigkeit soll sogar so weit gehen, dass die Ausführung des Prozesses, welcher Gegenstand des Monitorings ist, nicht auf einer Workflowengine stattfinden muss. Diese Anforderung ergibt sich nicht zuletzt auch aus dem Anwendungsbeispiel Green-IT, da hier oftmals Messwerte zum Einsatz kommen, welche nicht von der Workflowengine erhoben werden, sondern aus unterschiedlichsten Quellen stammen.

Als Modellierungssprache für Prozesse soll, wenn möglich, BMPN 2.0 zum Einsatz kommen, da es weit verbreitet ist und seine grafische Darstellung auch für Menschen intuitiv ist, die keine technischen Kenntnisse besitzen. Für die Modellierung der KPIs sollte ebenfalls eine grafische Sprache zum Einsatz kommen, da auch hier die Einbindung nicht technischer Rollen von Bedeutung ist.

Um das erarbeitete Konzept an den genannten Beispielen überprüfen zu können, soll es prototypisch implementiert werden.

1.2. Anforderungen

1.2.1. Kennwertmodellierungssprache

Die ausgewählte Modellierungssprache für Prozesskennwerte muss die Modellierung von KPIs explizit ermöglichen. Die Sprache muss sich außerdem möglichst nahtlos in die verwendete Prozessmodellierung einfügen. Ist dies nicht der Fall, wird die Komplexität unnötig erhöht und der Einarbeitungsaufwand, insbesondere für Anwender ohne umfassende IT-Kenntnisse, steigt stark an. Aus demselben Grund muss die Sprache intuitiv zu verwenden sein, daher ist eine grafische Sprache einer textuellen Sprache vorzuziehen.

Basierung auf BPMN ist eine weitere positiv zu berücksichtigende Eigenschaft, da BPMN weit verbreitet und für nicht technische Anwender weitgehend intuitiv ist [PCBV10]. Jedoch

ist dies keine zwingende Voraussetzung. Dennoch sollte die zugrunde liegende Sprache zumindest eine gewisse Verbreitung haben.

Eine automatisierte Generierung eines Dashboards aus den Prozesskennwerten ist ebenfalls eine harte Anforderung. Kann aus dem Modell nicht automatisiert ein Dashboard erstellt werden, so kann der Benutzer in seiner alltäglichen Arbeit keinen Nutzen aus ihm ziehen. Das Modell kann dann nur zur Dokumentation dienen.

Wird jedoch die Generierung eines Dashboards ermöglicht, so hat der Benutzer in der täglichen Arbeit ein Werkzeug zur Hand, mit welchem er seine Prozesse zeitnah und komfortabel überwachen und beurteilen kann. Dies ermöglicht ihm auf aktuelle Ereignisse flexibel und aufgrund einer guten Informationsbasis zu reagieren. Das Dashboard muss dabei die Integration beliebiger Datenquellen unterstützen. Nur so ist eine Allgemeingültigkeit des Monitorings gewährleistet, was die Verwendung für unterschiedliche Aufgaben ermöglicht, da viele relevante Informationen oftmals nicht im Worflowmanagementsystem vorhanden sind.

Weiterhin darf eine Ausführung des mithilfe des Dashboards überwachten Prozesses auf einer Workflowengine keine zwingende Voraussetzung sein. Auch heute noch werden in Unternehmen Prozesse nicht immer auf Workflowengines ausgeführt. Durch ein von der Workflowengine losgelöstes Monitoring können auch diese Unternehmen von den Möglichkeiten der Prozessüberwachung profitieren.

1.2.2. Sichtenkonzept

Das Sichtenkonzept muss die Definition von Sichten auf Basis von Rollen ermöglichen. Jeder Rolle soll dabei genau eine Sicht zugeordnet sein. Dadurch soll der intuitive Rollenbegriff der realen Welt nachgebildet werden.

Die Modellierung der Sichten soll mit einer Sprache stattfinden, die sich nahtlos in die bereits bestehende Umgebung von Prozessmodellierungssprache und Kennwertmodellierungssprache einfügt. Daneben soll aber auch eine Speicherung in einem allgemeingültigeren XML-Dialekt möglich sein. Dadurch wird die Anpassbarkeit an andere Kennwertmodellierungssprachen nicht im Vorhinein verbaut.

1.3. Gliederung der Arbeit

In diesem Kapitel wurde die Problemstellung der Arbeit umrissen und die Gründe erläutert. Anschließend wurden die Anforderungen an das zu entwickelnde Konzept beschrieben. In Kapitel 2 werden vorhandene Konzepte aus dem Bereich der Prozesskennwertmodellierung und der Sichtenmodellierung vorgestellt. Darauf aufbauend wird in Kapitel 3 untersucht in

wie weit diese Konzepte den Anforderungen genügen und ob sie somit als Basis für diese Arbeit dienen können.

In Kapitel 4 wird schließlich aufbauend auf den im vorherigen Kapitel gefundenen Ansätzen ein Sichtenkonzept entworfen, welches die Anforderungen erfüllt. Kapitel 5 beschreibt die erstellte prototypische Implementierung des Konzepts. Der Prototyp wird anschließend in Kapitel 6 anhand zweier Beispiele geprüft.

Zuletzt werden in Kapitel 7 die Ergebnisse der Arbeit zusammengefasst und mögliche Erweiterungen aufgezeigt.

2. Verwandte Arbeiten

Im folgenden Kapitel werden existierende Konzepte vorgestellt. Dabei werden Konzepte zweier unterschiedlicher Themenbereiche betrachtet.

Zum einen werden Ansätze zur Modellierung von Prozesskennwerten vorgestellt. Dabei wird insbesondere die Sprache zur Kennwertmodellierung betrachtet und welche Prozessmodellierungssprache zum Einsatz kommt. Ebenfalls von großem Interesse sind die automatische Generierung eines Dashboards aus den Modellen und die Art der Anbindung an eine Workflowengine.

Zum anderen werden Ansätze zur Modellierung von Sichten gesucht. Dabei wird besonderer Wert darauf gelegt, ob das Sichtenkonzept auf Prozesskennwerte anwendbar ist.

2.1. Vorhandene Konzepte für Kennwertmodellierung und Monitoring

Es gibt bereits mehrere Ansätze zur Modellierung von KPIs. Ebenfalls gibt es Ansätze zum Modellieren von Events in Prozessmodellen, welche sich für das Messen von KPIs nutzen lassen. Im Folgenden sollen diese Ansätze vorgestellt werden, insbesondere im Hinblick auf die Erfüllung der Anforderungen.

2.1.1. Ansätze für BPEL

Für das Monitoring von BPEL-Prozessen gibt es in der Literatur mehrere Ansätze. Was allen diesen Ansätzen gemeinsam ist, ist das Fehlen einer grafischen Modellierungssprache, was die Einbindung nicht technischer Benutzer in den Modellierungsprozess erschwert. Außerdem werden in keinem der Ansätze Prozesskennwerte modelliert, sondern das Monitoring wird lediglich auf der Basis von Events betrieben.

[Bar+06] stellt einen Ansatz vor, in welchem mit Hilfe einer separaten Sprache, genannt Run-Time Monitor specification Language (RTML), Monitore definiert werden. Die Definition geschieht dabei getrennt vom Prozessmodell. Die definierten Monitore können sich entweder auf eine konkrete Instanz beziehen, oder auf eine Menge von Ausführungen, von den Autoren als Klasse bezeichnet. So kann man beispielsweise sowohl den Bestellwert

einer Prozessinstanz eines Bestellprozesses überwachen als auch den durchschnittlichen Bestellwert über aller Instanzen.

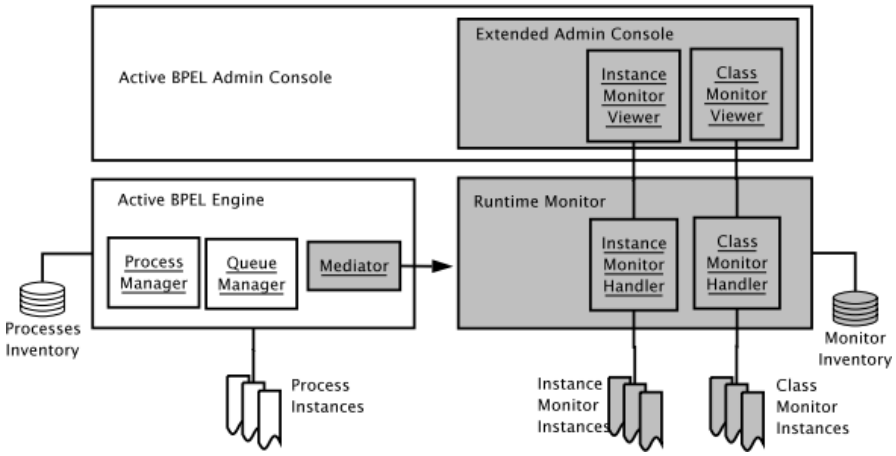

Abbildung 2.1.: Architektur des Monitorings auf Basis von RTML. Abbildung entnommen aus [Bar+06].

Eine automatische Generierung von Java-Code für die Monitore wird ebenfalls unterstützt. Die Monitoring-Engine setzt jedoch die Ausführung des Prozesses auf einer Workflowengine voraus. Die Autoren gehen davon aus, dass eine solche parallel zur Monitoring-Engine im selben Applicationserver betrieben wird.

Ähnlich dazu stellt [BG05] einen Ansatz, in dieser Arbeit als Dynamic Monitoring bezeichnet, vor. Hier werden getrennt vom BPEL-Modell Regeln für das Monitoring definiert. Eine spezielle Komponente, der Monitoring-Manager, ist zur Laufzeit für die Durchführung der in den Regeln definierten Monitoringtätigkeiten verantwortlich. Er entscheidet zur Laufzeit auch über die tatsächliche Ausführung der Monitoringtätigkeiten. Dies geschieht zum Beispiel basierend darauf, welche dynamisch gebundenen Services nun tatsächlich aufgerufen werden, oder aufgrund von Einstellungen. Daher wird der Ansatz von den Autoren als „dynamisch“ bezeichnet.

Zur Verbindung der BPEL-Engine mit dem Monitoring-Manager ist kein Eingriff in die BPEL-Engine nötig, da das Prozessmodell in BPEL um die entsprechenden Aufrufe des Monitoring-Managers erweitert wird. Außerdem ist durch die getrennte Modellierung des Monitorings eine spätere Änderung der Monitoring-Regeln ohne Änderungen am BPEL-Prozess möglich. Beide Modelle können daher nebenläufig und von spezialisierten Fachkräften entwickelt werden. Wie schon der vorherige Ansatz ist auch dieser Ansatz in seiner bisherigen Form auf BPEL beschränkt und setzt die Ausführung des Prozesses auf einer BPEL-Engine voraus.

Ein weiterer Ansatz wurde in [SM06] von Spanoudakis und Mahbub entwickelt. Auch hier werden Webservices oder BPEL-Prozesse überwacht. Wie die vorherigen ist auch dieser

Ansatz jedoch auf BPEL als Prozessmodellierungssprache festgelegt und ermöglicht nicht die grafische Modellierung von KPIs. Die Notwendigkeit der Ausführung auf einer BPEL-Engine ist in diesem Fall zwar nicht gegeben, jedoch wird daneben nur ein Monitoring von Webservices vorgesehen.

2.1.2. Ansatz auf Basis der User Requirement Notation

Ein weiterer Ansatz [Che07] zur Modellierung von Kennwerten für Prozesse basiert auf der User Requirements Notation (URN) [SAZ04]. Zu diesem Zweck wird das Teilmodell Goal-oriented Requirement Language (GRL) der URN erweitert. Die Prozesskennwerte können mit dieser Sprache grafisch modelliert werden. In weiteren Schritten wird das Modell für das Monitoring vorbereitet. Die Daten für das Monitoring können aus unterschiedlichen Quellen stammen, beispielsweise aus einem Datawarehouse.

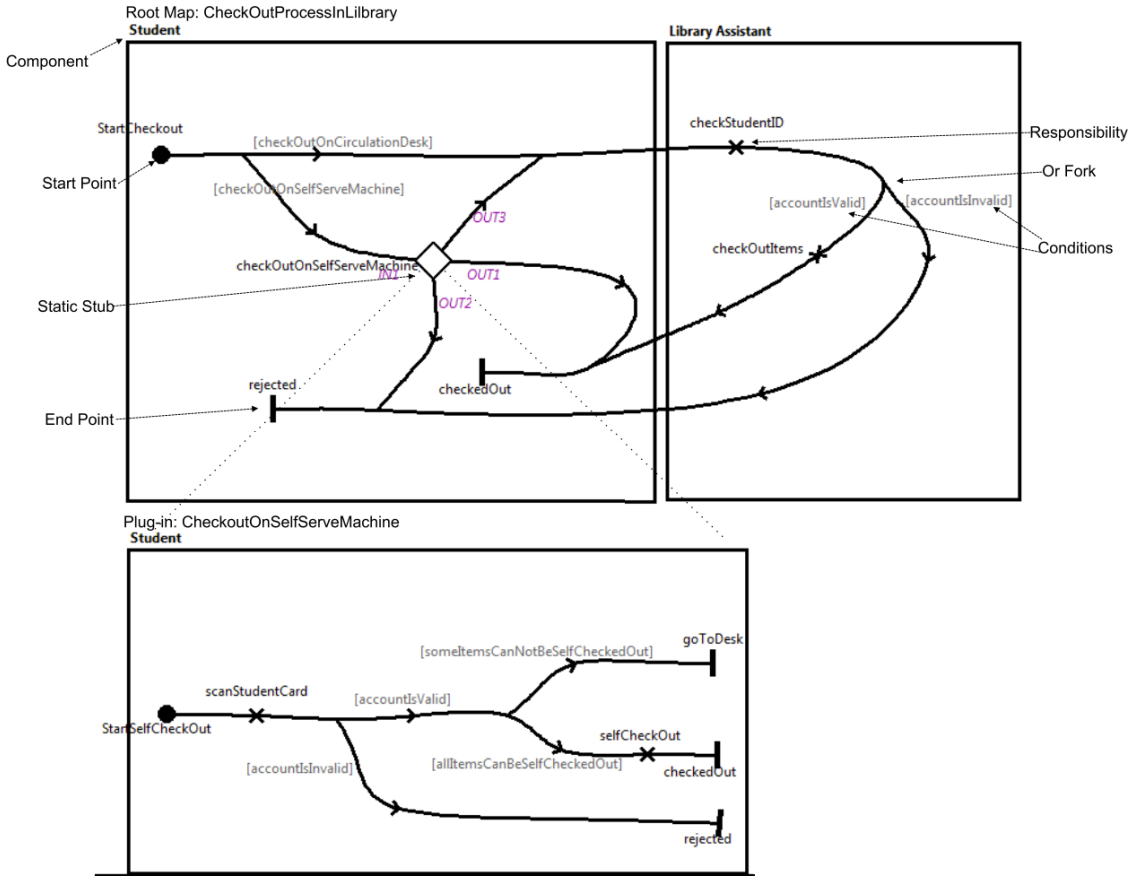

Abbildung 2.2.: Beispiel eines Prozesses in URN. Abbildung entnommen aus [Che07].

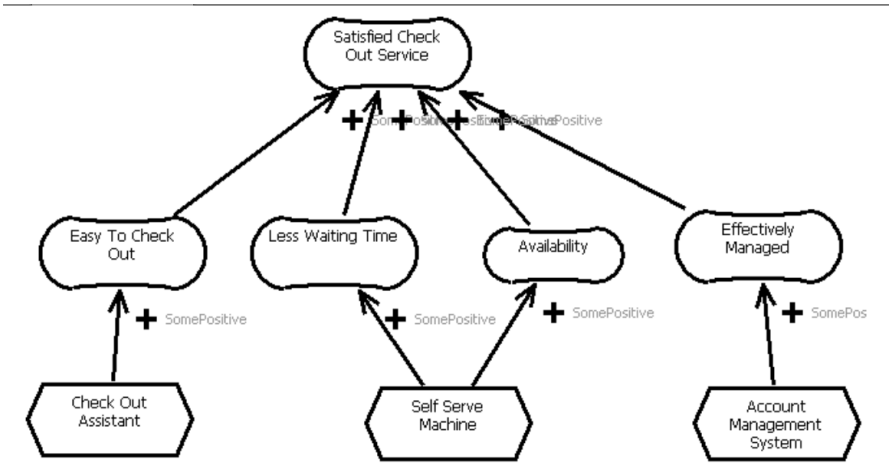

Abbildung 2.3.: Goal-Modell zum Prozess in Abbildung 2.2. Abbildung entnommen aus [Che07].

Das Konzept wurde prototypisch mithilfe des Eclipse Modeling Frameworks [Ecl] und des Editors jUCMNav [Juc] implementiert und getestet.

2.1.3. Business Event Modeling Notation

In [DGB07] wird eine grafische Notation namens Business Event Modeling Notation (BEMN) vorgestellt. Mit dieser Sprache ist die Modellierung von komplexen Events möglich, wodurch CEP-Engines besser in die Welt der Prozessmodelle integrierbar werden sollen. Insbesondere soll die Notwendigkeit von textuellen Sprachen für die CEP-Engines durchbrochen werden.

Die Modellierung der Events kann dabei getrennt vom Prozessmodell, aber auch direkt im Prozessmodell stattfinden. Durch die teilweise Verwendung von in BPMN bereits vorhandenen Elementen fügt sich die Sprache nahtlos in BPMN-Modelle ein. Allerdings betrachtet dieser Ansatz nicht die unternehmerische Sicht auf Prozesse. Daher werden in diesem Ansatz weder KPIs modelliert, noch existiert eine Erweiterung für das Monitoring der Events.

2.1.4. aPro

Ein weiterer Ansatz zur Modellierung von Prozesskennwerten wird in [KK12b] und [KK12a] vorgestellt. Es handelt sich um eine modulare Architektur für die Prozessoptimierung. Die Architektur orientiert sich am Livecycle eines Prozessmodells und deckt alle Schritte der Prozessanpassung ab. Abbildung 2.5 zeigt eine Übersicht über die Architektur.

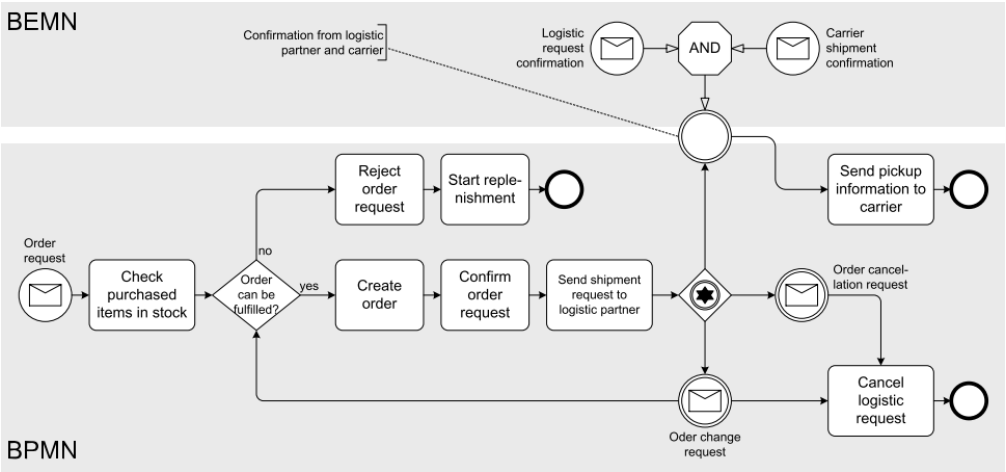

Abbildung 2.4.: BPMN-Prozess angereichert mit Elementen der BEMN. Abbildung entnommen aus [DGB07].

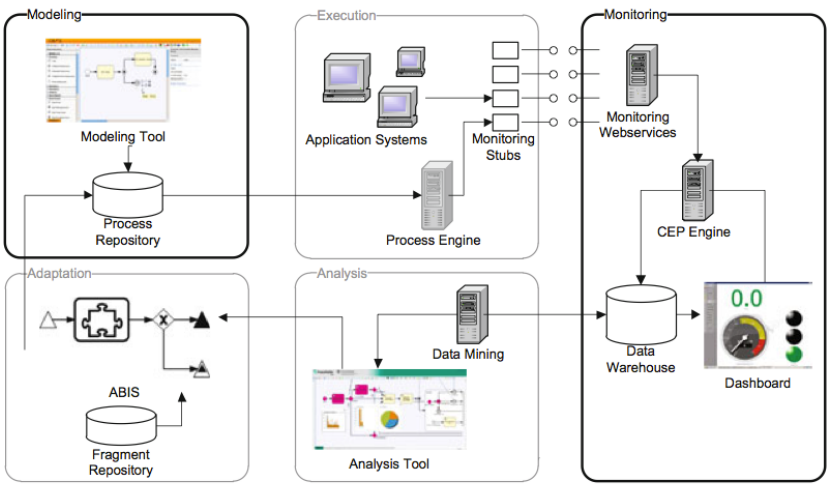

Abbildung 2.5.: aPro-Architektur. Abbildung entnommen aus [KK12b].

Ein Teil dieser Architektur ist eine Erweiterung zu BPMN, mit der Messpunkte und darauf basierend KPIs definiert werden können. Zur Speicherung kommt ein XML-Format namens ProGoalML zum Einsatz. Einen beispielhaften Prozess, welcher mit Messpunkten und KPIs angereichert wurde, zeigt Abbildung 2.6.

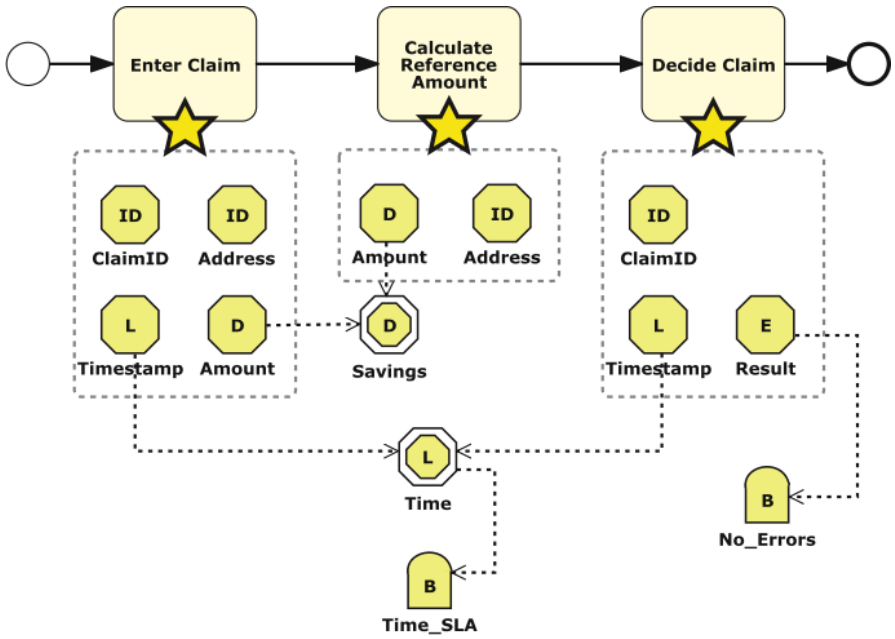

Abbildung 2.6.: BPMN-Prozess angereichert mit Elementen aus ProGoalML. Abbildung entnommen aus [KK12b].

Ein Prototyp zur automatisierten Umwandlung der ProGoalML-Datei in ein XML-Format namens VisML [Kin12] mit der Definition eines Dashboards zur Anzeige der KPIs wurde ebenfalls implementiert. In der VisML-Datei sind sowohl die Definitionen der Datenquellen enthalten als auch die Definitionen der Charts zur Anzeige der Daten. Neben der VisML-Datei werden auch noch die nötigen Abfragen für die CEP-Engine erstellt, welche die Daten für das Dashboard liefert. Die Zusammenhänge zwischen den einzelnen erstellten und generierten Artefakten zeigt Abbildung 2.7. Die Abbildung zeigt ebenfalls die zentrale Rolle der ProGoalML-Datei, aus welcher alle anderen Artefakte für das Monitoring generiert werden.

Die VisML-Datei kann dann in einem Dashboard dargestellt werden. Hierzu wurde ebenfalls ein Prototyp implementiert, welcher die Charts mithilfe der Google-Charts-API [Goo] oder mit eigens entwickelten Flash-Charts anzeigt. Das Dashboard frischt seine Daten in regelmäßigen Zeitabständen auf und ermöglicht so ein Monitoring des Prozesses in quasi-Echtzeit.

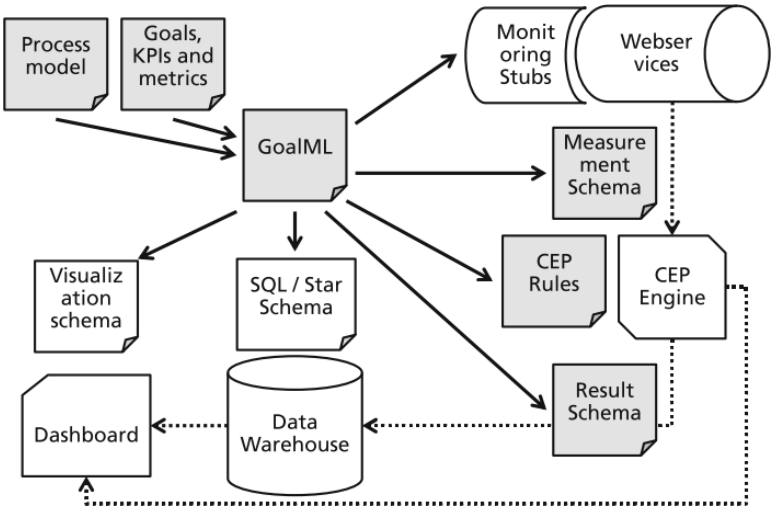

Abbildung 2.7.: Zusammenhang zwischen verschiedenen Artefakten der aPro-Architektur. Abbildung entnommen aus [KK12b].

2.1.5. KAOS

Ein auf der Linear Temporal Logic basierender Ansatz zur Modellierung von Zielen wird in [KG06] vorgestellt. Die Sprache definiert grafische Elemente zur Verkettung von Zielen (siehe Abbildung 2.9). Die Definition einzelner Zeile geschieht mithilfe der Linear Temporal Logik in einem eigenen, vom Prozessmodell getrennten, Modell. Auch ist dieser Ansatz nicht auf eine bestimmte Prozessmodellierungssprache beschränkt, sondern ist allgemein gehalten.

Das Ziel dieses Ansatzes ist die formale Validierung von Prozessmodellen und deren Anpassung an geänderte Ziele. Entsprechend ist ein Monitoring mit Daten aus der Prozessausführung nicht vorgesehen.

2.1.6. PPI-Ontologie

In [RORRC10] wird eine Ontologie zur Beschreibung von Process Performance Indicators (PPI) vorgestellt. Mithilfe dieser Ontologie können unterschiedliche Kennwerte beschrieben werden. Es sind einfache und komplexe Kennwerte vorgesehen sowie aggregierte Kennwerte mehrerer Prozessinstanzen. Damit weißt der Ansatz einige Ähnlichkeiten zur Kennwertmodellierungssprache von aPro auf. Allerdings fehlt diesem Ansatz zum Zeitpunkt der Arbeit noch eine grafische Sprache zur Definition der Prozesskennwerte.

Für die Zukunft sehen die Autoren noch einige Erweiterungen vor. So soll unter anderem die fehlende grafische Sprache definiert werden. Diese soll dann auf Basis der Editors Oryx [Tsc07] implementiert werden.

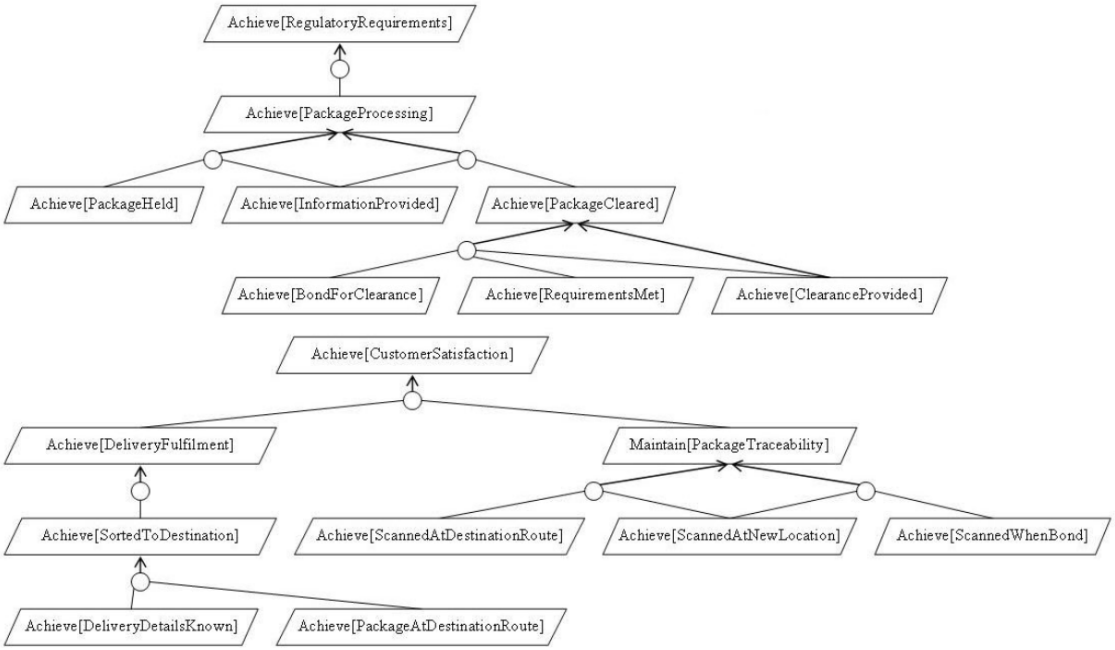

Abbildung 2.8.: Zielmodell in KAOS. Abbildung entnommen aus [KG06].

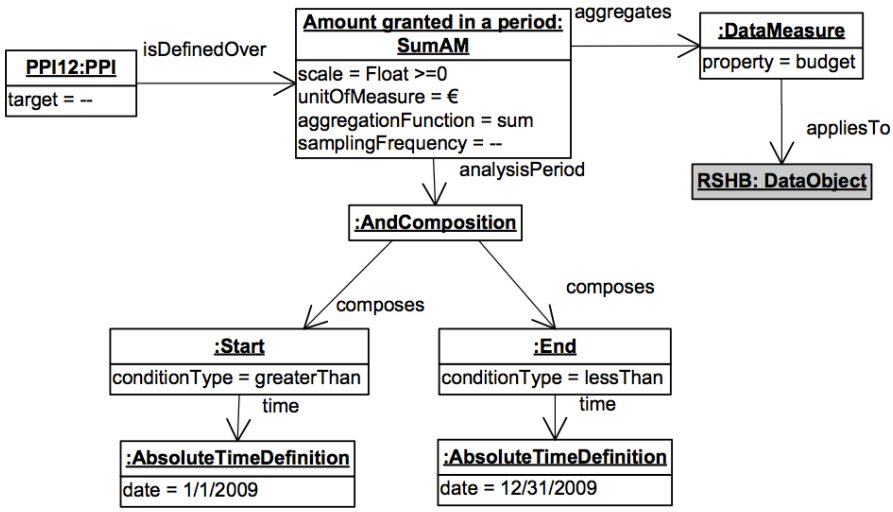

Abbildung 2.9.: Prozesskennwert in der Ontologie von [RORRC10]. Abbildung entnommen aus [RORRC10]

2.2. Vorhandene Sichtenmodellierungkonzepte

Im Folgenden werden einige Sichtkonzepte beleuchtet. Diese Konzepte beziehen sich meistens nur auf Prozessmodelle. Auf Prozesskennwerte bezieht sich nur einer der Ansätze. Eine freie Definierbarkeit der Sichten ist ebenfalls nicht in allen Ansätzen vorgesehen. Dennoch sollen auch diese beleuchtet werden um eventuell Ansätze zu identifizieren, welche sich auf Prozesskennwerte übertragen lassen.

2.2.1. Process Viewing Patterns

Ein sehr mächtiges Konzept für Sichten auf Prozessmodelle wird in [SLS10] vorgeschlagen. Die unterschiedlichen Sichten werden dabei durch Transformationen erreicht. Diese Transformationen können auch verkettet werden, wobei die Ausgabe einer Transformation als Eingabe für die nächste dient. Es werden unterschiedliche Typen von Transformationen, sogenannte Patterns, vorgeschlagen, die unterschiedlichen Zwecken dienen. So werden unter anderem Patterns zur Vereinfachung von Prozessen, zur Anreicherung von Prozessen mit Informationen oder zur Anpassung der Darstellung von Prozessen vorgeschlagen.

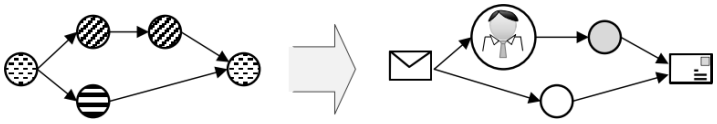

Abbildung 2.10.: Beispiel für die Anwendung des Appereance-Patterns auf ein Prozessmodell. Abbildung entnommen aus [SLS10].

Einige der Patterns bieten sich auch zur Unterstützung des Monitorings an, um zum Beispiel durch Anpassung der Darstellung eines Prozesses weitere Informationen darzustellen, die in einem Schaubild auf einem Dashboard nur schwer darzustellen wären. Diese Möglichkeiten sollen jedoch hier nicht weiter betrachtet werden, da sie die Zielsetzung der Arbeit bei weitem überschreiten. Im Bezug auf diese Arbeit sind insbesondere die Patterns zur Vereinfachung von Prozessen interessant.

Die Patterns sind allgemein gehalten und beziehen sich nicht auf konkrete Modellierungssprachen. Es wird jedoch als konkretes Beispiel die Anwendung auf BPEL betrachtet. Eine Anwendung der Patterns auf BPMN wäre ebenfalls möglich. Allerdings ist dieser Ansatz nur auf Prozessmodelle bezogen und betrachtet nicht das Monitoring. Entsprechend betrachtet er keine Prozesskennwerte.

2.2.2. Abstract Views on BPEL Processes

[Str09] stellt ein Konzept zur Erstellung von Sichten auf BPEL-Prozesse vor. Das vorrangige Ziel ist dabei das Verbergen von Details bei Prozessen mit Schnittstellen nach außen. So sollen zum Beispiel bei einem Reisebuchungsprozess für die externen Beteiligten, das Hotel und die Fluglinie, jeweils nur die für ihre Buchung relevanten Aktivitäten sichtbar sein.

Das Sichtkonzept bedient sich bei einigen Ideen der Model Driven Architecture (MDA), um die einzelnen Aktivitäten zu annotieren und dann den Prozess in zwei Schritten in die gewünschten Sichten zu transformieren. Wie beim vorherigen Ansatz wird hierbei nicht das Monitoring von Prozesskennwerten betrachtet, die Grundidee ist jedoch auch auf Kennwerte anwendbar.

2.2.3. BPEL Monitoring mit View-Slidern

In [Lat10] wurde ein Konzept zur Visualisierung einiger Prozesskennwerte direkt am Prozessmodell entwickelt. Eine Besonderheit dieses Konzepts ist die Kombination des Monitorings mit Sichten auf das Prozessmodell. Durch das Verschieben eines Schiebereglers können verschiedene Typen von Aktivitäten schrittweiße aus dem Modell entfernt werden. Durch einen zweiten Schieberegler kann die Darstellung der überwachten Daten schrittweiße eingeschränkt werden. Dadurch ist eine höhere Übersichtlichkeit bei großen Prozessmodellen gegeben.

Abbildung 2.11.: Task in zwei unterschiedlich detaillierten Sichten. Abbildung entnommen aus [Lat10].

Der Ansatz bezieht seine Daten von der ausführenden Workflowengine, kann also keine Prozesse betrachten, welche nicht auf einer Workflowengine ausgeführt werden. Weiterhin können nur Daten einer Instanz eines Prozesses überwacht werden. Eine Aggregation von Werten mehrerer Ausführungen ist nicht vorgesehen. Dadurch können viele relevante Kennwerte nicht abgebildet werden.

2.2.4. PISA

Ein weiterer Ansatz für Sichten im Monitoring wird in [MR00] vorgestellt. In diesem Ansatz wird das Monitoring in drei Sichten zerlegt, die Prozess-Sicht, die Ressourcen-Sicht und die Objekt-Sicht. Jede der Sichten ist dabei darauf zugeschnitten, unterschiedliche Optimierungen zu unterstützen und zu erleichtern.

Die Daten werden hauptsächlich aus dem Audit-Trail extrahiert. Auch dieser Ansatz geht also von der Ausführung auf einer Workflowengine aus. Allerdings legt sich der Ansatz nicht auf eine bestimmte Engine fest, mit der Folge, dass je nach Engine der Informationsgehalt des Audit-Trails stark unterschiedlich sein kann. Eine Beschränkung auf eine bestimmte Prozessmodellierungssprache existiert ebenfalls nicht.

Der Ansatz wurde in einem Prototyp namens PISA implementiert. PISA besteht aus mehreren Komponenten, unter anderem einem Server, welcher die Aufbereitung der Daten koordiniert und sie den Clients zur Anzeige zur Verfügung stellt. Die Daten für das Monitoring werden über ein Interface, das Data Source Interface, an welches Data Source Adapter angebunden werden, bezogen.

Im Gegensatz zu den anderen Ansätzen implementiert PISA ein Rechtesystem, mit welchem sich unter anderem die Sichtbarkeit von Auswertungen für den Benutzer eingrenzen lässt. Beispielsweiße kann so dafür gesorgt werden, dass Benutzer nur Prozesse ihrer eigenen Abteilung sehen können.

3. Prüfung vorhandener Ansätze

In diesem Kapitel werden die gefundenen verwandten Arbeiten aus Kapitel 2 auf die Erfüllung der Anforderungen geprüft. Das Ziel ist zum einen die Auswahl einer geeigneten Kennwertmodellierungssprache als Grundlage für die Arbeit. Zum anderen sollen mögliche Ansätze für Sichten auf ihre Anwendbarkeit für das zu lösende Problem untersucht werden.

3.1. Prüfung der Konzepte für Kennwertmodellierung und Monitoring

Es wurden mehrere Ansätze zur Modellierung von Prozesskennwerten vorgestellt. Diese Ansätze sollen in diesem Kapitel auf ihre Verträglichkeit mit den Anforderungen geprüft werden. Der am besten geeignete Ansatz wird in dieser Arbeit als Grundlage für die Sichtenmodellierung gewählt werden.

3.1.1. Run-Time Monitor specification Language

Der Ansatz von [Bar+06] ermöglicht das Monitoring von Instanzkennwerten und Klassenkennwerten von BPEL-Prozessen. Dazu existiert eine automatische Generierung von Javacode für die Monitore der einzelnen Kennwerte. Allerdings setzt dieser Ansatz eine Ausführung des Prozesses auf einer Workflowengine voraus.

Die Definition der Kennwerte geschieht nicht in Form von KPIs, sondern in Form von Events. Weiterhin besitzt dieser Ansatz keine grafische Sprache. Dies ist kein Ausschlussgrund, ist im Kontext dieser Arbeit jedoch negativ zu bewerten. Dazu kommt, dass die Definition der Events getrennt vom eigentlichen Prozess geschieht. Dies widerspricht ebenso der geforderten nahtlosen Integration der Kennwertmodellierungssprache in die Prozessmodellierungssprache.

3.1.2. Dynamic Monitoring

Der Ansatz von [BG05] modelliert Events für BPEL-Prozesse. Für diesen Ansatz gelten nahezu dieselben Einschränkungen wie für den vorherigen Ansatz. Auch dieser Ansatz modelliert Kennwerte getrennt vom Prozessmodell und betrachtet keine KPIs. Ebenso setzt er die Ausführung des Prozesses auf einer Workflowengine voraus.

3.1.3. Ansatz von Spanoudakis und Mahbub

Wie bereits die vorherigen Ansätze ist dieser Ansatz kaum für diese Arbeit geeignet. Er definiert keine grafische Sprache und bezieht sich auch nicht konkret auf Prozesskennwerte. Infolgedessen bietet der Ansatz auch kein Dashboard zur Betrachtung der gemessenen Werte. Es müsste also viel Vorarbeit geleistet werden um die grundlegenden Anforderderungen zu erfüllen.

3.1.4. Ansatz auf Basis der User Requirement Notation

Der auf der URN basierende Ansatz von [Che07] betrachtet als erster Ansatz nicht nur Events, sondern bezieht sich explizit auf KPIs. Auch bietet er eine grafische Sprache zur Modellierung der Kennwerte.

Da dieser Ansatz jedoch auf der, im BPM-Umfeld wenig verbreiteten, URN basiert, widerspricht er den Anforderungen. Da Teilmodelle der URN, welche so nicht in BPMN vorhanden sind, erweitert werden, ist eine einfache Portierbarkeit des Ansatzes auf BPMN ebenfalls nicht gegeben. Der Ansatz eignet sich daher nicht als Grundlage für diese Arbeit.

3.1.5. Business Event Modeling Notation

Der in [DGB07] vorgestellt Ansatz bietet eine grafische Modellierungssprache für komplexe Events auf Basis von BPMN. Die Integration in BPMN ist sehr gut und entspricht vollständig den Anforderungen.

Allerdings werden dabei keine Prozesskennwerte betrachtet, was die Anwendung für nicht technische Benutzer stark erschwert. Weiterhin ist das Ziel dieses Ansatzes die bessere Integration von Events in BPMN. Daher setzt dieser Ansatz eine Ausführung auf einer Workflowengine voraus und sieht keine Generierung eines Dashboards vor. Dieser Ansatz ist daher nur mit Einschränkungen als Grundlage für diese Arbeit geeignet.

3.1.6. aPro

Der in [KK12b] und [KK12a] vorgestellte Ansatz ermöglicht die Modellierung von Prozesskennwerten direkt in einem BPMN-Diagramm. Dazu wird BPMN um grafische Elemente für Messpunkte und Prozesskennwerte erweitert. Neben Kennwerten zu einer Prozessinstanz sind auch instanzübergreifende Messungen möglich. Die Anforderungen an die Sprache sind damit vollständig erfüllt.

Weiterhin ist keine Workflowengine nötig und es können beliebige Datenquellen genutzt werden. Aus dem Diagramm kann nach der Modellierung automatisiert ein Dashboard zur Überwachung der Kennwerte erstellt werden. Damit sind auch alle weiteren Anforderungen erfüllt.

Weitere Pluspunkte dieses Ansatzes sind:

- Prototyp auf Basis eines freien Editors
- Modularer Aufbau vereinfacht Erweiterungen
- Automatische Generierung aller notwendigen Artefakte, einschließlich CEP-Regeln und Dashboard, aus einer Datei
- Ansatz wird aktiv weiterentwickelt

3.1.7. KAOS

Der in [KG06] vorgestellte Ansatz ermöglicht die Definition von Zielen mit einer von der Prozessmodellierungssprache unabhängigen Sprache. Die Zielsetzung dieses Ansatzes ist jedoch die formale Validierung von Prozessen. Entsprechend beinhaltet er keine der für die Arbeit relevanten Funktionen für das Monitoring. Es existiert also keine automatische Dashboardgenerierung. Der Ansatz ist damit nicht als Grundlage dieser Arbeit geeignet.

3.1.8. PPI-Ontologie

Der Ansatz aus [RORRC10] ist in seinen Möglichkeiten dem Ansatz aPro sehr ähnlich. Er beinhaltet ähnliche Messungen und soll in Zukunft auch eine grafische Sprache und einen Prototyp auf Basis von Oryx bekommen. Allerdings wurde die angekündigte Sprache zum Zeitpunkt der Arbeit noch nicht veröffentlicht. Daher eignet sich der Ansatz zum jetztigen Zeitpunkt nicht als Basis für diese Arbeit.

3.2. Prüfung der Sichtenmodellierungskonzepte

Neben Ansätzen zur Modellierung von Kennwerten wurden auch Ansätze zur Modellierung von Sichten gesucht. Im Folgenden sollen diese Ansätze daraufhin untersucht werden, ob sie für das in dieser Arbeit zu erarbeitende Sichtenkonzept als Grundlage geeignet sind.

3.2.1. Process viewing patterns

Der in [SLS10] vorgestellte Ansatz erlaubt die Definition von Sichten auf Prozesse mithilfe von Patterns. Die Patterns sind allgemein gehalten und beziehen sich nicht auf konkrete Modellierungssprachen. Es wird jedoch als konkretes Beispiel die Anwendung auf BPEL betrachtet. Eine Anwendung der Patterns auf BPMN wäre ebenfalls möglich.

Allerdings ist die Herangehensweise über Patterns weitaus mächtiger als es die zu Beginn definierten Anforderungen verlangen. Die hohe Komplexität des Ansatzes ist für den Anwendungszweck dieser Arbeit jedoch nicht gerechtfertigt. Es würde dadurch der Anforderung widersprochen, dass auch Benutzer ohne technischen Hintergrund das entwickelte Konzept nutzen können müssen. Der Ansatz eignet sich daher nicht als Grundlage des Sichtenkonzepts.

3.2.2. Abstract Views on BPEL Processes

In [Str09] wird ein Konzept zur Erstellung von Sichten auf BPEL-Prozesse vorgestellt. Wie beim vorherigen Ansatz wird hierbei nicht das Monitoring von Prozesskennwerten betrachtet. Die Arbeit und insbesondere der begleitend entwickelte Prototyp kann also nicht unverändert als Grundlage dieser Arbeit verwendet werden.

Die naheliegende Idee des Taggings der einzelnen Elemente lässt sich jedoch grundsätzlich auch auf andere Elemente anwenden. Daher wird diese Idee die Grundlage dieser Arbeit sein. Um die Anforderungen an diese Arbeit zu erfüllen, muss das Konzept jedoch noch erweitert werden.

3.2.3. BPEL Monitoring mit View-Slidern

In [Lat10] wurde ein Konzept zur Visualisierung einiger Prozesskennwerte direkt am Prozessmodell entwickelt. Dieser Ansatz weist jedoch auch einige Nachteile auf. Die Implementierung ermöglicht nur das Monitoring von BPEL-Prozessen, was den Anforderungen an die in dieser Arbeit zu entwickelnde Lösung nicht gerecht wird. Eine Unterstützung von BPMN wäre jedoch mit einigem Aufwand implementierbar. Ein weiteres Problem ist die nötige Anbindung an eine Workflowengine. Die Anforderungen an diese Arbeit verlangen jedoch

eine Loslösung von der Ausführung des Prozesses auf einer Workflowengine. Auch hier wäre mithilfe von selbst implementierten Adaptern sicherlich eine Anpassung an die hier geltenden Anforderungen möglich. Ein weiteres sehr großes Problem mit diesem Ansatz ist jedoch das Fehlen jeglicher Diagramme zur Anzeige der vom Monitoring überwachten Daten. Dadurch ist die Menge an darstellbaren Kennwerten aufgrund des Platzes im Prozessmodell stark beschränkt.

Des Weiteren unterstützt das System nur die Einschränkung der dargestellten Daten anhand des vordefinierten Schiebereglers. Die Definition beliebig vieler Rollen ist in diesem Ansatz nicht vorgesehen, da die Einschränkung der Daten lediglich auf dem Typ der angezeigten Daten basiert und nicht beliebig definiert werden kann. Eine weitere Einschränkung ist, dass immer nur Daten zu einer einzelnen Instanz eines Prozesses angezeigt werden.

3.2.4. PISA

Der in [MR00] vorgestellte Ansatz sieht zwar mehrere Sichten vor, es handelt sich dabei jedoch nicht um Sichten zu verschiedenen Rollen im Sinne dieser Arbeit. Zwar sind auch die Sichten dieses Ansatzes an unterschiedliche Rollen und Aufgaben angepasst, die Erstellung weiterer Sichten für beliebige Rollen ist jedoch nicht möglich. Daher mangelt es ebenso an einer grafischen Sprache zur Definition der Sichten.

Im Gegensatz zu den anderen Ansätzen implementiert PISA zwar auch ein Rechtesystem, mit welchem sich unter anderem die Sichtbarkeit von Auswertungen für den Benutzer eingrenzen lässt. Beispielsweiße kann so dafür gesorgt werden, dass Benutzer nur Prozesse ihrer eigenen Abteilung sehen können. Wie jedoch schon die vorherigen Ansätze werden auch hier keine Prozesskennwerte modelliert. Dadurch eignet sich dieser Ansatz trotz seiner Flexibilität und seines Rechtekonzepts nicht als Grundlage für diese Arbeit.

3.3. Erläuterung der Auswahlkriterien

Die detaillierte Bewertung und die Auswahl eines Ansatzes geschieht aufgrund der im Folgenden beschriebenen Kriterien.

3.3.1. Konzepte für Kennwertmodellierung und Monitoring

Die folgenden Kriterien sind die Grundlage zur Auswahl eines Ansatzes für die Kennwertmodellierung.

Prozesskennwerte modellierbar Es können Prozesskennwerte als solche modelliert werden. Eine Modellierung von Events reicht nicht aus.

Modellierung grafisch Die Modellierung der Prozesskennwerte kann mit einer grafischen Sprache erfolgen.

Modellierung BPMN-Basiert Als Prozessmodellierungssprache kommt BPMN zum Einsatz. Dieses Kriterium ist nicht zwingend.

Autom. Dashboardgenerierung aus KPIs Aus den Prozesskennwertmodell kann automatisiert ein Dashboard zur Anzeige der Kennwerte generiert werden.

Sichtenkonzept für Monitoring Es existiert ein Sichtenkonzept für die Kennwerte, welches beschreibt, welche Rolle im Unternehmen welche Kennwerte betrachten darf.

Instanzübergreifende Messungen Der Ansatz ermöglicht das Modellieren von instanzübergreifenden Messungen um zum Beispiel Durchschnittswerte über mehrere Instanzen zu überwachen.

Keine Anbindung an Workflowengine nötig Der Ansatz setzt nicht die Ausführung des Prozesses auf einem Workflowengine voraus.

Beliebige Datenquellen Der Ansatz ermöglich die Integration beliebiger Datenquellen um so beliebige Werte überwachen zu können.

3.3.2. Sichtenmodellierungskonzepte

Die folgenden Kriterien sind die Grundlage für die Bewertung der Ansätze zur Sichtenmodellierung.

Sicht auf Prozessmodell Der Ansatz ermöglicht das Modellieren von Sichten auf Prozessmodelle.

Sicht auf Monitoring Der Ansatz ermöglicht das Modellieren von Sichten auf Kennwertmodelle und Dashboards.

Auf Prozesskennwerte anwendbar Der Ansatz ist auf Prozesskennwerte anwendbar.

Freie Definition der Rollen Es ist möglich, beliebige Rollen zu definieren und für diese Rollen beliebige Elemente als sichtbar und unsichtbar definieren.

Entspricht Sichtenbegriff der Arbeit Der Sichtenbegriff des Ansatzes entspricht dem auf Rollen basierenden Sichtenbegriff dieser Arbeit.

3.4. Auswahl eines Ansatzes

In den vorherigen Abschnitten wurde eine Reihe von Ansätzen auf die Erfüllung der Anforderungen geprüft. Viele der Ansätze erfüllen einen Teil der Anforderungen, widersprechen aber anderen Anforderungen. Andere Ansätze bringen wertvolle Konzepte für die Implementierung für Sichten, jedoch löst keiner dieser Ansätze das dieser Arbeit zugrunde liegende Problem vollständig.

Tabelle 3.1 zeigt einen Vergleich der einzelnen Kennwertmodellierungskonzepte im Hinblick auf die Erfüllung der Anforderungen. Der Vergleich zeigt, dass keiner der ersten drei Ansätze, basierend auf BPEL, die Modellierung von Kennwerten vorsieht. Auch fällt auf, dass keiner der Ansätze auf eine Anbindung an eine Workflowengine verzichten kann. Damit sind diese drei Ansätze nicht mit den Anforderungen kompatibel.

Betrachtet man bei den verbleibenden Ansätzen die Anforderungen bezüglich der Kennwertmodellierungssprache, so fällt auf, dass nur zwei der Ansätze, die Modellierung auf Basis der URN und aPro, die Anforderungen erfüllen. Erwähnenswert ist hier noch die PPI-Ontologie aus [RORRC10]. Dieser Ansatz erfüllt zum Zeitpunkt der Arbeit die Anforderungen nicht, jedoch ist eine Erweiterung des Ansatzes in diese Richtung von den Autoren vorgesehen.

Von den zwei Ansätzen, die diese Anforderungen erfüllen, erfüllt nur einer, die Architektur aPro, auch die restlichen Anforderungen ausreichend. Dazu gehören insbesondere die Unabhängigkeit von der Ausführung des Prozesses auf einer Workflowengine und Instanzübergreifende Messungen. Die Entscheidung fällt also zugunsten von aPro als Grundlage für diese Arbeit aus.

Neben den Kennwertmodellierungskonzepten wurden auch unterschiedliche Sichtenkonzepte in Zusammenhang mit Prozessen und Prozesskennwerten betrachtet. Tabelle 3.2 zeigt einen Vergleich der einzelnen Konzepte.

Erwähnenswert sind hierbei besonders zwei Ansätze. Das Konzept der View-Slider betrachtet als einziges Konzept Monitoring und Prozessmodellierung. Allerdings sind die einzelnen Sichten in diesem Konzept nicht rollenbasiert und nicht beliebig definierbar. Daher eignet es sich nicht als Grundlage für das Sichtenkonzept dieser Arbeit.

Das Konzept der Abstract Views hingegen entspricht als einziges dem Sichtenbegriff dieser Arbeit und ermöglicht die Definition beliebiger Rollen. Allerdings bezieht sich das Konzept lediglich auf Prozessmodelle und ist nicht ohne Änderungen auf Prozesskennwerte anwendbar. Allerdings eignet sich dieses Konzept gut als Grundlage für das hier zu entwickelnde Sichtenkonzept. Daher wird die naheliegende Idee der annotation von Elementen mit Rollen als Grundlage für das hier zu entwickelnde Sichtenkonzept dienen. Jedoch werden einige Erweiterungen des Konzepts nötig sein um die Anforderungen an das Konzept zu erfüllen.

Kriterium	RTML	Dynamic monitoring	Spanoudakis und Mahbub	URN	BEMN	aPro	KAOS	PPI-Ontologie	PISA
Prozesskennwerte modellierbar	-	-	-	+	-	+	-*	+	-
Modellierung grafisch	-	-	-	+	-	+	-	-	-
Modellierung BPMN-Basiert	-	-	-	-	+	+	+-**	-	-
Autom. Dashboardgenerierung aus KPIs	+	-	-	+-	-	+	-	-	-
Sichtenkonzept für Monitoring	-	-	-	-	-	-	-	-	+
Instanzübergreifende Messungen	+	-	-	-	-	+	-	+	+
Keine Anbindung an Workflowengine nötig	-	-	+	+	-	+	+	+	-***
Beliebige Datenquellen	-	+	-	+	+	+	-	+	+-***

Legende:

+ : Kriterium erfüllt

- : Kriterium nicht erfüllt

+- : Kriterium teilweise erfüllt

* Keine Prozesskennwerte im Sinne der Arbeit
** Beliebige Sprachen zur Prozessmodellmodelierung möglich.
*** Erweiterung vorgesehen.

Tabelle 3.1.: Vergleich der Prozesskennwertmodellierungskonzepte

Kriterium	Process viewing patterns	Abstract Views	View-Slider	PISA
Sicht auf Prozessmodell	+	+	+	-
Sicht auf Monitoring	-	-	+	+
Auf Prozesskennwerte anwendbar	+	-	-	-
Freie Definition der Rollen	-	+	-	-
Entspricht Sichtenbegriff der Arbeit	-	+	-	-

Legende:

+ : Kriterium erfüllt

- : Kriterium nicht erfüllt

+- : Kriterium teilweise erfüllt

Tabelle 3.2.: Vergleich der Sichtenkonzepte

4. Sichtenkonzept

In diesem Kapitel soll basierend auf den Ergebnissen der Analyse ein Sichtenkonzept für KPIs entworfen werden. Da in der Analyse kein Ansatz gefunden wurde, der alle Anforderungen abdeckt, wird ein neues Sichtenkonzept auf Basis der Kennwertmodellierungssprache ProGoalML aus der aPro-Architektur entworfen werden. Nichtsdestotrotz werden einige Ideen aus der Analyse Eingang in das hier entworfene Konzept finden. Eine grobe Umsetzung des Konzepts in Form eines Prototyps wird im nächsten Kapitel beschrieben.

4.1. Definition wichtiger Begriffe

Im Folgenden werden einige wichtige Begriffe erläutert und ihre Bedeutung in dieser Arbeit eingegrenzt.

KPI KPI steht für „Key Performance Indicator“ und bezeichnet einen Prozesskennwert, beispielsweise die Ausführungsdauer eines Prozesses. Prozesskennwerte können sich entweder auf eine konkrete Instanz eines Prozesses beziehen oder auf eine Menge von Prozessen. In letzterem Fall sprechen wir von einer aggregierten KPI. [Par10]

Ziel Ein Ziel ist eine Eigenschaft einer Prozessinstanz oder auch einer Menge von Instanzen, die entweder erfüllt ist oder nicht. Es ist äquivalent zu einem Goal in ProGoalML.

Monitoring Als Monitoring wird in dieser Arbeit die Überwachung von Kennzahlen eines Prozesses bezeichnet. Ob die Daten dabei in Echtzeit aus einem Workflowmangementsystem übernommen werden oder aus Datensammlungen, wie zum Beispiel dem Audit-Trail oder einem Warehouse, stammen, ist unerheblich. Die überwachten Daten werden dabei dem Benutzer in geeigneter Art in einem Dashboard angezeigt.

Sicht Eine Sicht im Kontext der Arbeit meint eine Eingrenzung der Sichtbarkeit auf einige Teile eines Ganzen. Im Gegensatz zu Sichten in SQL soll dabei das Zusammensetzen eines neuen Ganzen aus Teilen von mehreren Ganzen nicht möglich sein. Eine Sicht im Kontext der Arbeit ist also mit einem Rechtekonzept vergleichbar, welches die Sichtbarkeit einzelner Teile regelt.

Rolle Eine Rolle bezeichnet die Zuständigkeit für eine bestimmte Aufgabe oder ein Aufgabengebiet. Zur Abarbeitung der Aufgabe ist typischerweise nur eine Teilmenge des Dashboards relevant. Weiterhin kann es einer Rolle auch per Definition, zum Beispiel durch Gesetze, verboten sein, gewisse Daten einzusehen. Einer Rolle ist also eine Teilmenge des Dashboards zugeordnet, die für die Bearbeitung der Aufgabe notwendig ist. Jeder Rolle ist damit eine Sicht zugeordnet.

4.2. Formale Definition des Sichtenkonzepts

Um das Sichtenkonzept zu erstellen, müssen zunächst einige Konstrukte formal definiert werden. Die Konstrukte werden für das Sichtenkonzept von großer Bedeutung sein, und müssen deswegen genau definiert und abgegrenzt werden.

4.2.1. Rollen

Das Sichtenkonzept basiert auf Rollen. Eine Rolle entspricht dabei einer Sicht auf das aus den Kennwerten generierte Dashboard. Jeder Rolle werden diejenigen Elemente zugewiesen, welche für sie sichtbar sind. Die von unterschiedlichen Rollen einsehbaren Elemente eines Dashboards müssen dabei nicht zwingend disjunkt sein, sondern können für jede Rolle unabhängig von den restlichen Rollen frei definiert werden.
Damit bildet diese Definition sehr gut den intuitiven Rollenbegriff der realen Welt ab, in welcher Personen mit unterschiedlichen Rollen bei der Ausübung ihrer Rollen durchaus mitunter dieselben Ressourcen verwenden.
Eine Rolle hat folgende Attribute:

- Bezeichnung
- Beschreibung (optional)
- Eigenschaften (optional, beliebig viele)

Formale Definitionen:

Rollen Sei n ein Name (name), d eine Beschreibung (description) und $P_r \subseteq P$ eine Menge von Eigenschaften (properties). Dann ist eine Rolle (role) ein Tupel $r \in R = (n, d, P_r)$.

Dabei gilt: $\forall r_1, r_2 \in R : n_{r_1} = n_{r_2} \Rightarrow r_1 = r_2$

Eigenschaften Sei k ein Schlüssel (key) und v ein Wert (value). Dann ist eine Eigenschaft (property) ein Tupel $p \in P = (k, v)$.

Dabei gilt: $\forall r \in R \forall p_1, p_2 \in P_r : k_{p_1} = k_{p_2} \Rightarrow p_1 = p_2$

Rollenzuordnung Sei $e \in E$ ein Element und $R_e \subseteq R$ die Menge von Rollen dieses Elements. Dann ist eine Rollenzuordnung (role assignment) ein Tupel $a \in A = (e, R_e)$.

Dabei gilt: $\forall a_1, a_2 \in A : e_{a_1} = e_{a_2} \Rightarrow a_1 = a_2$

4.2.2. Benutzer

Ein Benutzer ist ein Inhaber einer Rolle. Ein Benutzer kann dabei mehrere Rollen haben und eine Rolle kann von mehreren Benutzern ausgeübt werden.
Ein Benutzer hat folgende Attribute:

- ID
- Name
- Rollen (optional, beliebig viele)
- Eigenschaften (optional, beliebig viele)

Formale Definition:

Benutzer Sei i ein Bezeichner (ID), n ein Name (name) und $R_u \subseteq R$ und $P_u \subseteq P$ eine Menge von Eigenschaften (properties). Dann ist ein Benutzer (user) ein Tupel $u \in U = (i, n, R_u, P_u)$.

Dabei gilt: $\forall u_1, u_2 \in U : i_{u_1} = i_{u_2} \Rightarrow u_1 = u_2$

Eigenschaften Sei k ein Schlüssel (key) und v ein Wert (value). Dann ist eine Eigenschaft (property) ein Tupel $p \in P = (k, v)$.

Dabei gilt: $\forall u \in U \forall p_1, p_2 \in P_u : k_{p_1} = k_{p_2} \Rightarrow p_1 = p_2$

4.2.3. Einstellungen

Um das Konzept flexibel zu halten, sieht das Konzept die Möglichkeit vor, eigene Einstellungen zu Sichtendefinitionen zu speichern. Einstellungen werden, ähnlich den Eigenschaften der Benutzer, in einem Key-Value Format gespeichert. Im Unterschied zu den Eigenschaften eines Benutzers gibt es hier jedoch zusätzlich die Möglichkeit, rollenspezifische Einstellungen zu speichern. In diesem Fall kommt zum Schlüssel noch die Rolle hinzu.

Formale Definition:

Einstellungen Sei k ein Schlüssel (key) und v ein Wert (value). Dann ist eine Einstellung (preference) ein Tupel $p \in P = (k, v)$.

Dabei gilt: $\forall p_1, p_2 \in P : k_{p_1} = k_{p_2} \Rightarrow p_1 = p_2$

Rollenspezifische Einstellungen Sei k ein Schlüssel (key), r eine Rolle (role) und v ein Wert (value). Dann ist eine rollenspezifische Einstellung (rolespecific preference) ein Tupel $p \in P = (k, r, v)$.

Dabei gilt: $\forall p_1, p_2 \in P : (k_{p_1} = k_{p_2} \wedge r_{p_1} = r_{p_2}) \Rightarrow p_1 = p_2$

4.3. XML-Format

Das XML-Format definiert zum einen die Rollen und die Benutzer des Dashboards, zum anderen die Sichten auf das Dashboard. Es dient vorrangig der editorneutralen Speicherung aller notwendigen Daten für die Generierung von Sichten. Das XML-Format besteht aus zwei Dateien. Die erste Datei enthält alle Benutzer und deren Rollen. Diese Datei kann, je nach Umfang, für mehrere Sichtdefinitionen oder gar für alle Sichtdefinitionen verwendet werden. Die zweite Datei enthält die eigentliche Sichtdefinition und bezieht sich auf Rollen aus der ersten Datei.

Listing 4.1 zeigt eine beispielhafte XML-Datei mit einigen Rollendefinitionen und Benutzern. Diese Datei ist von konkreten Sichtdefinitionen unabhängig und kann dadurch für beliebig viele Sichtdefinitionen verwendet werden. Insbesondere ist eine globale Datei mit Rollendefinitionen möglich, welche sämtliche Rollen und Benutzer eines Unternehmens enthält.

```
<userRoles>
  <roles>
    <role name="SHE-Manager">
      <description>Umweltfragen, Sicherheitsfragen,
         Gesundheitsfragen</description>
      <properties>
        <property key="customAlertQuery" value="*Environment*"/>
      </properties>
    </role>
    <role name="Tester">
      <description>Automobiltests</description>
    </role>
    <role name="Manager">
      <description/>
    </role>
  </roles>
  <users>
    <user id="1" name="Tine Tester">
```

```
    <roles>
      <role>Tester</role>
    </roles>
  </user>
  <user id="2" name="Markus Manager">
    <roles>
      <role>Manager</role>
    </roles>
  </user>
  <user id="3" name="Arne Allrounder">
    <roles>
      <role>Tester</role>
      <role>Manager</role>
    </roles>
  </user>
  <user id="4" name="Urs Umwelt">
    <roles>
      <role>SHE-Manager</role>
    </roles>
  </user>
 </users>
</userRoles>
```

Listing 4.1: Einfaches Beispiel einer Rollendefinition.

Listing 4.2 zeigt eine zu Listing 4.1 gehörende Sichtdefinition. Im „meta“ -Element der Datei wird dafür die entsprechende Rechtedefinition referenziert. Die eigentliche Sichtdefinition befindet sich im „elements“-Element der Datei.

```
<views>
 <meta>
  <lastmodified>2012-10-20T22:13:57Z</lastmodified>
 </meta>
 <preferences>
  <global>
    <preference key="propagateVisibility" value="0"/>
    <preference key="showParameterVisualizations" value="true"/>
    <preference key="generateMasterDashboard" value="false"/>
    <preference key="includeCustomAlertsInMasterDashboard"
       value="true"/>
```

```
    <preference key="includeCustomAlertsInRoleDashboards"
       value="false"/>
    <preference key="removeUnusedDatasets" value="true"/>
    <preference key="removeUnusedDatasources" value="true"/>
  </global>
  <roleSpecific/>
</preferences>
<elements>
  <element stencil="http://b3mn.org/stencilset/bpmn2.0#kpi"
     name="totalFuelConsuption">
    <role>SHE-Manager</role>
    <role>Tester</role>
  </element>
  <element stencil="http://b3mn.org/stencilset/bpmn2.0#kpi"
     name="totalWaterConsumption">
    <role>SHE-Manager</role>
    <role>Tester</role>
  </element>
  <element stencil="http://b3mn.org/stencilset/bpmn2.0#parameter"
     name="CreateReport___timestamp">
    <role>Tester</role>
    <role>Manager</role>
  </element>
</elements>
</views>
```

Listing 4.2: Einfaches Beispiel einer Sichtdefinition.

4.3.1. Benutzer- und Rollendefinitionssprache

Im Folgenden wird die XML-Sprache zur Definition von Benutzern und Rollen im Detail beschrieben. Eine grafische Übersicht gibt Abbildung 4.1. Das vollständige Schema in textueller Form findet sich in Anhang A.1.

Das Wurzelelement **userRoles** hat genau ein Attribut *version*. Dieses Attribut gibt die zugrunde liegende Schema-Version an und ist für das hier beschriebene Schema „1.0“. In diesem Element sind genau zwei Kindelemente erlaubt. Beide Kindelemente sind Pflichtelemente.

- Das Element **roles** enthält die Definition aller verfügbaren Rollen.
- Das Element **users** enthält die Benutzer und ihre Rollenzuordnung.

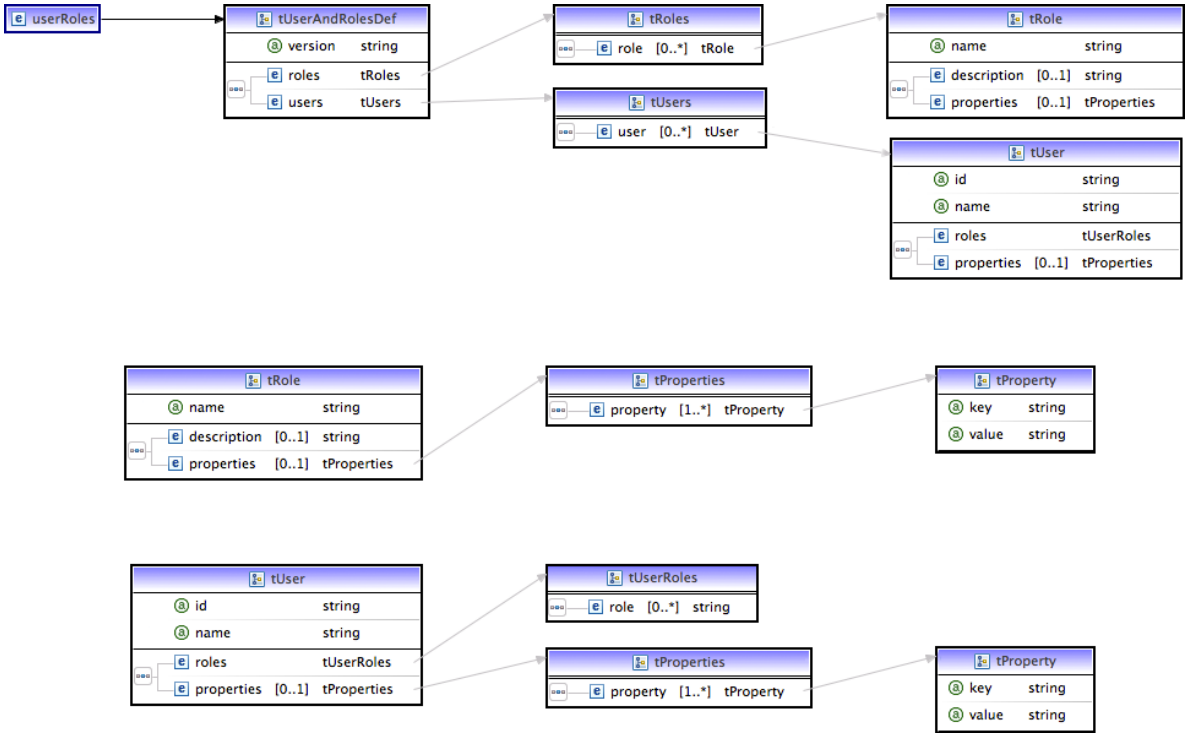

Abbildung 4.1.: Grafische Darstellung des XML Schemas der Benutzer- und Rollendefinitionssprache.

roles Das Element **roles** enthält eine Liste aller Rollen und ihrer Eigenschaften. Dazu beinhaltet es eine Liste von beliebig vielen **role**-Elementen.

Ein **role**-Element besteht aus dem Pflicht-Attribut *name*, welches den global eindeutigen Namen der Rolle enthält sowie den folgenden Kindelementen:

- Das Element **description** enthält einen String mit einer genaueren Beschreibung der Rolle. Das Element ist optional.
- Das Element optionale **properties** dient der Speicherung von frei definierbaren Eigenschaften der Rolle. Dazu enthält es als Kindelemente eine Liste mit mindestens einem **property**-Element. Jedes der Elemente hat genau die Attribute *key* und *value*, welche die Eigenschaften als Schlüssel-Wert-Paare kodieren. Der Schlüssel einer Eigenschaft muss dabei, entsprechend der formalen Definition, innerhalb der Menge der Eigenschaften einer Rolle eindeutig sein.

users Das Element **users** enthält eine Liste aller Benutzer. Dazu beinhaltet es eine Liste von beliebig vielen **user**-Elementen.

Ein **user**-Element besteht aus dem Attribut *name*, welches den Namen des Benutzers enthält, sowie dem Attribut *id*, welches die global eindeutige ID des Benutzers beinhaltet. Beide Attribute sind Pflichtelemente. Neben den Attributen enthält das Element die folgenden Kindelemente:

- Das Element **roles** dient der Definition aller Rollen des Benutzers. Dazu enthält das Element als Kindelemente eine Liste von **role**-Elementen, die als Text den Namen einer Rolle beinhalten. Ein Benutzer kann beliebig viele Rollen haben.

  Das Konzept erzwingt nicht, dass die referenzierte Rolle in der Datei auch wirklich definiert wurde. Dies zu erzwingen, sofern es notwendig ist, ist Aufgabe einer konkreten Implementierung.

- Das Element optionale **properties** dient zur Speicherung von frei definierbaren Eigenschaften des Benutzers. Dazu enthält es als Kindelemente eine Liste mit mindestens einem **property**-Element. Jedes der Elemente hat genau die Attribute *key* und *value*, welche die Eigenschaften als Schlüssel-Wert-Paare kodieren. Der Schlüssel einer Eigenschaft muss dabei, entsprechend der formalen Definition, innerhalb der Menge der Eigenschaften eines Benutzers eindeutig sein.

4.3.2. Sichtdefinitionssprache

Analog zur Benutzer- und Rollendefinition wird die Sichtdefinitionssprache im Folgenden im Detail beschrieben. Zur Übersicht wurde in Abbildung 4.2 wieder eine grafische Repräsentation des XML-Schemas erstellt. Das vollständige Schema in textueller Form findet sich in Anhang A.2.

Das Wurzelelements **views** hat genau ein Attribut *version*. Dieses Attribut gibt die zugrunde liegende Schema-Version an und ist für das hier beschriebene Schema „1.0“. In diesem Element sind genau drei Kindelemente erlaubt. Alle drei Kindelemente sind Pflichtelemente.

- Das Element **meta** dient zur Speicherung von Metainformationen.

- Das Element **preferences** dient zur Speicherung von sichtbezogenen Einstellungen.

- Das Element **elements** enthält die eigentliche Sichtdefinition.

Die drei Abschnitte werden im Folgenden detaillierter beschrieben.

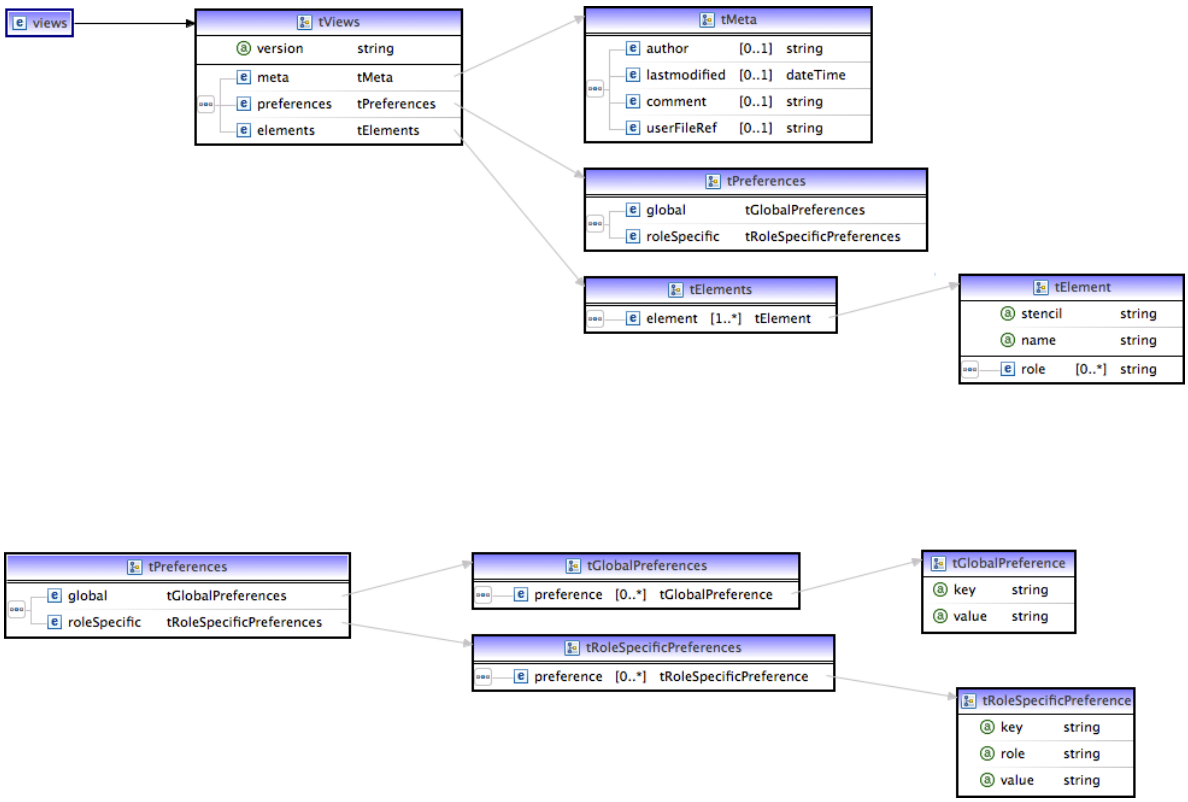

Abbildung 4.2.: Grafische Darstellung des XML Schemas der Sichtdefinitionssprache.

meta Das Element **meta** enthält allgemeine Informationen zur Sichtendefinition. Es enthält vier Kindelemente, die allesamt optional sind.

- Das Element **author** enthält den Namen des Autors der Sichtdefinition.
- Das Element **lastmodified** enthält einen Zeitstempel der letzten Bearbeitung.
- Das Element **comment** enthält einen allgemeinen Kommentar zur Sichtdefinition, zum Beispiel Hinweise für spätere Bearbeiter.
- Das Element **userFileRef** enthält eine Referenz auf die Datei mit der Benutzer- und Rollendefinition.

preferences In diesem Abschnitt der Datei können Einstellungen abgespeichert werden.

- Im Element **global** können globale Einstellungen definiert werden. Dazu hält es eine Liste von beliebig vielen **preference**-Elementen. Jedes der **preference**-Elemente hat genau die Attribute *key* und *value*, welche die Einstellungen als Schlüssel-Wert-Paare

kodieren. Der Schlüssel einer Einstellung muss dabei, entsprechend der formalen Definition, innerhalb der Menge aller globalen Einstellungen eindeutig sein.

- Im Element **roleSpecific** wird analog zu den globalen Einstellungen ebenfalls eine Liste von beliebig vielen **preference**-Elementen gehalten. Im Gegensatz zu den allgemeinen Einstellungen besitzen diese **preference**-Elemente als drittes Attribut noch das Attribut *role*. In diesem Attribut wird die Rolle angegeben, auf welches es sich bezieht. Die Kombination aus Schlüssel und Rolle einer Einstellung muss dabei, entsprechend der formalen Definition, innerhalb der Menge aller rollenspezifischen Einstellungen eindeutig sein.

Die Definition der Einstellungen ist bewusst allgemein gehalten. Dadurch ist eine gute Erweiterbarkeit und Anpassbarkeit gegeben. Zwei unterschiedliche Implementierungen dieses Konzepts werden dadurch nicht zwingend dieselben Einstellungen unterstützen. Sollten mehrere unterschiedliche Implementierungen parallel betrieben werden und sollten diese Datenaustausch betreiben müssen, so ist eventuell ein Konverter nötig, welcher Einstellungen der einen Implementierung in Einstellungen für die andere Implementierung übersetzt.

elements In diesem Abschnitt findet die eigentliche Sichtdefinition statt. Ähnlich den Einstellungen wurde auch dieser Abschnitt allgemein gehalten um Anpassbarkeit und Erweiterbarkeit zu gewährleisten.

Die Definition von Sichtbarkeiten erfolgt pro Element, wobei ein Element im Kontext dieser Arbeit einer KPI oder einem Ziel entspricht. Dieser Abschnitt der Datei enthält daher eine Liste von **element**-Elementen, welche jeweils ein Element des Diagramms abbilden. Es muss mindestens ein Element existieren.

Um ein Diagramm-Element eindeutig zu kennzeichnen werden zwei Attribute benötigt. Das Attribut *name* referenziert das Element anhand seines Namens. Da aber ein Name innerhalb der Menge aller Elemente nicht zwingend eindeutig ist, man denke zum Beispiel an eine KPI und ein Ziel gleichen Namens, wird ein weiteres Attribut benötigt. Daher wird zusätzlich zum Namen des Elements auch noch der entsprechende Stencil und damit der Typ des Elements im Attribut *stencil* referenziert. Es beinhaltet den Namespace des Stencils.

Dem so eindeutig referenzierten Element kann nun mit dem Element **role** eine beliebig mächtige Menge von Rollen zugewiesen werden. Diese Rollen sollten in der Datei mit den Benutzer- und Rollendefinitionen definiert sein. Das Sichtenkonzept erzwingt die Definition der Rollen jedoch nicht.

4.4. Grafische Sprache

Die grafische Sprache soll zur Modellierung der Sichten durch den Benutzer dienen. Die Modellierung findet direkt im KPI-Modell statt. Um dem Benutzer die Verwendung so einfach wie möglich zu gestalten, wird dazu außerdem derselbe Editor verwendet.

4.4.1. Syntaktische Elemente

Um die Sichten zu definieren müssen KPIs und Ziele mit Rollen annotiert werden können. Die grafische Sprache muss also um ein entsprechendes Element zur Rollenangabe erweitert werden. Es sind prinzipiell zwei Wege denkbar: Entweder kann ein bereits in BPMN existierendes Element wiederverwendet werden oder es wird ein neues Element erstellt. Zur Wiederverwendung bietet sich hier das Annotations-Element von BPMN an.

Die Rollen werden in beiden Fällen als Liste erfasst. Dabei ist es unerheblich, ob die Rollen durch ein entsprechendes Trennzeichen getrennt in ein Attribut des Elements geschrieben werden oder ob das Element ein Attribut vom Typ Array bereitstellt, in welchem die Liste der Rollen erfasst wird. Sollte ein Editor für letzteres eine Möglichkeit bieten, so sollte diese aber verwendet werden, wenn es der Benutzbarkeit dient.

Rollendefinition mit BPMN-Annotations BPMN bietet ein Annotation-Element, welches zur Angabe von Rollen genutzt werden kann. Dazu wird die Annotation an eine KPI oder ein Ziel gebunden und die einzelnen Rollen durch ein Trennzeichen, beispielsweise einen Zeilenumbruch, getrennt in die Annotation geschrieben. Ein Beispiel zeigt Abbildung 4.3.

Diese Vorgehensweise hat den Vorteil, dass keine neuen Elemente eingeführt werden müssen. Je mehr Erweiterungen ihre eigenen Elemente einführen, umso unübersichtlicher kann die resultierende Sprache aus BMPN und den Erweiterungen werden. Durch die Verwendung existierender Elemente, wird dieses Problem umgangen.

Allerdings hat diese Vorgehensweise auch den gravierenden Nachteil, dass Rollendefinitionen und andere Annotations nicht mehr auf den ersten Blick zu unterscheiden wären. Soll es außerdem möglich sein, normale Annotations und Rollendefinitionen an denselben Elementtypen zu verwenden, muss die Syntax der Rollendefinition um ein Schlüsselwort erweitert werden. Dieses Schlüsselwort dient dann zur Unterscheidung zwischen normalen Annotationen und Rollendefinitionen.

Dazu kommt, dass diese Vorgehensweise tendenziell Konflikte mit anderen Erweiterungen verursachen könnte, die ebenfalls das Annotation-Element verwenden und dieses möglicherweise erweitern, oder dasselbe Schlüsselworte im Annotations-Element einsetzen.

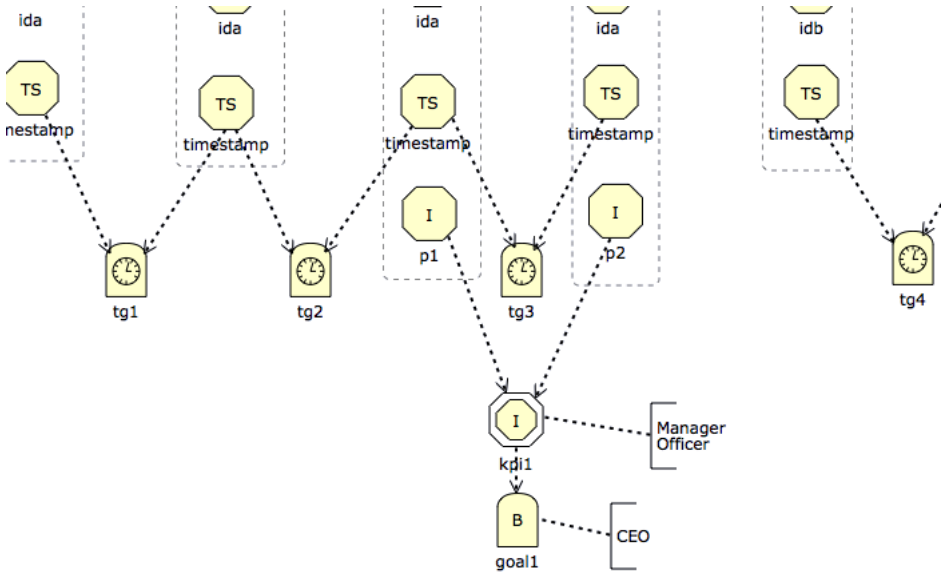

Abbildung 4.3.: Rollen an KPIs unter Verwendung des Annotation-Elements von BPMN.

Rollendefinition mit eigenen Elementen Um die genannten Probleme bei der Verwendung des Annotation-Elements zu umgehen, wird die vorhandene Sprache um ein eigenes Element zur Rollendefinition erweitert. Ein Beispiel für ein solches Element zeigt Abbildung 4.4.

Das neue Element namens „Associated Roles“ unterscheidet sich nur durch sein Icon vom Annotation-Element. Dadurch kann jedoch auf ein spezielles Schlüsselwort zur Kennzeichnung von Rollendefinitionen verzichtet werden. Außerdem hebt sich das neue Element bei Diagrammen mit vielen Annotations-Elementen besser von diesen ab.

Das Icon ist bewusst an die Use-Case-Diagramme der UML angelehnt. Die Verwendung eines ähnlichen Icons macht einem Softwaretechniker das Verstehen des Elements einfacher. Für Benutzer, welche nicht mit den Methoden der Informatik vertraut sind, entspricht das Icon einem Strichmännchen und ist dadurch deutlich intuitiver zu verstehen, als es das Annotation-Element aus BPMN ist.

Man handelt sich dabei jedoch die vorherigen Abschnitt genannten Probleme ein, dass durch weitere Elemente die resultierende Sprache immer komplexer wird. Allerdings ist die zusätzliche Komplexität durch ein zusätzliches Element so gering, dass dieser Nachteil die Vorteile nicht aufwiegt. Daher verwendet dieses Konzept das beschriebene Element zur Angabe von Rollen.

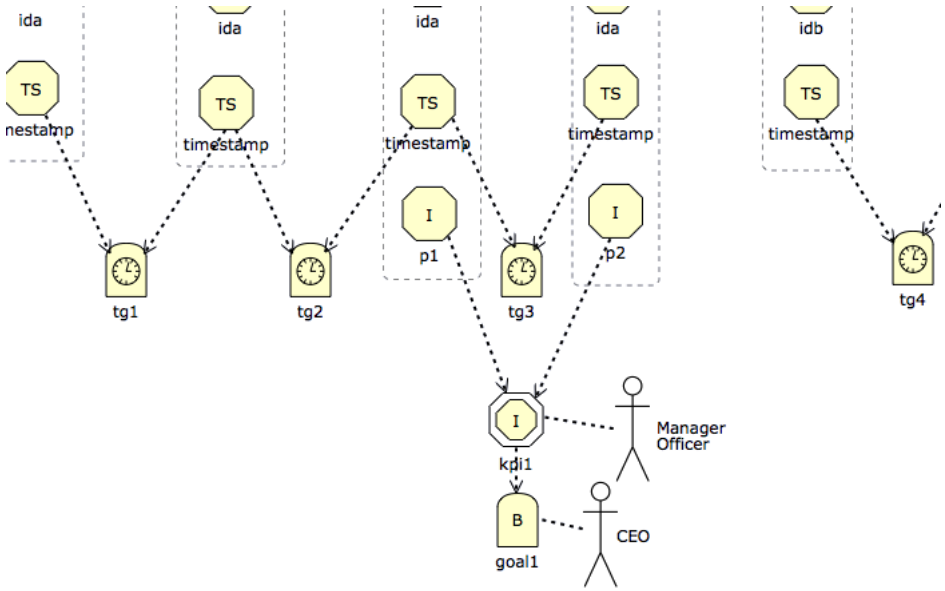

Abbildung 4.4.: Rollen an KPIs unter Verwendung einer eigenen Spracherweiterung.

Kanten Um die Rollendefinitionselemente mit den KPIs und Zielen zu verbinden, müssen einige Regeln definiert werden. Da es sich bei der Sichtendefinition weder um eine zeitliche Abfolge noch um einen gerichteten Informationsfluss handelt, wird eine ungerichtete Kante zum Einsatz kommen.

Implementierungsbedingt haben auch ungerichtete Kanten in Editoren mitunter eine Richtung. In diesem Fall ist es erlaubt, die Richtung der Verbindung auf eine der beiden möglichen Richtungen festzulegen. Diese willkürliche Festlegung hat jedoch keine semantische Bedeutung für die Sichtengenerierung.

Die Rollendefinitionselemente können mit folgenden Elementen von aPro verbunden werden:

- Key Performance Indicator
- Aggregated Key Performance Indicator
- Goal
- Timing Goal

4.4.2. Semantische Definition

Die bisherige Definition des Elements zur Rollendefinition ist zwar sehr intuitiv zu verwenden und zu verstehen, ist jedoch formal noch nicht vollständig. Da bisher lediglich die Syntax definiert wurde, kann es zu Situationen kommen, in denen die genaue Bedeutung des Sichtenmodells nicht eindeutig ist. Daher wird die syntaktische Definition nun noch durch eine semantische Definition ergänzt, bei welcher zuerst der allgemeine Fall definiert wird und dann auf die Sonderfälle eingegangen wird.

Allgemeine Definition Ein Element hat genau dann eine Rolle r, wenn es mit einem Rollendefinitionselement verbunden ist, welches die entsprechende Rolle in seiner Rollenliste enthält. Elemente ohne Rollendefinition besitzen daher keine Rollen und werden in keiner Sicht enthalten sein.

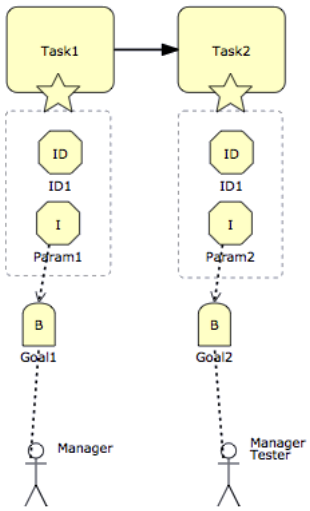

Abbildung 4.5.: Auftreten derselben Rolle in unterschiedlichen Elementen.

Mehrfach auftretende Rollen Dieselbe Rolle kann in mehreren Rollendefinitionselementen vorkommen. In diesem Fall beziehen sich beide Elemente auf dieselbe Rolle. Diese Regelung ergibt sich unmittelbar aus der allgemeinen Definition.

Aus Abbildung 4.5 ergeben sich also folgende Sichten:

- *Manager*: $\{Goal1, Goal2\}$
- *Tester*: $\{Goal2\}$

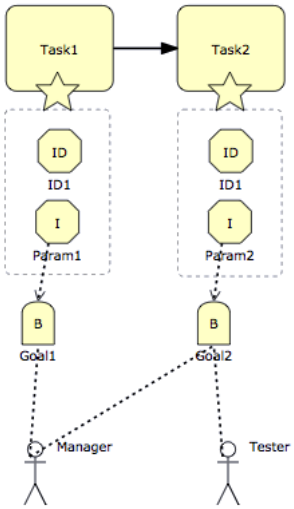

Abbildung 4.6.: Auftreten derselben Rolle an unterschiedlichen KPIs.

Mehrere Rollendefinitionen an einem Element Ein Element kann mit mehreren Rollendefinitionselementen verbunden sein. In diesem Fall hat das Element alle Rollen, die in den Rollendefinitionselementen vorkommen. Es handelt sich also um eine additive Rollenzuweisung. Diese Regelung ergibt sich unmittelbar aus der allgemeinen Definition.

Aus Abbildung 4.6 ergeben sich also folgende Sichten:

- *Manager*: $\{Goal1, Goal2\}$
- *Tester*: $\{Goal2\}$

4.5. Rollenvererbung

Um dem Modellierer Schreibarbeit zu sparen und Modelle dadurch übersichtlicher zu halten, wird das Sichtenkonzept um ein Konzept zur Rollenvererbung erweitert. Einstellungen zur Rollenvererbung können, wie auch alle anderen Einstellungen, im Preferences-Abschnitt der XML-Datei gespeichert werden.

Abbildung 4.7 zeigt zwei Beispiele, in denen Rollenvererbung zum Einsatz kommen kann. Entsprechend der semantischen Definition haben die Elemente ohne Rollendefinitionselement keine Rollen und sind demnach in keiner Sicht sichtbar. Durch Rollenvererbung könnten die Rollen zum Beispiel auf Vorgänger oder Nachfolger vererbt werden. Dadurch würden auch die bisher rollenlosen Elemente die Rolle Manager bekommen, ohne zusätzliche Rollendefinitionselemente oder Kanten quer durch das Diagramm. Dadurch kann die

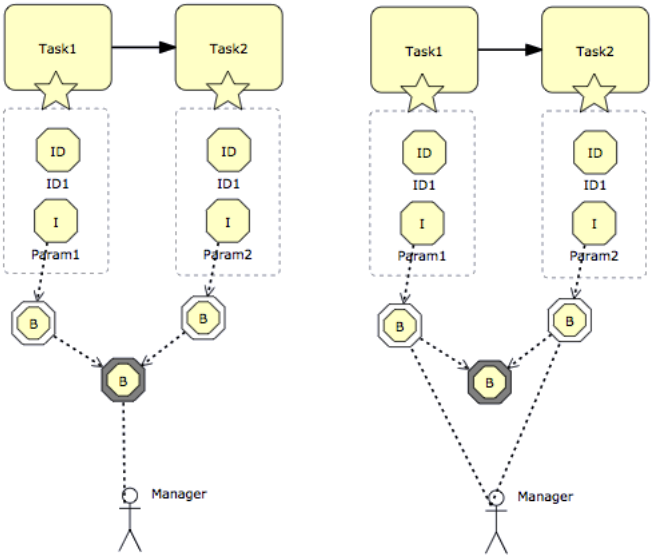

Abbildung 4.7.: Situationen, in welchen Rollenvererbung zum Einsatz kommen könnte.

Übersichtlichkeit des Diagramms gewahrt werden, die Rollenverteilung im Diagramm ist dann jedoch nicht mehr auf den ersten Blick ersichtlich.

Ein sinnvolles Vererbungskonzept ist stark vom entsprechenden Einsatzbereich abhängig, und muss daher immer auf den Einsatzbereich abgestimmt werden. So hat zum Beispiel der Aufbau der Diagramme einen erheblichen Einfluss auf den Nutzen eines solchen Konzepts. Bei sehr tiefen KPI-Diagrammen kann Vererbung zu einer deutlichen Verringerung der Komplexität der Rollendefinition führen. Weitere Eigenschaften, welche die Rollenvererbung beeinflussen können, sind zum Beispiel interne Richtlinien der Organisation, welche das Rechtekonzept einsetzt, oder der Ausbildungsgrad der Mitarbeiter, welche das abstrakte Denken und die bevorzugten Arbeits- und Modellierungsmethoden beeinflussen können.

Drei mögliche Varianten der Rollenvererbung werden im Folgenden vorgestellt. Wie erwähnt sind dies jedoch nicht alle denkbaren Varianten. Das Konzept legt auch nicht fest, ob immer nur eine Vererbung aktiv sein kann oder ob mehrere Vererbungsmodi zugleich aktiv sein können. Diese Entscheidungen müssen je nach den Erfordernissen des Einsatzgebiets getroffen werden.

4.5.1. Rollenvererbung aufwärts

Ein Element hat genau dann die Rolle r, wenn eine der beiden folgenden Bedingungen zutrifft:

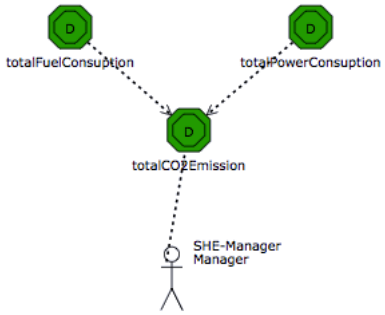

Abbildung 4.8.: Beispiel mit Vorschau für Rolle Manager bei aufwärts gerichteter Rollenvererbung.

1. Dem Element wurde die Rolle *r* durch ein „Associated Roles"-Element zugewiesen
2. Einer der Nachfolger des Elements hat die Rolle *r*

Ein Beispiel zeigt Abbildung 4.8.

4.5.2. Rollenvererbung abwärts mit einem Vorgänger

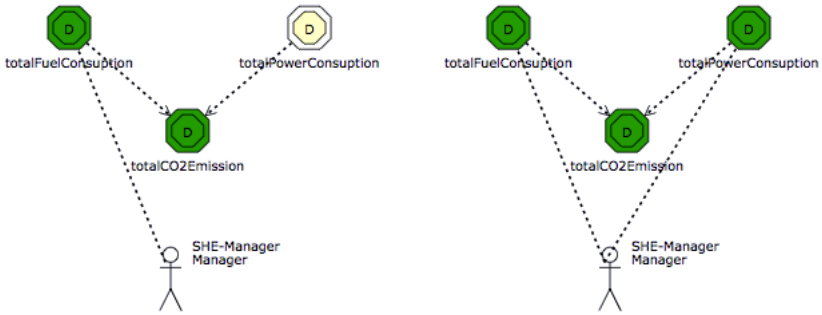

Abbildung 4.9.: Beispiel mit Vorschau für Rolle Manager bei abwärts gerichteter Rollenvererbung mit einem Vorgänger.

Ein Element hat genau dann die Rolle *r*, wenn eine der beiden folgenden Bedingungen zutrifft:

1. Dem Element wurde die Rolle *r* durch ein „Associated Roles"-Element zugewiesen
2. Einer der Vorgänger des Elements hat die Rolle *r*

Ein Beispiel zeigt Abbildung 4.9.

4.5.3. Rollenvererbung abwärts mit allen Vorgängern

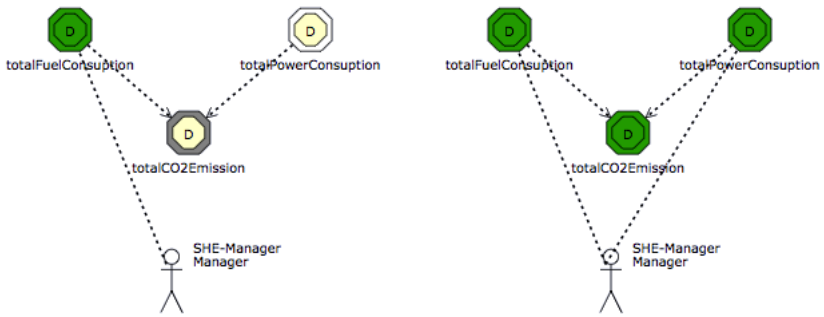

Abbildung 4.10.: Beispiel mit Vorschau für Rolle Manager bei abwärts gerichteter Rollenvererbung mit allen Vorgängern.

Ein Element hat genau dann die Rolle r, wenn eine der beiden folgenden Bedingungen zutrifft:

1. Dem Element wurde die Rolle r durch ein „Associated Roles"-Element zugewiesen
2. Alle Vorgänger des Elements haben die Rolle r

Ein Beispiel zeigt Abbildung 4.10.

4.6. Custom Alerts

Der aPro Prototyp sieht einen sogenannten Custom Alert vor. Dieser Typ von Alert dient als eine Art Catch-All-Alert. Er stellt also eine Schnittstelle für beliebige Alerts dar. Auf diese Art können auch unerwartete Fehler, wie zum Beispiel Überlastung eines Servers, im Dashboard dargestellt werden.

Will man diese Alerts ebenfalls in ein Sichtenkonzept integrieren, so muss ein anderer Weg gegangen werden als für Kennwerte. Dies ergibt sich aus der Catch-All-Eigenschaft, aufgrund welcher zum Zeitpunkt der Modellierung nicht bekannt ist, welche Arten von Alerts auftreten können. Es kann daher nicht, wie bei Zielen, spezifisch für einen Alert Rollen zugewiesen werden. Für Rollendashboards ist es jedoch wünschenswert, wenn die Custom Alerts ebenfalls in Grenzen beeinflusst werden könnten.

Um dies zu erreichen bieten sich Reguläre Ausdrücke oder Filterfunktionen an, mit welchen der Inhalt des Alerts auf Relevanz geprüft werden kann. Die Eingrenzung von Alerts geschieht pro Rolle. Als Speicherort dienen daher die Properties einer Rolle unter dem Schlüssel „customAlertQuery“.

Neben der Eingrenzung durch Reguläre Ausdrücke bieten sich noch weitere Mechanismen an. Denkbar sind zum Beispiel auch Funktionen, welche den Inhalt des Alerts als Parameter entgegen nehmen und einen booleschen Wert als Rückgabe aufweisen. Im Falle des Wertes „true“ wird der Alert weitergegeben, ansonsten nicht. Die für die Funktion verwendete Sprache ist in diesem Fall von der verwendeten Implementierungssprache abhängig. Kapitel 7.2.5 zeigt noch einige weitere Möglichkeiten zum Umgang mit Custom Alerts und ähnlichen Datenquellen auf, die jedoch die bisherigen Möglichkeiten von aPro übersteigen.

4.7. Einbettung der Sichtengenerierung in bestehende Arbeitsabläufe

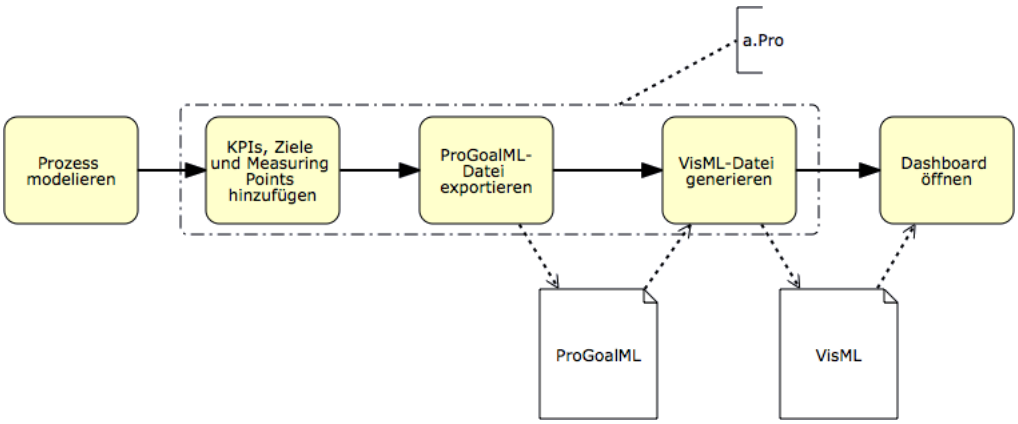

Abbildung 4.11.: Prozess zur Erstellung eines Dashboards in aPro

In Abbildung 4.11 wird der Prozess zur Erstellung eines Dashboards im aPro Prototyp gezeigt. Wie man erkennt, sieht aPro nur die Generierung eines einzigen vollständigen Dashboards aus der ProGoalML-Datei vor. Diese Umwandlung muss so erweitert werden, dass sie mehrere Dashboards entsprechend der Sichtdefinitionen erstellt. Um dies zu erreichen, gibt es drei mögliche Ansatzpunkte:

- Aufteilen der ProGoalML-Datei (siehe Abbildung 4.12)
- Anpassung des Umwandlungsalgorithmus (siehe Abbildung 4.13)
- Aufteilen der VisML-Datei (siehe Abbildung 4.14)

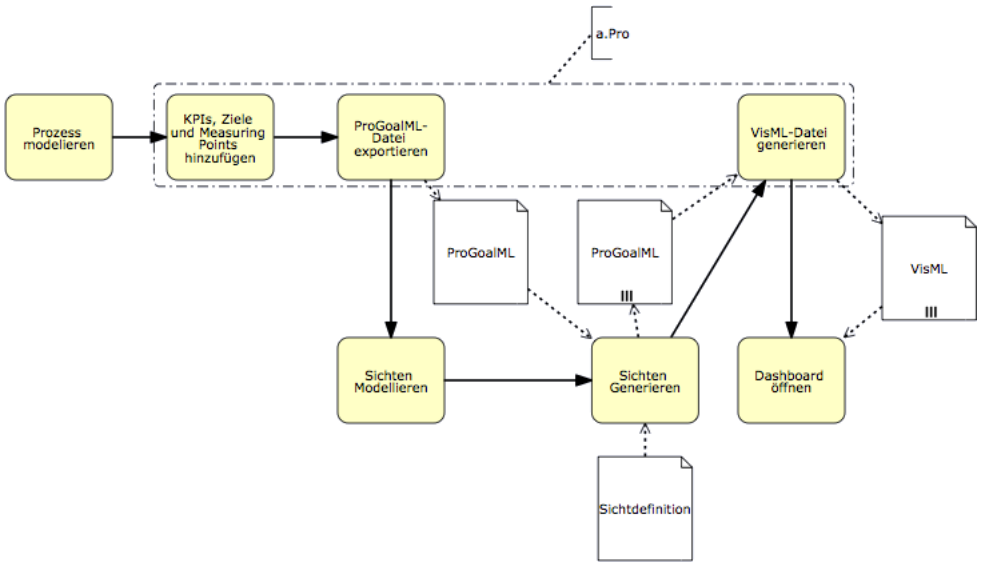

Abbildung 4.12.: Prozess zur Erstellung mehrerer Dashboards in aPro durch Aufspaltung der ProGoalML-Datei.

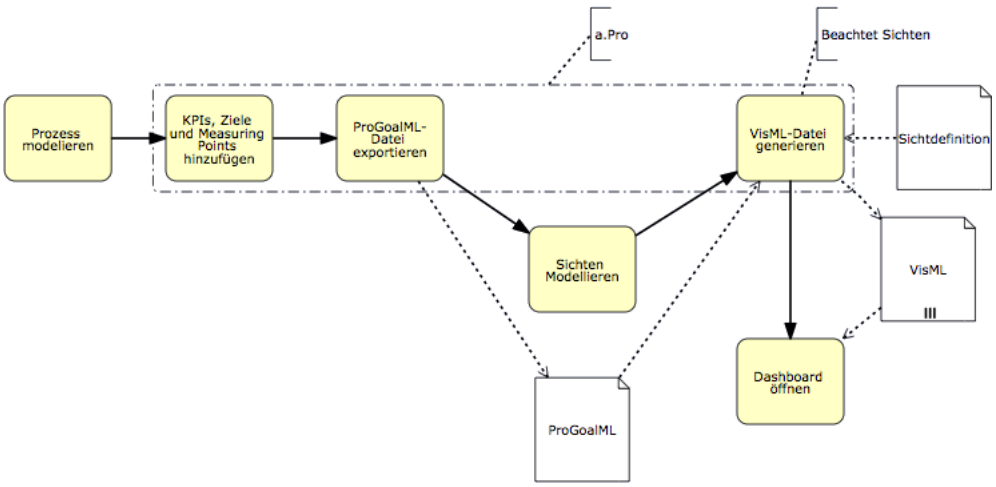

Abbildung 4.13.: Prozess zur Erstellung mehrerer Dashboards in aPro mit angepasstem Umwandlungsalgorithmus.

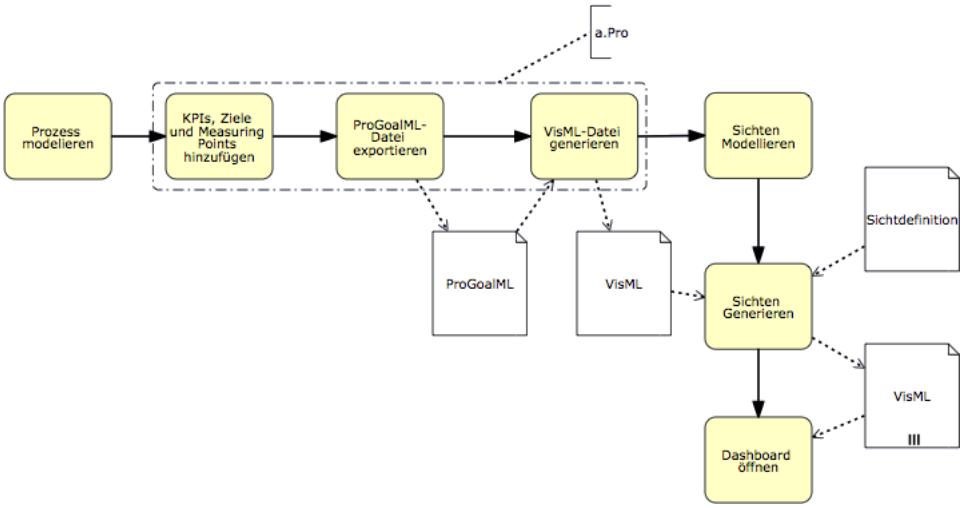

Abbildung 4.14.: Prozess zur Erstellung mehrerer Dashboards in aPro durch Aufspaltung der VisML-Datei.

Von den gezeigten Ansätzen erweist sich die Aufteilung der VisML-Datei als die geeignetste Lösung. Dieser Ansatz benötigt die geringsten Eingriffe in bestehende Software. Sämtliche Modifikationen im Workflow befinden sich außerhalb der Zuständigkeit von aPro. Insbesondere kann dadurch der Umwandlungsmechanismus von einer ProGoalML-Datei in eine VisML-Datei unverändert übernommen werden. Neben der einfacheren Implementierung bringt dies einige Vorteile bei zukünftigen Erweiterungen der Software und bei der Wartung, da die Zuständigkeiten klar getrennt sind.

Ein weiterer Vorteil dieses Ansatzes vereinfacht die Verwendung des Konzepts in der Praxis. Durch die Trennung der Dashboardgenerierung und der Sichtengenerierung können diese Schritte getrennt durchgeführt werden. Dadurch können diese Arbeiten von unterschiedlichen Anwendern durchgeführt werden. Außerdem können Sichten zu späteren Zeitpunkten geändert und neu generiert werden, ohne dass das gesamte Dashboard erneut generiert und deployt werden muss.

Neben diesen Vorteilen umgeht der Ansatz einige Probleme der anderen beiden Ansätze. So kann es beim ersten Ansatz zu invaliden ProGoalML-Dateien kommen, wenn abgeleitete KPIs sichtbar sein sollen, die KPI, von welcher abgeleitet wird, jedoch nicht. Weiterhin würden Parameter, welche von KPIs in mehreren unterschiedlichen Sichten verwendet werden, zur mehrfachen Generierung von CEP-Abfragen, Resultshemata und weiteren Artefakten führen. Durch Änderungen am Algorithmus der Dashboardgenerierung könnten diese Probleme gelöst werden. Allerdings gingen dadurch die bereits genannten Vorteile der klar getrennten Module verloren.

5. Prototyp

Zur Demonstration des Konzepts wurde ein Prototyp basierend auf Oryx und dem Prototyp von aPro erstellt. Als Plattform zur Anzeige des generierten Dashboards dient ein einfaches HTML-Dashboard, welches vom aPro-Prototyp mitgeliefert wird.

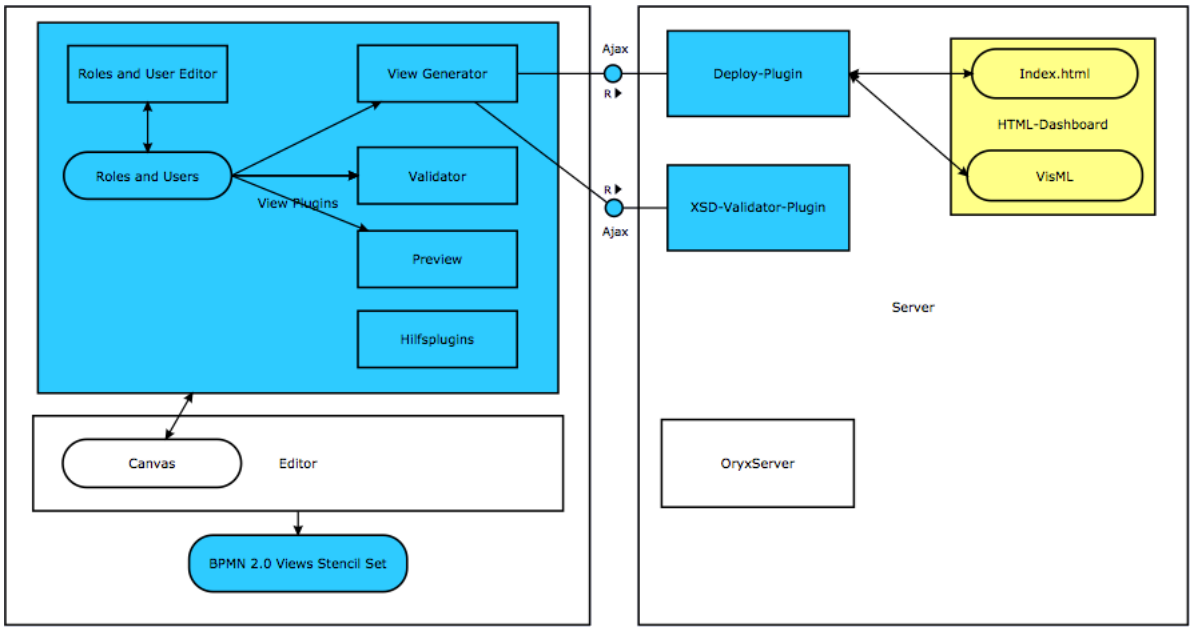

Abbildung 5.1.: FMC-Diagramm der wichtigsten Komponenten. In blau Komponenten des Prototypen, in gelb Komponenten des Prototyps von aPro.

Eine grobe Übersicht über die wichtigsten Komponenten des Prototyps zeigt Abbildung 5.1. Die dort abgebildeten hellblauen Komponenten sind neue Komponenten des Prototyps. In hellgelb hervorgehoben sind Komponenten, welche von aPro mitgeliefert wurden und nicht im Rahmen der Diplomarbeit entwickelt wurden. Die restlichen weißen Komponenten sind im Grundumfang von Oryx enthalten.

Bei der Erstellung des Prototyps wurde darauf geachtet, dass er nicht stark von dem aPro-Prototyp abhängig ist und nach Möglichkeit nicht in dessen Quellcode eingreift. Viele Funktionen des Prototyps sind völlig unabhängig von dem aPro-Prototyp nutzbar, jedoch bei weitem nicht alle. Auch war das Vermeiden von Abhängigkeiten zum aPro-Prototyp nicht

immer möglich, so mussten für die Sichterzeugung von deployten Dashboards natürlich einige Annahmen über die von einer bestimmten Version des aPro-Prototyps erzeugten Datenstrukturen auf dem Server getroffen werden.

Der Prototyp ermöglicht zwei unterschiedliche Herangehensweisen an die Sichtdefinition. Zum einen kann eine Sicht per XML-Datei definiert werden und jede beliebige visML Datei als Input verwendet werden. Zum anderen kann die Sichtdefinition aber auch vollständig mit dem aPro-Prototyp integriert in demselben Editor und demselben Modell stattfinden. Dabei ist es möglich, ein auf dem lokalen Server deployes Dashboard in Sichten zu zerlegen.

5.1. Verwendete Software

Für diesen Prototyp werden mehrere bestehende Softwaresysteme verwendet. Dazu gehört zum einen die Architektur aPro mit ihrem Prototyp, zum anderen der Editor Oryx. Beide sollen im Folgenden grob erläutert werden.

5.1.1. aPro und ProGaoIML

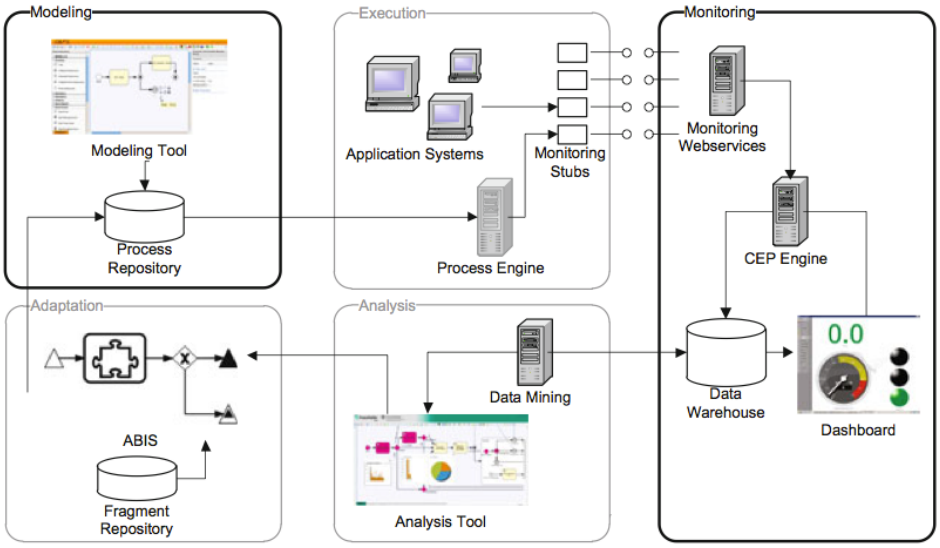

Abbildung 5.2.: aPro-Architektur. Abbildung entnommen aus [KK12b].

aPro [KK12b] ist eine am Fraunhofer IAO in Zusammenarbeit mit der Universität Stuttgart entwickelte Architektur für die Optimierung von Prozessen. Dabei setzt aPro jedoch nicht

voraus, dass der Prozess auf einer Workflowengine ausgeführt wird. Einen Überblick über die Architektur gibt Abbildung 5.2.

5.1.2. aPro und ProGaolML

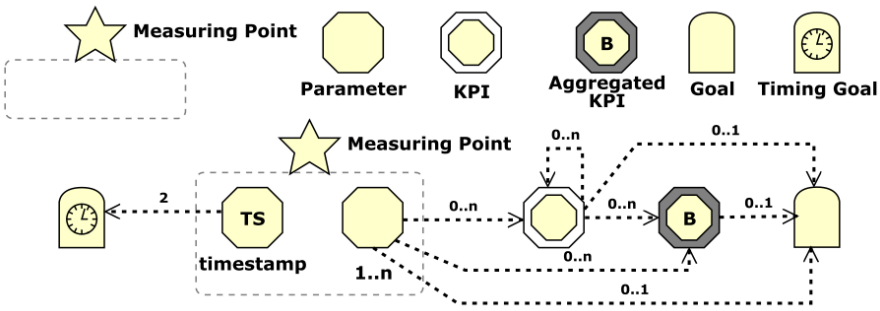

Abbildung 5.3.: ProGoalML-Elemente und ihre Verknüpfungsmöglichkeiten. Abbildung entnommen aus [KK12b].

Ein Teil von aPro ist die Sprache ProGoalML, welche die Definition von KPIs und Zielen ermöglicht. Dazu wird BPMN um einige Elemente erweitert. Messpunkte definieren Parameter, die an bestimmten Tasks gemessen werden. Die Werte aus den Parametern können dann in KPIs und aggregierten KPIs, oder in Zielen verwendet werden. Als spezieller Zieltyp wurde außerdem das sogenannte Timinggoal definiert, welches nur Zeitstempel als Eingabe entgegennimmt. Die Elemente und ihre möglichen Verknüpfungen zeigt Abbildung 5.3.

Analog zu den grafischen Elementen existiert außerdem eine XML-Sprache, welche dieselben Sachverhalte abbildet. Auf Basis dieser Datei können die nötigen Dokumente für das Monitoring generiert werden, unter anderem CEP-Regeln für die CEP-Engine, welche die zu überwachenden Daten entgegennimmt und Monitoring-Stubs um die Daten von den überwachten Prozessen entgegenzunehmen und weiterzuleiten.

Von aPro existiert ein Prototyp, basierend auf Oryx. Der Prototyp wurde im Rahmen eines Studienprojekts entwickelt und dient auch als Basis für den Prototyp dieser Arbeit.

5.1.3. Oryx

Oryx [Tsc07] ist ein webbasierter Editor für grafische Modelle. Er unterstützt neben einigen anderen Diagrammtypen insbesondere BPMN [Pol07]. Eine Erweiterung des Editors um eigene Diagrammtypen ist über sogenannte StencilSets möglich.

Oryx ist Open Source unter der MIT-Lizenz und kann damit frei verwendet und erweitert werden. Neben der freien Version existiert ein kommerzieller Ableger namens Signavio. Für diese Arbeit kommt jedoch ausschließlich Oryx zum Einsatz.

Erweiterbarkeit

Oryx ist bereits von Haus aus erweiterbar ausgelegt. Neben dem Erstellen neuer Diagrammtypen durch sogenannte StencilSets, wird auch das Erstellen von Plugins [Tsc07] unterstützt. Außerdem ermöglicht Oryx das Erstellen von Profilen. Wird der Editor in einem solchen Profil gestartet, stehen dem Benutzer die vom Ersteller des Profils vorgesehenen StencilSets und Plugins zur Verfügung. So ist es möglich, dass der Editor sich je nach Profil unterschiedlich verhält. Die grundlegenden Bedienelemente bleiben jedoch, unabhängig vom gewählten Profil, immer gleich.

Client-Plugins Mithilfe von Client-Plugins kann Oryx um neue Funktionen erweitert werden. Oryx stellt dazu eine Facade bereit, die den registrierten Plugins einige Oryx-Funktionen zur Verfügung stellt.

Die wichtigste von der Facade zur Verfügung gestellt Funktion ist die Methode *offer()*, welche einem Plugin das Anbieten von Funktionalität in der Editor-GUI ermöglicht. Weitere Funktionen ermöglichen dem Plugin den Zugriff auf den Diagramminhalt des geöffneten Diagramms. Das Plugin kann so entweder Informationen aus dem Diagramm extrahieren oder das Diagramm manipulieren.

Eine weitere von Oryx angebotene Funktionalität sind Events. Einige Events sind von Oryx und dessen standardmäßig installierten Plugins vordefiniert. Durch sie können Plugins auf bestimmte Ereignisse reagieren. Andererseits können Plugins auch eigene Events definieren, auf die wiederum von anderen Plugins reagiert werden kann.

Ein großer Teil der Funktionen in Oryx ist selbst als Plugins implementiert. Das betrifft insbesondere die auf bestimmte Diagrammtypen begrenzten Funktionen. Daneben sind auch im Rahmen von universitären Projekten einige Plugins entstanden. Ein Beispiel ist der aPro -Prototyp.

Server-Plugins Auch auf der Server-Seite können Plugins in Form von Servlets erstellt werden. Dadurch stehen dem Benutzer die mächtigen Funktionalitäten der Java-Bibliothek zur Verfügung. Im Rahmen dieser Arbeit werden Server-Plugins nur in sehr geringem Maß angewendet.

StencilSets Neben der Erweiterung durch Plugins sieht Oryx auch die Erstellung eigener Diagrammtypen und die Erweiterung bestehender Diagrammtypen vor [Pet07].

Zur Definition von Diagrammtypen dienen StencilSets, welche aus mehreren Teilen bestehen:

- Eine Pixelgrafik jedes Stencils
- Eine SVG-Grafik jedes Stencils
- Eine json-Datei mit allen Eigenschaften des StencilSets

Die SVG-Datei eines StencilSets ist dabei mit einigen Oryx-spezifischen Informationen angereichert. So können in der Datei Textbereiche definiert werden, welche den Inhalt von Attributen des Stencils, beispielsweise dessen Namen, darstellen. Außerdem können Angaben zu Manipulationsmöglichkeiten und zu Andockpunkten für Kanten in der SVG-Datei gemacht werden.

In der json-Datei wird dann jeder Stencil mit all seinen Attributen und den zugehörigen Grafiken definiert. Außerdem können Regeln für die Verbindung der Stencils untereinander angegeben werden sowie Regeln für Inklusionsbeziehungen zwischen Stencils. So kann zum Beispiel sichergestellt werden, dass ein Stencil nur eine bestimmte Anzahl eingehender Kanten hat oder nur mit bestimmten anderen Stencils verbunden werden kann.

5.2. Funktionsbeschreibung

Die folgenden Abschnitte beschreiben den Funktionsumfang des Prototyps im Detail. Es werden alle Stencils, Eingabemasken und Funktionen beschrieben.

Bei der Beschreibung wird von der englischen Standardeinstellung ausgegangen. Daneben liefert der Prototyp auch deutsche Sprachdateien mit. Weitere Sprachen, die in Oryx standardmäßig enthalten sind, werden jedoch nicht unterstützt.

5.2.1. Stencil

Der Prototyp erweitert BPMN um einen Stencil namens „Associated Roles“ zur Angabe von Rollen. Der Stencil steht unter dem Punkt „Views“ im „Shape Repository“ zur Verfügung und kann wie jeder andere Stencil auch im Editor manipuliert werden.

Attribute Der Stencil bietet lediglich ein Attribut namens „Roles“. In dieses Attribut werden die Rollen, getrennt durch Zeilenumbruch, eingetragen. Das Attribut kann entweder bei ausgewähltem Element im Bereich „Properties“ des Editors angegeben werden oder nach Doppelklick in die freie Fläche des Elements direkt im Diagramm eingegeben werden.

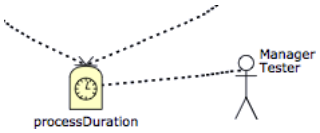

Abbildung 5.4.: Beispiel der Verwendung des „Associated Roles“ Stencils.

Der Editor nimmt beim Eintragen der Rolle keine automatische Prüfung vor, ob die Rolle in der Benutzerdefinition überhaupt definiert ist. Es steht jedoch eine entsprechende Prüfung zur Verfügung. Genaueres ist im Kapitel 5.2.7 nachzulesen.

Verwendung Die Verbindung des Elements „Associated Roles“ mit Elementen der oben genannten Stencils wird mithilfe der „Association (undirected)“ von BPMN hergestellt. Dieser Stencil befindet sich im Abschnitt „Connecting Objects“ im Shape Repository. Obgleich die Kante von ihrer Natur her ungerichtet ist, unterscheidet Oryx eine von- und eine zu-Seite der Kante. Infolgedessen ist die Kante aus Sicht des Editors gerichtet, wie jede andere Kante. Aus Gründen der Einfachheit ist daher die Verwendung der Kante auf eine Richtung beschränkt. Dies wirkt sich nicht negativ auf die Ausdrucksfähigkeit des Diagramms aus.

Die semantische Bedeutung des Shapes kann im Kapitel 4.4.2 des Konzepts nachgelesen werden.

5.2.2. Aufruf der Plugins

Abbildung 5.5.: Ausschnitt der Oryx Toolbar mit den drei Menüs des Prototyps.

Der Prototyp besteht aus mehreren Plugins, die mehrere Einträge zur Toolbar hinzufügen. Zur besseren Verwendbarkeit wurden die Einträge thematisch zu Gruppen zusammengefügt. Abbildung 5.5 zeigt den entsprechenden Ausschnitt der Toolbar.

Im ersten Menü finden sich alle Plugins zur Sichtengenerierung. Im zweiten Menü wurden Einstellungen und allgemeine Hilfsmittel gruppiert. Im letzten Menü befinden sich alle Plugins zum Export von Dateien.

5.2.3. Pflegen der Rollen und Benutzerdaten

Der Prototyp stellt eine einfache Benutzerverwaltung zur Verfügung. Es können dort Rollen und Benutzer definiert werden. Das Pflegen der Rollen und Benutzer ist für die meisten Funktionen des Prototyps nicht zwingend notwendig. Lediglich für die Verwendung der Funktionen im Bezug auf Custom Alerts ist die Pflege der Rollen zwingend notwendig. Der Editor zum Pflegen von Rollen und Benutzerdaten findet sich in der Toolbar im Menü mit den Einstellungen des Prototyps unter dem Menüpunkt „Edit roles and users“.

Rollen pflegen

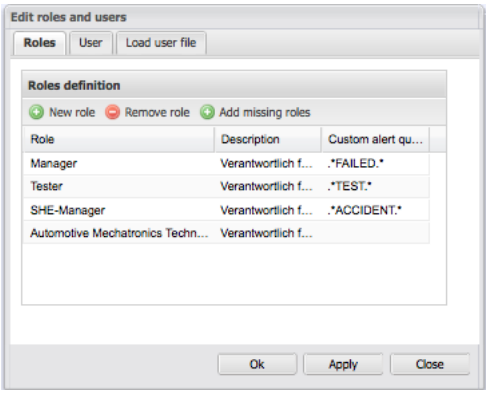

Abbildung 5.6.: GUI zum Pflegen der Rollen.

Das Pflegen der Rollen muss weitgehend von Hand stattfinden. Der Benutzer hat dabei die Möglichkeit, mit den Buttons über der Tabelle (siehe Abbildung 5.6) neue Einträge hinzuzufügen oder zu entfernen. Um dem Benutzer Schreibarbeit zu ersparen und die Gefahr von Tippfehlern zu verringern wurde außerdem eine Funktion eingebaut, welche das automatische Übernehmen aller noch fehlenden Rollen aus dem Diagramm ermöglicht.

Eine Rolle besteht im Prototyp aus den folgenden Feldern:

- **Role** Die Bezeichnung der Rolle
- **Description** Eine genauere Beschreibung (optional)
- **Custom alert query** Ein regulärer Ausdruck zur Einschränkung der Custom Alerts (optional) (siehe Kapitel 5.2.8)

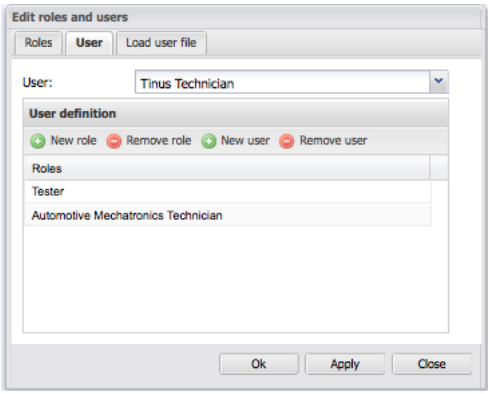

Abbildung 5.7.: GUI zum Pflegen der Benutzer.

Benutzer pflegen

Bestehende Benutzer bearbeiten Mithilfe der Combobox über der Tabelle (siehe Abbildung 5.7) kann der Benutzer einen bereits angelegten Benutzer auswählen. Die Rollen des Benutzers werden dann in der Tabelle angezeigt.

Mithilfe der Buttons über der Tabelle kann der Benutzer nun Rollen für diesen Benutzer hinzufügen oder löschen. Dabei erfolgt keine Prüfung, ob die verwendeten Rollen auch tatsächlich definiert und damit verfügbar sind. Es existiert jedoch eine entsprechende Prüfung. Genaueres ist im Kapitel 5.2.7 nachzulesen.

Benutzer anlegen und löschen Will man einen Benutzer anlegen, so geschieht dies mit dem entsprechenden Button über der Tabelle. Es öffnet sich eine Dialogbox zum Eingeben des Namens des neuen Benutzers. Danach kann der Benutzer wie gewohnt bearbeitet werden. Etwaige Änderungen an bestehenden Benutzern müssen vor dem Anlegen eines neuen Benutzers nicht gespeichert werden.

Zum Löschen eines Benutzers muss dieser zuvor in der Combobox ausgewählt worden sein. Danach kann er über den entsprechenden Button über der Tabelle gelöscht werden.

Laden Rollen und Benutzerdaten können in einem XML-Format exportiert werden. Um eine solche Datei wieder einzulesen stellt der dritte Reiter des Dialogs eine Lademaske bereit. Die Datei kann dabei auf zwei Wegen geladen werden: Entweder man wählt die lokal gespeicherte Datei mithilfe des Buttons aus, woraufhin der Inhalt der Datei in den Textbereich geladen wird, oder man kopiert den Dateiinhalt von Hand direkt in das Textfeld.

Nach einem Klick auf „Load“ werden die Daten aus der Datei geladen. Eventuell bereits im Dialog vorhandene Daten werden dabei überschrieben. Die geladenen Daten werden jedoch

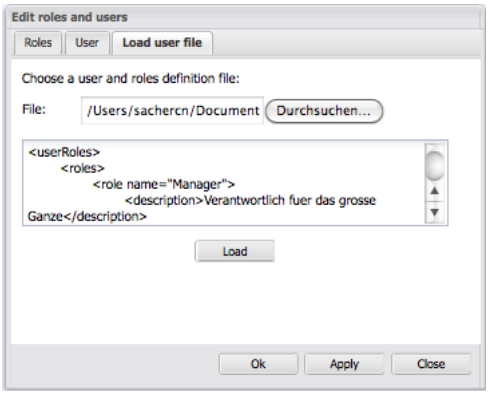

Abbildung 5.8.: GUI zum Laden einer Datei mit Rollen und Benutzerdaten.

erst nach einem Klick auf „OK“ oder „Apply“ dauerhaft übernommen. Wird der Button „Close“ gedrückt, so werden die Änderungen seit dem Öffnen des Dialogs, beziehungsweise seit dem letzten Übernehmen der Änderung, im Editor wieder hergestellt.

5.2.4. Einstellungen

Im Menü für Einstellungen und Hilfsmittel wird unter dem Menüpunkt „Preferences“ ein Dialog angeboten, mit dem einige grundlegende Einstellungen geändert werden können.

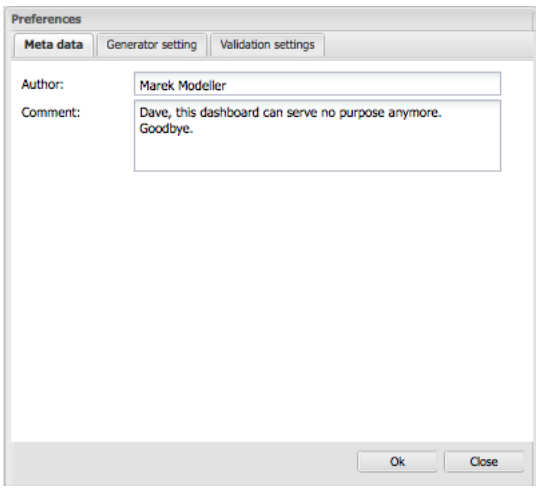

Abbildung 5.9.: GUI zur Angabe von Meta-Daten.

Metadaten Im Reiter „Meta“ können allgemeine Angaben zum Modell gemacht werden. Es kann zum einen ein Autor angegeben werden, zum anderen kann ein Kommentar angegeben werden. Der Inhalt beider Felder wird beim Export der Sichtdefinition als XML-Datei mit in den Meta-Abschnitt der XML-Datei gespeichert.

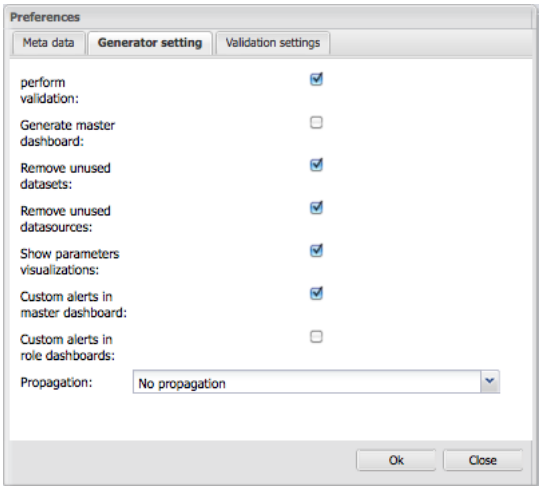

Abbildung 5.10.: GUI der Generierungs-Einstellungen.

Generierung Im Reiter „Generator settings“ können einige Einstellungen getätigt werden, die direkt die Generierung der Sichten beeinflussen. Im Detail können folgende Einstellungen getätigt werden:

- **Perform validation** Wenn gesetzt, wird vor der Generierung von Sichten eine Validierung durchgeführt.
- **Generate master dashboard** Wenn gesetzt, wird neben den Dashboards für die definierten Sichten noch ein Masterdashboard generiert, bei dem alle Elemente sichtbar sind. Die übrigen Einstellungen, zum Beispiel betreffend Custom Alerts, sind jedoch weiterhin wirksam. Das Masterdashboard kann sich also durchaus vom Ausgangsdashboard unterscheiden.
- **Remove unused datasets** Wenn gesetzt, werden ungenutzte Datasets aus den generierten Dashboards gelöscht.
- **Remove unused datsources** Wenn gesetzt, werden ungenutzte Datasources aus den generierten Dashboards gelöscht.
- **Show parameters visualizations** Wenn gesetzt, werden Visualisierungen für Parameter in den generierten Dashboards belassen, ansonsten werden sie entfernt.

- **Custom alerts in master dashboard** Wenn gesetzt, werden Custom Alerts in das Master Dashboard aufgenommen, ansonsten nicht.
- **Custom alerts in role dashboards** Wenn gesetzt, werden Custom Alerts in die Rollendashboards aufgenommen, ansonsten nicht. Die Einstellung wird von eventuell definierten Custom Alert Queries überschrieben.
- **Propagation** Definiert die verwendete Form der Rollenvererbung. Verfügbar sind die Rollenvererbung aufwärts und die Rollenvererbung abwärts mit allen Vorgängern. Näheres siehe Kapitel 4.5.

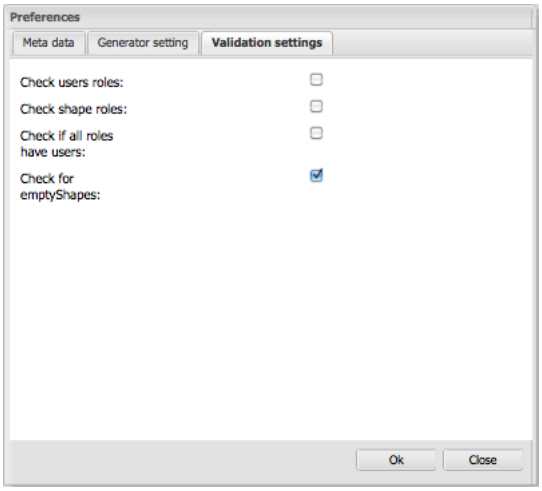

Abbildung 5.11.: GUI der Validierungs-Einstellungen.

Validierung Um Fehler vor der Generierung entdecken zu können, bietet der Prototyp einige Prüfungen an, die häufige Fehler aufspüren. Die Prüfungen können in den Einstellungen einzeln an- und abgeschaltet werden. Folgende Prüfungen sind verfügbar:

- **Check users roles** Prüft, ob alle den Benutzern zugeordneten Rollen auch definiert sind.
- **Check shape Roles** Prüft, ob alle in den Shapes im Editor angegebenen Rollen auch definiert sind.
- **Check if all roles have users** Prüft, ob alle Rollen von mindestens einem Benutzer ausgeübt werden.
- **Check for empty shapes** Prüft, ob für alle Shapes vom Typ „Associated Roles“ mindestens eine Rolle angegeben wurde.

5.2.5. Vorschau

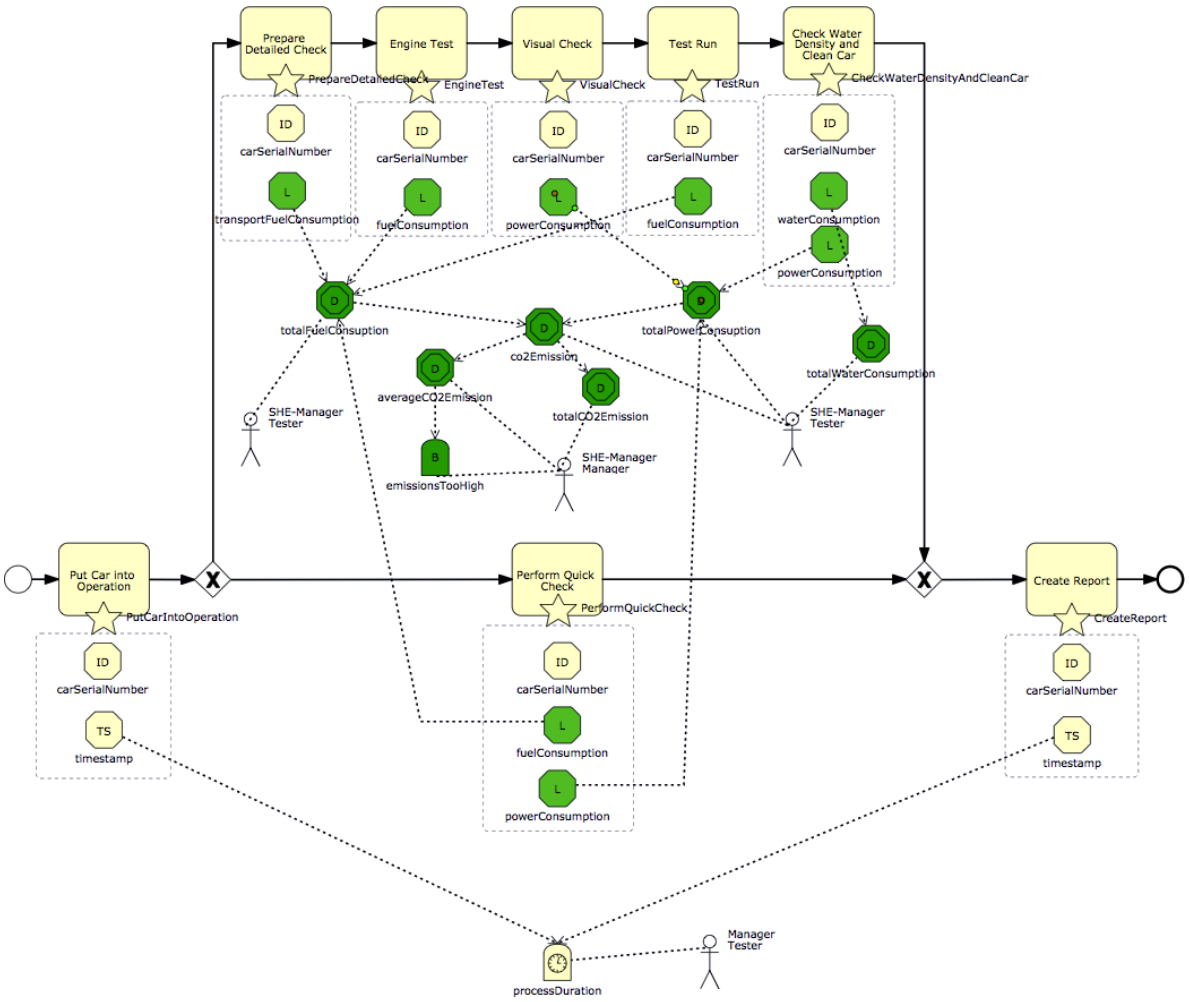

Abbildung 5.12.: Diagramm mit einer Vorschau für die Rolle SHE-Manager.

Um dem Benutzer ein einfacheres Beurteilen der einzelnen Sichten zu ermöglichen, wurde eine Vorschau implementiert. Dabei ist zum einen eine Vorschau nach Rolle möglich, zum anderen eine Vorschau für bestimmte Benutzer. Die Funktionen können über das Menü mit den Funktionen zur Sichtengenerierung aufgerufen werden.

Vorschau für Rolle Wird das Plugin zum Anzeigen einer Vorschau für eine Rolle gestartet, so wird der Benutzer zuerst aufgefordert, eine Rolle auszuwählen (siehe Abbildung 5.13.) Danach werden alle Elemente, die in dieser Rolle enthalten sind, in grün eingefärbt. Dabei wird die Einstellung zur Rollenvererbung (siehe Kapitel 5.2.4) beachtet. Neben den Elementen

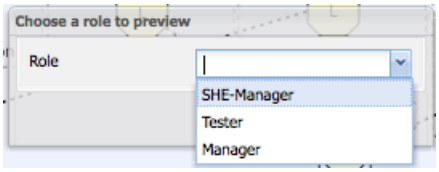

Abbildung 5.13.: GUI zur Auswahl der Rolle, für die eine Vorschau angezeigt werden soll.

mit der gewählten Rolle werden alle Parameter eingefärbt, welche zur Berechnung der sichtbaren Elemente nötig sind. Die Farbe für die Parameter ist ein etwas helleres Grün. Wird das Plugin bei angezeigter Vorschau erneut gestartet, so wird die Vorschau aufgehoben.

Die Vorschau kann für alle im Diagramm verwendeten Rollen angezeigt werden. Dabei ist es unerheblich, ob die entsprechenden Rollen auch in der Rollen- und Benutzerverwaltung definiert wurden.

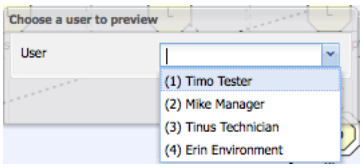

Abbildung 5.14.: GUI zur Auswahl des Benutzers, für den eine Vorschau angezeigt werden soll.

Vorschau für Benutzer Neben der Vorschau für eine bestimmte Rolle ist auch eine Vorschau für einen Benutzer möglich. Dabei wird analog zur Vorschau für Rollen zuerst der Benutzer gewählt (siehe Abbildung 5.14). Die Vorschau färbt dann alle Elemente ein, die in mindestens einer der Rollen eines Benutzers enthalten sind. Dabei ist zu bedenken, dass die sichtbaren Elemente des Benutzers natürlich nicht alle in derselben Sicht enthalten sein müssen. Die Färbung der Parameter erfolgt analog zur Vorschau für Rollen.

Im Gegensatz zur Vorschau für Rollen ist bei der Vorschau für Benutzer die Pflege der Rollen- und Benutzerverwaltung von Bedeutung, da die Benutzer und deren zugeordnete Rollen von dort übernommen werden.

5.2.6. Werkzeuge und Hilfsmittel

Rollen verstecken Im Menü der Einstellungen und Hilfsmittel befindet sich ein Menüpunkt zum Verstecken der Rollen. Dieses Plugin versteckt alle Elemente vom Typ „Associated Roles“ und die von ihnen ausgehenden Kanten. Bei erneutem Aufruf des Plugins werden

die Kanten wieder sichtbar gemacht. Dabei kann es zu einigen von Oryx verursachten Darstellungsfehlern kommen, welche die Funktionalität jedoch nicht beeinträchtigen. Solange die Elemente ausgeblendet sind, werden sie allerdings bei Generierung und Export nicht beachtet.

5.2.7. Validierung

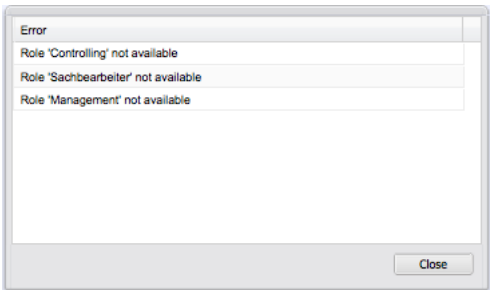

Abbildung 5.15.: Beispielhaftes Ergebnis einer Validierung.

Die Validierung findet entweder automatisch vor der Sichtengenerierung statt, wenn die entsprechende Option in den Einstellungen gesetzt wurde, oder von Hand über den entsprechenden Menüpunkt. Die Befunde werden nach der Validierung in einem Fenster angezeigt (siehe Abbildung 5.15).

5.2.8. Generierung

Die wichtigste Funktion des Prototyps ist die Sichtengenerierung. Der Prototyp unterscheidet zwischen zwei Arten der Sichtgenerierung. Zum einen kann eine Sicht für ein bereits deployes Dashboard aus aPro generiert werden, zum anderen kann eine VisML-Datei in Dateien für jede Sicht zerlegt werden.

Beide Arten der Sichtgenerierung können ihre Sichtdefinition aus zwei unterschiedlichen Quellen beziehen. Werden Dateien mit Sichtdefinition oder Rollen und Benutzerdaten angegeben, dann werden diese verwendet. Wird eine oder beide der Dateien nicht angegeben, so wird stattdessen an der Stelle der Inhalt des Editors verwendet.

Behandlung von Parametern Für Parameter können keine Rollen vergeben werden. Ihre Sichtbarkeit ergibt sich implizit aus der Sichtbarkeit der KPIs und Ziele. Ein Parameter ist sichtbar für eine Rolle r, genau dann, wenn es eine KPI oder ein Ziel gibt, welches den Parameter direkt oder Iindirekt für seine Berechnung benötigt und für die Rolle r sichtbar ist.

Abbildung von Elementnamen auf Visualisierungsnamen In aPro sind die Namen der Visualisierungen und Alerts nicht identisch mit den Namen der KPIs, Ziele oder Parameter, denen sie zugeordnet sind. Der Prototyp geht bei der Generierung von Sichten davon aus, dass das Namensschema des aPro-Prototyps Verwendung gefunden hat, auch wenn die VisML-Datei nicht von aPro generiert wurde. Das Schema sieht wie folgt aus:

```
Visualisierungsname = KPI_NAME
```

Listing 5.1: Benamungschema für Visualisierungen von KPIs und Zielen.

```
Visualisierungsname = TASK_NAME + "___" + PARAMETER_NAME
```

Listing 5.2: Benamungschema für Visualisierungen von Parametern.

```
Visualisierungsname = ZIEL_NAME + "Alert"
```

Listing 5.3: Benamungschema für Alerts.

Custom Alerts

Im Prototyp wurde eine Funktion eingebaut um pro Rolle einen Query in Form eines Regulären Ausdrucks vergeben zu können. Im Rollendashboard wird das Dataset des Alerts dann um diesen Query erweitert. So können die von bestimmten Dashboards angezeigten Custom Alerts auf relevante Bereiche eingegrenzt werden. Neben den Querys gibt es in den Einstellungen noch globale Schalter, welche Custom Alerts in Rollendashboard an- und abschalten. Selbiges gilt für das Masterdashboard. Diese Schalter sind den Querys jedoch untergeordnet. Das heißt, dass ein definierter Query angewendet wird, auch wenn Alerts abgeschaltet sind.

Zum jetzigen Zeitpunkt werden die hier vorgestellten Querys für Custom Alerts zwar bei der Generierung der Dashboards vom Prototyp korrekt in das Dashboard eingebaut, das Dashboard selbst implementiert diese Funktion jedoch noch nicht. Es werden also im Dashboard weiterhin alle Custom Alerts angezeigt, wenn ein Query definiert ist.

Rollen und Benutzer im generierten Dashboard

Zum Zeitpunkt der Arbeit ist sich das verwendete Dashboard der Rollen und Benutzer noch nicht bewusst. Um aber zukünftige Erweiterungen zu ermöglichen, werden in der

generierten VisML-Datei im Meta-Teil die zum Dashboard gehörende Rolle und deren Benutzer vermerkt, sofern die Benutzer im Prototyp gepflegt wurden.

Sichten für VisML generieren

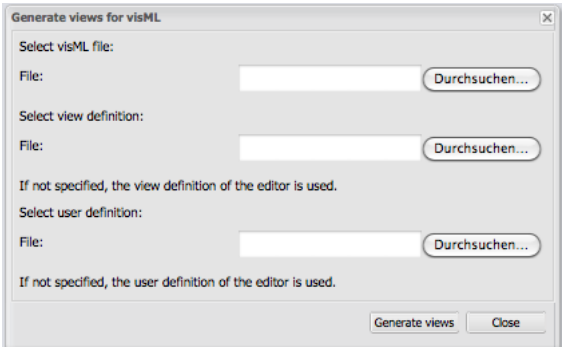

Abbildung 5.16.: GUI zum Generieren von Sichten aus einer VisML-Datei.

Das Plugin unter dem Menüpunkt „Views for VisML“ im Menü der Plugins für die Sichtengenerierung erlaubt das Zerlegen einer VisML-Datei in je eine VisML-Datei für jede definierte Sicht. Das Plugin erwartet drei Dateien. Zum einen die VisML-Datei, für welche die Sichten generiert werden sollen, zum anderen die zwei Dateien mit den Sichtdefinitionen und den Rollen und Benutzerdaten. Die beiden letzteren Dateien sind optional. Werden sie nicht angegeben, so wird das Modell, beziehungsweiße die Rollen und Benutzerdaten aus dem Editor verwendet.

Die Generierung erstellt mit Hilfe der Sichtdefinitionen eine VisML-Datei pro generierter Sicht und öffnet diese in einem neuen Browserfenster. Die Dateien können dann vom Benutzer gespeichert und weiterverwendet werden.

Sichten für deployes Dashboard generieren

Das Plugin unter dem Menüpunkt „Views for deployed dashboard“ im Menü der Plugins für die Sichtengenerierung erlaubt nach Angabe eines Namens das Erstellen von Dashboards für definierte Sichten. Im ersten Feld ist der Name des von aPro generierten und deployten Dashboards anzugeben. Das Feld ist ein Pflichtfeld und muss daher gefüllt werden. Außerdem erwartet das Plugin, analog zum Erstellen von Sichten für VisML-Dateien, zwei Dateien mit den Sichtdefinitionen und den Rollen und Benutzerdaten. Die beiden Dateien sind optional. Werden sie nicht angegeben, so wird das Modell, beziehungsweiße die Rollen und Benutzerdaten aus dem Editor, verwendet.

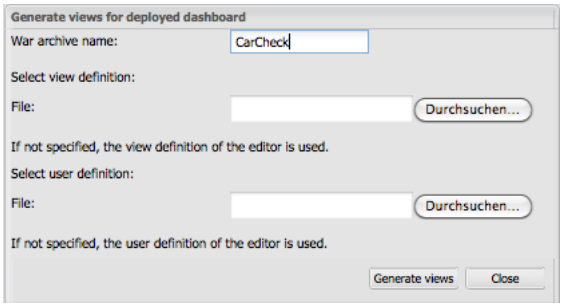

Abbildung 5.17.: GUI zum Generieren von Sichten für ein deployes Dashboard.

Die Generierung holt dann automatisch die VisML-Datei des deployten Dashboards vom Server. Aus dieser erstellt sie je eine VisML-Datei pro Sicht und speichert diese wieder auf dem Server. Außerdem legt sie pro Sicht eine HTML-Datei zur Anzeige des HTML-Dashboards für die Sicht an. Der Name der Datei ist SICHTNAME_index.html für rollenbasierte Sichten und _index.html für das Master-Dashboard. Der Rest der URL kann unverändert von der URL des ursprünglichen Dashboards übernommen werden.

6. Evaluation an zwei Beispielen

Im Folgenden wird das entwickelte Konzept mittels des Prototyps anhand zweier Beispiele überprüft. Dazu wird der vollständige Arbeitsablauf von der Modellierung des Prozesses bis zu den fertigen Rollendashboards durchlaufen. Als Beispiel wurden ein Prozess aus der Herstellung eines Automobils und ein Prozess aus der Versicherungsbranche gewählt. Der erste Prozess war bereits Grundlage für zahlreiche Illustrationen in anderen Teilen dieser Arbeit und wurde aus [Now+11] entnommen.

6.1. Beispiel Automobiltest

In dem Beispielprozess wird ein soeben fertiggestelltes Auto einigen Prüfungen unterzogen. Meist wird das Auto dabei einer schnellen Prüfung anhand von Checklisten unterzogen. In einigen Fällen wird es jedoch einer detaillierten Prüfung unterzogen. In beiden Fällen wird nach den Prüfungen ein Bericht über die Prüfungen und deren Ergebnisse erstellt.

Um die Umweltauswirkungen der Prüfungen zu erfassen sollen entsprechende Messungen durchgeführt werden und in einem Dashboard visualisiert werden. Das Dashboards wird von Inhabern drei unterschiedlicher Rollen betrachtet, welche unterschiedliche Anforderungen an die dargestellten Daten haben.

Für den Prozess sollen hauptsächlich Messwerte zu den Umweltauswirkungen erfasst werden. Dazu gehören summierte Verbrauchswerte sowie einige von diesen abgeleitete Größen. Daneben soll noch die Durchlaufzeit des Prozesses gemessen werden. Damit ist es möglich, eine Minimierung der Umwelteinwirkungen gegen möglicherweise längere Durchlaufzeiten abzuwägen.

Auf die gesammelten Daten bekommen drei Rollen Zugriff. Der zuständige Tester kann die durch den Test verursachten Umweltauswirkungen in Form der Verbrauchswerte überwachen können. Der Umweltbeauftragte ist prinzipiell an denselben Daten interessiert. Für ihn ist jedoch auch die Umrechnung dieser Vergleiche in CO_2-Äquivalente von Interesse, da er so eine stoffunabhängige Vergleichsgröße hat. Als dritte Rolle wird der Manager über Durchlaufzeiten und verursachte CO_2-Äquivalente informiert. Sollte eine der Größen einen Grenzwert überschreiten, so wird er mit einem Alert informiert.

Die folgenden Abschnitte decken alle nötigen Arbeitsschritte von der Modellierung bis zum Betrachten des Dashboards ab. Jedoch wird dabei nur auf die für diese Arbeit relevanten Dinge eingegangen. Grundkenntnisse in der Bedienung von Oryx und teilweise des aPro-Prototyps werden vorausgesetzt. Lediglich dort, wo Besonderheiten zu beachten sind, wird auch auf Funktionen fremder Systeme eingegangen.

6.1.1. Prozessmodellierung

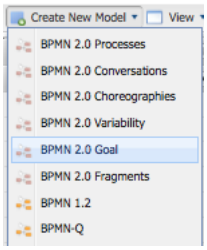

Abbildung 6.1.: Erstellen eines Ziel-Modells.

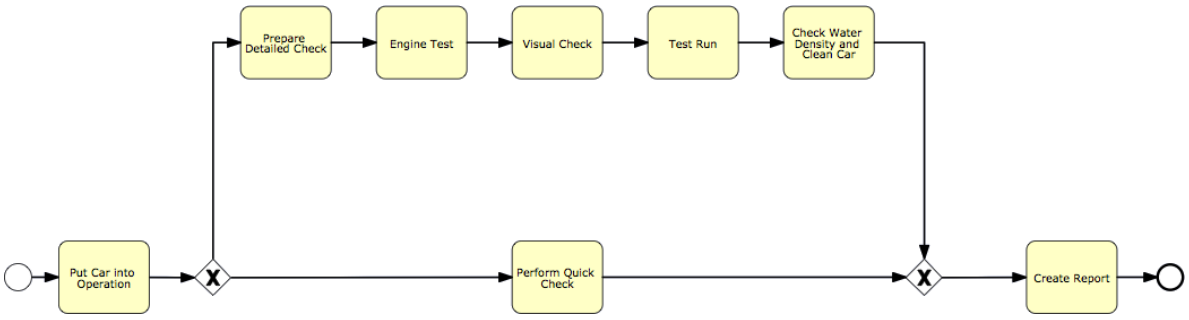

Abbildung 6.2.: Fertiges BPMN-Modell.

Zu Beginn muss der Prozess modelliert werden, wenn der Prozess nicht bereits als Modell vorliegt. Dazu wird ein Modell vom Typ **BPMN 2.0 Goal** erstellt (siehe Abbildung 6.1). Wenn das geschehen ist, kann im Editor das BPMN-Modell erstellt werden. Dabei werden dem Modell vom Prototyp keine besonderen Beschränkungen auferlegt. Allerdings sind der Modellierung von KPIs einige Grenzen gesetzt. Genaueres dazu findet sich in [KK12a]. Das fertige Dashboard zeigt Abbildung 6.2.

6.1.2. Kennwertmodellierung

Der nächste Schritt des Beispiels ist das Erweitern des Modells um KPIs und Ziele. Dazu können alle von aPro gelieferten Elemente verwendet werden. Es müssen lediglich die

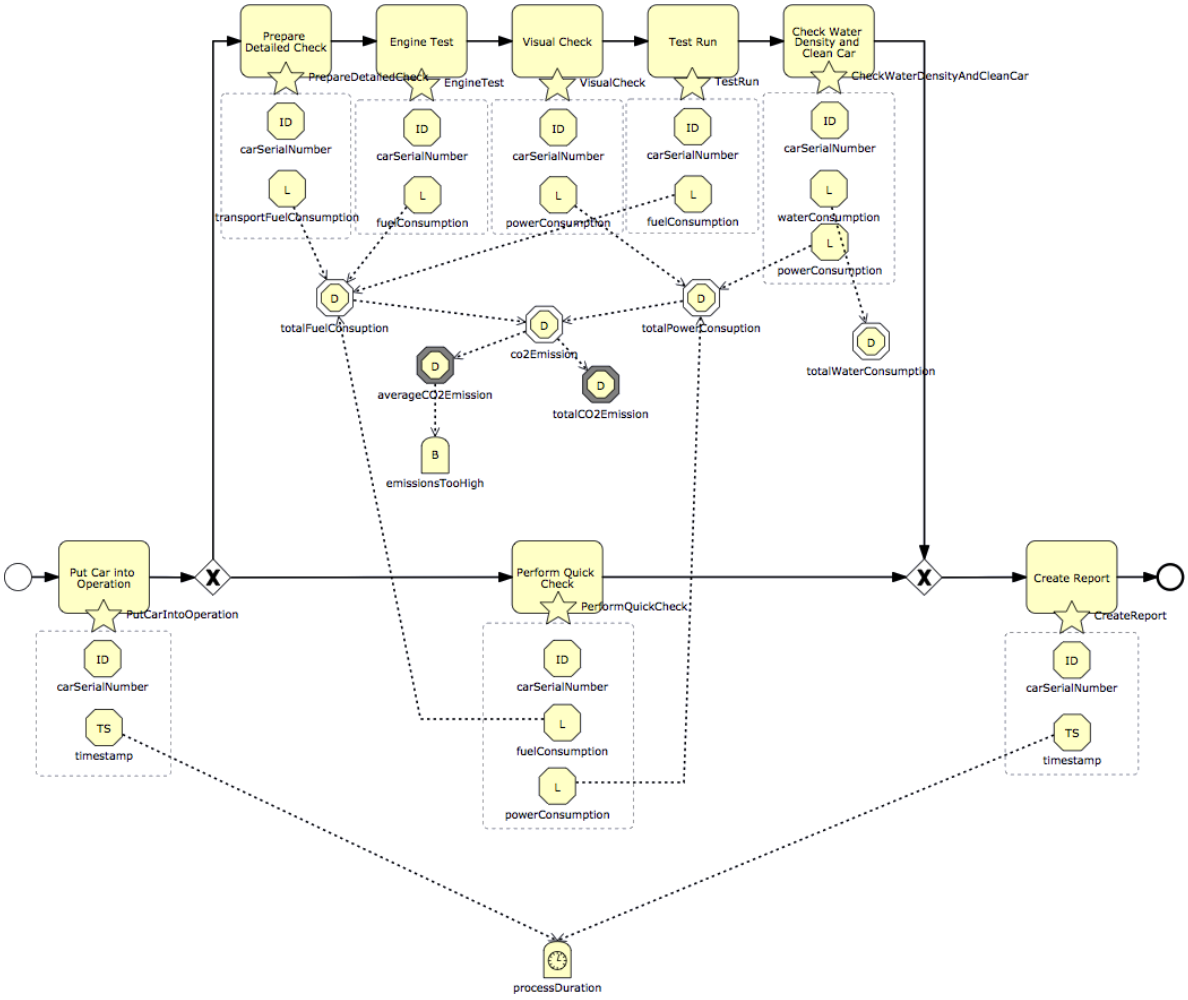

Abbildung 6.3.: Mit aPro erweitertes BPMN-Modell.

üblichen Beschränkungen von aPro beachtet werden. Der Prototyp selbst bedingt keine weiteren Beschränkungen.

In diesem Prozess wird bei allen Tasks, welche Ressourcen in Form von Treibstoff, Strom, oder Wasser verbrauchen, der Verbrauch dieser Stoffe gemessen. Aus diesen wird dann, nach Ressource getrennt, die Summe gebildet. Aus dem Verbrauch an Treibstoff und Strom wird in einem zweiten Schritt die Menge an CO_2-Emissionen errechnet, die zu dem Verbrauch äquivalent ist. Von diesem Kennwert werden dann noch die Summe und der Durchschnitt der letzen 100 Instanzen gebildet. Der Durchschnitt wird außerdem mit einem Ziel überwacht und soll einen definierten Grenzwert nicht übersteigen. Ein weiteres Ziel überwacht die Durchführungszeit einer Instanz des Prozesses, welche eine definierte Dauer ebenfalls nicht überschreiten soll. Das fertige Diagramm zeigt Abbildung 6.3.

6.1.3. Sichtenmodellierung

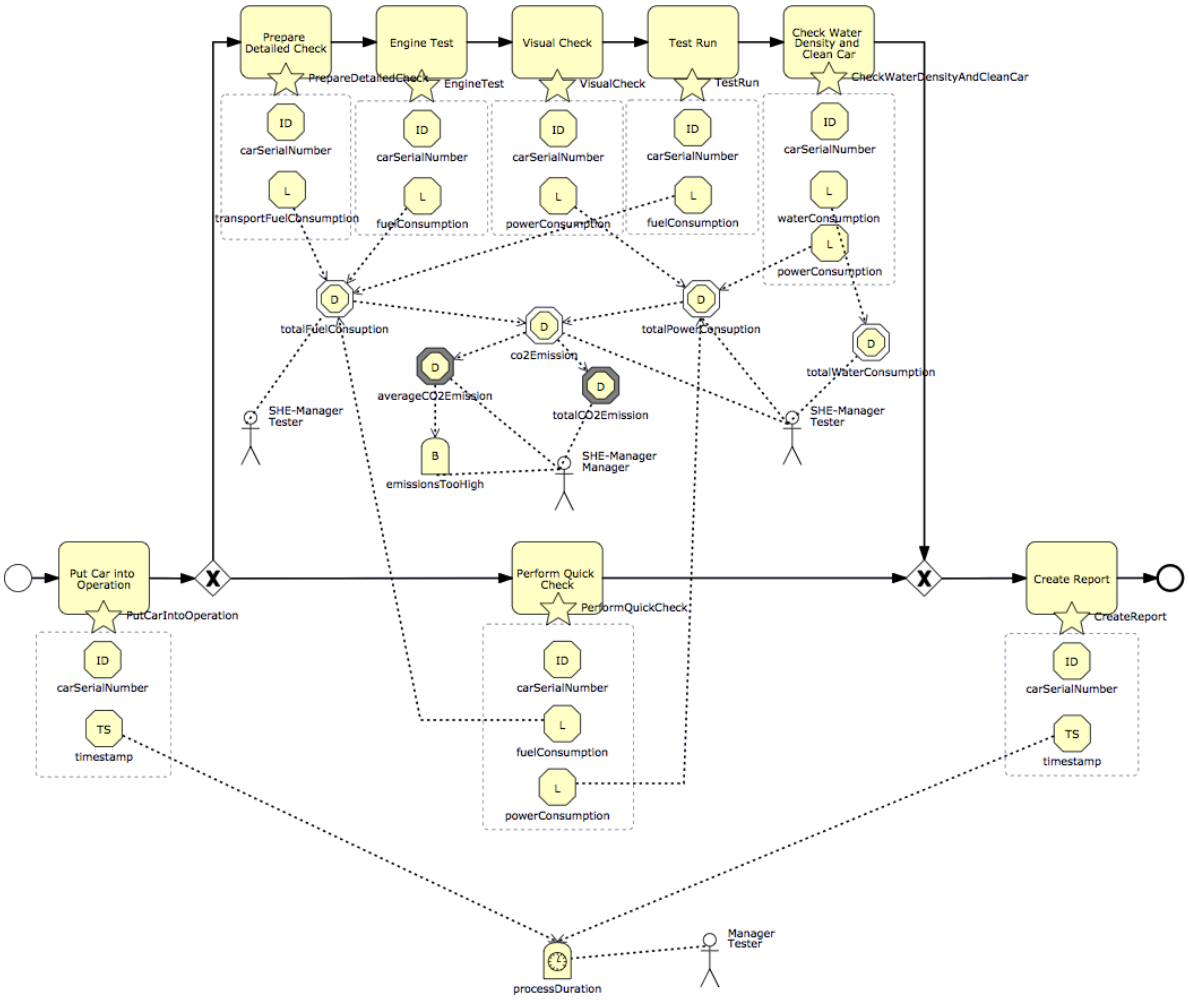

Abbildung 6.4.: Fertiges Modell mit Sichtbarkeitsdefinitionen.

Im letzten Modellierungsschritt werden die Sichten hinzugefügt. Dafür wird das „Associated Roles“-Element aus der Gruppe „Views“ verwendet. Mit ungerichteten Kanten aus der Gruppe „Connecting Objects“ werden die Sichtelemente mit den KPIs und Zielen verbunden. Die Rollen werden in das Textfeld der Sichtelemente geschrieben und durch einfache Zeilenumbrüche getrennt.

Für die Rolle Tester sollen die summierten Verbrauchswerte und die CO_2-Äquivalente eines Prozessdurchlaufs sichtbar sein. Diese Daten ermöglichen es dem Tester die Auswirkungen seines Tuns direkt zu beurteilen und insbesondere seine Arbeitsaufgaben selbstständig im Hinblick auf die Umweltauswirkungen zu optimieren. Außerdem soll der Tester informiert

werden, wenn der Prozess die maximal zulässige Durchlaufzeit überschreitet, um in diesem Fall Gegenmaßnahmen treffen zu können. Der Umweltbeauftragte des Unternehmens soll ebenfalls die summierten Verbrauchswerte und deren CO_2-Äquivalente sehen können. Zusätzlich soll er jedoch auch deren Summe und Durchschnitt betrachten können. Wird die maximal zulässige durchschnittliche Emission überschritten, so soll er benachrichtigt werden. Als dritte Rolle kann der Manager ebenfalls die Summe und den Durchschnitt der CO_2-Äqivalente sehen und wird ebenfalls über eine Überschreitung benachrichtigt. Zusätzlich wird er auch über eine Überschreitung der maximalen Durchlaufzeit benachrichtigt. Das fertige Modell zeigt Abbildung 6.4.

6.1.4. Dashboardgenerierung

Aus dem Modell wird im nächsten Schritt ein Dashboard erzeugt. Dazu wird das Plugin „Deploy the monitoring WAR archive to a remote or local destination“ gestartet. Hier muss ein eindeutiger Name für das Dashboard vergeben werden. Im Reiter „Webform“ wird die Option „Generate webform“ aktiviert und im Reiter „Dashboard“ die Option „Generate Dashboard“. Mithilfe des Webforms können Testdaten von Hand eingegeben werden, die dann im Dashboard visualisiert werden. Das fertige Dashboard mit einigen Daten zeigt Abbildung 6.5.

Abbildung 6.5.: Ausschnitt des Dashboards für den Automobiltest vor der Zerlegung in einzelne Sichten.

Das deployte Dashboard muss nun in noch in die definierten Sichten zerlegt werden. Hierzu nehmen wir zuerst einige Einstellungen vor. Eine Benutzerdatei soll bei der Generierung nicht zum Einsatz kommen. Es können also alle Validierungen für Benutzer abgeschaltet werden. Außerdem wird kein Master-Dashboard benötigt. Die genauen Einstellungen zeigt Abbildung 6.6.

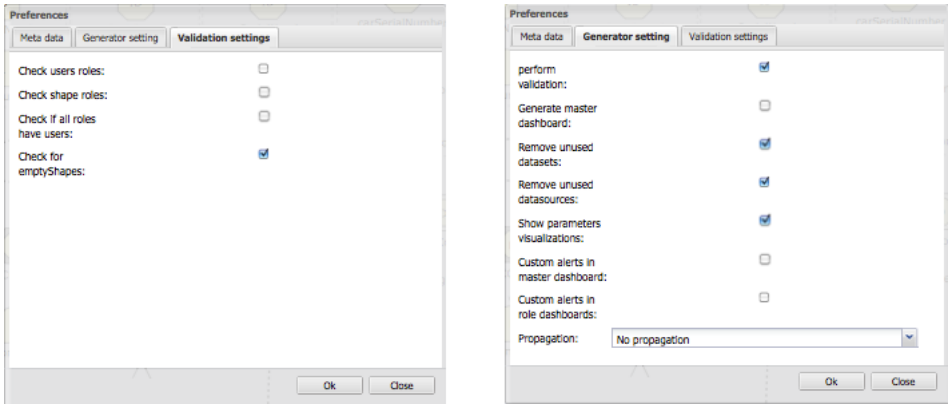

Abbildung 6.6.: Generator- und Validatoreinstellungen.

Danach muss das Plugin „Views for deployed Dashboard“ gestartet und der Name des Dashboards angegeben werden. Weitere Einstellungen sind nicht notwendig, da die Inhalte des Editors verwendet werden sollen. Nach Bestätigung des Fensters wird das Dashboard in die definierten Sichten zerlegt. Das Ergebnis sind drei Dashboards, welche unter den URLs *dashboard/Manager_index.html*, *dashboard/Tester_index.html* und *dashboard/SHE-Manager_index.html* abgelegt werden.

Abbildung 6.7 zeigt beispielhaft das Dashboard für die Rolle Manager. Im Dashboard wurden bereits Daten über das Webform eingegeben. Neben den einzelnen Schaubildern wird außerdem ein für den Manager sichtbarer Alert aufgrund einer überschrittenen maximalen Laufzeit einer Prozessinstanz angezeigt.

6.2. Beispiel Versicherung

Das zweite Beispiel entstammt der Versicherungsbranche. Das Beispiel ist fiktiv, basiert jedoch auf realen Prozessen. Es beschreibt einen Prozess zur Prüfung einer eingereichten Rechnung eines Schadensfalles. Zuerst wird die Rechnung automatisiert erfasst und geprüft. War die Prüfung negativ, so wird die Rechnung einem Mitarbeiter zur Prüfung vorgelegt und in der Folge entweder endgültig abgelehnt oder nachträglich für vertretbar befunden. Wurde die Rechnung nicht abgelehnt, so erfolgt nun, wenn möglich, eine automatische Nachkalkulation. Diese wird ebenfalls von einem Mitarbeiter geprüft und gegebenenfalls

Abbildung 6.7.: Ausschnitt des Dashboards für den Automobiltest für die Rolle Manager.

noch einmal von einem Mitarbeiter überarbeitet. Das BPMN-Modell des Prozesses zeigt Abbildung 6.8.

Für den Prozess sollen Messwerte aus unterschiedlichen Bereichen erfasst werden. Zum einen sollen Daten zur Rechnungssumme, ihrer automatisierten Einstufung und der in der Nachkalkulation erreichten Kostenersparnis gesammelt werden. Zum anderen sollen jedoch auch Daten zum verursachten Arbeitsaufwand des Prozesses ausgewertet werden.

Diese Daten sind für insgesamt drei Rollen von Interesse. Der Sachbearbeiter hat Zugriff auf die Rechnungssumme und die Einstufung. Außerdem kann er die errechnete Einsparung betrachten. Der Controller sieht jeweils den Durchschnitt der letzten Instanzen für diese Messwerte. Außerdem hat er Zugriff auf die erfassten Daten zum Arbeitsaufwand. Als dritte Rolle ist das Management hauptsächlich an der durchschnittlichen Einsparung interessiert.

6.2.1. Kennwertmodellierung

Das Kennwertmodell dieses Beispiels ist umfangreicher als das des vorherigen Beispiels. Beim Erfassen der Rechnung wird die Rechnungssumme gemessen sowie deren Durchschnitt über die letzten 100 Instanzen. Danach wird die Rechnung auf Auffälligkeiten geprüft und entweder als korrekt (Wert: ok) eingestuft oder es sind Auffälligkeiten vorhanden (Wert: negativ). In letzterem Fall wird das Ergebnis noch einmal von einem Mitarbeiter geprüft und die Einstufung entweder bestätigt oder die Rechnung nachträglich als korrekt eingestuft.

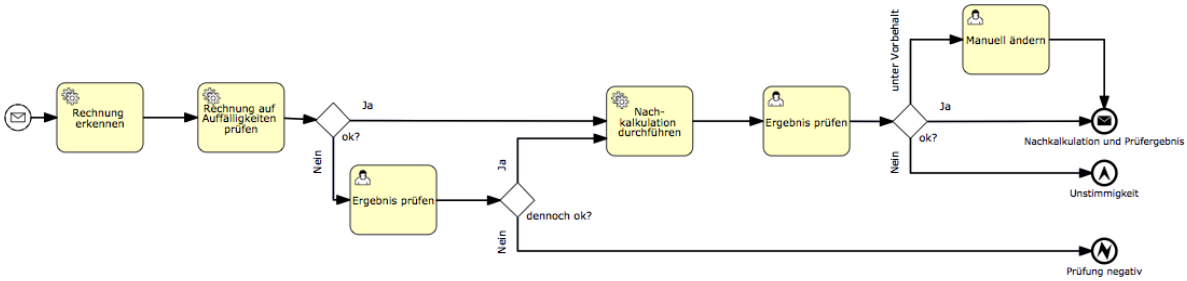

Abbildung 6.8.: Beispiel aus der Versicherungsbranche.

Wurde die Rechnung als korrekt eingestuft, so wird eine manuelle Nachkalkulation durchgeführt um zu sehen, ob die Rechnung zu hoch angesetzt wurde. Bei dieser Operation wird die Einsparung gemessen, aus welcher in einem zweiten Schritt noch die durchschnittliche Einsparung der letzten 100 Durchführungen errechnet wird. Das Ergebnis der Nachkalkulation wird anschließend von einem Mitarbeiter geprüft. Wurden in der Prüfung keine Auffälligkeiten gefunden, so wird der Prozess beendet. Bei kleineren Fehlern wird eine manuelle Nachkalkulation durchgeführt und der Prozess danach beendet. Wird bei der Nachkalkulation jedoch eine große Differenz zwischen Wert der Leistung und der Rechnungssumme festgestellt, so wird der Prozess mit dem Ergebnis „Unstimmigkeit“ beendet.

Aus der Dauer aller von Mitarbeitern ausgeführten Aufgaben wird dann die Arbeitszeit berechnet. Um aus dem Ruder laufende Instanzen zu erkennen, wird die Arbeitszeit mit einem Ziel überwacht. Aus Arbeitszeit werden wiederum die Personalkosten berechnet, welche die Prozessinstanz verursacht hat. Zusammen mit der Einsparung durch die Nachkalkulation wird außerdem der Quotient der Einsparung pro Arbeitszeit und deren Durchschnitt über die letzten 100 Instanzen ermittelt. Zuletzt wird aus der Arbeitszeit ebenfalls noch ein Durchschnitt über 100 Instanzen berechnet und mit einem Ziel überwacht. Der Grenzwert für den Durchschnitt liegt dabei deutlich unter dem Grenzwert für die maximale Arbeitszeit einer Prozessinstanz, da zu erwarten ist, dass die meisten Instanzen reibungslos durchlaufen und nur in wenigen Instanzen Unstimmigkeiten gefunden werden, die einer größeren manuellen Intervention bedürfen.

Als letzter Kennwert wird noch die Durchlaufdauer des gesamten Prozesses berechnet. Hierfür befinden sich Messpunkte an sämtlichen Pfaden zu den End Events um für jeden denkbaren Ausgang des Prozesses einen korrekten Wert berechnen zu können. Auch die Durchlaufdauer des Prozesses wird wiederum von einem Ziel überwacht. Das fertige Modell zeigt Abbildung 6.9.

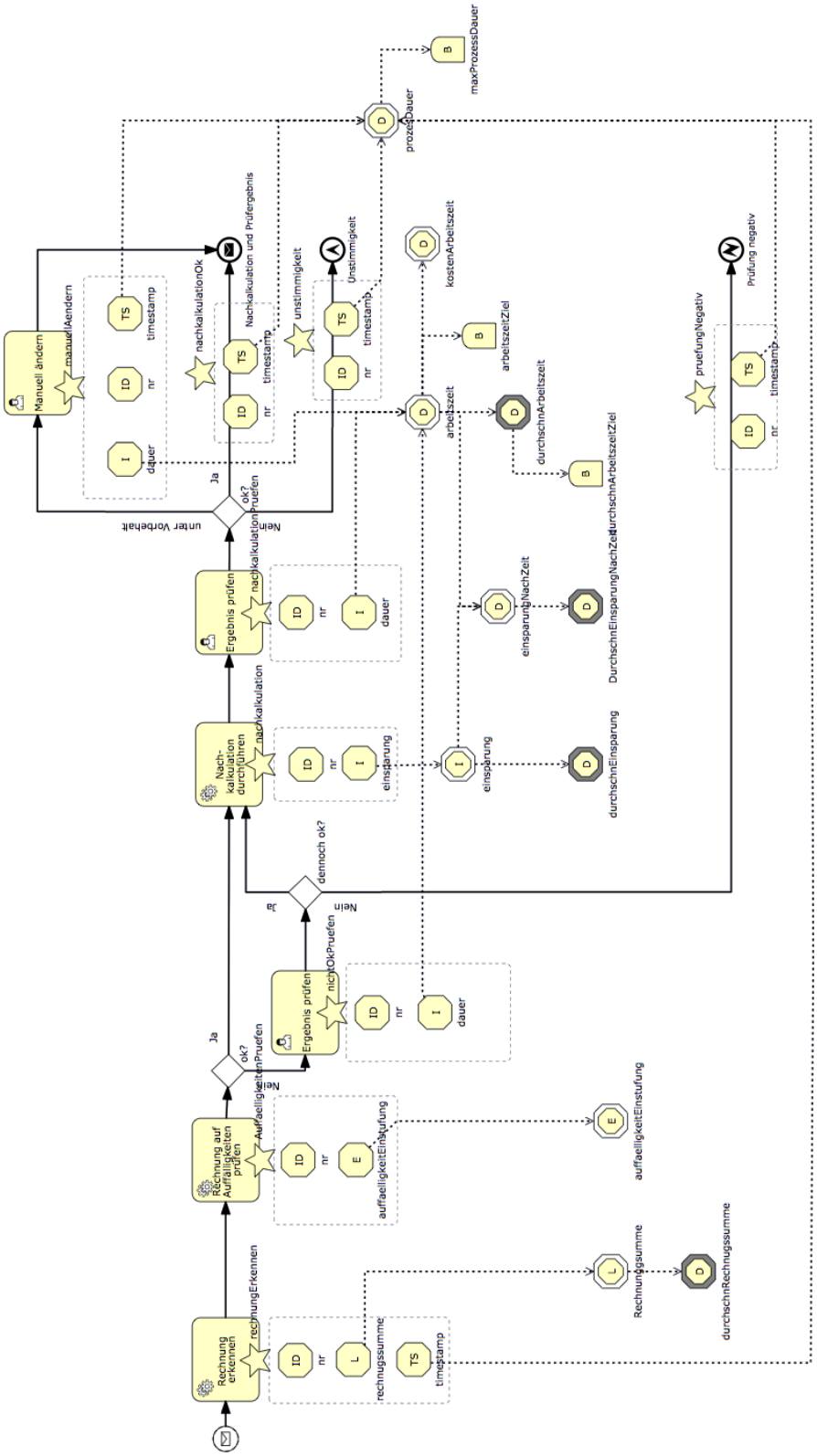

Abbildung 6.9.: Beispiel aus der Versicherungsbranche mit Kennwerten.

6.2.2. Sichtenmodellierung

Wie schon im letzten Prozess ist das Management in diesem Prozess hauptsächlich an aggregierten Daten interessiert. Daher bekommt es Zugriff auf die Ziele der Durchlaufzeit des Prozesses und der durchschnittlichen Arbeitszeit. Daneben bekommt das Management noch Zugriff auf die durchschnittliche Einsparung und die durchschnittliche Einsparung pro Zeiteinheit Arbeitszeit.

Das Controlling bekommt ebenso Zugriff auf die meisten aggregierten Kennwerte. Außerdem bekommt es Zugriff auf die Kennwerte zur Arbeitszeit und zur Prozessdauer. Dadurch ist ein guter Überblick über die verwendete Arbeitszeit und deren Wirtschaftlichkeit gegeben.

Der Sachbearbeiter wiederum bekommt nur Zugriff auf die direkt mit den Rechnungen in Beziehung stehenden Kennwerte. Die Kennwerte zur Arbeitszeit und die aggregierten Kennwerte soll er nicht betrachten können. Das fertige Modell zeigt Abbildung 6.10.

Bei diesem Beispiel zeigt es sich, dass es sehr sinnvoll ist, „Associated Roles“-Elemente für mehrere KPIs zu verwenden, wenn mehrere räumlich nah beieinander liegende KPIs dieselbe Rolle zugeordnet bekommen sollen. Abbildung 6.11 zeigt den Beispielprozess unter Verwendung von je einem „Associated Roles“-Element pro KPI. Speziell die rechte Hälfte des Beispiels wird dadurch deutlich unübersichtlicher.

Auch zeigt sich hier, dass der Nutzen von Vererbung stark von den Regelungen eines Unternehmens betreffend der Zugänglichkeit von Informationen abhängig ist. So könnten in diesem Beispiel einige Kanten eingespart werden, wenn man Rollenvererbung abwärts zulässt. Abbildung 6.12 zeigt, wie Rollenvererbung die Sichtendefinition mit einer kleinen Änderung der Sichtbarkeiten vereinfachen kann. Dabei ist jedoch unbedingt zu beachten, dass dies auch an einigen Stellen unerwünschte Nebeneffekte hat. Abbildung 6.13 zeigt, wie dadurch für die Rolle „Sachbearbeiter“ Elemente als sichtbar gelten, die im ursprünglichen Diagramm für diese Rolle nicht sichtbar waren.

6.2.3. Dashboardgenerierung

Analog zum ersten Beispiel wird auch hier zuerst das Dashboard generiert und anschließend in seine Sichten zerlegt. Im Unterschied zum ersten Beispiel wird hier jedoch der Umweg über das XML-Format genommen. Vor der Zerlegung des Dashboards in die einzelnen Sichten wird daher zunächst die Sichtendefinition in XML exportiert und lokal gespeichert. Bei der Sichtengenerierung wird dann die entsprechende Datei im Dialogfenster angegeben. Die Sichten werden danach wie gewohnt generiert. Das fertige Dashboard für die Rolle Sachbearbeiter zeigt Abbildung 6.14.

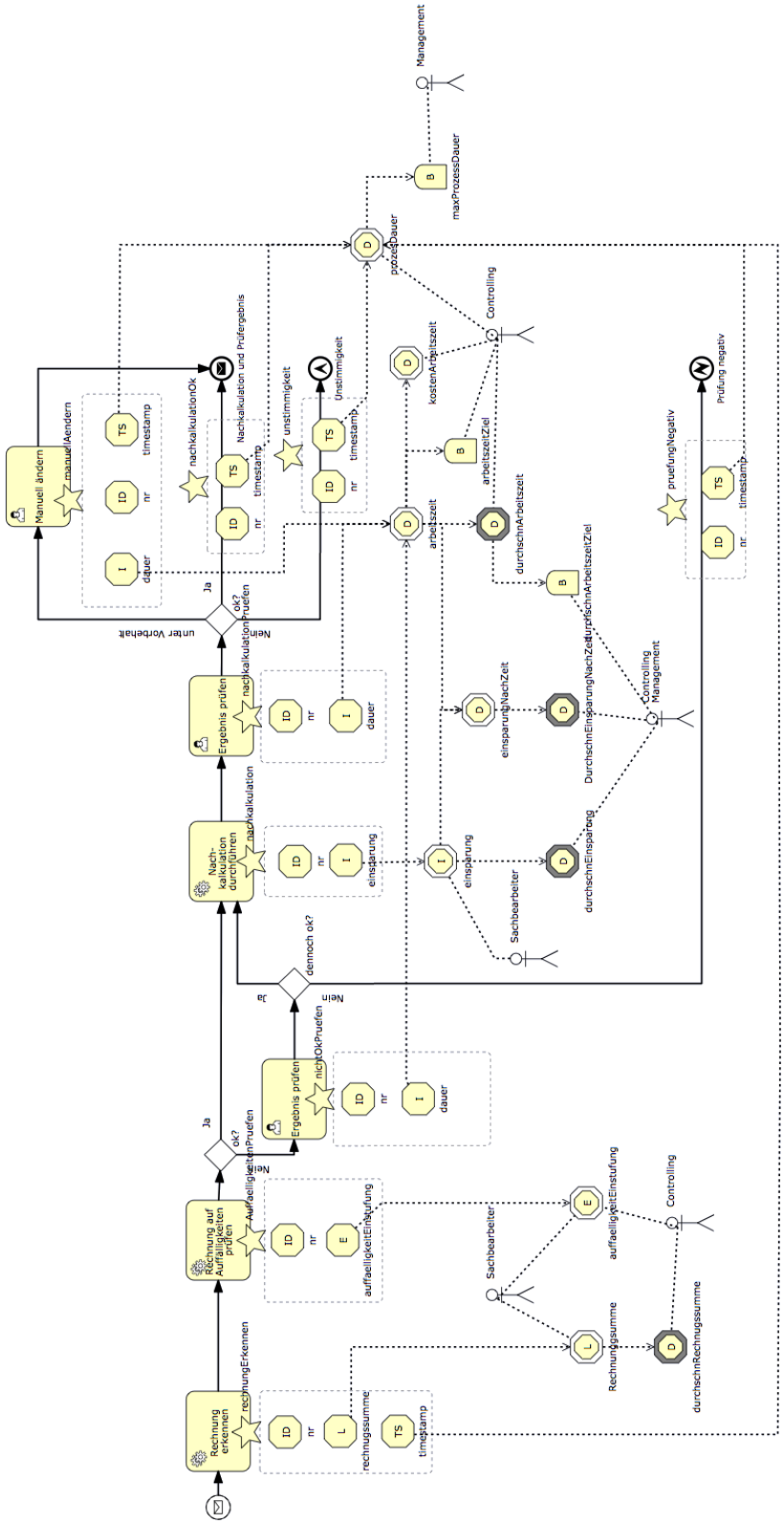

Abbildung 6.10.: Beispiel aus der Versicherungsbranche mit Kennwerten und Sichten.

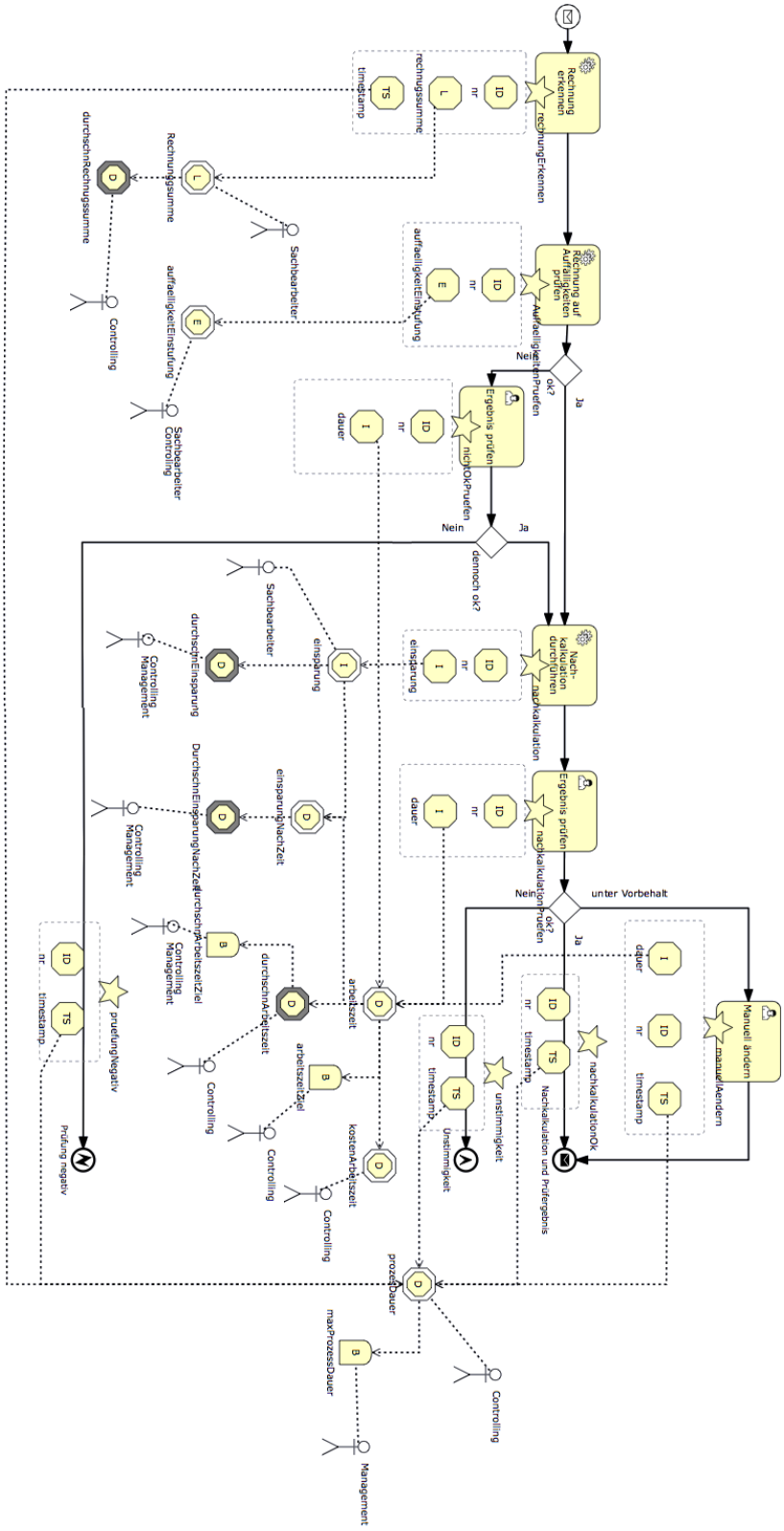

Abbildung 6.11.: Beispiel aus der Versicherungsbranche mit einzeln definierten Sichtbarkeiten.

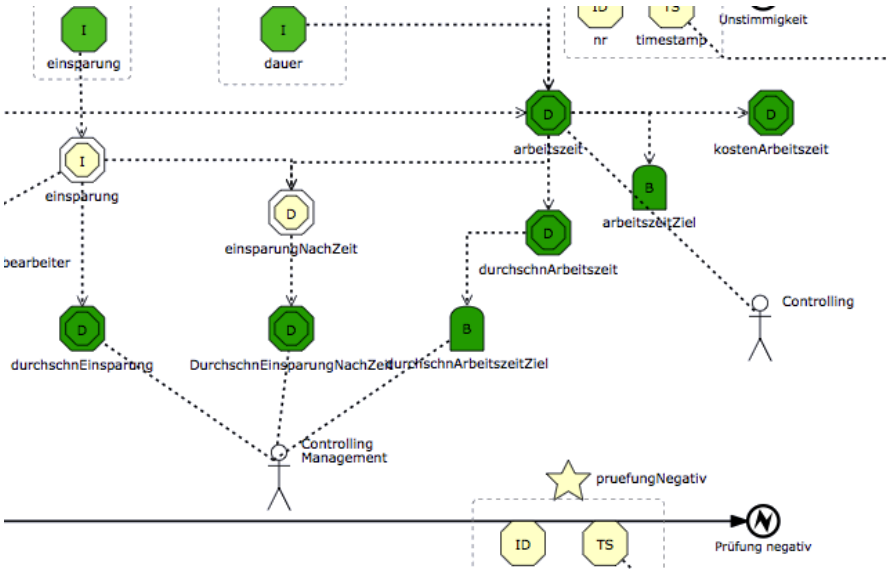

Abbildung 6.12.: Vereinfachung der Sichtendefinition der Rolle „Controlling“ mit abwärts gerichteter Rollenvererbung.

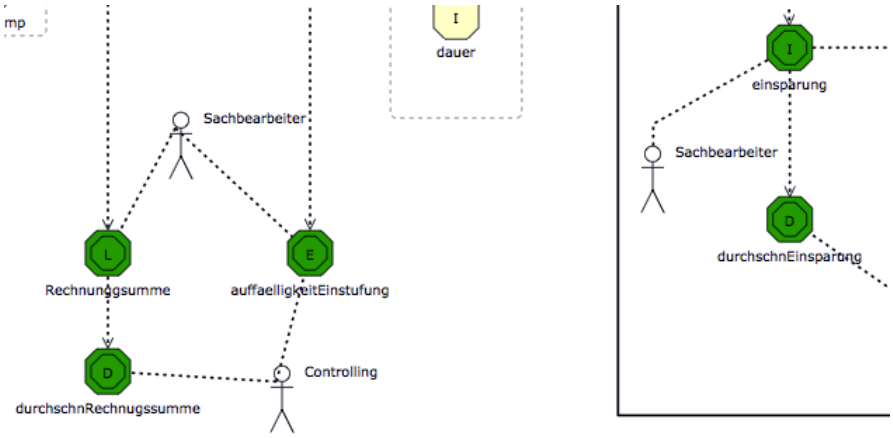

Abbildung 6.13.: Unerwünschte Veränderung der Sichtbarkeiten der Rolle „Sachbearbeiter“ durch Rollenvererbung.

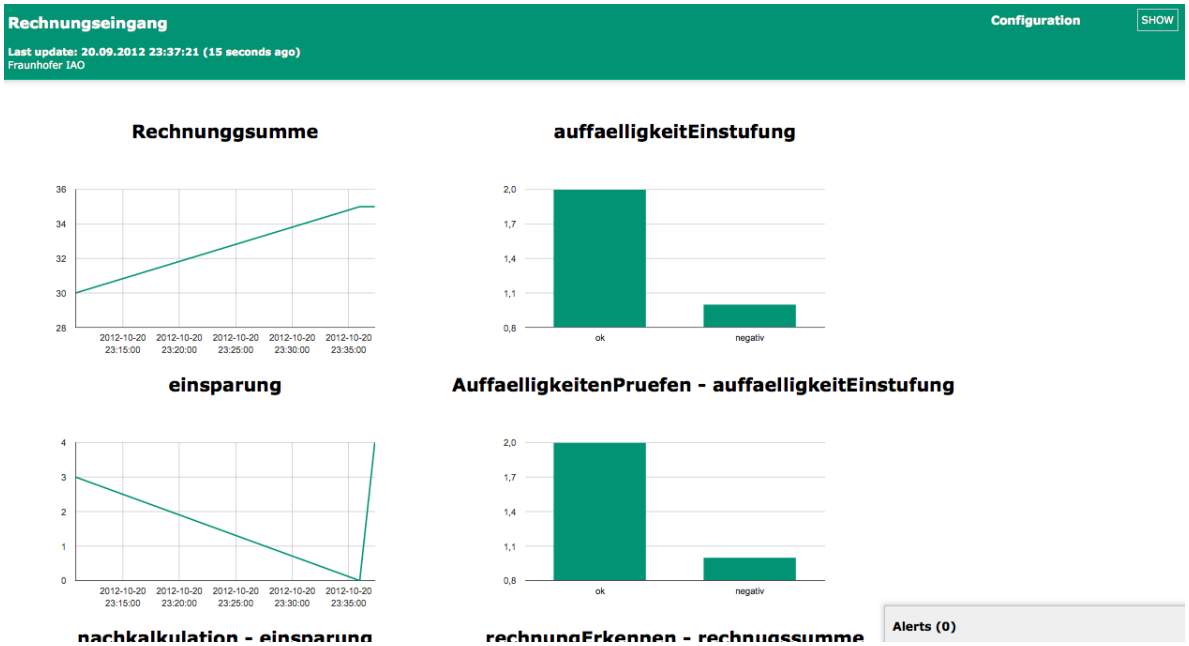

Abbildung 6.14.: Dashboard der Rolle Sachbearbeiter des Beispiels aus der Versicherungsbranche.

6.3. Ergebnis der Evaluation

In der Evaluation wurden zwei Prozesse aus unterschiedlichen Anwendungsbereichen modelliert und mit der Definition von Prozesskennwerten angereichert. Für diese wurden dann wiederum mehrere Sichten definiert und schlussendlich Dashboards für die Sichten generiert.

Es hat sich gezeigt, dass die Modellierung von Sichten auf der Basis von aPro den Anforderungen entspricht. Die drei Sprachen zur Prozessmodellierung, Prozesskennwertdefinition und Sichtendefinition fügen sich gut ineinander und machen damit einem BPMN-erfahrenen Anwender den Einstieg vergleichsweise einfach. Es war außerdem möglich, Messungen in Prozessen unterschiedlichster Natur zu definieren, was die Wahl von aPro als Grundlage als sinnvoll bestätigt.

Die Definition der Sichten war mit der grafischen Modellierungssprache einfach möglich und dennoch hatte der Anwender bei der Sichtendefinition alle nötigen Freiheiten. Im zweiten Prozess zeigte sich außerdem, dass es sinnvoll ist, mehrere KPIs mit einem „Associated Roles“-Element verbinden zu können. Auch hat sich gezeigt, dass die Rollenvererbung in einigen Fällen ein mächtiges Mittel zur Reduktion der Anzahl nötiger Elemente sein kann. Dadurch werden insbesondere große Diagramme übersichtlicher. Allerdings ist dabei

auch viel Sorgfalt nötig um Fehler und damit unerwünschterweise sichtbare Kennwerte zu vermeiden.

7. Zusammenfassung und Ausblick

7.1. Zusammenfassung

Das Ziel dieser Arbeit war der Entwurf eines Sichtenkonzepts für die zielgruppenorientierte Visualisierung von Zielen und KPIs. Die konkreten Anforderungen an dieses Konzept wurden in Kapitel 1.2 genannt und erläutert.

Darauf aufbauend wurde in Kapitel 2 nach bestehenden Ansätzen gesucht, welche entweder das Problem lösen oder als Grundlage dienen können. Die gefundenen Ansätze wurden dann in Kapitel 3 mit den Anforderungen verglichen. Dabei wurde kein Ansatz gefunden, welcher die gestellten Anforderungen ausreichend erfüllt. Als Grundlage für das Sichtenkonzept konnte mit aPro jedoch eine Architektur gefunden werden, welche sich als geeignet herausstellte, da mit ihr die Erfüllung aller Anforderungen möglich ist. Neben einer Kennwertmodellierungssprache namens ProGoalML existierte auch eine prototypische Implementierung auf Basis des Editors Oryx und ein HTML-Dashboard zur Anzeige des automatisch generierten Dashboards.

Auf dieser Basis wurde in Kapitel 4 ein Sichtenkonzept entworfen, das die Definition von Sichten für Prozesskennwerte ermöglicht. Zu Beginn wurden einige Begriffe formal definiert. Auf Basis dieser Definitionen wurde dann ein XML-Format erstellt, das die Speicherung der Sichten entsprechend der Definitionen ermöglicht. Weiterhin wurde eine grafische Sprache entworfen um die Sichten direkt im Kennwertmodell modellieren zu können. Sowohl die Kennwertmodellierungssprache als auch die Sichtenmodellierungssprache fügen sich nahtlos in BPMN ein und ermöglichen damit ein intuitives Arbeiten. Zusätzlich zur grafischen Sprache wurde noch ein Konzept zur Rollenvererbung erarbeitet um die Erstellung großer Sichtenmodelle zu vereinfachen.

Es wurde außerdem eine prototypische Implementierung erstellt, welche in Kapitel 5 beschrieben wurde. Anhand zweier Beispiele aus unterschiedlichen Bereichen wurde das entwickelte Konzept mit dem Prototyp getestet. Es stellte sich dabei heraus, dass der Prototyp die Anforderungen erfüllt. In beiden Beispielen konnten die Sichten auf die Kennwerte wie gewünscht modelliert werden. Darüber hinaus erwies sich die Funktion der Rollenvererbung als mögliches Mittel zur Steigerung der Übersichtlichkeit bei komplexen Beispielen.

Im Folgenden sollen nun noch einige mögliche Erweiterungen des Konzepts aufgezeigt werden, welche nicht im Rahmen dieser Arbeit weiterverfolgt wurden, da sie nicht Teil der

Anforderungen waren. Die Erweiterungen sind unterschiedlich komplex in ihrer Umsetzung und können je nach ihrem Einsatzgebiet unterschiedlich relevant sein.

7.2. Ausblick

7.2.1. Automatische Verteilung der Sichten

In dem hier erstellten Konzept werden die Sichten lediglich generiert. Eine automatisierte Verteilung an die Inhaber der entsprechenden Rollen ist nicht vorgesehen. In Unternehmen wird eine solche Funktion jedoch von Vorteil sein und viel Aufwand ersparen. Nicht zuletzt können so auch mögliche Fehlerquellen ausgeschaltet werden, wie beispielsweise die versehentliche Verteilung eines Dashboards an einen falschen Benutzer bei manueller Verteilung.

Für eine solche Erweiterung müsste die Benutzerdefinition um Angaben zum Verteilweg erweitert werden. Beispielsweise müsste eine Mailadresse der Benutzer hinterlegt werden, an welche der Link zum Dashboard verteilt werden soll. Eine andere Möglichkeit wäre die Verknüpfung mit einer existierenden Datenbank mit Benutzerinformationen. In diesem Fall müsste die ID der Benutzer in der externen Datenbank in der Benutzerdatei gespeichert werden. In beiden Fällen können die zusätzlichen Felder ohne Erweiterung des Schemas als Eigenschaften gespeichert werden. Über die in der externen Datenbank gespeicherten Benutzerdaten kann dann das Dashboard verteilt werden.

Selbstverständlich sind hier noch viele weitere Wege denkbar. Ein allgemeingültiger Ansatz kann hier kaum ausgemacht werden, da die möglichen Lösungswege sich je nach der Umgebung, in welcher das System betrieben wird, unterscheiden.

7.2.2. Anbindung an eine Rechteverwaltung

Das Sichtenkonzept, wie es hier entworfen wurde, ermöglicht keine Rechtekontrolle. Auch der Prototyp implementiert keinerlei Zugriffsbeschränkung auf die generierten Dashboards. Daher kann prinzipiell jeder Benutzer das Dashboard einsehen, der Zugriff auf den Server hat und die URL des Dashboards kennt. In Unternehmen kann dieses Verhalten jedoch unerwünscht sein.

Für die produktive Anwendung in Unternehmen müsste daher eine Zugriffsbeschränkung für die generierten Sichten der Dashboards implementiert werden. Dazu gibt es prinzipiell mehrere Möglichkeiten.

Eine Möglichkeit wäre eine Zugriffsbeschränkung über eine vom Dashboard implementierte Zugriffsbeschränkung. In diesem Fall müsste, ähnlich der automatischen Verteilung von

Dashboards, in der Benutzerdatei für jeden Benutzer eine Referenz auf den korrespondierenden Benutzer im Rechtesystem des Dashboards gespeichert werden (siehe auch Kapitel 5.2.8). In der generierten VisML-Datei für das Dashboard müsste dann eine Liste mit den berechtigten Benutzern gespeichert werden. Das Dashboard gibt dann nur den in der VisML genannten Benutzern Zugriff auf das Dashboard. Selbstverständlich darf die generierte VisML-Datei in diesem Fall nicht frei zugänglich sein. Dieser Ansatz hat jedoch Probleme, wenn sich die einer Rolle zugeordneten Benutzer ändern. In diesem Fall müssten alle betroffenen VisML-Dateien angepasst werden.

Wenn das Dashboard jedoch die existierenden Rollen und die zugeordneten Benutzer kennt, dann entfällt die Notwendigkeit, die einzelnen Benutzer in die VisML-Datei abzuspeichern. Es reicht dann, wenn in der VisML-Datei die zugehörige Rolle angegeben ist. Das Dashboard übernimmt in diesem Fall selbst die Benutzerzuordnung. Auch entfällt das Anpassen der Dateien, wenn Benutzer ihre Rollen wechseln.

Auf welche Art das Dashboard die Zugriffbeschränkung implementiert, ist bei beiden Wegen zweitrangig. Eine sinnvolle Implementierung wird hier auf bereits vorhandene Rechteverwaltungen im Unternehmen zurückgreifen um den Wartungsaufwand gering zu halten.

Neben den genannten Möglichkeiten ist natürlich auch eine rudimentäre Art der Zugriffskontrolle auf Basis von Zugriffsbeschränkungen auf einzelne Ordner möglich. In diesem Fall könnte die Sichtengenerierung automatisch die entsprechenden Rechte für die Ordner mit den generierten Dashboards setzen. Auch hier müssten wieder zusätzliche Daten in Form von Eigenschaften in der Benutzerdatei vorgehalten werden.

7.2.3. Sichten auf Prozesse

Bei der Anzeige des Dashboards zum Monitoring der Kennwerte ist es sinnvoll, auch das Modell des Prozesses, welcher überwacht wird, anzuzeigen. So bekommt der Benutzer einen besseren Überblick über die Bedeutung der Kennwerte. Es ist sogar möglich, die Daten direkt mit den einzelnen Elementen des Prozessmodells zu verknüpfen, so dass der Benutzer des Dashboards die Daten direkt an den betreffenden Elementen betrachten kann und durch Interaktion mit den Elementen die Visualisierungen der Kennwerte einblenden und ausblenden kann.

Unabhängig von der Art der Verknüpfung zwischen Prozessmodell und Dashboard kann es wünschenswert sein, Prozesse auf die für das Monitoring relevanten Elemente zu beschränken. Eine solche Funktionalität bietet dem Anwender mehrere Vorteile:

Sehr große Prozesse könnten durch eine Begrenzung auf relevante Elemente an Übersichtlichkeit gewinnen. Dadurch wird dem Benutzer das Erkennen von Zusammenhängen erleichtert

und er kann schnellere und bessere Rückschlüsse aus den Daten ziehen. Dies ermöglicht ihm schneller auf Probleme oder Optimierungspotential zu reagieren.

Bei Prozessen mit deutlich getrennten Rollen, welche von unterschiedlichen Abteilungen ausgeführt werden, könnten die Regeln eines Unternehmens verbieten, dass einzelne Rollen die Prozessschritte fremder Rollen kennen. In diesem Fall muss der Prozess entsprechend eingeschränkt werden, wenn er im Dashboard angezeigt werden soll. Gleiches gilt, wenn bei Prozessen über Unternehmensgrenzen hinweg unternehmensexterne Rollen Zugriff auf das Dashboard haben sollen.

Sichten auf Prozesse zu definieren ist ein sehr komplexes Thema, wie viele Veröffentlichungen zeigen ([SLS10] [Str09] [Lat10] [EG08] [BRB07]). Das Zusammenspiel mit Sichten auf KPIs birgt außerdem weitere Schwierigkeiten, da die Sichten auf KPIs allein nicht ausreichen um gleichzeitig eine Sicht auf den Prozess eindeutig zu definieren. Je nach Prozess kann ein stark unterschiedliches Verhalten erwünscht sein.

Ein Beispiel für eine Situation, in welcher das erwünschte Verhalten nicht eindeutig klar ist, zeigt Abbildung 7.1. Im dort gezeigten Fall ist fraglich, ob die Verzweigung in der Manager-Sicht nicht enthalten sein soll, oder ob zumindest das Vorhandensein der Verzweigung ersichtlich sein sollte, nicht jedoch ihr Inhalt im unteren Zweig. Ebenso lässt sich argumentieren, dass der Inhalt des unteren Zweigs nicht gelöscht werden sollte, da er keine sicherheitskritischen Informationen trägt, aber für das Verständnis des Prozesses als Ganzes von Bedeutung ist. Die gleiche Argumentation gilt auch für den mittleren Task des oberen Zweigs.

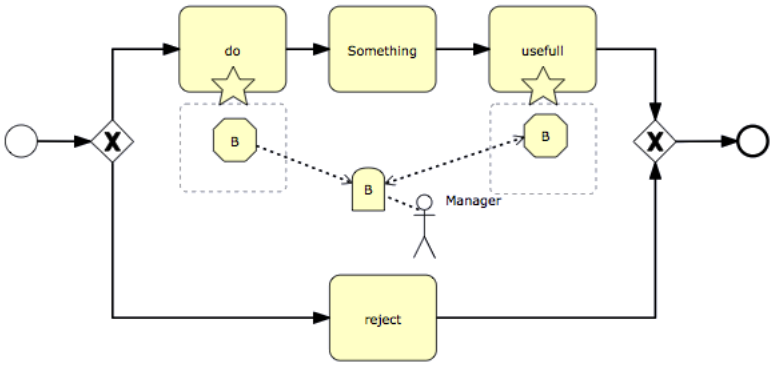

Abbildung 7.1.: Für Sichten auf Prozesse nicht eindeutige Situation.

Situationen, wie die eben genannten, lassen sich in beliebiger Komplexität konstruieren. Je komplexer die im Prozess verwendeten Elemente, desto komplexer werden dabei die auftretenden Fragestellungen. Sichten auf Prozesse können daher nur sehr begrenzt direkt

aus Sichten auf KPIs abgeleitet werden. Es muss zumindest eine Möglichkeit vorgesehen werden, die Sichten auf Prozesse direkt zu beeinflussen.

7.2.4. Prozessoptimierung

Ein Ziel der Prozessüberwachung mittels Monitoring ist das Identifizieren möglicher Optimierungen. Ein weiterer Grund ist das Überwachen einzuhaltender Vorgaben. Diese Vorgaben können aus unterschiedlichen Quellen stammen. So können Vorgaben aus der Unternehmenspolitik entstehen, etwa die Einhaltung gewisser Qualitätskriterien, für die das Unternehmen steht. Andererseits können Vorgaben auch von außen kommen, zum Beispiel von Gesetz wegen, oder als Vorgabe durch den Auftraggeber. In allen genannten Fällen ist die Anpassung des überwachten Prozesses nötig, falls Vorgaben nicht eingehalten werden oder Optimierungspotential entdeckt wird.

Mit ABIS [Kï0] ist in der aPro-Architektur bereits ein Werkzeug zur Anpassung von Prozessen integriert. ABIS ermöglicht es, Referenzprozesse mit sogenannten Fragmenten anzupassen. Implementiert ist ABIS ebenfalls in Oryx, was eine Integration mit dem hier entwickelten Prototypen erleichtert.

7.2.5. Integration von Datenstreams in aPro

Im Gespräch mit den Autoren von aPro entstand die Idee der Integration von Newsfeeds in aPro. Dadurch können, analog zu den Custom Alerts, auch Daten im Dashboard angezeigt werden, welche nicht direkt mit Instanzen eines Prozesses in Verbindung stehen. Dafür eignen sich unter anderem die verbreiteten Techniken RSS [Rss] und Atom [NS05].

Die Einbindung dieser Datenquellen in das Sichtenkonzept kann analog zu den Custom Alerts über die rollenspezifischen Eigenschaften erfolgen. Neben den bereits bei den Custom Alerts verwendeten Querys können auch weitere Mechanismen in den Eigenschaften gespeichert werden. So ist auch die Eingrenzung von Meldungen auf Basis von Attributen denkbar. Dadurch können zum Beispiel nur Meldungen eines bestimmten Autors als sichtbar definiert werden oder nur Meldungen nach einem bestimmten Datum.

Um eine saubere Trennung zwischen Eigenschaften für Newsfeeds und allgemeinen Eigenschaften zu erreichen bietet sich auch die Erweiterung des Schemas an. So könnte eine standardisierte Möglichkeit der Abbildung von Sichten auf Newsfeeds geschaffen werden. Durch die modulare Architektur kann diese Anforderung ohne weitere Probleme umgesetzt werden.

Anhang

A. XML-Schema Definitionen

A.1. Rollen- und Benutzerdefinition

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:tns="http://userDef.viewDefinition.oryxeditor.org"
      targetNamespace="http://userDef.viewDefinition.oryxeditor.org"
         elementFormDefault="qualified">

      <element name="userRoles"
         type="tns:tUserAndRolesDef"></element>

      <complexType name="tUserAndRolesDef">
            <sequence>
                  <element name="roles"
                     type="tns:tRoles"></element>
                  <element name="users"
                     type="tns:tUsers"></element>
            </sequence>

            <attribute name="version" type="string" fixed="1.0"
               use="required" />
      </complexType>

      <complexType name="tRoles">
            <sequence>
                  <element name="role" type="tns:tRole"
                     maxOccurs="unbounded"
                        minOccurs="0">
                  </element>
```

```
        </sequence>
    </complexType>

    <complexType name="tUsers">
        <sequence>
            <element name="user" type="tns:tUser"
                maxOccurs="unbounded"
                    minOccurs="0"></element>
        </sequence>
    </complexType>

    <complexType name="tRole">
        <sequence>
            <element name="description" type="string"
                maxOccurs="1"
                    minOccurs="0">
            </element>
            <element name="properties" type="tns:tProperties"
                    maxOccurs="1" minOccurs="0"></element>
        </sequence>
        <attribute name="name" type="string"></attribute>
    </complexType>

    <complexType name="tUser">
        <sequence>
            <element name="roles"
                type="tns:tUserRoles"></element>
            <element name="properties" type="tns:tProperties"
                    maxOccurs="1" minOccurs="0"></element>
        </sequence>
        <attribute name="id" type="string"
            use="required"></attribute>
        <attribute name="name" type="string"></attribute>
    </complexType>

    <complexType name="tUserRoles">
        <sequence>
            <element name="role" type="string"
                maxOccurs="unbounded"
                    minOccurs="0"></element>
        </sequence>
    </complexType>
```

```
    <complexType name="tProperties">
        <sequence>
            <element name="property" type="tns:tProperty"
                maxOccurs="unbounded"
                    minOccurs="1"></element>
        </sequence>
    </complexType>

    <complexType name="tProperty">
        <attribute name="key" type="string"></attribute>
        <attribute name="value" type="string"></attribute>
    </complexType>
</schema>
```

Listing 1: XML-Schema Definition der Rollen- und Benutzerdatei.

A.2. Sichtdefinition

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
   targetNamespace="http://viewDef.viewDefinition.oryxeditor.org"
      xmlns="http://www.w3.org/2001/XMLSchema"
         xmlns:tns="http://viewDef.viewDefinition.oryxeditor.org"
         elementFormDefault="qualified">

    <element name="views" type="tns:tViews"></element>


    <complexType name="tViews">
        <sequence>
            <element name="meta" type="tns:tMeta"></element>
            <element name="preferences"
                type="tns:tPreferences"></element>
            <element name="elements"
                type="tns:tElements"></element>
        </sequence>
```

```
        <attribute name="version" type="string" fixed="1.0"
           use="required" />
    </complexType>

    <complexType name="tMeta">
        <sequence>
            <element name="author" type="string"
               maxOccurs="1" minOccurs="0">
            </element>
            <element name="lastmodified" type="dateTime"
               maxOccurs="1"
                  minOccurs="0">
            </element>
            <element name="comment" type="string"
               maxOccurs="1"
                  minOccurs="0">
            </element>
            <element name="userFileRef" type="string"
               maxOccurs="1"
                  minOccurs="0"></element>
        </sequence>
    </complexType>

    <complexType name="tPreferences">
        <sequence>
            <element name="global"
               type="tns:tGlobalPreferences"></element>
            <element name="roleSpecific"
               type="tns:tRoleSpecificPreferences"></element>
        </sequence>
    </complexType>

    <complexType name="tElements">
        <sequence>
            <element name="element" type="tns:tElement"
               maxOccurs="unbounded"
                  minOccurs="1"></element>
        </sequence>
    </complexType>

    <complexType name="tGlobalPreferences">
        <sequence>
```

```
                <element name="preference"
                   type="tns:tGlobalPreference"
                      maxOccurs="unbounded"
                         minOccurs="0"></element>
            </sequence>
        </complexType>

        <complexType name="preferenceType"></complexType>

        <complexType name="tGlobalPreference">
            <attribute name="key" type="string"></attribute>
            <attribute name="value" type="string"></attribute>
        </complexType>

        <complexType name="tRoleSpecificPreferences">
            <sequence>
                <element name="preference"
                   type="tns:tRoleSpecificPreference"
                      maxOccurs="unbounded"
                         minOccurs="0"></element>
            </sequence>
        </complexType>

        <complexType name="tRoleSpecificPreference">
            <attribute name="key" type="string"></attribute>
            <attribute name="role" type="string"></attribute>
            <attribute name="value" type="string"></attribute>
        </complexType>

        <complexType name="tElement">
            <sequence>
                <element name="role" type="string"
                   maxOccurs="unbounded"
                      minOccurs="0"></element>
            </sequence>
            <attribute name="stencil" type="string"
               use="optional"></attribute>
            <attribute name="name" type="string"></attribute>
        </complexType>
</schema>
```

Listing 2: XML-Schema Definition der Sichtendefinitionsdatei.

Literatur

[Bar+06] F. Barbon u. a. „Run-Time Monitoring of Instances and Classes of Web Service Compositions“. In: *2006 IEEE International Conference on Web Services ICWS06* 06 (2006), S. 63–71. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4032013. (Siehe S. 7, 15, 16, 27).

[BG05] L. Baresi und S. Guinea. „Towards Dynamic Monitoring of WS-BPEL Processes“. In: *ICSOC 2005, Third International Conference of Service-Oriented Computing, volume 3826 of Lecture Notes in Computer Science*. Springer, 2005, S. 269–282. (Siehe S. 16, 28).

[BRB07] R. Bobrik, M. Reichert und T. Bauer. *Parameterizable Views for Process Visualization*. Techn. Ber. TR-CTIT-07-37. Enschede: Centre for Telematics und Information Technology University of Twente, 2007. (Siehe S. 96).

[Che07] P. Chen. „Goal-oriented business process monitoring“. Magisterarb. Carleton University, 2007. (Siehe S. 7, 17, 18, 28).

[DGB07] G. Decker, A. Großkopf und A. P. Barros. „A Graphical Notation for Modeling Complex Events in Business Processes“. In: *EDOC*. 2007, S. 27–36. (Siehe S. 7, 18, 19, 28).

[Ecl] *Eclipse Modeling Framework Project (EMF)*. http://www.eclipse.org/modeling/emf/. (Siehe S. 18).

[EG08] R. Eshuis und P. Grefen. „Constructing customized process views“. In: *Data Knowl. Eng.* 64.2 (Feb. 2008), S. 419–438. ISSN: 0169-023X. DOI: 10.1016/j.datak.2007.07.003. URL: http://dx.doi.org/10.1016/j.datak.2007.07.003. (Siehe S. 96).

[Gar] *Gartner Reveals Five Business Process Management Predictions for 2010 and Beyond*. http://www.gartner.com/it/page.jsp?id=1278415. 2010. (Siehe S. 11).

[GG03] D. C. Goldenson und D. L. Gibson. „Demonstrating the Impact and Benefits of CMMI®: An Update and Preliminary Results“. In: *Program* October (2003), S. 1–55. URL: http://repository.cmu.edu/sei/548/. (Siehe S. 11).

[Goo] *Google Chart Tools*. https://developers.google.com/chart/. (Siehe S. 20).

[Juc] *jUCMNav - Eclipse plugin for the User Requirements Notation*. `http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome`. (Siehe S. 18).

[Kï0] F. M. Kötter. „Prozessvarianten in unternehmensübergreifenden Servicenetzwerken“. Magisterarb. Universität Stuttgart, 2010. (Siehe S. 97).

[KG06] G. Koliadis und A. Ghose. „Relating Business Process Models to Goal-Oriented Requirements Models in KAOS“. In: *Proceedings of the PacificRim Knowledge Acquisition Workshop PKAW2006* 4303.8 (2006), S. 25–39. URL: `http://www.springerlink.com/content/g8154846h5410418`. (Siehe S. 7, 21, 22, 29).

[Kin12] M. Kintz. „A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology“. In: *Proceedings of 2nd International Workshop on Model-based Interactive Ubiquitous Systems (MODIQUITOUS 2012), Copenhagen, Denmark, 2012.* 2012. (Siehe S. 20).

[KK12a] F. Koetter und M. Kochanowski. „A Model-Driven Approach for Event-Based Business Process Monitoring“. In: (2012). (Siehe S. 18, 29, 78).

[KK12b] F. Koetter und M. Kochanowski. „Goal-Oriented Model-Driven Business Process Monitoring Using ProGoalML“. In: *Proceedings of the 15th International Conference on Business Information Systems BIS 2012 in press* (2012), S. 72–83. (Siehe S. 7, 8, 18–21, 29, 60, 61).

[Lat10] G. Latuske. „Sichten auf Geschäftsprozesse als Werkzeug zur Darstellung laufender Prozessinstanzen“. Diplomarbeit. Universität Stuttgart, 2010. (Siehe S. 7, 24, 30, 96).

[MR00] M. Z. Muehlen und M. Rosemann. „Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues“. In: *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33*. IEEE Computer Society Press, 2000, S. 1–10. (Siehe S. 25, 31).

[Now+11] A. Nowak u. a. „An Architecture and Methodology for a Four-Phased Approach to Green Business Process Reengineering“. In: *Proceedings of the 1st International Conference on ICT as Key Technology for the Fight against Global Warming, ICT-GLOW 2011, August 29 - September 2, 2011, Toulouse, France*. Bd. 6868. Lecture Notes in Computer Science (LNCS). Springer-Verlag, 2011, S. 150–164. (Siehe S. 77).

[NS05] M. Nottingham und R. Sayre. *The Atom Syndication Format*. `http://tools.ietf.org/html/rfc4287`. 2005. (Siehe S. 97).

[Par10] D. Parmenter. *Key Performance Indicators (KPI): Developing, Implementing, and Using Winning KPIs*. John Wiley & Sons, 2010. ISBN: 9780470545157. (Siehe S. 37).

[PCBV10] S. Patig, V. Casanova-Brito und B. Vögeli. „IT requirements of business process management in practice - an empirical study“. In: *Proceedings of the 8th international conference on Business process management*. BPM'10. Berlin, Heidelberg: Springer-Verlag, 2010, S. 13–28. URL: http://dl.acm.org/citation.cfm?id=1882061.1882067. (Siehe S. 12).

[Pet07] N. Peters. *Oryx Stencil Set Specification*. Potsdam, Germany, 2007. URL: http://oryx-editor.googlecode.com/files/OryxSSS.pdf. (Siehe S. 63).

[Pol07] D. Polak. *Oryx BPMN Stencil Set Implementation*. Potsdam, Germany, 2007. (Siehe S. 61).

[RORRC10] A. del Río-Ortega, M. Resinas und A. Ruiz-Cortés. „Defining Process Performance Indicators: An Ontological Approach“. In: *On the Move to Meaningful Internet Systems: OTM 2010*. Bd. 6426. Lecture Notes in Computer Science 1. Springer, 2010, 555–572. ISBN: 978-3-642-16933-5. DOI: 10.1007/978-3-642-16934-2_41. URL: http://www.springerlink.com/index/G334V7K288546VNN.pdf. (Siehe S. 7, 21, 22, 29, 33).

[Rss] *RSS 2.0 Specification*. http://www.rssboard.org/rss-specification. (Siehe S. 97).

[SAZ04] K. Saleh und A. Al-Zarouni. „CAPTURING NON-FUNCTIONAL SOFTWARE REQUIREMENTS USING THE USER REQUIREMENTS NOTATION“. In: *Technology* (2004), S. 222–230. URL: http://lotos.site.uottawa.ca/ucm/pub/UCM/VirLibIIT04/IIT04.pdf. (Siehe S. 17).

[SLS10] D. Schumm, F. Leymann und A. Streule. „Process Viewing Patterns“. In: *Proceedings of the 14th IEEE International EDOC Conference (EDOC 2010)*. IEEE Computer Society, 2010. DOI: 10.1109/EDOC.2010.16. (Siehe S. 7, 23, 30, 96).

[SM06] G. Spanoudakis und K. Mahbub. „Non intrusive monitoring of service based systems“. In: *International Journal of Cooperative Information Systems* 15.3 (2006), S. 1–29. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.5108&rep=rep1&type=pdf. (Siehe S. 16).

[Str09] A. Streule. „Abstract views on BPEL processes“. Magisterarb. Universität Stuttgart, 2009. (Siehe S. 24, 30, 96).

[Tsc07] W. Tscheschner. *Oryx Dokumentation*. Potsdam, Germany, 2007. (Siehe S. 21, 61, 62).

Alle URLs wurden zuletzt am 8. November 2012 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Christian Sachers)