Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3709

Grafischer Webeditor für Compliance-Anforderungen

Philipp Gildein

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Frank Leymann
Betreuer/in:	Dipl.-Inf. Falko Kötter Dipl.-Inf. Christoph Fehling
Beginn am:	01. Dezember 2014
Beendet am:	02. Juni 2015
CR-Nummer:	H.4.1, H.5.2

Kurzfassung

Durch gesetzliche Regelungen wie den Sarbanes-Oxley Act (SOX), das europäische Äquivalent EURO-SOX oder BASEL II herrscht immer größerer Druck auf Unternehmen, dafür Sorge zu tragen, dass Regularien eingehalten werden, um kostspielige Verletzungen und PR-Schäden zu vermeiden. Für das Kerngeschäft der Unternehmen werden oftmals bereits Geschäftsprozesse eingesetzt, um für korrekte Abläufe zu Sorgen. Die dafür verwendeten Werkzeuge lassen jedoch oftmals die Unterstützung der sogenannten Compliance-Anforderungen außen vor, mit denen zusätzlich die Einhaltung von internen und externen Regel überprüft und sichergestellt werden kann. Da das Wechseln von bereits eingesetzten Werkzeugen kostenintensiv ist, werden Verfahren gesucht, die bestehende Prozesse um Compliance-Anforderungen erweitern und ein Gesamtkonzept aus Funktionalität und Benutzerfreundlichkeit bieten.

Diese Arbeit erweitert ein bestehendes Konzept zur Erstellung und Validierung von Compliance-Anforderungen, den Compliance Descriptor, um ein grafisches Modell sowie einen dazugehörigen Editor. Einen großen Funktionsumfang hat dieser zwar bereits geboten, durch eine grafische Darstellung wird die Umsetzung der Anforderungen jedoch deutlich vereinfacht und auch für Mitarbeiter ohne technische Ausbildung möglich. Zudem können Informationen über den Zusammenhang der Anforderungen besser ausgetauscht werden.
Um zusätzlich Erweiterungen des Konzepts für die Zukunft zu evaluieren, wird ein mobiler Editor prototypisiert. Dieser soll es ermöglichen auch mit Smartphones und Tablets am Compliance Descriptor zu arbeiten.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Motivation und Aufgabenstellung

In den letzten 15 Jahren stieg die Anzahl der Regeln und Gesetze, die Unternehmen einhalten und beachten müssen, stetig an. Zudem kommt eine immer größere werdende Menge an internen Richtlinien. Beispiele dafür sind BASEL II, der Sarbanes-Oxley Act (SOX) [36] in den USA oder EURO-SOX in der EU [11]. Deren Verletzung kann sowohl zu hohen Strafen als auch zu großen PR-Schäden führen. Das Einhalten dieser Anforderungen wird dadurch zu einer der wichtigsten Aufgaben von Unternehmen und bedarf kompetenter Mitarbeiter, deren Aufgabe es ist, sich um neuste Gesetzeslagen und Richtlinien sowie deren Einhaltung zu kümmern [17].
Gleichzeitig nimmt der Einsatz von Software zur Entwicklung und Ausführung von Geschäftsprozessen immer mehr an Bedeutung zu, mit Hilfe deren Abläufe im Geschäftsbetrieb modelliert werden können. Da die meisten der sogenannten Compliance-Anforderungen während dieser Abläufe geprüft werden könnten, bietet sich eine Kombination von Geschäftsprozessen und deren Überprüfung an. Dies findet jedoch bisher nur in sehr wenigen Fällen statt, da dafür entsprechende Softwarelösungen am Markt fehlen oder noch unausgereift sind.

Eine weitere Schwierigkeit stellt die Vielzahl der Fachgebiete dar, in die die Compliance hereinreicht. Geschäftsprozesse werden meist von betriebswirtschaftlich orientierten Abteilungen erstellt, während zu deren Ausführung Hilfsmittel von informationstechnischen Abteilungen zur Verfügung gestellt werden. Greift der Prozess zudem in die Produktion, den Kundenservice oder Support ein, kommen schnell zahlreiche Beteiligte zusammen. Dadurch ist Compliance ein fachgebietsübergreifendes Thema und bedarf der Zusammenarbeit verschiedener Abteilungen [19]. Ohne eine leicht verständliche Darstellung wird es für technisch weniger versierte Mitarbeiter jedoch schwierig mitzuwirken, weshalb dies im Blick behalten werden muss.

In der vorliegenden Arbeit soll deshalb ein bereits existierender Ansatz, der Compliance Descriptor [23], um ein grafisches Modell und einen Editor, der dieses Modell verwendet, erweitert werden. Bisher ist dieser nur durch ein Dateiformat sowie diverse Semantiken und Validierungen beschrieben. Daher geht es in dieser Arbeit insbesondere um eine benutzerfreundliche Darstellung und Bedienung des Editors.
Zusätzlich soll ein Prototyp einer mobilen Anwendung entwickelt werden, der die Funktionalitäten des Editors auf mobilen Endgeräten liefert. Dieser würde die Einstiegshürde in die Modellierung von Compliance-Anforderungen deutlich verringern.

1.2. Gliederung der Arbeit

Die Gliederung der vorliegenden Arbeit wird im Folgenden erläutert. Sie orientiert sich dabei an der Vorgehensweise der Entwicklung und strukturiert sich folgendermaßen:

Kapitel 1 - Einleitung: Kapitel 1 beschreibt die Aufgabenstellung und Gliederung der Arbeit und gibt einen kurzen Überblick über das Thema.

Kapitel 2 - Grundlagen: Kapitel 2 stellt einige wichtige Begriff vor und bietet einen Einblick in die Beschreibungssprache Business Process Model and Notation (BPMN).

Kapitel 3 - State of the Art: Kapitel 3 enthält einen Überblick über aktuelle Entwicklungen aus dem Umfeld der Compliance- und Geschäftsprozessvalidierung und stellt die Verfahren vor.

Kapitel 4 - Erstellung des grafischen Modells: Kapitel 4 analysiert den Compliance Descriptor und stellt die dafür entwickelte grafische Darstellung vor.

Kapitel 5 - Erstellung des grafischen Editors: Kapitel 5 zeigt die Umsetzung der grafischen Darstellung mit Hilfe des Oryx-Editors und die Implementierung von benutzerfreundlichen Erweiterungen

Kapitel 6 - Beispiel aus der Versicherungsbranche: Kapitel 6 stellt ein Beispiel für einen Geschäftsprozess aus der Versicherungsbranche vor und implementiert die daran gestellten Complianceanforderungen mit Hilfe des grafischen Editors.

Kapitel 7 - Mobiler Prototyp: Kapitel 7 beschreibt die Konzeption, Architektur und Erstellung des mobilen Prototypen und zeigt die Erstellung des Beispiels aus Kapitel 6 in einer mobilen Anwendung.

Kapitel 8 - Zusammenfassung und Ausblick: Kapitel 8 fasst den Inhalt dieser Arbeit zusammen und bietet einen Ausblick auf potenzielle Erweiterungen der entwickelten Editoren.

2. Grundlagen

In diesem Kapitel soll auf ein paar wichtige Begriffe und Technologien eingegangen werden, die die Grundlage für den weiteren Verlauf der Arbeit bilden. Dafür werden in den ersten beiden Abschnitten die Begriffe "Geschäftsprozessünd "Compliance"definiert. Abschließend wird in Abschnitt 2.3 auf die Business Process Model and Notation (kurz: BPM) eingegangen, eine grafische Beschreibungssprache für Geschäftsprozesse.

2.1. Geschäftsprozess

2.1.1. Definition

Ein Geschäftsprozess ist eine Anzahl von Einzeltätigkeiten, die in einer definierten Reihenfolge ausgeführt werden, um ein Ziel zu erreichen.
Das Gabler Wirtschaftslexikon enthält folgende Definition:

> "Folge von Wertschöpfungsaktivitäten mit einem oder mehreren Inputs und einem Kundennutzen stiftenden Output. Geschäftsprozesse können auf verschiedenen Aggregationsebenen betrachtet werden, z.B. für die Gesamtunternehmung, einzelne Sparten- oder Funktionalbereiche"[34]

Eine weitere Definition stammt von Bernd Oestereich:

> "Ein Geschäftsprozess ist eine Zusammenfassung einer Menge fachlich verwandter Geschäftsanwendungsfälle. Dadurch bildet ein Geschäftsprozess gewöhnlich eine Zusammenfassung von organisatorisch eventuell verteilten, fachlich jedoch zusammenhängenden Aktivitäten, die notwendig sind, um einen Geschäftsvorfall (z.B. einen konkreten Antrag) ergebnisorientiert zu bearbeiten. Die Aktivitäten eines Geschäftsprozesses stehen gewöhnlich in zeitlich und logischer Abhängigkeit zueinander." [28]

Geschäftsprozesse lassen sich in drei Gruppen einteilen:

- **Managementprozesse**
  Strategische Prozesse, die die höchste Ebene des Unternehmens betreffen.

- **Operative Prozesse**
  Prozesse, die das Kerngeschäft des Unternehmens betreffen und die meist in Zusammenhang mit dem Kunden stehen.

- **Unterstützende Prozesse**
  Prozesse, die der Unterstützung interner Abläufe, wie z.B. bei Einstellungsverfahren, oder der Einhaltung von Richtlinien, beispielsweise beim Umgang mit persönlichen Daten, dienen.

Neben der Modellierung von Prozessen, die im zweiten Teil dieser Definition erläutert wird,können diese Prozesse auch um Ausführungssemantiken erweitert werden. Dadurch lässt sich ein Prozess von einem sogenannten Arbeitsablaufsystem ausführen. Dieses kontrolliert die einzelnen Schritte, leitet die entsprechenden Folgeschritte ein und kümmer sich um das Senden und Empfangen von Nachrichten.

Ein beliebtes Beispiel für einen Geschäftsprozess, in diesem Fall einen operativen Prozess, ist die Einrichtung eines Konto bei einer Bank. Dabei gibt es klar definierte Start- und Endpunkte (der Antrag des neuen Kunden und die erfolgte Einrichtung) sowie Zwischenschritte, die zwischen diesen beiden Punkten liegen. An Zweigstellen kann es durch Überprüfungen und Entscheidungen zu unterschiedlichen Verläufen kommen, z.B. durch die Prüfung der Kreditwürdigkeit des Kunden.

2.1.2. Modellierung

Für die Modellierung von Geschäftsprozessen gibt es verschiedene Möglichkeiten. Unterscheiden lassen sich dabei prinzipiell graphische und textuelle Repräsentationen. Der gewählte Ansatz hängt dabei oftmals vom Einsatzgebiet sowie von der Erfahrung des Erstellers und des Zielpublikums ab.

Die Vorteile einer graphischen Modellierung liegen bei der Einfachheit der Erstellung und der Interpretation des dabei entstandenen Modells. So gibt es dafür meist Editoren, die dem Anwender viele Hilfeleistungen an die Hand geben und dadurch die Eingabe des Prozesses simplifizieren. Zudem zeigt die graphische Darstellung die Verbindungen und Abhängigkeiten klarer auf.
Dadurch eignet sich diese Darstellungsform sehr gut zum Austausch zwischen verschiedenen Abteilungen eines Unternehmens. Auch weniger erfahrene Mitarbeiter können so den Geschäftsprozess verstehen.

Auch die textuelle Repräsentation hat einige Vorteile. So verstecken graphische Darstellungen oftmals die Komplexitäten der dahinter liegenden Datenstrukturen. Deshalb können mit einer Textdarstellung theoretisch komplexere und ausführlichere Prozesse erstellt werden. Das ist auch der Grund, weshalb vor allem bei ausführbaren Geschäftsprozessen textuelle Modelle den graphischen bisher vorgezogen werden.

Während sich die meisten Standards bei Geschäftsprozessen auf eine der beiden Repräsentationen konzentrieren, gibt es vermehrt Bemühungen, beide Formen innerhalb eines Standards zu unterstützen. Ein großes Problem der graphischen Repräsentation ist nämlich, dass sich reine Graphiken schlecht aus einem Programm exportieren und in ein anderes Programm importieren lassen. Textformate haben dieses Problem nicht.

Im Abschnitt 2.3 wird eine grafische Darstellungsform von Geschäftsprozessen in Form der Business Process Model and Notation (BPMN) vorgestellt.

2.2. Compliance

2.2.1. Definition

Compliance, oder zu deutsch Regeltreue, bezeichnet die Einhaltung von Gesetzen, Richtlinien und Standards in Unternehmen. Compliance-Anforderungen können sowohl von außen vorgegeben sein als auch intern festgelegt werden. Eine weitere Definition von Eberhard Krügler verdeutlich dies:

> "Der Begriff Compliance steht für die Einhaltung von gesetzlichen Bestimmungen, regulatorischen Standards und Erfüllung weiterer, wesentlicher und in der Regel vom Unternehmen selbst gesetzter ethischer Standards und Anforderungen."[24]

Geprägt wurde der Begriff der Compliance vor allem in der amerikanischen Finanzwirtschaft. Nach finanziellen Skandalen um Firmen wie den Ölkonzern Enron wurden dort weitreichende Änderungen an der Finanzgesetzgebung vorgenommen, um das Vertrauen der Anleger in Aktien von Unternehmen wieder zu erhöhen. Dazu wurde 2002 der Sarbanes-Oxley Act (kurz: SOX) entworfen und in Kraft gesetzt. Dieser sieht eine deutlich höhere Dokumentationspflicht für Unternehmen, die in den USA tätig sind, vor.
Im europäischen Raum wurde analog dazu die Abschlussprüfungs-Richtlinie verabschiedet, auch kurz EURO-SOX genannt, sowie die Eigenkapitalvorschrift BASEL II. Auch diese stellen deutlich höhere Anforderungen an die Dokumentation von Unternehmen und stellen bei Verletzung hohe Strafen in Aussicht.

Compliance wird deshalb eine immer wichtiger werdende Aufgabe von Unternehmen und beschäftigt sie bis in die höchsten Ebenen. So definiert der Deutsche Corporate Governance Kodex, ein freiwilliges Regelwerk, dass sich vor allem an im DAX gelistete Unternehmen richtet, die Aufgabe des Vorstandes so:

> "Der Vorstand hat für die Einhaltung der gesetzlichen Bestimmungen und der unternehmensinternen Richtlinien zu sorgen und wirkt auf deren Beachtung durch die Konzernunternehmen hin (Compliance)."[20]

2.2.2. Business Process Compliance

Da viele Firmen bereits Geschäftsprozesse definiert haben, um wichtige Prozesse ihres Kerngeschäfts zu beschreiben und korrekt auszuführen, werden diese oftmals mit der Überprüfung von Compliance-Anforderungen kombiniert. Die Kombination von Geschäftsprozess und Compliance wird auch als Business Process Compliance (BPC) [33] bezeichnet. Durch entsprechende Softwarelösungen können dann die Compliance-Anforderungen gesammelt und modelliert werden. Bei der Entwicklung und Ausführung der Geschäftsprozesse können sie dann validiert werden.
Eine Alternative stellt die Überprüfung der Anforderungen nach Ende des Geschäftsprozesses dar. Dabei werden während der Ausführung relevante Daten gesammelt, analysiert und ein Bericht daraus generiert.
In Kapitel 3 werden vier verschiedene Verfahren vorgestellt, die Funktionen zur Erstellung und Validierung von Compliance in Geschäftsprozessen zur Verfügung stellen.

2.3. Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) [16] ist eine graphische Notation zur Beschreibung von Geschäftsprozessen. Die Arbeit an BPMN wurde 2001 von Stephen A. White, einem IBM-Mitarbeiter, begonnen und 2004 in der Version 1.0 von der Business Process Management Initiative (BPMI) veröffentlicht. Im Juni 2005 übernahm die Object Management Group (OMG), die unter anderem auch die Unified Modeling Language (UML) entwickelt hat, die Weiterentwicklung und Pflege des Standards.

2011 wurden mit der Veröffentlichung der Version 2.0 [16] die ersten größeren Änderungen am Standard veröffentlicht. Mit der neuen Version beschreibt BPMN nicht nur die graphische Darstellung sondern auch ein XML [37]-basiertes Austauschformat, mit dem BPMN-Modelle zwischen verschiedenen Programmen ausgetauscht werden können. Seit Mitte 2013 ist BPMN in der Version 2.0.1 zudem ein internationaler Standard (ISO/IEC 19510:2013).

Bei der Modellierung von Prozessen wird von BPMN ein graph-basierter Ansatz gewählt. Dies bedeutet, dass ein Prozess mit Hilfe von Knoten und Kanten beschrieben wird, jedoch ergänzt um weitere graphische Elemente, die es ermöglichen komplexere Abläufe darzustellen.

BPMN in der ersten Version besteht aus vier Kategorien solcher Elemente, die im folgenden näher beschrieben werden. Es enthält zudem drei verschiedene Diagrammtypen, die ebenfalls kurz veranschaulicht werden. Da das Hinzufügen eines Austauschformats eine der größten Änderungen in der Geschichte von BPMN war, soll auch dieses kurz angerissen werden.

2.3.1. Elemente

Vier Gruppen von Objekten mit insgesamt 23 Elementen stellen die Hauptbausteine von BPMN dar und sollen in diesem Abschnitt beschrieben und durch Bilder illustriert werden.

2.3.1.1. Flussobjekte

Flussobjekte repräsentieren alle Aktionen die innerhalb eines Geschäftsprozesses dessen Verhalten spezifizieren. Es gibt drei verschiedene Arten von Flussobjekten: Ereignisse, Aktivitäten und Gateways.

Ereignisse

Ereignisse können in drei verschiedenen Formen auftreten: als Startereignis, das den Ausgangspunkt eines Flusses markiert, als Zwischenereignis, welches entweder Informationen empfangen oder senden kann, oder als Endereignis, das das Ende eines Flusses markiert. Ein Zwischenereignis unterbricht dabei den Ablauf des Prozesses.

Abbildung 2.1.: Ereignisse

Dargestellt werden Ereignisse als Kreis. Ist das Ereignis von einem speziellen Typ, wie zum Beispiel eine Nachricht, dann wird zusätzlich ein Symbol im Kreis angezeigt.
Je nach Art des Ereignisses werden der Kreis und das Symbol unterschiedlich dargestellt. Bei einem Startereignis wird ein dünner Kreis gezeichnet, bei einem Zwischenereignis ein doppelter Kreis und bei einem Endereignis ein doppelter, gefüllter Kreis.
Ein nicht ausgefülltes Symbol steht dabei für ein eingehendes Ereignis, während es bei einem ausgehenden Ereignis ausgefüllt ist. Ausgehende Ereignisse sind dabei alle Endereignisse sowie manche Zwischenereignisse.

Aktivitäten

Auszuführende Tätigkeiten im Prozess werden durch Aktivitäten beschrieben. Dabei kann eine Aktivität aus mehreren Arbeitsschritten bestehen. Von BPMN werden vier verschiedene Haupttypen definiert: eine Aufgabe, die eine Arbeitseinheit darstellt, eine Transaktion, die logisch zusammenhängende Aktivitäten verknüpft, ein Teilprozess, der zu einem übergeordneten Prozess gehört und von diesem entweder parallel ausgeführt werden kann oder unterbricht, und eine Aufruf-Aktivität, die die Kontrolle über den Prozess an einen global definierten Aufruf abgibt.

Aktivitäten werden als Rechteck mit abgerundeten Ecken dargestellt. Ein Teilprozess enthält zusätzlich ein + in einem Quadrat, um den Inhalt des Teilprozesses aufklappen zu können. Transaktionen verwenden einen doppelten Rand, während eine Aufruf-Aktivität über einen dickeren Rand verfügt.

Ergänzt werden können Aufgaben und Teilprozesse durch Markierungen, die sich graphisch im unteren Teil des Elementes befinden. Beiden gemeinsam sind dabei die Schleife, die die Aktivität bis zur Erfüllung einer Bedingung wiederholt, die multiple Instanz, die die Aktivität mehrmals parallel oder sequentiell ausführt sowie die Kompensation, die einen Ausgleich (z.B. Lohn oder Gebühren) für vorhergehende Aktivitäten auszahlt.
Aufgaben können die Kompensation zusätzlich in einer Schleife ausführen. Ein Teilprozess kann zusätzlich eine Ad Hoc-Markierung erhalten, die Aktivitäten dieses Teilprozesses in beliebiger Reihenfolge ausführt.

Um Aufgaben noch genauer spezifizieren zu können, kann darauf aufsetzend auch noch ein Typ angegeben werden. Dieser wird als Symbol in der linken oberen Ecke angezeigt. Dadurch kann eine Aufgabe als händisch oder durch einen Benutzer auszuführend markiert, ein Skript ausgeführt, eine Nachricht empfangen oder gesendet, oder ein automatisierter Service verwendet werden.

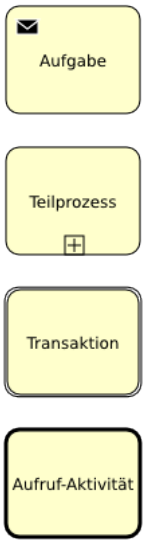

Abbildung 2.2.: Aktivitäten

Gateways

Das dritte und letzte Flussobjekt ist das Gateway-Element, dass Kanten im Sequenzfluss aufteilt und wieder zusammenfügt. Gateways werden durch eine Raute dargestellt, die je nach Verzweigungsart ein Symbol enthalten.

Abbildung 2.3.: Gateways

Fünf solcher Verzweigungsarten werden von BPMN definiert: ein exklusives Gateway, welches auf Grund einer angegebenen Bedingung eine der Kanten aktiviert, ein ereignisbasiertes Gateway, bei dem das nächste Element ein Ereignis sein muss und die Kante des zuerst aktivierten Ereignisses aktiviert wird, ein paralleles Gateway, bei dem alle Kanten aktiviert werden und erst zusammengeführt werden, wenn alle abgeschlossen haben, ein inklusives Gateway, bei dem je nach Bedingung eine oder mehrere Kanten aktiviert werden können, sowie ein komplexes Gateway, mit dem komplizierte Bedingungen gekennzeichnet werden können.
Dabei werden exklusive Gateways mit einem X markiert, ereignisbasierte mit einem Fünfeck in einem doppelten Kreis, parallele mit einem $+$, inklusive mit einem O und komplexe mit einem $*$.

2.3.1.2. Pools und Swimlanes

Pools und Swimlanes (oder kurz Lanes) erlauben eine bessere Organisation von Geschäftsprozessen. Die Pools dienen dabei zur Abgrenzung der am Prozess beteiligten Parteien und repräsentieren meist eine Organisation, während die Lanes dabei helfen, die Struktur dieser Organisation abzubilden.

Ein Pool ist unterteilt in eine oder mehrere Lanes, die alle Aktivitäten enthalten. Dargestellt werden Pools in zwei Orientierungen, einer horizontalen und einer vertikalen. Dabei befindet sich der Titel des Pools am linken oder oberen Ende eines Rechtecks und enthält daneben bzw. darunter die einzelnen Lanes.
Ein Sonderfall sind sogenannte Black-Box-Pools oder eingeklappte Pools. Diese enthalten keinen Prozess und beschränken sich rein auf den Nachrichtenaustausch mit anderen Pools.

Lanes unterteilen die Pools und enthalten die Flussobjekte des Graphen. Ihre Darstellung ist gleich der der Pools. Sie haben jedoch keinen eigenen Rand, sondern fügen sich in den umgebenden Pool ein.
Eine Lane kann auch noch in weitere Unterlanes aufgeteilt werden, um die Struktur feiner darzustellen.

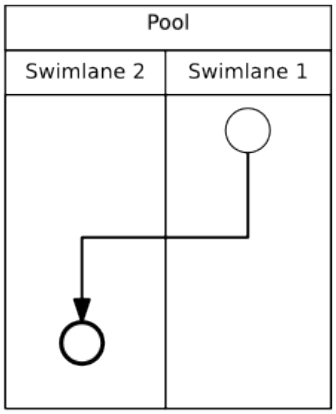

Abbildung 2.4.: Pool mit zwei Swimlanes

2.3.1.3. Verbindungsobjekte

Verbindungsobjekte sind die Kanten im Graphen. Sie verbinden die Flussobjekte. Zwischen zwei Aktivitäten ist die normale Verbindung ein Sequenzfluss, eine schwarze durchgezogene Linie mit ausgefülltem Pfeil.
Ausgenommen davon sind Verbindungen zwischen Aktivitäten zweier verschiedener Pools, bei denen ein Nachrichtenfluss verwendet werden muss. Dieser ist eine gestrichelte Linie mit nicht ausgefülltem Pfeil am Ende und einem leeren Kreis am Anfang.

Zur Verbindung von Flussobjekten und Artefakten gibt es zudem eine assoziative Verbindung, die aus einer gestrichelten Linie ohne Anfangs- oder Endpunkte besteht.

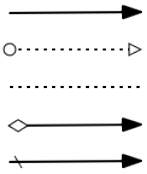

Abbildung 2.5.: Verbindungen

Der normale Sequenzfluss kann zudem noch mit Bedingungen verknüpft werden. Dazu gibt es zum Einen den bedingten Fluss, der mit einer Raute beginnt und eine Bedingung enthält, zum Anderen den Standardfluss, dessen Strich an einer Stelle durchgestrichen ist und der aktiviert wird, wenn alle anderen Bedingungen nicht zutreffen.

2.3.1.4. Artefakte

Artefakte werden verwendet um zusätzliche Informationen im Geschäftsprozess unterzubringen, die für den eigentlich Ablauf nicht benötigt werden.

Mit einer Anmerkung kann ein Beschreibungstext zu einem Flussobjekt verfasst werden und mit einer assoziativen Verbindung an diesem angebunden werden. Sie wird mit einem halben Rechteck dargestellt.

Mit Hilfe einer Gruppe können Aktivitäten thematisch gruppiert werden, ohne dabei Auswirkungen auf deren Ausführung zu haben. Diese werden durch eine abwechselnd gepunktete und gestrichelte Linie mit runden Ecken gekennzeichnet.

Datenobjekte bieten die Möglichkeit Daten zu lesen und zu schreiben. Ein normales Datenobjekt, repräsentiert durch das Symbol eines Dokumentes, repräsentiert dabei eine Information. Erweitert um drei Striche, wird daraus ein Listen-Datenobjekt, das eine Liste von Informationen enthalten kann. Um Daten aus externen Quellen zu erhalten, kann ein Datenobjekt mit einem nicht ausgefüllten Pfeil verwendet werden. Für die Ausgabe von Daten wird der Pfeil ausgefüllt, um ein Datenoutput-Objekt zu erhalten. Zur Verbindung von Datenobjekten und Flussobjekten wird zudem noch eine Datenassoziation hinzugefügt, die die assoziative Verknüpfung um eine Pfeilspitze erweitert.

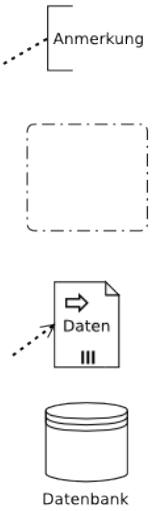

Abbildung 2.6.: Artefakte

Letzter Bestandteil der Artefakte ist die Datenbank. Sie erlaubt sowohl lesenden als auch schreibenden Zugriff auf Daten und ihre Daten sind auch nach Beendigung des Prozesses weiter verfügbar. Dargestellt wird Sie durch eine versinnbildlichte Datenbank.

2.3.2. Diagramme

Die vorgestellten Elemente werden in drei verschiedenen Diagrammtypen verwendet. Diese enthalten zum Teil noch weitere Elemente, die dann aber spezifisch für diese Diagramme sind.

2.3.2.1. Kollaborationsdiagramm

Das Kollaborationsdiagramm ist das Standarddiagramm von BPMN und seit der ersten Version Teil der Spezifikation. Es kann alle bereits vorgestellten Elemente enthalten und bildet daraus einen Prozessgraphen, der die Kollaboration zwischen verschiedenen Organisationen und deren Unterorganisationen abbildet. Dazu enthalten sie meist mehr als einen Pool um die einzelnen Teilnehmer am Prozess modellieren zu können.

2.3.2.2. Konversationsdiagramm

Mit Version 2.0 kam das Konversationsdiagramm neu hinzu. Es beschreibt die Verbindungen zwischen verschiedenen Organisationen während eines Prozesses auf einer höheren Ebene als das Kollaborationsdiagramm. Dadurch kann es als simple Übersicht dienen, während ein weiteres Diagramm die Details des Prozesses genauer beschreibt. Teilnehmer des Prozesses werden dabei als Black-Box-Pools dargestellt. Um die Konversationen darstellen zu können, werden drei neue Elemente eingeführt.

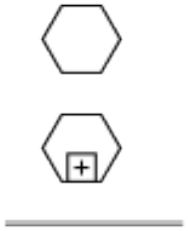

Abbildung 2.7.: Konversationsdiagramm

Ein Konversationsknoten ist ein Hexagon mit einer Beschriftung und beschreibt die Art der Konversation. Besteht das Hexagon aus einer dicken Linie, so ist dies eine global definierte Konversation, enthält es ein + so ist es eine Teilkonversation, die im aufgeklappten Zustand genauer definiert werden kann. Mit Hilfe einer Konversationsverbindung werden diese mit den beteiligten Pools verbunden. Die Verbindung wird durch eine doppelte Linie dargestellt.

2.3.2.3. Choreographiediagramm

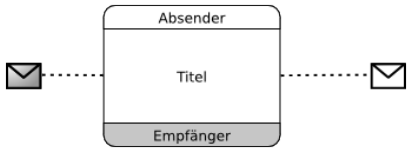

Abbildung 2.8.: Choreographiediagramm

Das zweite Diagramm, das mit Version 2.0 eingeführt wurde, ist das Choreographiediagramm. Es konzentriert sich auf die Interaktionen zwischen verschiedenen Prozessen und dem Nachrichtenfluss zwischen ihnen. Dabei liegt der Fokus auf den versendeten und empfangenen Nachrichten und deren Inhalt. Dazu werden für dieses Diagramm vier neue Elemente definiert.

Eine Choreographieaufgabe ist ein Schritt einer Choreographie und enthält Absender, Empfänger und Titel der Nachricht. Dazu wird die Darstellung einer normalen Aufgabe in drei Teile aufgeteilt, von denen jeder jeweils einen Part der Nachricht enthält. Um ein + in einem Quadrat erweitert, ergibt sich daraus ein Teilprozess, der die Choreographie verfeinern kann. Mit einem dicken Rand wird daraus eine Aufrufchoreographie, die global im Prozess definiert ist.
Ein Briefsymbol, das mit einer gepunkteten Linie mit einer Choreographieaufgabe verbunden ist, enthält den Inhalt der Nachricht und kann je nachdem, ob sein Inneres ausgefüllt oder leer ist, ausgehender oder eingehender Art sein.

2.3.3. Austauschformat

Seit der BPMN-Version 2.0 enthält der Standard neben der graphischen Notation auch ein XML-basiertes Austauschformat. Während zuvor von den BPMN-Werkzeugen verschiedene Eigenentwicklungen verwendet wurden, die miteinander zumeist nicht kompatibel waren, kann seit dem derselbe Graph in so gut wie jedem Programm verwendet werden. Dadurch wurde auch der direkte Einsatz in beliebigen Ausführungsprogrammen ermöglicht.

Das XML-Format besteht prinzipiell aus zwei verschiedenen Teilen. Im ersten Teil wird der Prozess und seine Eigenschaften beschrieben, während im zweiten Teil die Anzeige der einzelnen Knoten und Kanten des Prozessgraphen beschrieben wird. Der Hauptknoten der XML-Datei ist ein *definitions*-Tag, der den Prozess innerhalb eines *process*-Tags und die graphische Beschreibung in einem *BPMNDiagram*-Tag enthält.

Im *process*-Tag werden die einzelnen Elemente des Prozesses aufgelistet. Dabei enthält jedes Element zumindest ein *id*-Attribut, das spätestens im *BPMNDiagram* verwendet wird. Flussobjekte erhalten zudem noch ein *name*-Attribut, das einen Klarnamen enthält. Verbindungsobjekte enthalten dafür *sourceRef*- und *targetRef*-Attribute, die die Identifizierer ihrer Anfangs- und Endpunkte enthält.

Die Beschreibung der graphischen Darstellung innerhalb des *BPMNDiagram*-Tags besteht dagegen nur aus wenigen Elementen:
Mit einer *BPMNPlane* können verschiedene Ebenen dargestellt werden. Darin befinden sich *BPMNShape*-Tags, die über ihr *bpmnElement*-Attribut auf ein Element verweisen und über einen enthaltenen *Bounds*-Tag die Position auf der Zeichenebene angeben. Ebenfalls enthalten sind *BPMNEdge*-Tags, die die Positionierung der Kanten im Graphen angeben. Über das *bpmnElement*-Attribut kann der Identifizierer der entsprechenden Kante des Prozesses angegeben werden. Um nicht nur gerade Linien zu erhalten können innerhalb ein oder mehrere *Waypoint*-Tags angegeben werden, die Punkte zwischen den Elementen enthalten.

3. State of the Art

Im dritten Kapitel soll ein Überblick über momentan aktuelle Entwicklungen bei Ansätzen zur Validierung von Compliance-Anforderungen gegeben werden. Dabei ist das Hauptkriterium für die Auswahl der Verfahren das Vorhandensein einer grafischen Darstellung, da die Erstellung einer solchen das Ziel dieser Arbeit darstellt. Einzige Ausnahme davon ist der Compliance Descriptor, der im vierten Abschnitt dieses Kapitels behandelt wird und momentan noch über keine solche Darstellung verfügt. Davor wird im ersten Abschnitt eine Erweiterung für die Abfragesprache BPMN-Q vorgestellt, der zweite Abschnitt beschäftigt sich mit einer weiteren Abfragesprache, SeaFlows, während im dritten Abschnitt mit CoReL direkt Anforderungen formuliert werden. Ein Vergleich der Ansätze findet am Ende des Kapitels statt.

3.1. BPMN-Q

BPMN-Q[1] ist eine am Hasso-Plattner-Institut in Potsdam entwickelte graphische Abfragesprache, die einen Abfragegraphen mit einem oder mehreren Prozessgraphen vergleicht. Dabei wird eine zu BPMN sehr ähnliche Darstellung gewählt, um die Erstellung von Abfragen für Anwender von BPMN möglichst simpel zu halten.
In [2][3][4] wird die Sprache dazu verwendet, um Compliance-Anforderungen zu modellieren und Prozessmodelle auf deren Einhaltung hin zu überprüfen. Dazu werden einige Erweiterungen für BPMN-Q spezifiziert, die dabei helfen die Ausführungsreihenfolge und die Auswirkung von Datenkonditionen besser abfragen zu können.

3.1.1. Abfragesprache

Bei der Anwendung eines BPMN-Q-Graphen auf ein Prozessmodell wird die Abfrage mit Hilfe eines Abfrageverarbeiters interpretiert und das Modell durchsucht. War die Abfrage erfolgreich wird der gefundene Teilgraph zurückgegeben.
Um die Zahl der möglichen Anfragen zu erhöhen, wird das Arsenal an BPMN-Elementen um einige abfragespezifische Elemente erweitert:

Abbildung 3.1.: a) BPMN Elemente b) BPMN-Q Elemente (aus [1])

- **Generisches Element** Findet jeden beliebigen Typ von Element. Dazu werden bei der Ausführung des Graphens alle möglichen Elemente an dieser Stelle eingesetzt und geprüft.
- **Split** Generisches Element, das alle möglichen Arten von Gateways findet. Das Split-Element findet dabei den Beginn einer Aufspaltung.
- **Join**
  Als Gegenstück zum Split-Element findet es das Ende einer Aufspaltung.
- **Anonyme Aktivität**
  Beginnt der Name einer Aktivität mit einem @-Zeichen, so ist dies eine unspezifizierte oder anonyme Aktivität. Die Abfrage liefert dafür alle Aktivitäten, die sich an entsprechender Stelle im Graph befinden, zurück.
- **Kante mit Ausschluss**
  Schließt die spezifizierte Aktivität davon aus, am Ende der Kante positioniert zu sein.
- **Pfad**
  Das Pfad-Element sucht nicht nach einer Aktivität, sondern nach allem, was sich im Prozessmodell zwischen dem Anfang des Pfads im Abfragegraph und seines Endes befindet. Verbindet die Pfadabfrage z.B. eine Aktivität A mit einer Aktivität B, so gibt die Suche alle Elemente, die zwischen A und B liegen sowie deren Verknüpfungen zurück.
- **Pfad mit Ausschluss**
  Wie das bereits bekannte Pfad-Element, schließt jedoch bei der Pfadabfrage alle Pfade aus, die die ausgeschlossene Aktivität enthält.

Wie dieser Graph bei der Anwendung umgewandelt und ausgeführt wird, wird im Abschnitt "Validierung"näher beschrieben.

3.1.2. Erweiterungen

Pfaderweiterungen

Wenn mit BPMN-Q Compliance-Anforderungen formuliert werden, ist durch das erfolgreiche Finden eines Subgraphen noch nicht garantiert, dass die Anforderung auch wirklich erfüllt wurde.

So ist noch nicht sichergestellt, dass der gefundene Pfad auch bei jedem möglichen Szenario ausgeführt wird, da die Ausführung von Kontrollknoten im Prozessmodel nicht von BPMN-Q abgedeckt wird.
Zum Zweiten wird für die Abdeckung aller Möglichkeiten von Complianceanforderungen die Fähigkeit benötigt, die Ausführungsrichtung zweier Aktivitäten angeben zu können. Auch dies wird von Standard-BPMN-Q nicht bereitgestellt.

Um das erste Problem zu lösen wird ein Modellüberprüfer eingesetzt, der den Subgraph, der nach Anwendung des BPMN-Q Graphen entsteht, überprüft.
Durch die Einführung zweier Erweiterungen kann das zweite Problem gelöst werden:

- $\ll$ *precedes* $\gg$
  Gibt an, dass eine Aktivität vor der anderen ausgeführt werden muss.

- $\ll$ *leads to* $\gg$
  Gibt an, dass eine Aktivität auf jeden Fall nach der anderen ausgeführt werden muss.

Auch wenn beide Erweiterungen auf den ersten Blick ziemlich gleich wirken gibt es doch einen wichtigen Unterschied: wenn z.B. A und B durch ein Oder-Gateway getrennt sind, dann geht zwar A B voraus (*precedes*), A führt aber nicht zu B (*leads to*), da nicht garantiert ist, dass B auch ausgeführt wird.

Diese Erweiterungen werden zu Pfaden hinzugefügt, weshalb zusätzlich die Regeln dieses Pfades gelten. A und B müssen also nicht direkt aufeinander folgen, sondern müssen nur entlang eines Pfades des Prozessmodells liegen.

Datenkonditionen

Um auch die Auswirkung von Datenkonditionen auf die Einhaltung von Compliance-Anforderungen überprüfen zu können, muss dieser Aspekt von BPMN auch in BPMN-Q integriert werden [3]. Dazu werden zwei verschiedene Typen von Datenkonditionen hinzugefügt, die zu vier verschiedenen Überprüfungen führen. Der erste Typ ist eine eingehende Verbindung von einer Datenkondition zu einer Aktivität, der Zweite eine ausgehende Verbindung von einer Aktivität zu einer Datenkondition.
Die vier dadurch möglichen Überprüfungen sind:

- **Datenregel**
  Eine eingehende Verbindung von einer Datenkondition zu einer Aktivität. Die Kondition muss erfüllt sein, bevor die verbundene Aktivität ausgeführt wird, um die Regel erfüllen zu können.

- **Bedingtes** *leads to*
  Dazu muss die Aktivität zu Beginn des *leads to* eine ausgehende Verbindung zu einer Datenkondition haben. Dies bedeutet, dass nur wenn die Kondition erfüllt ist, die Aktivität am Ende des *leads to* ausgeführt werden muss.

- **Bedingtes** *precedes*
  Analog zum bedingten *leads to* muss hier die Kondition einmal zugetroffen haben, bevor die Aktivität am Ende der Kante ausgeführt wird. Dabei darf sich die Kondition auch vor der Ausführung dieser Aktivität wieder ändern.

- **Bedingter Ausschluss**
  Analog zu den zwei bedingten Varianten trifft hier eine Datenkondition auf eine Kante, die die Ausführung einer oder mehrer Aktivitäten ausschließt. Dadurch kann ausgeschlossen werden, dass eine Aktivität nicht ausgeführt wird, sobald eine Datenkondition erfüllt wird.

3.1.3. Validierung

Um ein Prozessmodell durch einen BPMN-Q-Graphen validieren zu können, muss im ersten Schritt natürlich der Abfragenverarbeiter aufgerufen werden. Dadurch sollte man für jede BPMN-Q-Abfrage einen Subgraphen erhalten. Ist dies für eine Abfrage nicht der Fall wurde die Compliance-Anforderung nicht erfüllt.

Um für die übrigen Validierungsschritte die Komplexität zu verringern wird im zweiten Schritt der erhaltene Graph reduziert. Dazu werden als erstes alle Aktivitäten entfernt, die für die Abfrage der Regel nicht betrachtet werden, also nicht im BPMN-Q Graphen enthalten sind. Anschließend werden leere und nicht benötigte Gateways und Schleifen, nicht benötigte Startaktivitäten sowie einige BPMN-spezifische Aktivitäten entfernt.

Im dritten Schritt wird der BPMN-Q-Graph und die *leads to* und *precedes* Erweiterungen in einen Linear Temporal Logic (LTL) Ausdruck umgewandelt. Linear Temporal Logic besteht aus atomaren Ausdrücken und den logischen Konnektoren $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\Leftrightarrow$ und erweitert das Ganze um temporale Ausdrücke wie *immer*, *eventuell*, *nächstes* und *bis*. Die Past Linear Temporal Logic (PLTL), die hierbei verwendet wird, kehrt diese temporalen Ausdrücke zusätzlich in die Vergangenheit um und fügt die Ausdrücke *immer in der Vergangenheit*, *einmal in der Vergangenheit*, *letztes* und *seit* hinzu.
Bei der Umwandlung wird der gesamte Ausdruck mit dem Equivalent von *immer* umgeben und innerhalb davon werden alle Pfadkonstrukte mit $\wedge$ verknüpft. Ein $A\,leads\,to\,B$ Pfad wird zu B folgt *eventuell* auf A und ein $A\,precedes\,B$ wird zu A *kam einmal in der Vergangenheit vor* B.

Im nächsten Schritt wird aus dem reduzierten Graphen eine endliche Zustandsmaschine generiert. Sie wird für die spätere Modellüberprüfung benötigt. Dafür wird ein Ansatz aus [10] verwendet, um zuerst ein Petrinetz zu erstellen, aus welchem danach die Zustandsmaschine erstellt wird.
Zur Erstellung des Petrinetzes werden Teile des BPMN-Graphen auf Teile eines Petrinetzes abgebildet, wobei Datenkonditionen zu eigenen Zuständen im Netz führen, die extra betrachtet werden müssen. Über den Erreichbarkeitsgraphen des Petrinetzes wird anschließend die Zustandsmaschine generiert.

Als letzter Schritt wird die Modellüberprüfung mit Hilfe der endlichen Zustandsmaschine und des PLTL-Ausdrucks durchgeführt. Dazu wird bei diesem Ansatz NuSMV[1] verwendet, ein Prüfer, der an der Carnegie Mellon University entwickelt wurde.

[1] http://nusmv.fbk.eu/

3.1.4. Graphischer Editor

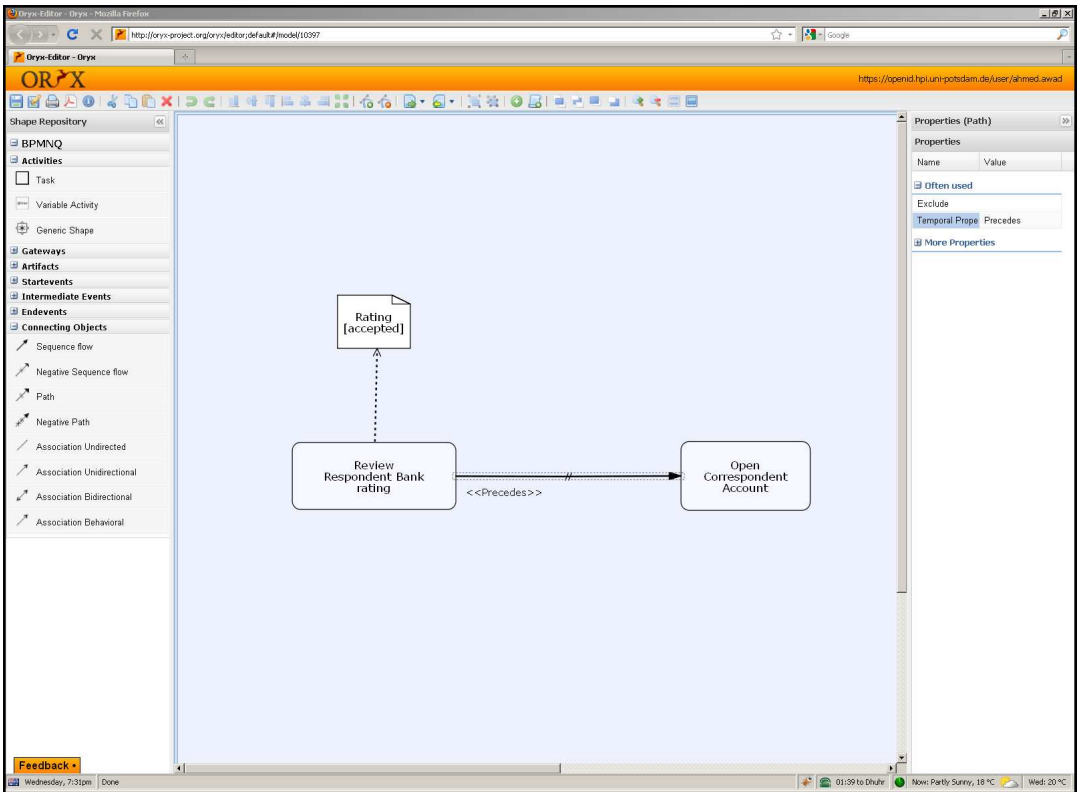

Abbildung 3.2.: Oryx-Editor mit BPMN-Q Graph (aus [4])

Um den BPMN-Q Graphen zu gestalten, wurde eine Erweiterung für den Oryx-Editor entwickelt, die neben der Erstellung von Anfragen auch die Validierung der Graphen und die Anzeige der gefundenen Fehler erlaubt.
Oryx wurde ebenfalls am Hasso-Plattner-Institut zur Erstellung von BPMN-Diagrammen entwickelt. Da große Teile dieser Arbeit ebenfalls auf dem Oryx-Editor aufbauen, wird darauf später eingegangen.

3.2. SeaFlows

SeaFlows [26][18][25] war ein Forschungsprojekt an der Universität Ulm, das in den Jahren 2005 bis 2011 durchgeführt wurde. Es wurden in dessen Rahmen Werkzeuge für die Überprüfung von Compliance-Anforderungen in Geschäftsprozessen entwickelt und zwar sowohl grafische Tools zur Modellierung der Anforderungen als auch zu deren Validierung. Grafisch dargestellt wird das Ganze mit Hilfe des sogenannten Compliance Rule Graph (CRG). Dessen Validierung kann während der Erstellung des Geschäftsprozesses durchgeführt werden.

Um Compliance-Anforderungen zu formulieren, wird von SeaFlows der CRG definiert, welcher im folgenden Abschnitt beschrieben wird. Auf die Validierung des CRG wird im zweiten Abschnitt eingegangen. Seine Anwendung durch den grafischen Editor auf Basis von Eclipse wird im dritten Abschnitt näher betrachtet.

3.2.1. Compliance Rule Graph (CRG)

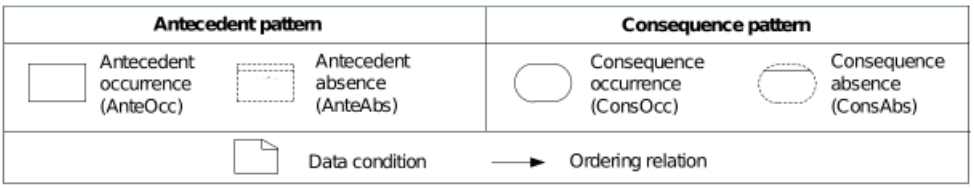

Abbildung 3.3.: Bestandteile des Compliance Rule Graphen (aus [26])

Der CRG besteht aus sechs verschiedenen Bestandteilen, deren Anordnung die erwünschte Reihenfolge ihrer Ausführung angibt. Ein CRG besteht dabei aus mehreren Subgraphen, die jeweils eine Regel formulieren. Die Elemente des Graphen verweisen auf Aktivitäten des Businessprozesses und werden durch (Nicht-)Ausführung bzw. die An- oder Abwesenheit des Elements aktiviert:

- **Vorangehendes Auftreten**
  Die durch das Element angegebene Aktivität muss zur Aktivierung der Regel aufgetreten sein.

- **Vorangehende Abwesenheit**
  Als Negierung des vorherigen Elementes wird die Regel hier durch die Abwesenheit der angegebenen Aktivität aktiviert.

- **Auftretende Auswirkung**
  Nach der Aktivierung einer Regel muss diese Aktivität auftreten bzw. nach Ende der Regel aufgetreten sein.

- **Abwesende Auswirkung**
  Negiert das vorherige Element, was bedeutet, dass die Auswirkung während der Regel nicht aufgetreten sein darf.

- **Ordnungsrelation**
  Gibt durch die Richtung des Pfeiles die Reihenfolge an, in denen die Aktivitäten im Prozess auftreten müssen bzw. abwesend sein müssen.

- **Datenkondition**
  Erweitern das Auftreten einer Aktivität bzw. die Auswirkung eines Teilgraphen um eine Kondition. Sie enthalten dazu einen booleschen Ausdruck, der bei Erreichen des Elements

ausgewertet werden kann. Dadurch können vor allem die Einstiegs- und Auswirkungsbedingungen genauer spezifiziert werden. So können Subgraphen beispielweise nur evaluiert werden, wenn ein bestimmter Grenzwert erreicht wird.

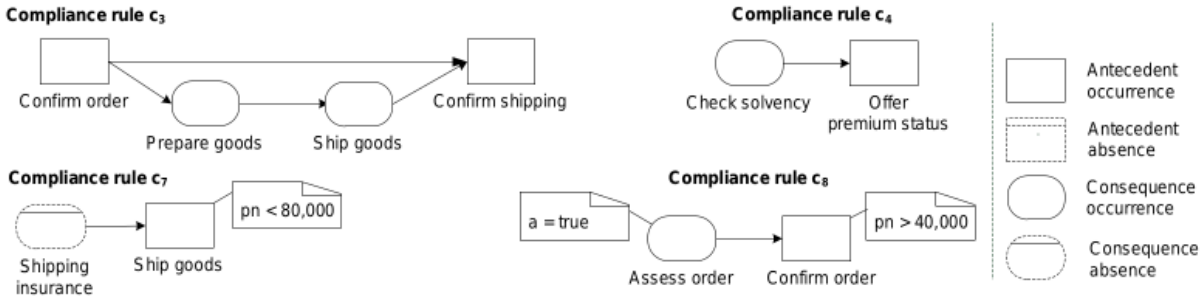

Abbildung 3.4.: Beispiele des Compliance Rule Graphen (aus [26])

Abbildung 3.4 zeigt 4 Beispielgraphen, anhand derer sich das Prinzip des CRG recht einfach zeigen lässt:
Die Regel c_3 bedeutet, dass zwischen der Bestätigung der Bestellung und der Verstandbestätigung die Ausführung der zwei Aktivitäten "Prepare goods"(Vorbereitung der Güter) und SShip goods"(Versand der Güter) stattfinden muss. Sie wird dabei durch das Auftreten der Bestätigungen aktiviert, woraufhin das Auftreten der beiden anderen Aktivitäten kontrolliert werden kann.
c_4 zeigt, dass bei einer Regel nicht das erste Element zur Aktivierung führen muss. Dabei wird überprüft, dass wenn Premiumstatus angeboten wird, die Zahlungsfähigkeit des Kunden zuerst geprüft werden muss.
c_7 beinhaltet sowohl ein negiertes Element als auch eine Datenkondition. Damit wird beschrieben, dass beim Versand von weniger als 80.000 Teilen (pn) keine Versandversicherung abgeschlossen werden muss.
Regel c_8 zeigt auf beiden Seiten Datenkonditionen. Hier wird geprüft, dass bei einer Bestellung von mehr als 40.000 Teilen (pn) zuerst eine Bestätigung (a) vom Kunden geholt werden muss und auf eine positive Antwort überprüft wird.

3.2.2. Validierung

Um die Validierung von Prozessmodellen gegen den CRG durchzuführen, werden von SeaFlows zwei verschiedene Verfahren angewendet. Die Ausführungsreihenfolge und Struktur des Modells wird mit Hilfe eines strukturellen Complianceprüfers geprüft. Für die Prüfung der Auswirkung von Datenkonditionen und Daten generell wird ein verhaltensbezogener Complianceprüfer eingesetzt.

Struktureller Complianceprüfer

Zur strukturellen Prüfung wird der CRG im ersten Schritt automatisch um fünf Strukturkriterien ergänzt. Dazu werden die einzelnen Elemente des Graphen und deren Relation zueinander analysiert.
Die fünf Kriterien sind:

- **Enthält** A
  Das Prozessmodell muss die Aktivität A enthalten.

- A **schließt** B **aus**
  Überprüft das Prozessmodell auf die Position von A und B. Dieses Kriterium trifft zu, wenn sich beide auf unterschiedlichen Zweigen eines exklusiven Knoten beziehen.

- A **impliziert** B
  Für alle Vorkommnisse von A und B im Prozessmodell muss gegeben sein, dass sich beide immer auf derselben Seite von Zweigen eines exklusiven Knoten befinden.

- A **impliziert** $B_1, B_2, ..., B_n$
  Überprüft das Prozessmodell auf diesselbe Weise wie das vorhergehende Kriterium. Dabei werden jedoch nicht nur zwei Aktivitäten überprüft, sondern eine Liste von Aktivitäten.

- A **geht** B **voran**
  Bei diesem Kriterium wird überprüft, ob es einen gerichteten Pfad von A nach B im Prozessmodell gibt.

Im zweiten Schritt wird das Prozessmodell auf die gefundenen Kriterien hin untersucht. Fehler werden anschließend in einem dritten Schritt gesammelt und für den Ersteller des Prozesses aufbereitet und dargestellt. Dieser kann auf Grundlage der Fehlermeldungen entscheiden, wie er die Einhaltung der Complianceanforderungen innerhalb des Prozesses gewährleistet.

Verhaltensbezogener Complianceprüfer

Während der strukturelle Prüfer das Vorhandensein und die Reihenfolge von Elementen prüfen kann, untersucht der verhaltensbezogene Prüfer den Einfluss von Daten auf das Verhalten des Prozessmodells und damit auch dessen Einfluss auf den CRG.
Daten haben in zweierlei Hinsicht Einfluss auf den CRG: Zum einen können Elemente des CRG Datenkonditionen enthalten, die von Daten des Prozessmodells beeinflusst werden. Zum anderen können auch Elemente ohne Datenkondition von Daten beeinflusst werden, indem z.B. die verknüpfte Aktivität im Modell hinter einem durch Daten beeinflussten Knoten sitzt.

Um bei der Überprüfung nicht das komplette Modell durchsuchen zu müssen, wird ein abstraktes Prozessmodell sowie ein abstrakter CRG erstellt. Dieses ist, was Daten angeht, kompakter und somit leichter überprüfbar. Dazu werden zuerst die Daten identifiziert, die für den CRG relevant sind und die für datenbasierte Knoten benötigt werden. Um das Prüfen dieser Daten zu vereinfachen wird zudem deren Dimension reduziert. Statt z.B. bei einer Zahl x auf jede mögliche Zahl $1, 2, ..., n$ zu prüfen, wird eine Regel erstellt, die dies auf Grundlage der vorgefundenen Konditionen z.B. auf $x > 5 \wedge x < 10$ vereinfacht.

Mit Hilfe von Modellüberprüfungstechniken wird anschließend die Prüfung des Modells durchgeführt. Im Falle von SeaFlows wird der SAL (Symbolic Analysis Laboratory) Modellüberprüfer[5] verwendet, der vom Stanford Research Institute entwickelt wurde.

3.2.3. Grafischer Editor

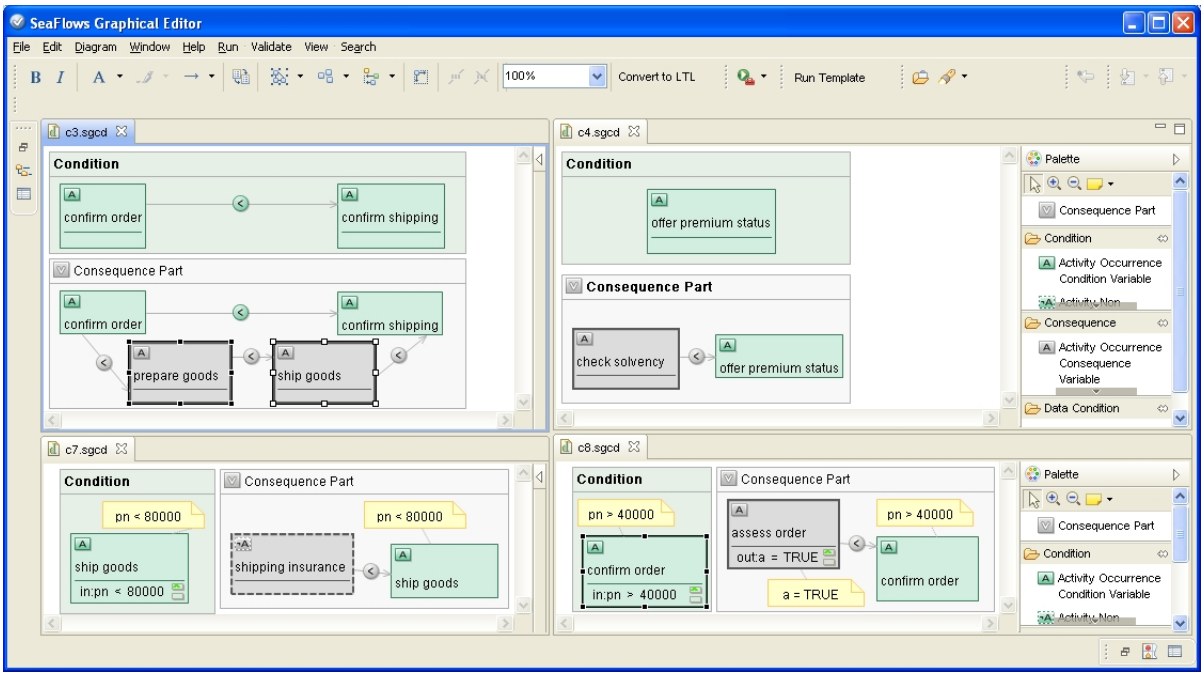

Abbildung 3.5.: SeaFlows Graphical Editor (aus [26])

Um den CRG leichter modellieren zu können, wird von SeaFlows ein Editor zur Verfügung gestellt. Dieser wurde auf Basis des Eclipse Modeling Framework[2] und des Eclipse Graphical Modeling Framework[3] entwickelt und ermöglicht es, Aktivitäten aus dem Businessprozess auszuwählen und im CRG zu verwenden. Im Editor werden zuerst die Konditionen der Regel, die zur Aktivierung führen, definiert und anschließend mit den Auswirkungen im zweiten Teil des Editors verbunden. Nach Fertigstellung eines Graphens werden einzelne XML-Dateien für jeden Subgraphen erstellt und exportiert.

Integriert wird zudem die kommerzielle Prozessmanagementlösung AristaFlow BPM Suite[4]. Damit werden zum einen die Aktivitäten ausgelesen, die bei der Erstellung eines CRG ausgewählt werden können, zum anderen wird auch die Validierung des Prozesses durchgeführt und es werden die entsprechenden Fehlermeldungen angezeigt.

[2] https://www.eclipse.org/modeling/emf/
[3] https://www.eclipse.org/modeling/gmp/
[4] http://www.aristaflow.com

3.3. CoReL

Die Compliance Representation Language (CoReL)[14][13] ist eine grafische Modellierungssprache für Compliance-Anforderungen, die an den Universitäten von Luxemburg und Osnabrück entwickelt wurde und 2011 zum ersten Mal veröffentlicht wurde.
Die Sprache kann sowohl während der Modellierung von Geschäftsprozessen als auch während deren Ausführung evaluiert werden. Diese Evaluierung wird auf Grund von Regeln durchgeführt, die in einer Vielzahl von Sprachen formuliert werden. Durch diese Flexibilität sollen sich möglichst viele verschiedene Arten von Anforderungen formulieren lassen.

Trotz Verwendung eines grafischen Modells ist kein grafischer Editor für CoReL bekannt. Zudem wurde kein Austauschformat definiert, dass das Modell zwischen Editor und einem Validierungssystem transportieren könnte.

3.3.1. Grafisches Modell

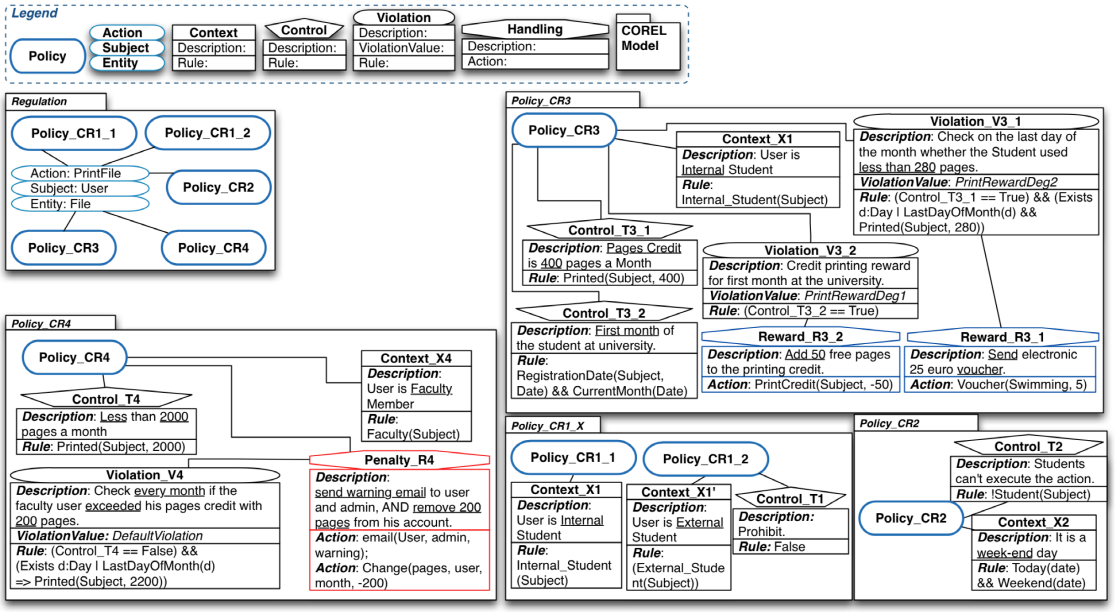

Abbildung 3.6.: CoReL Beispielgraph (aus [14])

Das Hauptkonzept von CoReL ist das Regelwerk. Es modelliert eine Complianceregel als Entscheidung, ob eine Aktion erlaubt, forciert oder verboten werden soll. Das Regelwerk hat einen einzigartigen Identifizierer und kann über Metadaten zusätzliche Informationen enthalten.

Ein sogenanntes ASE-Tripel (ASE steht dabei für *Action, Subject, Entity*) verbindet CoReL mit Aktivitäten eines BPMN-Geschäftsprozesses. Während die Aktion ein Pflichtfeld ist, können die

beiden anderen Felder auch leer gelassen werden, da es sich dabei auch um eine automatisch ausgeführte Aktivität handeln könnte. Durch Verbindungen mit einem oder mehreren Regelwerken können Konditionen für die Ausführung der Aktion gestellt werden.

Um ein Regelwerk zu beschreiben, können vier verschiedene Typen von Modellen verwendet werden. Alle vier enthalten dabei zumindest eine Beschreibung, die die enthaltenen Regeln in Klartext wiedergeben. Drei Modelle enthalten zusätzlich Regeln, die in einer nahezu beliebigen Sprache formuliert werden können. Dazu wird die Sprache im Namen der Regel angegeben. Dieser folgt dem Schema $R^{<Name>}_{<Sprache>}$.

Die vier Modelle sind:

- **Kontext**
  Das Kontext-Modell, modelliert den Zustand, in dem das System sein muss, damit das Regelwerk angewendet werden kann bzw. muss.

- **Kontrolle**
  Kontroll-Modelle werden dazu verwendet, um durch Regeln die Complianceanforderungen zu formulieren.

- **Verletzung**
  Verletzungen der Anforderung können explizit modelliert werden. Dazu kann in der Regel auch das Ergebnis von Kontroll-Modellen abgefragt werden. Trifft die Regel zu, so kann durch das Verletzungs-Modell ein angegebener Wert gesetzt werden. Dadurch können zum Beispiel verschiedene Stufen von Verletzungen eingeführt werden.

- **Behandlung**
  Das Behandlungs-Modell wird verwendet, um auf Verletzungen der Anforderung reagieren zu können. Dabei sind sowohl positive als auch negative Behandlungen möglich. Zur Unterscheidung wird die Randfarbe des Symbols verändert, rot für negativ, grün für positiv.

Dadurch, dass zur Formulierung von Regeln auf bereits definierte Sprachen gesetzt wird, ist das grafische Modell weniger komplex als in den zwei vorherigen Verfahren. Das hat aber den Vorteil, dass Anforderungen und Konsequenzen im gleichen Graph modelliert werden und die Zusammenhänge der einzelnen Regeln klarer werden.

3.3.2. Validierung

Zur Validierung verwendet CoReL zwei verschiedene Ansätze:
Um während der Modellierung eines Geschäftsprozesses eine Validierung durchzuführen, wird eine statische Prüfung durchgeführt. Diese läuft analog zu den Verfahren ab, die in den Kapiteln 3.1 und 3.2 beschrieben wurden. Durch die Vielzahl der unterstützten Sprachen wird versucht, diese auf gemeinsames Modell herunterzubrechen. Dazu werden die enthaltenen Regeln mit Hilfe entsprechender Serialisierer in das Metamodell übertragen und anschließend mit einem Modellüberprüfer validiert.

Auf die Validierung während des Ablaufens des Geschäftsprozesses wird hier ein größerer Wert gelegt. Dazu wird ein Überprüfungssystem definiert, dass in die Arbeitsablaufausführung integriert

wird.
Während der Arbeitsablauf stattfindet, wartet es auf das Eintreten bestimmter Konditionen, die durch die ASE-Tripel definiert wurden. Ist dies der Fall, so werden die Kontexte der verbundenen Regelwerke überprüft. Evaluiert der Kontext zu *wahr*, dann werden als nächstes die Regeln der Kontroll-Modelle überprüft.
Auf Grund deren Resultate werden als nächstes die Verletzungen kontrolliert und eventuelle Behandlungen durchgeführt. Je nach Resultat aller Überprüfungen wird ein Ergebnis an die Arbeitsablaufausführung geliefert.
Waren alle Kontrollen erfolgreich, so wird der Prozess fortgeführt. Trat eine Verletzung auf, so ist die Fortführung abhängig von der Konfiguration und des Wertes der Verletzung. Gibt es beispielsweise die Werte *Warnung* und *Fehler*, so könnten Verletzungen mit dem Wert *Warnung* ignoriert werden und der Prozess weiterlaufen.

3.4. Compliance Descriptor

Der Compliance Descriptor[23][22][15] wurde bis 2015 an der Universität Stuttgart und am Fraunhofer Institut für Arbeitswissenschaft und Organisation (IAO) im Rahmen des "Compliance Management in Adaptive Business Processes"(Co.M.B)-Projektes entwickelt.

Der Compliance Descriptor hebt sich durch seine Flexiblität hervor. Er soll für möglichst viele Anwendungsfälle einsetzbar sein. Während andere vorgestellte Verfahren nur während der Designphase oder während der Ausführung des Geschäftsprozesses angewendet werden können, kann der Compliance Descriptor auch während der Auslieferung der verwendeten Anwendungen und deren Infrastruktur zum Einsatz kommen. Es besteht zudem die Möglichkeit später weitere Phasen hinzuzufügen.
Dazu werden beim Compliance Descriptor, ähnlich wie bei CoReL, keine Abfragen formuliert, sondern die Compliance-Anforderungen an sich. Die eigentlichen Regeln können in verschiedenen Sprachen geschrieben werden. Im momentanen Zustand korrespondiert eine Sprache mit einer Phase, die überprüft werden soll.

Ein weiterer Unterschied stellt die Einbindung von Gesetzes- und Anforderungstexten direkt in den Compliance Descriptor dar. Dadurch kann sowohl beim Editieren als auch beim Validieren direkt Bezug auf die relevanten Texte genommen werden.

Im Gegensatz zu den bereits vorgestellten Verfahren hat der Compliance Descriptor noch kein grafisches Modell und verfügt deshalb auch noch über keinen Editor. Deswegen wird hier nur auf die durch das XML-Format zur Verfügung gestellte Funktionalität sowie deren Validierung eingegangen.

3.4.1. Funktionalität

Einstiegspunkt beim Compliance Descriptor sind die sogenannten Anforderungen. Diese stellen die Compliance-Anforderungen dar, die vom Geschäftsprozess erfüllt werden sollen. Dafür enthalten Anforderungen sogenannte Compliance Expressions, mit deren Hilfe die Compliance-Anforderungen formuliert werden. Die Ausdrücke enthalten Regeln und Einheiten und verknüpfen diese mit Operatoren.

Einheiten sind indirekte Verweise auf Teile eines Prozesses. Dies kann eine Aktivität des Geschäftsprozesses, ein Teil der Infrastruktur oder ähnliches sein. Über ihren Namen und Typ werden sie verbunden.

Regeln sind dabei als Teil von Anforderungen zu verstehen. Dazu werden die Compliance-Anforderungen in individuelle und einzeln zu prüfende Teile getrennt und jeweils als Regel implementiert. Über einstellbare Phasen und Sprachen kann beeinflusst werden, wann und wie Regeln ausgeführt werden. Dadurch lassen sich die Regeln einer Anforderung über mehrere Phasen verteilen und in mehreren Anforderungen verwendet werden.
Zudem können Regeln einen Variability Descriptor[27] enthalten. Dieser wurde ebenfalls an der Universität Stuttgart entwickelt und dient dazu, abstrakte Regeln formulieren zu können. Wird z.B. von einer Regel der zeitliche Abstand zwischen zwei Aktivitäten im Geschäftsprozess gemessen und geprüft, so kann diese Regel formuliert werden ohne den genauen Abstand bzw. die genauen Aktivitäten zu kennen. Diese können der Regel beim Aufruf als Parameter übergeben oder als Konstante gesetzt werden und werden durch den Variability Descriptor an der korrekten Stelle eingesetzt.

Um diese sogenannten Variability Points einsetzen zu können, werden innerhalb der Regel Bindungen definiert. Diese enthalten neben dem zu ersetzenden Punkt den einzusetzenden Wert bzw. die Nummer des Parameters der dafür verwendet werden soll. Parameter werden der Regel innerhalb der Compliance Expression übergeben.

Abgeschlossen wird der Compliance Descriptor durch das Gesetz. Es enthält den kompletten Text eines Gesetzes, einer Anordnung oder einer Richtlinie als XHTML-Dokument. Dadurch kann mit einem XPath[39]-Ausdruck, einer Abfragesprache für XML-Dateien, auf einzelne Absätze des Dokuments verwiesen werden. Diese sind Teil der Anforderung.

3.4.2. Validierung

Um die Validierung eines Geschäftsprozesses durchführen zu können, wird eine Reihe von Technologien eingesetzt. Diese unterscheiden sich je nach eingesetzter Sprache und der Phase, in der sie überprüft werden sollen.
Während generell Compliance-Anforderungen und damit auch Anforderungen des Compliance Descriptors überprüft werden sollen, können diese Regeln aus verschiedenen Phasen kombinieren und lassen sich deshalb nicht gleichzeitig prüfen. Deshalb werden Regeln einzeln überprüft und dafür an ihre entsprechende Komponente übergeben, um zur korrekten Zeit geprüft werden zu können. Dazu werden die Compliance Expressions aller Anforderungen auf Regeln durchsucht und alle einzigartigen Kombinationen von Regel und Parametern gesammelt. Von diesen Kombinationen wird die finale Regel durch Auflösen des Variability Descriptors erstellt. Da zu diesem Zeitpunkt alle Parameter bekannt sind, können die variablen Punkte einfach durch ihre Werte ersetzt werden. Anschließend werden alle so erstellten finalen Regeln an ihre entsprechende Komponente übergeben.

Um einen kompletten Bericht über die Einhaltung der Compliance-Anforderungen zu erhalten, muss abgewartet werden, bis der Geschäftsprozess beendet ist. Dies ist meist nicht erwünscht, da so Fehler während der Modellierung erst spät auffallen würden. Deshalb können vom Compliance Descriptor drei verschiedene Arten von Berichten generiert werden:

- **Lebenszyklusschritt-Bericht**
  Generiert einen Bericht nach dem Abschluss eines bestimmten Schritts im Lebenszyklus des Geschäftsprozesses. So lässt sich beispielsweise ein Bericht während oder nach der Designphase erstellen, mit Hilfe dessen Modellierungsfehler rechtzeitig korrigiert werden können. Dafür werden die Resultate aller noch nicht validierten Regeln als wahr interpretiert um ein Ergebnis zu erhalten, dass sich rein auf den betreffenden Schritt bezieht.

- **Zwischenbericht**
  Wird generiert, während noch nicht alle Regeln validiert wurden. Dadurch kann bereits gesehen werden welche Anforderungen fehlschlugen und welche noch gar nicht oder nur teilweise ausgeführt wurden. Regeln, die noch nicht validiert wurde, werden als unbekannt interpretiert. Das Ergebnis von nicht vollständig fertiggestellten Anforderungen ist deshalb ebenfalls unbekannt.

- **Vollständiger Bericht**
  Nach Abschluss des gesamten Geschäftsprozesses wird ein vollständiger Bericht generiert, der das Ergebnis aller Regeln beinhaltet.

Um den Geschäftsprozess während der Designphase zu testen, wird die Linear Temporal Logic (LTL) verwendet, wie sie bereits in Kapitel 3.1 und 3.3 beschrieben wurde. Analog dazu findet auch die Validierung des Geschäftsprozesses statt.

Für die Überprüfung der Auslieferung wird Topology and Orchestration Specification for Cloud Applications (TOSCA)[6] verwendet, ein Industriestandard, der es erlaubt, Applikationen unabhängig von der gewählten Cloudplattform zu beschreiben und auszuliefern. Mit einer Erweiterung die TOSCA um Regelwerke ergänzt [41], können Anforderungen an die Auslieferung gestellt werden. Dadurch lässt sich z.B. der Ort des Rechenzentrums, an die die Auslieferung stattfinden soll, einschränken. Für die Überprüfung wird ein solches Regelwerk generiert und bei Auslieferung von TOSCA angewendet.

Zur Validierung während der Laufzeit des Geschäftsprozesses wird die Process Goal Modeling Language (ProGoalML) [21] verwendet. Mit Hilfe von ProGoalML lassen sich Geschäftsprozesse um Messpunkte erweitern, mit denen Key Performance Indicators (KPI) berechnet und Ziele definiert werden können. Durch das Verwenden der Ziele zur Überprüfung von KPIs kann ProGoalML damit zur Validierung von Complianceanforderungen verwendet werden. Für das Aufzeichnen und Überwachen der von ProGoalML benötigten Werte wird aPro verwendet, eine Monitoringlösung, die als Webanwendung ausgeliefert werden kann und den Geschäftsprozess überwacht und über die Verletzung eines Ziels informieren kann.

3.5. Zusammenfassung

	BPMN-Q	**SeaFlows**	**CoReL**	**Compliance Descriptor**
Modellierungs-sprachen	BPMN	BPMN	BPMN	BPMN
Regelsprachen	PLTL	Eigene / CRG	LTL, CTL, FCL, ...	LTL, ProGoalML, TOSCA, ...
Verknüpfte Gesetze/Regeln	✗	✗	✗	✓
Austauschformat	XML	XML	✗	XML
Modellierungsart	Abfrage	Abfrage	Anforderungen	Anforderungen
Grafisches Modell	✓	✓	✓	✗
Grafischer Editor	✓	✓	✗	✗
Validierung				
Designphase	✓	✓	✓	✓
Auslieferung	✗	✗	✗	✓
Laufzeit	✗	✗	✓	✓

Tabelle 3.1.: Überblick über die Verfahren

In diesem Kapitel wurden vier verschiedene Verfahren zur Validierung von Geschäftsprozessen in Hinsicht auf Compliance-Anforderungen vorgestellt. Tabelle 3.1 zeigt einen Vergleich der Ansätze im Hinblick auf ihre Features.

Alle vier Verfahren eignen sich gut dafür, eigene Prozesse während der Modellierung zu überprüfen. Soll jedoch mehr als nur diese eine Phase validiert werden, so reduziert sich die Nützlichkeit von BPMN-Q und SeaFlows deutlich.

Von den zwei verbleibenden Ansätzen hat CoReL das Problem, dass noch recht wenige Teile spezifiziert sind. Es gibt weder ein Austauschformat noch einen Editor dafür.
Auch dem Compliance Descriptor fehlen noch zwei wichtige Teile: ein grafisches Modell und ein Editor. Die Entwicklung dieser beiden Punkte ist Aufgabe dieser Arbeit und diese damit nach deren Abschluss vorhanden. Der Compliance Descriptors bietet zudem die Möglichkeit,

auch die Auslieferung des Prozesses und dessen Infrastruktur zu beeinflussen. Ein Feature, dass keines der konkurrierenden Verfahren ermöglicht. Des weiteren können dort als als einziges die Anforderungen mit den Gesetzes- oder Regularientexten verbunden werden, auf denen diese basieren. Dadurch wird vor allem Mitarbeitern ohne technische Ausbildung die Arbeit mit und an den Anforderungen erleichtert, da direkt auf den Text Bezug genommen werden kann.

4. Erstellung des grafischen Modells

In diesem Kapitel wird ein grafisches Modell für den Compliance Descriptor erstellt. Dafür wird zuerst der Descriptor genauer untersucht. Dazu werden die Anforderungen an dieses Modell zusammengetragen, anschließend wird der Descriptor auf seine Bestandteile hin untersucht, um daraus im dritten Abschnitt des Kapitels das eigentliche Modell zu entwerfen. Der vierte Abschnitt dient der Überprüfung der Anforderungen in Bezug auf das entworfene Modell.

4.1. Bestandteile des Compliance Descriptors

Den einzigen Anhaltspunkt über die kompletten Bestandteile des Compliance Descriptor liefert eine XSD-Datei. XSD[40] steht für XML Schema Definition und definiert, wie eine XML-Datei auszusehen hat. Sie beschreibt das Austauschformat, das zu diesem Zeitpunkt die Hauptmethode zur Erstellung eines Compliance Descriptors darstellt. Aus diesem Schema lassen sich die folgenden Typen und Beziehungen zwischen diesen Typen extrahieren:

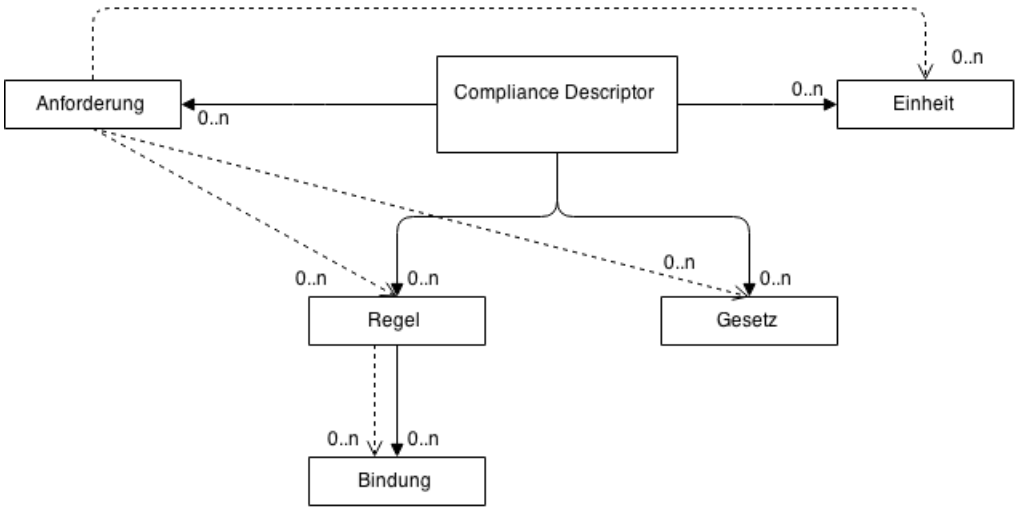

Abbildung 4.1.: Compliance Descriptor Struktur

Pfeile mit durchgezogenen Linien geben dabei die Struktur innerhalb des Schemas wieder, wohingegen die gestrichelten Pfeile die indirekten Verbindungen darstellen. Die indirekten Verbindungen werden dabei über einzigartige Identifizierer wie den Namen hergestellt. Für das grafische Modell sind gerade diese indirekten Verbindungen interessant, da sie die Zusammenhänge deutlich besser veranschaulichen als die Verbindungen innerhalb der XML-Datei.
Die fünf Elemente des Compliance Descriptors sowie die zwei verschiedenen Verbindungsarten werden in den nächsten Abschnitten näher beschrieben.

4.1.1. Anforderung

Anforderungen sind die kodifizierten Voraussetzungen, die von Gesetzen und Vorschriften gestellt und von Prozessen eingehalten werden sollen. Dazu enthalten sie neben einem Namen einen Compliance-Ausdruck, in dem auf verknüpfte Regeln und Einheiten verwiesen wird, um Aussagen über die Einhaltung der Anforderung treffen zu können. Die Regeln können darin mit Hilfe von *AND*- und *OR*-Operatoren verknüpft werden und mit Hilfe einer Klammerung strukturiert werden. Ein Beispielausdruck könnte folgendermaßen aussehen:

```
WirdNacheinanderAusgefuehrt("Aktivitaet1", "Aktivitaet2") AND
DauertNichtLaengerAls("Aktivitaet1", 15)
```

In diesem Ausdruck könnte z.B. überprüft werden, dass zwei Aktivitäten in der korrekten Reihenfolge ausgeführt werden und dass die Ausführung der ersten Aktivität nicht länger als 15 Minuten dauert.

Durch eine Verknüpfung mit einem Gesetzelement kann zusätzlich zur Formulierung der Anforderung im Ausdruck auch noch ein Verweis auf den entsprechenden Gesetzestext hinzugefügt werden. Ein XPath-Ausdruck kann verwendet werden um nicht nur auf den gesamten Text zu verweisen, sondern direkt auf den entsprechenden Paragraphen.

Anforderungen enthalten die folgenden Eigenschaften:

- **ID:** Der einzigartige Identifizierer der Anforderung, die zur Verknüpfung verwendet werden kann.
- **Beschreibung:** Eine Beschreibung der Anforderung in einem kurzen Text.
- **Compliance Ausdruck:** Eine Kombination von Regeln, die zur Validierung der Anforderung ausgewertet werden müssen.

4.1.2. Regel

Um Anforderungen kompakt zu halten, werden diese aus Regeln zusammengesetzt. Diese sind allgemein gehalten und können in mehreren Anforderungen verwendet werden.

Durch die Wahl der Sprache sowie der Phase, in der die Regel angewendet werden soll, ist es möglich verschiedene Aspekte des Prozesses zu überprüfen. Mit Hilfe von LTL lässt sich die Ausführungsreihenfolge von Aktivitäten bereits in der Designphase überprüfen, während durch Unterstützung von TOSCA die Möglichkeit besteht die Auslieferung des Prozesses zu beeinflussen. Läuft der Prozess bereits, können mit ProGoalML KPIs und die Einhaltung von Grenzwerten überwacht werden.

Regeln enthalten dazu entweder einen Regelausdruck, der in der gewählten Sprache die Regel beschreibt, oder einen Verweis auf eine Regel in einem anderen Dokument.

Zusätzlich kann eine Regel einen Variability Descriptor enthalten, der mögliche Veränderungen am Prozess beschreibt. Dazu wird das XML-Dokument des Variability Descriptors direkt in den

Compliance Descriptor eingebunden. Bei der Validierung des Compliance Descriptors werden die Bindungen aufgelöst und mit Hilfe des Variability Descriptors in die abstrakte Regel eingesetzt um eine ausführ- und validierbare Regel zu erhalten.

Regeln enthalten die folgenden Eigenschaften:

- **ID:** Der einzigartige Identifizierer der Regel, der zur Verknüpfung verwendet werden kann.
- **Beschreibung:** Eine Beschreibung der Regel in einem kurzen Text.
- **Sprache:** Die Sprache, in der die Regel geschrieben wurde, z.B. LTL.
- **Phase:** Die Phase, in der die Regel ausgeführt werden soll. Bei LTL wäre dies z.B. während der Designphase.
- **Regel:** Die eigentliche Regel; kann entweder als Ausdruck übergeben werden oder über eine URL eingebunden werden.
- **Bindungsstrategie:** Die Strategie, mit der Bindungen beim Variability Descriptor eingebunden werden.

4.1.3. Bindung

Bindungen sind Teil einer Regel und beschreiben die Variabilitätspunkte der Regel. Sie enthalten den Namen des Punktes sowie entweder einen konstanten Wert oder die Position des Parameters in der Regel. Im Regelausdruck der Anforderung werden Regeln ähnlich wie Funktionen in anderen Sprachen verwendet, und es können ihnen auch Parameter übergeben werden. Ein solcher Aufruf kann zum Beispiel so aussehen:

```
NameDerRegel("Name der Aktivitaet")
```

Dabei wäre der erste Parameter der Regel der Name der Aktivität und könnte in einer Bindung verwendet werden.

Eine Bindung enthält die folgenden Eigenschaften:

- **Variabilitätspunkt:** Der Name des Variabilitätspunktes; sollte innerhalb der Regel einzigartig sein.
- **Konstante:** Gibt den konstanten Wert der Bindung an.
- **Parameter:** Gibt die Nummer des Parameters an, der für diese Bindung verwendet werden soll. Entweder eine Konstante oder ein Parameter muss gesetzt sein.

4.1.4. Gesetz

Gesetzesobjekte sind reine Inhaltsobjekte. Sie enthalten den vollständigen Text des Gesetzes bzw. der Vorschrift als XHTML- oder XML-Dokument. Dadurch kann der Text direkt in das XML-Dokument des Compliance Descriptor eingebettet werden und Anforderungen direkt auf einzelne Paragraphen des Texte verweisen. So sind die Compliance-Anforderungen sowohl in Form eines Compliance-Ausdrucks als auch in Textform verfügbar. Dies kann sowohl bei der Erstellung des Compliance Descriptors zum Nachschlagen als auch bei der Meldung von Compliance-Verstößen hilfreich sein und dem Nutzer bei der Verwendung des Compliance Descriptors helfen.

Gesetze enthalten die folgenden beiden Eigenschaften:

- **Titel:** Der Titel des Gesetzes; wird zum Verknüpfen innerhalb der XML-Datei verwendet und sollte deshalb einzigartig sein.
- **Inhalt:** Der Gesetzestext als XHTML- oder XML-Dokument.

4.1.5. Einheit

Einheiten sind indirekte Verweise auf Teile des Geschäftsprozesses. Dies kann z.B. eine Aktivität des Prozesses oder ein Teil der Infrastruktur sein. Über den Namen der Einheit und ihres Typs wird sie mit der entsprechenden Einheit implizit verknüpft.

Eine Einheit enthält die folgenden Eigenschaften:

- **Name:** Der Name der Einheit
- **Typ:** Der damit verbundene Typ von Objekt.
- **Beschreibung:** Eine Beschreibung der Einheit. Kann beispielsweise bei der Erstellung eines Prozesses dazu verwendet werden, um genau zu beschreiben, welche Einheit vom Compliance Descriptor erwartet wird.

4.1.6. Verbindung

Die Verbindung stellt eine einfache Verknüpfung zweier Elemente dar. Sie besteht zwischen einer Anforderung und einer Regel oder einer Einheit. Eine Verbindung enthält keine Eigenschaften.

4.1.7. Gesetzesverbindung

Ein Spezialfall bei den Verbindungen zwischen Elementen ist die Verbindung zwischen einer Anforderung und einem Gesetz. Hierbei muss noch die Möglichkeit bestehen, einen XPath-Ausdruck unterzubringen um auf einen bestimmten Paragraphen des Gesetzes verweisen zu können. XPath-Ausdrücke sind Anfragen an das XML-Dokument, um Teile von diesem auf Grund von Tags und Attributen zu finden und zurückzuliefern.

Deshalb enthält die Gesetzesverbindung eine Eigenschaft:

- **XPath:** Der XPath-Ausdruck, der auf den betroffenen Abschnitt im Text des Gesetzes verweist.

4.2. Anforderungen

Im dritten Kapitel wurden drei Ansätze mit einem existierenden grafischen Modell vorgestellt und analysiert. Dabei wurden auch einige Nachteile entdeckt, die nun im Modell des Compliance Descriptors vermieden werden sollen.
Zu technisch komplexe Darstellungen oder Regeldefinitionen durch das Modell schrecken unerfahrene Anwender durch die hohe Einarbeitungszeit ab und verhindern es, einen schnellen Überblick über das Diagramm zu erhalten. Trotzdem soll möglichst wenig von der Funktionalität eingebüßt werden, um alle Möglichkeiten des Compliance Descriptors auszunutzen. Von BPMN-Q wurden pro Anforderung oder Regel eine eigene Abfrage in einem eigenen Dokument formuliert. Da dabei für den Benutzer die Zusammenhänge schlecht nachvollziehbar sind und die gesamten Anforderungen an den Prozess schlecht sichtbar sind, sollte dies vermieden werden.
Der Compliance Descriptor richtet sich vor allem an Anwender von BPMN-Geschäftsprozessen. Deshalb wäre es von Vorteil die Darstellung möglichst nah am BPMN-Modell zu halten.

Aus diesen Einsichten werden die vier folgenden Anforderungen formuliert:

1. **Simple Darstellung**
   Eine simple Darstellung soll den Inhalt des Diagramms schnell erfassbar und auch für weniger technikaffine Nutzer lesbar machen.

2. **Komplexität ermöglichen**
   Für erfahrene Anwender sollen möglichst alle Funktionen des Compliance Descriptors zugänglich sein.

3. **Darstellung aller Elemente in einem Diagramm**
   Das Diagramm soll nicht in unterschiedliche Abschnitte oder einzelne Anforderungen aufgeteilt werden. Dadurch lassen sich Elemente mehrmals verwenden und können als ein Gesamtkonzept verstanden werden.

4. **BPMN-inspirierte Formen**
   Durch die Orientierung an BPMN-Elementen sollen die Formen des Modells für den Nutzer bekannt und damit leicht verständlich sein.

4.3. Das grafische Modell

Nachdem in den vorhergehenden Abschnitten bereits der Compliance Descriptor beschrieben wurde und die Anforderungen definiert wurden, soll nun das daraus erstellte grafische Modell vorgestellt werden. Dabei wird auf die einzelnen Elemente des Descriptors eingegangen und dann ein minimaler Beispielgraph gezeigt.

4.3.1. Regel

Abbildung 4.2.: Regel inklusive Bindung

Eine Regel des Compliance Descriptors wird ähnlich wie eine Aktivität im BPMN-Diagramm dargestellt. Beide haben die Gemeinsamkeit, dass sie die Hauptelemente des jeweiligen Diagrammtyps sind. Aus diesem Grund wurde diese Darstellungsweise gewählt. Die ID der Regel wird im oberen Bereich des Regelelements dargestellt. Weitere Eigenschaften der Regel lassen sich leider nicht direkt im Element anzeigen, ohne die Komplexität der Ansicht deutlich zu erhöhen. Diese müssen über andere Möglichkeiten bei der Implementierung zugänglich gemacht werden.

4.3.2. Bindung

Da Bindungen ein Teil einer Regel sind, werden sie innerhalb der Regel dargestellt. Grafisch sind sie deutlich schlanker, haben aber eine generell ähnlichen Form wie die Regel. Der Name des Variabilitätspunktes wird innerhalb des Elementes zur Kennzeichnung verwendet. Da im Variability Descriptor die Namen der einzelnen Punkte mit einem Dollarzeichen beginnen, um sie klar als Variablen zu kennzeichnen, beginnt auch der Name des Punktes in der Bindung mit einem Dollarzeichen.

4.3.3. Anforderung

Anforderungen vereinen mehrere Regeln in sich, um daraus einen Compliance-Ausdruck zu bauen. Deshalb werden sie ähnlich einer Regel dargestellt, ihr Rand ist jedoch doppelt ausgeführt um damit die Kombination mehrerer Regeln auszudrücken. Die ID der Anforderung wird innerhalb des Elements dargestellt. Der Compliance-Ausdruck wird in der grafischen Darstellung zwar nicht angezeigt, jedoch sind die Bestandteile des Ausdrucks durch die Verbindungen von der Anforderung zu den anderen Elementen sichtbar. Die genaue Komposition muss dann durch die Implementierung des Editors zugänglich gemacht werden.

Abbildung 4.3.: Anforderung

4.3.4. Gesetz

Abbildung 4.4.: Gesetz

Ein Gesetz wird ähnlich einer Regel dargestellt, wird jedoch um das Paragraphzeichen ergänzt um zu verdeutlichen, dass es sich dabei um ein Gesetz handelt. Der Titel des Gesetzes oder der Richtlinie wird innerhalb des Elements dargestellt, während der Inhalt des Elements nicht angezeigt wird. Die Anzeige oder Bearbeitung eines Dokuments mit vielen Seiten innerhalb des Diagramms ist jedoch nicht praktikabel möglich.

4.3.5. Einheit

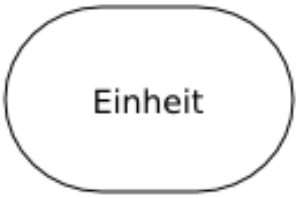

Abbildung 4.5.: Einheit

Die Einheit wird pillenförmig dargestellt. Als einzige Form findet man sie nicht in einem BPMN-Diagramm. Der Name der Einheit wird mittig in der Form angezeigt. Der Typ und die Beschreibung sind nicht enthalten ist.

4.3.6. Verbindung

Eine Verbindung wird als schwarze Linie mit gefüllter Pfeilspitze dargestellt.

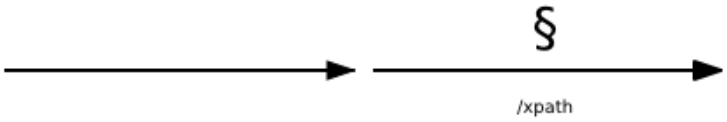

Abbildung 4.6.: Verbindung und Gesetzesverbindung

4.3.7. Gesetzesverbindung

Eine Gesetzesverbindung wird ähnlich der normalen Verbindung dargestellt, es wird jedoch in der Mitte der Linie ein zusätzliches Paragraphzeichen angezeigt um den Unterschied zwischen den beiden Verbindungstypen zu kennzeichnen. Zudem wird der XPath-Ausdruck unterhalb der Linie angezeigt, um direkt Informationen über den gewählten Paragraphen zu erhalten.

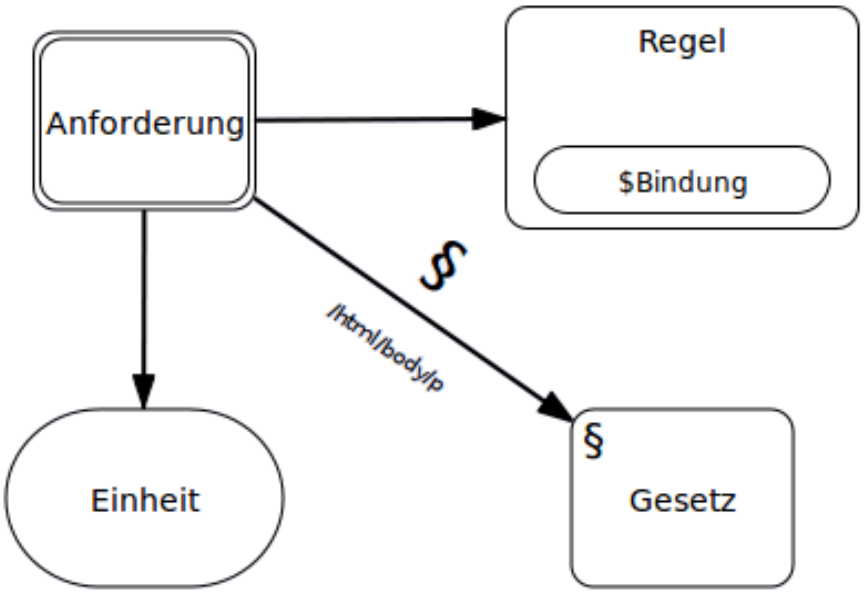

Abbildung 4.7.: Minimaler Beispielgraph

4.4. Überprüfung der Anforderungen

Bisher wurde der Compliance Descriptor genauer untersucht, vier Anforderungen aus der State of the Art Analyse extrahiert und das grafische Modell vorgestellt. Im letzten Schritt sollen jetzt die gestellten Anforderungen aus Abschnitt 4.2 überprüft werden:

1. **Simple Darstellung ✓**
   Durch reduzierte Informationen und klare Linien sollte es gelungen sein, eine simple Darstellung anzubieten.

2. **Komplexität ermöglichen ✗**
   Die ganze Funktionalität des Compliance Descriptors kann mit diesem grafischen Modell nicht dargestellt werden. Jedoch besteht immer noch die Möglichkeit, dies durch den Editor auszugleichen und dort die restlichen Funktionen abzubilden.

3. **Darstellung aller Elemente in einem Diagramm ✓**
   Mehrere Anforderungen können in einem Diagramm modelliert werden und Regeln können in verschiedenen Anforderungen eingesetzt werden.

4. **BPMN-inspirierte Formen ✓**
   Bis auf ein Element orientieren sich alle der gewählten Formen an BPMN, und der einzige Ausreißer ist nicht sehr weit entfernt.

Drei der vier Anforderungen konnten bereits vom grafischen Modell erfüllt werden. Die verbleibende Anforderung kann jedoch während der Implementierung des Editors noch erfüllt werden. Dies wird im folgenden Kapitel verdeutlicht.

5. Erstellung des grafischen Editors

Die grafische Darstellung, die im vorherigen Kapitel erstellt wurde, soll nun umgesetzt werden. Dafür wird der Oryx-Editor verwendet und erweitert. Dieser wird im ersten Abschnitt dieses Kapitels vorgestellt. Im zweiten Abschnitt werden die Anforderungen an die Umsetzung des grafischen Editors zusammengestellt. Der dritte Abschnitt erklärt die Implementierung der einzelnen Teile, die für die volle Funktionalität des Editors benötigt werden, und der letzte Abschnitt überprüft die Umsetzung der Anforderungen durch die Implementierung.

5.1. Der Oryx-Editor

Der Oryx-Editor ist ein grafischer Webeditor, der am Hasso-Plattner-Institut (HPI) in Potsdam entwickelt wurde, um BPMN-Diagramme zu erstellen. Die Arbeit daran begann 2006 mit drei zusammenhängenden Bachelorarbeiten[8][35][32], in deren Rahmen eine erste Version des Editors entwickelt wurde. Durch die Veröffentlichung unter einer Open Source-Lizenz und einem regen Entwicklungsfortschritt wurde der Oryx-Editor von vielen Universitäten und Unternehmen gut angenommen. Als 2009 die ursprünglichen Entwickler am HPI daraus ein kommerzielles Produkt unter dem Namen SSignavio"[1] entwickelten wurde die Weiterentwicklung von Oryx allerdings eingestellt.

Oryx gliedert sich in zwei verschiedene Teile, das Backend, das auf dem Server läuft und sich vor allem um die Datenhaltung kümmert, und das Frontend, d.h. die Webapplikation mit der Diagramme modelliert werden können.

5.1.1. Backend

Als Backend setzt Oryx auf einen Java-Server, der mit Hilfe eines Applikationsserver ausgeführt wird. Um Daten und erstellte Diagramme zu speichern wird PostgreSQL[2] als Datenbank eingesetzt. Durch die Verwendung von sogenannten Servlets[30], kleinen Programmen die über eine URL aufgerufen werden können, ist das Backend sehr modular aufgebaut und kann leicht um weitere Endpunkte erweitert werden.

Neben der Datenhaltung kümmert sich das Backend auch um die Authentifizierung der Benutzer. Sie findet mit Hilfe von OpenID[29], einem Standard für dezentrale Authentifizierung, statt. Dabei kann sich der Benutzer mit einer URL anmelden und wird dann auf die Seite seines OpenID-Anbieters weitergeleitet. Nach erfolgreicher Anmeldung wird er zurückgeleitet und ist dann authentifiziert. Durch Verwendung von OpenID erspart man sich so, Daten der Benutzer

[1] http://www.signavio.com
[2] http://www.postgresql.org/

wie Passwörter speichern zu müssen. Für den Benutzer besteht zudem der Vorteil, dass er sich nicht für jeden Account bei einem Webdienst einen neuen Benutzername und ein neues Passwort merken muss.

5.1.2. Frontend

Das Frontend von Oryx besteht aus zwei verschiedenen Teilen: Das sogenannte Process Repository dient als Einstieg in die Anwendung. Dort können neue Diagramme erstellt und bereits vorhandene durchsucht und geöffnet werden. Durch das Öffnen eines Diagramms wird der zweite Teil des Frontends, der Editor, in einem neuen Browsertab geöffnet.

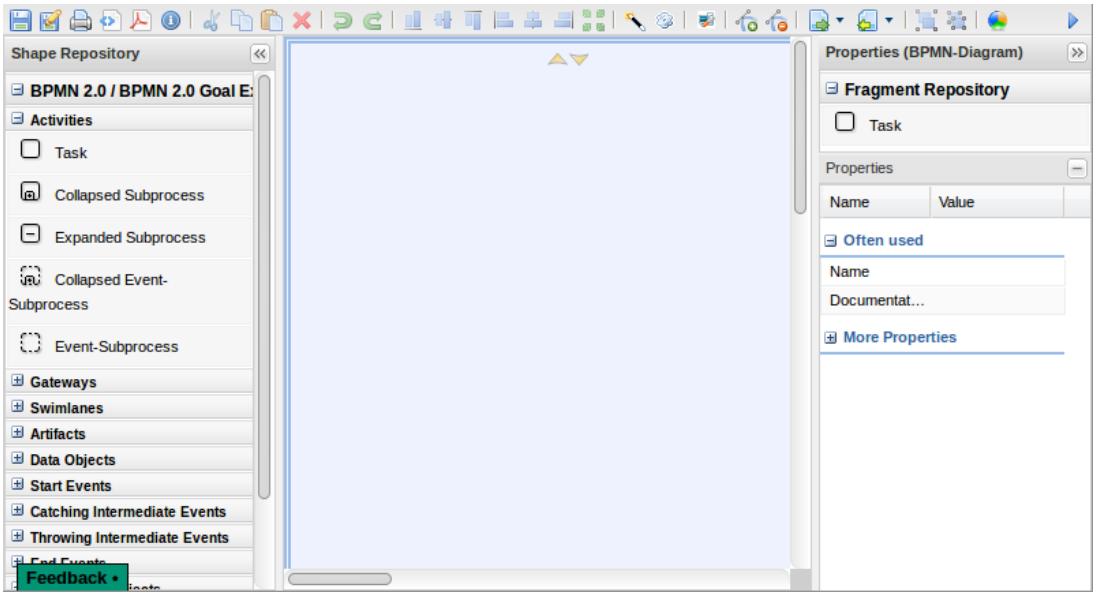

Abbildung 5.1.: Oryx-Editor mit BPMN-Stencilset

Die Benutzeroberfläche des Editors ist in vier Abschnitte gegliedert: Am oberen Rand des Editors befindet sich die Werkzeugleiste, die eine Reihe von Knöpfen mit Symbolen enthält. Durch das Klicken auf einen Knopf kann die entsprechende Funktionalität aktiviert werden. Beispiele für Knöpfe sind z.B. der Speichern-Knopf, der Exportieren-Knopf oder der Validierungs-Knopf. Der rechte Rand enthält das Shape Repository, welches die vom momentanen Diagrammtyp unterstützten Formen enthält. Diese können durch Drag and Drop, also das Ziehen mit dem Mauszeiger, auf die gewünschte Stelle im eigentlichen Editorbereich gezogen werden.
Dieser befindet sich in der Mitte des Fensters und nimmt den Großteil des verfügbaren Platzes ein. Dort lassen sich die Elemente ebenfalls per Drag and Drop bewegen. Zusätzlich gibt es noch die Möglichkeit eine Abkürzungsfunktion zu verwenden. Nach dem Klicken auf ein Element werden rechts davon leicht durchsichtig andere Elemente und Kanten angezeigt mit denen dieses Element verbunden sein kann. Wird eines davon per Klick verschoben, so wird automatisch das entsprechende Element inklusive einer Verbindung zwischen den Beiden erstellt.
Am rechten Rand des Editors befinden sich die Eigenschaften. Klickt man auf eines der Elemente, werden dort die Eigenschaften des gewählten Elements angezeigt und können bearbeitet werden.

Der Webeditor wurde mit den Standard-Webtechnologien HTML, CSS und Javascript geschrieben. Um nicht alle Funktionen selbst schreiben zu müssen, baut Oryx auf einigen verbreiteten Bibliotheken auf: Ext.js wird verwendet, um Funktionalität und Aussehen einer normalen Desktopanwendung nachzubilden. Dafür bietet Ext.js[3] viele Elemente, die dynamisch initialisiert und angezeigt werden können. Zudem wird Prototype.js[4] verwendet, eine Bibliothek die Javascript um eine Art Klasse und Klassenvererbung erweitert, sowie Funktionen für Anfragen an den Server und das Auswählen von Elementen des Editors erlaubt.

5.1.3. Erweiterbarkeit

Da in dieser Arbeit der Oryx-Editor um ein neues grafisches Modell und damit verbundene Plugins erweitert werden soll, ist die Erweiterbarkeit von Oryx entscheidend.
Der Server lässt sich einfach durch das Hinzufügen neuer Servlets erweitern. Dadurch dass diese kleine eigenständige Programme sind müssen sie nur dem Applikationsserver bekannt gemacht werden um zu funktionieren.
Für den Editor gibt es verschiedene Möglichkeiten zur Erweiterung. Sogenannte Stencilsets[31] werden verwendet um Diagrammtypen zu definieren. Sie bestehen aus einer JavaScript Object Notation (JSON)[12]-Datei, einem Dateiformat, das den Objekttypen aus Javascript verwendet um Daten zu kodieren, sowie Scalable Vector Graphic (SVG)[38]-Dateien, einem auf XML aufbauenden Format für skalierbare Vektorgrafiken. SVG wird von Oryx dazu verwendet um die Graphen darzustellen. Da es auf XML aufbaut, kann es direkt in die (X)HTML-Seite integriert werden und biete viele Funktionen zum Zeichnen. Die einzelnen SVG-Dateien beschreiben das Aussehen der einzelnen Elemente des Diagramms, während die JSON-Datei diese verwendet und um semantische Informationen erweitert. Darunter fallen unter anderem die Eigenschaften des Elements, die anzuzeigenden Namen und mögliche Verknüpfungen zwischen den einzelnen Elementen. Der Oryx-Editor ist modular aufgebaut. Die meiste Funktionalität wird von Plugins implementiert. Durch die Entwicklung eigener Plugins kann also der Funktionsumfang von Oryx erweitert werden.

5.1.4. Verwendete Version

Da die Weiterentwicklung von Oryx eingestellt wurde und es auch aus den Reihen der Nutzer nicht mehr zur (öffentlichen) Weiterentwicklung kommt, werden einige Versionen von unterschiedlichsten Institutionen nicht-öffentlich weiterentwickelt. Eine solche Version wird auch vom Institut für Architektur von Anwendungssystemen (IAAS) und vom Fraunhofer Institut für Arbeitswissenschaft und Organisation (IAO) unterhalten und weiterentwickelt. Sie wurde dabei in aPro integriert und um Erweiterungen zur Arbeit mit aPro ergänzt.

aPro ist eine Monitoringlösung, die Oryx zur Modellierung von Geschäftsprozessen verwendet und die entstandenen Prozesse als eigenständige Webanwendungen starten und überwachen kann. Oryx stellt dabei das Hauptinterface dar und wurde dazu durch zahlreiche Plugins erweitert. Aufbauend auf der aktuellsten Version von aPro und dem darin enthaltenen Oryx-Editor wird in diesem Kapitel die Implementierung des grafischen Webeditors aufgezeigt.

[3] http://www.sencha.com/products/extjs/
[4] http://prototypejs.org/

5.2. Anforderungen

Auch in diesem Kapitel sollen wieder Anforderungen formuliert werden, die durch die Implementierung des grafischen Editors erfüllt werden sollen. Eine der Anforderungen wird dabei aus dem vorherigen Kapitel übernommen, da sie dort ja nicht voll erfüllt werden konnte. Die meisten Anforderungen ergeben sich allerdings aus der Aufgabenstellung dieser Arbeit. Daraus ergeben sich die fünf folgenden Anforderungen:

1. **Komplexität ermöglichen**
   Diese Anforderung konnte im vorherigen Kapitel nicht voll erfüllt werden. Deshalb sollte sie durch den Editor nun erfüllt werden.

2. **Darstellung und Editierbarkeit des grafischen Modells**
   Das grafische Modell soll voll dargestellt werden und im Editor bearbeitbar sein.

3. **Nutzung der Abkürzungsfunktionalität**
   Die Abkürzungsfunktionalität von Oryx soll voll ausgenutzt werden. Ausgehend von einer Anforderung sollte es möglich sein, darüber alle verknüpften Elemente zu erstellen.

4. **Editor für Compliance-Ausdrücke**
   Ein Ausdruckeditor soll die Erstellung von Compliance-Ausdrücken vereinfachen. Dazu sollen zahlreiche Hilfestellungen geliefert werden, wie das schnelle Hinzufügen relevanter Operatoren, Regeln oder Einheiten.

5. **Validierung des Modells**
   Mit einer Validierungsfunktion soll überprüft werden können, ob das Modell auch wirklich ein korrekter Compliance Descriptor ist. Sollte dies nicht der Fall sein, so sollten Fehlermeldungen an den entsprechenden Elementen angezeigt werden.

6. **Export der XML-Datei**
   Ist das grafische Modell valide, so soll es möglich sein eine XML-Datei mit dem Compliance Descriptor herunterzuladen und diese in weiteren Anwendungen zu nutzen oder selber noch bearbeiten zu können.

5.3. Implementierung des grafischen Editors

Die Implementierung des grafischen Editors findet in zwei Abschnitten statt. Im ersten Schritt wird das sogenannte Stencilset erstellt, das die Darstellung des Modells im Editor regelt. Danach wird in einem zweiten Schritt eine Reihe von Plugins entwickelt, die den Editor um nützliche Funktionen bei der Erstellung eines Compliance Descriptors ergänzen.

5.3.1. Stencilset

Stencilsets beschreiben die Struktur und das Aussehen eines grafischen Modells im Oryx-Editor. Dazu bestehen sie aus einer JSON-Datei, die die Struktur enthält, und einigen SVG-Dateien, die das Aussehen der einzelnen Elementen beschreiben. Die JSON-Datei wird dabei als erstes erstellt, da sie einen größeren Einfluss auf die Funktionalität des Editors hat und das Aussehen jederzeit noch geändert werden kann. Strukturell besteht die Datei aus drei verschiedenen Bereichen:

- **Der Kopfbereich:**
  Hier werden ein paar grundlegende Eigenschaften des Diagramms gesetzt, wie der Name des Typs.

- **Der Stencil-Bereich:**
  Im Stencil-Bereich werden die einzelnen Elemente, von Oryx Stencil genannt, definiert. Dazu wird als Erstes ein Hauptelement definiert, dass den kompletten Inhalt des Diagramms enthält. Es wird jedoch nicht als Element im Editor angezeigt. Trotzdem hat es eine einzige Eigenschaft, den Namen des gesamten Diagramms, der vom Benutzer gesetzt werden kann. Als Nächstes werden die restlichen Elemente mit den bereits bekannten Eigenschaften definiert. Bei der Definition der Eigenschaften kann neben dem Namen u.a. auch ein Typ angegeben werden. Dadurch lässt sich der Werte z.B. auf wenige bekannte und vorgegebene Werte einschränken. Dies wird beispielsweise bei der Sprache und Phase einer Regel genutzt.

- **Der Regel-Bereich:**
  In diesem Bereich können Regeln, die die Bearbeitung des Diagramms beeinflussen, definiert werden. Dazu ist dieser Bereich noch einmal in mehrere Unterbereich unterteilt. Mit Verbindungsregeln kann definiert werden, welche Elemente durch welche Art von Verbindung miteinander verbunden werden kann. Im Falle des Compliance Descriptor gehen normale Verbindung von einer Anforderung zu einer Regel oder Einheit, während Gesetzesverbindungen von einer Anforderung zu einem Gesetz gehen. Durch Containmentregeln kann bestimmt werden, welche Elemente andere Elemente aufnehmen können. Da das Diagramm an sich selbst ein Element ist, muss hier definiert werden, dass alle Elemente bis auf die Bindung enthalten sein können. Bindungen dürfen dafür Bestandteile von Regeln sein. Kardinaliätsregeln erlauben es die Anzahl an aus- oder eingehenden Verbindungen zu beschränken. Mit Layoutregeln kann die Positionierung von ein- und ausgehenden Verbindungen verändert werden. Und als Letztes können mit Morphingregeln mögliche Veränderungen definiert werden. Ein Beispiel dafür sind Ereignisse bei BPMN, deren exakter Typ dadurch noch verändert werden kann. Leere Morphingregeln für Elemente werden zudem benötigt, damit Abkürzungen zu Elementen hinzugefügt werden, die mit diesen verbunden werden können.

Nach Fertigstellung des Stencilsets, kann mit den SVG-Grafiken begonnen werden. Dafür wird für jedes Element eine eigene Grafik angelegt und mit einem Vektorgrafikprogramm erstellt. Anschließend sollte die SVG-Datei noch um einige von Oryx vorgegebene Tags und Attribute erweitert werden. Dadurch kann unter anderem angegeben werden, an welchen Stellen sich Verbindungen an das Element andocken lassen oder wie sich das Element bei einer Veränderung der Größe verändert und in welchen Bereichen dies überhaupt stattfinden darf.

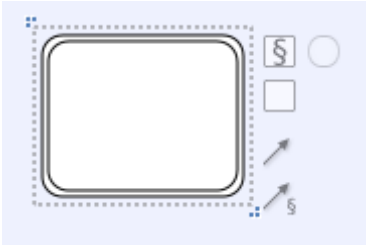

Abbildung 5.2.: Abkürzungsfunktionalität an einem Anforderungselement

Um das Ganze noch ein wenig aufzuhübschen können zudem noch Icons hinzugefügt werden, die im Shape Repository sowie im Process Repository angezeigt werden. Diese sollen auf kleinem Raum möglichst deutlich den Typ des Elements darstellen.

Nach Fertigstellung des Stencilsets muss dieses noch bei Oryx registriert werden, wozu ein einfacher Eintrag in einer XML-Konfigurationsdatei ausreicht.

5.3.2. Plugins

Abbildung 5.3.: Die Plugins in der Werkzeugleiste

Das grafische Modell ist mit dem Stencilset bereits benutzbar, doch erst durch die mit Plugins hinzugefügte Funktionalität wird es auch für die spätere Anwendung bei der Validierung von Geschäftsprozessen interessant. Davor ist es noch zu einfach möglich einen inkorrekten Compliance Descriptor zu erstellen und dies erst spät im Entwicklungsprozess festzustellen.
Dafür werden insgesamt vier verschiedene Plugins integriert, die fünf Funktionen und damit Knöpfe in der Werkzeugleiste zur Verfügung stellen.
Von links nach rechts sind dies der Ausdruckeditor, die Validierung, der JSON-Import, der JSON-Export und der XML-Export. Der JSON-Im-/Export ist zwar nicht durch eine Anforderung gefordert, wird aber für die Implementierung des mobilen Prototypen in Kapitel 7 benötigt. Da dieser nicht direkt auf die Daten in Oryx zugreifen kann, müssen sie auf diesem Wege zum mobilen Editor und von ihm zurück kopiert werden können.

Plugins werden mit Hilfe von Javascript entwickelt und als jeweils einzelne Klassen erstellt. Um sie Oryx bekannt zu machen müssen sie in einer XML-Datei registriert werden und in ein Profil eingebunden werden. Profile beschreiben die verwendeten Plugins eines Stencilsets. Da das Standardprofil bereits sehr voll ist und zudem einige Funktionalitäten enthält die für den Compliance Descriptor nicht benötigt werden, wird deshalb ein eigenes Profil erstellt, das nur die nötigsten Plugins enthält.

5.3.2.1. Ausdruckeditor

Abbildung 5.4.: Ausdruckeditor

Der Ausdruckeditor kann durch das Auswählen einer Anforderung und dem anschließend Drücken des Formeleditor-Knopfes in der Werkzeugleiste geöffnet werden. In ihm kann der Compliance-Ausdruck deutlich effizienter bearbeitet werden als im normalerweise zur Verfügung gestellten

Textfeld.
Das oberste Element des Editors ist ebenfalls ein normales Texteingabefeld, in dem der Ausdruck wie üblich editiert werden kann. Darunter befinden sich fünf verschiedene Werkzeugleisten, die jeweils verschiedene Funktionen zur Verfügung stellen: Mit Hilfe der ersten Werkzeugleiste lassen sich die Operatoren *AND*, *OR* sowie öffnende und schließende Klammern an die momentan ausgewählte Stelle platzieren. Die zweite Leiste enthält nur eine Abkürzung für einen konstanten Text. Dabei wird jedoch zusätzlich noch ein Fragezeichen zwischen die beiden Anführungszeichen gesetzt, das zugleich selektiert wird. Dadurch werden Stellen markiert an denen noch Eintragungen durchzuführen sind. Für die dritte Reihe werden alle mit der Anforderung verbundenen Regeln gesammelt und angezeigt. Wird eine solche Regel eingefügt, werden eventuell vorhandene Parameter ebenfalls mit einem Fragezeichen markiert eingesetzt. Verbundene Einheiten werden mit den Knöpfen in der vierten Leiste hinzugefügt. Die letzte Leiste fügt nichts hinzu sondern bietet einige erweiterte Funktionalitäten. Mit dem ersten Knopf kann die gesamte Formel gelöscht werden, der Zweite selektiert automatisch das nächste Fragezeichen um es zu ersetzen und der Dritte bewegt die Selektierung im Textfeld zum Ende des momentanen Ausdrucks.

Schlussendlich kann mit den zwei Knöpfen am unteren Rand des Editors dieser wieder geschlossen werden. Beim Ersten werden dabei alle Änderungen verworfen und nur durch Drücken des zweiten Knopfes gespeichert.

5.3.2.2. Validierung

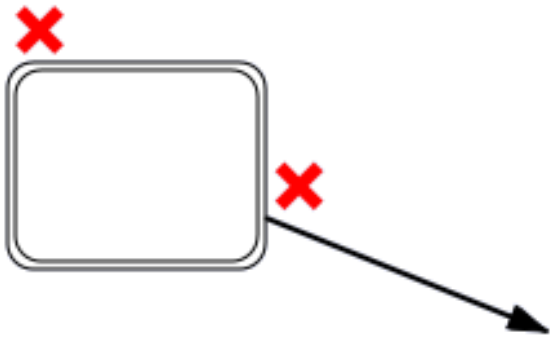

Abbildung 5.5.: Fehlgeschlagene Validierung

Für die Validierung ist die wichtigste Komponente ein Servlet, das die Backendkomponente von Oryx erweitert. An dieses Servlet wird die JSON-Repräsentation des momentanen Diagramms geschickt. Auf Serverseite wird dieses auf Fehler überprüft, u.a. auf leere Pflichtfelder, ins Leere führende Verbindungen oder inkorrekte XPath-Ausdrücke oder URLs. Wurden Fehler gefunden, so werden diese zusammen mit den einzigartigen Identifizierern der inkorrekten Elemente zurück an den Browser gesendet. Diese werden dort mit einem roten Kreuz und rotem Text (falls vorhanden) markiert. Bleibt man mit dem Mauszeiger kurz über diesem Kreuz wird zudem die Fehlermeldung angezeigt. Dadurch wird diese nur wenn nötig angezeigt und nimmt dem eigentlichen Diagramm keinen Platz weg. Wird ein zweites Mal auf den Validierungsknopf gedrückt, so verschwinden die Markierungen wieder.

5.3.2.3. Import

Der JSON-Import ist das schlankeste Plugin. Es enthält nur einen Knopf in der Werkzeugleiste und die dahinter liegende Funktion zeigt auf Klick ein großes Textfeld an, in das der JSON-Code hineinkopiert werden kann. Mit Drücken des Bestätigen-Knopfes wird anschließend das aktuelle Diagramm geleert und durch das neu importierte Diagramm ersetzt.

5.3.2.4. Export

Das Export-Plugin fügt der Werkzeugleiste zwei verschiedene Knöpfe hinzu. Der JSON-Export zeigt dabei ein simples Textfeld an, aus dem der JSON-Code des momentanen Diagramms kopiert werden kann. Für den XML-Export ist etwas mehr Aufwand nötig. Dem Server wird dazu ein neues Servlet hinzugefügt, dass das aktuelle Diagramm als JSON erhält, dieses in die Compliance Descriptor Klassen aus der Bibliothek umwandelt und anschließend als XML exportiert. Davor wird jedoch noch ein kurzer Validierungslauf durchgeführt, um zu verhindern das inkorrekte Daten an das Export-Servlet geschickt werden. Die entstandene Datei wird anschließend an den Benutzer zurückgeliefert, der sie herunterladen kann.

5.4. Überprüfung der Anforderungen

Wie bereits im letzten Kapitel werden die Anforderungen nach Ende der Umsetzung überprüft. Die fünf Anforderungen waren:

1. **Komplexität ermöglichen ✓**
   Diese Anforderung konnte bei der letzten Überprüfung nicht erfüllt werden. Nach Abschluss des Editors kann auch sie nun abgehakt werden. Alle Eigenschaften von Elementen des Descriptors können über die Eigenschaftenleiste des Editors editiert werden und damit auch alle Möglichkeiten voll ausgeschöpft werden.

2. **Darstellung und Editierbarkeit des grafischen Modells ✓**
   Das grafische Modell wurde vollständig im Editor implementiert.

3. **Nutzung der Abkürzungsfunktionalität ✓**
   Durch Verwendung der Morphingregeln kann auch die Abkürzungsfunktionalität genutzt werden. Dadurch lassen sich neue Anforderungen schnell zusammenklicken.

4. **Editor für Compliance-Ausdrücke ✓**
   Ein Ausdruckeditor wurde als Plugin hinzugefügt und vereinfacht die Arbeit mit Compliance-Ausdrücken.

5. **Validierung des Modells ✓**
   Eine Validierungsfunktion wurde zum Oryx-Server hinzugefügt und diese markiert inkorrekte Elemente im Browser.

6. **Export der XML-Datei ✓**
   XML-Dateien können mit einem Knopfdruck exportiert werden.

Diesmal konnten alle fünf Anforderungen erfüllt werden. Zudem konnten noch zwei weitere Funktionen implementiert werden, die vom mobilen Prototypen benötigt werden. Die Verwendung des Editors wird im folgenden Kapitel erläutert.

6. Beispiel aus der Versicherungsbranche

In diesem Kapitel soll an Hand eines beispielhaften Geschäftsprozesses aus der Versicherungsbranche und einigen damit verbundenen Compliance-Anforderungen die Modellierung eines Compliance Descriptors gezeigt werden. Dafür wird das grafische Modell und der Editor verwendet, die in den vorherigen beiden Kapiteln beschrieben wurden.
Im ersten Abschnitt wird dazu der Geschäftsprozess beschrieben. Der zweite Abschnitt zeigt die damit verbundenen Compliance-Anforderungen und im dritten Abschnitt wird die Modellierung des Compliance Descriptors im Editor gezeigt.

6.1. Der Prozess

Einer der wichtigsten Geschäftsprozesse in der Versicherungsbranche stellt das Verwalten und Bearbeiten von Versicherungsansprüchen dar. Der im Folgenden verwendete, aus vier Schritten bestehende Prozess ist eine stark vereinfachte Version davon. Er besteht aus den folgenden vier Schritten:

1. **Eingang des Anspruchs:**
   Ein Versicherungsanspruch geht per Post, Fax, etc. bei der Versicherung ein, wird in ein strukturiertes Datenformat konvertiert und in der Kundendatenbank unter dem jeweiligen Kunden gespeichert.

2. **Bearbeitung des Anspruchs:**
   Der Anspruch wird automatisch von einem Programm oder einem Mitarbeiter auf Glaubhaftigkeit und Plausibilität untersucht.

3. **Entscheidung des Anspruchs:**
   Der Anspruch wird auf Grund des Untersuchungsergebnisses akzeptiert oder zurückgewiesen.

4. **Beantwortung des Anspruchs:**
   Eine Nachricht mit dem Ergebnis der Entscheidung über den Anspruch wird an den Kunden versendet.

Die folgende Grafik zeigt einen entsprechenden BPMN-Graphen, der diesen Geschäftsprozess abbildet.

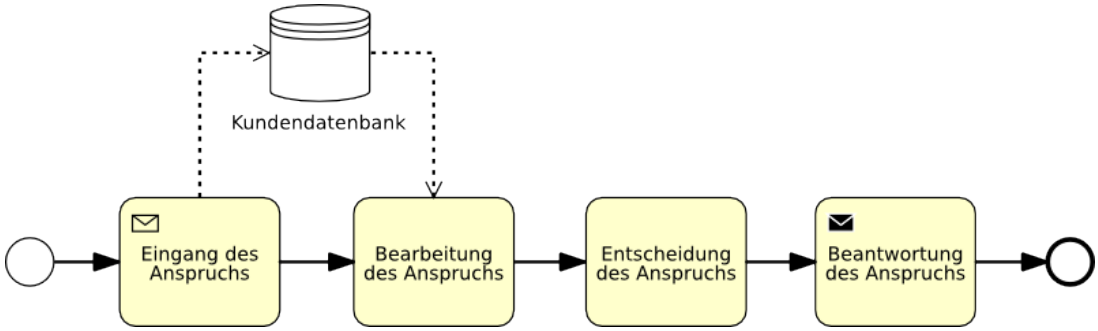

Abbildung 6.1.: BPMN-Graph des Geschäftsprozesses

6.2. Compliance-Anforderungen

Reguliert werden die Schritte des Prozesses durch den Code of Conduct des GDV (Gesamtverband der Deutschen Versicherungswirtschaft e.V.), der das Verhalten von Versicherungen im Umgang mit ihren Versicherten reguliert, und das Bundesdatenschutzgesetz, das das Speichern, die Ansicht und das Weitergeben von Kundendaten reguliert.

Aus diesen lassen sich die folgenden Anforderungen extrahieren:

- Der GDV Code of Conduct [9] sieht vor, dass der Anspruchinhaber frühzeitig informiert wird, ob seine Daten für Marketingzwecke verwendet werden. Dies kann der Fall sein sobald der Anspruch in der Datenbank eingepflegt wurde. Da diese Benachrichtigung nicht extra versendet werden soll, wird sie mit dem Ergebnis des Anspruchs versendet. Dadurch ergibt sich eine Anforderung an die Gesamtlänge des Prozesses. Dieser sollte innerhalb von 14 Tagen beendet und die Benachrichtigung sowie das Ergebnis versendet sein.

- Das Bundesdatenschutzgesetz [7] reguliert unter anderem die Speicherung der Daten und gibt vor, dass die personenbezogenen Daten Deutschland nicht verlassen dürfen. Deshalb gibt es die Anforderung, dass sich die Kundendatenbank innerhalb von Deutschland befinden muss und die Daten auch beim Transport nicht das Land verlassen.

6.3. Implementierung mit Hilfe des Editors

Die beiden Anforderungen aus dem letzten Abschnitt können nur mit Hilfe des Editors nachgebildet werden. Dazu werden zuerst die beiden Anforderungen und die jeweils relevanten Gesetze erstellt. Dafür kann die Abkürzungsfunktion des Editors verwendet werden. Beide Gesetze sind bereits in HTML-Form erhältlich und können in das entsprechende Textfeld kopiert werden. Über XPath-Ausdrücke auf der Gesetzesverbindung wird auf die jeweils relevanten Paragraphen verwiesen.

Im nächsten Schritt werden die relevanten Komponenten des BPMN-Geschäftsprozesses extrahiert und als Einheiten in den Descriptor eingepflegt. Dies sind die Kundendatenbank sowie die beiden Aktivitäten Ëingang des Anspruchsünd "Beantwortung des Anspruchs".

Nun können die Regeln integriert werden. Für die erste Anforderung werden dafür zwei Stück benötigt. Die erste Regel ”gefolgtVon”überprüft die Reihenfolge der ihr übergebenen Aktivitäten. Dabei muss die erste Aktivität vor der Zweiten kommen. Sie kann bereits während der Designphase validiert werden. Die zweite Regel ”maxZeitZwischenAktivitäten”kann dagegen erst während der Ausführung überprüft werden und misst dazu die Zeit, die zwischen den Ausführungen der zwei Aktivitäten verstreicht. Ist die Zeitspanne zu groß, schlägt die Validierung der Regel fehl. Die zweite Anforderungen benötigt für ihre Funktion nur eine Regel. Mit äuslieferungsort”wird der Ort der Auslieferung überprüft. Dazu wird eine Komponente und ein Ort übergeben. Bei der Auslieferung wird überprüft ob der Ort an den die Auslieferung der Komponente stattfindet, durch den übergebenen Parameter abgedeckt wird.

Im letzten Schritt werden die Compliance-Ausdrücke der Anforderungen geschrieben. Mit Hilfe des Ausdrucksedit or können dazu einfach die entsprechenden Regeln und Einheiten zusammengeklickt werden. Das Endresult zeigt das folgende Bild:

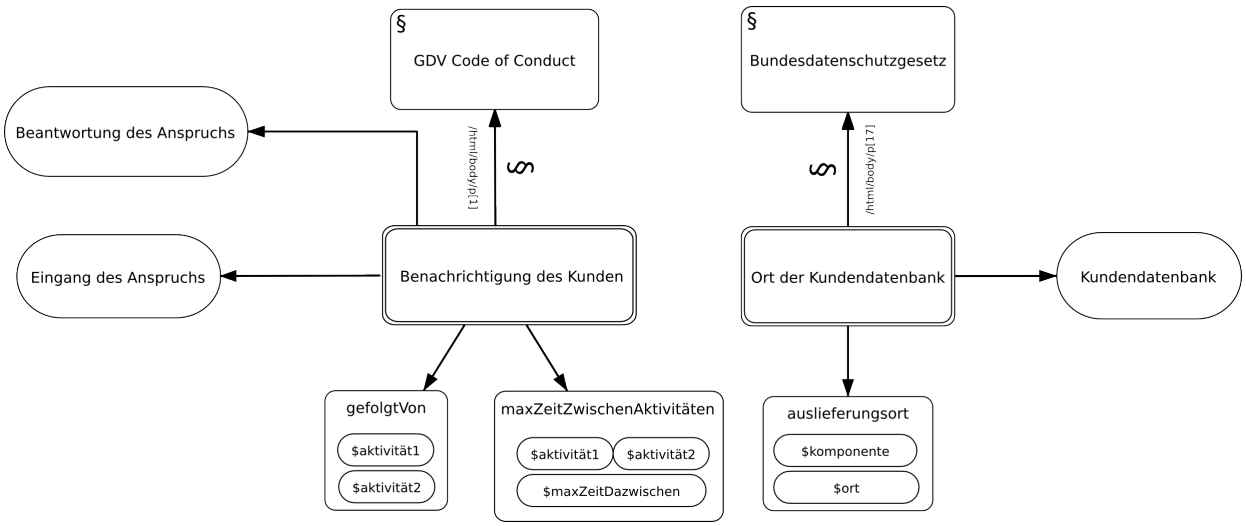

Abbildung 6.2.: Compliance Descriptor

Das dazugehörige XML-Dokument, das durch die Export-Funktion des Editors erstellt wurde, befindet sich im Anhang dieser Arbeit (A.1.1)

7. Mobiler Prototyp

Der Anteil der Menschen, die statt eines Laptops oder PCs lieber eine portablere Alternative wie ein Tablet oder Smartphone verwenden, wird immer größer. Diesen Anforderungen sollte ein Editor daher heute genügen.
Oryx war jedoch nie für die Verwendung auf kleinen Bildschrimen konzipiert und ist dort auch nicht gut verwendbar. Als Beispiel zeigen die zwei folgenden Bilder die Ansicht des Editors auf einem iPhone der ersten Generation.

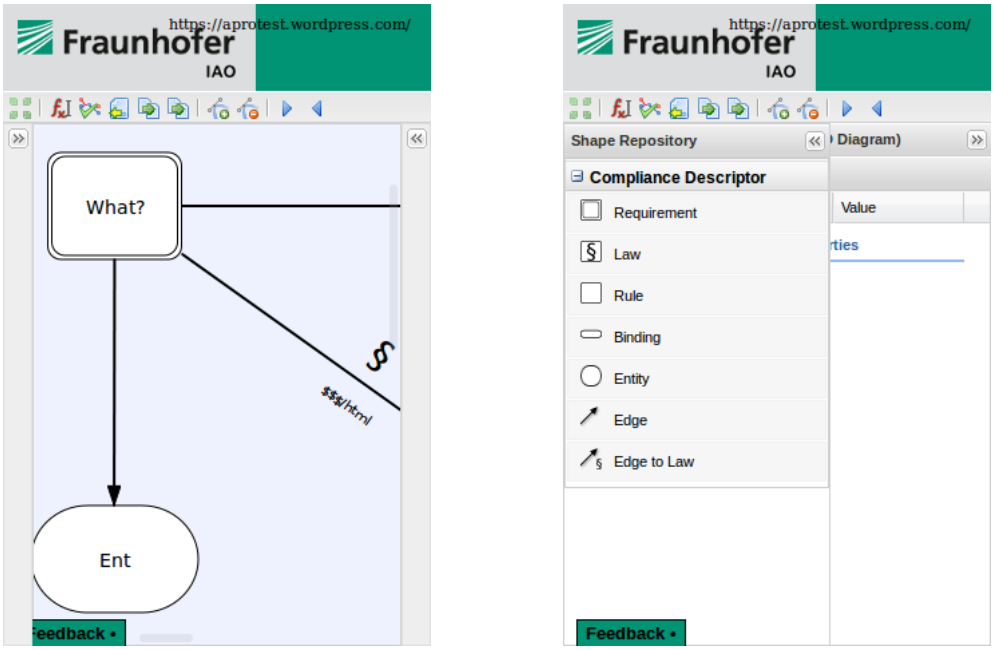

Abbildung 7.1.: Oryx-Editor auf einem iPhone der ersten Generation

Der Inhalt des Compliance Descriptors ist nur mit langwierigem Scrollen oder in weit herausgezoomten Zustand sichtbar. Elemente lassen sich zwar auswählen, das Verschieben von Elementen ist jedoch nicht möglich. Deshalb lassen sich auch keine neuen Elemente hinzufügen oder zwei Elemente verbinden. Die Eigenschaften von Elementen lassen sich bearbeiten, durch die kleine Darstellung von Eingabefeldern und Knöpfen ist es jedoch sehr schwierig die korrekten Felder zu selektieren.
Die Toolbar am oberen Rand enthält so viele Elemente, dass diese auf fünf Seiten verteilt sind. Sowohl die Toolbarknöpfe als auch die Knöpfe zum Wechseln der Toolbarseiten sind mit dem Finger nicht ausreichend genau zu bedienen.

Auf Grund dieser Nachteile des Oryx-Editors und um auch für diesen Anwendungsfall gerüstet zu sein, soll neben dem grafischen Webeditor auf Basis von Oryx noch eine Alternative für mobile Endgeräte zumindest prototypisch implementiert werden.
Um nicht für jede mobile Plattform eine komplette Implementierung durchführen zu müssen, soll dies mit Hilfe einer plattformübergreifenden Lösung entwickelt werden. Dabei steht eine Vielzahl von Lösungen zur Verfügung, durch eine Vorselektierung wurde die Auswahl jedoch auf die folgenden vier reduziert:

- **Xamarin Platform**[1]
  Auf der Programmiersprache C# und der Laufzeitumgebung Mono aufsetzendes Produkt, das die Generierung von Anwendungen für alle drei großen Plattformen erlaubt. Während die Benutzeroberfläche für jede Plattform einzeln entwickelt werden kann, um die unterschiedlichen Gepflogenheiten einhalten zu können, können alle gemeinsam genutzen Funktionalitäten in Bibliotheken für alle Anwendungen zusammengefasst werden. Für Anwendungszwecke die über einfache Anwendung herausgehen und einfach auf Smartphones installiert werden können, müssen für Xamarin monatliche Gebühren bezahlt werden.

- **Adobe PhoneGap / Apache Cordova**[2]
  Mit HTML, CSS und Javascript erstellte Webseiten verpackt PhoneGap zusammen mit plattformspezifischen Code zu Apps. Während der Entwicklung kann deshalb das Meiste im Browser getestet werden ohne einen Emulator oder ein Smartphone verwenden zu müssen. Eigene Javascript-Bibliotheken sowie Plugins erlauben den Zugriff auf die Kamera, die Orientierungssensoren oder den Kompass des Handys. PhoneGap ist frei unter der Apache License erhältlich und damit Open Source.

- **Appcelerator Titanium**[3]
  Ein auf Javascript aufbauendes Framework, das durch eine eigene XML-basierte Frontendsprache native Designelemente verwenden kann. Unterschiede zwischen den Plattformen können durch spezifischen Code implementiert werden. Dadurch werden nicht alle Teile des Codes auf allen Plattformen wiederverwendet. Zusätzlich bietet Titanium eine eigene MVC (Model-View-Controller) Implementierung und ein Framework zur Erstellung eines Datenbackends sowie für Pushnachrichten. Weite Teile der Appcelerator Angebote sind Open Source und unter der Apache License freigegeben.

- **Adobe AIR**[4]
  Basierend auf dem Flash Player und der Action Script Programmiersprache erweitert AIR die Reichweite von Flash auf mobile Endgeräte. Das Layout wird hierbei mit XML-Dateien beschrieben und mit Flash programmiert. Der Code ist auf allen unterstützten Plattformen ohne Änderungen einsetzbar, es lassen sich jedoch auch Plugins nutzen, die dem Code native Funktionalitäten zur Verfügung stellen. Für die Entwicklung von AIR Anwendungen wird ein Abonnement der Adobe Creative Cloud benötigt, das monatliche Kosten verursacht.

Da der Fokus dieser Diplomarbeit auf der Entwicklung eines grafischen Webeditors liegt, ist es naheliegend hier ebenfalls auf die Webtechnologien HTML, CSS und Javascript zu setzen. Damit scheiden bereits zwei Lösungen, Xamarin und AIR, direkt aus. Im Vergleich der zwei verbleibenden Technologien stellt sich die Frage, mit welchen Bibliotheken man bereits vertraut

[1] http://xamarin.com/platform
[2] http://www.phonegap.com
[3] https://www.appcelerator.com/product/
[4] http://www.adobe.com/products/air/

ist. PhoneGap lässt einem komplett freie Wahl bei den verwendeten Bibliotheken, während Appcelerator Titanium dabei in großen Teilen auf die Verwendung der eigenen mitgelieferten Bibliotheken besteht.
Durch die Verwendung von PhoneGap ergibt sich dadurch ein schnellerer Einstieg. Da nur ein Prototyp entwickelt werden soll, sollte dieser Faktor hierbei nicht unterschätzt werden.

7.1. Aufbau des Prototypen

Bestehen wird der Prototyp aus nur einem Screen, dem Editor für das Compliance Descriptor Diagramm.
Dieser Editor nimmt dabei auch den Großteil der Fläche ein. Er wird ergänzt durch eine Navigationsleiste am oberen Rand sowie eine Schaltfläche am unteren Rand, die das Hinzufügen von neuen Elementen ermöglicht. Die Navigationsleiste enthält auf großen Bildschirmen drei verschiedene Knöpfe:

- **Import**
  Erlaubt das Importieren von Diagrammen aus dem Oryx-Editor im JSON-Format. Eine entsprechende Export-Funktion wurde dem Oryx-Editor in Kapitel X hinzugefügt.

- **Export**
  Exportiert das bearbeitete Diagramm wieder in das JSON-Format. Es kann daraufhin im Oryx-Editor über die ebenfalls hinzugefügte Import-Funktion wieder importiert werden kann.

- **Einstellungen**
  Hier können die globalen Einstellungen des Diagramms, wie z.B. dessen Name, bearbeitet werden.

Auf kleinen Bildschirmen sind diese Knöpfe versteckt und können durch einen Klick bzw. eine Berührung des Menüknopfes angezeigt werden.

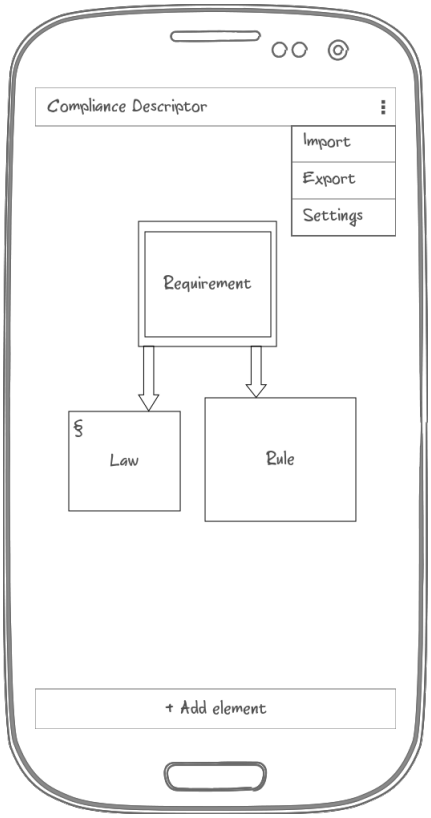

Abbildung 7.2.: Mockup der Mobile App

Alle Knöpfe öffnen sogenannte modale Dialoge. Diese Dialoge sperren den Rest der Anwendung, während sie angezeigt werden. Das ist für die kleinen Bildschirme der mobilen Endgeräte, mit denen diese App verwendet werden soll ideal, da sich der Benutzer so auf den Inhalt des Dialogs konzentrieren kann, während der Rest der Applikation ausgeblendet bzw. im Hintergrund ist.
Die Dialoge enthalten im Hauptteil meist ein Formular und zwei Buttons am unteren Ende um die Aktion abzubrechen oder sie abzuschließen. Folgende modale Dialoge sind Bestandteile der Anwendung:

- **Importdialog**
  Öffnet sich nach Betätigen des Importknopfes in der Navigationsleiste und enthält ein Textfeld um den JSON-Export aus dem Oryx-Editor einzufügen. Nach der Bestätigung mit dem ImportButton wird das momentane Diagramm geleert und durch das, aus dem Inhalt des Textfeldes, geladene Diagramm ersetzt.

- **Exportdialog**
  Durch Drücken des Exportknopfes öffnet sich dieser Dialog. Er enthält wie der Import-Dialog ein Textfeld. Dieses ist mit dem aus dem aktuellen Diagramm generierten JSON-Export gefüllt.

- **Einstellungsdialog**
  Der dritte Navigationsleistenknopf öffnet den Einstellungsdialog. Dieser erlaubt die Veränderung der globalen Diagrammeinstellungen. Momentan lässt sich dort der Name des Compliance Descriptor Diagrammes ändern.

- **Hinzufügendialog**
  Der am unteren Bildschirmrand befindliche Hinzufügenknopf öffnet einen Dialog mit fünf Knöpfen, je einen für jeden Typ von Elementen des Compliance Descriptors. Durch Drücken eines dieser Knöpfe lässt sich ein Element des entsprechenden Typs zum Diagramm hinzufügen.
  Sie werden standardmäßig am oberen linken Bildschirmrand platziert und können vom Benutzer per Drag and Drop an die korrekte Stelle im Diagramm gezogen werden. Ihre Eigenschaften sind zu Beginn leer und müssen durch den Benutzer gesetzt werden.

- **Bindungsdialog**
  Einen Spezialfall gibt es für das Hinzufügen einer neuen Bindung. Wird dieses Element im Hinzufügendialog ausgewählt, so öffnet sich ein zweiter Dialog, der die Auswahl der Regel, zu der diese Bindung gehört, erlaubt.

- **Elementdialog**
  Per Klick auf ein Element im Diagramm öffnet sich der Elementdialog. Er enthält die Eigenschaften des jeweiligen Elementes und erlaubt es diese zu bearbeiten. Durch Bestätigen des Dialogs werden diese anschließend abgespeichert und stehen im Export zur Verfügung. Zudem besteht die Möglichkeit, durch Drücken des Löschen-Knopfes das Element aus dem Diagramm zu entfernen. Dadurch werden auch eventuell bestehende Verbindungen und enthaltene Elemente gelöscht.

Der wichtigste Bestandteil der Anwendung ist der Editorbereich. In diesem werden alle aktiven Elemente des Compliance Descriptor Diagramms angezeigt und können bearbeitet werden. Durch Drag and Drop lässt sich ihre Position und mit Hilfe des oben beschriebenen Elementdialog ihre Eigenschaften verändern. Neue Verbindungen zwischen einer Anforderung und einem zweiten Element sollen sich ebenfalls per Drag and Drop erstellen lassen. Dazu wird es ein spezielles Element geben, von dem aus sich Verbindungen ziehen lassen werden. Der Spezialfall der Bindungen wird zwar ebenfalls durch eine Verbindung dargestellt, diese kann aber nur beim Hinzufügen eines Elements durch die Auswahl einer Regel im Bindungsdialog erstellt werden.
Durch Klicken auf eine Verbindung kann diese zudem wieder entfernt werden. Besteht die Verbindung zwischen einer Regel und einer Bindung, so entfernt diese auch die Bindung, da es sonst ohne eine Verbindung im Diagramm liegen würde.

7.2. Verwendete Bibliotheken

PhoneGap stellt keine Anforderungen an die von uns verwendeten Bibliotheken, sondern erfordert nur, dass sich am Ende alle verwendeten Dateien in einem Verzeichnis befinden und die *index.html*-Datei den Einstieg in die Anwendung darstellt.

Die Verwaltung der verwendeten Bibliotheken sowie deren Versionen und Abhängigkeiten vereinfacht bower[5]. Bower ist ein Paketmanagementtool, das für CSS- und Javascript-Abhängigkeiten konzipiert wurde, also genau diesen Anwendungsfall trifft.
Dafür wird im Hauptverzeichnis der Anwendung eine *bower.json*-Datei angelegt, die im JSON-Format die Abhängigkeiten der App und deren Versionen enthält. Mit einem Aufruf von *bower install* werden diese in das *bower_components*-Verzeichnis installiert und können anschließend in der *index.html*-Datei eingebunden werden.

[5] http://www.bower.io

CSS

Um für die Gestaltung der Anwendung ein solides Grundgerüst zu erhalten, wird Bootstrap[6] verwendet, ein CSS-Framework, das 2011 von Twitter-Entwicklern ins Leben gerufen wurde und inzwischen von vielen Webseiten verwendet wird. Für den Prototypen benötigt man nur einen Bruchteil der Funktionalität von Bootstrap, es bildet jedoch ein passendes Fundament, auf dem die Funktionalitäten aufbauen können.

Erweitert wird das Ganze um die Positionierung von Navigation und Hinzufügen-Button und den Style der einzelnen Elemente des Compliance Descriptors. Dieser ist möglichst nahe am Stil des Stencilsets im Oryx-Editor gehalten.

Javascript

Die Basis für den Javascript-Code bildet beim Prototypen Backbone.js[7] (im weiteren Verlauf Backbone genannt). Diese Bibliothek stellt vier simple Klassen zur Verfügung, die vom Benutzer erweitert werden können.
Models enthalten die Daten und simple Businesslogik und werden in Collections gesammelt, die im Vergleich zu normalen Arrays erweiterte Funktionalitäten sowie Events beim Hinzufügen, Ändern und Löschen zur Verfügung stellen. Views helfen dabei Teile der Webseite zu dynamisieren. Sie vereinfachen das Rendern und Hinzufügen neuer Elemente sowie das Abfangen von Events.

Model und View bilden den Model-View Teil des Model-View-Controller Konzeptes ab, eine Controllerkomponente ist bei Backbone nicht vorgesehen. Normalerweise vom Controller übernommene Aufgaben werden deshalb im View oder vom zusätzlich integrierten Router übernommen, der sich um die Übergänge zwischen verschiedenen Seiten kümmert. Da die Mobilanwendung nur über eine Seite verfügen wird, kommt sie ohne einen Router aus und kann die Controllerfunktionalität dem View übergeben.

Backbone hat zwei Abhängigkeiten, die von der App ebenfalls verwendet werden um die Entwicklungsarbeit zu vereinfachen. jQuery[8] ist die wahrscheinlich am weitesten verbreitete Javascriptbibliothek. Sie stellt Funktionen zur Selektierung von Elementen, zur Manipulation dieser Elemente und für Anfragen an den Server zur Verfügung.
Die zweite Abhängigkeit ist Underscore.js[9], eine Bibliothek desselben Entwicklers, die zum einen Funktionen neuerer Javascriptstandards für alte Browser nachimplementiert, zum anderen Komfortfunktionen z.B. für kürzere Abfragen oder Objekt- und Arraymanipulationen hinzufügt.

Erweitert wird Backbone durch Backbone.Relational, einer Bibliothek die es erlaubt Beziehungen zwischen verschiedenen Modellen zu definieren. Beim Einlesen der Daten werden diese bekannten Beziehungen dazu genutzt, automatisch die korrekte Collection zu verwenden und diese mit den korrekten Modellinstanzen zu befüllen. Ohne diese Funktion müsste man jedes Model einzeln instanziieren und mit den richtigen Modellen in Verbindung bringen.

[6] http://www.getbootstrap.com
[7] http://backbonejs.org
[8] http://www.jquery.com
[9] http://underscorejs.org

Wichtigstes Element der mobilen Anwendung ist der Diagrammeditor. Dafür wird jsPlumb[10] verwendet, eine Bibliothek die auf die Darstellung und Veränderung von Graphen spezialisiert ist. Um das Diagramm darzustellen werden HTML-Elemente innerhalb des Editor-Containers erstellt, die jeweils ein Element repräsentieren. Eine Funktion verbindet diese Elemente anschließend über ihre Identifizierer und erstellt den entsprechenden Code um diese Verbindungen im Browser darzustellen.
jsPlumb integriert zudem eine Drag and Drop-Funktionalität, über die die Elemente verschoben werden können. Dazu wird jQuery UI[11] als Abhängigkeit eingebunden und dessen *draggable*-Modul verwendet.

Um die Touchfähigkeit der mobilen Endgeräte nutzen zu können, werden zudem Tocca.js[12] und jQuery TouchPunch[13] eingebunden. Während Tocca.js die vom Browser erstellten Events erweitert, um neben dem Klicken mit der Maus auch das Berühren des Bildschirms abzufangen, fügt TouchPunch diese Funktionen zu jQuery UI hinzu. Teile davon werden von jsPlumb verwendet um die Drag and Drop-Funktionalität zur Verfügung zu stellen. Dadurch ist das Verschieben der Elemente nach dem Einbinden von TouchPunch auch auf dem Smartphone möglich.

Drei kleinere Bibliotheken ergänzen die Auswahl an Bibliotheken. Diese fügen jeweils eine einzelne Funktion hinzu.
multiline[14] ermöglicht einfachere, mehrzeilige Zeichenketten. In älteren Javascriptversionen waren diese nur durch das Konkatenieren mehrerer einzeiliger Zeichenketten möglich. Neuere Versionen unterstützen dies zwar ohne Konkatenation, es muss jedoch am Ende jeder Zeile ein Rückwärtsschrägstrich stehen. multiline umgeht das Problem, indem die Möglichkeit Funktionen in Zeichenketten umwandeln zu können clever ausgenutzt wird. Wird dem *multiline*()-Aufruf eine Funktion übergeben, deren einziger Inhalt ein mehrzeiliger Kommentar ist, so wird der Inhalt des Kommentars als Zeichenkette zurückgegeben.
Da Oryx einzigartige Identifizierer generiert um Elemente des Diagramms unterscheiden zu können, muss auch die mobile Anwendung die Möglichkeit haben solche zu generieren. Oryx nutzt dabei sogenannte UUIDs, kurz für Universally Unique Identifier, in der Version 4 und stellt diesen einen *oryx_*-Prefix voran. Version 4 bedeutet dabei, dass die UUIDs zufällig oder pseudo-zufällig generiert wurden und nicht wie z.B. bei Version 1 auf Grundlage von Zeitstempeln generiert wurden.
Mit Hilfe von node-uuid[15] können diese zufälligen UUIDs generiert werden und anschließend für neue Elemente oder Verbindungen genutzt werden.
Die letzte der drei Bibliotheken, underscore.deepExtend[16], erweitert Underscore.js um eine neue Funktion, die die bereits existierende Funktion *extend* abwandelt. *extend* erweitert ein Javascriptobjekt (der erste übergebene Parameter) um eine Kopie der Inhalte der weiteren Parameter (ebenfalls Javascriptobjekte). Dabei iteriert die Funktion jedoch nur über die erste Ebene und übernimmt so Referenzen auf Objekte in der zweiten Ebene, anstatt diese ebenfalls zu kopieren. *deepExtend* geht dabei, wie der Name schon sagt, tiefer und kopiert auch alle weiteren Ebenen der Objekte. Dadurch verhindert man, versehentlich Teile von referenzierten Objekten zu ändern, die an anderer Stelle zu Fehlern führen könnten.

[10] http://www.jsplumb.org
[11] http://www.jqueryui.com
[12] https://gianlucaguarini.github.io/Tocca.js/
[13] http://touchpunch.furf.com/
[14] https://github.com/sindresorhus/multiline
[15] https://github.com/broofa/node-uuid
[16] https://github.com/mateusmaso/underscore.deepextend

7.3. Implementierung der mobilen Anwendung

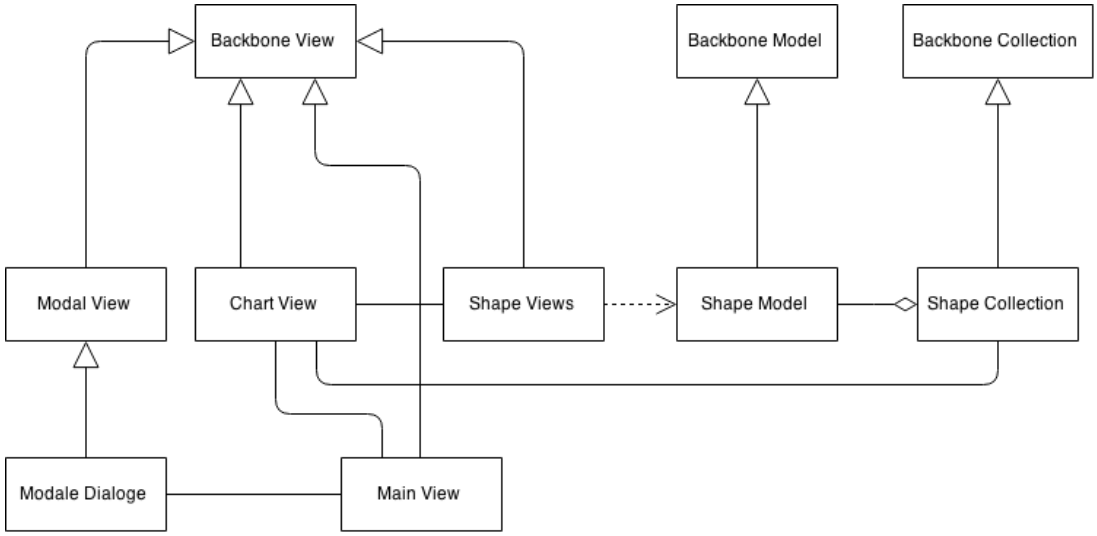

Abbildung 7.3.: Komponenten der mobilen Anwendung

Die einzelnen Formen des Compliance Descriptor Diagramms werden durch die Shape-Klasse, die vom Backbone-Model erbt, repräsentiert. Mit Hilfe von Backbone.Relational werden die jeweils enthaltenen Formen direkt eingebunden und können über die Relation abgefragt werden. Eine Collection dieser Shapes enthält das gesamte Diagramm.

Um die Ansichten der mobilen Anwendung zu implementieren, werden die Views von Backbone verwendet und ineinander geschachtelt. So enthält der Main-View den Chart-View, welcher wiederum viele Shape-Views enthält.
Der Main-View kümmert sich zudem um die von der Navigation sowie dem Hinzufügen-Knopf generierten Events. Auf Grund dieser Events werden modale Dialoge erzeugt, die alle auf der Modal-Klasse aufbauen, die ebenfalls von einem View erbt. Diese enthält speziellen Code, um die Darstellung und das Schließen des modalen Dialogs zu vereinheitlichen und damit die einzelnen Klassen der Dialoge deutlich schlanker zu machen. Darin muss noch der Inhalt des Dialogs angegeben werden und falls nötig, ein Event, der durch das Drücken des Bestätigen-Buttons ausgelöst wird, abgefangen werden. In dieser Funktion kann dann z.B. der Import des Diagramms aus dem eingegebenen JSON ausgelöst werden.
Eine Besonderheit stellt der Element-Modal dar, der durch das Klicken auf ein Element des Diagramms angezeigt wird. Dieser Dialog ist deutlich komplexer, da dabei die vorhandenen Eigenschaften des betroffenen Models ausgelesen werden müssen und als Formularfelder im Dialog angezeigt werden. Dabei wird anhand des jeweiligen Feldnamens entschieden, was für ein Feld im Formular angezeigt werden soll. Nach Bestätigen des Speichern-Buttons werden die Werte aus diesen Feldern wieder ausgelesen und zurück in das Model gespeichert.

Hauptaufgabe des Chart-View ist das Initialisieren von jsPlumb sowie das Anzeigen und die Eventverarbeitung des Diagramms. Um die Datenverarbeitung dabei möglichst nahe an der

grafischen Repräsentation zu halten, kümmert sich der Chart-View auch um den Im- und Export des Diagramms und initialisiert zu Beginn sowohl die Shape-Collection, die das Diagramm enthält, als auch ein Array, das deren Views enthält.

Das grafische Äquivalent des Shape-Models ist der Shape-View. Für jedes Model wird beim Import ein entsprechender View erstellt. Während beim Shape-Model nur eine Variante existiert, gibt es vom Shape-View für jede Form eine eigene Abwandlung der Klasse. Dadurch passt sich auch die Anzeige der entsprechenden Elemente im Diagramm an.
Da eine automatische Ausrichtung der Elemente an der vorhandenen Anzeigefläche von jsPlumb leider nicht zur Verfügung gestellt wird, muss die Position der Elemente aus dem Oryx-Editor verwendet werden. Nach dem Verschieben eines Elementes muss diese auch wieder zurückgespeichert werden, damit die Darstellung zwischen den zwei Applikationen möglichst wenig abweicht. Um die Anzeige von Elementen innerhalb der Anzeigefläche gewährleisten zu können, wird zudem eine Abweichung errechnet und beim Anzeigen und Speichern berücksichtigt werden. Dazu wird die minimale Position aller Elemente berechnet, um einen Mindestabstand verringert und bei der Anzeige von allen Positionen abgezogen werden.

7.4. Beispiel aus der Versicherungsbranche in der mobilen Anwendung

Um die Arbeit mit der mobilen Anwendung zu veranschaulichen, wird das Beispiel aus Kapitel 6 hier noch einmal verwendet und im mobilen Editor neu erstellt. Auch diesmal werden die beiden Anforderungen zuerst erstellt. Dazu wird der Hinzufügenknopf am unteren Bildschirmrand und der danach aufgehende Dialog verwendet. Der Name der Anforderungen muss anschließend über den Elementdialog gesetzt werden.

Mit Hilfe von Copy and Paste klappt auch das Hinzufügen der Gesetze und ihrer Texte ohne Probleme. Nach Herstellen der Verbindung zwischen Anforderung und Gesetz, kann im Elementdialog der Anforderung der XPath-Ausdruck gesetzt werden.

Analog dazu werden auch die drei Einheiten und die drei Regeln wieder erstellt. Beim Hinzufügen der Bindungen zeigt sich ein kleiner Unterschied, da hier nach dem Hinzufügen noch der Bindungsdialog aufgeht und die Auswahl einer Regel erfordert.

Das Erstellen der Compliance-Ausdrücke ist am mobilen Prototypen leider deutlich weniger benutzerfreundlich als im Oryx-Editor. Alle Teile des Ausdrucks müssen von Hand eingegeben werden. Und durch die nicht vorhandene Validierung kann auch danach die Korrektheit nicht überprüft werden. Das Endresultat der Arbeit zeigen die zwei folgenden Bildschirmfotos:

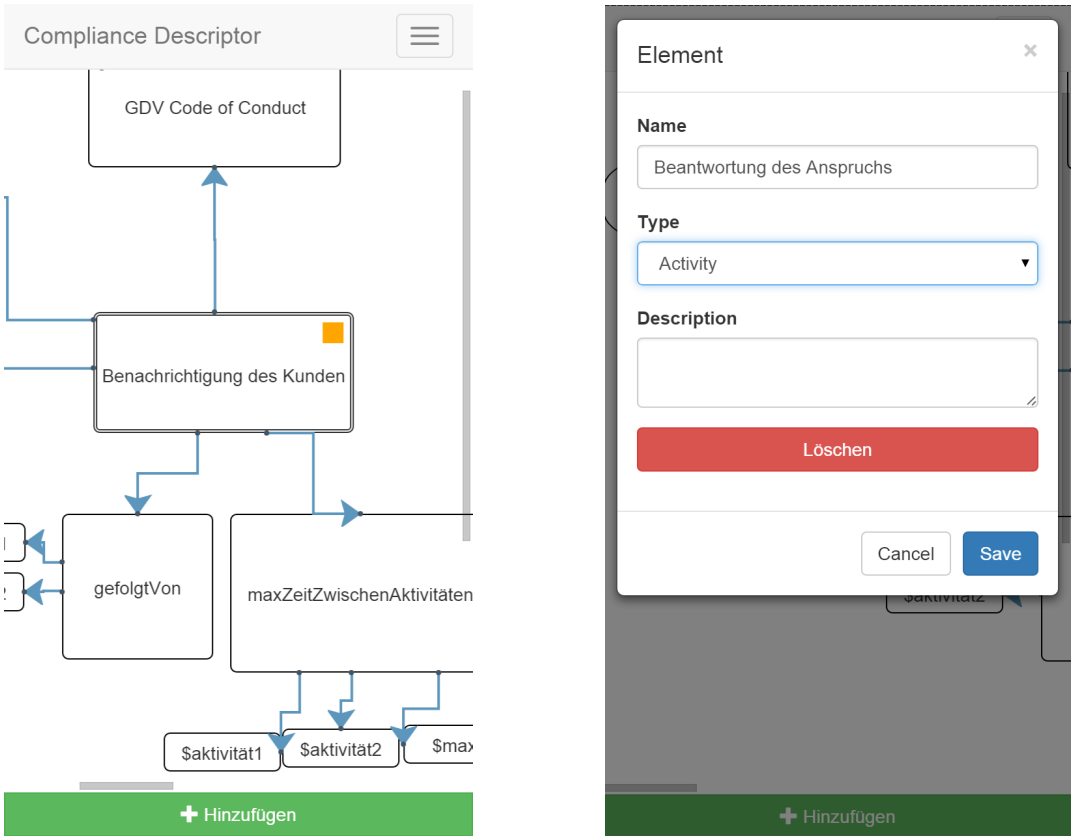

Abbildung 7.4.: Compliance Descriptor und Elementdialog auf mobilem Endgerät

Als letzter Schritt wurde zum Test zudem ein Export des Diagramms durchgeführt und mit Hilfe des JSON-Imports im Oryx-Editor wieder importiert. Dies funktioniert bis auf kleinere Darstellungsunterschiede exzellent, ändert jedoch nichts an der korrekten Umsetzung des Compliance Descriptors.

8. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurden aktuelle Verfahren zur grafischen Darstellung und Modellierung von Compliance-Anforderungen vorgestellt und untersucht. Basierend auf dem Compliance Descriptor wurde zudem eine solche Darstellung erstellt und mit Hilfe des Oryx-Editors implementiert. Für die komfortable Bedienung des Editors wurden zudem vier Erweiterungen hinzugefügt, die den Benutzer bei der Erstellung der Descriptoren unterstützen. Dadurch ist es jetzt möglich sich einen Compliance Descriptor einfach im Editor zusammenzuklicken und -zuziehen.
Zusätzlich wurde die Möglichkeit untersucht komplexe Diagrammeditoren auch auf mobilen Endgeräten zu nutzen und dazu der Prototyp einer mobilen Anwendung erstellt.

Sowohl der Oryx-Editor als auch der mobile Prototyp bieten noch einige Möglichkeiten Verbesserungen und Komfortfunktionen hinzuzufügen. So wären direkt verbundene Editoren für den Variability Descriptor oder LTL-Regeln, die Oryx prinzipiell bereits unterstützt, eine willkommene Hilfe beim Erstellen eines Compliance Descriptors. Auch die Auswahl von Gesetzesparagraphen über einen XPath-Editor ist vorstellbar.

Die mobile Anwendung bietet sicherlich die größten Weiterentwicklungsmöglichkeiten. Eine direkte Integration mit Oryx wäre dabei sicherlich am hilfreichsten. Dadurch könnten Modelle direkt am Oryx-Backend abgeholt und dort wieder gespeichert werden. Auch eine Anbindung an die Validierungs- und XML-Export-Schnittstellen wäre denkbar. Zudem wäre der Formeleditor auch auf Smartphones implementierbar und sogar nützlicher als am PC.

Literatur

[1] A. Awad. „BPMN-Q: A Language to Query Business Processes“. In: *In Proceedings of EMISA'07*. 2007, S. 115–128.

[2] A. Awad, G. Decker und M. Weske. „Efficient Compliance Checking Using BPMN-Q and Temporal Logic“. In: *In BPM '08: Proceedings of the 6th International Conference on Business Process Management*. 2008, S. 326–341.

[3] A. Awad, M. Weidlich und M. Weske. „Specification, Verification and Explanation of Violation for Data Aware Compliance Rules“. English. In: *Service-Oriented Computing*. Hrsg. von L. Baresi, C. Chi und J. Suzuki. Bd. 5900. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, S. 500–515. ISBN: 978-3-642-10382-7.

[4] A. Awad, M. Weidlich und M. Weske. „Visually specifying compliance rules and explaining their violations for business processes“. In: *Journal of Visual Languages and Computing* 22.1 (2011). Special Issue on Visual Languages and Logic, S. 30–55. ISSN: 1045-926X.

[5] S. Bensalem u. a. „An Overview of SAL“. In: *LFM 2000: Fifth NASA Langley Formal Methods Workshop*. Hrsg. von M. Holloway. NASA Langley Research Center. Hampton, VA, 2000, S. 187–196.

[6] T. Binz u. a. „TOSCA: Portable Automated Deployment and Management of Cloud Applications“. In: Advanced Web Services. Springer, 2014, S. 527–549. ISBN: 978-1-4614-7534-7.

[7] *Bundesdatenschutzgesetz(BDSG)*. 1990. URL: http://www.gesetze-im-internet.de/bundesrecht/bdsg_1990/gesamt.pdf.

[8] M. Czuchra. „Bachelorarbeit: Oryx-Embedding Business Process Data Into the Web“. In: *Final bachelor's paper, Hasso Plattner Institute at the University of Potsdam* (2007).

[9] Gesamtverband der Deutschen Versicherungswirtschaft e.V. (GDV). *GDV - Code of Conduct*. 2012. URL: http://www.gdv.de/wp-content/uploads/2013/03/GDV_Code-of-Conduct_Datenschutz_2012.pdf.

[10] R. Dijkman, M. Dumas und C. Ouyang. „Semantics and Analysis of Business Process Models in BPMN“. In: *Inf. Softw. Technol.* 50.12 (2008), S. 1281–1294.

[11] BITKOM und DIN. „Kompass der IT-Sicherheitsstandards“. In: (2009). URL: http://www.din.de/sixcms_upload/media/2896/Kompass%20der%20IT-Sicherheitsstandards.pdf.

[12] European Computer Manufacturers Association (ECMA). *The JSON Data Interchange Format*. 2013. URL: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

[13] M. El Kharbili. „CoReL: Compliance Representation Language“. In: (2012).

[14] M. El Kharbili u. a. „Corel: Policy-based and model-driven regulatory compliance management“. In: *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*. IEEE. 2011, S. 247–256.

[15] C. Fehling, F. Koetter und F. Leymann. *Compliance Modeling - Formal Descriptors and Tools*. Techn. Ber. 2014.

[16] Object Management Group. *Business Process Model And Notation (BPMN) Version 2.0*. 2011. URL: http://www.omg.org/spec/BPMN/2.0/.

[17] A. Halfmann. „Siemens versiebenfacht Zahl der Compliance-Mitarbeiter“. In: (2009). URL: http://www.dnwe.de/news-cg-extern/items/siemens-versiebenfacht-zahl-der-compliance-mitarbeiter.html.

[18] D. Knuplesch u. a. „On Enabling Data-aware Compliance Checking of Business Process Models“. In: *Proceedings of the 29th International Conference on Conceptual Modeling*. ER'10. Vancouver, BC, Canada: Springer-Verlag, 2010, S. 332–346. ISBN: 3-642-16372-6, 978-3-642-16372-2.

[19] M. Kochanowski u. a. „Compliance in BPM today - an insight into experts' views and industry challenges“. In: *Informatik 2014. Big Data - Komplexität meistern*. 2014.

[20] Regierungskommission Deutscher Corporate Governance Kodex. *Deutscher Corporate Governance - Kodex*. 2014. URL: http://www.dcgk.de//files/dcgk/usercontent/de/download/kodex/D_CorGov_Endfassung_2014.pdf.

[21] F. Koetter und M. Kochanowski. „Goal-Oriented Model-Driven Business Process Monitoring Using ProGoalML“. In: *Business Information Systems*. Hrsg. von W. Abramowicz, D. Kriksciuniene und V. Sakalauskas. Bd. 117. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2012, S. 72–83. ISBN: 978-3-642-30358-6.

[22] Falko Koetter u. a. „Unifying Compliance Requirements across Business and IT“. In: *Proceedings of the IEEE EDOC Conference*. IEEE, 2014, S. 1–10.

[23] F. Koetter u. a. „Unifying Compliance Management in Adaptive Environments through Variability Descriptors (Short Paper)“. In: *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*. 2013.

[24] E. Krügler. „Compliance - ein Thema mit vielen Facetten“. In: *VDI UmweltMagazin, 7-8/2011* (2011). URL: http://www.hlfp.de/fileadmin/redaktion/1._Aktuelles/Fachartikel/Compliance_-_ein_Thema_mit_vielen_Facetten.pdf.

[25] Linh Thao Ly. „SeaFlows - A Compliance Checking Framework for Supporting the Process Lifecycle“. 2013.

[26] L. Ly u. a. „SeaFlows Toolset – Compliance Verification Made Easy for Process-Aware Information Systems“. English. In: *Information Systems Evolution*. Hrsg. von P. Soffer und E. Proper. Bd. 72. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2011, S. 76–91. ISBN: 978-3-642-17721-7.

[27] R. Mietzner u. a. „Variability modeling to support customization and deployment of multi-tenant-aware Software as a Service applications“. In: *Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems*. IEEE Computer Society. 2009, S. 18–25.

[28] B. Oestereich u. a. „Objektorientierte Geschäftsprozessmodellierung mit der UML“. In: Test, 2003, S. 223.

[29] OpenID. *OpenID Authentication 2.0*. 2007. URL: https://openid.net/specs/openid-authentication-2_0.html.

[30] Oracle. *javax.servlet.Servlet*. 2011. URL: http://docs.oracle.com/javaee/7/api/javax/servlet/Servlet.html.

[31] N. Peters. „Oryx Stencil Set Specification“. In: *Final bachelor's paper, Hasso Plattner Institute at the University of Potsdam* (2007).

[32] D. Polak. „Bachelorarbeit: Oryx - BPMN Stencil Set Implementation“. In: *Final bachelor's paper, Hasso Plattner Institute at the University of Potsdam* (2007).

[33] S. Sadiq, G. Governatori und K. Naimiri. „Modeling control objectives for business process compliance“. In: *In Proceedings of BPM 2007*. 2007.

[34] Prof. Dr. Gerhard Schewe. *Gabler Wirtschaftslexikon, Stichwort: Geschäftsprozess*. 2015. URL: http://wirtschaftslexikon.gabler.de/Archiv/5598/geschaeftsprozess-v11.html.

[35] W. Tscheschner. „Bachelorarbeit: Oryx Dokumentation“. In: *Final bachelor's paper, Hasso Plattner Institute at the University of Potsdam* (2007).

[36] United States Code. *Sarbanes-Oxley Act of 2002, PL 107-204, 116 Stat 745*. Codified in Sections 11, 15, 18, 28, and 29 USC. 2002. URL: http://files.findlaw.com/news.findlaw.com/cnn/docs/gwbush/sarbanesoxley072302.pdf.

[37] World Wide Web Consortium (W3C). *Extensible Markup Language (XML) 1.1 (Second Edition)*. 2006. URL: http://www.w3.org/TR/2006/REC-xml11-20060816/.

[38] World Wide Web Consortium (W3C). *Scalable Vector Graphics (SVG) 1.1*. W3C Recommendation. 2011. URL: http://www.w3.org/TR/SVG11/.

[39] World Wide Web Consortium (W3C). *W3C XML Path Language Version 2.0*. W3C Recommendation. 2010. URL: http://www.w3.org/TR/xpath20/.

[40] World Wide Web Consortium (W3C). *W3C XML Schema Definition Language (XSD) 1.1*. W3C Recommendation. 2004. URL: http://www.w3.org/TR/xmlschema11-1/.

[41] T. Waizenegger u. a. „Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing“. In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*. Bd. 8185. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, S. 360–376. ISBN: 978-3-642-41029-1.

Alle Links wurden zuletzt am 01.06.2015 überprüft.

A. Anhang

A.1. Code-Listings

A.1.1. Compliance Descriptor aus Kapitel 6

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<compliancedescriptor name="Versicherungsanspruch"
    xmlns="http://comb.iao.fraunhofer.de">
    <rules>
        <rule id="gefolgtVon">
            <language>LTL</language>
            <phase>design-time</phase>
            <ruleExpression>
                <expression>([]($aktivitaet1 -&gt; &lt;&gt;
                    $aktivitaet2)))</expression>
            </ruleExpression>
            <bindings parameter="1" variabilityPoint="$aktivitaet1"/>
            <bindings parameter="1" variabilityPoint="$aktivitat2"/>
        </rule>
        <rule id="auslieferungsort">
            <language>TOSCA</language>
            <phase>deployment</phase>
            <ruleExpression>
                <expression>...</expression>
            </ruleExpression>
            <bindings parameter="1" variabilityPoint="$komponente"/>
            <bindings parameter="2" variabilityPoint="$ort"/>
        </rule>
        <rule id="maxZeitZwischenAktivitaeten">
            <language>ProgoalML</language>
            <phase>run-time</phase>
            <ruleExpression>
                <expression>...</expression>
            </ruleExpression>
            <bindings parameter="1" variabilityPoint="$aktivitaet1"/>
            <bindings parameter="2" variabilityPoint="$aktivitaet2"/>
            <bindings parameter="3" variabilityPoint="$maxZeitDazwischen"/>
        </rule>
    </rules>
    <requirements>
```

```
34          <requirement id="Benachrichtigung des Kunden">
35              <lawURLs>
36                  <law>GDV Code of Conduct</law>
37                  <paragraphReference>
38                      <xpath>/html/body/p[1]</xpath>
39                  </paragraphReference>
40              </lawURLs>
41              <complianceExpression>gefolgtVon("Eingang des Anspruchs" ,
                ↪  "Beantwortung des Anspruchs" )  AND
                ↪  maxZeitZwischenAktivitaeten( "Eingang des Anspruchs" ,
                ↪  "Beantwortung des Anspruchs" , "14 Tage"
                ↪  )</complianceExpression>
42          </requirement>
43          <requirement id="Ort der Kundendatenbank">
44              <lawURLs>
45                  <law>Bundesdatenschutzgesetz</law>
46                  <paragraphReference>
47                      <xpath>/html/body/p[17]</xpath>
48                  </paragraphReference>
49              </lawURLs>
50              <complianceExpression>auslieferungsort("Kundendatenbank",
                ↪  "Deutschland")</complianceExpression>
51          </requirement>
52      </requirements>
53      <laws>
54          <law title="Bundesdatenschutzgesetz">
55              <html>...</html>
56          </law>
57          <law title="GDV Code of Conduct">
58              <html>...</html>
59          </law>
60      </laws>
61      <entities>
62          <entities type="Component" name="Kundendatenbank"/>
63          <entities type="Activity" name="Eingang des Anspruchs"/>
64          <entities type="Activity" name="Beantwortung des Anspruchs"/>
65      </entities>
66  </compliancedescriptor>
```

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift