ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Formular-basierte Modellierung, Ausführung und Änderung von Prozessmodellen

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Janine Barner
janine.barner@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Jens Kolb

2011

Fassung 30. November 2011

Satz: PDF-LaTeX 2ε

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

In vielen Unternehmen hat sich Prozessmodellierung bereits etabliert, um interne Abläufe und Aufgaben in Form von Prozessmodellen festzuhalten. Wichtige Aspekte dabei sind das Erhöhen der Kundenzufriedenheit, Kostenreduktion und die Reduktion der Produktionskosten durch Prozessoptimierung und -automatisierung [22]. Für das Modellieren und Ausführen von Prozessen existieren etliche Tools und umfangreiche Business-Process-Management-Systeme (BPM-Systeme). Die Notation, in der Prozesse dargestellt werden variiert aber von System zu System. Dabei dienen graphenbasierte Notationen wie Petri-Netze, UML oder BPMN als Grundlage für die Darstellung von Prozessen.

Durch BPM-Systeme [16] können Daten und Aufgaben elektronisch festgehalten und verteilt werden. Komplexe Arbeitsschritte werden automatisiert und die Weiterleitung von Aufgaben wird zeitlich optimiert. Dabei können Aufgaben mehreren Mitarbeitern zugewiesen werden, was die Bearbeitung zusätzlich beschleunigt. Auch räumliche Grenzen spielen keine Rolle mehr und unternehmensübergreifende Aufgaben können leicht durchgeführt werden. Durch die Wiederverwendbarkeit der Prozessvorlagen werden routinierte Abläufe also optimal unterstützt und die Effizienz von Prozessen kann gesteigert werden. Die transparente Überwachung eines Prozesses ermöglicht außerdem ein schnelles Identifizieren von potentiellen Fehlerquellen. Durch die ständige Überwachung und Optimierung von Prozessen in Unternehmen kann eine Verbesserung in Zeit, Kosten, Qualität und Service erreicht werden.

Viele Systeme haben die grundlegenden Workflow-Konzepte adaptiert [22] und die graphenbasierte Prozessmodellierung übernommen. Durch die Darstellung eines Prozesses als gerichteter Graph ist es für Laien aber oft schwierig einen Prozessablauf nachzuvollziehen bzw. selbst zu modellieren, da Graphen im Arbeitsalltag selten auftauchen. Trotz

der gemeinsamen graphenbasierten Darstellung unterscheiden sich BPM-Systeme oft in ihren Notationen und Symbolen. Diese sind teilweise nicht intuitiv verständlich und Nutzer müssen erst eingelernt werden, um mit dem System und dessen Notation vertraut zu werden. Auch die Übersicht in einer graphenbasierten Darstellung wird in größeren Prozessen aufgrund vieler Kantenübergänge beeinträchtigt.

Aufgrund hohem Konkurrenzdruck in Unternehmen wird es jedoch immer wichtiger, dass ein Prozessmodell intuitiv verstanden wird. Ein intuitiver Umgang mit einem BPM-System hinsichtlich der Darstellung, der Modellierung und der Ausführung eines Prozesses ermöglicht eine effizientere Nutzung und spart dadurch Zeit und Kosten. Benutzerschulungen werden somit überflüssig und Arbeitsabläufe werden durch die intuitive Nutzung von Elementen beschleunigt.

Um einer breiteren Masse an Nutzern den Umgang mit solchen Systemen zu erleichtern, wird hier eine formularbasierte Darstellung vorgestellt. Formulare sind aus dem Alltag bekannt und können intuitiv verwendet werden. Datenelemente können als ausfüllbare Felder dargestellt werden, welche zur Laufzeit schrittweise gefüllt werden. Um den Umgang mit dem System zu erleichtern, wird die Zahl an Modellierungselementen auf die wichtigsten Elemente beschränkt. Durch einige interaktive Elemente können Daten einzelner Aktivitäten problemlos eingeklappt und damit ausgeblendet werden. Dies fördert erheblich die Übersicht eines Prozesses. Auch Kantenübergänge stören die Ansicht in dieser Darstellung nicht mehr. Eine intuitive Repräsentation von parallelen Aktivitäten bzw. einer Auswahlverzweigung ermöglicht auch Laien einen Prozess nachzuvollziehen und zu modellieren.

1.2 Zielsetzung

Zielsetzung dieser Arbeit ist es eine für unerfahrene Nutzer verständliche Darstellung von Prozessmodellen zu entwickeln, welche intuitiv verwendet werden kann. Hierfür wird eine formularbasierte Darstellung zur Prozessmodellierung und -ausführung entwickelt und deren Anwendbarkeit analysiert. Dabei steht immer im Vordergrund einfach verständliche Modellierungs-Elemente zu verwenden. Mit dieser neuen Notation werden Prozessmodelle als editierbares und ausführbares Formular dargestellt. Die vereinte Darstellung der Modellierung und Laufzeit dient dabei erheblich dem Verständnis. Außerdem werden verschiede-

ne Ansätze betrachtet, um ein Prozessmodell übersichtlicher zu gestalten.Dies wird hauptsächlich durch interaktive Elemente erreicht. Durch eine einfache Symbolik wird dem Nutzer visualisiert, wenn ein Element erweiterbar ist, und wann nicht. Animationseffekte sind dabei ein zusätzliches Hilfsmittel um den Vorgang nachzuvollziehen.

Auch die Modellierung eines Prozesses als Formular muss dementsprechend intuitiv und ohne Einarbeitungszeit möglich sein. Aktivitäten, Datenelemente und zusätzliche Modellinformationen, wie Bearbeiterzuordnung, müssen leicht modellierbar und editierbar sein. Hierfür werden entsprechende Konzepte vorgestellt. Zur Laufzeit muss die Darstellung den Nutzer unterstützen und leiten. So wird beispielsweise immer die aktuell zu bearbeitende Aktivität hervorgehoben. Auch weitere Konzepte wie Annotationen und eine Übersicht über bereits abgeschlossene Aktivitäten werden diskutiert.

Bevor die formularbasierte Darstellung erläutert wird, werden in Kapitel 2.1 die Grundlagen der Prozessmodellierung erläutert. In Kapitel 2.2 werden anschließend zwei verwandte Ansätze vorgestellt und diskutiert. Kapitel 3 erläutert die Grundelemente und die Funktion interaktiver Modellierungs-Elemente in der formularbasierten Darstellung. Des Weiteren wird ein komplexeres Beispiel umgesetzt, um die Machbarkeit des Konzeptes zu zeigen. Die Vorgehensweise der Modellierung eines formularbasierten Prozessmodells wird in Kapitel 5 beschrieben. Hierfür werden zwei unterschiedliche Konzepte vorgestellt und deren Anwendbarkeit diskutiert. Das Laufzeitverhalten wird in Kapitel 6 untersucht. Dazu werden interne Zustände als Statecharts abgebildet und eine unterstützende Darstellung der Laufzeitansicht beschrieben und zusätzliche Konzepte, wie Annotationen, vorgestellt. In Kapitel 7 erfolgt dann ein Resümee und eine Diskussion weiterer Aspekte und Einsatzmöglichkeiten.

2 Grundlagen und verwandte Arbeiten

Dieses Kapitel legt Grundlagen der Prozessmodellierung, die für das Verständnis der Arbeit wichtig sind. Dabei werden insbesondere die grundlegendsten Elemente des ADEPT-Basismodells in Kapitel 2.1 vorgestellt. Des Weiteren werden in Kapitel 2.2 verwandte Arbeiten angesprochen.

2.1 ADEPT-Basismodell

In diesem Kapitel werden die grundlegenden Elemente der Prozessmodellierung erläutert. Dazu werden insbesondere die Modellierungskonstrukte des ADEPT-Basismodells (auch Metamodell genannt) erklärt, da dieses sehr ausdrucksstark und auf Grund seiner Struktur leicht verständlich ist [13].

ADEPT[1] ist ein Forschungsprojekt der Universität Ulm. Im Bereich des Geschäfts-Prozess-Managements bietet es eine flexible Ausführung von Unternehmensprozessen, die Realisierung robuster prozessorientierter Anwendungen sowie ein einfach zu benutzendes System [17]. Als Workflow-Management-System unterscheidet ADEPT zwischen Prozessvorlage und Prozessinstanzen. Dabei dient die Prozessvorlage der Modellierung eines Prozessablaufs. Basierend auf einer Vorlage können beliebig viele Prozessinstanzen erstellt werden. Eine Prozessinstanz beschreibt einen laufenden Prozess. Außerdem bietet ADEPT die Möglichkeit von ad hoc Änderungen einer Prozessinstanz zur Laufzeit. Bereits gestartete Prozessinstanzen können somit modifiziert werden ohne die Korrektheit oder die Konsistenz zu gefährden(Vertiefung siehe [14] oder auch [13]).

Das ausdrucksstarke und flexible ADEPT-Basismodell dient im Weiteren als Grundlage der verschiedenen Modellierungskonstrukte. Es basiert auf markierten, attributierten Graphen und verfolgt einen blockstrukturierten Ansatz. Bei der Blockstrukturierung werden

[1] ADEPT (**A**pplication **D**evelopment Based on **P**re-Modeled Activity **T**emplates [8])

Kontrollstrukturen, wie Sequenzen und Verzweigungen auf Blöcke mit eindeutigem Start- und Endknoten abgebildet. Das Basismodell stellt die Konzepte für die Beschreibung von Kontroll- und Datenflüssen bereit. Dabei werden im Kontrollfluss Ausführungsreihenfolgen und -bedingungen festgelegt und im Datenfluss die Verknüpfung ihrer Ein- und Ausgabeparameter bestimmt. Außerdem definiert das Basismodell Regeln für die korrekte Strukturierung von Workflow-Modellen und ein korrektes dynamisches Verhalten [13].

2.1.1 Kontrollfluss

Der Kontrollfluss beschreibt die Reihenfolge der ausführbaren Aktivitäten. Die Modellierung von Kontrollflüssen basiert in ADEPT auf dem Konzept der regelmäßigen Blockstrukturierung, welche durch blockorientierte Programmiersprachen inspiriert wurde. Dabei werden Kontrollstrukturen, wie Sequenzen und Verzweigungen auf Blöcke abgebildet, welche einen eindeutigen Start und Endknoten besitzen. Wie in Abbildung 2.1 zu sehen, können diese Blöcke beliebig ineinander geschachtelt werden, dürfen sich aber nicht überschneiden. In Abbildung 2.1 sind alle elementaren Kontrollstrukturen vertreten. So kann man auch gut erkennen, dass bei einer parallelen Verzweigung (Startknoten B) zwei neue Blöcke entstehen, die durch den Endknoten I beendet werden.

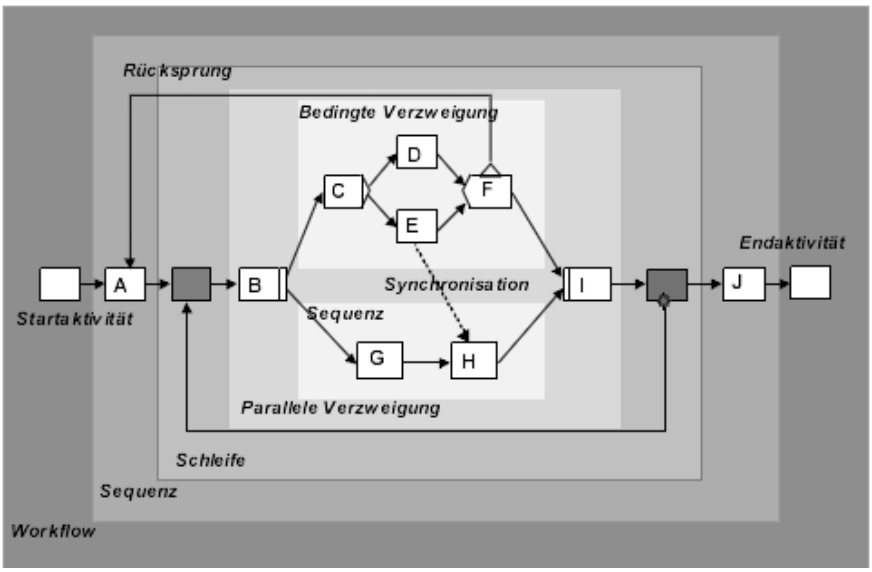

Abbildung 2.1: ADEPT Prozessmodell [13]

Diese Blockstrukturen werden auch Kontrollblöcke genannt; sie sind für die Strukturierung von Kontrollflussgraphen, sowie deren Analyse ein wichtiges Hilfsmittel [13].

Im Weiteren werden die elementaren Kontrollflusskonstrukte des ADEPT-Basismodells vorgestellt: Sequenz, parallele Verzweigung, bedingte Verzweigung und Schleifen. ADEPT bietet noch weitere Konstrukte, wie verschiedene Kantentypen. Diese werden hier jedoch nicht weiter erläutert.

Sequenz

Die Sequenz stellt das einfachste blockorientierte Kontrollflusskonstrukt dar. Aktivitäten werden sequenziell bearbeitet, also direkt hintereinander ausgeführt. Wie in Abbildung 2.2 zu sehen, werden die Aktivitäten A,B und C hintereinander mit zwischenliegenden Kontrollkanten dargestellt. Dabei muss gelten, dass die Elemente der Sequenz genau einen Vorgängerknoten und einen Nachfolgerknoten besitzen.

Abbildung 2.2: Sequenz-Kontrollblock

Verzweigungsarten

Da die sequenzielle Abbildung von Aktivitäten für komplexe Prozessabläufe nicht ausreicht, unterstützt ADEPT außerdem drei Verzweigungsarten: Parallele Verzweigung (AND), bedingte Verzweigung (XOR) und parallele Verzweigung mit finaler Auswahl. Zu beachten ist, dass eine Verzweigung immer einen eindeutigen Verzweigungsknoten (Split-Knoten) sowie einen korrespondierenden Synchronisationsknoten (Join-Knoten) besitzt. Im Folgenden werden die parallele und die bedingte Verzweigung weiter erläutert.

AND-Verzweigung

Die AND-Verzweigung entspricht einem parallelen Ablauf von Aktivitäten. Die Arbeitsschritte sind also voneinander unabhänging bearbeitbar. Wie in Abbildung 2.3 zu sehen, startet die parallele Verzweigung mit dem Split-Knoten A und endet mit dem Join-Knoten E. Kno-

ten E darf erst ausgeführt werden, wenn alle parallelen Aktivitäten B, C und D abgearbeitet sind.

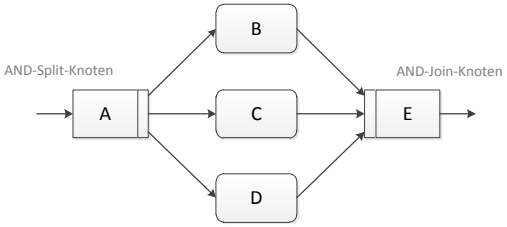

Abbildung 2.3: AND-Kontrollblock

XOR-Verzweigung

Eine XOR-Verzweigung beschreibt eine bedingte Aufgabenbearbeitung. Nur einer der Verzweigungspfade wird zur Laufzeit ausgewählt und bearbeitet. Die anderen Pfade werden nicht verarbeitet. Besonders häufig sind prädikative Verzweigungen. Dabei wird ein Prädikat im Verzweigungsknoten ausgewertet. Der Input für die Auswertung erfolgt über Datenelemente(s. Datenfluss). Für die Auswahl eines Knotens muss also die Entscheidungsaktivität des Split-Knotens erfüllt sein. Wie in Abbildung 2.4 zu sehen, ist das auszuwertende Prädikat a < b, wobei a und b Eingaben vom Typ Integer sind. Knoten B wird hier nur ausgeführt wenn die Bedingung true liefert. C wird ausgeführt wenn a < b false ist.

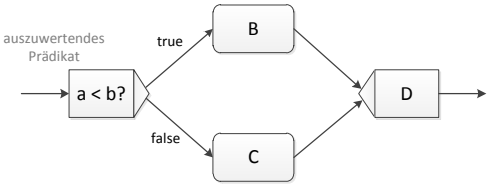

Abbildung 2.4: XOR-Kontrollblock

Schleife

Auch die Abbildung von Schleifen, also zyklische Abläufe in Prozessen, wird vom ADEPT-Basismodell unterstützt. ADEPT stellt eine Schleife auch als Kontrollblock dar und stellt somit, anders als z.B. bei Petrinetzen, ein explizites Konstrukt dafür bereit. Eine Schleife bestitzt einen eindeutigen Start- und Endknoten. Diese sind mit einer Schleifenrücksprung-

kante verknüpft. Die Abbruchbedingung oder auch Schleifenbedingung steht im Endknoten, diese entscheidet, ob die Schleife nochmal durchlaufen wird oder nicht.
Abbildung 2.5 zeigt exemplarisch eine for-Schleife. Zuerst wird der Schleifenzähler initialisiert. Dann folgen zwei Aktionen A und B nach denen der Schleifenzähler inkrementiert wird. Mit der Schleifenbedingung i < 5 wird die Schleife fünfmal durchlaufen.

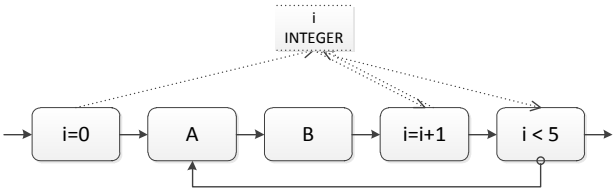

Abbildung 2.5: Schleifen-Kontrollblock

2.1.2 Datenfluss

Wie im Abschnitt 2.1.1 schon angesprochen, brauchen Prozesse zum Datenaustausch auch Dateneingaben und Datenausgaben. In ADEPT basiert der Datenfluss zwischen Knoten auf Prozessvariablen, welche Datenelemente genannt werden. Aktivitäten werden über Datenflusskanten mit Datenelementen verknüpft. Dabei entspricht eine Linie vom Knoten zu einem Datenelement einem Schreibzugriff und eine Linie von einem Datenelement zu einem Knoten einem Lesezugriff. In Abbildung 2.5 wird z.B. das Datenelement bei der Inkrementation gelesen und wieder geschrieben. Bei der Auswertung der Schleifenbedingung wird entsprechend gelesen. Aktivitäten und Datenelemente können im Allgemeinen auch mehrere aus- bzw. eingehende Datenkanten besitzen.

Für jedes Datenelement muss zur Entwurfszeit dessen Datentyp festgelegt werden. Somit werden Typkonvertierungsfehler zwischen Datenelementen und Aktivitäten ausgeschlossen. Einfache Datentypen sind z.B. Integer, Double, Boolean oder String.

Besonders wichtig bei der Modellierung des Datenflusses ist die Prüfung der korrekten Versorgung von Aktivitätenparametern. Es wird zwischen obligaten und optionalen Parametern unterschieden. Obligate Parameter müssen stets geschrieben werden, bevor sie gelesen werden; das heißt insofern einem obligaten Parameter kein Default-Wert zugeordnet ist, muss die korrekte Versorgung durch ein Datenelement sichergestellt sein.

2.1.3 Strukturierungsregeln

Damit Korrektheit und Konsistenz sichergestellt werden können, müssen bei der Modellierung bestimmte Regeln eingehalten werden. In [13] werden hierzu einige Strukturierungsregeln für Kontroll- und Datenfluss definiert. Die für diese Arbeit relevanten Regeln für einen korrekten Kontrollfluss werden im Folgenden erläutert.

Strukturierungsregel KF-1:
Es gibt genau einen Start- und einen Endknoten, wobei ein Startknoten keine eingehende Kante und ein Endknoten keine ausgehende Kante besitzt.

Strukturierungsregel KF-2:
Jeder Aktivitätenknoten außer des Start- und Endknotens besitzt eine eingehende und eine ausgehende Kontrollkante. Jeder Knoten besitzt also einen direkten Vorgänger und einen direkten Nachfolger.

Strukturierungsregel KF-3:
Die Blockstrukturierung muss eingehalten werden. Das heißt zu jeder Verzweigung an einem Split-Knoten muss es symmetrisch eine Zusammenführung an einem eindeutigen Join-Knoten geben. Eine AND-Verzweigung benötigt z.B. ein eindeutiges AND-Join.

Strukturierungsregel KF-4:
Bezüglich normaler Kontrollkanten dürfen im Graphen keine Zyklen auftreten. Das heißt es darf keine Folge von Kontrollkanten geben, in der ein Knoten mehrmals auftritt.

Die Korrektheit für den Kontrollfluss zu gewährleisten, ist mit den vorgestellten Regeln im Gegensatz zum korrekten Datenfluss einfacher. Folgende Regeln müssen für einen korrek-

ten Datenfluss eingehalten werden:

Strukturierungsregel DF-1:
Obligate Eingabeparameter müssen sicher versorgt werden. Jeder obligate Eingabeparameter eines Knotens muss mit genau einem Datenelement verbunden sein. Des Weiteren muss ein obligat gelesenes Datenelement von mindestens einem vorausgehenden Aktivitätenknoten geschrieben werden. Eine Ergänzung der Regel besagt zudem, dass das Verzweigungsdatenelement eines XOR-Split-Knotens spätestens nach der Beendigung des Knotens geschrieben sein muss.

Strukturierungsregel DF-2:
Ein paralleler Schreibzugriff auf Datenelemente ist unzulässig. In einer parallelen Verzweigung dürfen zwei Aktivitätenknoten, die in veschiedenen Verzweigungspfaden liegen, nicht das selbe Datenelement schreiben. So wird sichergestellt, dass es bei der Ausführung nicht zu Inkonsistenzen kommt.

Strukturierungsregel DF-3:
Direkt aufeinander folgende Schreibzugriffe sollten vermieden werden. Der Wert eines gerade geschriebenen Datenelements bleibt ungenutzt, wenn er durch die nächste Aktivität überschrieben wird. Ein solcher Zugriff muss nicht ausgeschlossen werden, der Nutzer sollte jedoch darauf hingewiesen werden.
Die vorgestellten Regeln sind für eine korrekte Ausführung sehr wichtig. Ansonsten kann es zu Dateninkonsistenzen kommen. Fehlende Eingabedaten und Überschreiben von Daten sind somit weitestgehend ausgeschlossen. Zur Überprüfung der Strukturierungsregeln existieren effiziente Verfahren, die hier aber nicht weiter erläutert werden.

In diesem Kapitel wurden Kontroll- und Datenflusselemente des blockorientierten ADEPT-Basismodells vorgestellt, welche die Grundlage für die Modellierung eines Prozessmodells bilden. Um Korrektheit und Konsistenz zur Laufzeit zu gewährleisten, wurden außerdem grundlegende Strukturierungsregeln erläutert.

2.2 Verwandte Arbeiten

In diesem Kapitel werden zwei unterschiedliche Ansätze einer interaktiven und benutzerfreundlichen Prozessmodellierung vorgestellt. In Kapitel 2.2.1 wird das Konzept der interaktiven Fragebögen erläutert, mit welchem die formularbasierte Konfiguration von Referenz-Prozessmodellen ermöglicht wird.

In Kapitel 2.2.2 wird eine SaaS-Lösung (Software as a Service) von IBM vorgestellt. IBM Blueworks Live ist ein Portal, um Aufgaben zu verteilen und zu überwachen.

2.2.1 Questionnaire Driven Configuration

Referenz-Prozessmodelle sind übergeordnete Modelle, die für ein bestimmtes Anwendungsszenario spezialisiert werden können. Bei vielen solcher Modelle ist die Konfiguration ohne Grundwissen der zugrunde liegenden Notation sehr schwierig. Es fehlt eine adäquate Repräsentation der möglichen Änderungen, Alternativen und insbesondere deren Zusammenhänge bzw. Abhängigkeiten.

Eine Möglichkeit realisierbare Konfigurationen darzustellen bilden interaktive Fragebögen (interactive questionnaires)[20]. Diese Fragen können beantwortet werden ohne ausführliches Wissen über das zugrunde liegende Referenz-Modell zu haben. Antworten werden dabei durch sogenannte Fakten (facts) beschrieben wobei die Anzahl an Fakten einer Frage endlich sein muss.
Zur Laufzeit werden dann von einem Nutzer in einem interaktiven Fragebogen die Fragen beantwortet. Dabei werden jedoch nur Fragen gestellt, die durch vorherige Fragen nicht ausgeschlossen werden konnten. Auch die Zahl der erlaubten Antworten ist abhängig von den vorherigen Antworten. Wird eine Antwort ausgewählt, werden die entsprechenden Fakten gesetzt und die Frage in der Übersicht der beantworteten Fragen angezeigt. Ein Rollback der Antwort ist so jederzeit möglich.

In Abbildung 2.6 ist eine mögliche Struktur von Fragen und Antworten zu einem gegebenen Prozess dargestellt. Ist der Default-Wert eines Fakts wahr, so wird er mit einem T (für true) markiert. Ein M (für mandatory) hinter dem Namen symbolisiert, dass dieser Fakt

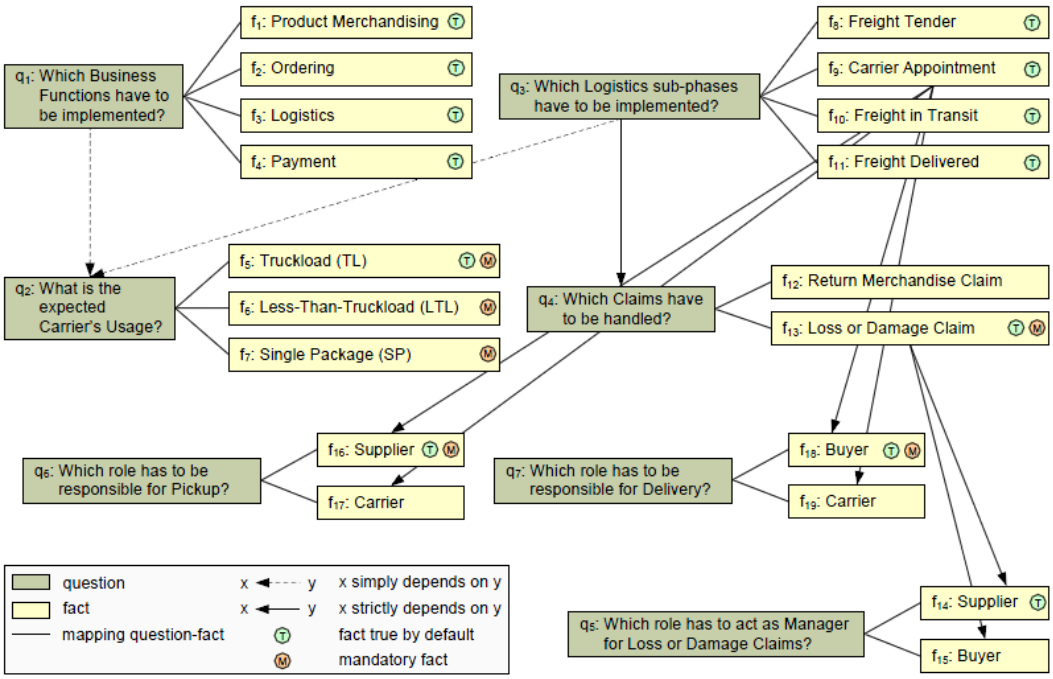

Abbildung 2.6: Strukur eines interaktiven Fragebogens [19]

explizit vom Nutzer gesetzt werden soll. Außerdem werden in der Abbildung die Abhängigkeiten von Fragen und Antworten sichtbar. Es wird unterschieden zwischen einfacher und strikter Abhängigkeit. Eine einfache Abhängigkeit wird durch einen gestrichelten Pfeil dargestellt. Sie wird genutzt, wenn eine Frage von einer anderen abhängen könnte. Eine strikte Abhängigkeit wird dagegen verwendet, um eine zwingende Reihenfolge von zwei Fragen festzulegen. Sie wird durch einen durchgezogenen Pfeil dargestellt.

Abhängigkeiten geben also Aufschluss über die Reihenfolge der Fragen. Um die Korrektheit einzuhalten, müssen aber noch explizite Constraints aufgestellt werden. Diese können teilweise auch über Fragen definiert werden. Eine Belegung der Fakten entspricht demnach nur dann einer gültigen Konfiguration, wenn alle Constraints eingehalten wurden. Formale Constraints zu Abbildung 2.6 und ausführliche Definitionen für die Generierung eines Fragebogens können in [19] nachgeschlagen werden.

Um zu veranschaulichen, wie der Nutzer mit interaktiven Fragebögen umgehen kann wurde ein formularbasiertes Tool erstellt. Der Nutzer wird hier durch den kompletten Konfigurati-

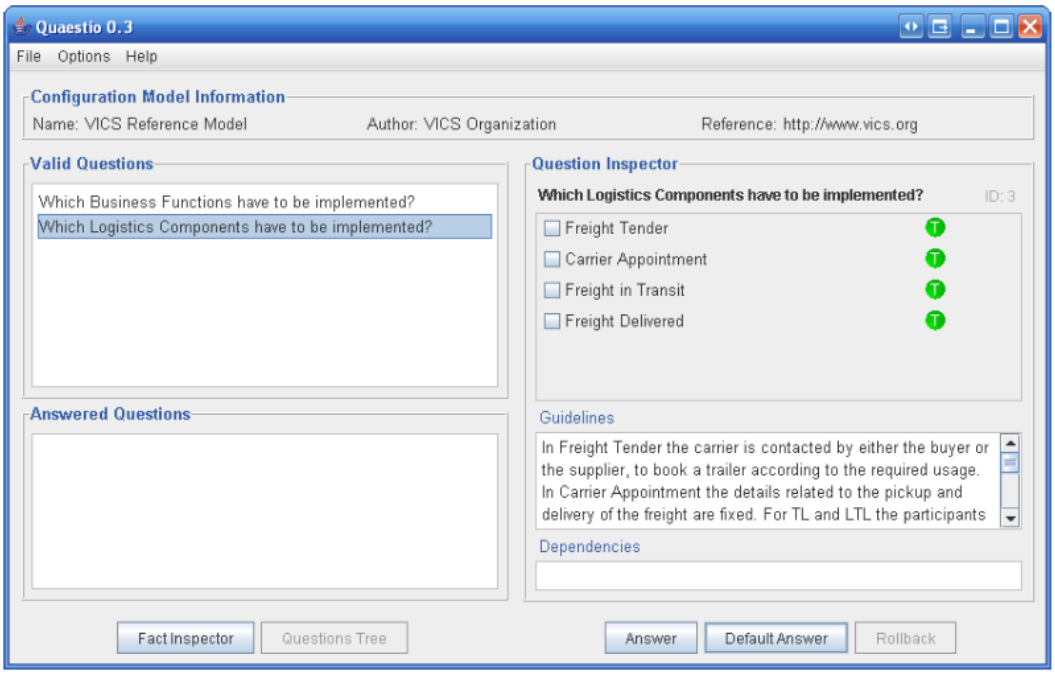

Abbildung 2.7: Quaestio

onsvorgang geleitet. Wie in Abbildung 2.7 zu sehen, besteht das Hauptfenster aus zwei Hauptteilen: Links wird eine Liste der offenen und darunter eine Liste der beantworteten Fragen dargestellt. Auf der rechten Seite befindet sich der Fragen-Inspektor (Questions Inspector), welcher die Details zu einer ausgewählten Frage darstellt. Dazu gehören Abhängigkeiten und Richtlinien für die Konfiguration der Frage. Das Eingabeformat eines Modells wird in einer XML-Schema-Datei beschrieben (siehe [18]).

Unter anderem bietet der Ansatz zwei besondere Eigenschaften: Zum einen werden überspringbare Fragen automatisch beantwortet, zum anderen kann es auf Nachfrage eine Konfiguration selbstständig beenden, insofern alle notwendigen Fakten beantwortet wurden. Zudem kann bei jeder Frage ein Rollback vorgenommen werden.

Dieser Ansatz zeigt, wie es mit einem einfachen Ansatz möglich ist, einen Nutzer bei der Konfiguration eines Prozessmodells zu unterstützen. Der Nutzer muss kein Experte sein und braucht kein Vorwissen über die Notation des Prozess-Modells. Er wird durch die Konfiguration geleitet und bestimmte Aufgaben kann das System selbstständig übernehmen.

2.2.2 IBM Blueworks Live

An dieser Stelle wird das Konzept von IBM's SaaS-Lösung (Software as a Service) Blueworks Live vorgestellt.

Blueworks Live ist ein Cloud-basierter BPM Service der aus IBM's BPM BlueWorks und Blueprint entstanden ist [21]. Charakterisierend sind die Community- und Collaboration-Features, welche Nutzer bei der Dokumentation und dem Management von Prozessen unterstützen. Auch die Automatisierung von einfachen Prozessen ist durch die Erstellung einer Prozessapplikation möglich. In dem integrierten grafischen Wizard lassen sich dazu einfache Prozesse modellieren. Man kann zwischen einem einfachen Prozess oder einer Checkliste wählen, welche keine explizite Ausführungsreihenfolge festlegt. Aktivitäten können angelegt und bestimmten Nutzern zugeordnet werden. Danach kann der Prozess gestartet werden. Angehängte Dokumente und Kommentare können problemlos von Nutzern hinzugefügt werden, welche dann für alle Nutzer sichtbar sind. Auch das Weiterreichen eines Arbeitsschrittes an einen anderen Nutzer ist möglich. Daher ist diese Funktionalität besonders für Prozesse gedacht, welche bislang z.B. über E-Mail-Austausch stattgefunden haben. Um den Fortschritt eines Prozesses zu verfolgen, können Nutzer einen Prozess abonnieren. Dem Ersteller selbst wird der Fortschritt auf seinem Aktivitäten-Feed angezeigt.
Das Konzept von Social- und Cloudbasierten BPM-Systemwn findet immer mehr Anklang und wird auch in Systemen anderer Hersteller wie Appian [1], Intalio [3] und ARISalign [2] eingesetzt.

Blueworks Live ist ein verständliches und benutzerfreundlicher Ansatz, um Prozesse zu erstellen und zu teilen. Die Prozesserstellung und -darstellung basiert jedoch wieder auf einem graphenbasierten Modell und das System ist nur für einfache meist sequenzielle Prozesse gedacht. Um auch komplexere Prozesse einfach und verständlich darzustellen wird in dieser Arbeit eine formularbasierte Darstellung und Modellierung analysiert und diskutiert.

3 Konzept zur formularbasierten Darstellung von Prozessmodellen

In diesem Kapitel wird eine Abbildung von einer aktivitätenorientierten Prozessmodellierungs-Notation (vgl. Kapitel 2) auf eine formularbasierte Darstellung vorgestellt. Dazu wird in Kapitel 3.1 die Darstellung von Sequenzen, parallelen Abläufen, Verzweigungen und von Datenelementen mit Hilfe von Formularen betrachtet. Danach wird in Kapitel 4 in einer Fallstudie ein Prozessbeispiel vorgestellt, an welchem veranschaulicht wird, wie Formulare genutzt werden können, um das Verständnis für den Nutzer erhöhen zu können.

3.1 Abbildung der ADEPT-Kontrollflusselemente

Dieses Kapitel erläutert, wie in der formularbasierten Darstellung Sequenzen, AND- und XOR-Verzweigungen und Schleifen abgebildet werden.

Sequenz
Eine Sequenz ist eine Abfolge von Aktivitäten, welche hintereinander ausgeführt werden. In der aktivitätenorientierten Prozess-Notation wird eine Sequenz durch Knoten dargestellt, die von links nach rechts nacheinander ausgeführt werden. In der formularbasierten Darstellung wird dies durch übereinanderliegende Elemente abgebildet, wobei jedes Element einer Aktivität entspricht, die von oben nach unten ausgeführt wird. Die Elemente dienen als Container für den Datenfluss, der in Kapitel 3.2 beschrieben wird. Abbildung 3.1 zeigt die Abbildung der Sequenz der Aktivitäten A, B und C auf ein Formular.

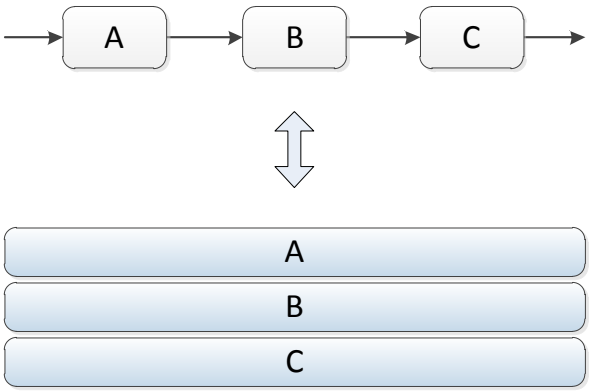

Abbildung 3.1: Sequenz in Formular-Darstellung

AND-Verzweigung

Parallele Abläufe teilen sich in der aktivitätenorientierten Prozess-Notation an einem Split-Knoten auf und werden an einem korrespondierenden Join-Knoten wieder zusammengeführt. In der formularbasierten Notation werden parallele Aktivitäten nach einem übergeordneten Element nebeneinander dargestellt. Wie in Abbildung 3.2 zu sehen, ist A das übergeordnete Element und repräsentiert den gesamten AND-Kontrollblock der aktivitätenorientierten Prozess-Notation, welcher zusätzlich durch die gestrichelte Umrandung kenntlich gemacht ist. Jede Spalte in dem gestrichelten Bereich entspricht dabei einem parallelen Verzweigungspfad. Das Element A entspricht in der formularbasierten Notation also dem kompletten Kontrollblock aus dem aktivitätenorientierten Prozessmodell und kann somit als Ganzes ein- und ausgeklappt werden. Der Pfeil rechts auf dem Element zeigt in der Abbildung an, dass das Element geöffnet ist. Der AND-Join wird erreicht, wenn alle parallelen Verzweigungspfade abgeschlossen sind. Es folgt ein Element in Breite des übergeordneten Elements (hier z.B. D). Die vorgestellte Blockstruktur wird also eingehalten.

Die Anzeige der parallelen Verzweigungspfade ist beschränkt und kann ab einer bestimmten Anzahl an Verzweigungspfaden nicht komplett angezeigt werden. In diesem Fall werden die Beschriftungen der ersten vier parallelen Elemente voll angezeigt und alle weiteren mit

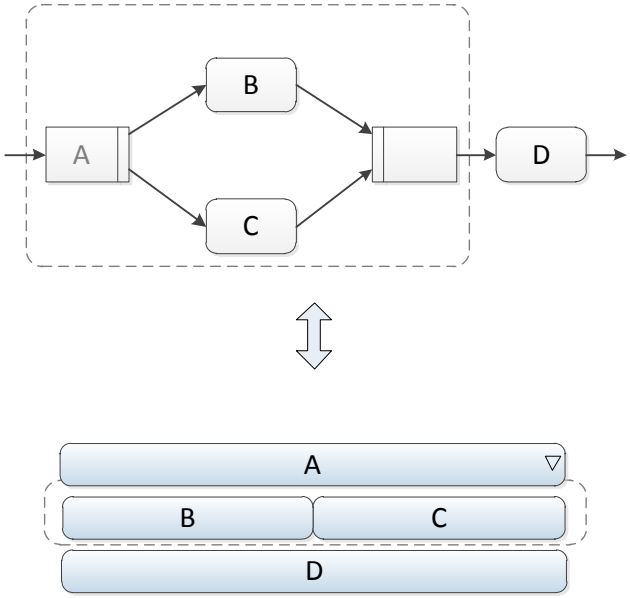

Abbildung 3.2: AND-Kontrollblock in Formular-Darstellung

„. . . “ beschriftet. In Abbildung 3.3 wird der fünfte Verzweigungspfad mit „. . . “ angezeigt und kann durch einen Klick vollständig angezeigt werden. Die Breite des Elementes A ist in einer Anwendung natürlich wesentlich breiter als im Beispiel, womit auch mehr Text darstellbar ist (siehe Fallstudie Kapitel 4).

Abbildung 3.3: AND-Kontrollblock: Ansicht mit Platzhalter

Eine andere Möglichkeit die Darstellung in einem solchen Fall zu optimieren, ist das Umschalten auf eine andere Ansicht. Dabei werden parallele Elemente unter ihrem Hauptelement untereinander dargestellt. Eine Einrückung der parallelen Elemente hilft dabei zu erkennen, wo die Verzweigung beendet wird (siehe Abbildung 3.4).

Abbildung 3.4: AND-Kontrollblock: Ansicht Pfade untereinander

XOR-Verzweigung

Ähnlich wie eine AND-Verzweigung wird eine XOR-Verzweigung an einem übergeordneten Element gesplittet. Wie im ADEPT-Basismodell enthält dieses Element ein Prädikat bzw. die Frage welche ausgewertet werden muss. In Abbildung 3.5 enthält das Element A das Prädikat und beschreibt gleichzeitig wieder den gesamten XOR-Kontrollock. Im übergeordneten Element sind die Pfade mit den möglichen Antworten bzw. Auswahlmöglichkeiten beschriftet. Zur Laufzeit wird die Auswahl entweder vom Nutzer oder vom System durch vorher geschriebene Datenelemente getroffen. Da hier nur ein Pfad zur Laufzeit ausgewählt wird, können für eine bessere Übersicht die restlichen Pfade ausgeblendet werden.

Um sich einen Pfad genauer anzusehen, kann dieser auch aufgeklappt werden. Dabei werden die restlichen Pfade in ihrer Breite auf ein Minimum reduziert und mit „. . . “ beschriftet. In dem ausgewählten Pfad können somit auch mehrere verschachtelte Subprozesse dargestellt werden.

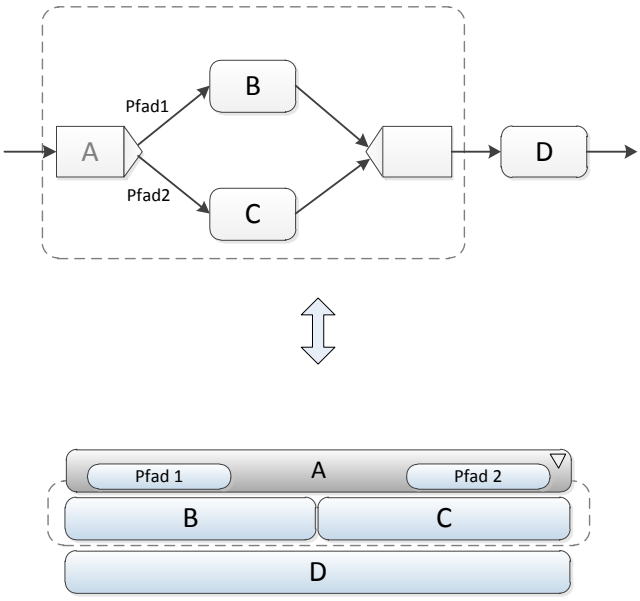

Abbildung 3.5: XOR-Kontrollblock in Formular-Darstellung

Schleife

Schleifen werden mit extra Elementen modelliert, die den Start und das Ende der Schleife festlegen. Um kenntlich zu machen, wo eine Schleife beginnt und wo sie endet werden

die Elemente entweder dementsprechend beschriftet oder mit einem Zeichen versehen. In Abbildung 3.6 ist eine Schleife in Formular-Darstellung abgebildet. Hier ist das Start- und Endelement textuell markiert.

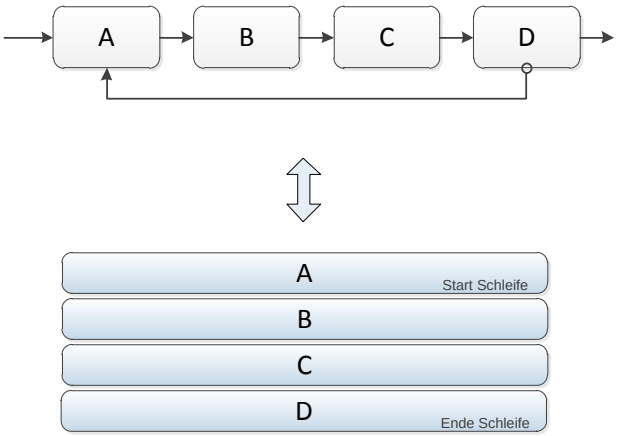

Abbildung 3.6: Schleifen-Kontrollblock in Formular-Darstellung

3.2 Abbildung der ADEPT-Datenflusselemente

Die intuitive formularbasierte Darstellung wird entsprechend auch für den Datenfluss verwendet. In der aktivitätenorientierten-Notation werden die Datenelemente als zusätzliche Blöcke dargestellt, welche den Knoten vom Erscheinungsbild sehr ähnlich sind und die Übersichtlichkeit beeinträchtigen. In einer Formular-Darstellung kann ein Datenelement je nach Typ gleich als Formularfeld also z.B. als Beschriftung eines Textfelds dargestellt werden. Die Datenelemente werden unter einem Knotenelement in Form einer aufklappbaren Detailinformation angezeigt. Nach der Modellierung des Datenflusses ist das Formular in der gleichen Form ausführbar. Das heißt das Prozessmodell dient hier nicht nur der Darstellung eines Prozesses, sondern kann in der gleichen Darstellung auch ausgeführt werden.

Zur Laufzeit braucht immer nur das Formularelement aufgeklappt sein, welches für eine Eingabe benötigt wird. Durch diese Darstellung ist sofort klar, welche Datenelemente von einem Element geschrieben werden. Datenelemente, die von diesem Element gelesen wer-

den, werden als Input aufgeführt. Um zu erkennen, wo ein geschriebenes Datenelement im weiteren Prozessablauf noch gelesen wird, kann dieses markiert werden. Daraufhin werden alle Knoten, in denen das Datenelement vorkommt, farblich hervorgehoben.

Da ein Datenelement immer einen Datentyp besitzen muss, kann die Darstellung der Eingabe entsprechend angepasst werden. Ein Datenelement vom Typ Boolean kann somit als einfache Auswahl dargestellt werden. Auch für die Eingabe von Integer Werten kann ein Textfeld entsprechende Restriktionen enthalten. Somit wird schon bei der Eingabe Typsicherheit gewährleistet.

In Abbildung 3.7 sind als Beispiel vier Datenelemente mit drei unterschiedlichen Datentypen abgebildet. In der formularbasierten Notation wird die Detailinformation erst beim aufklappen eines Elementes angezeigt und reduziert die Darstellung somit auf ein Minimum. Das Beispiel zeigt, wie Input und Output des Elementes B getrennt und übersichtlich aufgelistet werden. Der Datentyp boolean des Datenelements „Kunde“ wird dabei durch ein Auswahlkästchen dargestellt.

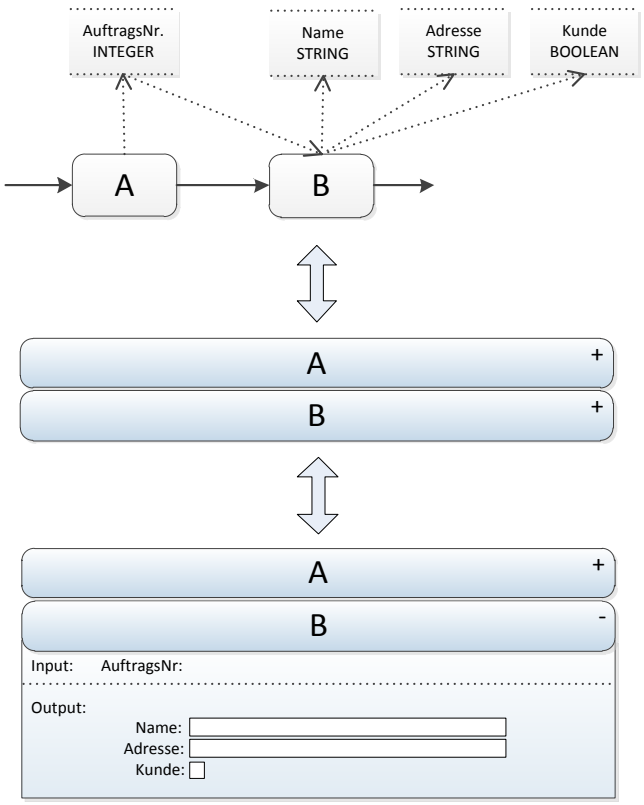

Abbildung 3.7: Datenelemente im Formular

Korrektheitskriterien

In diesem Abschnitt wird analysiert, inwiefern die Korrektheitskriterien aus Kapitel 2.1.3 des ADEPT-Basismodells auch auf dem formularbasierten Modell eingehalten werden.

Strukturierungsregel KF-1 fordert genau einen Start- und einen Endknoten. In der formularbasierten Darstellung sind diese implizit gegeben, da das Modell von oben nach unten abgearbeitet wird.

Laut KF-2 soll jeder Aktivitätenknoten eine eingehende und eine ausgehende Kante besitzen. Aktivitätenknoten sind im formuarbasierten Modell elementare Knoten und dürfen keine parallelen Elemente unter sich besitzen. Sie sind klappbar, um Daten- und Detailinformationen zu sehen. Nur übergeordnete Elemente, welche einem Split-Knoten entsprechen, dürfen mehrere Elemente unter sich besitzen.

Somit wird auch die Regel KF-3 eingehalten, welche die Blockstrukturierung fordert: Eine AND oder XOR-Verzweigung wird immer durch ein übergeordnetes Element eingeleitet. Mit diesem Element lassen sich die darunterliegenden Elemente einklappen. Im formularbasierten Modell gibt es keine expliziten Join-Knoten zu einer AND oder XOR-Verzweigung. Eine Verzweigung ist beendet sobald ein Element mit mindestens der gleichen Breite wie das übergeordnete auftritt.

Die Regel KF-4 wird vom formularbasierten Modell leicht eingehalten, denn im Modell können keine Zyklen auftreten. Es wird von oben nach unten bearbeitet und es gibt keine Kontrollkanten, welche auf ein höheres Element zeigen könnten. Schleifen werden mit Schleifen-Elementen modelliert und haben dadurch immer ein Start- und Endelement.

Auch die Regeln für einen korrekten Datenfluss können weitestgehend übernommen werden. So muss ein Datenelement natürlich erst geschrieben werden, bevor es gelesen wird. Und damit es bei der Ausführung nicht zu Inkonsistenzen kommt, sind parallele Schreibzugriffe auch hier unzulässig.
Typsicherheit wird schon beim Erstellen gewährleistet, da das Ausfüllen der Datenelemente mit den entsprechenden Restriktionen verbunden ist. Ein Boolean kann also nur mit true oder false ausgefüllt werden und ein Zahlenwert also ein Integer nur mit ganzen Zahlen.

3.3 Allgemeine Visulaisierungskonzepte

In diesem Kapitel werden verschiedene Visualisierungskonzepte vorgestellt, um das Verständnis und die Nutzung der Formular-Darstellung zu erleichtern. Durch die Konzepte wird ein Prozessmodell abstrahiert und somit übersichtlicher. Der Nutzer kann also selbst entscheiden, wie viel Information dargestellt werden soll.

Datenabstraktion
Für eine bessere Übersicht über das Prozessmodell können bestimmte Elemente zugeklappt werden. So können Zusatzinformationen, wie Input und Output eines Elements einfach ausgeblendet werden. Das Prozessmodell wird abstrahiert und der Nutzer kann sich einen besseren Überblick verschaffen. Elemente, welche Detailinformationen besitzen und zugeklappt sind, sind mit einem Pluszeichen markiert. Sie können also ausgeklappt werden. Ist die Breite des Elements ausreichend, um alle Informationen übersichtlich darzustellen, werden die Detailinformationen mit einer einfachen Animation eingeblendet. Liegt das Element in einer Verzweigung und ist sehr schmal muss es zuerst in der Breite angepasst werden. Durch einen Klick wird das Element also erst in der Breite verändert und danach ausgeklappt. Um dem Nutzer diese Bewegung visuell deutlich zu machen werden diese zwei Animationen verschmolzen.

Verzweigungsabstraktion
Auch AND- und XOR-Kontrollblöcke können als Ganzes auf- und zugeklappt werden. So können parallele und bedingte Abläufe als Subprozesse aufgefasst werden. Um im zugeklappten Zustand zu unterscheiden, ob ein Element einen Subprozess oder nur Dateninformationen öffnet, werden hier zwei unterschiedliche Klappsymbole verwendet. Für einen aufklappbaren Prozess wird ein Pfeil verwendet und für Detailinformationen ein Pluszeichen.

Da das Aufklappen von übergeordneten Elementen das Erscheinungsbild des formularbasierten Prozessmodells eventuell stark verändert, kann es sein, dass der Nutzer nicht sofort erkennt, welche Elemente neu hinzugekommen sind. Um dieses Problem zu lösen werden

auch hier Animationen verwendet um den Prozess zu öffnen. Zuerst wird das übergeordnete Element verbreitert und danach die Elemente des zugeklappten Prozesses dargestellt. Dadurch erkennt der Nutzer sofort, welche Elemente geöffnet werden. (siehe Fallstudie Kapitel 4).

Bei einem hohen Verzweigungsgrad des Prozessmodells wird eine Darstellungseinschränkung vorgenommen. Das heißt es kann nur eine bestimmte Anzahl an übergeordneten Elementen angezeigt werden. Um weitere ausgeblendete Elemente in einem Verzeigungspfad einzusehen muss zuerst der entsprechende Pfad ausgewählt werden. Dadurch wird die Breite der Anzeige erhöht und es können mehr Elemente in der darunterliegenden Ebene dargestellt werden. Wie bereits angesprochen wäre auch das Umschalten der Ansicht auf eine Liste der parallelen Pfade denkbar. In Abbildung 4.5 ist beispielsweise der XOR-Kontrollblock Turnier eingeklappt. Und auch das Öffnen eines AND- oder XOR-Kontrollblocks in einem neuen Fenster als eigener Prozess wäre eine mögliche Lösung.

In diesem Kapitel wurden die Grundelemente des formularbasierten Ansatzes sowie mögliche Interaktionen vorgestellt. Außerdem wurden entsprechende Regeln angesprochen, um Korrektheit und Konsistenz sicherzustellen. Im folgenden Kapitel wird ein konkretes Beispiel umgesetzt, um die Anwendbarkeit der Konzepte zu zeigen.

4 Fallstudie

Nachdem die Grundlagen zur formularbasierten Darstellung geklärt sind wird hier ein konkretes Beispiel eines Prozessmodells umgesetzt. Zunächst wird der Ausgangsprozess analysiert und dann als Formular umgesetzt. Dabei wird insbesondere die verbesserte Übersicht deutlich und es wird die Funktionsweise der interaktiven Elemente erläutert.

4.1 Beispielprozess Schachturnier

Die Grundlage der Fallstudie stellt der vereinfachte Ablauf eines Schachturniers mit Vorbereitungen und Siegerehrung [7]. Das Prozessmodell wurde zunächst in der ADEPT-Notation umgesetzt (siehe Abbildung 4.1).

Zur Veranschaulichung eines einfachen Datenflusses wurden drei Datenelemente zugefügt: Datum vom Typ date, Adresse und Räume vom Typ string. Weitere zwei Datenelemente sind im ADEPT-Basismodell für den XOR-Kontrollblock und für die Schleife nötig.

Der Prozess startet mit zwei Aktivitäten: In der ersten Aktivität „Räumlichkeiten beantragen“ werden die Datenelemente Räume und Adresse vom Typ string geschrieben. Im nächsten Knoten „Eintragungen in Terminkalender“ wird das Datenelement Datum vom Typ date geschrieben. Danach folgt eine AND-Verzweigung. Hier werden parallel die Vorbereitungen bis zum Schachtag ausgeführt, wie das Einkaufen der Preise, Urkunden, Pokale und Lebensmittel. Nachdem die Vorbereitungen abgeschlossen sind, findet am Tag des Turniers das Treffen der Mitarbeiter statt, hierfür wird das Datenelement Datum gelesen. Vor dem Turnier selbst finden noch verschiedene Aufbauarbeiten statt, die parallel ablaufen. Es werden noch die Bretter aufgebaut und die Computer für die Anmeldung der Teilnehmer installiert. Hierfür werden die anfangs geschriebenen Datenelemente Adresse und Räume gelesen.

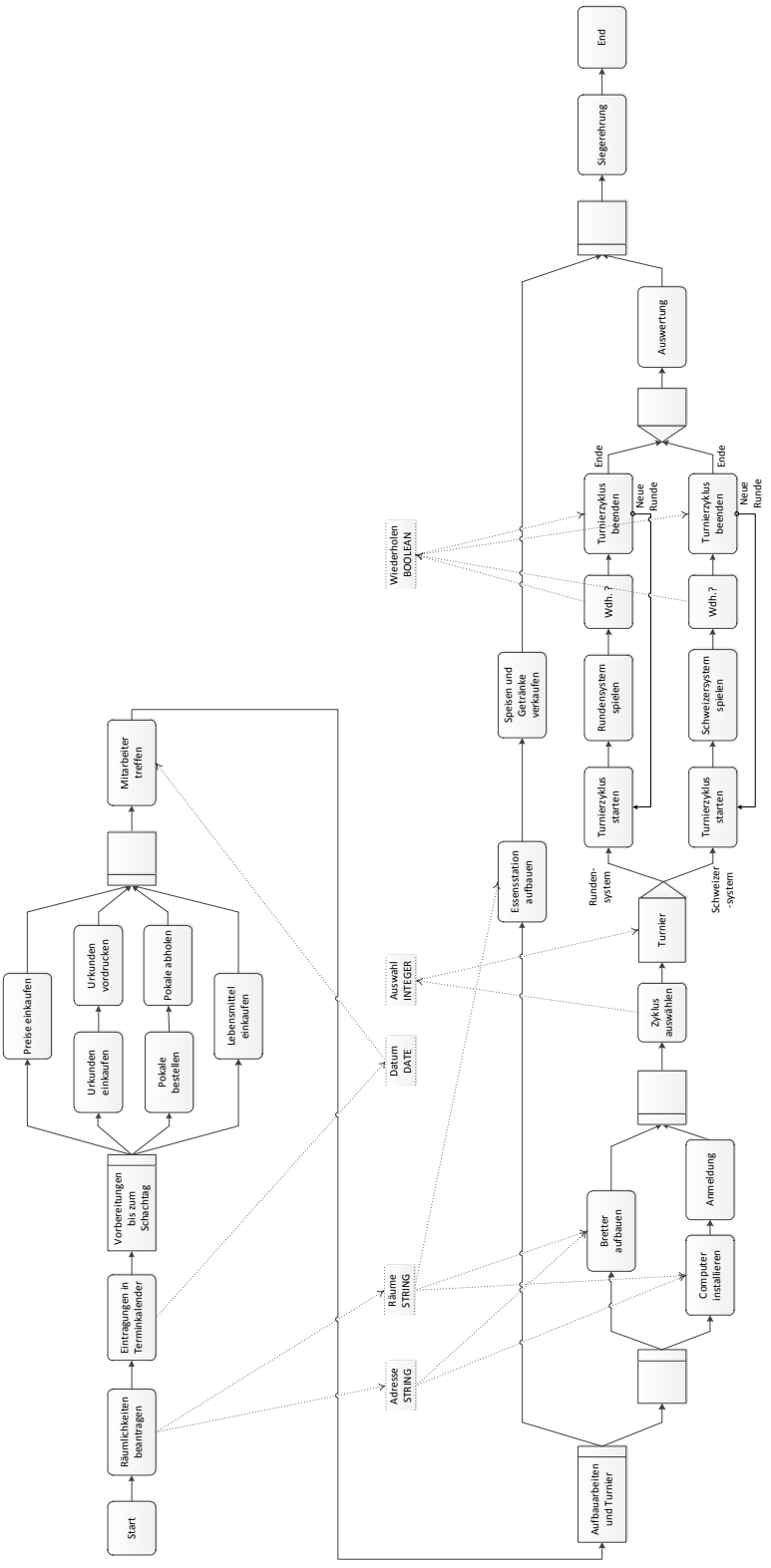

Abbildung 4.1: Schachprozess in Notation des ADEPT-Basismodells

Für die Auswahl der Turnierart wurde dem Prozessmodell noch ein Knoten „Zyklus auswählen“ hinzugefügt, durch den die bedingte Wahl zwischen Runden- oder Schweizersystem mit einem Formular getroffen wird. Es wird also entschieden, ob das Rundensystem oder das Schweizersystem gespielt wird. Da ein Turnier aus mehreren Runden besteht, wird es als Schleife modelliert. Die Anzahl der Runden hängt vom gewählten System ab. Ob eine weitere Runde stattfindet, wird wieder durch ein zusätzliches Element festgelegt, in dem ein Boolean-Wert gesetzt wird. Parallel zum Turnier findet der Speise- und Getränkeverkauf statt. Ist das Turnier beendet, wird es ausgewertet und schlussendlich findet noch die Siegerehrung statt.

4.2 Formularbasierte Umsetzung

Um zu veranschaulichen, wie ein konkreter Prozess in der formularbasierten Darstellung aussieht, wurde der vorgestellte Schachprozess mit den in Kapitel 3.1 erläuterten Elementen prototypisch umgesetzt. Für die Umsetzung wurde HTML mit JQuery [4] verwendet, da es Methoden bereitstellt, die für die Darstellung von interaktiven klappbaren Elementen hilfreich sind. Farben und Formen sind minimal gehalten, da das Augenmerk auf der Funktionsweise liegt.

In der Grundansicht des Prozesses sollten möglichst alle Elemente ohne ihre Datenelemente sichtbar sein. Das heißt, alle klappbaren übergeordneten Elemente sind geöffnet(siehe Abbildung 4.2). Sollten die Elemente nicht vollständig darstellbar sein, werden je nach Platz bis zu fünf parallele Elemente angezeigt und die restlichen Elemente auf eine minimale Breite verkleinert. Um den Prozess weiter einsehen zu können, muss in diesem Fall erst ein Verzweigungspfad ausgewählt werden. Wie in der Abbildung zu sehen, ist der Schachprozess aber als Ganzes problemlos darstellbar. Außerdem wird deutlich, welchen Vorteil die Darstellung hinsichtlich der Übersicht birgt. Alle Aktivitäten sind dargestellt und passen problemlos auf eine Seite. Kantenübergänge nehmen in der neuen Darstellung keinen Platz weg.

Die formularbasierte Darstellung wird von oben nach unten gelesen, und auch so abgearbeitet. Die Struktur des Prozesses ist gut erkennbar: Die ersten drei breiten Elemente sind Sequenzelemente, daraufhin folgt der AND-Kontrollblock und nach einem weiteren Se-

quenzelement der Prozess der Aufbauarbeiten und des Turniers. Hier folgt nach zwei parallelen Pfaden eine XOR-Verzweigung. Diese unterscheidet sich deutlich von einer AND-Verzweigung und die auswählbaren Pfade (Rundensystem oder Schweizersystem) sind hervorgehoben.

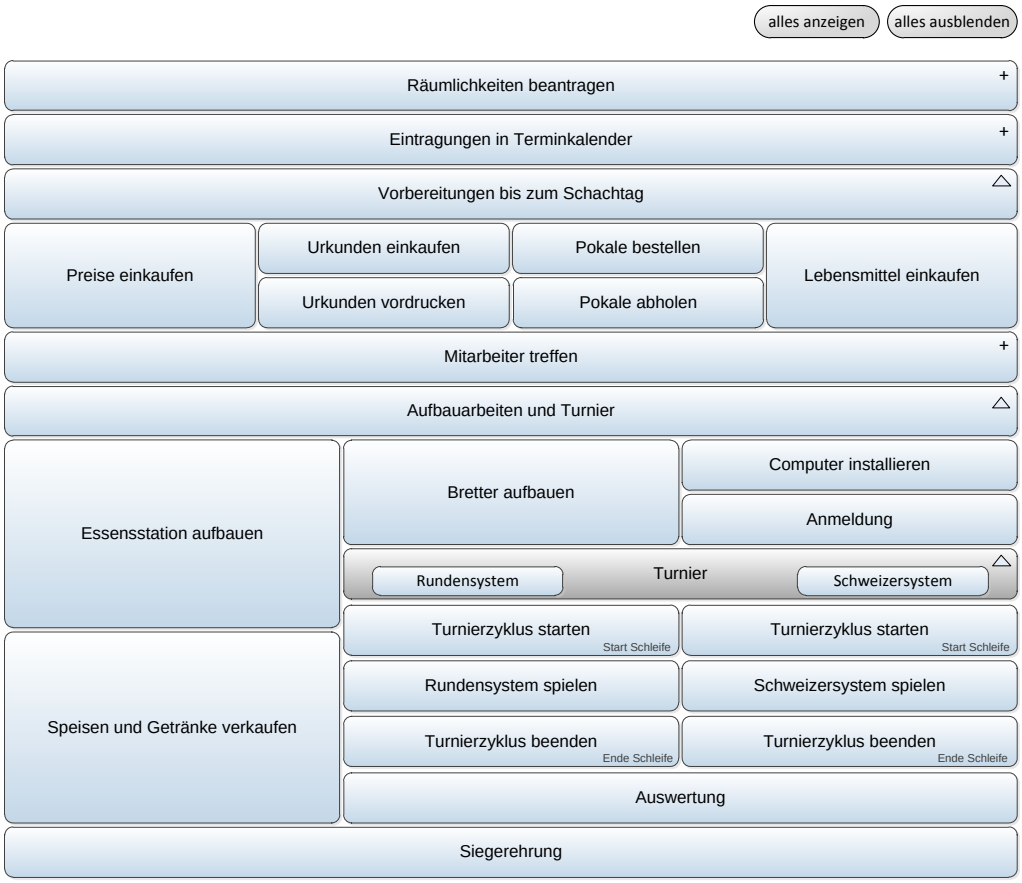

Abbildung 4.2: Fallstudie: Normale Ansicht

In der Umsetzung wurden bereits zwei Buttons integriert, mit denen sich alle Elemente zuklappen bzw. aufklappen lassen. Abbildung 4.3 zeigt das Modell, nachdem alle Elemente zugeklappt sind. Hier erkennt man an den unterschiedlichen Zeichen zum Öffnen der Elemente, ob es sich um ein einfaches Element oder ein übergeordnetes Element handelt. In dem Beispiel steht das „+“ für aufklappbare Detailinformationen und das „v“ für einen aufklappbaren Prozess, also AND- oder XOR-Verzweigungspfade.

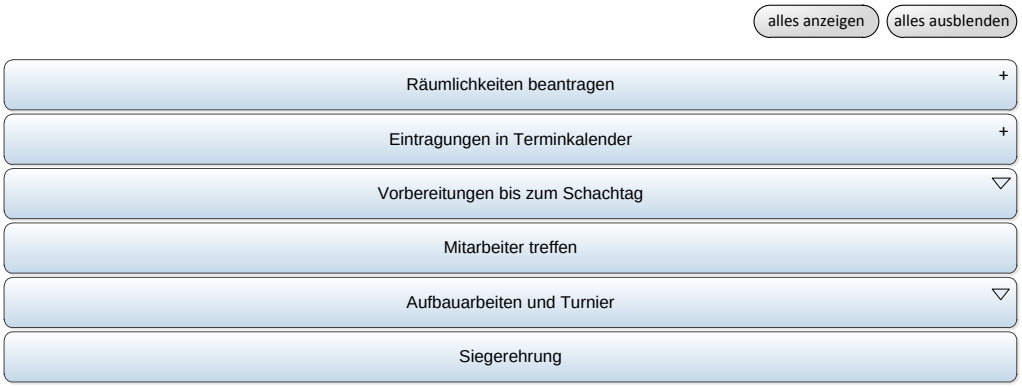

Abbildung 4.3: Fallstudie: Alle Elemente eingeklappt

Will der Nutzer ein bestimmtes Element näher begutachten kann auf dieses geklickt werden und es wird mit einer Animation in der Breite vergrößert. Auch die klappbare Detailinformation mit Datenelementen kann so besser eingesehen werden. In Abbildung 4.4 wurde beispielsweise Zyklus1 der bedingten Verzweigung ausgewählt. Alle anderen Elemente wurden in ihrer Breite reduziert und die Beschriftung durch „. . . “ ersetzt. Auch die Auswahl für Zyklus2 wurde ausgegraut, um Zyklus1 hervorzuheben. Durch ein erneutes Klicken auf Zyklus1 stellt sich die Ansicht wieder auf die normale Breite ein.

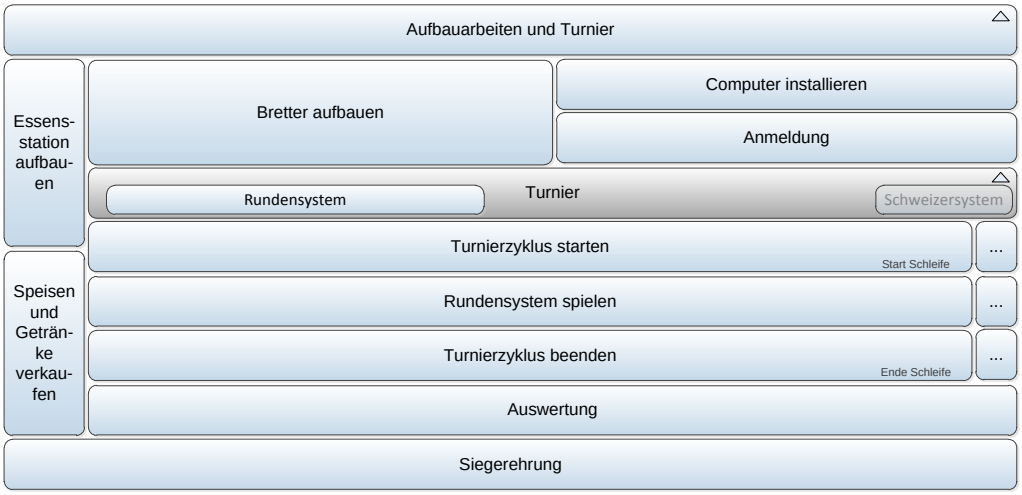

Abbildung 4.4: Fallstudie: Ausschnitt Klick auf Rundensystem

Zur Veranschauung wurden einige Datenelemente hinzugefügt. Die Detailansicht, wie in Abbildung 4.5 zu sehen, ist hier einfach gehalten, das heißt es wurde hier auf Annotationen und zusätzliche Details vorerst verzichtet. Das Detailfenster lässt sich bei allen mit „+“ markierten Elementen aufklappen. Hier können Datenelemente eingetragen werden. Je nach Datentyp wird das Element anders dargestellt, um eine korrekte Eingabe zu gewährleisten.
Um zu sehen, wo ein Datenelement wieder gelesen wird, kann der Nutzer das Datenelement anklicken. Daraufhin werden alle Elemente, in denen das Datenelement vorkommt farblich markiert (siehe Abbildung 4.5). Sollte sich ein solches Element in einem nicht aufgeklappten Subprozess befinden, wird das übergeordnete Element markiert.

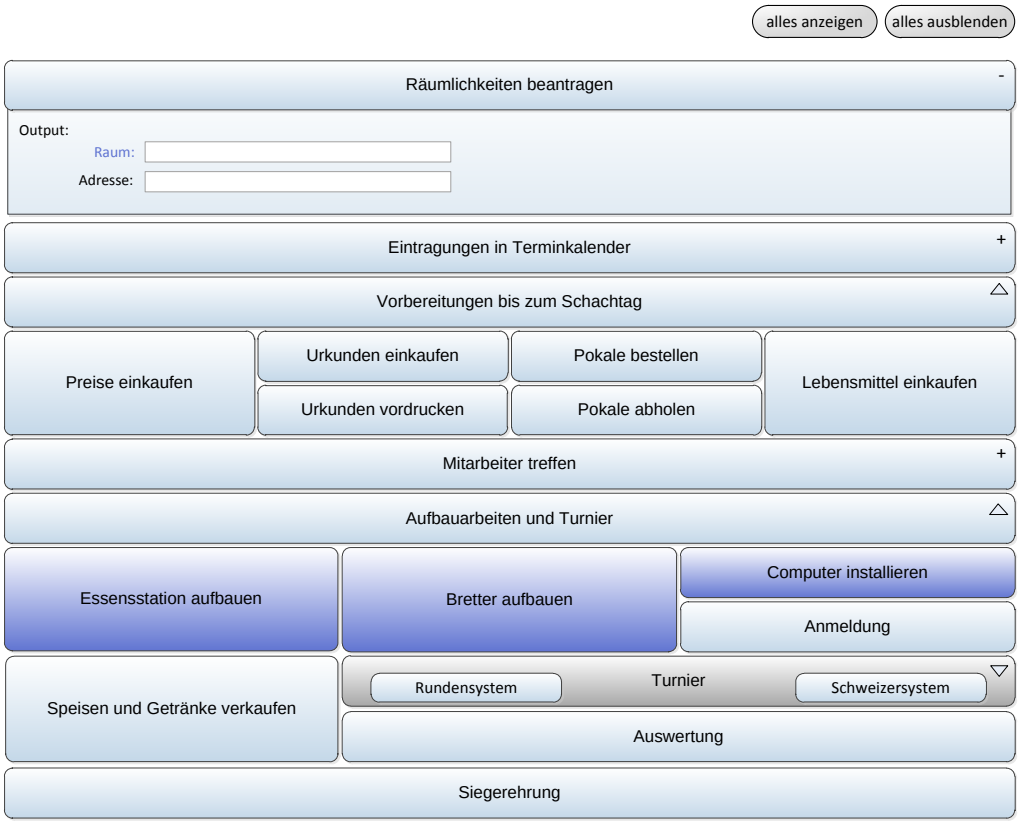

Abbildung 4.5: Fallstudie: Markierung eines Datenelementes

In dieser Fallstudie wurde nur die Grundansicht mit einfachen Farben und Formen umgesetzt. In einer vollständigen Umsetzung in einem System muss auch ein ausführliches

Menü zur Verfügung stehen, in welchem die Ansichten gewechselt werden können. Dabei soll auf eine Modellierungs- und eine Laufzeitansicht umschaltbar sein. Die Laufzeitansicht kann zusätzlich reduziert werden, indem Elemente, welche nicht vom Nutzer sondern vom System selbst ausgeführt werden ausgeblendet werden.

4.3 Technische Details zur prototypischen Umsetzung

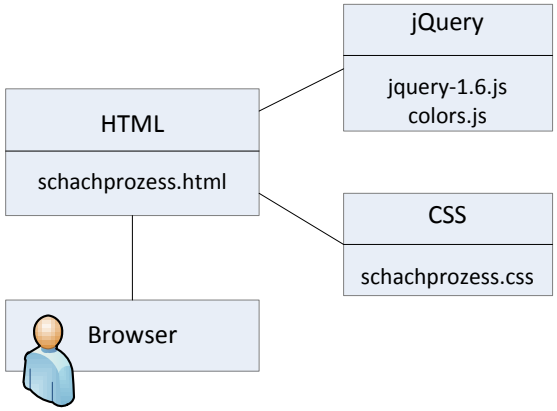

Abbildung 4.6: Schematische Darstellung der Komponenten

Die formularbasierte Umsetzung wurde als interaktive Webseite umgesetzt. Wie in Abbildung 4.6 dargestellt, wurde außer HTML und CSS zusätzlich jQuery verwendet. JQuery bietet eine umfangreiche JavaScript-Klassenbibliothek, mit der Animationen gut umsetzbar sind. Im Folgenden werden einige Code-Ausschnitte näher erläutert um zu verdeutlichen, wie jQuery hier eingesetzt wurde.

JQuery bietet eine einfache Methode, um Elemente ein- bzw. auszuklappen: `slideToggle()`. Im Grunde reicht diese Methode aus, um ein Element zu animieren. Um dem Nutzer aber zu zeigen, ob ein Element ausklappbar ist, wurde zu jedem Element ein entsprechendes Zeichen gewählt. Ein Plus- oder Minus-Zeichen kennzeichnet ausklappbare Detailinformationen und ein Pfeil, hier dargestellt durch v, einen ausklappbaren Prozess. Um die Zeichen entsprechend anzupassen und zu erkennen, in welchem Zustand sich ein Element befin-

det, wurde die Hilfsmethode `open()` eingeführt (siehe Listing 4.1). Diese Methode kann für alle Elemente verwendet werden, die in voller Breite dargestellt sind.

```
jQuery.fn.open = function(){
        return this.each(function() {
                var id = '#' + $(this).attr('id');
                $(id).next('div').slideToggle("slow" , function(){
                        if($(id+' .sign').html() == "-")
                                $(id+' .sign').html("+");
                        else if($(id+' .sign').html() == "+")
                                $(id+' .sign').html("-");
                        else if($(id+' .sign').html() == "^")
                                $(id+' .sign').html("v");
                        else
                                $(id+' .sign').html("^");
                });
        });
};
```

Listing 4.1: Funktion open

Listing 4.2 zeigt, wie der Aufruf der Methode aussehen kann.

```
//Beispiel eines Aufrufs der Methode open
$('#el1').click(function(){$(this).open();});
```

Listing 4.2: Aufruf der Funktion open

Das nächste `div` Element, welches mit Hilfe von CSS ausgeblendet ist, wird anhand eines Klicks auf das Element mit der id `id=„el1“` geöffnet. Um Elemente in einem Verzweigungspfad zu öffnen muss aber noch auf die anderen Verzweigungspfade geachtet werden. Diese müssen zusätzlich vergrößert bzw. verkleinert werden. Da in dem Beispielprozess nur der XOR-Verzweigungsblock animiert wurde, ist der entsprechende Code hier nur auf dieses Element zugeschnitten. Listing 4.3 zeigt die Hilfsmethode `turnierausklappen()` mit der auch der linke Verzweigungspfad animiert wird, indem der gesamte Subprozess entsprechend vergrößert bzw. verkleinert wird.

```
1 //Hilfsmethode aus/einklappen turnier
2 jQuery.fn.turnierausklappen = function(){
3         $('#xor').open();
4         //linke seite mit vergroessern/verkleinern
5         if($('#xor .sign').html() == "^")
6                 $('#elem3').animate({height:180},"slow");
7         else
8                 $('#elem3').animate({height:315},"slow");
9 };
```

Listing 4.3: Funktion turnierausklappen

Um einen der XOR-Verzweigungspfade zu vergrößern, muss die Größe der Elemente im anderen Pfad in der Breite reduziert werden. Außerdem muss der Text zwischengespeichert werden, um ihn nach einem erneuten Klick wieder einzufügen.

Die prototypische Umsetzung von einer aktivitätenorientierten Darstellung zu einem interaktiven Formular zeigt, welche Vorteile eine formularbasierte Prozessmodellierung besitzt. Der Prozess ist auf einer Seite darstellbar, wodurch der Nutzer eine bessere Übersicht über das Prozessmodell erlangt. Des Weiteren sind Elemente interaktiv und passen sich der gewünschten Abstraktionsebene an. Das Prozessmodell dient auch nicht nur der Darstellung des Prozesses, sondern auch der Ausführung. Es ist also keine gesonderte Ansicht für die Laufzeit nötig.

5 Modellierung und Änderungen von Formularen

Die reine Darstellung eines Prozesses in einer anderen Form ist natürlich ohne den Modellierungs- und Laufzeitaspekt nicht sehr nützlich. Deshalb wird in diesem Kapitel erläutert, welche Möglichkeiten es gibt, einen Prozess als Formular zu modellieren bzw. zu ändern. Dabei soll das Erstellen eines Prozesses in formulabasierter Darstellung einfach und intuitiv sein. Im Folgenden werden zwei Modellierungskonzepte vorgestellt. Zum einen eine Drag and Drop-Variante (siehe Kapitel 5.1), in der vorgegebene Elemente durch ziehen und richtiges Platzieren angeordnet werden. Zum anderen ein Konzept, in dem mit Menüs gearbeitet wird, welche über Rechtsklick aufgerufen werden können.

5.1 Konzept: Drag and Drop

Zunächst wird das Konzept Drag and Drop analysiert. Dieses Konzept kommt bei Modellierungstools öfters zum Einsatz und kann sehr intuitiv genutzt werden. Elemente können direkt am Rand der Seite mit entsprechendem Untertitel angezeigt werden und von dort auf die Arbeitsfläche gezogen werden. Der Nutzer hat es somit einfacher ein Objekt anhand seiner Form zu identifizieren, als in einer Liste zu finden. Im Folgenden wird der allgemeine Aufbau beschrieben, wie Elemente eingefügt und gelöscht werden können und was bei der Modellierung von Verzweigungen zu beachten ist.

5.1.1 Elemente einfügen und löschen

Das Erstellen bzw. das Ändern eines Formular-Prozesses erfordert eine eigene Ansicht. In dieser Ansicht gibt es am linken Rand ein Auswahlfenster für die einzelnen Elemente. Die Arbeitsfläche nimmt den Rest der Ansicht ein. Da es sich um nicht viele unterschiedliche

Modellierungselemente handelt, ist keine zusätzliche Gruppierung notwendig. Die Elemente in der Auswahlleiste tragen sprechende Titel und für eine genauere Beschreibung wird eine Popup-Hilfe eingeblendet. In der Auswahlleiste kann der Nutzer unter folgenden Elementen wählen: Sequenzenelement, übergeordnetes Element für bedingte und parallele Verzweigung, Schleifen Start- und Endknoten und Datenelement. Ein Beispiel für die Editieransicht dieses Konzeptes ist in Abbildung 5.1 zu sehen.

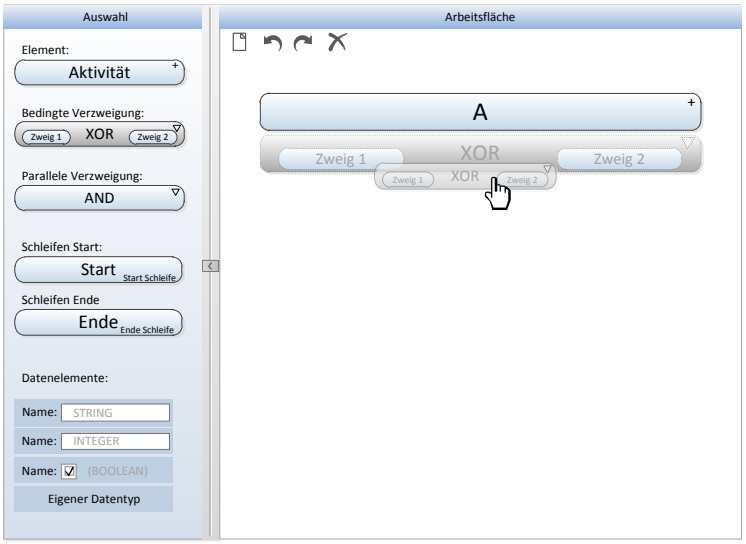

Abbildung 5.1: Beispiel für die Editieransicht

Um ein Element einzufügen, wird dieses aus der Auswahlleiste an die entsprechende Stelle im Arbeitsbereich gezogen und beschriftet. Dabei muss der korrekte Kontrollfluss eingehalten werden. Um dies bereits bei der Modellierung zu ermöglichen, werden Elemente magnetisch an die entsprechende Stelle gezogen. Wenn der Nutzer also drei Sequenzen-Elemente einfügt, können diese an einer beliebigen Stelle losgelassen werden. Sie werden dann automatisch untereinander in der gleichen Breite angeordnet. Um ein Element zu löschen, wird das entsprechende Element markiert und entweder über die Tastatur oder über einen Button gelöscht. Ein Element kann jedoch nur gelöscht werden, wenn dadurch die Korrektheit des Kontrollflusses nicht beeinflusst wird.

Besonders bei der bedingter und paralleler Verzweigung muss darauf geachtet werden,

dass zuerst ein übergeordnetes Element eingefügt wird. Erst unter diesem ist das Einfügen von Elementen nebeneinander möglich. Um die Möglichkeiten aufzuzeigen, wo ein Element eingefügt werden kann, wird die entsprechende Stelle beim Darüberfahren mit dem Element farblich markiert. So kann eine Verzweigung durch das Einfügen eines breiteren Elementes auch wieder beendet werden. Die Breite von nebeneinanderliegenden Elementen wird automatisch angepasst. Bei einer bedingten Verzweigung müssen die Verzweigungspfade beschriftet werden und ein eindeutiges Prädikat hinterlegt werden. Die Entscheidung, des zu wählenden Zweigs, kann so entweder durch den Nutzer manuell oder durch zuvor festgelegte Datenelemente automatisch zur Laufzeit getroffen werden.

Soll die Auswahl der Bearbeiterzuordnung in der Editieransicht stattfinden, werden hier zusätzliche Buttons bzw. ein Dialogfenster benötigt, um Bearbeiter einem bestimmten Element zuzuordnen.

5.1.2 Datenelemente

Um einem Element ein Datenelement hinzuzufügen, kann dieses aus der Auswahlleiste auf das entsprechende Element gezogen werden. Dabei öffnet sich ein Dialogfenster, in welchem die Details zu Datentyp, Lese- und Schreibvorgängen festgelegt werden können. Ein Beispiel für diesen Dialog ist in Abbildung 5.2 zu sehen. Dem Datenelement ist automatisch ein Schreibzugriff zugeordnet, das aktuelle Element wird also schon unter „Output von“ aufgelistet. Änderungen sind jederzeit möglich. Wichtig ist das Festlegen des Datentyps, der je nachdem welches Datenelement man aus der Auswahl gewählt hat, bereits eingetragen ist. Weitere Lese- und Schreibzugriffe können durch einen Klick auf das entsprechende Plus-Zeichen hinzugefügt werden. Aus einer Dropdownliste können dann vorhandene Elemente ausgewählt werden. Mit Hilfe des Datentyps wird das Erscheinungsbild unter einem Element entsprechend angepasst, so dass die Typkonsistenz gesichert wird.

Das Dialogfeld zum Editieren eines Datenelements muss jederzeit wieder aufrufbar sein, zum Bespiel durch einen Klick auf ein Datenelement in der Editieransicht. Ein weiterer Dialog, der es auch erlaubt vorhandene Datenelemente einem gewählten Element zuzuordnen, kann in einer Registerkarte angehangen werden. Für eine bessere Übersicht über die vorhandenen Datenelemente können diese seperat in einer Liste mit den jeweiligen Zu-

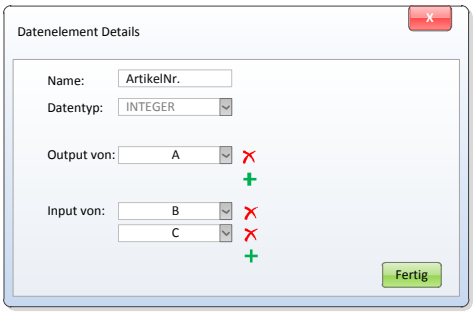

Abbildung 5.2: Beispiel für den Dialog eines Datenelements

griffen angezeigt werden. Bei der Bearbeitung eines Datenelementes ist auch eine farbliche Markierung der bereits als Lese- oder Schreibzugriff festgelegten Elemente hilfreich.

5.2 Konzept: Menügeführt

Eine andere Möglichkeit ist das Modellieren und der Ändern eines Prozess-Formulars über Menü-Einträge. Das Bearbeiten von Elementen durch Rechtsklick ist kein unbekanntes Konzept und kann ohne einen zusätzlichen Konfigurationsmodus genutzt werden.

Durch einen einfachen Rechtsklick auf die Arbeitsfläche erscheint ein Menü, in dem die wesentlichen Punkte, die auch durch Drag and Drop möglich sind in einer Liste aufgeführt werden. Aufgrund der vielen Auswahlmöglichkeiten werden diese zu Menüpunkten gruppiert. Das Einfügen eines Elementes funktioniert somit durch einen einfachen Rechtsklick auf die Arbeitsfläche, wobei das Menü geöffnet wird und ausgewählt werden kann, welches Element eingefügt werden soll. Um die Korrektheit und Konsistenz zu gewährleisten, werden in dem Menü nur zulässige Vorgänge angezeigt. Unzulässige Aktionen werden ausgegraut, damit ersichtlich ist, dass es diese Funktionen zwar gibt, sie aber momentan nicht genutzt werden können.

Das Einfügen von parallelen Elementen funktioniert auch hier nur, wenn bereits ein XOR- oder AND-Element als Verzweigungselement hinzugefügt wurde. Wenn nun auf dieses Element geklickt wird kann der Verzweigungsgrad angegeben werden und es können Elemen-

te darunter nebeneinander platziert werden. Um ein Element zu löschen, klickt man dieses an und wählt den Menüpunkt löschen. Sollte das Löschen eines Elementes die Korrektheit des Kontrollflusses beeinflussen so kann der Löschen Menüpunkt entweder ausgegraut werden oder es wird eine Fehlermeldung angezeigt. Wird ein Verzweigungselement gelöscht, werden alle zugehörigen darunterliegenden Elemente ebenfalls gelöscht.

Für Laien ist dieses Konzept nicht so intuitiv wie Drag and Drop, da Elemente nur in einer Liste angezeigt werden. Es eignet sich eher für Nutzer, die sich bereits mit den Elementen auskennen und wissen, welche Aktionen ausgeführt werden können. So soll es auch eine Abbildung der Aktionen auf Tastaturkürzel geben, um für Experten das Modellieren und Ändern des Formulars zu beschleunigen.

6 Laufzeitverhalten und damit einhergehende Konzepte

In diesem Kapitel wird beschrieben, wie sich formularbasierte Prozessmodelle zur Laufzeit verhalten und welche Konzepte diesbezüglich sinnvoll wären. Zunächst werden die verschiedenen Laufzeit-Zustände und mögliche Zustandsänderungen analysiert. Für die Darstellung der Zustände und Zustandsübergänge einer Aktivitäteninstanz wird ein Zustandsdiagramm (Statechart) verwendet [22].

6.1 Laufzeitverhalten

Bevor eine Instanz des Modells erstellt werden kann, muss zunächst sichergestellt sein, dass der Datenfluss korrekt ist. Eine Fehlermeldung bzw. ein Warnzeichen an der entsprechenden Stelle weist den Nutzer auf noch anzupassende Elemente hin. So muss zum Beispiel ein obligates Datenelement bevor es gelesen wird, mindestens einmal von einem darüberliegenden Element geschrieben worden sein (siehe Struktuierungsregel DF-1 aus Kapitel 2.1.3).

Ist die Korrektheit sichergestellt, kann das Formular ausgeführt werden. Die Reihenfolge ist dabei durch die Darstellung selbst gegeben. Das heißt, Aktivitäten werden von oben nach unten im Formular abgearbeitet. Im Folgenden werden die unterschiedlichen Zustände und Zustandsänderungen von Aktivitäteninstanzen beschrieben.

6.1.1 Zustände

In Abbildung 6.1 ist ein Zustandsdiagramm einer Aktivitäteninstanz abgebildet. Hier werden die verschiedenen Zustände, die eine Aktivität während der Ausführung besitzen kann, dargestellt. Zunächst befindet sich die Aktivitäteninstanz im `init` Zustand, das heißt sie wurde erstellt. Die Aktivität ist in diesem Zustand noch nicht bearbeitbar. Erst wenn alle Vorbedingungen erfüllt sind wechselt dieser Zustand in den `ready` Zustand. Die Aktivitäteninstanz ist aktiviert und kann nun gestartet werden. Sie kann im `read` Zustand auch deaktiviert werden und befindet sich dann im Zustand `disabled`.

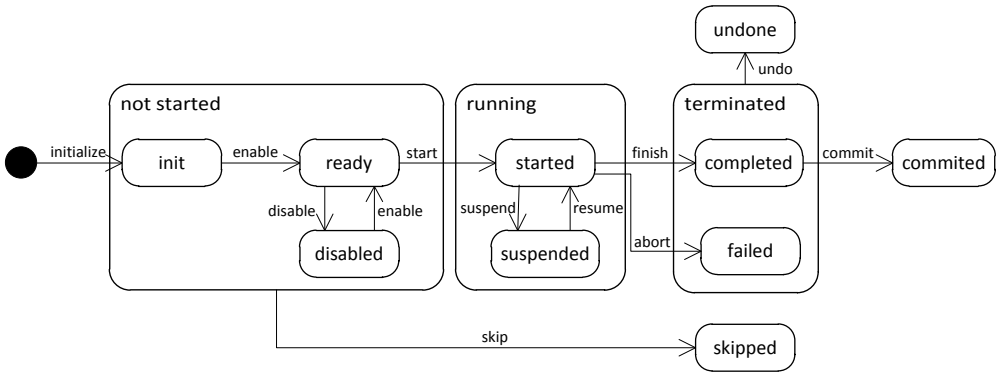

Abbildung 6.1: Statechart für Aktivitäteninstanzen

Die Aktivität wird jetzt entweder vom Nutzer oder falls eine automatische Aktivität vorliegt vom System gestartet. Die Zustände `init, ready` und `disabled` können auch unter dem Oberbegriff `not started` zusammengefasst werden. Wird die Aktivität nun ausgeführt, befindet sie sich im Zustand `started`. Kann die Aufgabe nicht in einem Schritt abgearbeitet werden, ist es möglich die Aufgabe zu unterbrechen. Dabei wechselt der Zustand zu `suspended`. Die Aufgabe kann jederzeit wieder fortgesetzt werden und befindet sich dann wieder im Zustand `started`. Diese zwei Zustände werden unter dem Begriff `running` zusammengefasst.

Aus dem Zustand `started` kann die Aktivität entweder beendet oder abgebrochen werden. Wird sie beendet, befindet sie sich im Zustand `completed`, das heißt allen Ausgabeparametern wurde ein Wert zugewiesen. Die Bearbeitung war also erfolgreich. Bei einem Abbruch wechselt die Aktivität hingegen in den Zustand `failed`. Die Zustände `completed` und `failed` werden im weiteren auch dem Zustand `terminated` zugeordnet.

Befindet sich die Aktivität im Zustand `completed`, so kann sie durch Abschicken der Daten in den Zustand `commited` wechseln. Im Zustand `terminated` kann die Aktivität auch zurückgesetzt werden und befindet sich dann im Zustand `undone`.

Ein weiterer Zustand ist der Status `skipped`. Dieser Zustand wird erreicht, wenn sich eine Aktivität in einem Zweig einer bedingten Verzweiung befindet, der nicht mehr zur Ausführung kommt. Die Bedingung für die Ausführung der Aktivität hat sich also geändert und der Zustand der Aktivität wechselt dann von `not started` auf `skipped`.

6.1.2 Darstellung eines Formulars zur Laufzeit

Um zu verdeutlichen in welchem Zustand sich eine Aktivität zur Laufzeit befindet, muss eine einheitliche Zustandsmarkierung für die formularbasierten Prozessmodelle festgelegt werden. Wird eine neue Prozess-Instanz erzeugt, befinden sich alle Aktivitäten im `init` Zustand. Sie sind also noch nicht aktiviert und werden in ihrer normalen Darstellung angezeigt. Nachdem die Instanz gestartet wurde wird der Zustand der obersten Aktivität in den Zustand `ready` geändert. Um darzustellen, dass diese Aktivität bereit ist gestartet zu werden, wird sie farblich mit einer grünen Umrandung oder mit einem Symbol markiert. Wird die Aktivität gestartet wechselt ihr Zustand zu `running`. Hier wird, wie in Abbildung 6.2 dargestellt, das gesamte Feld eingefärbt. Ist die Aktivität beendet wird sie ausgegraut, um den Blick auf die folgenden Aktivitäten zu lenken. Aktivitäten in einer Schleife werden erst ausgegraut, wenn diese beendet wurde. Eine fehlgeschlagene Aktivität im Zustand `failed` wird rot oder mit einem Kreuz markiert.

Bereits bearbeitete Aktivitäten können in einer Übersicht zusammengefasst werden (siehe Kapitel 6.2.3). Bei einer bedingten Verzweigung können nicht ausgewählte Zweige ausgegraut oder gänzlich ausgeblendet werden.

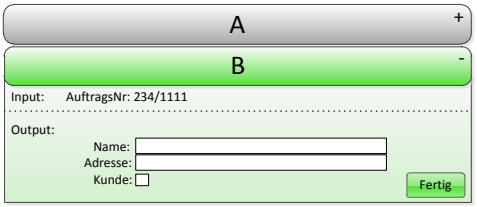

Abbildung 6.2: Beispiel für die Laufzeit-Ansicht

6.2 Weitere Konzepte zur Laufzeit-Darstellung

Mit der Ausführung des Formulars werden neue Aspekte deutlich. So kann zum Beispiel eine Zusammenfassung der bereits eingetragenen Daten hilfreich sein. Oder ein Nutzer möchte eine Nachricht an eine bestimmte Aktivität anhängen. Wie solche Konzepte umgesetzt werden könnten wird im Folgenden näher erläutert.

6.2.1 Detailinformationen einer Aktivität

Um DetailInformationen zu einer Aktivität zu erhalten, speichert jede Aktivität automatisch Detailinformationen. Dazu gehört unter anderem der Name des Nutzers der die Aktivität bearbeitet hat und auch das Datum mit Uhrzeit. Diese Informationen können automatisch vom System gespeichert werden, wenn eine Aktion erfolgreich abgeschlossen ist, zum Beispiel wenn ein Nutzer die eingetragenen Daten übermittelt hat. Da sich der Nutzer am BPM-System anmelden muss, ist das Übernehmen dieser Daten problemlos möglich. Solche Informationen sind besonders wichtig um später nachzuvollziehen von wem Einträge vorgenommen wurden. Auch für den weiteren Verlauf der Instanz können die Informationen bei einer Konfliktsituation nützlich sein.

Detailinformationen müssen nicht immer angezeigt werden und können durch eine weitere klappbare Ebene eingeblendet werden.

6.2.2 Annotationen

Sehr nützlich sind auch Annotationen, die von einem Nutzer zu einer beliebigen Aktivität hinzugefügt werden können. So kann ein Nutzer direkt beim Ausfüllen seiner Aktiviät Annotationen von anderen Nutzern zu dieser Aktion einsehen. Wie in Abbildung 6.3 zu sehen, tauchen Annotationen beim Ausklappen eines Elementes direkt am Anfang auf.

Jeder Nutzer kann Annotationen zu jedem Element hinzufügen, auch wenn er nicht die

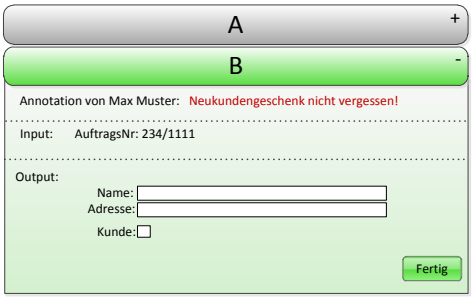

Abbildung 6.3: Beispiel für eine Annotation

Berechtigung hat, das jeweilige Element zu bearbeiten. Das Hinzufügen muss also möglich sein, ohne dass ein Nutzer geschützte Daten einsehen kann. Beim Aufklappen des Elementes müssen solche Daten gegebenenfalls ausgeblendet werden. Wichtig ist, dass bei Annotationen immer der Name des Erstellers mit angezeigt wird, um zum Beispiel Notizen von verschiedenen Nutzern zu unterscheiden.

Eine Erweiterung von Annotationen wäre das Anfügen solcher auch an einzelne Datenelmente. So würden diese Annotationen in jedem Element auftauchen, in dem auch das Datenelement geschrieben oder gelesen wird.

6.2.3 Übersicht der eingetragenen Daten

Bei der fortlaufenden Bearbeitung einer Instanz kann es von Vorteil sein, eine Übersicht zu den bisher eingetragenen Daten anzeigen zu lassen. Im Folgenden werden zwei Umsetzungsmöglichkeiten einer Übersicht vorgestellt.

Eine mögliche Lösung für eine Zusammenfassung der bereits bearbeiteten Elemente ist eine Übersicht direkt über dem aktuellen Element. Alle vorherigen Elemente werden also zusammengefasst. Statt die abgearbeiteten Elemente selbst anzuzeigen wird in der Übersicht hauptsächlich Wert darauf gelegt die Datenelemente übersichtlich darzustellen. So können beispielsweise schrittweise eingegebene Kundeninformationen zusammengefasst in der Übersicht bei der weiteren Bearbeitung sehr hilfreich sein. Die Übersicht soll problemlos ein- und ausschaltbar sein, das heißt der Button hierfür muss leicht zugreifbar platziert werden.

Eine weitere Möglichkeit die Übersicht darzustellen, ohne die alten Elemente auszublenden, wäre direkt neben dem aktuellen Element. Durch einen Button der automatisch immer am Rand des aktuellen Elements hängt kann diese Übersicht seitlich ausgeklappt werden. Die eingetragenen Daten werden dann untereinander angezeigt. Bei dieser Lösung muss ausreichend Platz seitlich zur Verfügung stehen. Die Ansicht könnte sonst u.U. zu lang werden.

6.2.4 Bearbeiterzuordnung

Auch eine Bearbeiterzuordnung wäre in der formularbasierten Darstellung leicht umsetzbar. Da sich ein Nutzer immer in das BPM-System einloggen muss, kann jedem Nutzer eine individuelle Ansicht und Aufgabe zugeteilt werden. Der Ersteller des Prozesses kann im System einen oder mehrere Nutzer für die Bearbeitung einzelner Elemente oder von Subprozessen vorsehen. Um die Bearbeiterzuordnung festzulegen, kann eine gesonderte Ansicht integriert werden oder die Zurodnung folgt in der Modellierungsansicht. Dabei wird der entsprechende Nutzer aus einer Liste ausgewählt und hinzugefügt.

In diesem Kapitel wurde gezeigt, wie ein formularbasiertes Prozessmodell ausgeführt und dargestellt wird. Danach wurden zusätzliche Konzepte, wie eine allgemeine Übersicht, erläutert, welche die Darstellung und Nutzung erleichtern.

7 Zusammenfassung und Diskussion

7.1 Zusammenfassung

Um Kosten und Zeit zu sparen, werden in vielen Unternehmen Prozesse mit Hilfe eines BPM-Systems dokumentiert und automatisiert. Interne Abläufe werden dadurch transparent abgebildet und standardisiert. Durch BPM-Systeme können Fehlerquellen schneller gefunden und Abläufe optimiert werden.

Die Notation, in der Prozesse in solchen Systemen dargestellt werden basiert jedoch häufig auf gerichteten Graphen und kann sehr komplex ausfallen. Für Laien ist es deshalb oft schwierig einen Prozess ohne Grundkenntnisse der Prozessnotation nachzuvollziehen. Um von der, von Workflows übernommenen Notation, abzuweichen und auch die Prozessmodellierung dem Zeitalter des Internets und der webbasierten Systeme anzupassen, wurde in dieser Arbeit eine formularbasierte Prozessmodellierung vorgestellt. Diese soll den Nutzer bereits in der Darstellung unterstützen und auf Bekanntes aufbauen. Eine verbesserte Übersicht wird durch die neue Anordnung und Darstellung der Elemente erreicht. Zudem helfen interaktive Elemente dem Nutzer Informationen aus- bzw. einzublenden.

Um nachzuvollziehen, wie aktivitätenorientierte auf formularbasierte Elemente abgebildet werden, wurden die Grundkonstrukte des ADEPT-Basismodell und dessen Korrektheitsregeln erläutert. Das neue Modell erfüllt diese Regeln, die Elemente des Formulars werden entsprechend von oben nach unten angeordnet und auch dementsprechend gelesen. Dies entspricht einer intuitiveren Darstellung und kann einfacher interpretiert werden. Wichtig ist dabei für parallele und bedingte Verzweigungen ein übergeordnetes Element anzulegen. Diese können als Subprozess eingeklappt werden und verbessern, wie auch einklappbare Daten- und Detailinformationen die Übersicht deutlich. In einer Fallstudie wurde ein Beispielprozess analysiert und in der neuen Darstellung umgesetzt. Hier ist es für den Nutzer wichtig durch ineinander übergehende Animationen beim Ausklappen von Elementen ge-

leitet zu werden. So wird ein Element nicht einfach aufgeklappt, sondern vergrößert sich zuerst in seiner Breite, um Detailinformationen besser darstellen zu können.

Um den Nutzer bei der Modellierung zu unterstützen, kann ein Konzept mit Drag and Drop angewendet werden. Das Prinzip baut auf der Nutzungsweise bekannter Tools auf und ist daher leicht verständlich. Der Nutzer kann alle Elemente aus einer Arbeitsleiste auf den Arbeitsbereich ziehen und wird vom System bei der korrekten Anordnung unterstützt. Dabei ist auf die Korrektheit des Datenflusses zu achten. Beispielsweise muss ein Datenelement vor einem Schreibzugriff mindestens einmal geschrieben worden sein. Will der Nutzer zur Laufzeit übergehen, wird er im Fall eines inkorrekten Datenflusses darauf hingewiesen.

Zur Laufzeit wird in eine andere Ansicht gewechselt, in der beispielsweise vom BPM-System automatisch ausgeführte Pfade ausgeblendet werden können. Der Nutzer wird hier Schrittweise angeleitet und das Formular von oben nach unten abgearbeitet. Dabei werden einzelne Formularfelder ausgefüllt und bestätigt. Einzelne Elemente oder Subprozesse werden hierbei den entsprechenden Bearbeitern zugeordnet. Die Ansicht ändert sich von der normalen Ansicht also wenig, so ist die Ausführung der Aktivitäten für den Nutzer einfacher nachzuvollziehen. Weitere Konzepte, die zur Laufzeit sinnvoll sind wurden vorgestellt. Beispielsweise helfen Annotationen bei der besseren Kommunikation zwischen Bearbeitern und eine Übersicht über eingetragene Daten hilft bei der weiteren Bearbeitung.

7.2 Diskussion

Eine formularbasierte Darstellung und Modellerierung von Prozessen wurde in dieser Arbeit das erste Mal analysiert. Ein prototypisches Szenario wurde vorgestellt, welches den Ansatz für eine Implementierung liefert. Diese könnte als Webanwendung Gestalt finden, was Vorteile, wie Verfügbarkeit und bessere Kommunikation schafft. Da ein Mapping von der Graphennotation möglich ist, könnten auch bestehende Prozessmodelle relativ einfach umgesetzt werden. Mit einem hinterlegten Aufbau in XML ist ein Mapping auf eine andere Notation leicht umsetzbar [11, 12]. So könnte auch zwischen Darstellungen gewechselt werden.

Die Darstellung eines Prozesses als Formular schafft auch Vorteile hinsichtlich der Über-

sicht, wenn der Prozess keinen zu großen Verzweigungsgrad besitzt. Insbesondere eignet sich die Darstellung für eine eher nutzerseitige Bearbeitung, also Prozesse, welche viele Datenelemente besitzen und Nutzereingaben erfordern. Die Übersicht wird nicht durch Datenelemente beeinflusst und das Ausfüllen von Formularen ist eine alltägliche Aufgabe. Auch die Aufgabenverteilung kann in einem webbasierten System mit Hilfe der Formularnotation sehr gut umgesetzt werden. Jeder Nutzer hat im System seinen eigenen Arbeitsbereich mit zu erledigenden Aufgaben. Dabei ist es wichtig eine gute Kommunikationsmöglichkeit bereitzustellen, wie auch IBM's Blueworks Live (Kapitel 2.2.2) zeigt.

Die formularbasierte Prozessmodellierung eignet sich für Nutzer, die Prozesse auf einfache Weise erstellen, verteilen und ausführen möchten. Elemente sind intuitiv verständlich und nutzbar, und so eignet sich die Darstellung auch für Laien. Modellierung und Ausführung benötigen keine getrennte Darstellung mehr und gehen fließend ineinander über.

Literaturverzeichnis

[1] *Appian*. www.appian.com, zuletzt besucht am: 23.11.2011

[2] *ARISalign*. www.alignspace.com, zuletzt besucht am: 23.11.2011

[3] *Intalio*. www.intalio.com, zuletzt besucht am: 23.11.2011

[4] *JQuery 1.6*. jquery.com, zuletzt besucht am: 23.11.2011

[5] AALST, W.M.P. van d. ; BENATALLAH, B. ; CASATI, F. ; CURBERA, F. ; VERBEEK, E.: Business Process Management: Where Business Processes and Web Services Meet. In: *Data Knowledge Engineering*. 2007, S. 1–5

[6] AALST, W.M.P. van d. ; HOFSTEDE, A.H.M. ter: YAWL: yet another worklow language. In: *Information Systems, 30(4)*. 2005, S. 245–275

[7] BRAIG, E.: *Unterstützung von Zeitaspekten in Workfow-Management-Systemen*. Universität Ulm : Diplomarbeit, 2004

[8] DADAM, P. ; KUHN, K. ; REICHERT, M. ; BEUTER, Th. ; NATHE, M.: ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen. In: *Proceedings of GI-Jahrestagung (GISI'95)*. Zürich : Switzerland, 1995

[9] DADAM, P. ; REICHERT, M. ; ACKER, H. ; RINDERLE, S. ; GÖSER, K. ; KREHER, U. ; JURISCH, M. ; LAUER, M.: *ADEPT2 – Ein adaptives Prozess-Management-System der nächsten Generation*. Universität Ulm : Tagungsband Stuttgarter Softwaretechnik Forum, 2006

[10] HOFSTEDE, A. ter ; AALST, W.M.P. van d. ; WESKE, Mathias: Business Process Management: A Survey. In: WESKE, M. (Hrsg.): *Proceedings of 1st International Conference on Business Process Management (BPM'03)* Bd. 2678. Springer, 2003, S. 1019–1019

[11] JOHANNSEN, F.: *Transformation von Prozessmodellen: Bewertung XML-basierter Ansätze*. Hamburg : Salzwasser Verlag, 2007

[12] MENDLING, J. ; NÜTTGENS, M.: XML-basierte Geschäftsprozessmodellierung. In: *Wirtschaftsinformatik 2003/ Band II: Medien Märkte Mobilität*. Physica-Verlag, 2003, S. 161–180

[13] REICHERT, M: *Dynamische Ablaufänderungen in Workflow-Management-Systemen*. Universität Ulm, Diss., 2000

[14] REICHERT, M. ; BAUER, T. ; DADAM, P.: ADEPT – Realisierung flexibler und zuverlässiger unternehmensweiter Workflow-Anwendungen (Invited Paper). In: *Proc. KnowTech 2000*. Leipzig, Germany, 2000

[15] REICHERT, M. ; DADAM, P.: ADEPTflex—Supporting Dynamic Changes of Workflows Without Losing Control. In: *Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, 10 (2).* Springer, 1998, S. 93–129

[16] REICHERT, M. ; DADAM, P.: Geschäftsprozessmodellierung und Workflow-Management-Konzepte, Systeme und deren Anwendung. In: *Industrie Management, 16(3)*. 2000, S. 23–27

[17] REICHERT, M. ; DADAM, P. ; RINDERLE-MA, S. ; GÖSER, K. ; KREHER, U. ; JURISCH, M.: *Von ADEPT zur AristaFlow® BPM Suite – Eine Vision wird Realität: Correctness by Construction und flexible, robuste Ausführung von Unternehmensprozessen*. Technical Report. Universität Ulm, 2000

[18] ROSA, M. L. ; AALST, W.M.P. van d. ; DUMAS, M. ; HOFSTEDE, A.H.M. ter: Questionnaire-based variability modeling for system configuration. In: *Software Systems Modeling* Bd. 2. Springer, 2008, S. 251–274

[19] ROSA, M. L. ; AALST, W.M.P. van d. ; M.DUMAS ; HOFSTEDE, A.H.M. ter ; GOTTSCHALK, F.: *Generating Interactive Questionnaires From Configuration Models, Technical Report*. Queensland University of Technology, Australia : QUT ePrints, 2006

[20] ROSA, M. L. ; LUX, L. ; SEIDEL, S. ; DUMAS, M. ; HOFSTEDE, A.H.M. ter: Questionnaire-driven Configuration of Reference Process Models. In: *Advanced Information Systems Engineering*. Australia : Springer, 2007

[21] WARD-DUTTON, N.: *IBM breaks new ground with Blueworks Live*. MWD Advisors, 2010

[22] WESKE, M.: *Business Process Management*. Springer, 2007

Abbildungsverzeichnis

Name: Janine Barner Matrikelnummer: 664166

Erklärung

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

Ulm, den .....

Janine Barner