Prozessmodellierung und -ausführung im Web

Gero Decker, Lutz Gericke, Stefan Krumnow und Mathias Weske
Hasso-Plattner-Institut, Universität Potsdam
(gero.decker,mathias.weske)@hpi.uni-potsdam.de
(lutz.gericke,stefan.krumnow)@student.hpi.uni-potsdam.de

Abstract: Prozessorientierte Informationssysteme betreffen häufig jeden einzelnen Arbeitsplatz im Unternehmen. Bislang werden diese Systeme von Experten modelliert und realisiert, wobei Anwender typischerweise lediglich in sehr frühen Entwicklungsphasen beteiligt sind. Sie kennen häufig die Modelle ihrer Prozesse nicht und können ihren Beitrag zum Gesamtprozess nicht richtig einschätzen. Aus der fehlenden Transparenz der Prozesse kann eine geringe Benutzerakzeptanz sowie hoher Änderungsaufwand folgen.

In diesem Beitrag stellen wir einen Ansatz zur Prozessmodellierung und -ausführung im Web vor. Er basiert auf Web 2.0 Paradigmen und bezieht Anwender aktiv in die Modellierung und Systemgestaltung ein, sodass die Transparenz der Prozessmodelle und die Akzeptanz prozessorientierter Anwendungen erhöht werden. Der Ansatz ist in Oryx prototypisch realisiert. Basierend auf Anwendungfällen werden die Gesamtarchitektur des Oryx-Systems sowie ein Anwendungsbeispiel vorgestellt.

1 Einleitung

Die Modellierung und softwaregestützte Ausführung von Geschäftsprozessen sind zentrale Instrumente, um Arbeitsabläufe in Unternehmen zu verbessern und effizienter auszuführen [Wes07]. Dabei werden meist Workflow-Managementsysteme (WFMS) eingesetzt, die spezielle Modellierungs- und Ausführungskomponenten besitzen. Typische Workflow-Projekte besitzen zunächst eine Modellierungsphase, in denen Prozessexperten – häufig mit der Unterstützung externer Berater – die Prozessmodellierung und -verbesserung vornehmen. Diese Modelle werden dann mit technischen Informationen angereichert, um in einem WFMS implementiert zu werden.

Die Partizipation von Anwendern findet bei diesem Vorgehen lediglich in frühen Projektphasen statt. Die Modellierung wird von einem kleinen Expertenteam durchgeführt. Der Anwender hat häufig auch keine einfache technische Möglichkeit, auf den Prozess als Ganzes zuzugreifen, ihn zu verstehen und seine spezielle Rolle darin zu erkennen. Zudem erfordert jede Verbesserung des Prozesses, und sei sie noch so klein, einen kompletten Entwicklungszyklus, bei dem das Expertenteam involviert ist, um die Verbesserung zu realisieren.

Basierend auf der Web 2.0-Idee der aktiven Partizipation von Benutzern [VH07] stellen wir in diesem Papier einen Ansatz und das prototypische System Oryx vor, bei dem unterschiedliche Personengruppen auf alle für sie relevanten Prozesse des Unternehmens

zugreifen können. Darüber hinaus ist es möglich – entsprechende Zugriffsregelungen vorausgesetzt –, aktiv in die Modellierung von Prozessen eingebunden zu werden. Auf diese Weise wird die Transparenz bei der Prozessmodellierung erhöht und die Akzeptanz erhöht. Die Konzepte werden anhand eines durchgängigen Beispiels veranschaulicht.

Dieser Bericht ist wie folgt strukturiert. Zunächst werden in Abschnitt 2 unterschiedliche Anwendungsfälle diskutiert. Sodann diskutiert Abschnitt 3 die Modellierung von Geschäftsprozessen. Abschnitt 4 thematisiert die Überführung des Modells in ein Petrinetz, welches eine ausführbare Repräsentation des Modells darstellt. In Abschnitt 5 wird ein konkretes Beispiel für Modellierung und Ausführung vorgestellt, bevor eine Zusammenfassung diesen Bericht schließt.

2 Anwendungsfälle

Dieser Abschnitt erläutert die wichtigsten Use Cases für die Oryx Plattform. Diese untergliedern sich grob in eine Modellierung und Ausführung. Prototyping dient dabei als Zwischenschritt zwischen diesen beiden Phasen.

1: Modellierung Modelle dienen als Dokumentation komplexer Sachverhalte und als Basis für Diskussionen mit unterschiedlichen Prozessbeteiligten. Bei der Modellierung, beispielsweise unter Verwendung von Ereignisgesteuerten Prozessketten (EPK [KNS92]) oder der Business Process Modeling Notation (BPMN [bpm08]), sind durch ein Modellierungswerkzeug syntaktische Restriktionen einzuhalten.

Über den Standardsprachumfang hinaus werden in einzelnen Projekten immer wieder kleinere Spracherweiterungen nötig. Z.B. ist gewünscht, dass spezielle Aktivitätstypen grafisch unterschieden werden oder Wahrscheinlichkeiten bei Fallunterscheidungen modelliert werden können.

Da an der Modellierung meistens mehrere Personen beteiligt sind, muss leichter Zugriff auf das Modell für alle Beteiligten gewährleistet sein. Typischerweise gibt es immer eine Reihe von Personen, die auf ein Modell nur lesend zugreifen wollen oder lediglich kleine Änderungen durchführen, wie z.B. das Ändern eines Bezeichners. Um dieser Situation gerecht zu werden, muss Zugriff nur minimalen Aufwand mit sich bringen.

Das World Wide Web hat gezeigt, wie durch das Konzept von URLs Inhalte einfach zwischen Personen ge*shared* werden können. Bei editierbaren Inhalten wird die entsprechende Anwendung gleich mit ausgeliefert. Z.B. bei Wikis hat man sofort die Möglichkeit, Dokumente zu ändern, wobei diese Änderungen für alle anderen anschließend sofort sichtbar sind. Das gleiche Konzept soll für die Modellierung verwendet werden. Jedes Modell wird durch eine URL identifiziert und sofern eine Editierung des Modells gewünscht ist, wird die Modellierungsanwendung gleich mitausgeliefert. Änderungen am Modell werden dann auch sichtbar für alle anderen.

2: Prototyping von prozessorientierten Anwendungen Gerade Prozessmodelle werden oft als Grundlage für die Erstellung oder Anpassung von Applikationen verwendet. Ein spezieller Fokus liegt auf der Unterstützung der Aktivitäten, die durch Menschen ausgeführt werden. Hier muss man sich darauf einigen, welche Daten in welchem Prozessschritt vorliegen müssen.

Prozessmodelle können in Verbindung mit Definitionen von Formularen bereits zur Erstellung erster Prototypen genutzt werden. Anhand der Prototypen bekommen alle beteiligten Personen einen ersten Eindruck, womit die verschiedenen Mitarbeiter später konfrontiert sein werden.

Wie im Falle der Modelle ist es hier wiederum wünschenswert, dass alle Beteiligten möglichst einfachen Zugriff auf den Prototypen haben. Außerdem soll das Weiterleiten von URLs ausreichen, um Kollegen auf einen bestimmten Schritt in der Prozessausführung hinzuweisen. Essentiell beim Prototyping ist, dass Änderungen am Prozessmodell und an den Formularen schnell im Prototyp reflektiert werden. Die Erstellung und die Bereitstellung des Prototyps muss also idealerweise nur einen Klick erfordern.

Während erste Prototypen sich typischerweise nur auf die Prozesschritte und den Aufbau von Formularen konzentrieren, können in der nächsten Runde von Prototypen bereits Rollendefinitionen vorgenommen werden. Dies hat zur Folge, dass nur noch bestimmte Personen auf eine Aktivität Zugriff haben.

3: Implementierung und Betrieb von prozessorientierten Anwendungen Ähnlich wie es bei Wikis einen fließenden Übergang gibt zwischen ersten Entwürfen eines Dokuments hin zur finalen Version, kann es auch bei der Entwicklung prozessorientierter Anwendungen einen solchen fließenden Übergang geben. Implementierung und Betrieb können dabei scheinbar verschwimmen.

Besonderer Augenmerk liegt hier natürlich auf der Anbindung der Prozesssteuerung auch an externe Systeme. Diese werden entweder über Web Service Schnittstellen angesprochen oder über einfache Webformulare. Eine entsprechende Konfiguration von automatischen Aktivitäten in einem Prozess sind daher nötig.

In Analogie zu Wikis kann die fertige prozessorientierte Anwendung in der gleichen technischen Umgebung wie die Prototypen laufen. Es ist natürlich darauf zu achten, wie oft ein Prozess tatsächlich ausgeführt werden soll. Eventuell wird eine leistungsfähigere Infrastruktur benötigt, als dies für die Entwicklung von Prototypen der Fall ist. Aber gerade für Prozesse mit hohem Anteil an menschlichen Aktivitäten ist dies eher selten der Fall.

Eine wichtige Anforderung ist, dass Mitarbeiter verschiedener Organisationen auf das System Zugriff haben. Dies wird auch im Beispiel deutlich, welches in den Abschnitten 3 und 5 vorgestellt wird.

3 Prozessmodellierung in Oryx

Oryx wurde hauptsächlich zur webbasierten Modellierung von Geschäftsprozessen (Use Case 1) entwickelt. Er erfüllt die Anforderung an arbeitsteilige Erstellung von Prozessmodellen und ist durch seinen generischen Aufbau einfach um zusätzliche Notationen sowie neue Funktionalität erweiterbar. Abbildung 1 zeigt die Architektur des Modellierungstools, das in einen Browser geladenen und dort ausgeführt wird, sowie dessen Backend.

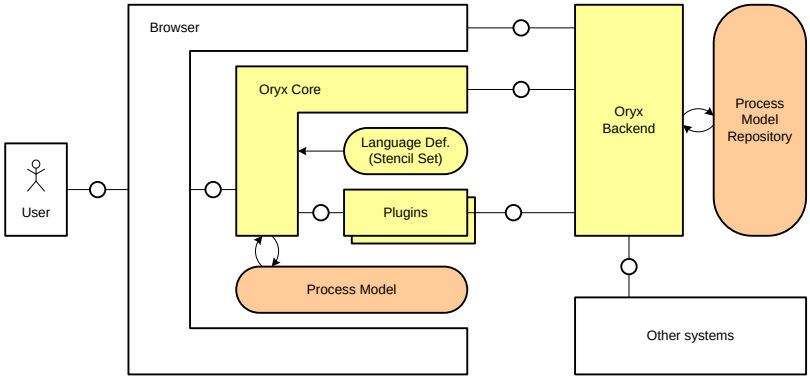

Abbildung 1: Architektur des Oryx

3.1 Webbasierte Modellierung mit Oryx

Die Modellierungsumgebung Oryx wurde als Sammlung von JavaScript Routinen, die als Teil des ausgelieferten Dokuments in einen Browser geladen werden, implementiert. Dieses Dokument enthält ein Prozessmodell und kann über eine URL eindeutig identifiziert werden. Um die Prozessmodelle zu repräsentieren wird der RDF Standard, bzw. Embedded RDF (eRDF) [Dav06] verwendet. Das eRDF Format ermöglicht es, Metadaten direkt in ein HTML Dokument zu schreiben und daraus RDF zu extrahieren.

Durch diese Form der Repräsentation sind Oryx Modelle einfach portierbar. Durch Extraktion des Modells im RDF Format kann dieses in anderen Modellierungs- oder auch in Ausführungsumgebungen verwendet werden.

Die mit dem Prozessmodell mitgelieferten JavaScript Routinen sind dafür verantwortlich, das grafische Benutzer Interface des Oryx (siehe Abbildung 2) im Browser zu erzeugen, Modellierungsfunktionalität bereitzustellen und Daten mit dem Backend auszutauschen. Die im aktuellen Diagramm modellierten Elemente finden sich als JavaScript Objekte im Browser wieder und können so einfach editiert werden. Beim Speichern eines Modells werden die Elemente im eRDF Dokument asynchron an das Backend geschickt und dort in einer Datenbank gesichert.

Das Benutzerinterface des Oryx stellt neben einer Zeichenfläche eine Toolbar, ein Shape

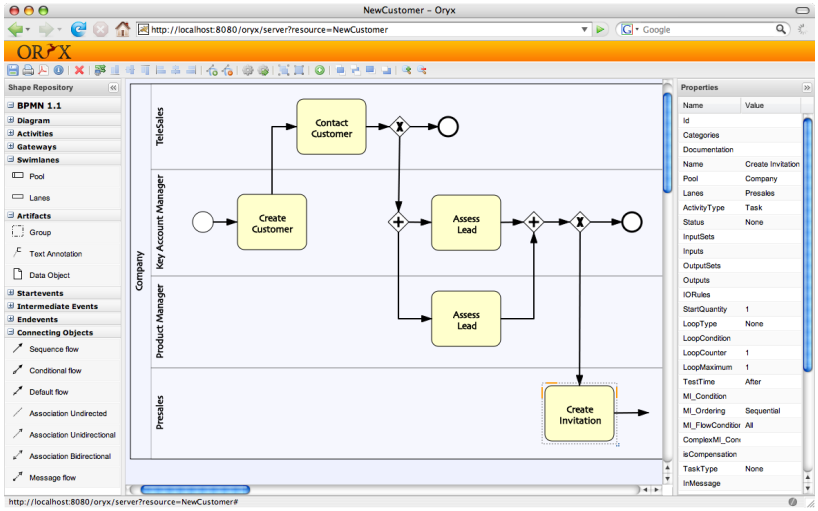

Abbildung 2: Beispielprozess im Oryx

Repository sowie eine Property Übersicht bereit. Zur technischen Realisierung wird das Framework ExtJS (`http://extjs.com`) verwendet, das Widgets zur Programmierung einer Rich Internet Application bereitstellt.

Um die Modelle, bzw. deren Knoten und Kanten, auf der Zeichenfläche anzuzeigen, werden diese als Scalable Vector Graphics (SVG) in den Browser geladen. Durch die Verwendung von SVG lassen sich Modellierungsfunktionen wie das Verschieben, Vergrößern, Verkleinern oder Editieren von Modellelemente leicht implementieren, da für die grafische Anzeige der SVG der Browser verantwortlich ist. Fast alle modernen Webbrowser bieten heutzutage SVG-Unterstützung, wenn auch nicht im vollen Umfang.

Wie bereits beschrieben ist jedes Prozessmodell im Oryx durch eine URL eindeutig addressiert. Verschiedene Mitarbeiter mit Zugriff auf den gleichen Server können Modelle durch bloßes Weiterreichen einer URL austauschen und zusammen daran arbeiten. Darüberhinaus kann jede Modellressource unterschiedliche Repräsentationen aufweisen. So kann neben dem im Oryx editierbaren Dokument auch eine PDF-Repräsentation oder eine ausführbare Repräsentation des Prozesses angefordert werden.

Um, ähnlich wie in Wikis, Änderungen an Prozessen Mitarbeitern zuordnen zu können, müssen diese dem System bekannt sein. Hierzu wurde eine OpenID [ope07] basierte Authentifizierung implementiert.

Da Geschäftsprozesse sensible Informationen enthalten können, ist es manchmal wünschenswert Lese- und Schreibrechte zu vergeben, bzw. vorzuenthalten (Autorisierung). Hierzu kann ein Oryx Nutzer auf einem Prozess entweder als *Owner*, *Contributor*, *Viewer* oder in keiner Rolle agieren. Zur Realisierung der Rechte, die die Nutzer einer Rolle erhalten, werden die Rollen auf HTTP-Request-Methoden, die für die jeweiligen Benutzer zulässig sind, abgebildet. Die genaue Abbildung der Rechte ist in Tabelle 1 dar-

gestellt.

Rolle	HTTP Methoden
Owner	GET, POST, PUT, DELETE
Contributor	GET, PUT
Viewer	GET
keine	keine

Tabelle 1: Rechteverwaltung im Oryx

3.2 Erweiterbarkeit des Oryx

Damit verschiedene Notationen in Modellierungsprojekten verwendet werden können, wurde Oryx mit einem Mechanismus zum Auflösen beliebiger Modellierungssprachen versehen. Eine Notation wird in Oryx durch ein Stencil Set repräsentiert. Diese Stencil Sets werden beim Öffnen eines Prozessmodels mit in den Editor geladen. Theoretisch kann ein Modell auch mehrere Notationen auf einmal verwenden.

Kernstück eines Stencil Sets ist eine JavaScript Object Notation (JSON) Datei, die das Metamodell der Notation enthält. In ihr werden die Typen aller Modellelemente definiert. Dabei werden zunächst Kanten und Knoten unterschieden. Jeder Elementtyp hat eine ID und einen Namen. Darüberhinaus verfügt ein Typ über eine Menge von Properties, die bei der Modellierung gesetzt werden können. Beispiele hierfür sind der Name oder der *LoopType* einer BPMN Task. Jede Property hat unter anderem einen Datentypen und einen Standardwert.

Eine SVG Datei wird als Vorlage für die Modellelemente eines Typs auf der Zeichenfläche referenziert. Dabei können XML-Knoten innerhalb der SVG Datei an Properties des Elementtyps gebunden werden, so dass die Propertywerte Einfluss auf die grafische Erscheinung des Modellelements bekommen. Durch diesen Mechanismus wird zum Beispiel der Name eines Elements angezeigt. Neben der SVG Datei wird auf eine 16 mal 16 Pixel große Grafik, die als Icon im Shape Repository dient, verwiesen.

Um die Beziehungen, die zwischen Modellelementen bestehen können, auszudrücken, werden im Stencil Set Kompositionsregeln angegeben. Es gibt dabei Regeln zu Verbindungen zwischen und zum Enthaltensein von Elementen. So kann beispielsweise ausgedrückt werden, dass eine BPMN *Task* in einer *Lane* liegen und mit anderen Tasks im selben *Pool* per Sequenzfluss verbunden werden darf.

Um Oryx um eine zusätzliche Notation zu erweitern, muss nur ein neuer Unterordner im Stencil Set Verzeichnis angelegt werden. In diesen Ordner werden die JSON sowie die Grafikdateien der neuen Notation gespeichert. Es existieren bereits eine Reihe von Stencil Sets: BPMN 1.0 und 1.1 werden durch Oryx ebenso unterstützt wie EPK, Petri und Workflow-Netze.

Neben der Erweiterbarkeit durch Stencil Sets kann Oryx auch funktional durch Plugins

erweitert werden. Plugins werden beim Laden des Oryx initialisiert und registrieren dabei ihre Funktionalität im Nutzerinterface. Wird die Funktion eines Plugins angestoßen, so hat dieses Zugriff auf die gesamten JavaScript Objekte des aktuellen Prozessmodells sowie auf alle anderen geladenen Ressourcen.

Der Großteil der Modellierungsfunktionalität in Oryx ist bereits in Plugins implementiert. Während der Oryx Core aus Abbildung 1 für den Aufbau der Zeichenfläche und des Editor Layouts verantwortlich ist, werden Funktionen wie das Speichern von Modellen oder das Ausrichten von Modellelementen durch Plugins realisiert. Selbst die Toolbar, in der die meisten Plugins durch Icons repräsentiert werden, ist als Plugin implementiert.

Plugin-Entwicklung kann vor allem dann notwendig werden, wenn weitere Notationen in den Oryx eingebunden werden und hierfür zusätzlich spezielle Funktionalität gewünscht ist. So können zum Beispiel PNML Im- und Exporter für Petrinetze eingebunden werden. Auch ein Dienst zur Transformation von BPMN in ausführbare Petrinetze ist durch ein Plugin im Oryx integriert.

Durch Verwendung von verschiedenen Webtechnologien ist es gelungen, mit Oryx eine im World Wide Web beheimatete Modellierungsumgebung zu implementieren. Benutzer haben so den Vorteil ohne Installation von jedem mit dem Internet verbundenen Rechner aus arbeitsteilig mit anderen Nutzern Geschäftsprozessmodelle zu erstellen und zu verwalten.

Mit seinem Stencil Set Mechanismus unterstützt Oryx verschiedene Notationen in Modellierungsprojekten und kann gegebenenfalls auch um zusätzliche Notationen erweitert werden. Zusätzlich kann auch weitere Funktionalität, die mit der Einführung einer neuen Notation notwendig wird, durch Stencil Set abhängige Plugins in den Oryx eingebunden werden.

4 Prozessausführung

Die in der Modellierungsumgebung erzeugten Modelle können durch die Transformationskomponente als Petrinetze zur Ausführung gebracht werden. Dabei werden als theoretische Grundlage gefärbte Petrinetze [Jen96] mit Guard-Conditions verwendet. Eine Worklist stellt das Frontend für den Nutzer dar, in der Tasks bearbeitet, delegiert und überwacht werden können. Im Folgenden werden die einzelnen Komponenten der Ausführung näher betrachtet.

4.1 Transformation

Als Eingabeformat dient RDF, wobei der vollständige BPMN Standard mittels Subjekt (z.B. Task), Prädikat (z.B. Assoziation) und Objekt (z.B. Attribut) abgebildet ist. Ausgabe der Transformation ist ein Dokument in PNML (Petri Net Markup Language [BCvH$^+$03]), das in der Engine deployed wird. PNML sieht die Definition von Transitionen, Stellen und Kanten vor, wobei beliebiger XML-Content für toolspezifische Angaben angegeben

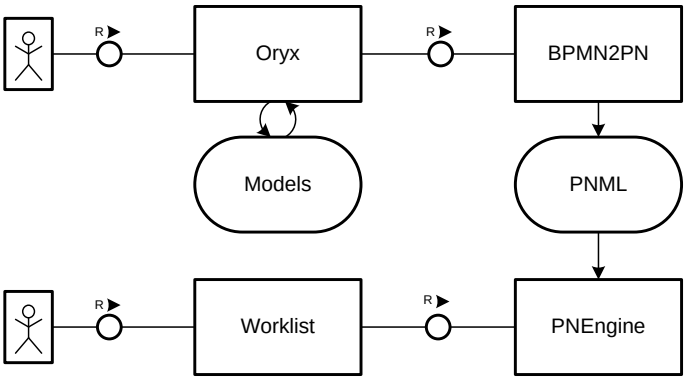

Abbildung 3: Oryx-Gesamtarchitektur

werden kann. Dieser Teil wird für spezielle Ausführungsattribute wie Guard-Conditions benutzt. Formulare, die aufbauend auf den definierten Datenobjekten generiert werden, können ebenso im PNML referenziert werden.

Die Umwandlung läuft in 3 Schritten ab:

1. Import der Daten aus RDF und Erzeugung einer Instanz des BPMN-Metamodells
2. Umwandlung der Datenstruktur von BPMN in Petrinetz
3. Export der Petrinetz Metamodell-Instanz im PNML-Format

Das BPMN Metamodell wurde in einer Java-Klassenhierarchie implementiert, um das importierte RDF-Dokument in eine Objektstruktur zu überführen. Ebenso gibt es ein erweitertes Petrinetz-Metamodell, das es ermöglicht, nicht nur zwischen normalen Transitionen und τ-Transitionen zu unterscheiden, sondern beispielsweise auch FormTransitions, die es erlauben, Formulare zur Bearbeitung der Transition zu hinterlegen. Durch die erzeugten Objektstrukturen können Validierungen oder Optimierungen auf die Netze angewendet werden, wie z.B. Soundness-Checks. Darüber hinaus lassen sich zu den Transitionen boolsche Ausdrücke angeben, die als Guard-Conditions regeln, wann eine Transition schaltbereit ist.

Die Umwandlung geschieht durch eine Transformationskomponente, die an [DDO08] angelehnt ist. Zunächst findet eine Überführung von Kontrollflusstrukturen, Datenflussstrukturen und Rollenzuweisungen statt, wie in Abbildung 4 zu sehen ist. Auffällig ist hier, dass es keine Startstelle gibt, sodass die Prozessinstanzierung durch Feuern der ersten Transition erfolgen kann. Weiterhin werden Guard-Conditions für die Engine aufbereitet und sogenannte Locators erzeugt, die Daten innerhalb der Token adressieren. Ein Beispiel dafür ist das unterschiedliche Schaltverhalten abhängig vom Auftragsvolumen, wie es im Token definiert ist. Abschließend wird ein bestimmter Aktivitätslebenszyklus implementiert. Dieser sieht die Zustände 'enabled', 'running' und 'finished' vor.

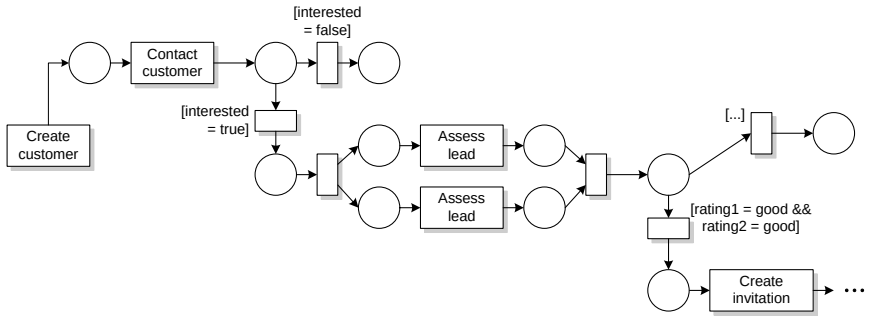

Abbildung 4: Beispielprozess als Petrinetz

4.2 Petrinetzausführung

Die Petrinetzausführungsengine folgt dem ressourcenorientierten Architekturstil (REST [Fie00]). Sie ist in Ruby implementiert und nutzt zur Kommunikation ausschließlich das HTTP-Protokoll [FGM+99]. Eine Ressource wird auf verschiedene Weise dargestellt, abhängig davon, ob mit einem Browser, einer weiteren Ausführungsengine oder einem anderer Dienst kommuniziert wird. So wird für einen Browser XHTML gerendert, für die aggregierte Worklist-Darstellung hingegen datenorientiertes XML.

Kommunikation mit Ressourcen geschieht über die HTTP-Verben GET, POST, PUT und DELETE. Am Beispiel der Ressource Petrinetz soll diese Sicht verdeutlicht werden.

HTTP Methode	Bedeutung	Eigenschaften
GET	Petrinetzbeschreibung ausgeben	seiteneffektfrei, idempotent
POST	Petrinetz aus PNML erstellen und URL zurückgeben	-
PUT	Petrinetzbeschreibung überschreiben	idempotent
DELETE	Petrinetz löschen	idempotent

Tabelle 2: REST-konforme Petrinetz-Schnittstelle

Durch die ressourcenorientierte Sicht wird erreicht, dass Petrinetze unter Nutzung einer uniformen Schnittstelle verteilt miteinander interagieren können. Dadurch lassen sich eine URL ohne Zusatzinformationen weiterreichen um ein 'Lesezeichen' im Prozess zu setzen. Die Engine läuft auf einem Webcontainer, der mehrere Threads zur Bearbeitung der Requests nutzt. Konflikte werden auf Datenbanktransaktionsebene aufgelöst.

Wie bereits angedeutet sind in der PNML-Beschreibung drei Arten von Transitionen vorgesehen: 'automatic', 'receive' und 'send'. Automatic steht für τ-Transitionen, die schalten, sobald sie aktiviert sind. Dabei kann durch ein hinterlegtes XSLT-Dokument eine Datentransformation von Input- auf Outputstellen durchgeführt werden. So können beispielsweise Dokumente aufgeteilt oder dupliziert werden. Receive-Transitionen stellen Transitionen dar, die erst durch Einwirkung von außen angestoßen werden. Hier gibt es zusätzlich

die Unterscheidung zwischen formularbasierter Nutzerinteraktion oder einfachem Schalten ohne Interaktion, bei dem aber eine automatisierte Transformation der Eingabedaten mittels XSLT stattfinden kann. Der Nutzer kann dabei aber auch ein externes System sein, das Daten von außen ins System bringt, z.B. ein Warenwirtschaftssystem, das einen Bestellprozess für fehlende Produkte anstößt. Send bezeichnet Transitionen, bei denen Inhalt an ein externes System geschickt werden kann, beispielsweise an einen Bezahldienst.

Die für den in erster Linie menschlichen Benutzer gedachten Repräsentation der Formular-Transitionen werden mittels XForms [xfo07] definiert. Dieser W3C-Standard wird Teil des XHTML 2.0 Standards sein. Es wird ein MVC-Muster genutzt, das es ermöglicht, Daten mit ihrer Darstellung und evtl. benötigten Steuerinformationen zu verbinden. Daten sind hier die Tokenwerte und durch XML-Dokumente repräsentiert. Nutzereingaben in Formularfelder werden dementsprechend direkt in das Ausgabemodell übernommen und finden so den Weg in die erzeugten Tokens. Die Spezifikation der Datenmodelle kann mittels XML Schema erfolgen.

4.3 Worklistclient

Der Worklistclient stellt die Benutzungsschnittstelle für den menschlichen Anwender dar. Eine solche Aufgabenliste präsentiert dem Nutzer alle momentan auszuführenden Tasks, die in der Engine aktiv sind.

Umgesetzt ist die Worklist-Oberfläche mit dem JavaScript-Framework ExtJS. Dabei wur-

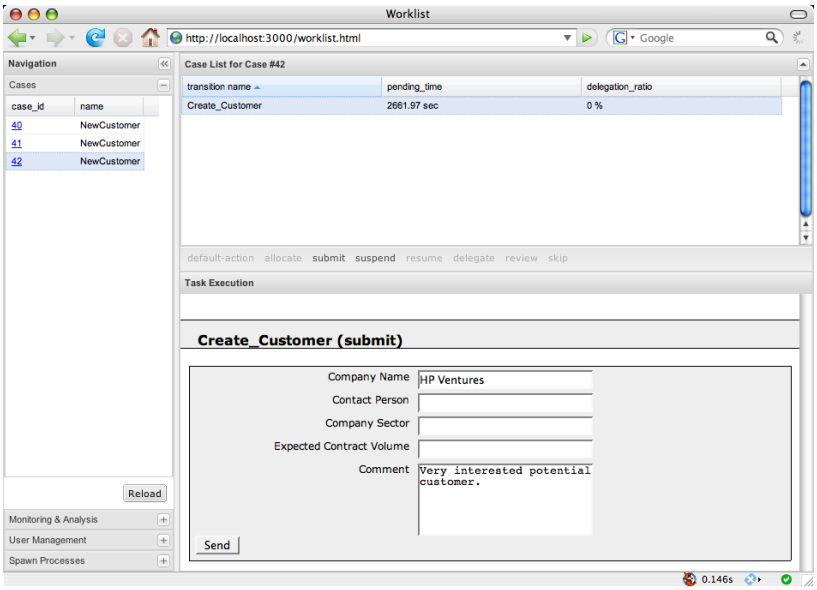

Abbildung 5: Beispielprozess im Worklistclient

de die dreigeteilte Oberfläche von Mail-Clients nachempfunden. Die Kommunikation mit der Engine findet per HTTP im Stile von AJAX statt. Asynchrone HTTP-Requests werden vom JavaScript XMLHttpRequest Object ausgeführt und deren Rückgabe im User Interface dargestellt. Aufgelistet werden alle Petrinetze, die bisher deployed wurden und es dem Nutzer ermöglichen neue Instanzen zu erzeugen. Die aktiven Tasks der laufenden Instanzen werden zusätzlich mit statistischen Informationen (z.B. Bearbeitungszeit) aufbereitet. Schaltaktionen können direkt ausgeführt werden und die XForms-Formulare werden im zentralen Fenster eingeblendet, da auf den Eingabedaten der Hauptaugenmerk liegt.

5 Beispiel

Im Folgenden wollen wir anhand des Beispiels aus Abbildung 2 zeigen, wie die Oryx Plattform bei der Umsetzung eines neuen Geschäftsprozesses eingesetzt werden kann.

Das Beispiel zeigt einen Ausschnitt eines Vertriebsprozesses. Ein Teil des Prozesses ist ausgelagert an einen TeleSales Dienstleister, wo im CallCenter Kunden telefonisch kontakiert werden. Der Prozess startet damit, dass potentielle Kunden durch den Key Account Manager in das System eingepflegt werden, sogenannte “Leads”. Die Kontaktdaten dieser potentiellen Kunden wurden eingekauft und dienen der Kaltakquise. Der TeleSales Dienstleister übernimmt das eigentliche Kontaktieren der potentiellen Kunden. Der CallCenter Mitarbeiter trägt die Ergebnisse jedes Telefongesprächs in das System ein. Hier wird bereits festgestellt, ob generelles Interesse besteht. Falls nicht, werden für den Kunden keine weiteren Aktivitäten nötig. Andernfalls überprüfen der Key Account Manager sowie der entsprechende Product Manager, ob der Kunde wirklich interessant ist. Ist dies nicht der Fall, wird der Kunde wieder aussortiert. Ansonsten werden weitere Aktivitäten eingeleitet, die aber nicht mehr im Prozessmodell zu sehen sind.

Die Merkmale, durch die sich dieser Prozess auszeichnet, sind

- Beteiligung und Kollaboration mehrerer Nutzer und
- Nebenläufigkeit.

Um diesen Prozess durch ein Workflow-System unterstützt zur Ausführung bringen zu können, muss er zunächst modelliert werden. Hierfür setzen sich Vertriebsmitarbeiter mit einem externen Berater zusammen und erstellen eine prototypische Version des Prozessmodells. Dabei sollen auch die Geschäftsleitung und weitere Mitarbeiter lesend die Fortschritte verfolgen. Da hierbei mehrere Bearbeiter über den Entwicklungszeitraum viele kleinere Änderungen vornehmen wollen, bietet sich der vollständig webbasierte Oryx als Modellierungswerkzeug an. Die Tatsache, dass Oryx ohne Installation verfügbar ist, unterstützt das Team darin, kurzfristig andere Bearbeiter lesend oder schreibend in die Entwicklung einzubeziehen.

Auf Grund der weiten Verbreitung der Notation und ihrer Ausführbarkeit in der Oryx-Plattfom wird bei der Modellierung BPMN verwendet. Den ausmodellierten Prozess haben wir bereits in Abbildung 2 gezeigt.

Der erste Prozessentwurf kann als Basis für das Prototyping dienen. Abbildung 4 hat bereits das Petrinetz vorgestellt, das aus dem Prozess erzeugt wird.

Durch das sofortige Ausführen in der Worklist (Abbildung 5), kann der Prototyp umgehend evaluiert werden. Schwachstellen können dabei entdeckt und durch Korrektur im Modell und erneuter Transformation in kurzen Iterationen behoben werden. Nach den ersten Zyklen können auch Mitarbeiter des Partners TeleSales in die Evaluierung und ggf. in die Modellierung einbezogen werden. Der Bedarf an zusätzlichen Daten an einem bestimmten Prozessschritt, kann auch sofort erfüllt und getestet werden. Diese agile Form der Prozessentwicklung ermöglicht es, schnell zu ausgereiften prozessorientierten Anwendungen zu kommen.

6 Zusammenfassung und Ausblick

In diesem Beitrag haben wir die webbasierte Prozessplattform Oryx motiviert und vorgestellt. Diese Plattform ermöglicht zum einen webbasiertes verteiltes Modellieren von Prozessmodellen. Zum anderen bietet sie die Möglichkeit der automatischen Erstellung und Bereitstellung von webbasierten prozessorientierten Anwendungen. Oryx ist sowohl technisch als auch von seinen Use Cases her eine typische Web 2.0 Anwendung.

Die Implementierung ist frei verfügbar als Open Source Projekt unter der MIT Lizenz. Eine lauffähige Installation ist zu finden unter `http://oryx-editor.org`.

Oryx ist der erste Schritt hin zu einem *Application Wiki*. Im Gegensatz zu einem klassischen Wiki, welches Dokumente verwaltet und dezentrales Editieren zulässt, bietet ein Application Wiki darüber hinaus die Möglichkeit, Aktivitätsabfolgen zu verwalten und dezentral solche Abfolgen zu editieren.

Während der Fokus der Oryx-Plattform bislang eher auf Kontrollflussaspekten gelegen hat, wird die Definition von Formularen und Datenstrukturen im Oryx momentan nur sehr rudimentär unterstützt. Bislang müssen Formulare außerhalb des Oryx editiert werden und werden lediglich als Konfigurationen an Prozessmodelle angehängt. Daher wird in laufenden Arbeiten die webbasierte grafische Editierung von Formularen angegangen.

Danksagungen. Wir danken dem Oryx-Team für die umfangreichen Entwicklungen an der Modellierungsplattform und an der Ausführungsengine.

Literatur

[BCvH+03] Jonathan Billington, Søren Christensen, Kees M. van Hee, Ekkart Kindler, Olaf Kummer, Laure Petrucci, Reinier Post, Christian Stehno und Michael Weber. The Petri Net Markup Language: Concepts, Technology, and Tools. In *24th International Conference on the Applications and Theory of Petri Nets (ICATPN)*, LNCS, Seiten 483–505, Eindhoven, The Netherlands, June 2003.

[bpm08] Business Process Modeling Notation, V1.1. Bericht, Object Management Group (OMG), Jan 2008. http://www.omg.org/spec/BPMN/1.1/PDF/.

[Dav06] Ian Davis. Rdf in Html. Bericht, Talis Information Limited, 2006. http://research.talis.com/2005/erdf/wiki/Main/RdfInHtml.

[DDO08] Remco M. Dijkman, Marlon Dumas und Chun Ouyang. Semantics and Analysis of Business Process Models in BPMN. *Information and Software Technology (IST)*, 2008.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach und T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Bericht, The Internet Engineering Task Force, 1999. http://www.ietf.org/rfc/rfc2616.

[Fie00] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. Dissertation, University of California, Irvine, 2000. Chair-Richard N. Taylor, http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm.

[Jen96] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Volume 1*. Springer, 1996.

[KNS92] G. Keller, M. Nüttgens und A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Heft 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1992.

[ope07] OpenID Authentication 2.0 - Final. Bericht, OpenID Foundation, December 2007. http://openid.net/specs/openid-authentication-2_0.html.

[VH07] Gottfried Vossen und Stephan Hagemann. *Unleashing Web 2.0: From Concepts to Creativity*. Morgan Kaufmann, 2007.

[Wes07] Mathias Weske. *Business Process Management*. Springer, 2007.

[xfo07] XForms 1.0 (Third Edition). Bericht, W3C, 2007. http://www.w3.org/TR/xforms/.