Universität Rostock Traditio et Innovatio

WIRTSCHAFTSINFORMATIK UNIVERSITÄT ROSTOCK

Entwicklung eines Vorgehens zur Verwaltung der Personalisierungs-regeln im Lernportal des KOSMOS Projektes

eingereicht am 26.03.2015

von

Dorian Wojda | Stindestr. 23 | 12167 Berlin

Matrikel-Nr.: 210206193

Gutachter:

Dipl.-Wirt. Inf. Dirk Stamer

Universität Rostock

Albert-Einstein-Str. 22

18059 Rostock

Zweitgutachter:

Dr. Birger Lantow

Universität Rostock

Albert-Einstein-Str. 22

18059 Rostock

Kurzfassung

Im Rahmen des KOSMOS Projektes an der Universität Rostock soll ein Portal (MeinKOSMOS) entwickelt werden, welches das Prinzip des *Lebenslangen Lernens (LLL)* verfolgt. Das KOSMOS unterstützt den Prozess des *LLL* eines Menschen, indem es den Werdegang und die Weiterentwicklung beruflicher Kenntnisse, Fertigkeiten und Fähigkeiten fördern möchte. Zugleich soll das KOSMOS Projekt dem aktuellen negativen gesellschaftlichen Wandel Deutschlands entgegen wirken. Die stetig sinkende Anzahl von Fachkräften in Deutschland wird zunehmend ein wichtiges Thema.

MeinKOSMOS soll als Lern-Portal den Nutzer weitgehend im Lernprozess begleiten und unterstützen. Diese Unterstützung soll sich in der hohen Personalisierung des Portals widerspiegeln. Abhängig von den Vorkenntnissen und Hintergründen eines Nutzers sollen Empfehlungen generiert werden. Solche Empfehlungen könnten relevante Applikationen, Prüfungserinnerungen, Dozenten, Dokumente oder Diskussionsthemen darstellen.

Diese Arbeit soll eine geeignete Vorgehensweise entwickeln, die den Schwerpunkt der Personalisierung innerhalb von MeinKOSMOS unterstützen soll. Umgesetzt wird die Personalisierung durch eine prototypische Empfehlungskomponente. Diese Empfehlungskomponente benutzt ein regelbasiertes System (RBS), um Entscheidungen bzw. Empfehlungen zu treffen. Jede Empfehlung folgt einer Logik, die in die Regel-Datei geschrieben ist. Aufgrund der prototypischen Entwicklung ist die Verwaltung der Regeln sehr umständlich. Jede Änderung bringt eine direkte Anpassung des Quelltexts in den Javadateien mit sich. Dies erfordert hohe technische Kenntnisse an die Bearbeiter. Zudem existieren keine Regelungen über die Sicherungen dieser Dateien.

Ferner soll hier eine Vorgehensweise erarbeitet werden, die sich auf die Erstellung, Weiterentwicklung und Verwaltung von Regeln konzentriert. Dabei soll eine Möglichkeit geschaffen werden, um die Regeln ohne einen direkten Eingriff in die Javadatei und ohne Programmierkenntnisse zu gestalten.

Schlagwörter: Drools, jBPM, MeinKOSMOS, Liferay, Personalisierung, Portal, RBS

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einführung **1**

2 Grundlagen **4**

2.1 KOSMOS Projekt 4

2.2 MeinKOSMOS 6

2.2.1 Portalsysteme 6

2.2.2 Technische Konzeption 9

2.2.3 Organisatorische Konzeption 10

2.3 Regelbasierte Systeme 12

2.4 Grundlagen von BPMN 15

2.5 Personalisierungskomponente 20

3 Konzept **27**

3.1 Allgemeine Anforderungen 27

3.2 Menschliche Dimension 28

3.3 Technische Dimension 30

3.3.1 Allgemeine Systemanforderungen 30

3.3.2 Schnittstelle zur Empfehlungskomponente 32

3.4 Gesamtkonzept 36

4 Umsetzung **38**

4.1 Systemeinrichtung 38

4.1.1 Integration als Portlet 39
4.1.2 Integration als eigenständiges System 40
4.2 Einblick in jBPM 43
4.3 Realisierung der Schnittstelle 47

5 Kritische Betrachtung **50**
5.1 Vergleich 50
5.2 Kritik, Probleme und Lösungen 53

6 Zusammenfassung und Ausblick **55**
6.1 Zusammenfassung 55
6.2 Ausblick 57

Anhang A: GitController Objektklasse **59**

Anhang B: Zusatzfunktion von MainController **63**

Anhang C : jBPM Installationsanleitung **65**

Anhang D : Konsolenausgabe jBPM als Portlet **70**

Anhang E : Support-Portlet Fehlerbehebung **71**

Anhang F : Test von Konzept und MeinKOSMOS **72**

Anhang G : Anleitung zum Git **74**

Anhang H : Inhalt der DVD-ROM **76**

Literaturverzeichnis **77**

Abbildungsverzeichnis

2.1 Erwerbspersonenpotenzial in Millionen bis 2025 [Arb11, S. 7] 5
2.2 Interface Beispiel eines Portals mit Portlets nach Lehner [Leh14] 7
2.3 Portalbestandteile nach Rütschlin [Joc01] 8
2.4 Screenshot von MeinKOSMOS (Stand März 2015) 12
2.5 RBS Bestandteile nach Rütschlin [Joc01] 13
2.6 Komplexes BPMN 2.0 Beispiel nach Kocian [Koc11, S. 16] 18
2.7 Nutzerrollen in MeinKOSMOS [Sam14, S. 28] 20
2.8 Weitere Objekte im Nutzerprofil [Sam14, S. 29] 21
2.9 Architektur auf Basis des Observer Patterns [Sam14, S. 41] 22
2.10 BPMN Regel "Prüfung"[Sam14, S. 53] 23
2.11 Teilnehmer Klassen Objekt [Sam14, S. 57] 24
2.12 BPMN Regel "Vorgegebene Applikation"[Sam14, S. 54] 24
2.13 Support-Portlet Interface aus MeinKOSMOS 25
2.14 MainController Klasse [Sam14, S. 61] 26

3.1 Liferay auf Tomcat mit Datenbank 31
3.2 'Hallo Welt' Beispiel eines Portlets 32
3.3 Ordnerstruktur von Support-Portlet 33
3.4 Managementtool integriert als Portlet in MeinKOSMOS 36

4.1 jBPM integriert als eigenständiges System mit 2 Varianten der Schnittstelle 41
4.2 Projektübersicht aus jBPM mit Projektdateien 44
4.3 Prozess-Modellierungstool von jBPM 45
4.4 Data-Modeller von jBPM 46
4.5 DRL-Editor mit Faktenübersicht 47
4.6 UML-Schema von GitController 48
4.7 Neues UML-Schema von MainController 49

Tabellenverzeichnis

2.1 Basiselemente bei BPMN 2.0 nach [bpm15] 16
2.2 Notationsvorschriften bei BPMN 2.0 19

3.1 Übersicht der Konzepteigenschaften 36

5.1 Übersicht des Vergleichs 53

Listings

2.1 Regel "vorgegebene Applikation"nach Ackermann [Sam14, S. 64] 25
2.2 Auslösen der Regeln nach Ackermann [Sam14, S. 62] 26

3.1 Import und Ausführung der Regeln nach Ackermann [Sam14] 34
3.2 Import von Drools aus MainController nach Ackermann [Sam14] 35

1 Einführung

"Weiterbildung ist ein zentrales gesellschaftliches Thema. In jeder Lebensphase ist das organisierte Weiterlernen des Menschen für die Gestaltung seines Lebens wichtig."[kos15] Diese Idee kann als *Lebenslanges Lernen (LLL)* bezeichnet werden. Die Universität Rostock hat sich zur Aufgabe gemacht genau dieses Prinzip im Rahmen vom KOSMOS Projekt umzusetzen. Ferner sollen Möglichkeiten geschaffen werden, die zielgruppenspezifischen Bedarfe für studierende und weiterbildende Personen an der Universität bereitstellen. Insbesondere sollte dabei beachtet werden, dass jeder Mensch einen eigenen und speziellen Werdegang durchlebt, auch außerhalb der akademischen Laufbahn. Zudem soll mit Hilfe des Projektes die Bildungsbeteiligung gestärkt werden. Somit soll es für jeden möglich sein eine Weiterbildung an der Universität Rostock wahrzunehmen und sich daraus resultierend angemessen an den gesellschaftlichen Wandel anzupassen. Eine höhere Bildungsbeteiligung würde zudem den aktuellen negativen Trend stetig sinkender Fachkräfte in Deutschland entgegenwirken.

Das KOSMOS Arbeitspaket 1.5 “Mediale Infrastruktur“ sieht die Erstellung eines unterstützenden IT-Systems für diese Ziele vor. Um diese Aufgabe zu bewältigen, wurde die Fakultät der Wirtschaftsinformatik an der Universität Rostock damit beauftragt. Dabei entstand das Konzept von MeinKOSMOS. MeinKOSMOS ist ein Lernportal, welches die Kernaufgabe hat, den Nutzern während der Lernphase personalisierte Inhalte zu liefern. Durch eine Personalisierung soll zudem das Verhalten und die Gewohnheiten eines Benutzers analysiert und entsprechend darauf reagiert werden. Aber auch die Informationen einer gesamten Studien- oder Weiterbildungsgruppe soll an den Einzelnen angepasst und durch bestimmte Empfehlungen vorgeschlagen werden.

Technisch umgesetzt wurde die Personalisierung durch eine Personalisierungskomponente. Diese Komponente entstand während der Masterarbeit “Bildung von Nutzerprofi-

len aus dynamischen Nutzungsdaten im Lernportal des KOSMOS Projektes“ von Ackermann [Sam14], somit baut diese Arbeit auf der Masterarbeit auf. Neben den Verhaltensbeobachtungen der Nutzer erzeugt die Komponente Empfehlungen, die anhand der Beobachtungen von weiteren analysierten Informationen erstellt werden. Die gesamte Logik der Empfehlungskomponente wird durch ein Regelbasiertes System (Drools Rule Engine) ausgeführt. Das RBS nutzt dabei Regeln und Fakten, um Empfehlungen zu generieren.

Das Ziel dieser Arbeit ist es, die Verwaltung dieser Regeln zu vereinfachen. Es ist davon auszugehen, dass die Anzahl der Regeln in der nächsten Zeit schnell anwachsen wird. Die momentane Lösung zur Erstellung und Verwaltung der Regeln ist sehr mühsam und umständlich zu handhaben. Gespeichert und implementiert werden die Regeln in einer einzigen Regel-Datei, somit ist die Übersicht der Regeln zu umfangreich. Zudem wurde die Dateisicherheit bis heute, in der prototypischen Umsetzung, komplett außer acht gelassen. Implementiert werden die Regeln in einer eigenen Spezifikationssprache, die von Drools entwickelt worden ist. Demzufolge sollte eine Methode gefunden werden, die diese Implementierung erleichtern würde.

Ebenso soll hier ein Vorgehen erarbeitet werden, welches die Verwaltung der Personalisierungsregeln vereinfacht und verbessert. Konkret sollen die oben genannten Problemstellungen behandelt werden.

Das nachfolgende Kapitel wird die notwendigen Grundlagen erklären, die wichtig zu einem besseren Verständnis dieser Arbeit führen. In diesem Zusammenhang wird es eine Einführung in das KOSMOS Projekt und das dazugehörende MeinKOSMOS Portal geben. Zusätzlich wird der Begriff Portal geklärt. Daraus ergibt sich ein relevanter Einblick in die Idee eines regelbasierten Systems (RBS) und die damit verbundenen Technologien. Im vorletzten Abschnitt werden die wichtigen Elemente von BPMN vorgestellt. Diese Modellierungssprache wird benutzt, um die Prozesse in den Personalisierungsregeln darzustellen. Zum Abschluss des Kapitels wird die Konzeption der Personalisierungskomponente geschildert.

Um die Ziele in dieser Arbeit umzusetzen sollte zunächst ein Konzept erstellt werden, welche die folgenden Fragen umfasst:

1. Wie ist der aktuelle Stand der Personalisierungskomponente?
2. Welche Aufgaben ergeben sich bei der Verwaltung der Regeln?
3. Wie kann die Verwaltung der Regeln verbessert werden?
4. Wie kann die Optimierung umgesetzt werden?

Im Kapitel 3 werden die Fragen “1.“, “2.“ und “3.“ beantwortet. Das Kapitel erfasst vorerst die allgemeinen Anforderungen des Konzepts, anschließend wird die menschliche (Kapitel 3.2) und die technische (Kapitel 3.3) Dimension betrachtet. Die menschliche Dimension bildet die Interaktionen der Verwaltung ab. Somit fasst das Kapitel nicht nur die menschlichen Anforderungen als Konzept zusammen, auch deren/die Aufgabenfelder und Aktivitäten werden erfasst. Die technische Dimension analysiert den aktuellen Stand der Personalisierungskomponente und entwickelt ein Vorgehen zur Verknüpfung des Konzepts mit der Komponente.

Die Umsetzung des Konzepts wird im Kapitel 3 erläutert. Damit wird die “4.“ Frage geklärt. Im darauffolgendem Kapitel 5.1 wird die Umsetzung mit dem Konzept verglichen. Das Ergebnis soll zum einem zeigen, welche Kriterien erfüllt worden sind und zum anderen die Veränderungs- und eventuelle Erweiterungsbedarf darstellen. Dir kritische Betrachtung folgt darauf im Kapitel 5 .

Zum Abschluss wird im letzten Abschnitt der Arbeit, ab Kapitel 6, eine Zusammenfassung gemacht und es erfolgt ein kleiner Ausblick des Konzepts in die Zukunft.

2 Grundlagen

Dieses Kapitel dient als Verständnisgrundlage zur Ausarbeitung dieser Arbeit. Im ersten Unterkapitel 2.1 wird geklärt was das KOSMOS Projekt ist und wieso es ins Leben gerufen worden ist. Im Kapitel 2.2 zeigt einen Einblick in das MeinKOMSOS Portal sowie seine technische und organisatorische Konzeption. Anschließend befasst sich das Kapitel 2.3 mit regelbasierten Systemen (RBS). Die Grundlagen von Business Process Model and Notation (BPMN) werden im Kapitel 2.4 beschrieben. Zum Schluss wird im Kapitel 2.5 die Personalisierungskomponente vorgestellt.

2.1 KOSMOS Projekt

"Weiterbildung ist ein zentrales gesellschaftliches Thema. In jeder Lebensphase ist das organisierte Weiterlernen des Menschen für die Gestaltung seines Lebens wichtig. Die Universität Rostock hat diese Idee des Lebenslangen Lernens [...] mit dem Projekt KOSMOS bewusst aufgegriffen" [kos15]. Das KOSMOS-Projekt unterstützt den Prozess des *Lebenslangen Lernens (LLL)* eines Menschen indem es den Werdegang und die Weiterentwicklung beruflicher Kenntnisse, Fertigkeiten und Fähigkeiten fördern will. Der gesellschaftliche Wandel Deutschlands ist dabei ein wichtiger Grund für die Existenz vom KOSMOS Projekt. Laut der Bundesagentur für Arbeit soll bis 2020 das Erwerbspersonenpotenzial (Abb. 2.1) in Deutschland um 11,6% sinken und für die Jahre danach mit einer hohen Tendenz fallen [Arb11, S. 7]. Um diesen negativen Trend entgegen zu wirken, sollen im Rahmen des KOMSOS Projektes neue Studienformate erschaffen und erprobt werden, womit ein leichterer Zugang zur Weiterbildung für interessierten Menschen aller Bildungsschichten angeboten wird.

Diese neuen Studienformate eignen sich sowohl für Menschen mit akademischen Erfahrungen, sowie für Menschen mit einem Fokus auf eher praxisbezogenen Erfahrungen. Die Vielfalt an Teilnehmern mit unterschiedlichen beruflichen, sozialen und familiären

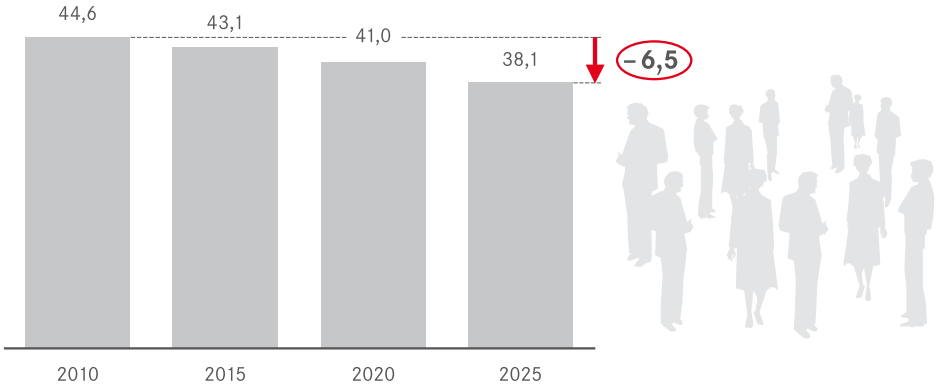


Abbildung 2.1: Erwerbspersonenpotenzial in Millionen bis 2025 [Arb11, S. 7]

Hintergründen erfordert eine hohe Flexibilität der Studienformate. *"Menschen sollen also in jeder Phase ihres Erwerbs- und Lebenszyklus universitäre Bildungsangebote nutzen und aktiv für sich aufnehmen können."* [kos15]. Eine hohe Flexibilität von Studien- und Weiterbildungsformaten ermöglicht somit eine bessere individuelle Anpassung für die Teilnehmer in den verschiedenen Lernphasen.

"Die Integration des Lebenslangen Lernens ist ohne Reorganisation der Institution Universität nicht zu leisten. Dementsprechend wird die Organisationsentwicklung mit dem Ziel verbunden, inhaltliche, strukturelle und organisatorische Rahmenbedingungen für Lebenslanges Lernen zu implementieren" [Dir13, S. 4]. Die Umsetzung von KOSMOS kann in zwei Aufgabenfeldern geteilt werden. Das erste Aufgabenfeld befasst sich mit der organisatorischen Sicht der Umsetzung. Fokussiert wird dabei unter anderem auf Fragen (nach [wet15]) wie:

- Welche Bedürfnisse bringen die Bildungsinteressenten mit?
- Wie können die Bildungsinteressenten individuell beraten werden?
- In welche Organisationsform kann die Idee der offenen Hochschule in die Struktur der Universität Rostock eingebunden werden?

Das zweite Aufgabenfeld beschäftigt sich mit der technischen Umsetzung eines langfristigen Netzwerkes [Dir13]. Im folgenden Kapitel wird dieses Aufgabenfeld erläutert.

2.2 MeinKOSMOS

Dieses Unterkapitel geht auf die technische und organisatorische Umsetzung der zuvor erläuterten Ziele ein. Das Arbeitspaket *1.5 "mediale Infrastruktur"* [Dir13] von KOSMOS hat diese Aufgabe als Ziel. *"Die zentrale Zielstellung des Arbeitspaketes ist dabei, ein Portal für den Einsatz in KOSMOS zu konzipieren und zu realisieren, das in verschiedenen Studienformaten und für unterschiedliche Zielgruppen eingesetzt werden kann"* [Dir13, S. 4]. Folglich wird hier charakterisiert was ein Portal bzw. Portalsystem ist und wieso es für die Umsetzung so gut geeignet wäre. Zudem wird hier auch MeinKOSMOS erläutert. MeinKOSMOS ist die konkrete Umsetzung eines Lernportals im KOSMOS Projekt.

2.2.1 Portalsysteme

Allgemein ist ein Portal ein webbasiertes IT-System mit einer großen Anzahl an Funktionen und Anwendungen welche miteinander in Kommunikation stehen. Dadurch werden viele Kommunikationskanäle geschaffen und bereitgestellt. Daraus ergibt sich ein einziger Einstiegspunkt für die Nutzer, um leicht an bestimmte Informationen zu kommen [Joc01]. Durch die Bereitstellung relevanter Informationen für den Nutzer bietet ein Portal somit eine hohe Personalisierung an. *"Für den Benutzer soll eine Umgebung geschaffen werden, in der er alle relevanten Informationen auf einen Blick erfassen kann oder zumindest leichten Zugriff auf sie hat"* [Joc01, S. 2].

Unterteilen lassen sich Portale in mehrere Portalarten: *B2B (business-to-business)* Portale, *B2E (business- to-employee)* Portale und *B2C (business-to-consumer)* Portale sowie *vertikale* und *horizontale* Portale. Diese Portale unterscheiden sich in der Regel in ihrem Zweck. Vertikale Portale beschäftigen sich mit einem einzigen Thema oder einem Segment, wohingegen horizontale Portale versuchen ein großes Spektrum an Themen und Segmenten abzudecken. B2B-Portale begleiten die Kommunikation zwischen Unternehmen, B2E-Portale unterstützen die Mitarbeiter während der Arbeitszeit und B2C-Portale stellen eine Kommunikation zwischen dem Unternehmen und den Kunden her [Joc01, S. 2]. An dieser Stelle wird nicht tiefer in das Thema eingegangen. Die Unterschiede im Kontext der Nutzung dieser Portaltypen ist nicht so relevant, wie die ähnliche Funktionalität dahinter. Nahezu alle Funktionen eines Portals wie Suchfunktionen, Mailsysteme, Kalender, Workflow-Systeme, Content-Management-Systeme, etc. laufen in eigenen ab-

getrennten Anwendungen, die sich individuell steuern lassen. Diese Anwendungen heißen Portlets und werden im Hauptfenster eines Portal angezeigt, für jeden Benutzer ist eine individuelle Darstellung möglich. Abbildung 2.2 zeigt eine beispielhafte Benutzeransicht in einem Portal mit einigen Portlets. Diese Portlets können nach Wunsch des Nutzers aktiviert, deaktiviert, eingestellt, angepasst, sortiert, hinzugefügt oder gelöscht werden [Joc01].

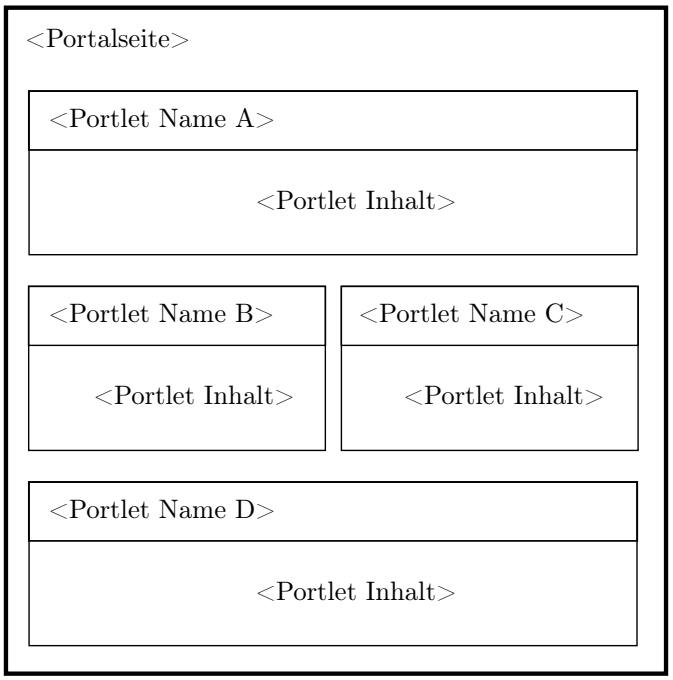


Abbildung 2.2: Interface Beispiel eines Portals mit Portlets nach Lehner [Leh14]

Es existieren geregelte Standards (z.B. JSR 268[1]) für die Struktur und Kommunikation von Portlets, wodurch eine große Anzahl an bereits verfügbaren Portlets unabhängig von Portalen, zur Verfügung steht. Teilweise können Portlets kommerziell (z.B. Liferay Marketplace[2]) erworben oder als OpenSource-Produkte genutzt und sogar weiterentwickelt werden.

[1] https://jcp.org/en/jsr/detail?id=268
[2] https://www.liferay.com/marketplace

In der Regel sind Portale Web- oder Intranet-basierte Anwendungen, was die Nutzbarkeit, Erreichbarkeit und Kompatibilität maximiert. Die technische Architektur eines Portal kann sich von Portal zu Portal unterscheiden. Dabei kommen unterschiedliche Server als Basis zum Einsatz (z.B. Apache Tomcat[3]) oder es werden andere Datenbanken genutzt. Um die Vielfalt einzugrenzen wird hier nur die allgemeine technische Struktur eines Portals erläutert. Die folgende Abbildung 2.3 veranschaulicht die Struktur [Joc01, S. 3].

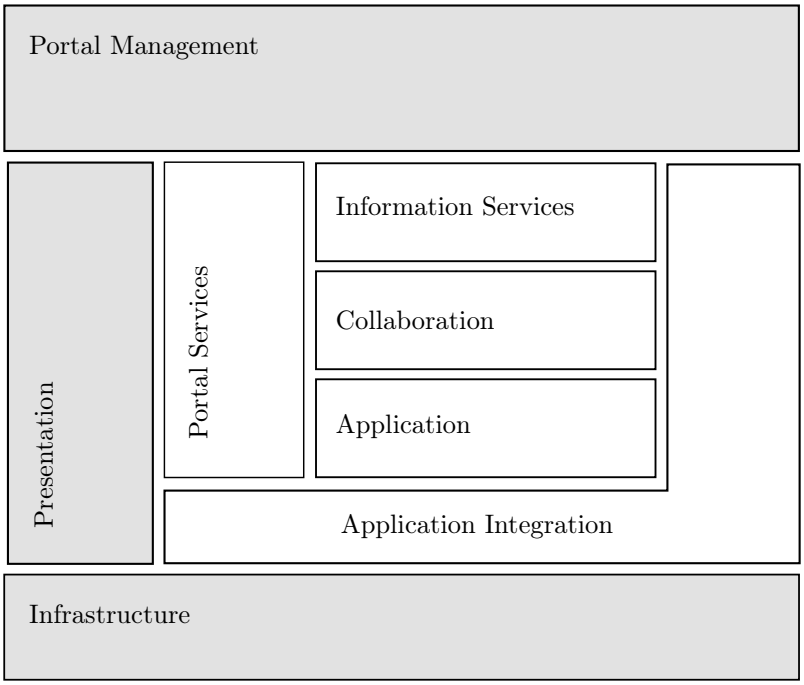


Abbildung 2.3: Portalbestandteile nach Rütschlin [Joc01]

Rütschlin unterteilt ein Portal anhand von Funktionalitäten innerhalb eines Portal. Somit haben die Bestandteile abgegrenzte Aufgaben. *Infrastructure* legt das technische Fundament (z.B. Server, Netzstruktur, Hardware, OS) für das Portal. *Presentation* kümmert sich um die Darstellung sämtlicher Elemente und Funktionen des Portals. *Portal Management* verwaltet das gesamte Portal, sowohl im Front-End als auch im Back-End. Dazu gehören auch Aufgaben wie Benutzerverwaltung oder Monitoring. *Portal Services* sind typischen Portalfunktionen wie Personalisierung oder Suche im Portal. *Information*

[3] http://tomcat.apache.org

Services sind klassische diGitale Informationen und Daten wie Nachrichten, Dokumentationen, oder Webinhalte. *Collaboration* sind Kalender, E-Mails, Groupware oder auch Workflow-Systeme. *Application* sind Funktionen und Anwendungen, die herkömmlich oder teilweise webbasiert sind. Schließlich kümmert sich *Application Integration* um eine Integration alle Bestandteile [Joc01, S. 3].

2.2.2 Technische Konzeption

MeinKOSMOS wurde dafür entwickelt die im Kapitel 2.1 vorgestellten Ziele und Anforderungen in einen langfristigen Netzwerk zu unterstützten. Ein Hauptkriterium von KOSMOS fordert eine personalisierte Begleitung während der Weiterbildung. Ein Portal, mit den im Kapitel 2.2.1 vorgestellten Eigenschaften, bietet sich besonders für diese Aufgabe an.

"Aufgrund des vorhandenen Marktes und der beschränkten Laufzeit des Projektes ist aber von einer kompletten Eigenentwicklung abzusehen, da diese im Umfang des angestrebten Portals nicht möglich wäre" [Dir13, S. 5]. Nach einer umfassenden Marktforschung wurde die OpenSource Software Liferay[4] als Lösung für das Lernportal ausgewählt. Einerseits biete Liferay eine umfassende Anpassung des Portal an eigene Wünsche und Anforderungen. Andererseits ist das OpenSource Projekt offen für eigene Erweiterungen und Veränderungen. Die Grundinstallation von Liferay läuft auf einem Java-Server (Apache Tomcat), diese liefert automatisch eine große Anzahl an Portlets. Dadurch werden grundlegende sowohl sehr fortgeschrittene Funktionen durch die Portlets abgedeckt. Die im Kapitel 2.2.1 erwähnten Standards für die Portlets eröffnen einen potenziell großen Markt für diese portalunabhängigen Anwendungen. Liferay selbst betreibt einen eigenen Store (Liferay Marktplace) [inc15].

Der Implementierung neuer Anwendungen für Liferay steht also nichts im Wege. Im Gegenteil, die Entwickler stellen einen *Software Development Kit* (SDK) oder auch *Integrated development environment* (IDE) zur Verfügung. Das Arbeiten mit dem SDK und IDE erleichtert durch zahlreiche Hilfsfunktionen und Tools die Entwicklung eines Portlets. Es können z.b. mit Hilfe einer einzigen XML-Datei mehrere Javaklassen erstellt

[4] https://www.liferay.com

werden, die als Schnittstellen zwischen dem Portlet und der Datenbank dienen. Es besteht auch die Möglichkeit mit einigen Klicks ganze Portlets nach JSR 268 Standard zu erstellen. Diese sind zwar blanko, es erleichtert dennoch das Anlegen der Struktur. Abgesehen davon werden die erstellten Projekte für die Integration im Liferay Portal perfekt vorbereitet [Bri11, S. 295-296].

2.2.3 Organisatorische Konzeption

Das Lernportal konzentriert sich auf die individuelle Begleitung und Unterstützung während der Weiterbildung. Zu diesem Zweck sind mehrere Informationsquellen nötig. MeinKOSMOS kann als einziger Einstiegspunkt genutzt werden. So ein *"Single Point of Entry"* [Dir13] kann auf mehrere IT-Systeme, Dienstleistungen und Anwendungen zugreifen und diese somit miteinander verknüpfen. Die Universität Rostock betreibt mehrere Systeme, die den Studenten während des Studiums als Hilfe zur Verfügung stehen, wie z.B. ILIAS[5] E-Learning-Plattform, StudIP[6] Learning-Managementsystem oder die Vorlesungsdatenbank[7]. Diese Systeme können im MeinKOSMOS durch den *"Single Point of Entry"* kommunizieren, ohne dass ein explizites Einloggen in den einzelnen Systemen notwendig ist [Sam14, S. 19]. Diese Verknüpfung ist für die Teilnehmer nicht sichtbar, spiegelt sich aber in einer persönlich zugeschnittenen Informationsdarstellung wieder. Dadurch können auch nutzerbezogene Informationen zwischen den Systemen für eine weitere Bearbeitung ausgetauscht werden.

Eine einseitig ermittelnde Informationsdarstellung ist jedoch nicht das Optimum. Die Erhebung personenbezogener Informationen in Bezug auf die Bildungsbiographie der Nutzer ist daher notwendig. Die Teilnehmer haben die Möglichkeit ihr Profil im Lernportal zu erstellen und zu ergänzen. *"Zur individualisierten Informationsbereitstellung werden verschiedene Informationen über die Nutzer, die während des Lernprozesses das Portal betreten, benötigt. Entsprechend soll das Portal ein Profil der Lernenden führen, und entsprechende Anpassungen auf Basis erkannter Aktionen automatisch durchführen"* [Dir13, S. 7]. Durch das Nutzerprofil entsteht also eine zentrale Stelle für die Speicherung der Daten, welches auch als Schnittstelle für weitere Systeme dienen kann. Wie

[5] https://ilias.campusmv.de
[6] https://studip.uni-rostock.de
[7] https://lsf.uni-rostock.de

die Anforderungen an das Nutzerprofil auszusehen haben wird näher im Abschnitt 2.5 beschrieben.

Geprägt und verändert wird das Nutzerprofil durch *Learning Analytics (LA)*. Diese Methode misst und beobachtet die Aktivitäten und das Verhalten eines Nutzers während des Lernens mithilfe digitaler Medien. Dabei werden verschiedene Nutzertypen ermittelt. Diese Typen können dann bestimmten Nutzerprofilen zugeordnet werden, um Vorschläge in Bezug auf Inhalte oder Darstellung anzubieten [Dir13]. So können z.B. öfter genutzte Anwendungen besser positioniert oder kaum genutzte Anwendungen ausgeblendet werden. Andere Betrachtungsweisen können sich mit dem Lernkontext des Nutzers befassen. Im Lernprozess entstehende Schwierigkeiten können automatisch oder manuell erkannt und ggf. and den Dozenten weiter geleitet werden [Sam14, S. 10]. Die Realisierung einer Komponente für LA wird ebenfalls im Kapitel 2.5 erläutert.

"Neben den reinen Selbstlernphasen soll das Portal auch die Gruppenarbeit stärker in den Fokus rücken, da diese auch von den Teilnehmern als besonders wertvoll an einer Weiterbildung erachtet wird..." [Dir13, S. 9]. Gruppenarbeit fördert zudem das Lernen innerhalb der Gruppe (Peer-Learning). Durch das Peer-Learning können die Defizite im Bezug auf die Bildungsbiographie kompensiert werden. Dieses Vorgehen steigert die Kommunikation zwischen den Studenten und hilft beim Aufbau von Netzwerken. Die Kommunikation zwischen Studenten und Tutoren soll ebenfalls gesteigert bzw. gefördert werden [Dir13].

Abbildung 2.4 zeigt die aktuelle Version (Stand März 2015) von MeinKOSMOS mit einigen bereits personalisieren Anwendungen. Erreichbar ist MeinKOSMOS momentan über `kosmos.informatik.uni-rostock.de` und befindet sich kurz vor der nächsten Phase, in der das Portal in den Erstbetrieb genommen wird.

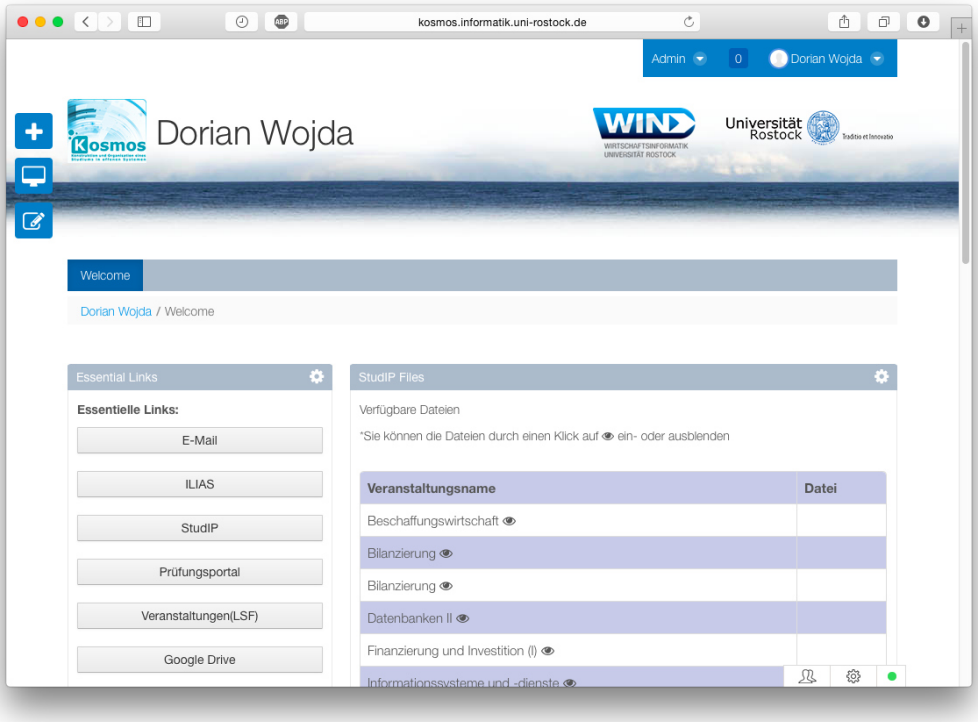


Abbildung 2.4: Screenshot von MeinKOSMOS (Stand März 2015)

2.3 Regelbasierte Systeme

Laut Haynes-Roth können Regelbasierte Systeme als Experten-Systeme angesehen werden, die praktisches menschliches Wissen speichern [HR85, S. 921]. Das gespeicherte Wissen wird zur Findung von Lösungen für praktische Probleme genutzt. Daher kommt der Begriff *Experte* zum Einsatz. Das System kombiniert gespeicherte Regeln mit bestimmten Fakten, um ein Ergebnis zu liefern. Die Abbildung 2.5 stellt die Beziehung der einzelnen Bestandteile eines RBS dar. Die Wissensbank (*Knowledge base*) beinhaltet

Regeln und Fakten, daher kann die Wissensbank in Fakten-Speicher (*Fact memory*) und Regel-Speicher (*Rule memory*) eingeteilt werden. Regeln-Speicher speichert die gesamten Regeln, die in einer bestimmten Regelsprache verfasst sind. Fakten-Speicher stellt während der Verarbeitung benötigte Fakten bereit. Es können dennoch weitere Fakten außerhalb des Speichers eingelesen werden [HR85].

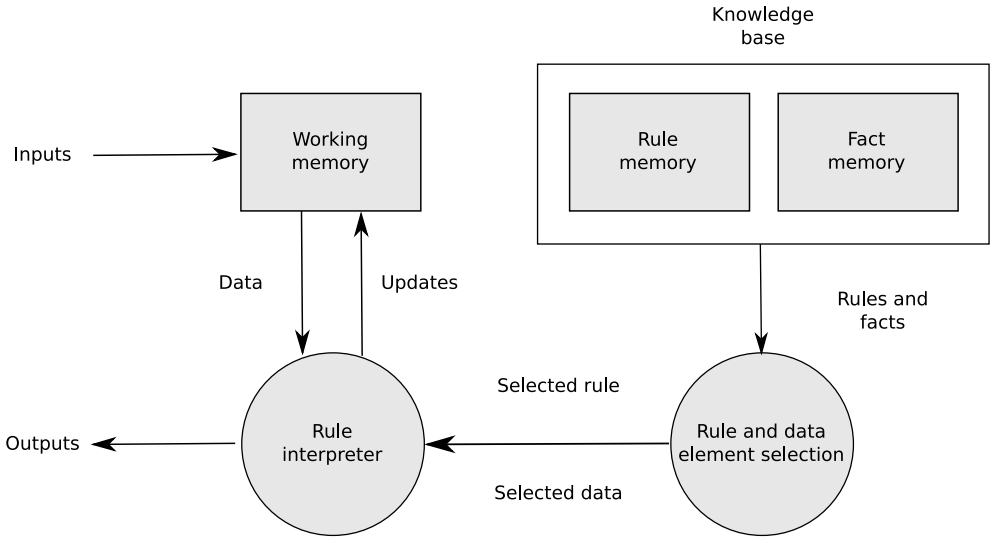


Abbildung 2.5: RBS Bestandteile nach Rütschlin [Joc01]

Neben der Wissensbank wird eine/ein *Inference Engine (Rule intepreter)* benötigt, um die Regeln zu verstehen und sie zusammen mit den Fakten verarbeiten zu können. *"Die Inference Engine ist die wichtigste Komponente eines regelbasierten Systems"* [BRS11, S. 195]. Die komplette Verarbeitung findet in einer *Working Memeory* (WM) statt und läuft in drei Phasen ab. Die erste Phase ist die *Match-Phase*. In dieser Phase werden alle passenden und erfüllten Regeln und Fakten in WM gemerkt. Treten dabei mehrere Varianten einer Regel durch verschiedene Kombinationen von Fakten auf, so werden diese Varianten in einem *Conflict-Set* oder auch *Agenda* genannt, gespeichert. Die nächste Phase ist die *Select-Phase*. Hier wird eine Ausführungsreihenfolge bzw. die Priorität der Regeln ermittelt. Diese Reihenfolge wird durch die eingeholten Fakten beeinflusst. In der letzten *Act-Phase* werden die Regeln ausgeführt. Unter bestimmten Umständen kann die Ausführung neue Fakten in den Vorgang einbeziehen. In diesem Fall kann die *Match-*

Phase erneut starten [HR85].

Unabhängig von der Regelsprache werden die Regeln in WENN- und DANN-Segmenten formuliert. Auch wenn die Regelstruktur einfach erscheint, können durch Kombinationen mehrerer Regeln und Segmenten komplexe Gebilde geschaffen werden. Grundsätzlich kann zwischen Produktions- und Folgerungsregeln unterschieden werden. Folgerungsregeln liefern nur ein Ergebnis, z.b. *true* oder *false*. Produktionsregeln führen zu einer weiteren Aktion hin. [Sam14]

In der Verarbeitung der Regeln gibt es ebenfalls grundlegende Unterschiede. Es kann die Rückwärts- oder Vorwärtsverkettung genutzt werden. In der Rückwärtsverkettung wird versucht ein Endstatus oder Ergebnis zu erreichen, indem die Hypothese bewiesen oder endgültig abgelehnt wird. *"Bei der Rückwärtsverkettung wird der Benutzer aufgefordert, Beobachtungen zu machen, Tests durchzuführen oder Werte einzugeben. Sind im voraus alle Fakten bekannt, ist diese Vorgehensweise weniger sinnvoll"* [BRS11, S. 214]. Die Vorwärtsverkettung ist eine Aneinanderreihung von Regeln. Dabei schafft eine Regel neue Fakten wodurch andere Regeln ausgeführt werden können. Also leitet an dieser Stelle die *Act-Phase* einen erneuten Start der Match-Phase. Die Vorwärtsverkettung endet, wenn keine neuen Fakten mehr abgeleitet werden können oder ein Zielfaktum generiert wurde [BRS11, S. 214].

RBS ermöglicht zudem eine Trennung zwischen Logik und dem Rest des Systems. Dies erleichtert die Verwaltung während der Betriebszeit enorm. Es können z.b. Wartungsarbeiten oder Updates durchgeführt werden ohne Einschränkungen in der Systemnutzung. Zudem können speziell für das Management gerichtete Anwendungen die Verwaltung der Regeln unterstützen. Ein bestimmtes Tool könnte z.B. aus der reinen Rule-Sprache eine visuelle Darstellung erzeugen, welche wiederum leichter zu verstehen und zugänglicher für mehr Bearbeitergruppen wären. Eine Verbindung mehrerer Anwendungen und Tools für das Management befindet sich schon in so genannten Business Rules Management Systemen (z.B. Drools[6]) oder Business Rules Management Suites (z.B. jBPM[7]).

[6] http://www.drools.org
[7] http://www.jbpm.org

2.4 Grundlagen von BPMN

Business Process Model and Notation (BPMN) ist eine grafische Spezifikation zur Modellierung von Geschäftsprozessen aus dem Bereich Geschäftsprozessmanagement (BPM). *"The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes"* [Whi04, S. 1]. Stephen White, der Hauptersteller der Spezifikation wollte auf diese Art Geschäftsprozesse problemlos für jedermann verständlich machen. Es sollten auch Menschen ohne technische Hintergründe oder Programmierkenntnissen die Modelle verstehen. Somit sollte eine Lücke zwischen der Implementierung und grafischen Modellierung von Geschäftsprozessen geschaffen werden [Whi04, S. 1]. Die erste Version von BPMN wurde offiziell Anfang 2004 von Object Management Group (OMG) veröffentlicht. Durch die sehr positive Resonanz aus der Wirtschaft wurde seitdem weiter an der Spezifikation gearbeitet [Koc11, S. 7]. Im Januar 2011 konnte OMG die nächste Version von BPMN (BPMN 2.0) veröffentlichen [OMG11].

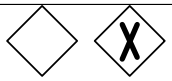
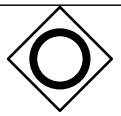
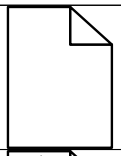
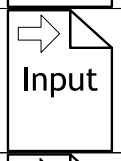
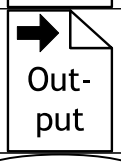
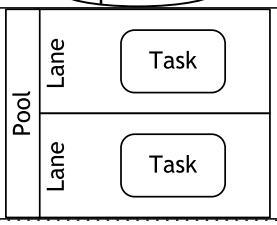
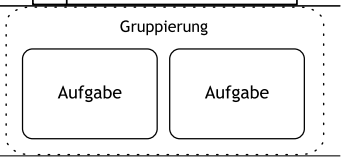
Hauptgegenstand von BPMN ist *Business Process Diagram (BPD)*, ein grafisches Modell welches an das Flussdiagramm aus *Unified Modeling Language (UML)* erinnert [Sch08, S. 17]. BPD bedient sich einer Reihe von Elementen aus verschiedenen Elementgruppen. BPMN 2.0 verfügt über mehr als 100 Elemente [OMG11]. Experten aus dem BPM Bereich kritisieren die breite Palette an Elementen, ihrer Meinung nach gestaltet sich die Modellierung dadurch unnötig kompliziert. Eine Eingrenzung und Unterteilung auf die wesentlichen Elemente ist jedoch von Prozess zu Prozess anders, dennoch existieren einige Elemente, die häufiger als andere benutzt werden [Fre]. Die Basiselemente lassen sich in vier Gruppen einteilen:

- **Flussobjekte** mit Aktivitäten, Entscheidungen (Gateways), Ereignissen (Events) und Verbindungselemente (Sequence Flow Connections)
- **Beteiliger** mit Pools und Lanes
- **Daten** wie Datenobjekt, Dateneingabe, Datenausgabe und Speicher
- **Artefakte** wie Gruppierungen und Anmerkungen

Flussobjekte sind Grundbestandteile eines Geschäftsprozesses und werden daher am häufigsten verwendet. Aktivitäten sind Aufgaben die im Laufe eines Prozesses ausgeführt werden, dabei gibt es mehrere Aktivitätstypen, wie z.b. Senden, Empfangen, maschinelle Scripte, Services oder Geschäftsregeln. Die Abfolge der Aktivitäten wird durch Entscheidungen (Gateways) und deren Verbindungen bestimmt. Während der Ausführung treten verschiedene Ereignisse auf. Neben dem Start- und Endereignis, die fast in jedem Modell vorhanden sind, gibt es standardisierte Ergebnisse wie z.b. Nachrichten, Timer, Abbrüche und Fehler. Diese besonderen Ereignisse häufen sich oftmals bei Transaktionsprozessen an. Verwendete Daten haben eigene Datenobjekte mit Ein- und Ausgaben sowie Speichern. Besonders in der IT ist eine gesonderte Betrachtung dieser Objekte sinnvoll. Ablaufsunabhängige Elemente (Artefakte) wie Anmerkungen oder Gruppierungen können auch verwendet werden. Sind mehrere Beteiligte im Prozess involviert, werden diese in einem Pool dargestellt, wobei jeder Beteiligte seinen eigenen abgegrenzten Bereich (Lane) hat [Sch08, S. 17]. Folgende Tabelle zeigt einen Überblick der Basiselemente.

Tabelle 2.1: Basiselemente bei BPMN 2.0 nach [bpm15]

Bezeichnung	**Beschreibung**	**Notation**
Aufgabe	Aufgabe ist eine Arbeitseinheit	
Transaktion	Transaktion sind gruppierte Aufgaben, die logisch zusammenhängen	
Geschäftsregel	Eine Aktivität die als Geschäftsregel vorliegt	
Startereignis	Symbolisiert den Beginn eines Prozesses	
Endereignis	Symbolisiert den Abschluss eines Prozesses	
Terminierung	Sofortiger Abbruch eines Prozesses	
Sequenzfluss	Definiert die Abfolge von Aufgaben	

Bedingter Fluss	Sequenzfluss der eine Bedingung enthält	
Exklusives Gateway	Eine XOR-Verzweigung, es kann nur eine Abfolge gewählt werden	
Inklusives Gateway	Eine OR-Verzweigung, abhängig von den Bedingungen kann/können eine oder mehrere Abfolge/Abfolgen gewählt werden	
Paralleles Gateway	Es werden alle Verzweigung durchlaufen, beim Zusammenführung wird auf alle Abläufe gewartet.	
Datenobjekt	Repräsentieren Informationen wie, Dokumente, E-Mails oder Datensätze	
Dateninput	Externer Input für den ganzen Prozess, der von einer Aktivität gelesen wird	
Datenoutput	Variable, die als Ergebnis eines ganzen Prozesses erzeugt wird	
Datenspeicher	Ort, auf den der Prozess lesend und schreibend zugreifen kann	
Pools mit Lanes	Getrennte Betrachtung von Beteiligten im Prozessablauf	
Gruppierung	Gruppierung von Prozessen	

Anmerkungen	Sind Notizen, die keinen Einfluss auf den Prozess haben, sie dienen nur zur besseren Verständlichkeit	Text-Anmerkung Aufgabe

Die obere Tabelle bietet einen Einblick in die grundsätzlich wichtigsten Elemente von BPMN 2.0 an. Zu den vorgestellten Aufgaben können zahlreiche Aufgaben-Typen und Markierungen gezählt werden. Neue Modelltypen wie Konversationen stellen zusammenhängende Nachrichtenflüsse dar. Der Nachrichtenaustausch zwischen mehreren Beteiligten kann in Choreographien abgebildet werden. Zudem kommen verschiedene Ereignistypen hinzu, was man aus der Abbildung [bpmn-plakat] entnehmen kann. Eine komplette Liste mit Elementen und deren Beschreibung sowie Notationen kann aus der offiziellen BPMN 2.0 Dokumentation [OMG11] entnommen werden. Ein komplexes Beispiel für einen BPMN 2.0 Geschäftsprozess zeigt die Abbildung 2.6.

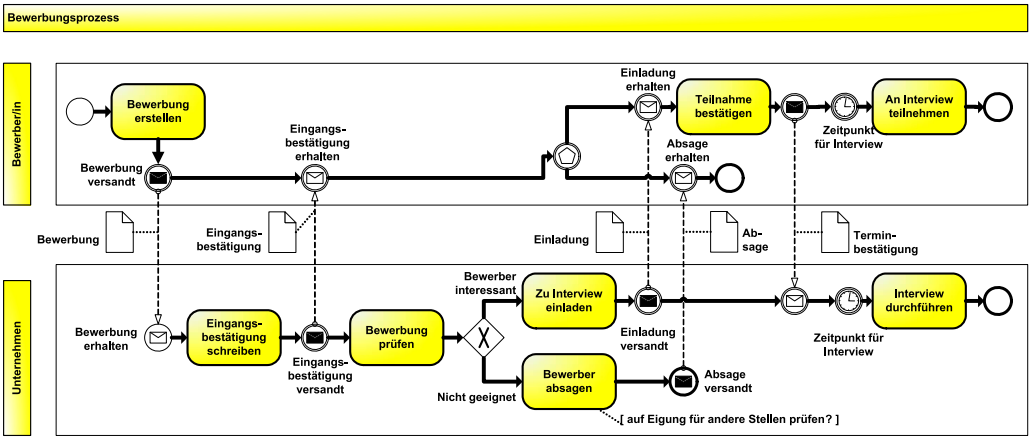
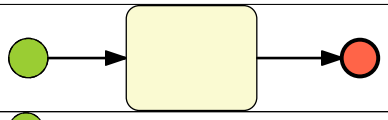
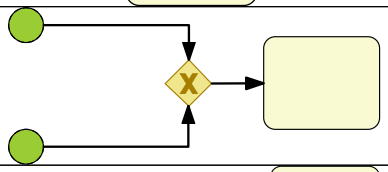
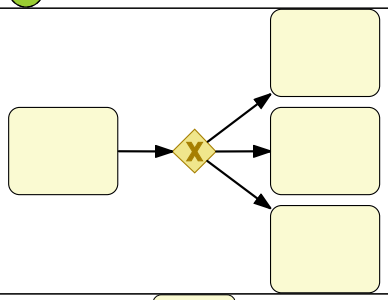
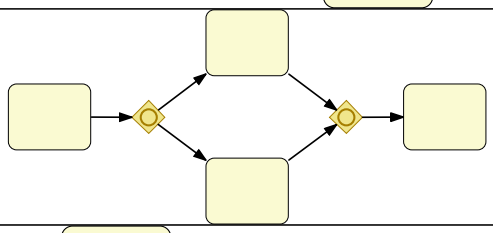
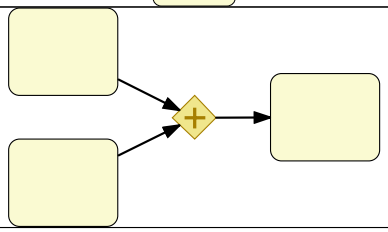


Abbildung 2.6: Komplexes BPMN 2.0 Beispiel nach Kocian [Koc11, S. 16]

Neben den vorgegebenen Notationen hat die Spezifikation auch eine Semantik und eine Syntax, die eingehalten werden soll. Folgende Tabelle zeigt einige Notationsvorschriffen [OMG11].

Tabelle 2.2: Notationsvorschriften bei BPMN 2.0

Beschreibung	**Notation**
Prozess mit einem Start- und Endereignis	
Mögliche Darstellung multipler Startereignisse	
Prozess mit einem exklusiven Gateway	
Prozess mit einem inklusiven Gateway	
Prozess mit einem parallelen Gateway	

2.5 Personalisierungskomponente

Schwerpunkt von MeinKOSMOS liegt darin den Nutzern einen für sie angepassten Content zu liefern. Dabei sollen Empfehlungen ermittelt und angeboten werden. Der unterstützende Nutzen von Empfehlungen wurde im Kapitel 2.2.3 genauer besprochen. An dieser Stelle folgt eine Einführung in die konkrete Umsetzung einer Komponente, die im MeinKOSMOS diese Aufgabe übernimmt.

Die Personalisierungskomponente steuert, verwaltet und erzeugt persönliche Empfehlungen innerhalb von MeinKOMSOS. Diese Komponente besteht aus zwei Bestandteilen; Beobachterkomponente und Empfehlungskomponente. Dazu zählt noch als Bindeglied ein Nutzerprofil. Dieses Nutzerprofil ist die zentrale Speicherstelle für statische sowie dynamische Informationen über den Nutzer selbst. Des Weiteren sind im Profil auch Objekte vorhanden, mit denen der Nutzer während der Studien- oder Weiterbildungszeit oft interagiert wie z.B Studiengang, Seminararbeit oder Gruppe. Nutzer (Abbildung 2.8) sind dabei Teilnehmer, Dozenten oder Organisatoren [Sam14, Anhang IX].

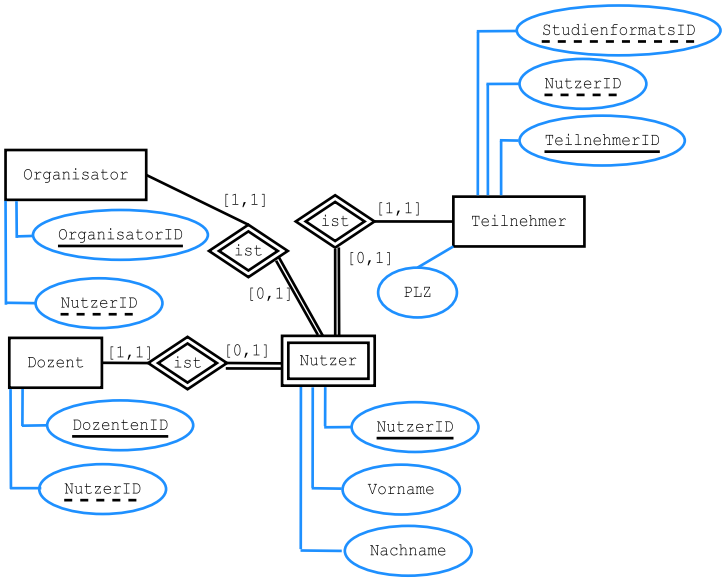


Abbildung 2.7: Nutzerrollen in MeinKOSMOS [Sam14, S. 28]

Implementiert ist das Profil als eine MySQL-Datenbank, die im MeinKOMSOS angelegt worden ist. Somit gibt das Profil vor, welche Informationen gesammelt werden sollen, um eine Personalisierung realisieren zu können. Neben den statischen Informationen wie

Name, *Vorname*, *Studienformat* oder *DozentID* werden auch dynamische Informationen wie *Assignment*, *Genutzte_ Applikation* oder *Präsenzphase* gespeichert [Sam14, Anhang IX].

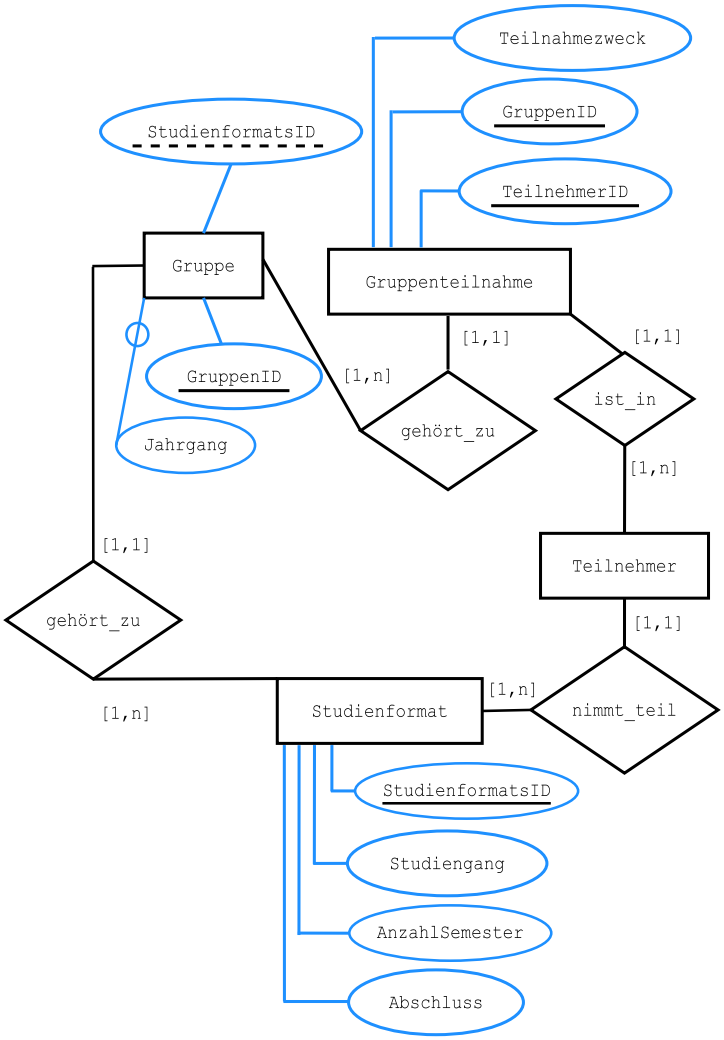


Abbildung 2.8: Weitere Objekte im Nutzerprofil [Sam14, S. 29]

Dynamische Informationen ändern sich mit der Zeit oder mit den Aktivitäten des Nutzers. Um diese Aktivitäten innerhalb von MeinKOMSOS zu erfassen, existiert die Beobachterkomponente. Diese Komponente funktioniert nach dem Prinzip von *Observer Patterns*, dabei werden einem Subjekt mehrere Observer zugewiesen, welche bestimmte

Zustände des Subjekts beobachten und bei einer Änderung informiert werden. Umgesetzt wird dies durch verschiedene Javaklassen. Abbildung 2.9 zeigt die prototypische Architektur von *Observer Patterns*. Dabei wird einem *Subjekt* ein oder mehrerer *Observer* zugewiesen. Bei Veränderungen der Zustände im Subjekt werden die einzelnen Observer benachrichtigt.

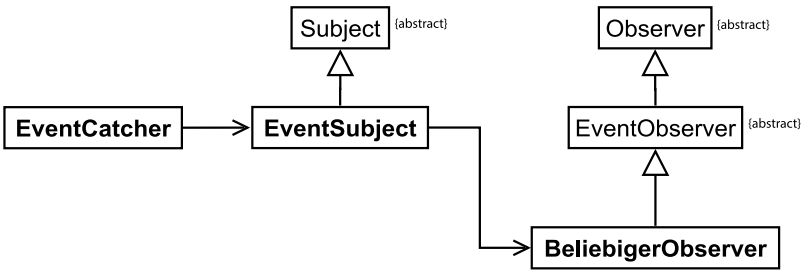


Abbildung 2.9: Architektur auf Basis des Observer Patterns [Sam14, S. 41]

Neben den *Observer Patterns* zählt noch das Prinzip der *Events* und *Actions* von Portlets. Jedem Event, also jeder Aktivität im Portal, folgt (*PostAction*) oder geht voran (*PreAction*) eine *Action* aus, welche abgefangen und ausgelesen werden kann, um auf die Aktivitäten von Nutzern zurück zu führen. Die Architektur des Prototyps der Beobachterkomponente vereint die zwei vorgestellten Prinzipien [Sam14, S. 41]. Ein tieferer Einblick in die Beobachterkomponente ist für diese Bachelorarbeit nicht notwendig.

Der zweite Bestandteil ist die Empfehlungskomponente. Sie dient dazu aus dem Nutzerprofil persönliche Empfehlungen für den Teilnehmer zu erstellen. Basis dafür ist das RBS *Drools*. Drools ist ein *Java Rule Management System* welches ermöglicht Geschäftsprozesse (GP) zu steuern und verwalten. Vor der Implementierung wurden die GP und Regeln in BPMN modelliert. Implementiert sind diese in Java oder in der *MVEL* Spezifikationssprache. MVEL wird dann verwendet, wenn die Regel Listenobjekte verarbeiten soll, anderenfalls kann Java genutzt werden. Die Regeln befinden sich in einer Regeldatei außerhalb der restlichen Klassendateien, und können optimal während der Laufzeit editiert werden. Für eine beispielhafte Veranschaulichung wird hier kurz eine BPMN Regel (Abb. 2.10) vorgestellt.

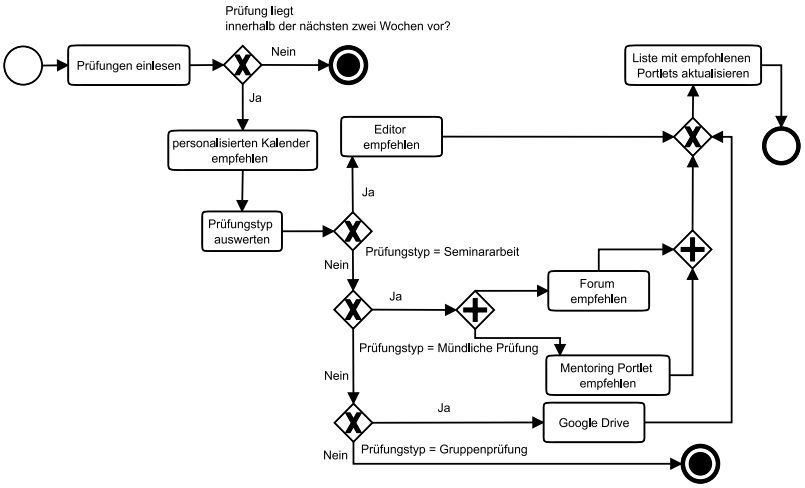


Abbildung 2.10: BPMN Regel "Prüfung"[Sam14, S. 53]

Die Regel soll eine Empfehlung für einen persönlichen Kalender erzeugen, sofern der Nutzer eine Prüfung oder mehrere Prüfungen in den kommenden 14 Tagen schreiben wird. Zudem soll angemessen auf die Prüfungsart reagiert werden. Bei einer bevorstehenden Seminararbeit kündigt ein Editor-Portlet eine Empfehlung für die Bearbeitung an. Wird eine Gruppenarbeit stattfinden, dann soll das Google Drive-Portlet für einen leichteren Austausch von Dateien angeboten werden. Steht eine mündliche Prüfung an, solle das Foren-Portlet für bessere Anknüpfungen an Diskussionsrunden und Mentoring-Programme entsprechende Portlets empfehlen [Sam14, S. 52].

In der ersten Linie beschäftigt sich die Empfehlungskomponente mit automatisierten Vorschlägen zur Verwendung von bestimmten Portlets. Es besteht zudem die Variante manuell erzeugter Empfehlungen (*Assignments*). Assignments werden durch den Dozenten oder den Lehrbeauftragten erstellt und sind bestimmten Studiengängen oder Weiterbildungsangeboten zugeordnet. Zudem haben die Assignments eine Start und Endzeitraum, in dem sie gültig sind. Neben den vorgestellten Empfehlungsobjekten Portlet und Assignments können laut Ackermann beliebige Objekte als Empfehlung, z.b. Sichern oder Dokumente dazu implementiert werden [Sam14, S. 58].

Technisch umgesetzt und implementiert ist die Empfehlungskomponente auf Basis von verschiedenen Javaklassen (Abb. 2.11). Diese Objekte bzw. Klassen werden im Regelbetrieb ausgewertet und verarbeitet. Eine zentrale Rolle spielt dabei das Teilnehmerobjekt,

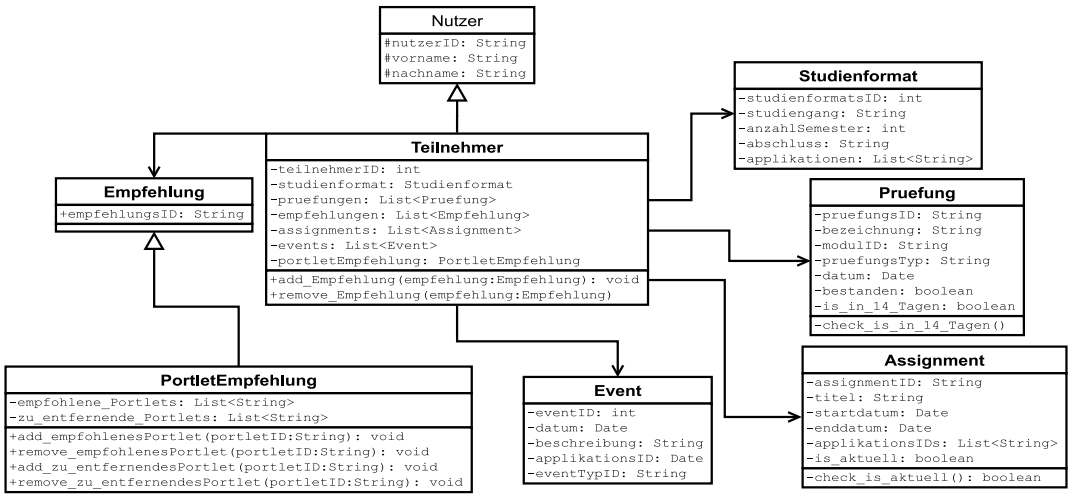


Abbildung 2.11: Teilnehmer Klassen Objekt [Sam14, S. 57]

welches eine Verknüpfung aller notwendigen Klassen darstellt. Ferner spiegeln die Klassen das Nutzerprofil wider und werden als Fakten während der Verarbeitung der Regeln genutzt [Sam14, S. 56].

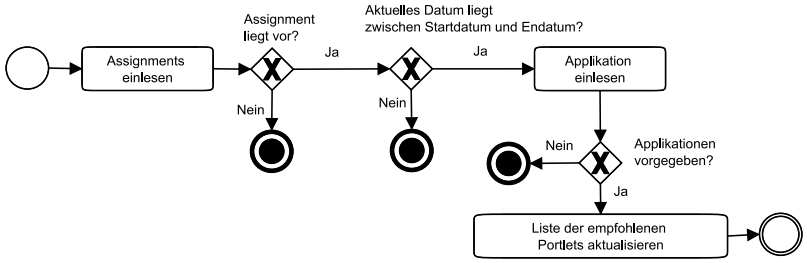


Abbildung 2.12: BPMN Regel "Vorgegebene Applikation"[Sam14, S. 54]

Um einen Einblick in den Regelbetrieb zu bekommen, wird hier kurz eine Regel *"Vorgegebene Applikation"* (Abb. 2.12) vorgestellt. Diese Regel soll prüfen, ob für den Teilnehmer Assignments mit vorgegebenen Portlets vorhanden sind. Liegt ein zeitlich gültiges Assignment vor und der Teilnehmer hat das Portlet noch nicht aktiviert, so soll die Liste mit empfohlenen Portlets aktualisiert werden [Sam14, S. 53].

Beim Regelbetrieb wird das Objekt der Klasse Teilnehmer als Fakt in WM übergeben und auf die Anzahl der Assignments geprüft. Sofern ein oder mehr Assignments vorhan-

den sind, wird eine weitere Regel (***bestimme_ AssignmentApplikationen***) gefeuert, die weitere Schritte unternimmt. Einen Teil der implementierten Regel zeigt das folgende Listing 2.1.

```
1  rule "vorgegebene_Applikation"
2  dialect "mvel"
3  when
4    $p : Teilnehmer(assignments.size() > 0);
5  then
6    $p = bestimme_AssignmentApplikationen($p);
7  end
```

Listing 2.1: Regel "vorgegebene Applikation"nach Ackermann [Sam14, S. 64]

Für die Schnittstelle zwischen Drools und der Portalseiten bzw. MeinKOSMOS Portal wurde ein Support-Portlet entwickelt. Dieses Portlet verfügt über die Funktionalität der Entfernung und der Hinzufügung von Portlets auf der privaten Seite der Teilnehmer. Dies kann automatisch im Hintergrund oder manuell über das Interface (Abb. 2.13) des Portlets geschehen [Sam14, S. 60].

Abbildung 2.13: Support-Portlet Interface aus MeinKOSMOS

Das Support-Portlet besteht aus drei Bestandteilen: *MainController*, der die Objekte der Klasse Teilnehmer beinhaltet, *PortletController*, der die Funktionen zur Steuerung der Portlets beinhaltet und einer JSP, die unter anderem für das Interface zuständig ist [Sam14, S. 60-61].

Abbildung 2.14 zeigt die Attribute und Methoden von MainController. Wie oben beschrieben beinhaltet diese Klasse den Teilnehmer als Objekt. Das Attribut *shouldreload* hält fest, ob eine Aktualisierung der Portalseite vorgesehen ist. Das Listenobjekt ***port-***

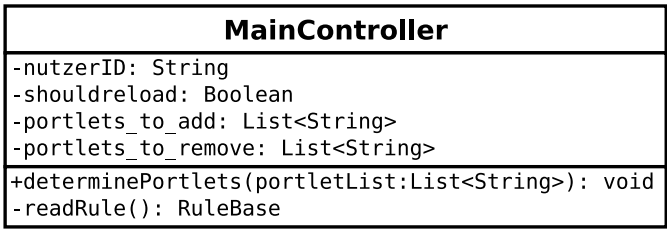


Abbildung 2.14: MainController Klasse [Sam14, S. 61]

lets_ to_ add speichert die IDs der Portlets, die hinzugefügt werden sollen. Dagegen speichert ***portlets_ to_ remove*** die IDs, die entfernt werden sollen. Die Funktion ***determinePortlets*** dient als Schnittstelle während des Regelbetriebs für die Ermittlung der bereits vorhandenen Portlets auf der Portalseite des Nutzers. Die Funktion ***readRule*** importiert die Regeln aus einer oder mehreren externen Regeldateien und erstellt laut Listing 2.2 auf deren Basis eine Drools Rulebase [Sam14, S. 61].

```
1 RuleBase ruleBase = readRule();
2 WorkingMemory workingMemory = ruleBase.newStatefulSession();
3 workingMemory.insert(teilnehmer);
4 workingMemory.fireAllRules();
```

Listing 2.2: Auslösen der Regeln nach Ackermann [Sam14, S. 62]

In der Rulebase wird ein WM erstellt und mit dem Teilnehmer Objekt als Fakt gefüllt. Anschließend leitet die Methode ***fireAllRules*** die Ausführung der Regeln ein. Konkret, wie im implementierten Beispiel einer Regel (Listing 2.1), wird die Regel *"vorgebene_ Applikation"* in MVEL verfasst. Im *when*-Teil (Left-Hand-Side) der Regel wird eine Variable ***$p*** erzeugt - unter der Bedingung, dass die Anzahl von Assignments größer Null ist. Sofern der *when*-Teil erfüllt ist, kommt es im *then*-Teil (Right-Hand-Side) zur Übergabe des Teilnehmer Objektes an weitere Funktionen bzw. Regeln [Sam14, S. 62].

3 Konzept

Dieses Kapitel befasst sich mit der Konzeption eines Vorgehens zur Verwaltung von Personalisierungsregeln. Das Konzept wird in zwei Dimensionen geteilt, welche in diesem Kapitel erarbeitet werden. Die menschliche Dimension beschäftigt sich mit der Verbindung zwischen Nutzer und System. Die technische Dimension behandelt hingegen die Integration des Konzeptes in das System von MeinKOSMOS. Auch eine Definition von allgemeinen Anforderungen findet hier statt. Im letzten Unterkapitel 3.4 wird eine zusammenfassende Betrachtung des Konzeptes gemacht, um eine zentrale Übersicht der Anforderungen für spätere Vergleiche zu schaffen.

3.1 Allgemeine Anforderungen

Als Erstes werden allgemeine Anforderungen für das Konzept ermittelt, die stets gelten sollen. Dies umfasst den Rahmen, in welchen sich die Elemente, Aktivitäten und Bestandteile befinden sollen.

Im Konzept sollen alle Regeln in einer Form implementiert werden, so dass sie für Drools Rule Engine interpretierbar sind. Dies sichert die Verwendung von den Bestandteilen der Personalisierungskomponente, die im Kapitel 2.5 vorgestellt wurden. Zugleich schließt es aber die Veränderungen oder die Anpassung dieser Bestandteile nicht aus.

Ein weiterer Aspekt ist die Benutzerfreundlichkeit. Unter benutzerfreundlich ist zu verstehen, dass den Benutzern eine intuitive, nicht überladene, grafische Arbeitsumgebung angeboten wird. Zudem soll der Arbeitsprozess als angenehm empfunden werden. Unnötige oder zeitintensive Vorgänge sollen vermieden werden. Leitkriterien oder Eigenschaften für die Benutzerfreundlichkeit sind somit [JR08, S. 4-6]:

- **Effektivität** zur Lösung von Aufgaben
- **Effizienz** im Umgang mit dem System

- **Zufriedenheit** des Nutzers während der Benutzung

Der Begriff und das damit verbundene Erhebungs- und Bewertungsverfahren reichen tief in die Materie und wurden hier nur grundlegend angesprochen. Im Konzept sollen diese Eigenschaften nicht unbeachtet bleiben.

3.2 Menschliche Dimension

In diesem Abschnitt soll ein Konzept mit Vorschlägen und Wünschen erarbeitet werden, welche den Menschen bei der Verwaltung von Personalisierungsregeln unterstützen soll. Die Betrachtung soll dabei auf die Verbindung zwischen Menschen und System fokussiert werden. Zudem werden hier notwendige Voraussetzung an den Benutzer sowie an seine Aktivitäten ermittelt.

Aktivitätsfelder des Benutzers konzentrieren sich auf die Erstellung neuer Regeln und die Veränderung bestehender Regeln. Die Erstellung hat diese Schritte:

- Modellierung vom Prozess
- Implementierung der Regel
- Inbetriebnahme der Regel

Die Modellierung ist der erste Schritt in der Erstellung neuer Regeln. Das Modell stellt den Prozess, während eine oder mehrere Regel/n durchlaufen wird, dar. Durch das Modell soll die Sichtweise auf die Lösung des Problems gelenkt und zugleich verständlicher gemacht werden. Für die Notation sollen weitverbreitete und anerkannte Standards (z. B. BPMN) genutzt werden. Der Benutzer kann dadurch eine größere Auswahl von Hilfsliteratur zur Erklärung dieser Vorschriften nutzen. Durch das Einsetzen von Standards erhöht sich zudem die Wahrscheinlichkeit, dass der Nutzer diese bereits beherrscht. Ein weiterer Vorteil könnte eine größere Kompatibilität zwischen verschiedenen Systemen sein, sollte es zum z. B. zum Import oder Export von Modellen kommen.

Für eine leichtere Durchführung der Modellierung sollte das Managementsystem einen grafischen Modellierungstool besitzen. Dieses Tool unterstützt durch die große Auswahl an bereits vorhandenen Notationen der Spezifikationssprache. Zudem sollte das Tool die

Möglichkeiten der Notationssprache vollständig ausnutzen. Die Syntax sollte während der Bearbeitung geprüft und dem Nutzer Hilfestellungen (Validierung , Fehlerbehebung, Vorlagen etc.) angeboten werden. Das Modellierungstool sollte eine *drag-and-drop* Methode oder eine vergleichbare benutzerfreundliche Bedienung für Bearbeitung nutzen.

Der zweite Schritt ist die Implementierung der Regeln anhand des zuvor erstellten Modells. Dabei können auch mehrere Regeln aus einem modellierten Prozess abgeleitet werden. Wie in den allgemeinen Anforderungen (Kapitel 3.1) festgelegt wurde, soll das Ergebnis der Implementierung durch die Drools Rule Engine interpretierbar sein. Auch bei diesem Schritt wäre ein ähnliches grafisches Modellierungstool vorteilhaft. Optimal wäre eine automatisierte Überführung aus dem Modell in die implementierte Regeldatei (DRL-Datei), ohne dass der Benutzer eingreifen müsste.

Im letzten Schritt soll der Benutzer die Entscheidung treffen können, ob die Regel betriebsbereit ist oder sich noch im Entwicklungsmodus befindet. Den Zeitpunkt dazu soll der Benutzer selbst wählen, sofern sichergestellt ist, dass die Regel fehlerfrei funktioniert. Weitere Tools und Werkzeuge sollen den Nutzer dabei unterstützen die Funktionalität der Regel zu überprüfen oder zu testen.

Die zweite Hauptaufgabe des Benutzers ist das Bearbeiten von Regeln, die im Managementtool bereits integriert sind. Diese Aufgabe verläuft analog zur Erstellung neuer Regeln, mit der Voraussetzung, dass die Regeln und Modelle im Managementtool bereits integriert sind. Somit gelten die gleichen Anforderungen wie beim Erstellen von Prozessen. Diese Aufgabe kann in einzelne Schritte aufgeteilt werden:

- Änderung / Anpassung des Modells
- Änderung / Anpassung der implementierten Regel
- Inbetriebnahme der veränderten Regel

Aus den vorgestellten Aktivitätsfeldern lassen sich Fähigkeiten und Kompetenzen ableiten, die der Nutzer zur Bewältigung der Aufgabe mitbringen sollte. So sind logische und analytische Fähigkeiten für die Verwaltung von Regeln unerlässlich. Die Kenntnis der Modelliersprache ist ebenfalls notwendig. Wird eine Spezifikationssprache für

die Implementierung benötigt, so muss diese dem Nutzer bekannt sein. Zudem sollten Hintergrundinformationen, wie der Aufbau des Nutzerprofils, Schnittstellen der Empfehlungskomponente oder der Kontext, indem sich die Regeln befindet, bekannt sein. Ohne diese Informationen kann das Potenzial des Nutzerprofils nicht ausgeschöpft werden.

Ein weiterer und wichtiger Aspekt ist die Arbeit an den Regeln während der Betriebszeit. Es müssen Vorkehrungen getroffen werden, so dass eine sichere Umsetzung der Neuerungen im Regelbetrieb möglich wäre. Es müsste sichergestellt werden, dass neue Regeln nicht mit veralteten Regeln oder veralteten Systemteilen in Konflikt geraten. Ein Versionskontrollsystem für die Dokumente würde sich als Lösung gut anbieten.

3.3 Technische Dimension

Dieses Unterkapitel befasst sich mit der technischen Konzeption des Verwaltungsvorgehens. Das Ziel soll eine Integration mit der Architektur von MeinKOSMOS sowie der Personalisierungskomponente sein. Um dieses Vorhaben abschließen zu können, müssen ergänzende Informationen zur technischen Umsetzung von MeinKOSMOS gesammelt werden. Auch eine detaillierte Betrachtung der Personalierierungskomponente ist notwendig.

3.3.1 Allgemeine Systemanforderungen

MeinKOSMOS nutzt die Portalsoftware Liferay (Version 6.2 CE GA1), die auf einem Tomcat Server (version 7.042) läuft. Die Speicherung der Daten erfolgt auf einer MySQL (Version 5.1) Datenbank. Das gesamte Informationssystem wird auf einem virtuellen Windows Server der Informatikfakultät an der Universität Rostock betrieben. Der Aufbau und die Architektur der DB und des Windows Servers ist für das hier vorgestellte Konzept vorerst nicht relevant. Wichtige Rolle für die Integration des Konzeptes spielt jedoch die Architektur (Abbildung 3.1) von Tomcat und Liferay (MeinKOSMOS). Zudem ist auch relevant zu wissen, wie die Einbindung der Personalisierungskomponente realisiert wurde.

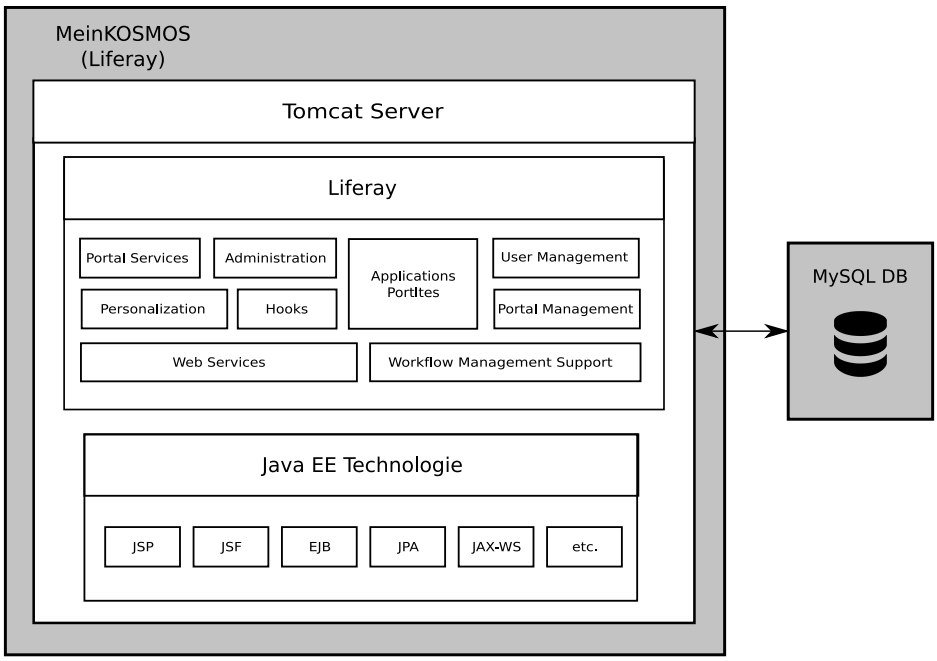


Abbildung 3.1: Liferay auf Tomcat mit Datenbank

Liferay ist nicht nur Tomcat gebunden, es kann auch auf anderen Applikationsservern laufen, wie z. B. GlassFish[1], JBoss[2] oder OracleAS[3]. Die Anbieter (RedHat) von Liferay werben mit einer großen Anpassungsfähigkeit des gesamten Portals. Möglich ist dabei nicht nur die Überschreibung der Kernfunktionen, sondern auch die Erweiterung. Abgesehen vom Erscheinungsbild des Portals lassen sich neue Anwendungen (Portlets) integrieren[lif15].

Für das Konzept ist es wichtig mehr über Portlets innerhalb von Tomcat zu erfahren. Diese Portlets befinden sich im ***webapps***-Ordner von Tomcat. Alle sich in diesem Ordner befindlichen Portes durchlaufen den *deploy*, während der Server startet oder der *deplay* von Portlets angeordnet wird. Mit dem *deploy* werden die Portlets im Server eingebunden, und können auf sämtliche Technologien von und APIs von Liferay zugreifen. Der Verlauf von *deploy* und damit verbundenen Erfolgs- oder Fehlermeldungen werden in der

[1] https://glassfish.java.net/de/
[2] http://www.jboss.org
[3] http://www.oracle.com/technetwork/middleware/ias/overview/index.html

Hauptkonsole von Liferay ausgegeben. Sollte ein *deploy* fehlschlagen, ist es zuerst dort nach der Ursachen zu schauen.

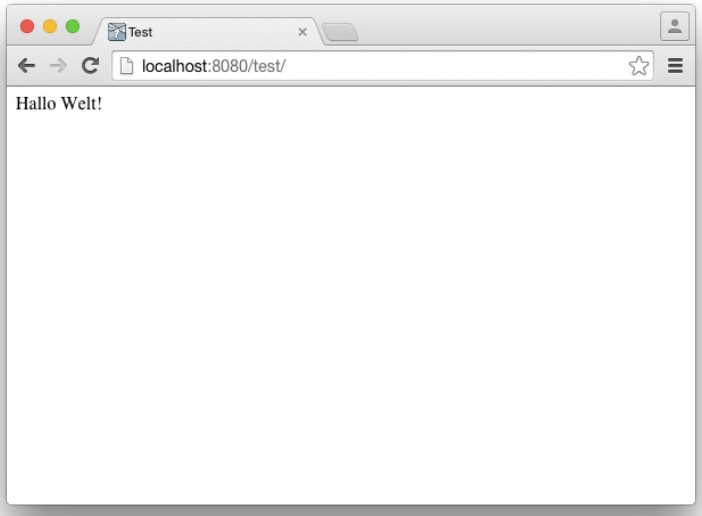


Abbildung 3.2: 'Hallo Welt' Beispiel eines Portlets

In der Regel bilden die Portlets einen festen Bestandteil im Interface eines Portals. Auf die Portlets kann aber auch über die Hauptdomain des Portals zugegriffen werden, ohne dabei das Interface zu nutzen. Einen Beispiel zeigt die Abbildung 3.2. In diesem Beispiel wird ein direkter Zugriff auf einen Poltet gezeigt, der sich im *webapps/test*-Ordner mit einer *index.jps*-Datei befindet. Wie im Beispiel zu sehen ist bildet das *http://localhost:8080/test/* die Zugriffsadresse ab. Dabei ist *localhost* der Hostname und *8080* der zugewiesene Port. Dieses Beispiel zeigt eine Möglichkeit die Portlets im Portal einzubinden, so dass diese nicht im Interface von einem Portal zu sehen und zu bedienen sind. Diese Möglichkeit könnte für das Konzept genutzt werden, um über diesen Weg das Managementtool einzubinden. Diese Möglichkeit bietet mehr Platz für das Interface des Managementtools.

3.3.2 Schnittstelle zur Empfehlungskomponente

Zur Realisierung einer Schnittstelle zwischen Managementtool und der Personalisierungskomponente ist ein genauer Einblick in die Struktur der Komponente erforderlich. Die

Personalisierungskomponente wird als Portlet im Tomcat eingebunden und befindet sich üblicherweise im *webapps*-Ordner. Aufgebaut ist das Portlet nach JSR 268 Standard. Abbildung 3.3 zeigt die Ordnerstruktur des Portlets.

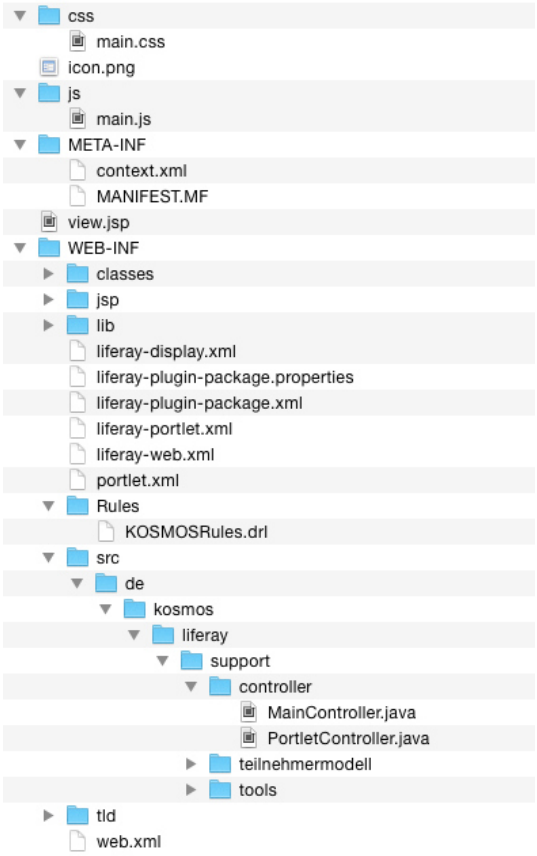


Abbildung 3.3: Ordnerstruktur von Support-Portlet

Die Konfigurationsdateien und die Spezifikationen sind in den *XML*-Dateien im Ordner *WEB-INF* zu finden. Eine der wichtigsten Dabei ist *poltet.xml*. In dieser XML-Datei werden grundlegende Information über das Portlet festgehalten, wie z. B. *portlet-name*, *display-name*, Zugriffsberechtigung oder die Information an welchem Ort sich View-Interface Datei (*view.jps*) befindet. Im *lib*-Ordner sind zudem alle notwendigen Java Libraries zu finden, die beispielweise gebraucht werden, um die Drools Regeln zu interpretieren oder, die Aufrufe zur einer MySQL Datenbank ermöglichen. Essenziell für die Funktionalität der Empfehlungskomponente sind Drools Regeln in der *WEB-*

INF/Rules/KOSMOSRules.drl-Datei. In dieser Datei sind alle benutzen Regeln sowie die zusätzlich gebrauchte Java-Funktionen. Wie im Kapitel 2.5 beschrieben wurde, wird diese Datei extern, also außerhalb des Java Codes, gelagert und beim Betrieb eingelesen. Im `MainController` findet sich der Aufruf, der dafür zuständig ist, dass diese Datei mithilfe eines globalen Pfades geladen wird. Das Listing 3.1 zeigt dazu den genauen Java Code an.

In der Zeile `"2"` wird der Speicherort der Regeldatei als globaler Pfad definiert. Dieser Pfad zeigt somit auf das Betriebsystem und kann auch auf Dateien außerhalb von MeinKOSMOS zugreifen. Bis einschließlich Ziele `"8"` wird ein *Paket* für eine RuleBase vorbereitet. Durch das Einfügen weiterer Pakete aus anderen DRL-Dateien könnten zusätzliche Regeln eingelesen werden. Eine Aufspaltung der zentralen Regeldatei in kleinere und übersichtlichere Dateien könnte an dieser Stelle ratsam sein. In Zeile `"9"` wird eine RuleBase als `ruleBase` deklariert. Anschließend in Zeile `"10"` wird der RuleBase das *Paket* übergeben, wodurch die RuleBase die importierten Regeln beinhaltet und bereit für die Interpretation ist. Anschließend in Zeile `"11"` wird die `ruleBase` für weitere Schritte bereitgestellt und ausgegeben. Diese Importmöglichkeit der DRL-Datei biete viele Wege für weitere Schnittstellen für das hier konzipierte Managementtool.

```
1  private static RuleBase readRule() throws Exception {
2    File f = new File("C:\\Users\\rs\\LIFERAY_Standalone\\_liferay_6
       .2\\liferay-portal-6.2.0-ce-ga1\\tomcat-7.0.42\\webapps\\Support2-
       portlet\\WEB-INF\\Rules\\KOSMOSRules.drl");
3    Reader source = new InputStreamReader(new FileInputStream(f) );
4    PackageBuilder builder = new PackageBuilder();
5    PackageDescr packageDescr = new PackageDescr("de.kosmos.liferay.
       support.controller");
6    builder.addPackage(packageDescr);
7    builder.addPackageFromDrl( source );
8    org.drools.core.rule.Package pkg = builder.getPackage();
9    RuleBase ruleBase = RuleBaseFactory.newRuleBase();
10   ruleBase.addPackage( pkg );
11   return ruleBase;
12 }
```

Listing 3.1: Import und Ausführung der Regeln nach Ackermann [Sam14]

Zudem ist wichtig zu betrachten an welcher Stelle Drools Rule Engine bereitgestellt wird. Bevor die Drools Elemente wie RuleBase oder PackageBuilder benutzt werden können, muss Drools durch einen Import der Library in `MainController` geladenen werden. Ab diesem Zeitpunkt sind sämtliche Drools Elemente verfügbar. Diese Aufgabe übernimmt ebenfalls `MainController` indem ein Import an den üblichen Stellen in der Javadatei statt findet. Der gesamte Code dazu ist im Listing zu finden. Hierbei wird die Library in die Javaklasse geladen und ohne weiteren notwendigen Schritten benutzt.

```
1 import org.drools.compiler.compiler.PackageBuilder;
2 import org.drools.compiler.lang.descr.PackageDescr;
3 import org.drools.core.RuleBase;
4 import org.drools.core.RuleBaseFactory;
5 import org.drools.core.WorkingMemory;
```

Listing 3.2: Import von Drools aus MainController nach Ackermann [Sam14]

Die vorgestellte und simple Einbindungen von Drools und der DRL-Datei lassen viele Möglichkeiten für eine Schnittstelle zum Managementtool. Die optimale Lösung wäre ein Managementtool, das als Portlet in MeinKOSMOS eingebunden wird. In dieser From integriertes Managementtool kann fast uneingeschränkt auf die Funktionen, Schnittstellen, Datenbanken und Dateien von MeinKOSMOS zugreifen. Zudem wäre der Zugriff auf das Managementtool ohne großen Aufwand von Außen für die Benutzer möglich. Ein Beispiel für solchen Zugriff wurde in der Abbildung 3.2 vorgestellt. Beispielhafte URL könnte dafür *http://kosmos.informatik.uni-rostock.de/**managementtool*** sein. Die DRL-Datei oder mehrere Dateien sollten für eine einfachere Verwaltung und Übersicht zentral gelagert werden. Dabei spielt es keine Rolle, ob dieser Ort innerhalb oder außerhalb von Support-Portlet oder Managementtool-Portlet sich befindet. Entscheidene Rolle spielen hier nur die Zugriffsrechte. Das Support-Portlet muss zumindest die Leserechte für die DRL-Datei bekommen, das Managementtool-Portlet hingegen Lese- und Schreibrechte. Abbildung 3.4 zeigt diese konzeptionelle Integration des Managementtool als Poltet in MeinKOSMOS.

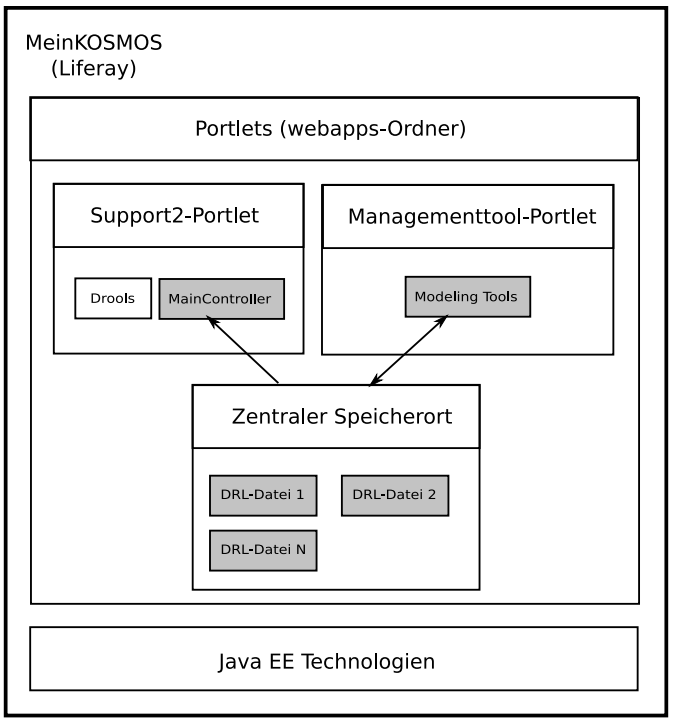


Abbildung 3.4: Managementtool integriert als Portlet in MeinKOSMOS

3.4 Gesamtkonzept

Dieser Abschnitt fast die beiden vorgestellten Dimensionen zusammen und schafft einen gesamten Überblick des Konzeptes. Hierfür werden die Eigenschaften und Kriterien in einer Übersichtstabelle (Tabelle 3.1) zusammengefast. Diese Tabelle wird zum Vergleich zwischen Konzept und Umsetzung im Kapitel 5 als Leitfaden benutzt.

Tabelle 3.1: Übersicht der Konzepteigenschaften

Eigentschaft	**Kurzbeschreibung**
Implementiert als Portlet	Tool integriert als Portlet im MeinKOSMOS
Prozess Modellierungstool	Grafisches Tool zum erstellen von Prozessen
Regel Modellierungstool	Grafisches Tool zum erstellen von Regeln
Validierung	Prüfung auf Fehler während der Bearbeitung

Modell Transformation	Automatische Transformation von Modellen in implementiere Prozesse oder Regeln
Betriebsfreigabe	Prozess mit einem Exklusives Gateway
Sichere Verwaltung	Versionsorientierte Dokumentenverwaltung
Benutzerfreundlichkeit	Effiziente, effektive und angenehme Arbeitsweise mit dem Tool

4 Umsetzung

Unterteilt ist dieses Kapitel in zwei Abschnitte. Der erste Abschnitt 4.1 befasst sich mit der Umsetzung des Grundsystems nach Konzeptvorgabe. Abschnitt 4.3 erklärt wie die Schnittstelle zwischen Empfehlungskomponente und Managementtool erstellt worden ist.

4.1 Systemeinrichtung

Der Umfang zur Erstellung eines eigenen Managementtools, zu den im Konzept vorgestellten Bedingungen, ist für den Zeitraum dieser Arbeit zu groß. Das Ziel ist also ein Open Source Managementtool zu finden, das diese Kriterien erfüllt. Das Ergebnis einer Recherche wird hier kurz vorgestellt. Einen Vorschlag für ein Managementtool (jBPM) hat Ackermann bereits in seiner Masterarbeit geäußert [Sam14, S. 81].

Activiti[1] zählt zu den in Betracht gezogenen Lösungen. Das Open Source Projekt besteht seit 2010 und verfügt mittlerweile über fast alle Kriterien, die für das Konzept gesucht werden. Das offizielle Installationspaket wird mit Tomcat geliefert und ist einfach zu konfigurieren. Dieses Managementtool wird über einen Webinterface gesteuert. Hier können alle angebotenen Funktionen genutzt werden, dazu zählen die Modellierung von Prozessen mit BPMN mittels eines grafischen Modellierungstools sowie das Vergeben von Aufgaben oder das Überwachen von Aktivitäten. Die Speicherung von Daten erfolgt in einer Repository Datenbank [Tea15].

Bonitasoft[2] wirbt mit einer einfachen *drag-and-drop* Bedienung zur Herstellung von Geschäftsprozessen, Entscheidungstabellen oder Formularen. Neben der kostenlosen Edition können auch andere Versionen kostenpflichtig erworben werden. Neben den hier kurz vorgestellten Lösungen existieren weitere Funktionen im Portfolio von Bonitasoft, die für das Konzept nicht relevant sind.

[1] http://activiti.org
[2] http://bonitasoft.com

Die Projekte von JBoss oder Red Hat bieten bessere Lösungen an. Dazu zählen **Red Hat JBoss BPM Suite**[3], **Drools**[4] oder **jBPM**. Alle diese Softwarelösungen unterstützen sowohl eine grafische Modellierung von Prozessen als auch das Editieren von Geschäftsregeln (Drools Rule). Abgesehen davon werden diese Projekte durch die Drools Community unterstützt, welche die Drools Rule Engine entwickelt hat.[Rad12, bon15, red15]

Die Recherche hat den Vorschlag von Ackermann bestätigt. jBPM verfügt über viele Werkzeuge, die die gesuchten Kriterien erfüllen. jBPM stellt eine Verschmelzung sämtlicher BPM-Tools von JBoss und Red Hat dar. Auf der offiziellen Seite von jBPM wird mit zahlreichen Funktionen zur Gestaltung und Verwaltung von Drools Regeln geworben. Das Tool eignet sich für Entwickler und für Analysten und verfügt daher auch Werkzeuge zum Simulieren von Szenarien. Die Prozesse werden in BPMN 2.0 mittels einer grafischen Oberfläche modelliert. Zum anderen ist jBPM auch ein Kollaborationssystem, welches das Anlegen, Vergeben und Überwachen von Aufgaben an die Benutzer anbietet. Die große Auswahl an Funktionen schafft großes Potenzial für weitere Einsatzgebiete in der Verwaltung von Personalisierungsregeln. Aus diesem Grund wurde jBPM für die Umsetzung gewählt.

4.1.1 Integration als Portlet

Für die Umsetzung wurde die aktuellste Version (6.2 Final) von jBPM ausgesucht. jBPM verfügt laut der Entwicklerseite über mehrere Distributionen für verschiedene Server. Somit ist das Tools nicht an den Standardserver (Wildfly bzw. JBoss) gebunden, wie von Red Hat empfohlen. Eine Distribution zur Integration auf einem Tomcat Server steht zum Download frei. Damit ist diese Distribution perfekt geeignet für das hier erstellte Konzept. Eine offizielle und aktuelle Anleitung für die Integration in Tomcat gibt es jedoch nicht.

Auf den Community Webseiten von jBPM, Tomcat und Liferay befinden sich einige Anleitungen für die Integration im Liferay, diese sind jedoch veraltet und haben kaum positive Resonanz von der Community. Die Versuche einer Integration auf einer lokalen Instanz von MeinKOSMOS waren erfolglos. Viele dieser Anleitungen forderten Schritte,

[3] http://www.redhat.com/de/technologies/jboss-middleware/bpm
[4] http://www.drools.org

die durch die neuen Versionen der Software nicht mehr durchführbar waren. So sollte z. B. eine `services.xml` Datei um bestimmte Einträge erweitert werden. Diese Datei existierte in der aktuellen Version nicht mehr. Das Zurückgreifen auf ältere Versionen der Software wurde mit Absicht nicht in Betracht gezogen [jbp15a, jbp15b, jbp15c, jbp15d].

Der erfolgreichste Versuch einer Integration als Portlet lieferte keine Fehler in der Konsole oder in den Logfiles. Das Portlet konnte dennoch nicht integriert und funktionsfähig gemacht werden. Tomcat hatte zwar den *deploy* angekündigt, aber ohne Erfolg oder einer Fehlermeldung abgeschlossen. Teile der Konsolenausgabe und Logfiles sind im Anhang D zu finden. Die Integration als Portlet, wie in der Abbildung 3.4 zu sehen ist, konnte nicht erfolgreich beendet werden. Im nächsten Abschnitt wird daher eine alternative Methode vorgestellt, um das Problem zu lösen.

4.1.2 Integration als eigenständiges System

Während der Umsetzungsphase kam es zu Problemen bei der Integration von jBPM als Portlet, daher wird hier eine alternative Lösung vorgestellt. Es wurde eine Umstrukturierung im Konzept vorgenommen, so dass ein weiteres System parallel zum MeinKOSMOS auf dem Windows Server laufen soll. Dadurch entfallen die Schnittstellen zur Liferay, die allen Portlets bereit stehen. Zudem verläuft der Zugriff von Außen nicht mehr über die geplante gemeinsame URL von MeinKOSMOS.

Somit entsteht eine neue Grundstruktur, die nicht mehr wie in Abbildung 3.4 statt findet. Die neue Übersicht der Struktur wird in der Abbildung 4.1 vorgestellt. Im Vorfeld, um mögliche Komplikationen zu vermeiden, wird in der neuen Grundstruktur die Speicherung der Daten innerhalb von jBPM betrachtet. Diese Betrachtung ist wichtig für das Kapitel 4.3, welches sich mit der Schnittstelle zwischen Empfehlungskomponente und jBPM beschäftigt. MeinKOSMOS und jBPM sind nun eigenständige Systeme, die unabhängig von einander laufen sich aber den selben Hostnamen teilen, allerdings mit verschiedenen Ports. jBPM kann als ein Portal angesehen werden, mit einer Anzahl von Portlets, Tools und Werkzeugen, die Funktionalität von jBPM definieren. Dazu zählen unter anderem der DRL-Editor für die Regeldateien oder der Prozess-Editor mit einer grafischen UI zum erstellen von BPMN 2.0 Prozessen. Die Speicherung von Dateien geschieht innerhalb eines Git Repositories, der nur während der Laufzeit von jBPM erreichbar ist. In der Abbildung 4.1 gezeigte **Variante 1** bildet eine direkte Verbindung zum jBPM Git Repository. Die Realisierung einer dauerhaften Verfügbarkeit dieses Git

Repositories wird durch die **Variante 2** abgebildet. Dabei wird eine lokale Kopie des jBPM Repository erstellt und als Schnittstelle zum Support-Portlet genutzt.

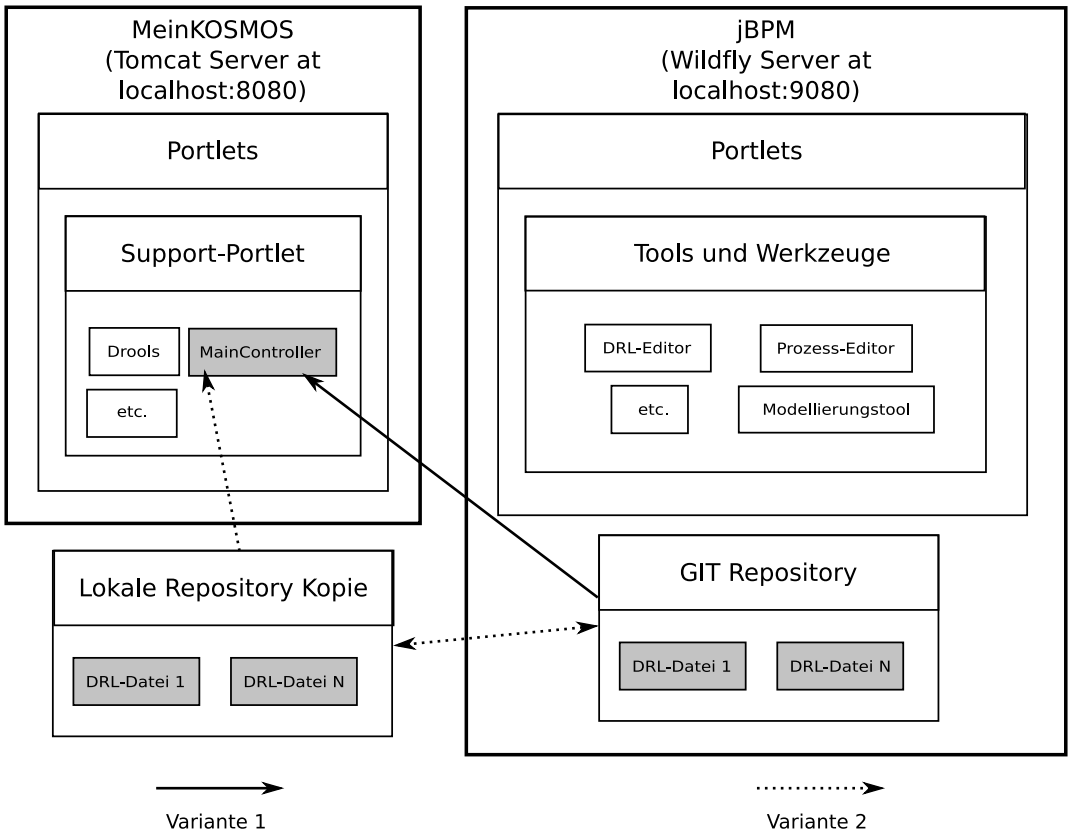


Abbildung 4.1: jBPM integriert als eigenständiges System mit 2 Varianten der Schnittstelle

Das standard Installationspaket von jBPM 6.2 liefert ein eigenständiges System, welches auf einem Wildfly-Server (Version 8.1.0) läuft. Neben dem Server wird eine H2 Datenbank erstellt, welche dem Managementtool zur verfügung steht. Die offizielle Installationsanleitung[5] von jBPM verspricht eine schnelle und einfache Installation per Terminal bzw. Kommandokonsole mittels Apache Ant[6]. Typisch für Apache Ant ist das Erstellen von Java Anwendungen mittels einer `build.xml`-Datei, in der alle wichti-

[5] http://docs.jboss.org/jbpm/v6.0/userguide/jBPMInstaller.html
[6] ant.apache.org

gen Einstellungen zentral gesammelt und somit leicht zugänglich sind. Die Konfiguration kann zentral in dieser Datei vorgenommen werden. Es könnten auch Funktionen in der `build.xml` definiert sein, die mittels Eingabekonsole aufgerufen werden [SL07]. Dieses Wissen ist notwendig, um die Installationsschritte von jBPM zu verstehen. Folgende Schritte zeigen wie jBPM Demo installiert und gestartet werden kann, dabei werden alle Konsolenbefehle aus dem Speicherort ausgeführt, wo das Installationspaket entpackt worden ist. Demo Installation bedeutet, dass Minimalbeispiele und standard Einstellungen bei der Installation erstellt werden.

1. jBPM Installationspaket am Speicherort JBPM_HOME entpacken
2. Konfiguration der `build.xml`-Datei
3. Durch den Konsolenbefehl `ant install.demo.noeclipse` jBPM ohne Eclipse installieren
4. Durch den Konsolenbefehl `ant start.demo.noeclipse` jBPM starten

Neben den vorgestellten Befehlen befinden sich weitere Befehle in der `build.xml`-Datei, wie z. B. `stop.demo` welcher jBPM ausschaltet oder `start.jboss` welcher nur den JBoss Server startet, ohne den anderen jBPM-Komponenten. Im Fall einer großen individuellen Anpassung (Schritt 2) soll die Konfigurationsdatei genau studiert und an entsprechenden stellen umgeschrieben oder erweitert werden. Unterstützende Befehle können der Dokumentation von JBoss entnommen werden.

Für die Integration in das bestehende System von MeinKOSMOS wurde eine Instanz von jBPM (6.2 Final) installiert. Vor der Installation mussten die Ports für den JBoss Server geändert werden, da diese standardgemäß gleich mit den Ports von MeinKOSMOS sind. Das Benutzen von den selben Ports würde zur Fehlern führen. Laut Documentation[7]

von JBoss gibt es eine bequeme Methode einen Port-Offset zu setzen, welcher alle benutzen Port um diesen Offset verändert (addiert). Neben dem Port-Offset wurde eine weitere Funktion angepasst. Diese Funktion meldet einen Feedback, sofern der Server gestartet ist. Durch den Port-Offset greift diese Funktion auf den Standardport *8080*, statt auf den neuen durch den Offset veränderten Port (im Beispiel 9080). Wird diese

[7] https://docs.jboss.org/author/display/WFLY8/Documentation

Funktion nicht angepasst, bekommt die Eingabekonsole einen fehlerhaften Feedback vom Server. Somit wird davon ausgegangen, dass der Server nicht gestartet ist, obwohl dieser bereits läuft. Die genaue Installationsanleitung kann aus dem Anhang C entnommen werden. Nach der Installation laufen beide Systeme und können über diese Adressen erreicht werden:

- MeinKOSMOS : `http://localhost:8080`
- jBPM : `http://localhost:(8080+offset)/jbpm-console`

4.2 Einblick in jBPM

Für den Zugriff über den Internetbrowser auf jBPM werden gültige Accounts benötigt. Standardgemäß existieren einige Accounts mit verschiedenen Rollen und damit Berechtigungen für Aktionen innerhalb von jBPM. Gespeichert sind sämtliche Accounts in einer `users.properties`-Datei, die sich im Konfigurationsordner von Wildfly befindet. Nachdem ein Account hinzugefügt worden ist muss eine oder mehrere Rollen dem Account zugeordnet werden. Im selben Ordner in der `roles.properties`-Datei werden die Nutzerrollen vergeben. Während der Umsetzung wurde der standard Account *krisv* mit dem *krisv* Passwort benutzt.

Für die Regeln im MeinKOSMOS wurde ein neues Repository in jBPM erstellt. Bevor die Dateien dort gespeichert werden können, muss im Repository ein Projekt erstellt werden. Jede Datei, die erzeugt wird ist einem Projekt zugewiesen und befindet sich im Projektordner. Ein Repository kann mehrere Projekte und somit Projektordner beinhalten. Für MeinKOSMOS wurde ein *KOSMOSRules* Repository mit dem Projekt *ActiveRules* angelegt. Abbildung 4.2 zeigt einen Bildschirmausschnitt aus jBPM, indem die Projektnavigation mit Projektdateien zu sehen ist. Wie dieser Abbildung zu entnehmen ist, werden alle Dateitypen (Business Processes, DRL) übersichtlich in Dateilisten von einander getrennt.

Im Reiter *New Item* kann der Dateityp für eine neue Datei ausgewählt werden. Zentrale Rolle bei der Verwaltung spielen die beiden Typen *DRL file* und *Business Process*.

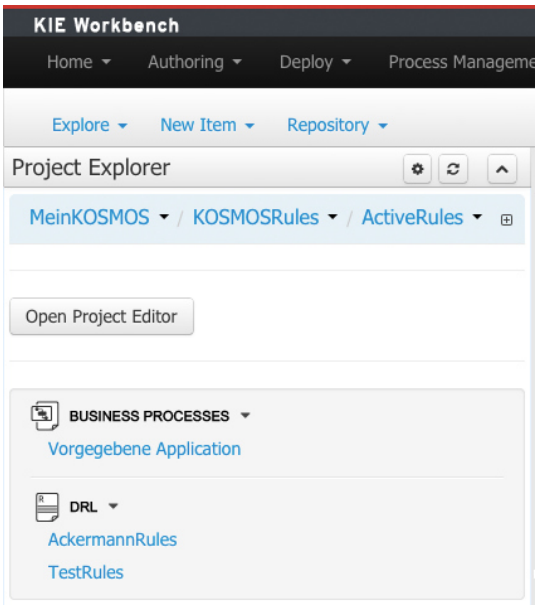


Abbildung 4.2: Projektübersicht aus jBPM mit Projektdateien

Um einen Einblick in das Modellierungstool von BPMN Prozessen zu bekommen, wurde der zuvor vorgestellte Prozess *Vorgegebene Anwendungen* (Abbildung 2.12) mit dem Modellierungstool modelliert. Abbildung 4.3 zeigt das Ergebnis. Abgebildet sind einige Werkzeuge, die das Modellierungstool anbietet. Neben den BPMN-Objekten lassen sich auch komplette vorgefertigte *Workflow Muster* einfügen, die einige Szenarien von Aufteilung und Zusammenführung der Gateways abdecken. Bedient wird das Modellierungstool mittels *drag-and-drop* Methode. Zudem werden während der Bearbeitung Vorschläge (Schnellauswahl) für die angrenzenden oder folgenden Objekte gemacht. Diese Schnellauswahl und die *drag-and-drop* Methode gestaltet das Arbeiten mit dem Tool unkompliziert und schnell. Objekteigenschaften wie Name, Priorität oder Farbschema werden im Eigenschaften-Fenster angepasst.

Eine weitere Option, die das Tool anbietet, ist die Validierung von Prozessen. Während der Validierung werden zum jedem Objekt dazugehörige Fehler angezeigt, sofern ein Objekt angeklickt wird. Sonst können alle Fehler im Nachrichtenfenster nachgeschaut werden. Das Tool verfügt über weitere Funktionen die im Kapitel 6.2 angesprochen werden. Grundlegende Vorstellung der Funktionen ist jedoch hiermit abgeschlossen.

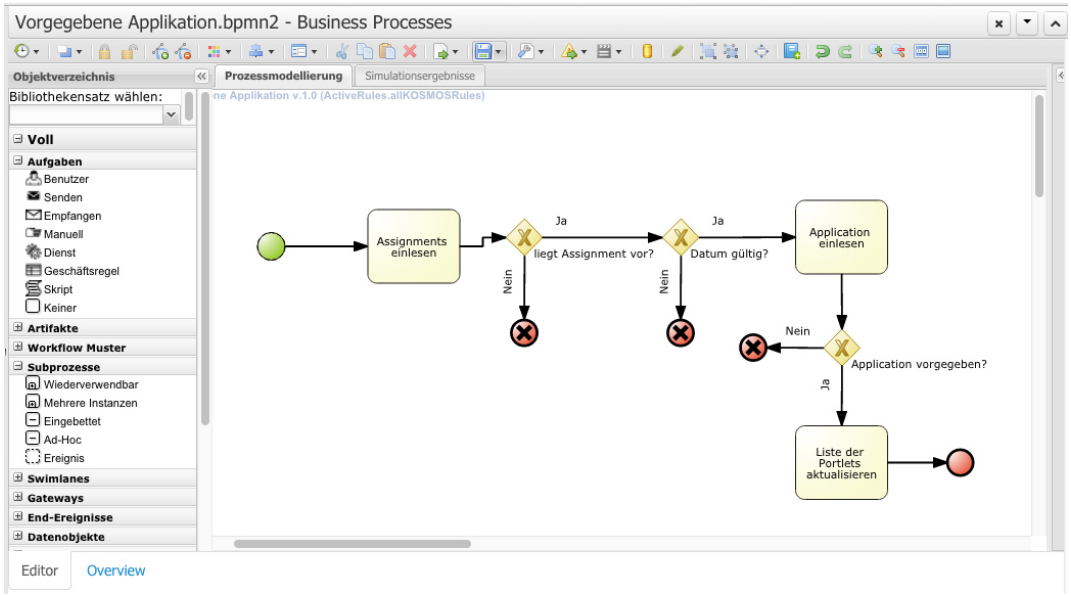


Abbildung 4.3: Prozess-Modellierungstool von jBPM

Das Editieren von Regeln in jBPM übernimmt der DRL-Editor. Es besteht keine Option aus den modellierten Prozessen automatisch, durch eine Transformation, Regeldateien zu erzeugen. Werkzeuge zur grafischer Gestaltung der Regeln werden nicht angeboten. Auch wenn die Anforderung eines grafischen Modellierungstools für die Regeln nicht erfüllt werden konnte, existiert ein Mehrnutzen durch den Editor. Neben der Validierung von Regeln besteht die Option einzusehen, welche Fakten an die Regeln übergeben werden. Fakten sind dabei Objektklassen mit Attributen. Abgebildet werden die Fakten aus den im Projektordner sich befindlichen Java-Dateien. Diese können mittels *New Item* erzeugt werden. Die andere und schnellerer Option ist die bereits bestehenden Objektlassen in den Projektordner zu importieren (uploaden). Abbildung 4.4 zeigt einen Abschnitt aus dem Objekteditor.

Nach dem Import werden die Objekte automatisch erkannt und stehen als Fakten zu verfügung. Damit hat der Benutzer direkt eine Übersicht aller Fakten, die er sonst mühsam erlernen müsste. Mit dem Klick auf einen Fakt wird dieser als Code in das Editor-Fenster geladen und spart damit Bearbeitungszeit. Abbildung 4.5 zeigt den DRL-Editor mit importierten Teilnehmermodell und Regeln. Gut zu erkennen ist das Klassenobjekt `Teilnehmer` mit Attributen wie `nutzerID`, `pruefungen` oder `studienformat`, die

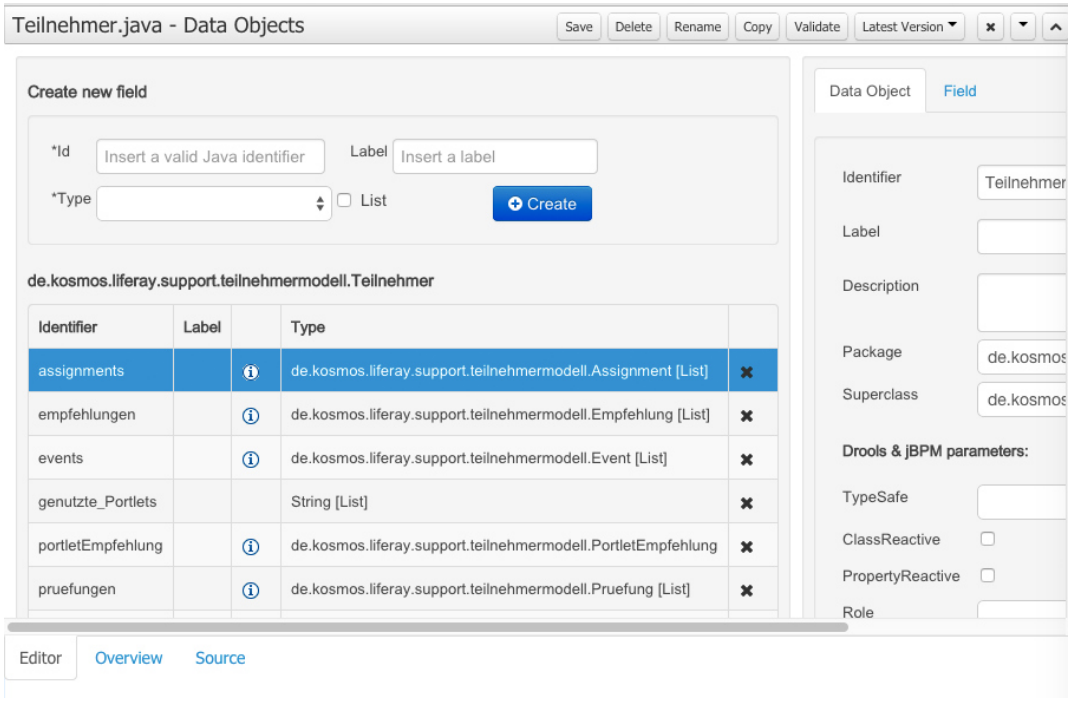


Abbildung 4.4: Data-Modeller von jBPM

als verfügbare Fakten stehen.

Beachtet werden muss, dass jBPM nur Objektattribute erkennen und validieren kann. Verwendet der Benutzer in den Regeln Objektfunktionen, so können diese nicht erkannt und validiert werden. Werden die Objektfunktionen korrekt angewendet, so verläuft der Einsatz der Regeln in MeinKOSMOS fehlerfrei, trotz Fehlermeldung während der Validierung. Im Kapitel 6.2 wird eine Lösung vorgestellt, die das Nutzerprofil an die Validierung anpasst.

jBPM bietet deutlich mehr als hier vorgestellt. Ein Ausblick für weitere Einsatzmöglichkeiten und somit Erweiterung des Nutzens für MeinKOSMOS wird im Kapitel 6.2 vorgenommen. An dieser Stelle wurden lediglich die Funktionen vorgestellt, die im Konzept gefordert worden sind.

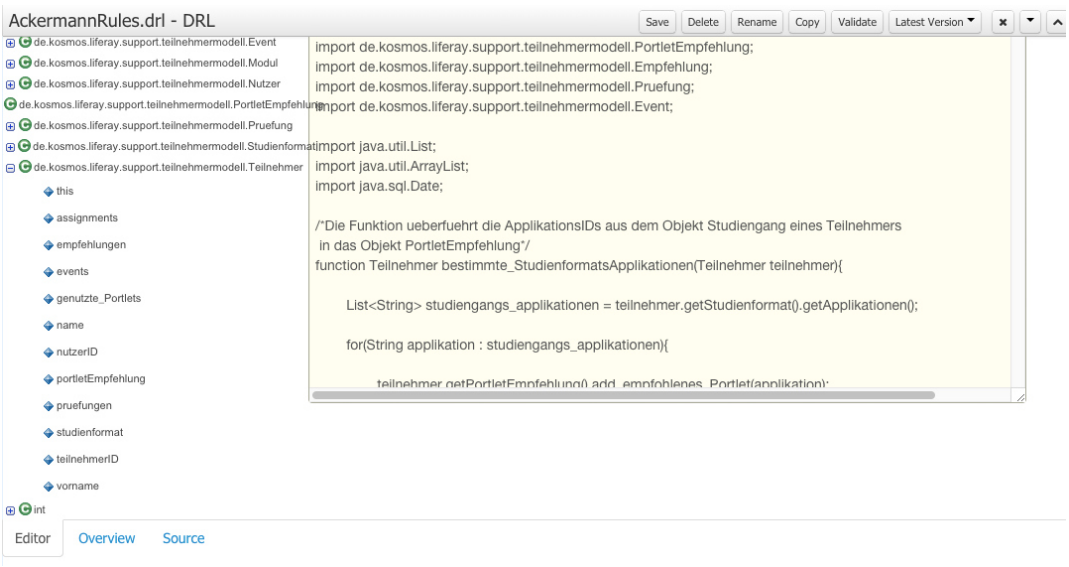


Abbildung 4.5: DRL-Editor mit Faktenübersicht

4.3 Realisierung der Schnittstelle

Für die Umsetzung der Schnittstelle wurden zwei alternative Varianten entworfen und zuvor im Kapitel 4.1.2 kurz angesprochen. Die erste Variante ist eine direkte Verbindung aus dem `MainController` mit einem Lesezugriff auf den Git Repository von jBPM. Die Veränderungen im jBPM könnten somit direkt an MeinKOSMOS übergeben werden. Zum Nachteil dieser Umsetzung ist die permanente Laufzeit von jBPM notwendig und es müsste sichergestellt werden, dass keine fehlerhaften Regeldateien übergeben an MeinKOSMOS werden.

Der zweite Umsetzungsweg trennt die direkte Verbindung zum jBPM Git, indem es eine lokale Kopie vom Git erstellt und auf diese zugreift. Wodurch der Nachteil einer permanenten Laufzeit von jBPM kompensiert wird. Jedoch fehlt es an der Aktualität der Regeln, denn die Veränderungen im jBPM Git wären nicht mehr direkt für MeinKOSMOS sichtbar sein. Die Verwaltung über die lokale Kopie von Git übernimmt das Eclipse Addon EGit[8]. Um aktuelle Regeldateien in MeinKOSMOS einzulesen, muss der lokale Git einen Pull aus dem jBPM Git durchführen. Abbildung 4.1 steht die schema-

[8] http://eclipse.org/egit/

tische Struktur dieser Umsetzungswege. Implementiert wird die Alternative mit einem zusätzlichen lokalen Git, da im Laufe von Testimplementierung festgestellt worden ist, dass die direkte Verbindung zum Git Schwierigkeiten bereitet.

Abgesehen von der Methode, wie auf den Git Repository zugegriffen wird, sind weiter Funktionen notwendig, um die Dokument im Git auszulesen. Für die zentrale Sammelstelle dieser Funktionen wurde eine `GitController` Klasse entwickelt. Abbildung 4.6 zeigt die Attribute und Funktionen dieser Klasse.

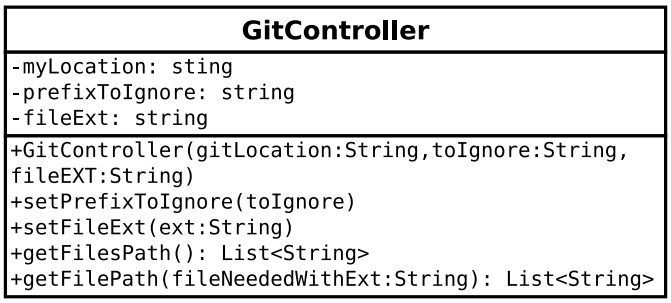


Abbildung 4.6: UML-Schema von GitController

Der `GitController` besitzt die Attribut `myLocation`, in dem der Speicherort von Git festgehalten wird, `prefixToIgnore` wo der Prefix für Dateinamen festgehalten wird, die bei der Suche ignoriert werden sollen und ein Attribut `fileExt` welches den gesuchten Dateitype beinhaltet. Die Suche von Dokumenten und die Ausgabe dieser in einer Liste übernimmt die Funktion `getFilesPath`. Wird nur eine bestimmte Datei gebraucht, so kann die Funktion `getFilePath` genutzt werden. Diese Funktion erwartet einen Dateinamen inkl. Dateityp als Parameter. Zu beachten ist, dass `GitController` nur auf einen lokalen Git zugreifen kann. Der gesamte Java Code von `GitController` befindet sich im Anhang A.

Im letzten Schritt zur Realisierung einer Schnittstelle sind Veränderungen in der Klasse `MainController` vorgenommen worden. Bevor neue Funktionen hinzugefügt worden sind, musste das Portlet angepasst werden, da es in seiner ursprünglichem Version aufgrund von fehlender Java Libraries Fehlermeldungen verursachte [Sam14, S. 75]. Im Anhang E befinden sich detaillierte Schritte, die notwendig waren, um die Fehlermeldung zu beheben. Anschließend konnte `MainController` um die Funktion `readAllRulesFromRepo`

erweitert werden. Die Funktion `readAllRulesFromRepo` nutzt ein Objekt der Klasse `GitController` um alle Regeldateien die im lokalem Git vorkommen auszuführen. Der Java Code für die angepasste `MainController` Klasse befindet sich im Anhang B.

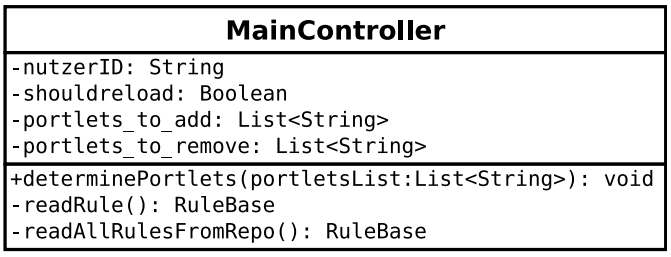


Abbildung 4.7: Neues UML-Schema von MainController

Nach der Implementierung beider Bestandteile wurde ein Test durchgeführt, indem mehrere Regeldateien mittels jBPM System erstellt worden sind. Daraufhin sollte MeinKOSMOS diese einlesen und interpretieren. Die Regeldateien, die sich im Entwicklungsmodus (Prefix) befanden wurden im Ladevorgang ‚wie geplant, ignoriert. Das genaue Vorgehen während der Tests kann im Anhang F nachgelesen werden. Die positiven Ergebnisse des Tests schließen die Umsetzung einer Schnittstelle zwischen MeinKOSMOS und `MainController` ab.

5 Kritische Betrachtung

Im ersten Teil dieses Kapitels wird ein Vergleich von Konzept und der implementierten Umsetzung gemacht. Leitfaden für den Vergleich bildet die im Kapitel 3.4 erstellte Übersichtstabelle mit den Hauptkriterien des Konzepts. Anschließend wird ein kritischer Blick auf die Umsetzung sowie auf die daraus resultierenden Problemen und Lösungen gemacht.

5.1 Vergleich

Für die Umsetzung wurde entschieden ein webbasiertes Business Process Managementtool namens jBPM zu nutzen. Das Tool hat laut Hersteller die Eigenschaften und Werkzeuge, die im Konzept definiert worden sind. Im Konzept war die Implementierung des Managementtools als Portlet vorgesehen. jBPM bietet eine Portlet-Version, die speziell zur Integration im Tomcat vorgesehen ist. Leider konnte diese Version bereits zu Beginn der Umsetzung nicht in MeinKOSMOS (Tomcat) integriert werden. Dadurch entfallen viele Schnittstellen zum Portal aus.

Aus der Performance-, Entwicklungs- und Managementsicht wäre so eine Integration von Vorteil. Es ist also interessant zu prüfen, woran genau die Integration gescheitert ist und man sollte versuchen eine Lösung für das Problem zu finden. Die restlichen implementierten Bestandteile der Umsetzung sind ohnehin in MeinKOSMOS integriert, wodurch eine Verknüpfung dieser Bestandteile in das jBPM-Portlet keine Probleme bereiten sollte. Vorerst wurde aber das Konzept an die Gegebenheiten angepasst und steht nach der Umsetzung als eigenständiges System dar.

Ein weiterer wichtiger Aspekt war die Realisierung eines grafischen Tools zur Verwaltung von BPMN Prozessen. Diese Prozesse bilden einen Bestandteil bei der Erstellung neuer Personalisierungsregeln. Das umgesetzte Managementtool verfügt über ein solches

Werkzeug. Es besteht die Möglichkeit von einfacher Modellierung per *drag-and-drop* Methode mithilfe einer großen Auswahl an vorgefertigten BPMN Elementen. Das Modellierungstool bietet zudem weitere Funktionen wie das Validieren von Prozessen, was die Qualität der Arbeit verbessert, das Exportieren oder Importieren von Prozessen oder das Ausführen von Testszenarien. Somit erfüllt die Umsetzung die Anforderungen an ein Modellierungstool für die Prozesse.

Ähnliche Werkzeuge erfordert das Konzept für die Implementierung von Drools Regeln. Leider konnte solch ein Modellierungstool bisher nicht Umgesetzt werden. Es gibt somit keine solide Lösung, die den Nutzer die Implementierung durch Programmierung ersetzen lassen und sich davon entfernen könnte. Das verfügbare Tool zur Bearbeitung von DRL-Dateien unterstützt den Benutzer dennoch auf andere Weise. Durch eine strikte Definition von benutzbaren Fakten mit Attributen hat der Nutzer stets eine Vorgabe, welche Attribute zur Umsetzung der Regeln verfügbar sind. Beachtet werden muss, dass ausschließlich nur Attribute als Vorlage definiert werden können. Greift die Regel auf eine Funktion, so ist die Validierung fehlerhaft. Regeln, die in der jetzigen Form implementiert worden sind, können somit nicht validiert werden, dies beeinflusst aber nicht die Funktionalität der Regel im Betrieb.

Das Konzept sieht für die Überführung aus Modellen in implementierte und betriebsbereite Dokumente eine automatische Transformation (Export) vor. Bei den Prozessmodellen existieren Export Möglichkeiten in PDF- oder BPMN2-Dateien. Die BPMN2-Dateien gelten als Standard und können durch entsprechende Anwendungen interpretiert und ausgeführt werden. Aufgrund fehlender Modelle von Regeln, kann keine Transformation stattfinden. Das Kriterium zur grafischen Bearbeitung der Regeln konnte nur teilweise umgesetzt werden.

Um während der Verwaltung von Regel-Dateien und anderen Dokumenten Konflikte, wie Überschreibung oder Löschung, zu vermeiden, sollte eine Versionsverwaltung eingeführt werden. jBPM speichert sämtliche Dokumente in einem Git Repository. Es besteht auch die Möglichkeit mehrere Repositories anzulegen, um möglichen Konflikten noch besser ausweichen zu können. Somit konnte die Anforderung umgesetzt werden.

Das Konzept verlangte zudem nach einer Lösung für die Aktivierung der Regeln. Es sollte dem Nutzer eine einfache Möglichkeit angeboten werden bestimmte DRL-Dateien zur Benutzung freizugeben bzw. bestimmte Dateien auszuschließen. Durch diese Option können die Regeln, die sich noch in der Entwicklung befinden durch MeinKOSMOS erkannt und entsprechen behandelt werden. Zugleich soll dem Benutzer möglich sein die Anzahl der übergebenen Dateien an MeinKOSMOS zu variieren. Die ursprüngliche Lösung erforderte einen Eingriff in den Java Code, sofern eine weitere Regeldatei geladen werden sollte. Die umgesetzte Lösung nutzt bei der Ausgabe gesuchter Dokumenten eine Liste, die uneingeschränkt lang sein kann. Veränderung der Anzahl von DRL-Dateien erfordert nicht mehr den Eingriff in Java Code. Durch verwenden eines Prefix im Dateinamen konnte eine Lösung für das Aktivierung von Regeln gefunden werden. Ausgelesen werden die Dokumente mittels `GitController` Klasse. Diese Klasse filtert bei der Suche alle Dateinamen aus, die mit einem bestimmten Prefix anfangen.

Ein letztes Kriterium im Konzept ist die Benutzerfreundlichkeit. Es konnte kein Verfahren angewendet werden, um diese objektiv zu bestimmen. jBPM war bis jetzt nur dem Entwickler zugänglich. Ein Zugriff außerhalb von MeinKOMSOS Systems konnte noch nicht eingerichtet werden. Eine öffentliche URL für den Zugriff auf jBPM wurde bereits beantragt, konnte aber zum Abschluss dieser Arbeit nicht umgesetzt werden.

Somit konnten nur Aussagen vom Autor dieser Arbeit getroffen werden.Im jBPM lassen sich viele Bereiche ein- und ausblenden, was mehr Platz für die relevanten Werkzeuge verschafft. Zudem lassen sich die angezeigten Fenster im jBPM skalieren. Das erstellen der BPMN-Modellen hat sich as sehr effektiv und effizient erwiesen. Die Schnellauswahl der Elemente ist sehr nützlich und verkürzt die Modellierung enorm. Während der Validierung werden eindeutige Fehlermeldungen eingeblendet, die selbsterklärend sind oder in Hilfsliteraturen nachgeschaut werden können. Es besteht die Möglichkeit zwischen geöffneten Dokumenten zu wechseln, ohne diese Suchen zu müssen, dank einer Schnellauswahl für geöffnete Fenster. Das Arbeiten mit dem Managementtool erwies bis jetzt nur kurze Latenzzeiten, wodurch sich die Bedingung flüssig anfühlt. Aufgrund der fehlender objektiver Beurteilung fehlt eine souveräne Bewertung der Benutzerfreundlichkeit. Die Wahrnehmungen vom Autor diese Arbeit sind jedoch sehr positiv.

Tabelle 5.1 zeigt eine Zusammenfassung des Vergleichen zwischen Konzept und Umsetzung.

Tabelle 5.1: Übersicht des Vergleichs

Gefordertes Kriterium	**Umsetzung**
Implementiert als Portlet	Nicht erfüllt
Prozess Modellierungstool	Erfüllt
Regel Modellierungstool	Nicht erfüllt
Validierung	Erfüllt
Modell Transformation	Teilweise erfüllt
Betriebsfreigabe	Erfüllt
Sichere Verwaltung	Erfüllt
Benutzerfreundlichkeit	Nicht objektiv

5.2 Kritik, Probleme und Lösungen

Kritisch zu beurteilen ist auch die Vernachlässigung einer Betrachtung der Performance von jBPM. Das Nutzen eines eigenständigen System verbrauch wesentlich mehr Ressourcen. Eine Lösung zur Integration von jBPM als Portlet wäre also immer noch wünschenswert.

Die Motivation dieser Arbeit war die Verwaltung von Regeln sicherer, benutzerfreundlicher und effizienter zu gestalten. Die Kriterien dafür wurden zum Teil zu Allgemein (Benutzerfreundlichkeit) oder zu spezifisch (Validierung) formuliert. Während der Erstellung dieser Kriterien konnte nur eine theoretische Betrachtung der Aktivitäten gemacht werden, es fehlt am Bezug auf die Praxis. Sofern jBPM in den Betrieb genommen wird, sollte eine erneute Betrachtung der Kriterien erfolgen. Dazu sollten die Nutzer befragt werden, um eventuelle auf unentdeckte Perspektiven aufmerksam zu machen.

Für das Editieren von Regeln konnte keine grafische Lösung eingeführt werden. Der angebotene Regeln-Editor erleichtert zum Teil das Erstellen von Regeln, ist aber noch auf den Code fokussiert. Der erforderliche Grad an technischer Kenntnisse der Nutzer konnte somit nicht gesenkt werden.

Werden DRL-Dateien erstellt, die vorerst von MeinKOSMOS ignoriert werden sollen, so wird im Dateinamen ein Prefix verwendet. Diese Methode ist simpel, zugleich aber auch fehleranfällig, z. B. durch inkorrekten Prefix im Namen. Es wäre zu prüfen, ob eine andere Lösung für das Problem besser geeignet wäre.

Die umgesetzte Schnittstelle kann in Frage gestellt werden. Umleitung auf einen lo-

kalen Repository gestalten die Verwaltung komplizierter. Nach jeder Veränderung muss ein Update der lokalen Kopie stattfinden, damit die Neuerungen im MeinKOSMOS erkannt werden. Der Versuch einer direkten Verbindung per SSH- oder GIT-Protokoll sind gescheitert. Eine SSH-Verbindung konnte hergestellt werden, es fehlte jedoch an Berechtigung für das Ausführen von Kommandos. Teile dieser Funktion befinden sich auskommentiert in der `GitController` Objektklasse. Eine direkte Verbindung ist effizienter. Sollte die aktuelle Lösung akzeptiert werden, so empfehlt sich die Einführung eines automatischen Update für den lokalen Repository.

6 Zusammenfassung und Ausblick

Die Zusammenfassung der Arbeit wird in diesem Kapitel behandelt. Anschließend wird ein kurzer Ausblick für das hier erarbeitete Konzept gegeben.

6.1 Zusammenfassung

Zu Beginn dieser Arbeit wurden grundlegende Kenntnisse erläutert, die zum Verständnis dieser Arbeit dienen sollen. Zuerst gab es eine kleine Vorstellung vom KOSMOS-Projekt, welches von der Universität Rostock geleitet wird. Die Intention dieses Projektes und die verbundenen Ziele wurden damit kurz erläutert. Anschließend wurde MeinKOSMOS, die technische Umsetzung eines Portals zum KOMSOS-Projekt, vorgestellt. Zudem wurde der Begriff Portal beschrieben, um ein besseres Verständnis dafür zu erhalten. In den letzten Kapiteln der Grundlagen wird mehr auf die Personalisierungskomponente mit ihrer Konzeption eingegangen. Diese Komponente bildet das Herzstück der Personalisierung innerhalb von MeinKOSMOS. Die Basis für die Personalisierungskomponente bildet ein regelbasiertes System (Drools Rule Engine), welches grundlegend erläutert wurde. Die wichtigen Bestandteile der Komponente wurden detailliert benannt und beschrieben, weil sie die Basis für diese Arbeit darstellen. Dazu zählt die Beobachterkomponente, die Empfehlungskomponente (`MainController` und `PortletController`) und das Nutzerprofil. Eine weitere wichtige Rolle zum Verständnis der Personalisierungskomponente ist der Einblick in die BPMN Spezifikationssprache, da sämtliche Personalisierungsprozesse mittels BPMN erstellt worden sind.

Im danach folgendem Kapitel 3 wurde ein Konzept zur Verbesserung der Verwaltung von Personalisierungsregeln entwickelt. Als Ergebnis liefert das Konzept Ansätze für ein Managementtool zur Lösung des Problems. Im Vorfeld wurden allgemeine Anforderungen festgelegt, die stets im Konzept gelten sollten. Anschließend wurde das Konzept in zwei Dimensionen (menschliche und technische) ausgearbeitet. Die menschliche Dimension

(Kapitel 3.2) befasste sich mit der Interaktion zwischen dem Menschen und dem Managementtool. Es gab einen Einblick in die Aktivitätsfelder, Aufgaben und Arbeitsschritte des Nutzers innerhalb des Konzeptes, sowie die daraus resultierenden Anforderungen und Kenntnisse des Nutzers, die er besitzen sollte. Die zweite technische Dimension (Kapitel 3.3) befasste sich vorerst mit dem genauen Einblick in die technische Architektur von MeinKOSMOS. Dabei wird die Idee vom Portlet erklärt und anschließend wird die konzeptionelle Einbindung in das MeinKOSMOS-Portal von einem Managementtool nur als ein Portlet vorgestellt. Im letztem Abschnitt 3.4 des Kapitels wurden erarbeitete Kriterien des Konzepts zusammenfassend in einer Übersichtstabelle (3.1) dargestellt. Diese Tabelle ist für den späteren Vergleich wichtig.

Die Umsetzung des Konzepts wurde im Kapitel 4.1.2 anschaulich dargestellt und bestand aus drei Schritten. Als erstes wurde die Entscheidung getroffen ein bereits vorhandenes Business Process Managementtool namens jBPM für die Umsetzung zu verwenden. Die Begründung der Entscheidung befindet sich im selben Kapitel. Im zweiten Schritt wurde das System eingerichtet, es fand eine Integration in MeinKOSMOS statt. Aufgrund erheblicher Schwierigkeiten bei der Integration von jBPM als Portlet in MeinKOSMOS (Tomcat Server) musste das Konzept angepasst werden. Nach der Anpassung wurde jBPM als eigenständiges System (Wildfly Server) eingerichtet und man konnte einen Einblick in jBPM machen. Dieser Einblick verschafft eine kleine Übersicht der Funktionen, die für das Konzept relevant sind, um die Kriterien zu erfüllen. Anschließend im Kapitel 4.3 wurde eine Schnittstelle zwischen jBPM und der Empfehlungskomponente realisiert. Für den Zweck wurde `GitController` implementiert. Diese Objektklasse wird im `MainController` eingesetzt. Die Hauptaufgabe von `GitController` ist es die DRL-Dateien aus dem Git Repository von jBPM zu filtern und an den`MainController` zu übergeben. Somit wird der `GitController` als Schnittstelle angesehen.

Im Kapitel 5 erfolgt eine kritische Betrachtung der Umsetzung. Der erste Abschnitt beschäftigt sich mit dem Vergleich zwischen Konzept und Umsetzung. Den Leitfaden für den Vergleich bietet die zuvor vorgestellte Übersichtstabelle (5.1) der Kriterien. Im Abschluss (Kapitel 5.2) wurde das Konzept und die Umsetzung kritisch bewertet. Es wird Kritik an den festgelegten Kriterien des Konzepts ausgeübt. Zudem wird das Scheitern einer Implementierung als Portale erneut betrachtet. Es folgt zudem eine Bewertung der

angebotenen Werkzeuge und Tools von jBPM. Es gibt zudem Kritik an den vernachlässigten Themen wie die Performance oder der Umstieg auf BPMN-Prozesse statt Regeln.

6.2 Ausblick

Zur Zeit wird mit Hochdruck daran gearbeitet eine geeignete Methode für die Personalisierung innerhalb von MeinKOSMOS umzusetzen. Neben der hier vorgestellten Lösung existiert noch ein prototypischer Ansatz, der einen Recommender System benutzt. In der nächsten Zeit wird sich herausstellen welche Lösung sich durchsetzen wird[Fil15]. Das

hier erarbeitete Vorgehen schafft eine Grundlage zur Verwaltung der Regeln und zeigt eine prototypische Vorgehensweise. Weil jBPM viel mehr anbietet, muss in der Zukunft analysiert werden, wie man das Potential weiter ausnutzen könnte. Die Umsetzung ei-

ner direkten Verbindung zum Git Repository würde das Vorgehen erheblich effizienter machen, daher sollte erneut versucht werden diese zu realisieren. Neben den Überlegun-

gen welche Portlets und Informationen den Nutzern empfohlen sollten, werden weitere gemacht, um die Frage zu beantworten, inwiefern die Portlets an welche Stelle und Position zur Oberfläche von MeinKOSMOS am optimalsten eingebunden werden. jBPM

ist zum Teil auch ein Kollaborationssystem, welches typische Funktion wie Aufgabenvergabe oder Aufgabenüberwachung an verschiedene Nutzer oder Gruppen ermöglicht. Es sollte geprüft werden, ob diese Funktionen für weitere Benutzergruppen relevant wären. Anstelle des Bearbeiters der Regeln könnten auch weitere Nutzergruppen das System verwenden, z. B. Dozenten, um eigene Empfehlungen für die Studenten zu generieren oder beispielweise Professoren, die bestimmte Aufgaben an Dozenten weitergeben oder übertragen könnten. Aufgrund der kleinen Anzahl von Portlets, die sich personalisieren

lassen, existieren nicht viele Personalisierungsregeln. Viele der Portlets sind noch in der Entwicklungsphase und können mit der Empfehlungskomponente nicht verknüpft werden. Sofern die Portlets die Entwicklungsphase verlassen, wäre es interessant zu sehen,

wie jBPM in der Praxis genutzt wird. Die Praxis würde verständlicherweise einen Einblick gewährleisten, um die Vorgehensweise/n besser validieren zu können. Seit 2008 wird

jBPM stetig weiter entwickelt. Es wäre somit wichtig diese Entwicklung nicht zu vernachlässigen. Es sollte in Zukunft geprüft werden, ob die weitere Updates mehr Nutzen für MeinKOSMOS hervorbringen wird und diese gegebenenfalls zu installieren. Da jBPM

in der ersten Linie für die Verwaltung von Geschäftsregeln genutzt wird, sollte in Betracht gezogen werden auf die Regeln zu verzichten und stattdessen die Personalisierung nur über die Prozesse realisieren zu lassen. Dies würde die Verwaltung der Regeln umso benutzerfreundlicher gestalten.

Anhang A: GitController Objektklasse

Das folgende Listing zeigt den kompletten Code von `GitController`. Diese Objektkasse wurde zum Auslesen eines lokalen Git Repositories erstellt.

```
1  package de.kosmos.liferay.support.controller;
2
3  import java.io.ByteArrayOutputStream;
4  import java.io.File;
5  import java.io.IOException;
6  import java.io.InputStream;
7  import java.util.ArrayList;
8  import java.util.List;
9
10 import org.eclipse.jgit.lib.Ref;
11 import org.eclipse.jgit.api.Git;
12 import org.eclipse.jgit.lib.Constants;
13 import org.eclipse.jgit.lib.ObjectId;
14 import org.eclipse.jgit.lib.ObjectLoader;
15 import org.eclipse.jgit.lib.Repository;
16 import org.eclipse.jgit.revwalk.RevCommit;
17 import org.eclipse.jgit.revwalk.RevTree;
18 import org.eclipse.jgit.revwalk.RevWalk;
19 import org.eclipse.jgit.treewalk.TreeWalk;
20 import org.eclipse.jgit.treewalk.filter.PathFilter;
21 /**
22  * Diese Klasse wird zum Auslesen eines lokalen Git Repo verwendet.
23  * @author Dorian Wojda
24  *
25  */
26 public class GitController{
```

```
27    private String myLocation = null;
28    private String prefixToIgnore = null;
29    private String fileExt = null;
30
31    /**
32     * Konstruktor der Klasse
33     * @param gitLocation
34     * @param toIgnore
35     * @param fileEXT
36     */
37    public GitController (String gitLocation,String toIgnore,String
       fileEXT){
38      this.myLocation = gitLocation;
39      this.prefixToIgnore = toIgnore;
40      this.fileExt = fileEXT;
41
42    }
43
44    public void setPrefixToIgnore(String toIgnore){
45      this.prefixToIgnore = toIgnore;
46    }
47
48    public void setFileExt(String ext){
49      this.fileExt = ext;
50    }
51
52    /**
53     * Gibt eine Liste mit globalen Pfaden zu gesuchenten Dateitypen
       aus
54     *
55     * @return
56     * @throws IOException
57     */
58    public List<String> getFilesPath() throws IOException{
59      List<String> tmpList = new ArrayList<String>();
60      //local location of Git repo
61          File gitWorkDir = new File(this.myLocation);
62          Git git = Git.open(gitWorkDir);
```

```
63          Repository repo = git.getRepository();
64          Ref head = repo.getRef("HEAD");
65         //a RevWalk allows to walk over commits based on some
       filtering that is defined
66          RevWalk walk = new RevWalk(repo);
67          RevCommit commit = walk.parseCommit(head.getObjectId());
68          RevTree tree = commit.getTree();
69          //http://stackoverflow.com/questions/19941597/jgit-use-
       treewalk-to-list-files-and-folders
70          //now use a TreeWalk to iterate over all files in the Tree
       recursively
71          TreeWalk treeWalk = new TreeWalk(repo);
72          treeWalk.addTree(tree);
73          treeWalk.setRecursive(false);
74          while (treeWalk.next()) {
75                if (treeWalk.isSubtree()) {
76                    treeWalk.enterSubtree();
77                } else {
78                   String currentFilePath = treeWalk.getPathString();
79                     String tmp = treeWalk.getPathString();
80                     String fileName = treeWalk.getNameString();
81                     if(tmp.substring(tmp.length() - (this.fileExt.
       length()+1)).equals("."+this.fileExt) && !fileName.substring(0,1).
       equals(this.prefixToIgnore)){
82                     tmpList.add(this.myLocation+currentFilePath);
83                       }
84                  }
85           }
86       return tmpList;
87    }
88
89    /**
90     * Gibt eine oder mehrere Dateien laut Parameter in einer Liste aus
91     *
92     * @param fileNeeded : Filename mit .drl
93     * @return
94     * @throws IOException
95     */
```

```
96    public List<String> getFilePath(String fileNeededWithExt) throws
      IOException{
97     List<String> tmpList = new ArrayList<String>();
98
99     //local location of GIT-Repo-Copy of jBPM-Repo
100         File gitWorkDir = new File(this.myLocation);
101         Git git = Git.open(gitWorkDir);
102         Repository repo = git.getRepository();
103         Ref head = repo.getRef("HEAD");
104         //a RevWalk allows to walk over commits based on some
      filtering that is defined
105         RevWalk walk = new RevWalk(repo);
106         RevCommit commit = walk.parseCommit(head.getObjectId());
107         RevTree tree = commit.getTree();
108         //http://stackoverflow.com/questions/19941597/jgit-use-
      treewalk-to-list-files-and-folders
109         //now use a TreeWalk to iterate over all files in the Tree
      recursively
110         TreeWalk treeWalk = new TreeWalk(repo);
111         treeWalk.addTree(tree);
112         treeWalk.setRecursive(false);
113         while (treeWalk.next()) {
114             if (treeWalk.isSubtree()) {
115                 treeWalk.enterSubtree();
116             } else {
117                 String currentFilePath = treeWalk.getPathString();
118                   String tmp = treeWalk.getPathString();
119                   String fileName = treeWalk.getNameString();
120                 if(tmp.substring(tmp.length() - 4).equals(".drl") &&
      !fileName.equals(fileNeededWithExt)){
121                     tmpList.add(this.myLocation+currentFilePath);
122                     }
123                 }
124           }
125     return tmpList;
126   }
127 }
```

Anhang B: Zusatzfunktion von MainController

Nachfolgendes Listing zeigt einen Java Code Abschnitt aus dem `MainController`, indem die `readAllRulesFromRepo` Funktion vorgestellt wird. Diese Funktion wurde dafür implementiert Dateien aus einem lokalem Repository in Drools Rule Engine einzuladen und auszuführen.

```
1  private static RuleBase readAllRulesFromRepo() throws Exception {
2    //lade den lokalen repo
3    // wichtig: am ende des pfades muss \\ für win oder / für unix
       stehen
4    GitController git = new GitController("C:\\Users\\rs\\git\\
       jBPMRules\\","_","drl");
5    List<String> allDRLFiles = git.getFilesPath();
6    //Use package builder to build up a rule package.
7    PackageBuilder builder = new PackageBuilder();
8    PackageDescr packageDescr = new PackageDescr("de.kosmos.liferay.
       support.controller");
9
10   for(int i = 0;i< allDRLFiles.size();i++){
11     File f = new File(allDRLFiles.get(i));
12     Reader source = new InputStreamReader(new FileInputStream(f) );
13     builder.addPackage(packageDescr);
14     //this wil parse and compile for each drl
15     builder.addPackageFromDrl( source );
16     System.out.println("In Drools wird geladen: "+allDRLFiles.get(i))
       ;
17
18   }
```

```
19    //get the compiled package (which is serializable)
20    org.drools.core.rule.Package pkg = builder.getPackage();
21    //add the package to a rulebase (deploy the rule package).
22    RuleBase ruleBase = RuleBaseFactory.newRuleBase();
23    ruleBase.addPackage( pkg );
24    return ruleBase;
25  }
```

Anhang C : jBPM Installationsanleitung

An dieser Stelle wird eine Schritt für Schritt Anleitung gezeigt, wie jBPM installiert wurde. Zuerst wird eine Übersicht der Schritte erfolgen, anschließend werden alle Schritte erläutert.

1. **jBPM Installationspaket am Speicherort JBPM_HOME entpacken**
2. **In der `build.xml`-Datei einen Port-offset setzen**
3. **In der `build.xml`-Datei Port für die Success-Feedback-Funktion um den Offset verändern.**
4. **Login Accounts anlegen / bereinigen**
5. **Mittels Konsole jBPM ohne Eclipse installieren**
6. **Mittels Konsole jBPM ohne Eclipse starten**
7. **Mittels Konsole jBPM ohne Eclipse stoppen**

---

1. jBPM Installationspaket am Speicherort JBPM_HOME entpacken

Es bestehen zwei Optionen, wie man jBPM installieren kann. Die erste Option wäre das Kopieren von **jbpm**-Ordner, der sich auf der hier beigefügt DVD befindet (**DVD : KONZEPT / FERTIGE INSTALLATION**). In diesem Ordner befindet sich die eine konfigurierte Version von jBPM, wie sie auf dem MeinKOSMOS System zu finden ist. Sollte man sich für diesen Weg entschiednen, so können die Schritte **2.** bis **5.** übersprungen werden.

Die zweite Option ist es, das Installationspaket von der jBPM-Webseite[1] runter zu laden.

[1] http://www.jbpm.org/download/download.html

Unabhängig von der gewählten Methode muss der Installationsordner am zentralem Ort gespeichert werden. Dieser Ort wird hier als JBPM_HOME bezeichnet. Auf dem MeinKOSMOS-System liegt jBPM in **C:\Users\rs\jbpm**.

---

2. In der `build.xml`-Datei einen Port-offset setzen

Ein Port-offset muss gesetzt werden, damit Liferay und jBPM nicht in Konflikt geraten. Dazu die `build.xml`-Datei öffnen und den folgenden Code (Zeile `8`) hinzufügen:

```
1  <target name="start.jboss">
2   <property name="jboss.full.path.win" location="${jboss.home}/bin/
     standalone.bat" />
3   <exec executable="${jboss.full.path.win}" spawn="yes" osfamily="
     windows">
4    <env key="JAVA_OPTS" value="-XX:MaxPermSize=256m -Xms256m -
     Xmx512m" />
5    <arg value="-b" />
6    <arg value="${jboss.bind.address}" />
7    <arg value="--server-config=standalone-full.xml" />
8    <arg value="-Djboss.socket.binding.port-offset=1000" />
9    <arg value="-Dorg.kie.demo=true" />
10   <arg value="-Dorg.kie.example=false" />
```

Die Datei `build.xml`-Datei befindet sich im Hauptordner von jBPM. Durch das Einfügen von `Djboss.socket.binding.port-offset=1000` wird zu allen benutzen Port `1000` addiert. Diese Zahl kann beliebig gewählt werden, für die Installation wurde 1000 ohne einen besondern Grund gewählt.

---

3. In der `build.xml`-Datei Port für die Success-Feedback-Funktion um den Offset verändern.

Durch einen Offset ändert sich der standard HTTP-Port (8080), dieser muss nun angepasst werden.

```
1  <waitfor maxwait="5" maxwaitunit="minute" checkevery="10"
       checkeveryunit="second" timeoutproperty="jboss.timeout">
2    <socket server="${jboss.bind.address}" port="9080" />
3  </waitfor>
4  <fail if="jboss.timeout" message="JBoss application server did not
       start within 5 minutes"/>
```

In Zeile 2 wurde aus dem Port `8080` der neue Port `9080` gemacht, damit kommt es nicht zur fehlerhaften Meldung, wenn der Server gestartet ist.

---

4. Login Accounts anlegen / bereinigen

Vor dem Start sollen aus Sicherheitsgründen die üblichen Login Accounts geändert werden. Dies kann in der
C:\Users\rs\jbpm\wildfly-8.1.0.Final\standalone \configuration\users.properties
Datei gemacht werden. Die Accounts werden in der Form `USER=PASSWORT` hinterlegt.

```
1  admin=???
```

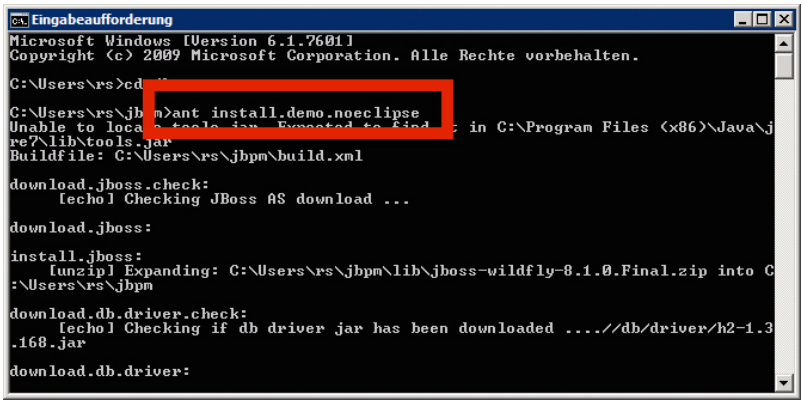
Es existiert nur ein Benutzer auf dem MeinKOMSO-System, der sich einloggen kann. Aus Sicherheitsgründen wurde das Passwort durch `???` ersetzt. Um das Passwort einzusehen, muss in der Datei nachgeschaut werden.
Jedem Nutzer muss eine Rolle zugewiesen werden. Die Rollen können in der Datei
C:\Users\rs\jbpm\wildfly-8.1.0.Final\standalone \configuration\roles.properties
angepasst werden.

```
1  admin=admin,analyst,kiemgmt
```

Die aktuelle Datei `roles.properties` beinhaltet nur die Einstellung für den Nutzer **admin**. Sollten neue Benutzer dazu kommen, so müssen die Rollen hier zugewiesen werden.

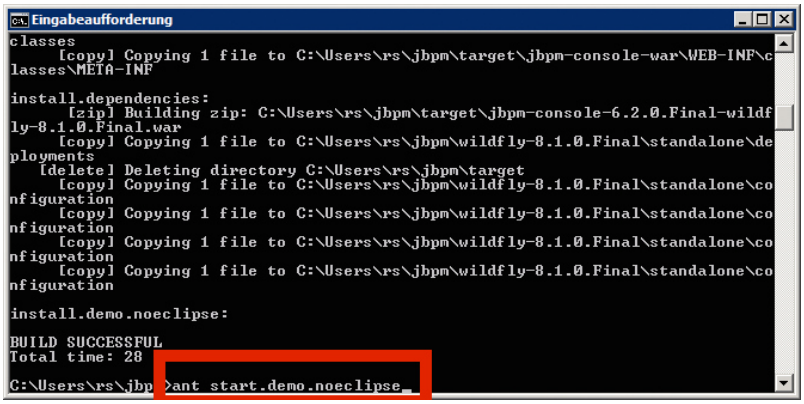
5. Mittels Konsole jBPM ohne Eclipse installieren

Zu beachten ist, das Apache Ant[2] zum Ausführen der Befehle installiert sein muss. Durch den Befehl `ant install.demo.noeclispe` wird jBPM installieren



6. Mittels Konsole jBPM ohne Eclipse starten

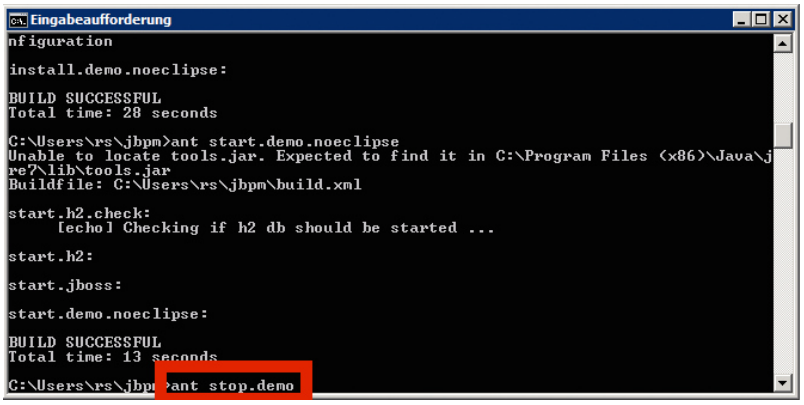
Wurde jBPM installiert, kann es mit dem Befehl `ant start.demo.noeclispe` gestartet werden.



[2] http://ant.apache.org

7. Mittels Konsole jBPM ohne Eclipse stoppen

Soll jBPM ausgeschaltet werden, so muss der Befehl `ant stop.demo` benutzt werden.



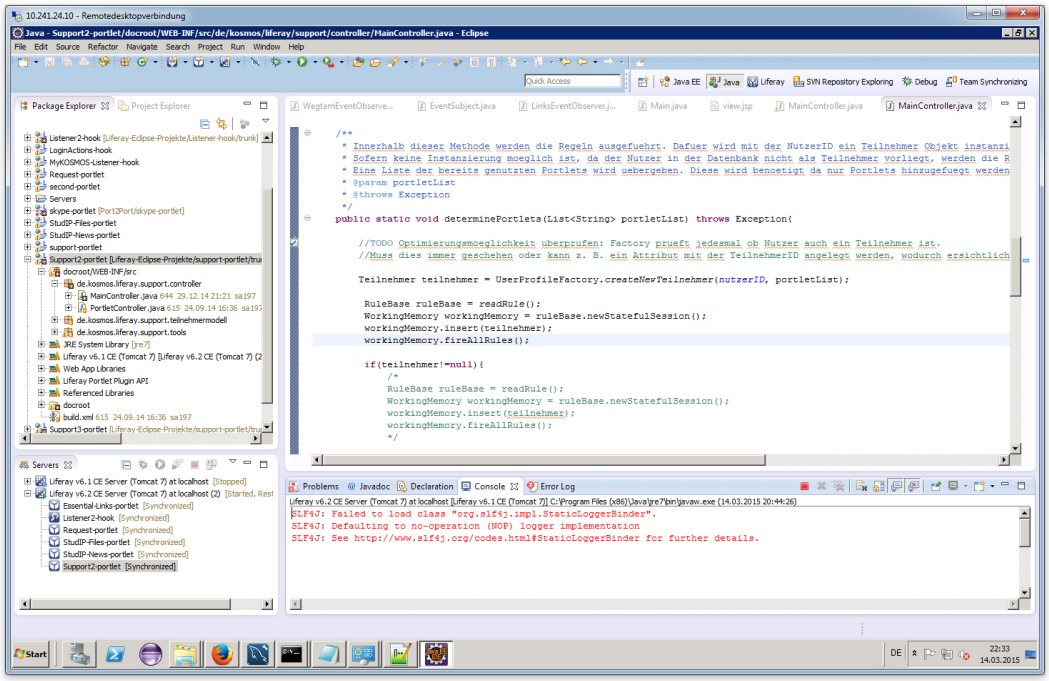
Anhang D : Konsolenausgabe jBPM als Portlet

Hier wird ein keiner Ausschnitt aus der Logdatei von Liferay gezeigt. Es wurde versucht jBPM als Portlet einzubinden. Ein *deploy* ist angekündigt (Zeile 2) worden, konnte aber nicht beendet werden. Es gab zudem keine Fehlermeldung. Die gesamte Logdatei ist auf der DVD (**DVD : / KONZEPT / jBPM ALS PORTLET / log.txt**) zu finden. Im selben Speicherort auf der DVD befindet sich der Ordner **jbpm-console-Portlet**. In diesem Ordner befindet sich das Portlet, was hier versucht wurde einzubinden.

```
1  Mrz 04, 2015 12:41:32 PM org.apache.catalina.startup.HostConfig
       deployDirectory
2  INFORMATION: Deploying web application directory /Users/dorian/
       Downloads/liferay-portal-6.2-ce-ga2/tomcat-7.0.42/webapps/jbpm-
       console
3  Mrz 04, 2015 12:41:33 PM org.apache.catalina.startup.HostConfig
       deployDirectory
4  INFORMATION: Deploying web application directory /Users/dorian/
       Downloads/liferay-portal-6.2-ce-ga2/tomcat-7.0.42/webapps/
       marketplace-portlet
5  12:41:36,317 INFO  [localhost-startStop-1][HotDeployImpl:198]
       Deploying marketplace-portlet from queue
```

Anhang E : Support-Portlet Fehlerbehebung

In der folgenden Abbildung werden Fehler aus der ursprünglichen Portlet-Version gezeigt. Diese Schritte waren notwendig, um den Fehler zu beheben:

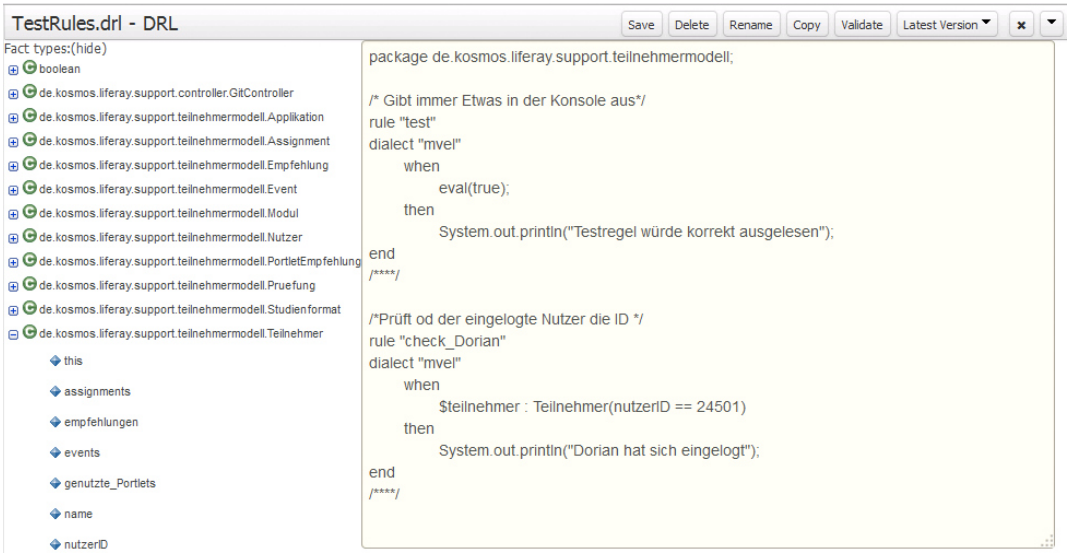


1. Entfernen von `slf4j-1.7.2.jar` aus der Library
2. Hinzufügen von `slf4j-nop.jar` in die Library

Anhang F : Test vom Konzept und MeinKOSMOS

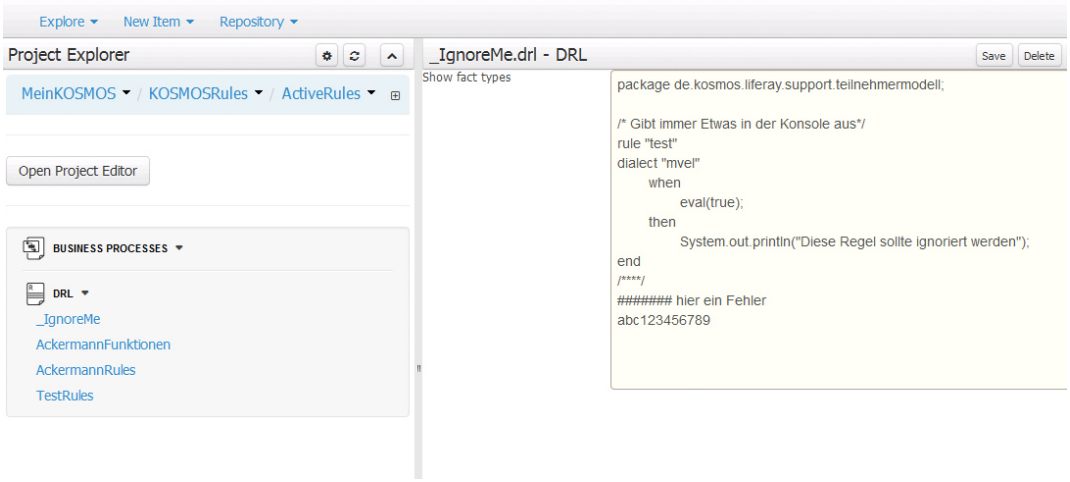
Für den Test wurden einige Regel in jBPM erstellt und anschließend über MeinKOSMOS und Support-Portlet ausgeführt. Die Folgenden Abbildungen dokumentieren den Test.

Zu Beginn wurden Textregeln erstellt. Diese Regeln waren in der Datei `TestRules.drl`. Die Regel `"test"` bekommt `true` als Fakt und wird daher immer ausgeführt. Die Ausführung dieser Regeln sollte den Text "Testregel wurde korrekt ausgelesenïn der Konsole von Liferay ausgeben. Die Regel `"check_Dorian"` nutzt das Nutzerprofil, um zu prüfen, ob Dorian, ein Nutzer mit der ID 24501, sich einloggt. Die ID des Nutzers wird aus der Datenbank von MeinKOSMOS ausgelesen.

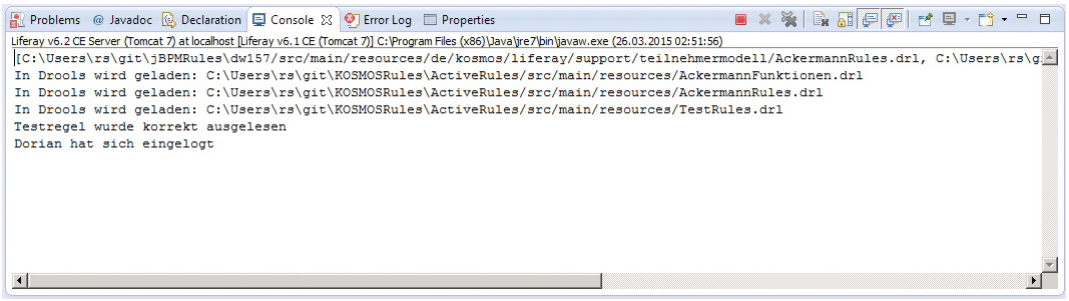


Als Nächstes wurde eine fehlerhafte Regeldatei angelegt und mit einem Prefix `"_"`

gekennzeichnet. Sollte die Datei eingelesen werden, dann entstehen Fehler, die in der Konsole zu sehen wären.



Nachdem die Regeln erstellt waren, wurde der Test durchgeführt, indem sich der Nutzer (ID 24501) eingeloggt hat. Folgendes Ergebnis lieferte die Konsole in Liferay (MeinKOSMOS):

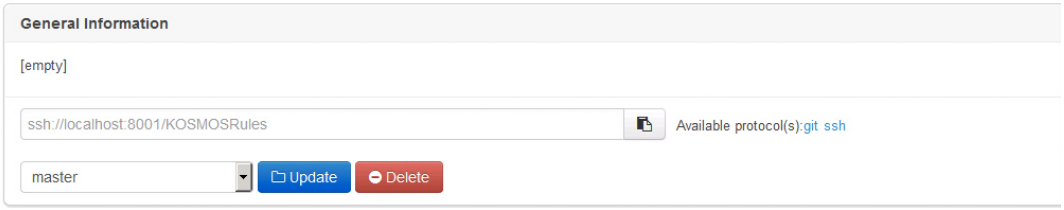


Das Testszenario ist zwar sehr minimalistisch, deckt aber alle implementierten Funktionen ab und bestätig einen fehlerfreien Einsatz von jBPM und MeinKOSMOS.

Anhang G : Anleitung zum Git

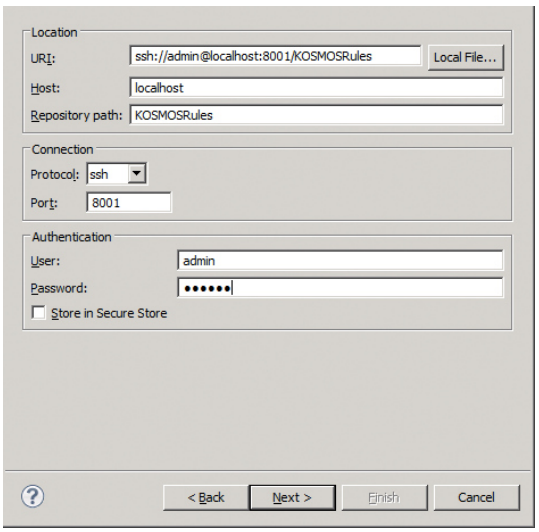
In dieser Anleitung wird kurz beschrieben, wie man eine Verbindung zum jBPM Git mithilfe von EGit[3] Addon herstellt.

Zum Klonen von einem Git wird zuerst die URI benötigt. Um dieser Auszulesen muss man in jBPM zu Git Übersicht navigiert werden (Authoring → Administration). Dort gelandet, kann man die URI zum Git leicht auslesen. Für den Import in EGit bietet sich das SSH Protokoll am besten an.



Die angezeigte URI kann kopiert und in EGit eingefügt werden.

[3]http://eclipse.org/egit/



Danach kann der Git wie ein gewöhnliches Repository genutzt werden.

Anhang H : Inhalt der DVD-ROM

- PDF dieser Arbeit
- Alle verwendete online und digitale Quellen
- Alle verwendeten Abbildungen
- Support3-Portlet angepasst an das Konzept
- Konfigurierter jBPM installationsordner
- Orginal jBPM installationsordner
- Konfigurierter jBPM-Portlet Ordner
- Orginal jBPM-Portlet Ordner
- jBPM Projekt im Git-Ordner
  - Regelndateien
  - BPMN2 Prozess-Dateien

Literaturverzeichnis

[Arb11] Arbeit, Bundesagentur für: Perspektive 2025: Fachkräfte Für Deutschland. (2011)

[bon15] Bonitasoft Compare Editions. http://www.bonitasoft.com/how-we-do-it/compare-editions. Version: 03 2015

[bpm15] BPMN 2 Poster. http://www.bpmb.de/index.php/BPMNPoster. Version: 03 2015, Abruf: 03.03.2015

[Bri11] Brian, Richard Sezvo; K.: Liferay in Action. MANNING SHELTER ISLAND, 2011

[BRS11] Bonn-Rhein-Sieg, Hochschule: Grundlagen regelbasierter Systeme. 2011. – Forschungsbericht

[Dir13] Dirk, Borchardt Ulrike; Sandkuhl Kurt; S.: Konzept zur Realisierung des KOSMOS Portals. (2013), November. http://www.kosmos.uni-rostock.de/fileadmin/KOSMOS/Kosmos_Dokumente/MeinKosmos-Konzept-final.pdf

[Fil15] Fila, Ingmar: Konzept und prototypische Imple- mentierung eines Recommendation Systems zur Unterstützung eines Lehr-Lern-Portals am Beispiel von myKOSMOS, Universität Rostock, Diplomarbeit, 2015

[Fre] Freund, Jakon: BPMN 2.0 "Sub-Classes". http://www.bpm-guide.de/2010/02/10/bpmn-2-0-sub-classes/

[HR85] HAYES-ROTH, FREDERICK: Rule-Bases System. In: Communications of the ACM (1985)

[inc15] inc., Liferay: Liferay Webseite. http://www.liferay.com/de/. Version: 03 2015

[jbp15a] Deploying kie-drools-wb on Tomcat. http://blog.athico.com/2014/04/deploying-kie-drools-wb-on-tomcat.html. Version: 03 2015

[jbp15b] How to Install jBPM6 on Tomcat7.x. https://apurvasingh67.wordpress.com/2014/03/03/how-to-install-jbpm6-on-tomcat7-x/comment-page-1/#comment-51. Version: 03 2015

[jbp15c] How to Set Up JBPM with Liferay. https://www.liferay.com/de/community/wiki/-/wiki/Main/How+to+Set+Up+JBPM+with+Liferay#section-How+to+Set+Up+JBPM+with+Liferay-Introduction. Version: 03 2015

[jbp15d] Liferay integration with jBPM. https://www.liferay.com/de/community/forums/-/message_boards/view_message/50034291#_19_message_38256634. Version: 03 2015

[Joc01] Jochen Rütschlin: Ein Portal – Was ist das eigentlich? In: Informatik 2001: Wirtschaft und Wissenschaft in der Network Economy – Visionen und Wirklichkeit. (2001)

[JR08] Jeffrey Rubin, Jared S. Dana Chisnell C. Dana Chisnell: Handbook of Usability Testing, 2nd Edition. Wiley, 2008

[Koc11] Kocian, Claudia: Geschäftsprozessmodellierung mit BPMN 2.0. Hochschule für angewandte Wissenschaften Neu-Ulm, 2011

[kos15] Über KOSMOS Webseite. http://www.kosmos.uni-rostock.de/ueber-kosmos/. Version: 03 2015

[Leh14] Lehner, Franz: Wissensmanagement: Grundlagen, Methoden und technische Unterstützung. Carl Hanser Verlag GmbH Co KG, 2014

[lif15] Besonderheiten des Portals (Liferay). https://www.liferay.com/de/products/liferay-portal/features/portal. Version: 03 2015

[OMG11] OMG: Business Process Model and Notation 2.0. 2011. – Forschungsbericht

[Rad12] Rademakers, Tijs ; TomBaeyens, JoramBarrez (Hrsg.): Activiti in Action. MANNING Manning Shelter Island, 2012

[red15] Red Hat JBoss BRMS Webseite. http://www.redhat.com/de/technologies/jboss-middleware/business-rules. Version: 03 2015

[Sam14] Samuel, Ackermann: Bildung von Nutzerprofilen aus dynamischen Nutzungsdaten im Lernportal des KOSMOS Projektes. Möllnerstraße 11, Universität Rostock, Diplomarbeit, September 2014

[Sch08] Schumm, David: Graphische Modellierung von BPEL Prozessen unter Verwendung der BPMN Notation, Universität Stuttgart, Diplomarbeit, 2008

[SL07] Steve Loughran, Erik H.: Ant in Action. Bd. 2. Manning Shelter Island, 2007

[Tea15] Team, Activiti.org: Activiti User Guide, 03 2015

[wet15] Bund-Länder-Wettbewerb Äufstieg durch Bildung: offene Hochschulen". http://www.wettbewerb-offene-hochschulen-bmbf.de. Version: 03 2015

[Whi04] White, Stephen A.: Introduction to BPMN. 2004. – Forschungsbericht

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Berlin, 26.03.2015

Dorian Wojda