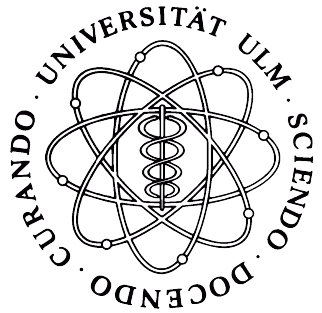
ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Entwicklung eines Workflow Clienten für flexible Ad-hoc-Änderungen auf Prozessinstanzebene



Diplomarbeit an der Universität Ulm, 2012

Vorgelegt von:
Florian Rapp
florian.rapp@uni-ulm.de

Gutachter:
Prof. Dr. Peter Dadam
Prof. Dr. Manfred Reichert

Betreuer:
Prof. Dr. Peter Dadam
Dipl. Inf. Andreas Lanz

Inhaltsverzeichnis

1 Einleitung

Das Bedürfnis, Geschäftsabläufe und Prozesse durch Informationssysteme zu unterstützen und dadurch effizienter zu gestalten, hat in den letzten Jahren immer mehr an Bedeutung gewonnen [5]. Der Ablauf und die Steuerung der Prozesslogik sollen durch geeignete Systeme und Softwarekomponenten verbessert werden und merklich an Flexibilität gewinnen. In diesem Kontext haben sich die *Workflow-Management-Systeme* (WfMS) hervorgetan. Diese Systeme ermöglichen es Prozessabläufe, die früher nur in den Köpfen der Mitarbeiter bestanden [9], durch ein gesamtheitliches System abzubilden, zu steuern und zu überwachen. Im Bereich der WfMS hat sich in den vergangenen Jahren in der Entwicklung viel getan. Die gesteigerten Anforderungen an die Flexibilität von Prozessabläufen hat den Fokus der Forschung auf möglichst flexible als auch robuste WfMS gelegt [6]. So konnten sich diese Systeme von fest verdrahteten Prozesssystemen weiterentwickeln zu Systemen die es ermöglichen, auf Anforderungen zur Laufzeit und möglicherweise auftretenden Problemen reagieren zu können, ohne die Integrität des laufenden Prozesses zu gefährden. Das grundlegende Prinzip dieser Systeme ist die Trennung von Prozessablauflogik und Anwendungslogik. Das WfMS steuert zentral den Prozess, welcher als Komposition von Anwendungsbausteinen verstanden werden kann. Diese grundsätzliche Trennung findet sich in allen WfMS-Lösungen der verschiedenen Anbieter wieder [5]. Ein weiteres gemeinsames Merkmal besteht in der Komposition der Prozesse. Jedes am Markt befindliche System erlaubt es einem geschulten Benutzer, Prozesse über eine spezielle Oberfläche zu erstellen. Über ein spezielles Repository können Anwendungsbausteine und Komponenten verwaltet und dem Prozess bereitgestellt werden, welcher diese dann verschaltet.

Im Optimalfall bieten die jeweiligen Systeme auch eine Option, um Prozesse zu ihrer Laufzeit, das heißt Ad-hoc, verändern zu können. Diese Änderungen erfolgen oftmals über das gleiche User Interface, über welches der Prozess initial konstruiert wurde. Die Zielsetzung hinter dieser Vorgehensweise ist die Erhaltung der Flexibilität. Durch das Vorhandensein aller Funktionen und Komponenten kann eine Prozessinstanz genauso vielseitig verändert werden wie ein neu entstehender Prozess.

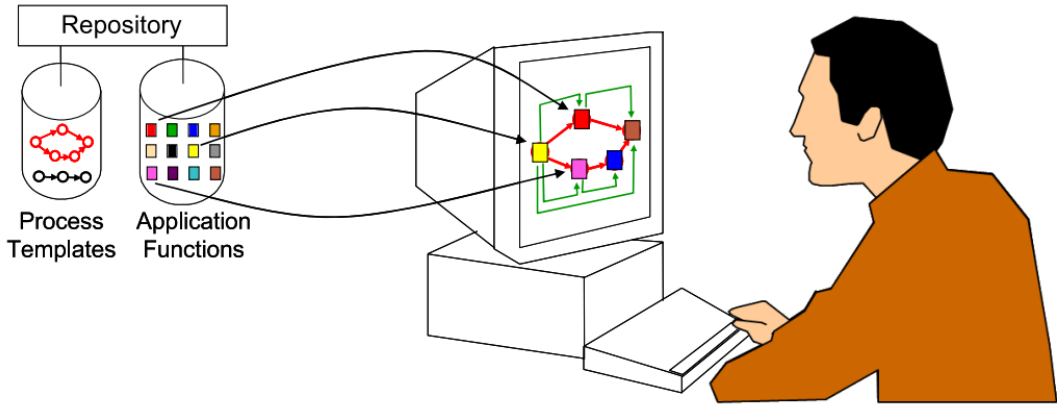


Abbildung 1.1: Experte komponiert Prozess [4]

Das Problem, dass die Erhaltung dieser Mächtigkeit mit sich bringt, ist die gleichbleibende Anforderung an den Bearbeiter der den Prozess verändern möchte. Oftmals werden Prozesse in den jeweiligen Firmen oder Institutionen von Experten erstellt und verwaltet. Soll zur Laufzeit jetzt ein Prozess verändert werden, ist in fast allen Fällen dieses Expertenwissen notwendig, um die gewünschte Änderung zu vollziehen. Aus diesem Umstand definiert sich die Anforderung an flexible Oberflächen und Systeme, die es auch dem weniger geschulten Anwender der WfMS ermöglicht, Prozessinstanzen verändern zu können, sollte dies notwendig werden (eine jeweilige Berechtigung zu einem solchen Vorgang vorausgesetzt). Bisherige Editoren sind in ihrer Vielfältigkeit und Komplexität nicht auf Endbenutzer abgestimmt und überfordern diese oftmals.
Die Schwierigkeit dieses Problem zu lösen besteht darin, dem Benutzer ein Werkzeug in die Hand zu geben, mit welchem er flexibel und zeitnah Änderungen an bestehenden Prozessen vornehmen kann bei gleichzeitiger Erhaltung der Mächtigkeit welche ein bestehendes WfMS bietet. Eine Analyse dieser Problemstellung sowie eines Lösungsansatzes soll die vorliegende Diplomarbeit liefern.

1.1 Ziel der Arbeit

Die Zielsetzung dieser Diplomarbeit ist es festzustellen, welche Anforderungen, Probleme und Herausforderungen bei der Erstellung eines benutzerfreundlichen Clienten für Ad-

hoc Änderungen auf Prozessinstanzebene bestehen, und in einem zweiten Schritt die Implementierung eines Prototyps, basierend auf einem bestehenden WfMS.
Ermittelt werden die grundlegenden Funktionalitäten und Möglichkeiten, die für einen normalen Benutzer bereitgestellt werden müssen, um flexible Änderungen durchführen zu können.

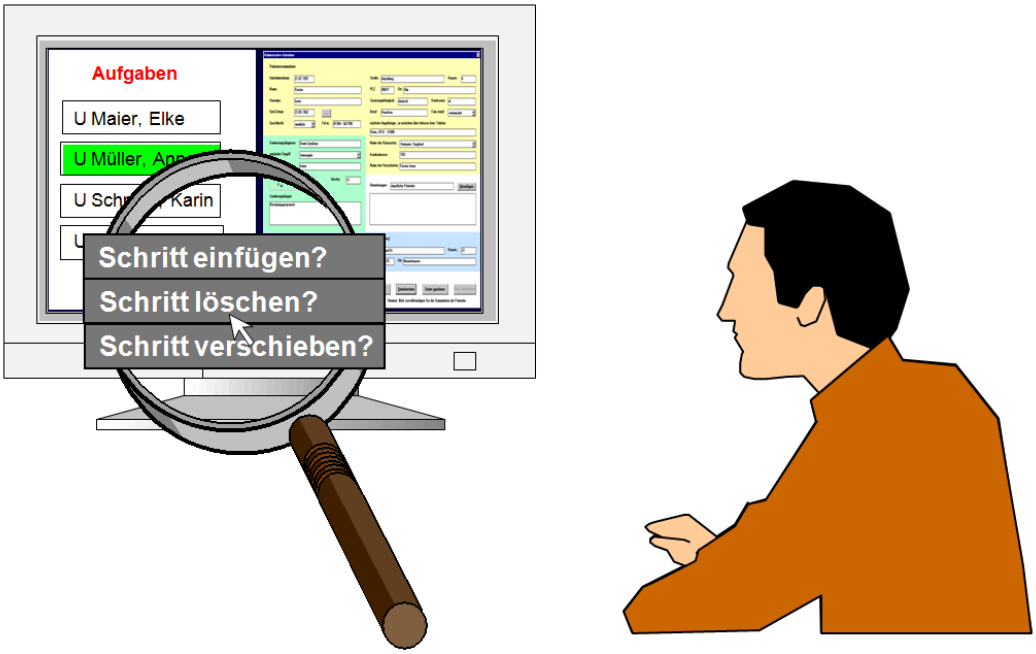


Abbildung 1.2: Benutzer kann Prozess dynamisch verändern [3]

Basierend darauf werden die technischen Anforderungen definiert die ein System erfüllen muss, um zum einen die benötigte Funktionalität für den Benutzer abzubilden, als auch zum anderen die strukturelle und semantische Integrität eines Prozesses zu gewährleisten.

Für die konkreten Lösungskonzeptionen sowie die Implementierung dient die *AristaFlow® BPM Suite* als Basis. Für weitere Referenzen in dieser Arbeit sei der prototypisch implementierte Client als "Flexclient"betitelt.
Am Ende der Arbeit soll eine Antwort auf die Frage: *Welcher Aufwand muss geleistet werden um ein benutzergerechtes Ändern von Prozessen zu ermöglichen?*, gegeben werden. Gezeigt werden soll, welche Schritte und Konzepte notwendig sind, um die gegebenen Anforderungen in Lösungskonzeptionen und schließlich in eine praktische Umsetzung zu transformieren.

1.2 Aufbau der Arbeit

Diese Arbeit ist unterteilt gemäß der nachfolgenden Gliederung. Kapitel 2 vermittelt Grundlagen, die im Kontext dieser Arbeit relevant sind. Abschnitt 2.1 soll einen grundlegenden Überblick über das Konzept der *Workflow-Management-Systeme* geben. Abschnitt 2.2 führt die *AristaFlow® BPM Suite* (AristaFlow) als Basis für die in dieser Arbeit entwickelten Konzepte ein.

Kapitel 3 soll einen kurzen Überblick über den Stand der Forschung im Bereich "flexibles Einfügen"geben. Behandelt werden verschiedene Ansätze, wie sie in bisherigen Lösungen zum Einsatz kommen. Ein Fazit zum aktuellen Stand schließt das Kapitel ab.

Kapitel 4 ermittelt die grundsätzlichen Anforderungen. Hier werden die benötigten Hauptfunktionen definiert und mit Hilfe von Beispielen verdeutlicht.

Kapitel 5 geht auf die genauen technischen Voraussetzungen ein, die ein System erfüllen muss. Behandelt werden hierbei die Gewährleistung eines korrekten Daten- und Kontrollflusses, die Zuordnung von Bearbeitern, Bereitstellung von Daten und Services sowie die Anforderungen an das User Interface.

Kapitel 6 führt eine Lösungskonzeption für die Umsetzung in einem realen System ein. Funktionen und Algorithmen, welche benötigt werden um die allgemeinen und technischen Anforderungen umzusetzen, werden detailliert definiert. Des weiteren wird gezeigt, welche Konzepte zur Speicherungen von Aktivitäten im Activity Repository erweitert oder neu erstellt werden müssen. Als letztes werden Konzepte für eine flexible und benutzerfreundliche Oberfläche definiert.

Kapitel 7 zeigt die Implementierung eines Prototypen auf Basis der *AristaFlow® BPM Suite*. Gezeigt werden soll, wie sich der Prototyp in die bisherige Architektur einordnet. Über einen speziellen Anwendungsfall wird demonstriert, wie der Benutzer mit dem System interagieren kann.

Kapitel 8 schließt die Arbeit mit einer Bewertung und einem Ausblick ab.

2 Grundlagen

Dieses Kapitel stellt die für den Kontext dieser Arbeit relevanten Grundlagen vor. Zuerst wird ein Überblick über *Workflow-Management-Syseteme* (WfMS) gegeben. Im zweiten Abschnitt wird speziell das WfMS *AristaFlow® BPM Suite* (AristaFlow) vorgestellt, welches die Basis für die Entwicklung unseres Prototypen und der dafür notwendigen Konzepte darstellt.

2.1 Workflow-Management-Systeme

Die Anforderung, Geschäftsprozesse durch spezielle Anwendungen zu unterstützen hat zu einer vermehrten Verbreitung von WfMS geführt [7]. Workflow-Management unterscheidet sich in seinem Ansatz einer prozessorientierten Anwendung deutlich von früheren, hart verdrahteten Lösungen der Prozessabbildung.
Diese konventionellen Implementierungen können charakterisiert werden durch ihre starre Abbildung der Prozesslogik direkt in den Anwendungsprogrammen [9]. Dieses Vorgehen führt zu dem Umstand, dass für jede Prozessinstanz auch eine Instanz des Anwendungsprogrammes gestartet werden muss. Gesteuert werden diese Prozessinstanzen generell durch eine zentrale Ablaufsteuerung. Die zentrale Steuerung legt somit fest, wann Arbeitsschritte auf den Rechnern der Anwender gestartet werden. Der Anwender hat somit keine Auswirkung auf die zeitliche Abfolge seiner Aufgaben. Jede seiner Aufgaben wird durch das Öffnen einer neuen Programminstanz automatisch gestartet. Dies verhindert das der Nutzer seine anstehenden Aktivitäten flexibel planen und ausführen kann. Auch auf etwaige Probleme auf Seiten der Arbeitsplatzrechner kann nicht flexibel reagiert werden. Sollte eine gestartete Programminstanz abstürzen, muss von der zentralen Steuerung umständlich darauf reagiert werden. Im Speziellen müssen alle zu erwartenden Problemfälle in der Ablaufsteuerung hinterlegt werden, sprich ebenfalls direkt in der Programmierung abgebildet sein.

Der Umstand, dass der komplette Ablauf eines Geschäftsprozesses hart verdrahtet im Programmcode abgebildet wird, sorgt für eine starre und unflexible Prozessunterstützung. Sollte ein Prozess geändert werden müssen, muss der zugrundeliegende Programmcode verändert werden. Dies bedeutet eine umständliche und aufwändige Änderung der gesamten Prozesssteuerung.

Moderne WfMS unterscheiden sich von dem Ansatz hart verdrahteter Lösungen deutlich. *Worfklow Management* bezeichnet die umfassende Verwaltung und Steuerung des kompletten Lebenszyklus von Geschäftsprozessen. Ein Workflow stellt dabei die schematische Abbildung eines realen Prozesses auf ein Informationssystem dar. Ein WfMS bietet verschiedene Funktionen und Komponenten, die die Erstellung, Verwaltung, Ausführung sowie Weiterentwicklung eines Prozesses unterstützen. Diese Komponenten lassen sich unterteilen in Modellierungs- und Laufzeitkomponenten [9].

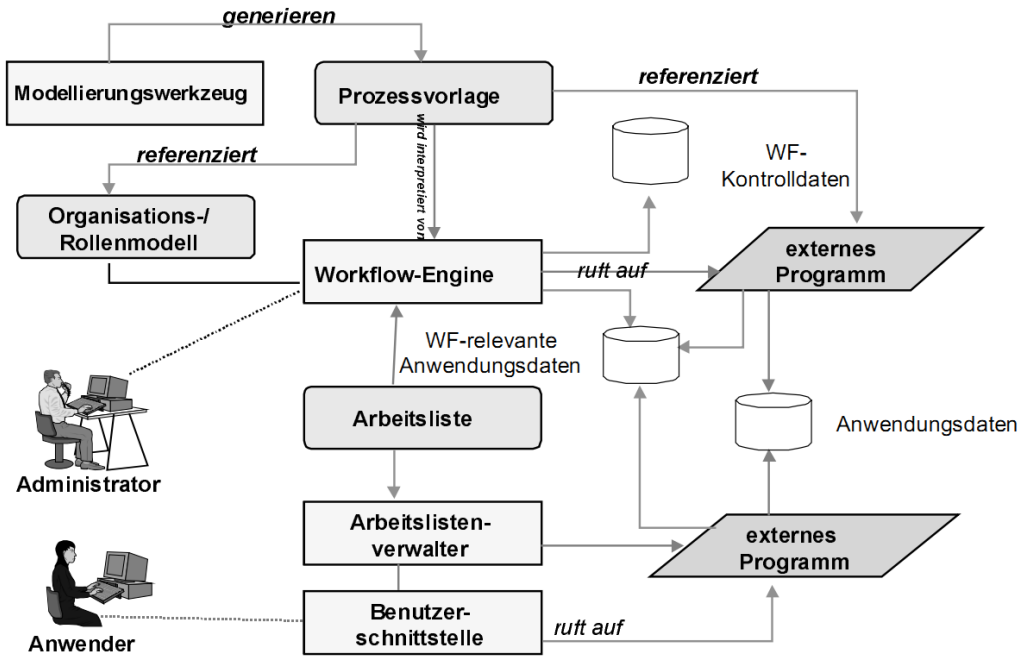


Abbildung 2.1: Komponenten eines WfMS [9]

2.1.1 Modellierung

Die Modellierung von Geschäftsprozessen stellt eine umfangreiche Menge von Anforderungen an ein Informationssystem. Komplexe Abläufe, Verzweigungen im Prozessschema und optionale Pfade müssen mitsamt ihren Bedingungen abgebildet werden können. Hierzu wird in WfMS meist eine grafische Darstellung gewählt, welche sich an der Graphentheorie orientiert. Die Darstellung als gerichteter Graph ermöglicht eine detaillierte Abbildung des realen Prozesses. Der wichtigste Aspekt bei der Modellierung, und auch bei der späteren Ausführung, ist die Unterscheidung zwischen Prozessablauf und Anwendungslogik (Abbildung 2.2). Auf der Ebene des Prozessablaufes wird der eigentliche, logische Prozess modelliert. Hierbei repräsentieren die einzelnen Knoten im Graph die Schritte und Aufgaben im Prozess. Jedem dieser Schritte wird dann ein Applikationsbaustein zugeordnet, welcher die eigentliche Funktionalität des Prozessschrittes beinhaltet. Ein solcher Baustein kann ein eigenes Programm oder den Aufruf einer externen Funktion beinhalten. Diese Trennung in zwei verschiedene Schichten ermöglicht eine weitgehende Modularisierung von Funktionalität. Sollte es nötig sein, einen Anwendungsbaustein zu verändern kann dies geschehen ohne etwas an der Struktur des Prozesses ändern zu müssen. Ebenfalls können Bausteine schnell und einfach ausgetauscht werden. Dieser Ansatz ähnelt dem Prinzip der *Web Services* [12]. Der Grundgedanke hierbei ist die Kapselung von Diensten in eigene Anwendungsbausteine, die dann nach Bedarf in die IT-Landschaft der Firma integriert werden können. *Web Services* können durch eine Beschreibungssprache wie WSDL [11] ihre Funktionen, Daten und benötigte Protokolle nach außen kenntlich machen und so als verteilter Service in andere Anwendungsprogramme integriert werden.
Die Anwendungsbausteine in WfMS werden über ein zentrales Repository verwaltet. In diesem wird ein Baustein mit allen seinen Eigenschaften wie Name, benötigte Parameter und realisierter Funktion definiert und kann dann zum Zeitpunkt der Modellierung abgerufen und einem Prozessschritt zugeordnet werden. Ein weiterer Aspekt der Modellierung ist die Zuordnung oder Festlegung von Bearbeitern. Über eine oftmals eigenständige Komponente, welche das Organisationsmodell einer Firma abbildet, kann ein WfMS dieses Modell dem Modellierenden verfügbar machen. Einem Prozessschritt können dann über eine definierte Metasprache eine Menge von möglichen, oder auch eine Menge von exakt definierten Bearbeitern zugeordnet werden.

Am Ende der Modellierung steht eine fertige Prozessvorlage, ein sogenanntes Prozesstem-

plate. Templates werden in einem eigenen Repository verwaltet und können später über Laufzeitkomponenten zur Ausführung gebracht werden.

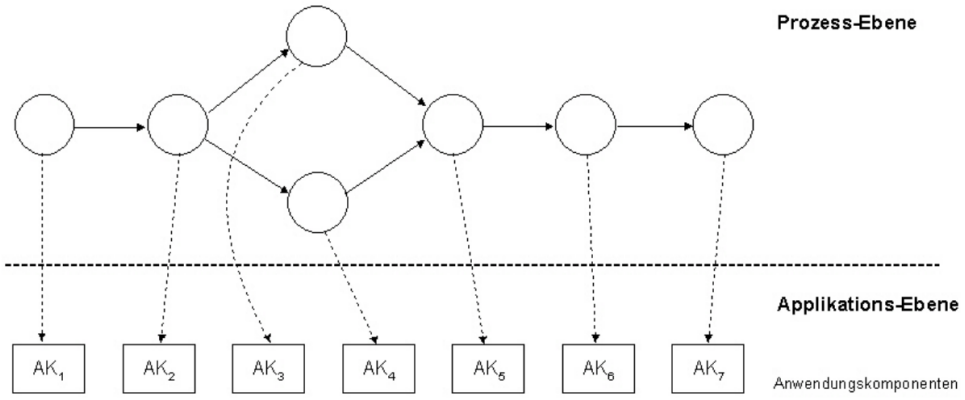


Abbildung 2.2: Prozesslogik und Applikationsbausteine werden getrennt. [5]

2.1.2 Laufzeitkomponenten

Der eigentliche Ablauf eines modellierten Workflows wird durch die Workflow-Engine (Abbildung 2.1) umgesetzt. Über eine Benutzerinterface kann ein dazu berechtigter Benutzer fertige Prozessvorlagen (Templates) einsehen und zur Ausführung bringen. Ausgeführt werden jedoch nicht die Templates an sich, sondern eine logische Kopie davon, welche als Prozessinstanz bezeichnet wird. Alle parallel laufenden Instanzen werden von der Workflow-Engine verwaltet und befinden sich in einem bestimmten Status. Je nachdem wie dieser Status definiert ist, wird für eine Instanz der nächste anstehende Schritt ermittelt und ausgewertet, welche Bearbeiter für die Ausführung der Funktionalität in Frage kommen (Abbildung 2.3).

Jeder für einen Schritt befähigte Benutzer bekommt diesen Schritt in einer persönlichen Arbeitsliste angezeigt. Startet ein Benutzer eine bestimmten Schritt, markiert das System diesen in den Arbeitslisten der anderen Bearbeiter oder blendet ihn aus. Für jeden gestarteten Schritt werden nun die benötigten Daten bereitgestellt und der hinterlegte Anwendungsbaustein aufgerufen. Dies geschieht bei Benutzeraktivitäten durch den Benutzer, bei automatischen Systemaktivitäten ermittelt und startet die Workflow-Engine den jeweiligen Applikationsbaustein selbst.

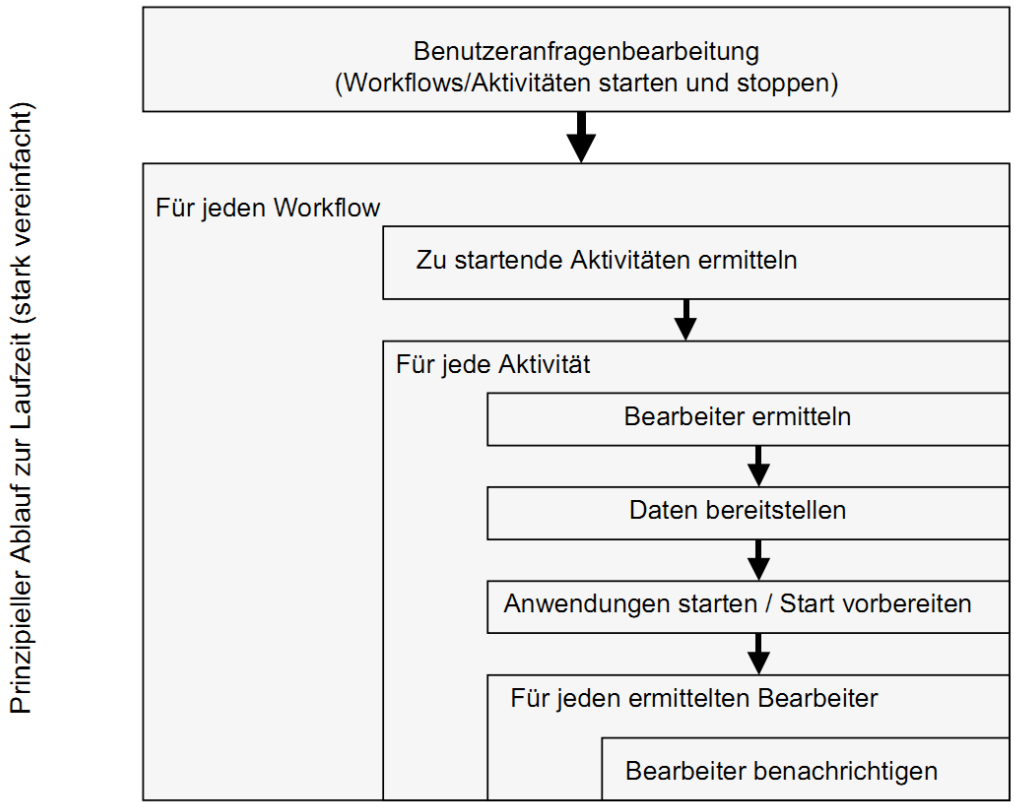


Abbildung 2.3: Vereinfachter Ablauf eines Workflows zur Laufzeit [9]

Der Lebenszyklus eines Workflow umfasst jedoch mehr als nur die Modellierung und Ausführung. Gerade Prozesse in Geschäftsbereichen, welche einem hohen Grad an Veränderung und Weiterentwicklung unterliegen, ist es notwendig, bestehende Prozesse ebenfalls weiterzuentwickeln, anzupassen und zu optimieren. Ein WfMS bietet deshalb im Optimalfall Komponenten an, welche diese Anforderungen umsetzen können. Prozesse, die sich in der Ausführung befinden, müssen zur Laufzeit verändert werden können. Nur wenn Schritte eingefügt oder auch der grundlegende Ablauf geändert werden können, kann flexibel auf benötigte Anpassungen reagiert werden. Es ist außerdem nicht nur notwendig laufende Prozessinstanzen ändern zu können. Oftmals muss das unterliegende Template geändert werden. Diese als Schemaevolution bezeichnete Änderungen sollen nach Möglichkeit auch auf laufende Prozesse angewandt werden können. Gerade langlaufende Prozesse wie in der Automobilbranche sind auf flexible Veränderungen angewiesen, da diese

Prozesse oftmals nicht unterbrochen oder neu aufgesetzt werden können.

Nicht zuletzt muss ein modernes WfMS die Überwachung von Prozessen zur Laufzeit ermöglichen. Das sogenannte Monitoring ermöglicht es, die Ausführung, und somit auch das grundlegende Design eines Prozesses, beobachten und analysieren zu können. Eine solche Komponente befähigt die zuständigen Stellen, flexibel auf Probleme im Prozess reagieren zu können. So können über diese Schnittstelle Ad-hoc Änderungen initiiert, oder die Neuzuordnung von Aufgaben über eine Delegation an einen anderen Bearbeiter vorgenommen werden.

2.2 AristaFlow BPM Plattform

Die *AristaFlow® BPM Suite* in ihrer heutigen Form basiert auf der grundlegenden Forschungsarbeit des *Instituts für Datenbanken und Informationssysteme* (DBIS) an der Universität Ulm durch die Projekte ADEPT1, ADEPT2 und AristaFlow [2].
Angeregt durch Forschungsarbeit im Bereich der Universitätskliniken wurde das Forschungsprojekt ADEPT1 gestartet. Gerade im klinischen Bereich unterliegen Prozessabläufe einem hohen Grad der Veränderung. Ein Informationssystem, welches solche Prozesse abbilden kann, muss selbst über einen hohen Grad an Mächtigkeit und Flexibilität verfügen.
Grundsätzlich wurden für ein System, welches diesen Anforderungen gerecht werden sollte, folgende zentralen Anforderungen definiert:

- Ein Prozessmanagement-System muss es ermöglichen, auch hochgradig komplexe Prozesse abzubilden. Hierzu zählt die Unterstützung von allen denkbaren Modellierungsartefakten wie Verzweigungen, Entweder-Oder-Pfade, Schleifen und Synchronisierung zwischen einzelnen Aktivitäten.
- Das System soll schon zum Modellierungszeitpunkt eine maximale Robustheit aufweisen. Prozesse dürfen nicht falsch modelliert werden können. Schon bei der Modellierung muss die Integrität und Fehlerfreiheit der Prozessstruktur gewährleistet werden.
- Ein hoher Grad an Flexibilität soll es ermöglichen, dynamisch auf benötigte Änderungen zur Laufzeit einzugehen und Prozesse entsprechend anzupassen.

Um diesen Anforderungen zu begegnen wurde das ADEPT Prozess-Meta-Modell entwickelt (Abbildung 2.4).

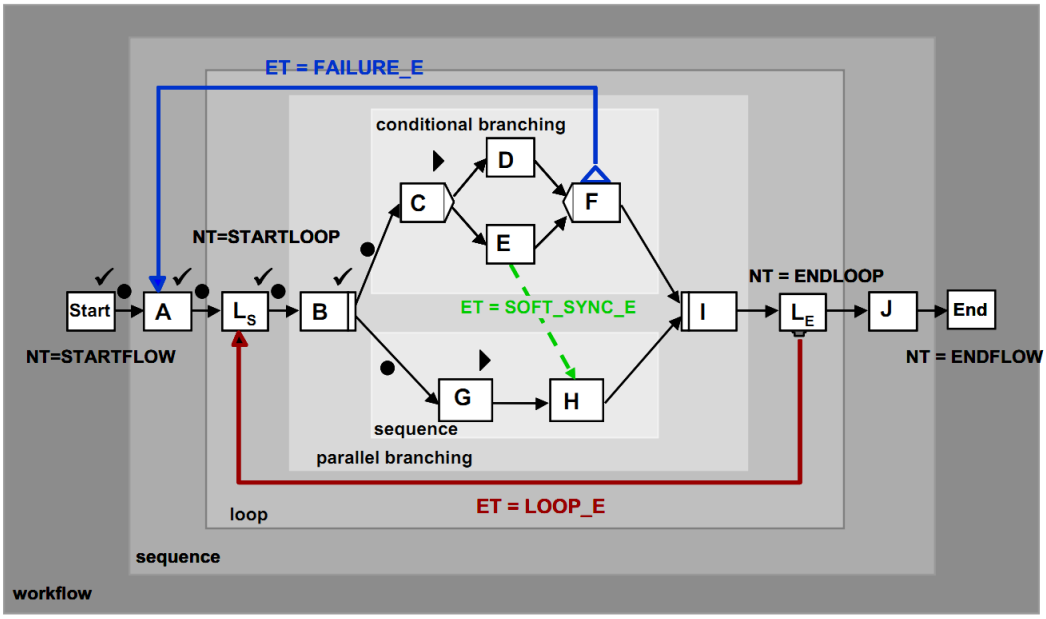


Abbildung 2.4: ADEPT Prozess-Meta-Modell [6]

Dieses Modell ermöglicht es über seine Ausdrucksmächtigkeit, beliebig komplexe Prozesse zu erstellen. Durch eine fest definierte Blockstruktur kann schon bei der Modellierung überprüft werden, ob die Integrität des formalen Prozesses durch eine Änderungsoperation gefährdet wird. Dies verhindert, dass der Benutzer einen Prozess völlig frei "zeichnen"kann. Modelliert werden kann nur nach festgelegten Regeln unter Beachtung des zugrunde liegenden Modells, was eine "Correctness by Construction"garantiert.

Mit dem Fokus auf dem Thema "Prozess-Schema-Evolution"wurde dieses grundlegende Konzept im Rahmen des ADEPT2 Projektes weiterentwickelt [4] [10]. Im Jahre 2008 wurde das Projekt AristaFlow in ein eigenständiges Unternehmen ausgegründet. Die *AristaFlow® BPM Suite* stützt sich auf jahrzehntelange Forschungsresultate und zählt zu den fortschrittlichsten, am Markt verfügbaren Plattformen im Bereich *Business Process Management* (BPM) und wurde deshalb im Rahmen dieser Arbeit als Basis für die Entwicklung eines Flexclients gewählt. Der konzeptuelle Aufbau von AristaFlow, soweit er die im folgenden vorgestellten Konzepte betrifft, wird näher in Kapitel 6 erläutert.

3 Stand der Forschung

Bevor wir in den folgenden Kapiteln definieren, welche Anforderungen für die Entwicklung unseres Clienten bedient werden müssen, soll in diesem Kapitel erläutert werden, wie in bisherigen Systemen mit der Anforderung an Flexibilität zur Laufzeit umgegangen wird. Dazu betrachten wir drei verschiedene, am Markt etablierte Systeme und erläutern jeweils kurz den gewählten Ansatz dieser Systeme.

3.1 YAWL

Das YAWL-WfMS ist ein System, welches auf der Prozessbeschreibungssprache YAWL basiert. Das von YAWL verfolgte Prinzip ist der Ansatz des **Late Binding** [8]. Hierbei wird die tatsächliche Auswahl der konkreten Implementierung erst zur Laufzeit getroffen. Es kann hierbei aus einer Menge von vorher definierten Aktivitäten ausgewählt werden. Die Entscheidung, welche Aktivität verwendet wird, kann regelbasiert oder durch entscheiden des Benutzers erfolgen. Dieses Prinzip wird von YAWL in einem sogenannten Worklet-Service umgesetzt. Wird ein solcher Service zur Laufzeit gerufen, können entsprechend vorheriger Ereignisse dynamisch Aktivitäten ausgewählt, oder auch weitere Services aufgerufen werden. Das grundlegende Prozesskonstrukt ist bei diesem Ansatz jedoch schon zum Modellierungszeitpunkt bekannt und kann nicht verändert werden.

3.2 IBM WebSphere Business Monitor

Der WebSphere Business Monitor ist ein Teil der IBM Produktreihe zur Unterstützung und Verwaltung von Geschäftsprozessen. Der Business Monitor stellt dabei eine Schnittstelle zur Überwachung und Veränderung von Prozessinstanzen dar. Flexibilität zur Laufzeit wird bei diesem Ansatz dadurch realisiert, dass Prozessschritte übersprungen, zurückgesetzt oder um weitere Aktivitäten erweitert werden können. Die sogenannten Ad-hoc

Tasks stellen dabei jedoch keine eigenständig neuen Prozessschritte dar. Diese Tasks müssen im Kontext einer anderen im Prozess definierten Aktivität entweder als Sub-Task oder Folge-Task ausgeführt werden [1]. Sub-Tasks müssen hierbei abgeschlossen werden, bevor die eigentliche Aktivität abgeschlossen wird. Durch das Ausführen von Folge-Tasks und Überspringen der ursprünglich gewählten Aktivität kann quasi ein Schritt ersetzt werden. Darüber hinaus ist es jedoch nicht möglich, die grundlegende Struktur des Prozesses zu verändern.

3.3 AristaFlow

Die Aristaflow Plattform wurde im Rahmen dieser Arbeit als Grundlage gewählt. Durch die Möglichkeit, Ad-hoc Änderungen an Prozessinstanzen durchzuführen, kann ein hohes Maß an Flexibilität zur Laufzeit erreicht werden (“Flexibilität durch Änderung“,) Sollten zur Laufzeit Änderungen notwendig sein, können entsprechende Prozessinstanzen in einem vollwertigen Modellierungskontext geöffnet werden. Für die Änderungen kann vollständig auf die Funktionen und Werkzeuge des Process Template Editor (Abbildung 3.1) zugegriffen werden.

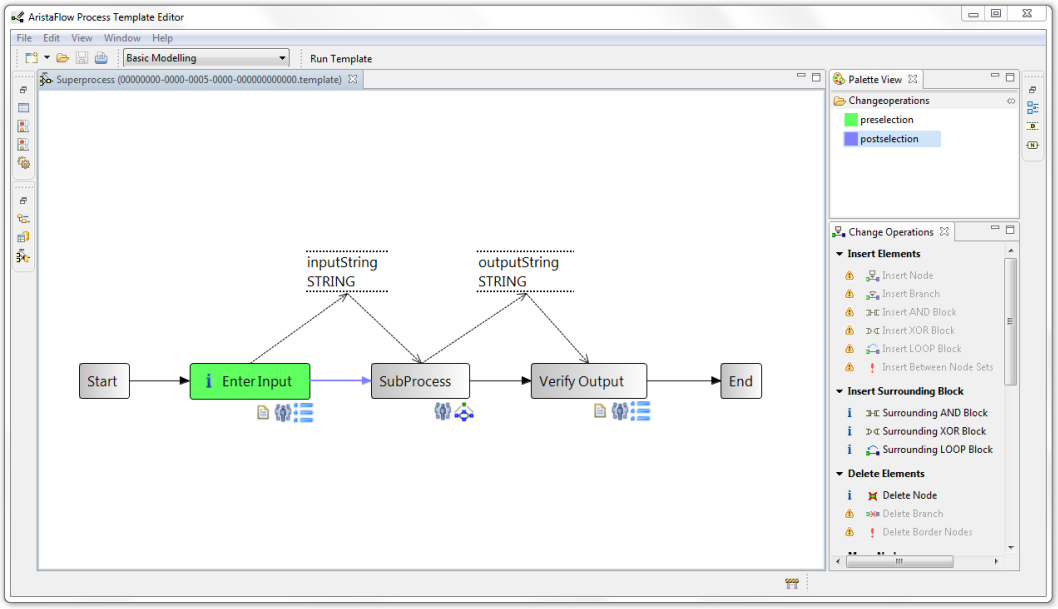


Abbildung 3.1: AristaFlow Process Template Editor

Dieser Ansatz ermöglicht es, die volle Mächtigkeit der ursprünglichen Modellierung auch für Laufzeitänderungen bereitzustellen. Dies bedeutet jedoch ebenfalls, dass auch der Nutzer, der diese Änderungen durchführt, über die gleichen Kenntnisse verfügen muss wie der ursprüngliche Modellierer des Prozesses.

3.4 Fazit

Alle vorgestellten Ansätze ermöglichen unter Berücksichtigung der jeweiligen Modellierungsaspekte ein gewisses Maß an Flexibilität. Veränderungen zur Laufzeit sind jedoch jeweils darauf ausgelegt, das ein komplexes Prinzip oder Funktionalität zur Laufzeit aufgegriffen und umgesetzt werden kann. Dies bedeutet, dass für Änderungen zur Laufzeit an all diesen Systemen ein Expertenbenutzer benötigt wird. Ein Endbenutzer mag Probleme in einem Prozess erkennen können, er ist aber in den wenigsten Fällen in der Lage dazu, mit den Werkzeugen für die Behebung dieser Probleme umzugehen. Wie dieser Umstand behoben werden kann, soll im Folgenden erläutert werden.

4 Grundsätzliche Anforderungen

Unser Ziel ist es, am Ende der Analyse Anforderungen für einen Prototypen zu erhalten. Damit diese fixiert werden können, ist es wichtig grundlegend zu verstehen, welche Aufgaben der Benutzer am System erledigen kann. In diesem Kontext sei der Benutzer verstanden als ein Anwender ohne Expertenkenntnis im Erstellen, Verwalten oder Überwachen von Prozessen. Vielmehr sei er als Anwender betrachtet, der ein WfMS im täglichen Gebrauch verwendet und bedient. Aus diesem Grund werden zuerst die benötigten Funktionen und Mechanismen zur Darstellung beschrieben und dann in Beispielen verdeutlicht. In Form eines Katalogs werden die Anforderungen dann im letzten Unterkapitel zusammengefasst.

4.1 Benutzer-relevante Aspekte

Am Anfang dieser Arbeit stand die zentrale Frage: “Welche Funktionen sollen dem Benutzer zur Verfügung stehen und wie können ihm diese präsentiert werden?“. Nachfolgend erläutert seien die Kernaspekte, die diese Frage beantworten.

Elementares Einfügen und Löschen

Die wichtigste Funktion die dem Benutzer ermöglicht werden soll, ist das elementare Verändern eines Prozesses zur Laufzeit. Dies stellt den häufigsten Anwendungsfall eines Clienten zum flexiblen Ändern zur Laufzeit dar. Nur wenn der Benutzer in der Lage ist, einen Schritt dynamisch in einen Prozess einfügen, innerhalb des Prozesses zu verschieben, oder seine Eigenschaften zu verändern, kann er zur Realzeit auch die Flexibilität umsetzen, die ein entsprechendes WfMS auf der technischen Seite bietet.

Neukomposition

Als Abwandlung des elementaren Einfügens sei die Neukomposition eines Prozesses verstanden. Nicht immer ist ein Experte verfügbar, welcher in der Lage ist, einen neuen Prozess zu erstellen. Auch wäre der Einsatz eines solchen Experten und der damit verbundene Aufwand nicht in allen Fällen gerechtfertigt. Triviale Prozesse, die etwa nur aus einigen wenigen Aktivitäten zur Benachrichtigung von Mitarbeitern bestehen, besitzen keinen hohen Grad an Komplexität. Diese Prozesse soll der Benutzer selber erstellen können. Er soll die Möglichkeit haben, einen leeren Prozess mit seinen gewünschten Aktivitäten zu füllen. Diese werden automatisch in einem korrekten Prozesskontext gespeichert und später zur Ausführung gebracht, falls dies nach Überprüfung und Verifizierung möglich sein wird.

Abstraktion von technischen Details

Damit der Benutzer möglichst einfach mit dem System interagieren kann ist es notwendig, so weit wie möglich von technischen Details zu abstrahieren. Der Nutzer soll nach Möglichkeit nur Einfügen, den Rest übernimmt das System. Für das Verschalten und Konfigurieren ist er nicht zuständig.

Benutzerfreundliches Interface

Ein Nachteil von mächtigen und komplexen Editoren ist ihre Unübersichtlichkeit. In den Händen eines Experten ist ein solcher Editor ein effektives Werkzeug, für den Endbenutzer allerdings ist er nicht geeignet. Unser Editor soll deshalb möglichst einfach und geradlinig sein. Der Nutzer soll ein einer einfachen Oberfläche nur die Informationen erhalten die er zur Bewältigung seiner Aufgaben benötigt. Probleme, die bei einer Tätigkeit auftreten, müssen dem Benutzer klar verständlich machen, warum eine Aktion nicht möglich ist und ihn zu einer möglichen Korrektur führen.

Integration und Stabilität

Nicht zuletzt damit der Client vom Endbenutzer angenommen werden kann, soll er in ein bestehendes Gesamtkonzept integriert werden. Wenn der Benutzer aus seinem normalen Arbeitskontext heraus Zugang zur neuen Funktionalität erhält, wird der Umgang deutlich

erleichtert. Durch die Nutzung einer bestehenden Plattform und der Integration des Clienten in dieselbe ist es möglich, bestehende Funktionalitäten wieder zu verwenden und darauf aufzubauen. So kann nicht nur die Mächtigkeit des bisherigen System, sondern auch die Integrität und Robustheit übernommen werden.

4.2 Anwendungsfälle

Die nachfolgenden Anwendungsfälle stellen einen Ausschnitt der real denkbaren Szenarien dar. Wir gehen davon aus, dass diese Aufgaben von einem normalen Endbenutzer durchgeführt werden sollen. Gezeigt werden soll, welche Anforderungen an einen Flexclient gestellt werden damit der Benutzer diese Aufgaben durchführen kann. Abschnitt 4.2.1 schildert ein Szenario, in dem ein Benutzer nachträglich eine Aktivität in einen schon laufenden Prozess einfügen soll. Abschnitt 4.2.2 beschäftigt sich mit dem Einfügen einer generischen Aktivität, welche flexibel auf die im Prozess vorhandenen Daten reagieren können muss. Der letzte Anwendungsfall 4.2.3 zeigt exemplarisch welche Anforderungen an eine Neukomposition eines Prozesses gestellt werden.

4.2.1 Nachträgliches Einfügen

Wir betrachten einen einfachen, konstruierten Bestellprozess in dessen Verlauf ein Kunde Waren per Vorkasse bestellt. Der Prozess wartet auf die eingehende Bezahlung des Kunden, verbucht diese und initiiert den Versand der Ware. Solange der Kunde nicht bezahlt, wird der Schritt “eingegangene Bezahlung verbuchen“nicht ausgeführt.

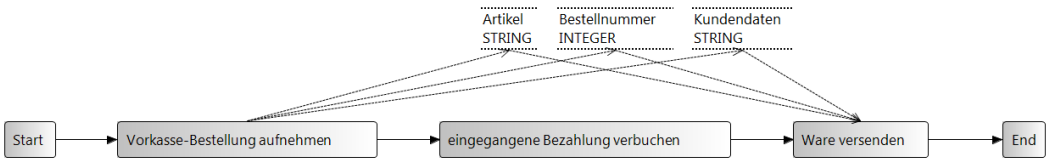


Abbildung 4.1: Vorkasse Prozess in AristaFlow Notation.

Auf mögliche Konflikte im Prozessablauf aus zeitlichen Überschreitungen sei in diesem Beispiel nicht eingegangen.

Der Kunde meldet sich nachträglich beim Unternehmen und möchte seine Bestellung erweitern. Es ist notwendig, dass eine zusätzliche Aktivität eingefügt wird, “Vorkasse-Bestellung erweitern“, und zwar vor dem Schritt “eingegangene Bezahlung verbuchen“. Wir gehen davon aus, dass die Bestellung über eine Bestellnummer eindeutig identifiziert wird. Des Weiteren soll die Aktivität vom selben Bearbeiter ausgeführt werden, der die ursprüngliche Bestellung im System gebucht hat. Diese Zuordnung soll automatisch geschehen. Die einzufügende Aktivität sei nur für Prozesse vom Typ “Vorkasse-Bestellung “anwendbar.

Damit die Aktivität in den Prozess eingefügt werden kann müssen folgende Kriterien als **Vorbedingung** erfüllt sein:

- Der Prozess befindet sich in einem Zustand, in dem noch Schritte eingefügt werden können, insbesondere darf der Prozess nicht komplett abgeschlossen sein.
- Der Benutzer kann eine gültige Position zum Einfügen der Aktivität auswählen, er findet einen korrekten Vor- und Nachbereich.
- Die Eingangsparameter der Aktivität sind genau spezifiziert. Dies bedeutet, dass auch nur Daten vom korrekten Datentyp verwendet werden können.
- Kontrollflussparameter bzw. alle von der Aktivität angeforderten Daten, wie in diesem Fall die Bestellnummer, sind im Prozess zum Einfügezeitpunkt vorhanden und können als eindeutig zu der Aktivität passend identifiziert werden.
- Die Nummer oder der Name des Bearbeiters der ersten Bestellaufnahme ist im System vorhanden und referenzierbar.
- Dieses Datum, welches den Bearbeiter identifiziert, muss eindeutig von Identifikatoren für andere Bearbeiter unterschieden werden können. Das System darf ausschließlich den Ausführenden des Schrittes “Vorkasse-Bestellung aufnehmen“erfassen und weiterverwenden.

Damit die Aktivität für den Prozess verwendet und korrekt verschaltet werden kann, stellen sich Anforderungen an die **Konfiguration**:

- Damit die einzufügende Aktivität für diesen Prozess verwendet werden kann, muss sie im Repository als Aktivität für Ad-hoc-Änderungen gekennzeichnet werden.

- Weiterführend ist diese Aktivität als ausschließlich für Prozesse des Typs “Vorkasse-Bestellung“anwendbar markiert. Es darf nicht möglich sein diese Aktivität für andere Prozesse zu verwenden.
- Die Parameter der Aktivität müssen in der Konfiguration hinterlegt sein.
- Regeln, welche die Bearbeiterzuordnung definieren, müssen ebenfalls bei der Aktivität gespeichert werden können. Hierbei muss ein Formalismus verwendet werden der es erlaubt die jeweiligen Bezüge, in diesem Fall auf eine vorangegangene Aktivität, abzubilden.

Die Aktivität konnte erfolgreich eingefügt werden wenn folgende **Nachbedingungen** erfüllt sind:

- Die Aktivität wurde an einer korrekten Position eingefügt.
- Die Daten wurden korrekt gemappt.
- Ein Bearbeiter wurde den Regeln entsprechend automatisch zugewiesen.

Eine tabellarische Übersicht über die einzufügende Aktivität:

Eingaben	**Ausgaben**	**Besonderheiten**
Bestellnummer	Bestellung aktualisiert	Aktivität nur für Prozesse des Typs “Vorkasse-Bestellung“anwendbar.
Identifikator des Bearbeiters der ursprünglichen Bestellung		Bearbeiter muss automatisch ermittelt werden können.
Einfügeposition		Bestellnummer sind im Prozess korrekt typisiert vorhanden.
Automatisch ermittelter Bearbeiter		

4.2.2 Einfügen einer generischen Aktivität

Besonders in großen und komplexen Prozessen, in welchen viele Daten oder Daten mit großem Umfang erhoben werden, ist es sinnvoll diese nicht durch den gesamten Prozess durchzuziehen sondern in einer Datenbank abzulegen. Im Prozess vorhanden sind dann nur die Referenzen oder IDs für diese Daten. So kann Inkonsistenz und Verlangsamung durch Überlastung vermieden werden.
In diesem Beispiel betrachten wir einen Ausschnitt eines frei konstruierten, umfangreichen Klinik-Prozesses. Ein Patient ist mit speziellen Beschwerden aufgenommen und in erster Instanz diagnostiziert worden und wartet nun auf seine Behandlung. Alle benötigten Grunduntersuchungen wie etwa MRT, Röntgen und CT seien bereits erfolgt.

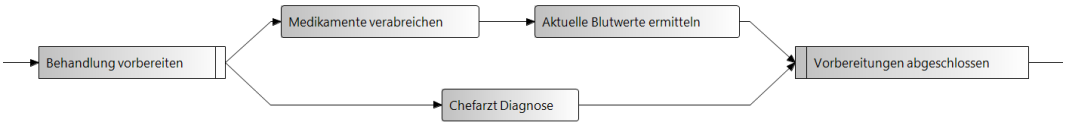


Abbildung 4.2: Klinik Prozess in AristaFlow Notation.

Bevor die Vorbereitungen zur Behandlung abgeschlossen werden können ist es notwendig, dass der Chefarzt eine Diagnose erstellt. Damit diese Diagnose vollendet werden kann, benötigt der Chefarzt jedoch eine weitere, noch nicht im Prozess vorhandene Aktivität "Lichtbilder auswerten". Diese Aktivität soll der Chefarzt nun selber in den Prozess einfügen. Es handelt sich hierbei um eine generische Aktivität, welche alle im Prozess erhobenen Daten vom Typ Lichtbild (identifiziert über die Nummer der Patientenakte) annehmen und darstellen kann. Hierbei kommt es zu keinem Fehler, falls im Prozess keine Lichtbilder vorhanden sein sollten. Nach Ausführung der Aktivität soll der Bericht des Chefarztes dann in der Datenbank zur Patientenakte gespeichert werden. Eine Ausführung dieser Aktivität soll nur Personen gestattet sein, welche die Rolle eines Chefarztes einnehmen können.

Zusätzlich zu den Anforderungen aus dem vorangegangenen Beispiel gelten folgende **Vorbedingungen**:

- Alle Lichtbilder, die in der Datenbank zur Akte gespeichert sind, müssen über einen Bezeichner referenziert werden können.

- Alle diese Bezeichner müssen im Prozess vorhanden sein.

- Es muss ein gemeinsames Identifizierungsmerkmal vorhanden sein welches es ermöglicht, die einzelnen Lichtbilder zu erfassen, seien sie vom Typ Röntgendiagnostik oder MRT-Aufnahme.

- Das System stellt einen Mechanismus bereit, der alle im Prozess vorhandenen Daten auf diesen gemeinsamen Kontext überprüft.

Für die **Konfiguration** der Aktivität ergeben sich folgende Ansprüche:

- Das Repository muss es erlauben, die Parameter für die Aktivität generisch zu definieren. Nicht nur spezielle Typen sollen anwendbar sein, sondern alle Typen mit einem gemeinsamen Merkmal.

- Diese Merkmale müssen im Repository verwaltbar sein. Es muss die Möglichkeit bestehen, verschiedene Datentypen beziehungsweise Identifikatoren für Datentypen in einen gemeinsamen Kontext zu stellen.

- Die Anforderung, dass der Bearbeiter der Aktivität die Rolle des Chefarztes einnehmen soll, kann in der Konfiguration für die Aktivität abgebildet werden.

Die Aktivität konnte erfolgreich eingefügt werden wenn folgende **Nachbedingungen** erfüllt sind:

- Die Aktivität wurde an einer korrekten Position eingefügt.

- Die Daten wurden korrekt gemappt.

- Es wurden alle im System vorhandenen Lichtbilder als Parameter verwendet.

- Mögliche Bearbeiter wurden auf die Rolle Chefarzt eingegrenzt.

Eine tabellarische Übersicht über die einzufügende Aktivität:

Eingaben	Ausgaben	Besonderheiten
Alle Lichtbilder	Bericht des Arztes in der Datenbank	Es ist nicht klar welche Daten vom Typ “Lichtbild“im Prozess vorhanden sind.
Menge von qualifizierten Bearbeitern		Mögliche Bearbeiter werden über konfigurierte Rolle identifiziert.
Einfügeposition		
Nummer der Patientenakte		

4.2.3 Neukomposition eines Prozesses

Als letzter Anwendungsfall betrachten wir eine Neukomposition eines Prozesses. Ein Benutzer, der über die entsprechenden Berechtigungen verfügt, soll über den Flexclient einen komplett neuen Prozess erstellen können. Als Beispiel diene der vereinfachte Prozess der Anmeldung einer Abschlussarbeit.

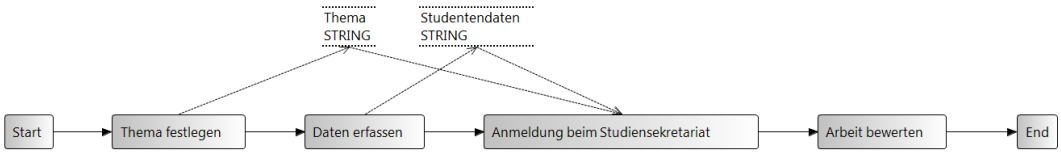


Abbildung 4.3: Abschlussarbeit Prozess in AristaFlow Notation.

Am Start des Erstellungsvorgangs soll dem Benutzer eine leerer Prozess, lediglich bestehend aus Start- und Endknoten vorgelegt werden. In diese Vorlage soll er die abgebildeten Schritte nach eigenem Ermessen einfügen. Es bleibt dem Benutzer überlassen, ob er die Schritte der Reihe nach einfügt oder ob er mit späteren Schritten beginnt. Er soll sich nicht mit der Zuordnung von Bearbeitern befassen müssen. Er legt lediglich fest, welche Rollen oder Organisationseinheiten (in diesem Beispiel etwa die Rolle “Professor“) den Prozess starten dürfen.

Damit ein komplett neuer Prozess erstellt werden kann müssen zusätzliche **Vorbedingungen** erfüllt werden:

- Ein neuer, leerer Prozess muss auf Anfrage des Benutzers hin erzeugt und präsentiert werden.
- Das Einfügen des ersten Schrittes erfolgt ohne Auswahl eines Vor- oder Nachbereiches, sondern automatisch und korrekt zwischen Start- und Endknoten.
- Dem Benutzer sollen als Auswahl von Aktivitäten nur diejenigen vorgelegt werden, die im Repository für eine Neukomposition markiert wurden.

Nachdem die erste Aktivität eingefügt wurde, entspricht der weitere Verlauf einem normalen, elementaren Einfügen. Da der Benutzer jedoch an keine Reihenfolge gebunden ist, können Schritte eingefügt worden sein die auf Daten zugreifen, die weder zum Einfügezeitpunkt noch generell im Prozess vorhanden sind. Wir betrachten den Fall in dem der Nutzer den Schritt "Anmeldung beim Studiensekretariat"einfügen will, bevor der Schritt "Daten erfassen"eingefügt wurde. Es ergeben sich folgende neue Anforderungen:

- Das System muss beim Einfügen einer Aktivität überprüfen, ob die geforderten Daten bereits im Prozess vorhanden sind.
- Sind geforderte Daten nicht vorhanden, überprüft das System, ob diese durch das Einfügen einer weiteren Aktivität nachgefordert werden können.
- Der Benutzer soll auf die mögliche Lösung aufmerksam gemacht werden, falls diese existiert. Sollte dies nicht der Fall sein muss darauf hingewiesen werden.
- Eine mögliche Lösung soll, nach Bestätigung des Nutzers, ebenfalls in den Prozess eingefügt werden.

Wichtig für eine unkomplizierte Erstellung des Prozesses ist die automatische Zuweisung von Bearbeitern. Da der Benutzer sich keine Gedanken über die sinnvolle Beschränkung von Bearbeiter-Regeln machen soll, müssen diese durch die **Konfiguration** abgedeckt werden:

- Aktivitäten im Repository müssen mit einer Regel konfiguriert werden können die den Initiator des Prozesses als Bearbeiter für die Aktivität zuweist.

- Regeln für die Bearbeiterzuordnung im Repository müssen Verweise auf den Prozessinitiator abbilden können.
- Ähnlich der Spezifikation einer Aktivität für das Einfügen in einen bestimmten Typ von Prozess muss es möglich sein, Aktivitäten für Neukompositionen zu markieren.

Eine Neukomposition war erfolgreich, wenn folgende **Nachbedingungen** erfüllt sind:

- Alle vom Benutzer gewünschten Aktivitäten wurden in einen Anfangs leeren Prozess eingefügt.
- Von Aktivitäten geforderte Daten sind vollständig vorhanden, keine Nachforderungen stehen aus.
- Für alle Aktivitäten wurde eine Bearbeitermenge, basierend auf der initialen Wahl des Benutzers, festgelegt.

Eine tabellarische Übersicht über den neu komponierten Prozess:

Eingaben	**Ausgaben**	**Besonderheiten**
Alle vom Benutzer ausgewählten Aktivitäten(iterativ)	Korrekt komponierter Prozess	System muss Nachforderung von Daten unterstützen.
Anfänglich festgelegte Bearbeiterzuordnung	Korrekte Zuordnung von Bearbeitern.	Bearbeiterzuordnung bezieht sich auf Prozessinitiator.
Jeweils iterativ die ausgewählte Einfügeposition		Aktivitäten für Neukomposition konfiguriert.
Von der jeweiligen Aktivität geforderte Daten		

4.3 Anforderungskatalog

Aus den vorangegangenen Beispielen ergibt sich ein vorläufiger, allgemeiner Anforderungskatalog. Hierbei sei zu diesem Zeitpunkt unterschieden zwischen Anforderungen an die Funktionalität sowie an die Verwaltung von Aktivitäten im Repository. Spezielle Anforderungen an die Darstellung des Flexclient und an das User Interface werden in den folgenden Kapiteln behandelt.

Anforderungen an die Funktionalität

1. Ermittlung von anwendbaren Aktivitäten: Das System muss für den vom Benutzer aktuell gewählten Kontext (spezieller Prozess, neuer Prozess) ermitteln, welche Aktivitäten relevant und verwertbar sind.

2. Validierung der gewählten Einfügeposition: Geprüft wird die vom Benutzer gewählte Einfügeposition. Diese Position ist definiert über den vom Benutzer gewählten Vor- und Nachbereich für die einzufügende Aktivität. Nach Überprüfung der Position unter Zuhilfenahme der vom WfMS bereitgestellten, nativen Mechanismen wird dem Benutzer mitgeteilt, ob ein Einfügen an dieser Position möglich ist.

3. Datenverfügbarkeit prüfen: Die im Prozess vorhandenen Daten werden überprüft, ob sie den Anforderungen der geforderten Daten einer Aktivität entsprechen. Sollten mehrere Daten den gleichen Grad an Übereinstimmung aufweisen, muss das Programm auf Basis eines vorab definierten Vorgehens eines davon auswählen, oder den Benutzer unter Hinweis auf den Ursprung des Datums diese Auswahl treffen lassen.

4. Kontextüberprüfung: Die Kontextüberprüfung stellt eine Erweiterung der Datenverfügbarkeit dar. Wie in unserem Beispiel des Klinikprozesses beschrieben, kann eine Aktivität Daten verschiedenen Typs akzeptieren, die über ein gemeinsames Merkmal verfügen. Sollte solch ein Parameter verlangt werden, muss der Kontext der verfügbaren Daten überprüft werden, anstatt nur den Datentyp und seinen Identifikator zu vergleichen. In Erweiterung dieser Überprüfung kann es möglich sein, dass Daten gefordert werde, die von bestimmten Aktivitäten stammen. Wie bei Merkmalen die einzelnen Daten zugeordnet sind, muss es ebenfalls möglich sein, die Merkmale der erzeugenden Aktivitäten zu überprüfen.

5. Nachforderung von Daten: Sollten die von einer einzufügenden Aktivität geforderten Daten noch nicht im Prozess vorhanden sein, soll überprüft werden, ob im Repository eine anwendbare Aktivität vorhanden ist, welche die geforderten Daten nachliefern kann.

Anforderungen an die Verwaltung von Aktivitäten

1. Aktivitäten müssen als prozessgebunden definierbar sein. Gemeint ist hierbei die Hinterlegung von Prozessen, für welche die jeweilige Aktivität anwendbar ist.

2. Analog zur ersten Anforderung muss es ebenso möglich sein, Aktivitäten für alle Prozesse oder nur für Neukompositionen verfügbar zu machen.

3. Die Konfiguration ermöglicht es, verschiedene Aktivitäten in einen gemeinsamen Kontext zu stellen. Über diesen gemeinsamen Kontext kann beim Überprüfen der im Prozess verfügbaren Daten sichergestellt werden, dass nur Daten weiterverwendet werden, die von diesen speziell markierten Aktivitäten stammen.

4. Analog zur Erstellung eines Kontextes für Aktivitäten muss diese Möglichkeit auch für die Parameter, also die geforderten Daten einer Aktivität, verfügbar sein.

5. Es muss eine Sprache für die Abbildung von Bearbeiterregeln definiert werden. Diese Sprache soll mächtig genug sein, um die geforderten Bezüge zwischen Aktivitäten abzubilden. Zusätzlich zu grundsätzlichen Regeln wie “Nur Bearbeiter der Stelle XY“oder “Nicht Person A“, müssen deutlich komplexere Regeln dargestellt werden können. Bezüge auf den Initiator des Prozesses oder auf Bearbeiter von früheren Aktivitäten müssen realisierbar sein.

5 Anforderungen an das System

Im letzten Kapitel wurde mit den grundlegenden Anforderungen an einen Flexclient spezifiziert, welche Anwendungsfälle und Szenarien zu erfüllen sind. Ein System, welches einen solchen Flexclient realisiert, muss in Hinblick auf die geforderten Funktionen Aufgaben, welche bisher von einem Expertenbenutzer umgesetzt wurden, automatisieren. In diesem Kapitel soll untersucht werden, welche Aufgaben genau vom System realisiert und umgesetzt werden müssen. Es wird geklärt, wie ein Flexclient sicherstellen kann, dass ein Benutzer ohne spezielle Erfahrung im Modellieren von Prozessen, eine Prozessinstanz zur Laufzeit so verändern kann, dass das Resultat nach wie vor der geforderten strukturellen Integrität entspricht. Es wird Schritt für Schritt definiert, welche Komponenten und Funktionen für dieses Ziel benötigt werden. Ebenfalls werden zu jedem Schritt die Anforderungen an die Präsentation des Flexclients für den Benutzer erläutert.

Wir haben in unseren Anforderungen das elementare Einfügen einer Aktivität als Kernanwendung unseres zu entwickelnden Clienten identifiziert. Andere Anwendungen, wie das Verschieben von Aktivitäten oder das Erstellen von komplett neuen Prozessen, seien als Abwandlung dieses Basisfalles verstanden. Im Folgenden soll der allgemeine Ablauf des Einfügens als Referenz für die jeweiligen Erläuterungen dienen. Abbildung 5.1 veranschaulicht diesen Ablauf in vereinfachter Form. Ersichtlich wird, das der Ablauf einen geschlossenen Kreis bilden kann. In den meisten Fällen wird ein Benutzer mit der Auswahl einer neuen Aktivität, welche er einfügen will, beginnen. Jedoch sind Anpassungen der einzelnen Aktivitäten wie etwa die Veränderung der Bearbeiterzuordnung ebenfalls denkbar. In diesen Fällen muss das System ebenso reagieren können und sicherstellen, dass alle Anforderungen erfüllt sind.

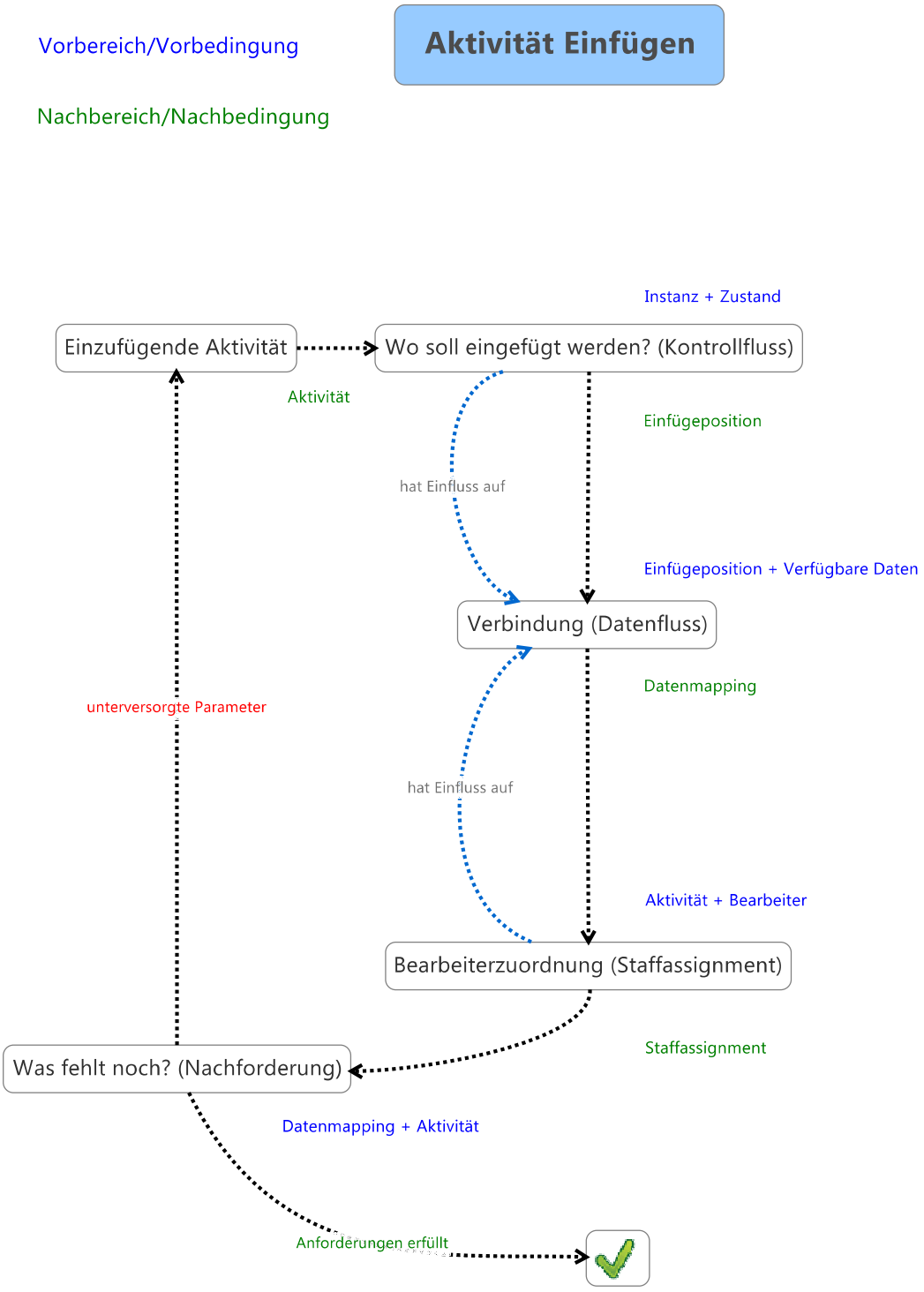


Abbildung 5.1: Vereinfachter Ablauf des Einfügens.

Damit ein Einfügen überhaupt möglich ist, muss eine Prozessinstanz in einen Zustand überführt werden, in dem sie bearbeitet werden kann. Sollte der Benutzer über die benötigten Rechte verfügen eine Instanz zu bearbeiten, soll das System diese “pausieren“und somit vorübergehend von der Ausführung suspendieren. Die Instanz kann dann in einem Flexclient geöffnet und für die Bearbeitung durch den Benutzer vorbereitet werden.

5.1 Einzufügende Aktivität

Die einzelnen Schritte des Einfügeprozesses können nicht völlig getrennt voneinander und in einer starren Reihenfolge betrachtet werden. Wie eingangs schon erwähnt steht jedoch meist der Wunsch einen neuen Schritt einzufügen am Anfang der Interaktion des Benutzers mit dem System. Das Ziel dieses Schrittes ist es festzulegen, welche Aktivität oder welche Aktivitäten in den vorliegenden Prozess eingefügt werden sollen.
Nachdem der Benutzer die Option “Aktivität einfügen“gewählt hat, sollen alle für den Prozess möglichen einzufügenden Aktivitäten präsentiert werden. Der Benutzer kann aus diesen Aktivitäten eine oder mehrere auswählen. Relevant ist hierbei der Modus des Einfügens. Es muss jeweils lineares Einfügen einer oder mehrerer Aktivitäten, paralleles Einfügen sowie entweder-oder Einfügen (XOR) als eigener Modus angeboten werden.

Anforderungen an die Systemfunktionalität:

- Das System stellt eine Schnittstelle zum Aktivitäten Repository bereit.
- Vom Repository erhaltene Aktivitäten müssen durch einen Algorithmus gefiltert werden. Nur Aktivitäten, welche für eine Verwendung im aktuellen Kontext konfiguriert werden, dürfen dem Nutzer präsentiert werden.

Anforderungen an die Präsentation:

- Aus der initialen Präsentation des Prozesses für den Benutzer besteht eine Option, um eine neue Aktivität einzufügen.
- Aktivitäten werden übersichtlich geordnet präsentiert. Name und Beschreibung sind verständlich.

- Der Benutzer muss in der Lage sein den Einfügemodus und infolge dessen gegebenenfalls mehrere Aktivitäten auf einmal auszuwählen.

5.2 Kontrollfluss

Oftmals ist in WfMS der Kontroll- und Datenfluss explizit getrennt. Auch das für diese Arbeit gewählt System AristaFlow folgt diesem Ansatz. Der Kontrollfluss umfasst alle Elemente des Prozesses die bestimmen, in welcher Reihenfolge und unter welchen Bedingungen Aktivitäten ausgeführt werden. Es ist also von höchster Priorität, dass die Intaktheit dieses Kontrollflusses erhalten bleibt. Nur wenn nach einer gewählten Änderungsoperation ein korrekter Kontrollfluss zustande kommt, kann eine Prozessinstanz endgültig verändert werden. Der Kontrollfluss spiegelt in Abbildung 5.1 den zweiten Schritt: "Wo soll eingefügt werden?", wieder. Nachdem der Benutzer die einzufügende(n) Aktivität(en) gewählt hat, legt er fest, an welche Position diese eingefügt werden. Für den Fall, dass mehrere Aktivitäten ausgewählt wurden muss sichergestellt werden, dass die gleiche Verfahrensweise auf alle diese Aktivitäten angewandt wird. So kann vermieden werden, dass nach Einfügen der ersten Aktivität eine Situation entsteht, welche die Auswahl der Einfügeposition invalidiert. In diesem Kontext seien mehrere einzufügende Schritte also als ein Block verstanden, für den im weiteren Beispiel wie mit einer einzelnen Aktivität verfahren wird.

Um eine Einfügeposition ermitteln zu können, muss als Vorbedingung der Zustand der einzelnen Knoten der geöffneten Instanz ermittelt werden können. Ein neuer Schritt kann niemals vor einem bereits abgeschlossenen Schritt eingefügt werden. Der Benutzer wählt aus den im Prozess vorhandenen Schritten sowohl einen Vor-, als auch einen Nachbereich aus. Durch dieses Vorgehen kann der Benutzer sowohl einen exakten Einfügepunkt definieren, als auch eine Menge von Knoten definieren, welche er abgeschlossen haben möchte, wenn der neue Schritt zur Ausführung kommt. Hat der Benutzer einen korrekten Einfügepunkt gefunden, hält das System diesen für den weiteren Verlauf fest.

Anforderungen an die Systemfunktionalität:

- Ein Algorithmus muss den Zustand der einzelnen Knoten überprüfen und diese einer initialen Knotenmenge für den Vor- und/oder Nachbereich zuordnen.

- Eine Schnittstelle zu den bereits bestehenden Funktionen des gewählten WfMS stellt sicher, dass die Überprüfung der Einfügeposition nach den festgelegten Regeln des zu Grunde liegenden Systems erfolgt.

Anforderungen an die Präsentation:

- Dem Benutzer wird jeweils eine Liste von in Frage kommenden Knoten für den Vor- und Nachbereich präsentiert.
- Eine Überprüfung während der Auswahl muss sicherstellen, dass sich die ausgewählten Bereiche nicht überschneiden.
- Sollte eine gewählte Position nicht valide sein, teilt das System dem Benutzer mit, warum dies nicht möglich ist und initiiert eine Neuauswahl.

Am Ende dieses Schrittes steht eine korrekte Einfügeposition, welche durch den gewählten Vor- und Nachbereich im weiteren Verlauf eindeutig identifiziert wird.

5.3 Datenfluss

Der nächste Schritt behandelt die Integration der neuen Aktivität in den bestehenden Datenfluss. Gemeint ist hierbei der explizite Datenfluss zu, sowie von der einzufügenden Aktivität. Es muss sichergestellt werden, dass alle zur Aktivität spezifizierten Parameter durch im Prozess vorhandene Daten versorgt werden können. Im Optimalfall können alle Parameter durch bereits geschriebene Daten im Prozess bedient werden. In speziellen Fällen jedoch, etwa der kompletten Neukomposition eines Prozesses sind diese Daten nicht vorhanden, sondern müssen zu einem späteren Zeitpunkt nachgeliefert werden, etwa durch das Einfügen einer weiteren Aktivität. Es soll also möglich sein, sowohl bestehende Daten direkt mit der neuen Aktivität zu verknüpfen, als auch einen Verweis auf eine spätere Verbindung oder ein komplett neues Datenelement anzulegen. Ausgabeparameter der Aktivität können sowohl mit bestehenden, als auch auf neuen Datenelementen verbunden werden. Hierbei ist es egal, ob neue erzeugte Datenelemente von späteren Prozessschritten gelesen werden.

Um ein Datenmapping festzulegen, müssen als Vorbedingung der gewählte Einfügezeitpunkt und alle im Prozess verfügbare Daten vorliegen. Das System muss diese Daten über-

prüfen, ob sie für die neue Aktivität verwendet werden können. Sollten mehrere Daten möglich sein, müssen diese als Auswahloption dem Benutzer vorgelegt werden.

Anforderungen an die Systemfunktionalität:

- Das System ermöglicht es, Aktivitäten und Parameter für diese mit speziellen Merkmalen auszuzeichnen. Dies erfolgt durch eine geeignete Markierung im Aktivitäten Repository.

- Ein Algorithmus muss für jeden Parameter der Aktivität überprüfen, ob passende Daten im Prozess vorhanden sind. Geprüft wird sowohl auf Datentyp als auch auf den Kontext des Datums. Das System ist in der Lage, falls es gefordert wird, nur solche Daten zu erfassen die mit einem speziellen Merkmal bezeichnet oder von einer Aktivität mit einem speziellen Merkmal erzeugt wurden.

- Das System ermöglicht Verknüpfungen mit neuen Datenelementen oder Verweise auf eine spätere Verknüpfung, insbesondere wenn keine passenden Daten verfügbar sind.

- Eine Erweiterung der Parameter muss möglich sein, falls eine Aktivität dafür konfiguriert wurde. Das System soll also vom Benutzer gewünschte, im Prozess vorhandene Daten als weitere Parameter der Aktivität mappen können, sollte dies per Definition erlaubt sein.

Anforderungen an die Präsentation:

- Dem Benutzer werden für jeden Parameter die in Frage kommenden Daten vorgelegt.

- Dargestellt werden sollen jedoch nur der Name des Datums sowie eine Beschreibung, welche Aktivität es erzeugt hat. Der Nutzer soll nicht mit Datentypen oder Identifikatoren in Berührung kommen.

- Dadurch dass nur passende Optionen für eine Verknüpfung bereitgestellt werden, ist eine falsche Auswahl des Benutzers nicht möglich.

Das Ergebnis des Schrittes ist ein vollständiges und korrektes Datenmapping.

5.4 Bearbeiterzuordnung

Nachdem Kontroll- und Datenfluss versorgt wurden, muss für die neue Aktivität eine Bearbeiterzuordnung festgelegt werden. Denkbar sind für diese Zuordnung zwei Varianten. Als erste Möglichkeit sei eine in der Aktivität fest definierte und nicht veränderbare Regel für die Zuordnung erwähnt. In diesem Fall ist bereits bei der Erstellung der Aktivität im Repository festgelegt, welche Bearbeiter bei der späteren Ausführung in Frage kommen. Das System kann in diesem Fall die festgelegten Regeln übernehmen und gegebenenfalls offene Referenzen auflösen. Die zweite mögliche Variante ist die Festlegung einer Bearbeiterregel durch den Benutzer selber. Das System muss hierbei ermitteln, welche Bearbeiter als Auswahloption in Frage kommen. Dabei ist wichtig das nur im jeweiligen Kontext sinnvolle Bearbeiter erfasst werden dürfen. Im konkreten Anwendungsfall werden fast immer spezielle Anwenderrollen oder Organisationseinheiten für diese Regeln herangezogen werden und in den wenigsten Fällen konkrete einzelne Bearbeiter.

Anforderungen an die Systemfunktionalität:

- Das System ermöglicht es, über eine beim Erstellen der Aktivität festlegbare Eigenschaft, fest vorgegebene oder veränderbare Regeln für die Bearbeiterzuordnung abzubilden.
- Sollte der Benutzer die Regel selbst festlegen dürfen, soll über einen Algorithmus ermittelt werden, welche Bearbeiter für die Auswahl in Frage kommen. Es muss geklärt werden, welche Bearbeiter in welchen Rollen bereits im laufenden Prozess vorkommen oder welche Bearbeiter im Kontext dieses Prozesses relevant sind. Durch diese Einschränkung kann verhindert werden, dass der Benutzer einen Schritt einer Menge von Bearbeitern zuordnet, die als völlig Fachfremde nichts mit dem laufenden Prozess zu tun haben.
- Das System kann eine vom Benutzer erstellte Auswahl in eine korrekte Bearbeiterregel umwandeln und weiterverwenden.
- Referenzen in fest vorgegebenen Regeln werden aufgelöst. So wird in etwa ein Bezug zum Prozessinitiator durch eben diesen ersetzt. Erst wenn alle Referenzen aufgelöst wurden kann fortgeschritten werden.

Anforderungen an die Präsentation:

- Dem Benutzer wird, im Falle einer nicht veränderbaren Regel, die schon getroffene Bearbeiterzuordnung als Information bereitgestellt.
- Sollte der Benutzer selber wählen können, muss er eine Liste von möglichen Bearbeitern beziehungsweise Rollen und Organisationseinheiten erhalten.

Dieser Schritt ist abgeschlossen, sobald eine korrekte Bearbeiterregel erstellt wurde.

5.5 Nachforderung

Obwohl keiner der genannten Schritte alleine betrachtet werden kann, stellt dieser Schritt eine Art Abschluss oder Ergebnis der jeweiligen Änderungsoperation dar. Egal ob einer oder mehrere neue Schritte eingefügt wurden, bevor eine Änderung endgültig durchgeführt werden kann, müssen alle Anforderungen erfüllt und verifiziert werden. In diesem Schritt wird überprüft ob sich der Prozess, genauer das Prozessmodell, wieder in einem korrekten Zustand befindet. Wurden die vorherigen Schritte erfolgreich abgeschlossen, kann jetzt endgültig eine neue Aktivität eingefügt werden. Als Vorbedingung stehen zu diesem Zeitpunkt alle vorher vom Benutzer getroffenen Entscheidungen und Daten zur Verfügung. Da sowohl Kontroll- und Datenfluss, als auch die Festlegung einer Bearbeiterzuordnung bereits verifiziert wurden, kann das System nun entsprechend der Regeln des verwendeten WfMS einen neuen Schritt einfügen. Diesem neuen Schritt wird dann die ausgewählte Aktivität mit dem gewählten Datenmapping und Bearbeiterregeln zugeordnet. Wichtig ist hierbei, dass auch Parameter die auf neue Datenelemente oder auf eine spätere Verknüpfung verweisen, ein regelkonformes Mapping darstellen. Das System muss den Benutzer auf diesen Umstand in entsprechender Weise aufmerksam machen. Der neue Schritt kann jedoch trotzdem eingefügt werden. Bevor der Prozess endgültig geändert und in der Workflow-Engine fortgesetzt werden kann, müssen diese unterversorgten Verknüpfungen behoben werden, entweder indem ein weiterer Schritt eingefügt werden kann der diese Daten liefert, oder durch das Ändern der Parameterzuordnung durch den Benutzer.

Anforderungen an die Systemfunktionalität:

- Das System stellt fest, ob alle benötigten vorherigen Schritte regelkonform abgeschlossen worden sind.

- Durch eine Schnittstelle zu den bereits bestehenden Funktionen des WfMS wird ein neuer Schritt eingefügt. Durch Nutzung der nativen Funktionen wird sichergestellt, dass bei jedem Einfügevorgang ein gleiches, vom Benutzer wiedererkennbares Einfügeverhalten gewährleistet ist.
- Sollten nach dem Einfügen des Schrittes Parameter unterversorgt sein, muss das System über einen Algorithmus ermitteln können, welche Aktivitäten im Repository die geforderten Daten liefern können. Eine Liste dieser Daten wird dem Benutzer für einen weiteren Einfügevorgang dann vorgelegt.
- Sollte sich der Prozess in einem korrekten Zustand befinden und keine Daten unterversorgt sein, muss das System auf Anweisung des Benutzers die geöffnete Prozessinstanz wieder auf den Ausführungsserver freigeben und erneut zur Ausführung bringen.

Anforderungen an die Präsentation:

- Dem Benutzer soll abschließend signalisiert werden, ob der Einfügevorgang erfolgreich war. Ist dies nicht der Fall, wird auf die entsprechende Fehlerstelle hingewiesen.
- Über eine geeignete Präsentation wird der Benutzer über offene Probleme, wie unterversorgte Daten, hingewiesen. Er muss die Möglichkeit haben, diese Probleme aus der jeweiligen Darstellung heraus beheben zu können.

6 Lösungskonzeption

Die Anforderungen, die in den vorigen Kapiteln erläutert wurden, haben definiert welche Funktionalitäten von einem Flexclient erwartet werden und gezeigt, was die technische Umsetzung eines solchen Clients in Form eines Systems für den Endbenutzer leisten muss. Diese Anforderungen werden bisher von keinem WfMS vollständig umgesetzt. Die oftmals angepriesene Flexibilität geht stets einher mit einem hohen Maß an Komplexität. Komponenten für die Modellierung und Veränderung von Prozessen liefern zwar eine umfangreiche und mächtige Grundfunktionalität, sind jedoch darauf ausgelegt, von einem Expertenbenutzer bedient zu werden.

Dieses Kapitel soll Lösungskonzepte vorstellen, die die von uns geforderte Flexibilität in Vereinheitlichung mit Benutzerfreundlichkeit und einer einfach zu bedienenden Oberfläche vereinen. Diese Konzepte setzen auf ein schon bestehendes System auf. Nur unter Nutzung einer bereits vorhandenen Menge von Funktionen kann gewährleistet werden, dass die Mächtigkeit eines bestehenden WfMS erhalten und der Flexclient in eine bestehende Landschaft integriert werden kann. Der folgende Abschnitt beschreibt die gewählte Plattform für unsere Entwicklung und erläutert diese grundlegenden Funktionalitäten.

6.1 Bestandsaufnahme

Wie in Kapitel 2 beschrieben, ist die Plattform unserer Wahl die *AristaFlow® BPM Suite*. AristaFlow erfüllt als modernes WfMS alle Grundanforderungen die für die Entwicklung eines Flexclients notwendig sind. Abbildung 6.1 zeigt vereinfacht die Grobarchitektur der AristaFlow Serverlandschaft.

Der **AristaFlow Server** stellt die zentrale Komponente dar. Er verwaltet die verschiedenen verfügbaren Prozessvorlagen sowie die schon gestarteten Prozessinstanzen. Er verwaltet weiterhin das Activity Repository und das Organisationsmodell [2]. Bei der Entwicklung

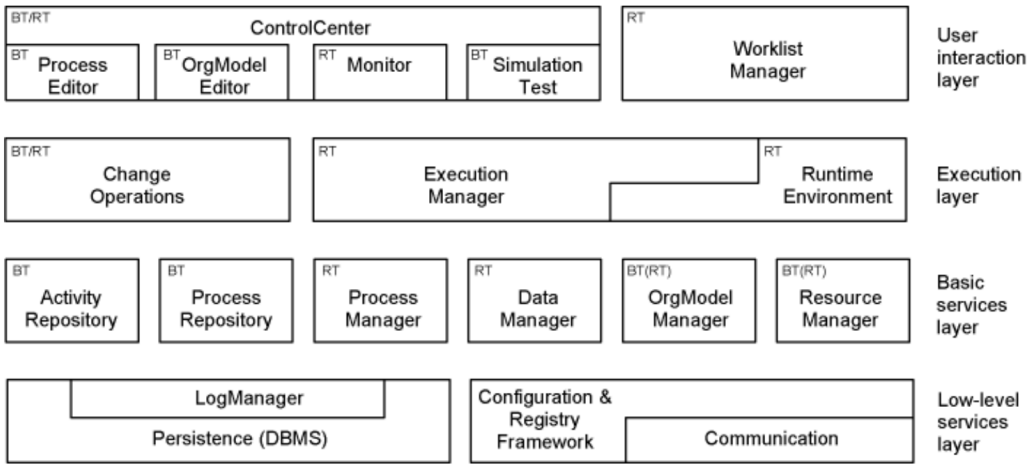


Abbildung 6.1: Grobarchitektur (BT: Buildtime, RT: Runtime) [2]

der Architektur wurde eine starke Orientierung am Service-Prinzip fixiert. Es wurde darauf geachtet, das verschiedene Module möglichst generisch implementiert werden. Dadurch können diese Module (Komponenten) über eine vielfältige API später einfach angesprochen werden.

Der **ChangeOperations-Service** kann als eigener Service von verschiedenen Bausteinen gerufen werden und ermöglicht es, basierend auf dem Zustand der gewählten Instanz, erlaubte Änderungsoperationen freizuschalten. So wird erreicht, dass sowohl bei der ursprünglichen Modellierung im Process Template Editor, als auch bei Ad-hoc Änderungen die über das Monitoring angestoßen werden, die gleiche Funktionalität bereitgestellt werden kann. Dieser Service stellt somit eine wichtige Komponente für die Entwicklung des Flexclients dar, welcher als Benutzeranwendung ebenfalls Änderungen zur Laufzeit realisiert.

Das **Activity Repository** ist für die Verwaltung der verschiedenen Aktivitäten zuständig. Über einen eigenen Editor (Abbildung 6.2) können Aktivitäten mit allen ihren Eigenschaften wie Parameter, Bearbeiterzuordnung und der eigentlichen Funktion definiert und verwaltet werden. Über die Realisierung des Acitivity Repositories als eigenständiger Service können diese Aktivitäten den späteren Anwendungen dann verfügbar gemacht werden.

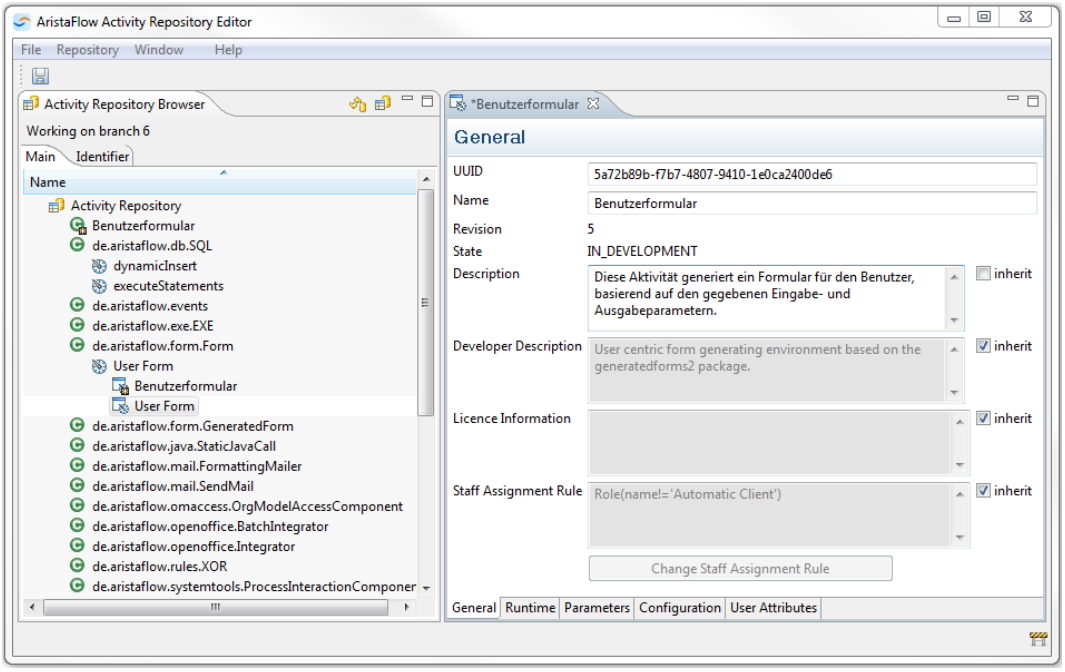


Abbildung 6.2: Erstellung einer Aktivität für ein Benutzerformular

Eine weitere, mächtige Komponente ist der **AristaFlow Monitor**. Er ist für Expertenanwender ausgelegt und bietet diverse Möglichkeiten, die Ausführung einer Prozessinstanz zu überwachen und nötigenfalls zu verändern. Der Experte kann den bisherigen Verlauf eines Prozesses nachverfolgen, Logdateien auswerten und eventuell aufgetretene Fehler analysieren. Ein wichtiger Bestandteil der Monitoring-Komponente ist die Möglichkeit, Ad-hoc Änderungen anzustoßen. Hierbei stehen dem Experten dieselben Mittel und Funktionen zur Verfügung, die auch der Process Template Editor für die initiale Erstellung des Prozesses bietet. Die Möglichkeit, Prozesse über den Monitor zur Laufzeit verändern zu können, ist die gewünschte Funktionalität, die der zu entwickelnde Flexclient einem Benutzer zur Verfügung stellen möchte. Hierbei soll der Benutzer jedoch nicht auf das erwähnte Expertenwissen angewiesen sein.

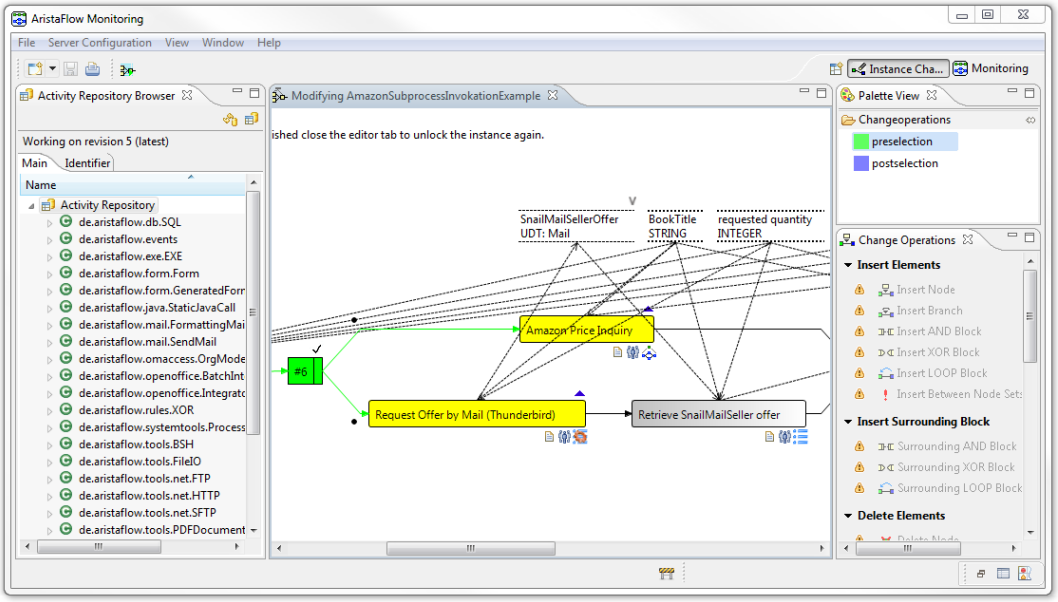


Abbildung 6.3: Ad-hoc Änderung über Monitoring

Als letzte Komponente sei der **OrgModel Editor** erwähnt. Dieser befasst sich mit der Erstellung und Verwaltung des Organisationsmodells. Abbildung 6.4 zeigt die verfügbaren Elemente des Orgmodells, mit Hilfe derer sich eine quasi beliebig komplexe Unternehmensstruktur abbilden lässt.

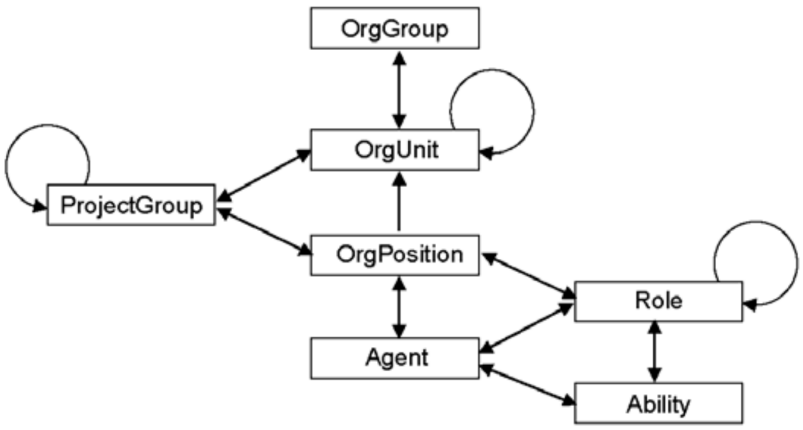


Abbildung 6.4: Elemente des AristaFlow-Organisationsmodells [2]

Zum OrgModell Editor gehört ein OrgModell Manager, welcher die Auswertung von Bearbeiterregeln, im Weiteren Staffassingment-Rules genannt, zur Laufzeit ermöglicht. Er kann ebenfalls als Service angesprochen werden. So ist es später möglich, für laufende Schritte die hinterlegte Regel auszuwerten, und diese Schritte dann in den Arbeitslisten der berechtigten Bearbeiter anzuzeigen.

Das Zusammenspiel dieser Komponenten stellt in ihrer Gesamtheit ein mächtiges Grundsystem dar. Dadurch, dass die einzelnen Module großteils als eigenständige Services realisiert wurde, ist es möglich, diese wiederzuverwenden. Die Grundlage für die Erweiterung dieses Systems durch einen Flexclient ist also durch die umfangreichen Bausteine zur Modellierung, Ausführung, Validierung und Veränderungen von Prozessen gegeben. Die Aufgabe die sich stellt ist es, die zusätzlich benötigte Funktionalität, die sich aus den Anforderungen an einen Flexclient ergeben haben, in dieses System zu integrieren. Dabei sollen wo immer möglich schon bestehende Funktionen genutzt werden.

6.2 Algorithmische Konzepte

Dieses Kapitel definiert Algorithmen und programmatische Abläufe, um die geforderten Systemfunktionen für eine tatsächliche Implementierung abbilden zu können. Alle Funktionen sollen später in einer Flexclient Anwendung zusammengefasst werden. Die vorgestellten Algorithmen werden hierzu in einer “Pseudocode“-artigen Notation verfasst. Dabei wird jeweils kenntlich gemacht, welche Komponenten und Serviceaufrufe der AristaFlow Plattform verwendet werden.

6.2.1 Ermittlung von relevanten Aktivitäten

Damit das System dem Benutzer eine mögliche Auswahl von einzufügenden Aktivitäten anbieten kann, muss diese Menge ermittelt werden. Das System erhält diese Aktivitäten vom Repository. Unser Algorithmus ist dann dafür zuständig, diese Aktivitäten zu filtern und nur diese wiederzugeben, die im aktuellen Kontext relevant sind.

Funktionaler Ablauf:

1. Nehme Verbindung auf mit Activity Repository Service und fordere alle verfügbaren Aktivitäten an.
2. Lege initial leere Liste an, welche am Ende dem Benutzer vorgelegt wird.
3. Überprüfe im aktuellen Prozesskontext, ob alle Aktivitäten, Aktivitäten die mit einem bestimmten Merkmal gekennzeichnet wurden oder ein bestimmtes Datum liefern können, oder nur Aktivitäten für eine Neukomposition angezeigt werden sollen.
4. Für alle Aktivitäten: Überprüfe, ob Aktivität dem gewünschten Merkmal entspricht. Falls ja, speichere Aktivität in Zielliste.
5. Übergebe fertige Liste der zuständigen User Interface Komponente.

6.2.2 Festlegung der Einfügeposition

Die Ermittlung einer geeigneten Einfügeposition ist die erste Interaktion unseres Systems mit denjenigen AristaFlow Funktionen, die sich mit der strukturellen Integrität des Prozesses befassen. Damit eine korrekte Einfügeposition gefunden werden kann, müssen mehrere aufeinander aufbauende Schritte vollzogen werden. Im ersten Teil wird ermittelt, welche Positionen überhaupt möglich sind. Da die Position später vom Benutzer ermittelt werden soll, wird die mögliche Auswahl eines Vor- und Nachbereiches für den Benutzer aufbereitet. Es muss darauf geachtet werden dass nur solche Schritte erfasst werden, die auch eine für den Nutzer relevante Funktionalität realisieren. Schritte wie AND-SPLIT/JOIN oder XOR-SPLIT/JOIN sollen vom Nutzer ferngehalten werden (erfasst werden sollen also nur Schritte vom Typ NORMAL). Für die Ermittlung der Listen muss bestimmt werden, in welchem Zustand sich die jeweiligen Knoten befinden:

1. Erstelle eine leere Liste für den möglichen Vor- und Nachbereich.
2. Für alle im Prozess vorhandenen Schritte überprüfe ob Typ = NORMAL:
   a) Schritt ist vom Typ NORMAL:
      i. Ordne Schritt der Liste für Vorbereich zu.
      ii. Falls Schritt nicht abgeschlossen: Ordne Schritt ebenfalls Nachbereich zu.
   b) Schritt ist nicht vom Typ NORMAL: Weiter mit nächstem Schritt.

3. Lege dem Benutzer nach Überprüfung aller Schritte die initialen Listen vor.

Der Benutzer erhält die geforderten Listen und kann aus diesen nun einen Vor- und Nachbereich festlegen (Die Verhinderung von Fehlern bei dieser Auswahl wird in Abschnitt 6.4 behandelt.). Wie in Kapitel 5 definiert wurde, wird für das Einfügen mehrerer Aktivitäten der gleiche Modus angewandt wie für das Einfügen einer Aktivität. Damit dem Benutzer eine stets gleichbleibende Erfahrung beim Einfügen vermittelt werden kann, ist es notwendig sicherzustellen, dass das Einfügen immer nach den gleichen Grundregeln abläuft. AristaFlow bietet über eine spezielle ChangeOperation das Interface InsertBetweenNodeSet an. Dieses Interface ermöglicht es, einen neuen Schritt in einen Prozess einzufügen. Hierbei wird eine geeignete und korrekte Position über die Angabe eines Vor- und Nachbereichs ermittelt. InsertBetweenNodeSet stellt dabei sicher, dass alle für den Vorbereich gewählten Knoten zum Einfügezeitpunkt abgeschlossen sind. Sollte es sich um eine komplexe Prozessstruktur mit diversen parallelen Zweigen handeln, wird dies über das Einfügen von Synchronisationskanten realisiert.
Diese Vorgehensweise erzeugt zwar ein Einfügeergebnis nach einem stets gleichen Muster, jedoch können die resultierenden Prozesse schnell und sehr stark wachsen. Wir definieren deshalb eine Erweiterung des Einfügekonzeptes, um den resultierenden Prozess zu vereinfachen. Sollten sowohl alle Schritte des Vorbereichs, als auch des Nachbereichs, auf einem direkten Pfad liegen, sich also nicht auf verschiedenen Zweigen eines AND- oder XOR-Splits befinden, reicht es, den letzten beziehungsweise ersten Schritt der beiden Listen als vereinfachten Vor- und Nachbereich zu betrachten. Es wird so verhindert, das unnötige Synchronisierungen im Prozess vorgenommen werden müssen. Ein formaler Ablauf ist denkbar wie folgt:

1. Für alle Schritte in Vorbereichsliste:
    a) Überprüfe, ob alle Schritte auf dem gleichen Pfad liegen.
    b) Falls nicht erfüllt: Fahre mit Schritt 4 fort.
2. Für alle Schritte in Nachbereichsliste:
    a) Überprüfe ob alle Schritte auf dem gleichen Pfad liegen.
    b) Falls nicht erfüllt: Fahre mit Schritt 4 fort.
3. Falls Pfadbedingung für beide Listen erfüllt:
    a) Entferne nicht mehr benötigte Schritte aus Vor- und Nachbereichslisten.

b) Sind letzter Schritt aus Vorbereich und erster Schritt aus Nachbereich auf dem gleichen Pfad?

i. Falls ja und beide sind direkte Nachfolger: Neuer Schritt kann direkt zwischen den beiden Knoten eingefügt werden und der Algorithmus terminiert.

ii. Falls keine direkten Nachfolger aber auf gleichem Pfad: Schritt kann parallel zu den bisherigen, zwischen den beiden liegenden Schritten eingefügt werden. Der Algorithmus terminiert.

4. Rufe InsertBetweenNodeSet auf um mit bestehenden Vor- und Nachbereichslisten eine korrekte Einfügeposition zu ermitteln.

6.2.3 Herstellung des Datenmappings

Die korrekte Herstellung eines Datenmappings ist von essentieller Bedeutung, um einen Schritt später ausführen zu können. Das System stellt sicher, dass der Benutzer für dieses Mapping nur korrekte Optionen auswählen kann. Es muss ermittelt werden, welche im Prozess vorhandenen Daten für die aktuelle Aktivität passend sind. Hierbei wird für jedes Datum der Datentyp, der Identifier für dieses Datum als auch ein spezieller Kontext in dem ein Datum steht, bestimmt. Dazu müssen nacheinander alle Parameter durchgegangen werden. Auch Parameter, die ein Benutzer möglicherweise noch zusätzlich definieren möchte (sollte die Aktivität dies erlauben) müssen auf diese Weise versorgt werden. Ein möglicher Ablauf dieser Überprüfung:

1. Für jeden zu versorgenden Parameter: Lege leere Liste für Verschaltungsoptionen an.
2. Füge jeder dieser Listen initial die Möglichkeiten: “NEUES DATENELEMENT“und “SPÄTER VERBINDEN“hinzu.
3. Lege Liste mit allen im Prozess vorhandenen und zum Zeitpunkt des Einfügens geschriebenen Datenelementen an.
4. Für jeden Parameter:

   a) Iteriere über Liste der gefundenen Datenelemente.

      i. Falls gesuchter Datentyp nicht kompatibel mit aktuellem Datentyp: weiter mit nächstem Element.

ii. Falls gesuchter Identifier nicht kompatibel mit aktuellem Identifier: Weiter mit nächstem Element.
iii. Falls Parameter nur Daten mit bestimmtem Merkmal akzeptiert: Vergleiche auf Merkmal. Bei keiner Übereinstimmung fahre mit nächstem Element fort.
iv. Falls Parameter nur Daten von Aktivitäten mit bestimmtem Merkmal akzeptiert: Vergleiche Merkmale der erzeugenden Aktivität mit gesuchtem Merkmal. Bei keiner Übereinstimmung fahre mit nächstem Element fort.
v. Falls keines der genannten Kriterien zu einem Abbruch geführt hat, füge aktuelles Datenelement in Liste der möglichen Datenelemente ein.

5. Zeige Benutzer für jeden Parameter eine Liste mit möglichen Verknüpfungen an. Sollte für einen Parameter, der als "NICHT NACHFORDERBAR", bei der Erstellung der Aktivität definiert wurde, keine passende Option gefunden werden, muss der Benutzer darauf Aufmerksam gemacht werden, dass die ausgewählte Aktivität zum aktuellen Zeitpunkt nicht eingefügt werden kann.

Durch die Ermittlung dieser möglichen Verschaltungsoptionen kann sichergestellt werden, das der Benutzer keine grundsätzlich falschen Entscheidungen beim Datenmapping treffen kann. Auch muss er sich nicht mit einer umständlichen Prozessnotation beschäftigen und herausfinden, wann welche Aktivität was für ein Datum erzeugt oder überschrieben hat. Die schlussendliche Auswahl aus mehreren geeigneten Datenelementen ist jedoch Sache des Benutzers. Der beschriebene Algorithmus wird analog auf die Bestimmung passender Ausgabedatenelemente angewandt.

6.2.4 Staffassignment

Der letzte große Baustein, der direkt auf Benutzereingaben reagieren können muss, ist die Bearbeiterzuordnung (Staffassignment). Wie in den Anforderungen definiert, sind zwei unterschiedliche Basisfälle denkbar. Zum Einen die schon beim Entwurf der Aktivität im Repository festgelegte Regel. Zum Anderen die Möglichkeit den Benutzer diese Regel über eine geeignetes Interface festlegen zu lassen. In beiden Fällen muss die Regel vom System überprüft und eventuell angepasst werden. Dies geschieht unter Zuhilfenahme des AristaFlow OrgModellManagers.

Potentieller Ablauf des Staffassignment:

1. Besitzt die Aktivität eine fest vorgegebene Staffassignment-Regel?

   a) Falls ja: Analysiere Regel auf noch aufzulösende Verweise.

      i. keine offenen Verweise: Weiter mit Schritt 3.
      ii. offene Verweise: Ersetze Variablen durch referenzierte Identifikatoren. Ermittle diese Identifikatoren über Datenelemente oder über Attribute von Aktivitäten die als Ziel eines Verweises in Frage kommen. Fahre mit Schritt 3 fort.

2. Falls Benutzer Regel verändern darf:

   a) Erstelle leere Liste für mögliche Entitäten aus dem OrgModell.

   b) Für jede Aktivität im Prozess: Ermittle OrgEinheit und OrgRole des Bearbeiters, der diese Aktivität ausgeführt hat und speichere diese in die angelegte Liste (distinct).

   c) Füge die Elemente der Liste unter Beachtung der AristaFlow OrgModell Notation zu einer Staffassignment-Regel zusammen

3. Überprüfe die erstellte/vollständig aufgelöste Regel mit Hilfe der Schnittstelle des OrgModellManagers.

4. Informiere bei Nichtbeanstandung durch den OrgModellManager den Benutzer über die resultierende Bearbeiterzuordnung. Im Fehlerfall muss der Benutzer auf den Umstand des Fehlers aufmerksam gemacht werden.

Auch bei dieser Funktionalität sind falsche Entscheidungen des Benutzers hinsichtlich einer korrekten Bearbeiterzuordnung nicht möglich. Wir gehen davon aus, dass Staffassignment-Regeln, die bereits im Repository fest vorgegeben sind, auch dort bereits bei der Erstellung überprüft werden. Des weiteren kann der Benutzer nur festlegen, welche OrgModell Entitäten er für seine Regel verwenden will. Die Erstellung der Regel wird vom System vorgenommen und ist somit nicht für Fehler des Benutzers anfällig. Sollten Verweise in einer Regel nicht aufgelöst werden können, etwa weil die Staffassignment-Regel auf einen Bearbeiter einer Aktivität aus einem bestimmten Kontext verweist, es aber keine solche Aktivität im Prozess gibt, muss der Benutzer davon in Kenntnis gesetzt werden.

6.2.5 Abschluss der Änderungsoperation

Am Ende der Änderungsoperation stehen im Erfolgsfall alle benötigten Informationen und Benutzereingaben fest. Sollten die vorherigen Überprüfungen erfolgreich gewesen sein, kann der Flexclient abschließend das tatsächliche Einfügen eines neuen Schrittes vornehmen. Hierzu wird ein neuer, leerer Schritt in den Prozess eingefügt, unter Verwendung der ermittelten Einfügeoperation und Einfügeposition. Die gewählte Aktivität kann dann mit dem ermittelten Datenmapping diesem Schritt zugewiesen werden. Alle diese Operationen erfolgen durch die native Funktionalität der AristaFlow Plattform, welche auch bisher beim Erstellen von Prozessen oder bei Ad-hoc Änderungen über den Monitor verwendet werden. Dadurch kann sichergestellt werden, dass jegliche Änderungen nicht nur systemweit auf die gleiche Art erfolgen, sondern dass ebenfalls jegliche strukturelle, semantische und datenbezogene Prozessvalidierungen , die das System bei der herkömmlichen Modellierung anwendet, auch in diesem Fall zum Einsatz kommen und somit eine korrekte Änderung einer Prozessinstanz ermöglichen.

Nach Abschluss der Änderungsoperation wird die geöffnete Instanz dem Benutzer erneut in der Flexclient Ansicht präsentiert. Der Benutzer kann nun das Bearbeiten fortsetzen oder die Instanz wieder freigeben. Bevor die Instanz jedoch freigegeben werden kann, müssen offen stehende Konflikte aufgelöst werden. Dies betrifft vor allem noch nicht vollständig versorgte Daten. Der Benutzer hat zwar in jedem Fall ein formal korrektes Datenmapping vorgenommen, jedoch können einzelne Parameter immer noch auf “SPÄTER VERBINDEN“oder “NEUES DATENELEMENT“verweisen. Das System stellt sicher, dass diese Verweise korrekt bedient werden. Dem Benutzer soll dieser Umstand in übersichtlicher Form mitgeteilt werden, etwa über eine Statusleiste. Dort hat er dann die Möglichkeit Fehler zu beheben. Unterversorgte Daten können durch den Benutzer versorgt werden, indem er das Datenmapping nachträglich in den Eigenschaften einer Aktivität ändert. Hierzu präsentiert das System ihm erneut den Dialog des Datenmappings, sollte dies gewünscht sein. Findet sich aus Sicht des Benutzers kein passendes Datenelement im Prozess, initiiert das System einen erneuten Einfügevorgang. Hierbei wird ein entsprechender Kontext gesetzt. Das System ermittelt bei diesem Vorgang nur Aktivitäten des Repositories, welche die geforderten Daten liefern können (Kontextsensitive Auswahl wie in Algorithmus 6.2.1 beschrieben).

6.3 Konzepte zur Speicherung von Aktivitäten

Damit die Aktivitäten und ihre Eigenschaften vom Flexclient korrekt verarbeitet werden können, ist es notwendig, bestimmte Konzepte zur Definition und Speicherung von Aktivitäten im Repository zu erweitern. Im Folgenden werden bestehende Konzepte des AristaFlow Activity Repostiory, soweit sie diese Arbeit betreffen, erläutert und durch neue Konzepte erweitert. Der Kontext der Entwicklung eines Flexclients schließt nicht eine Anpassung des Repositories für den Endbenutzer ein. Das Repository ist darauf ausgelegt, von einem Expertenanwender verwaltet zu werden. Nur durch Aufbereitung der Aktivitäten durch einen Experten kann sichergestellt werden, das alle benötigten Informationen später für die Schnittstellen des Flexclient bereitstehen.

6.3.1 Konfiguration

Das Activity Repository bietet viele verschiedene Möglichkeiten um Aktivitäten zu konfigurieren. Für eine detaillierte Erklärung und Einführung sei auf entsprechende Podcasts und Veröffentlichungen verwiesen, die über das AristaFlow-Forum zu finden sind [2].

Für die Aufgaben eines Flexclients wurde immer wieder betont, wie wichtig der Kontext ist, in welchem eine Änderungsoperation abläuft. Die Ermittlung von relevanten Aktivitäten hängt dabei von mehreren Parametern ab. Damit der jeweilige Kontext korrekt erfasst werden kann ist es notwendig, einige dieser Parameter neu zu definieren. Wir definieren für Aktivitäten folglich eine neue Konfigurationsgruppe “Kontext“. Innerhalb dieser Kontextkonfiguration unterscheiden wir verschiedene neue Optionen:

- Eine boolsche Option “On-the-Fly-Komposition“. Eine Anwahl dieser Option konfiguriert eine Aktivität als für eine Neukomposition zulässig.
- Eine weitere Option “prozessgebunden“, welche es erlaubt, Aktivitäten nur für bestimmte Prozesse verfügbar zu machen. Sollte diese Option ausgewählt werden, kann über eine Liste die Menge der verfügbaren Prozessvorlagen zugegriffen werden. Aus dieser Liste können dann eben diese Prozesse ausgewählt werden, die als gültige Anwendung für die jeweilige Aktivität in Frage kommen.
- Zur Auszeichnung eines Kontextes für Aktivitäten führen wir einen Mechanismus zur Markierung ein, ein sogenanntes “Tagging“. Ähnlich den schon vorhandenen

Identifiern für Datentypen, wird ein eigenes Repository für Tags angelegt. In der Kontextkonfiguration ist es dann möglich, eine beliebige Menge von Tags für die aktuelle Aktivität auszuwählen und dieser zuzuordnen.

Nicht nur für Aktivitäten an sich, sondern auch für Parameter und Daten im Prozess muss die Herstellung eines gemeinsamen Kontextes ermöglicht werden. Für die Identifizierung eines Datenelementes im Prozess definiert Aristaflow das Konzept der Identifikatoren. Jeder Parameter einer Aktivität erhält zusätzlich zu seinem Namen und Datentyp auch einen Identifikator. Bei der Prozessmodellierung können nur solche Daten verbunden werden, die in allen diesen Eigenschaften übereinstimmen.

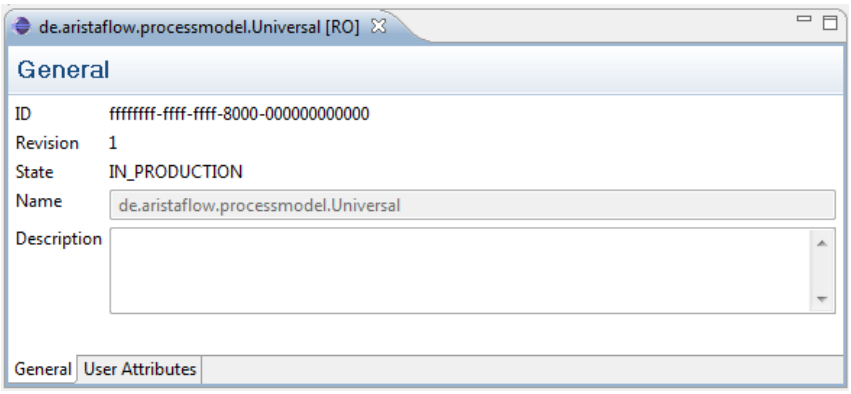


Abbildung 6.5: Darstellung des Universal Identifier

Durch die Verwendung des "Universal Identifier"können Parameter bereits kenntlich machen, dass sie Identifier verschiedenen Typs verarbeiten können. Dies reicht jedoch nicht aus, um Daten in einen gemeinsamen Kontext stellen zu können. Um dies zu erreichen, sind mehrere Konzepte denkbar:

- Analog zu den Aktivitäten ist ein einfaches Tagging möglich. Identifier werden mit bestimmten Begriffen ausgezeichnet. Diese Tags könnten als Voraussetzung bei der Parameterdefinition dienen.
- Im Kontrast zu der Zuordnung von Tags zu Identifieren steht das Zuordnen von Identifiern zu Gruppen. Identifier könnten hierbei einer oder mehreren Gruppen zugeordnet werden. Ein Parameter einer Aktivität könnte dann als Bedingung ein Datum verlangen, welches mit einem Identifier aus einer spezielle Gruppe bezeichnet ist. Der Vorteil dieses Ansatzes ist das Vorhandensein eines geschlossenen logischen

Konstruktes. Eine Gruppe könnte dadurch mit weiteren Eigenschaften und Optionen ausgestattet werden, sollte dies zu einem späteren Zeitpunkt notwendig werden.

- Der dritte Ansatz ist die Erweiterung des bisher bestehenden Konzeptes der Identifier um eine Hierarchie. Diese Hierarchie ermöglicht es, Eigenschaften zu vererben. Identifier eines gemeinsamen Vererbungsbaumes können über den zugrundeliegenden Basistyp als zusammengehörig oder zueinander passend identifiziert werden.

Welcher dieser Ansätze gewählt wird, ist jeweils im Kontext der gewählten Plattform abzuwägen. Für die Entwicklung auf Basis der Aristaflow Plattform ist die Lösung dieses Problems über die Erweiterung des Identifier-Konzeptes durch eine Vererbung naheliegend. Eines unserer Anwendungsbeispiele hatte die Verabeitung von generischen Lichtbildern als Hauptaspekt. Über die Einführung eines Basistypen "Lichtbild", und die Ableitung von weiteren Identifiern MRT, CT und Röntgen, könnte eine generische Aktivität flexibel auf die vorhandenen Daten reagieren. Ein Parameter würde für den Identifier Lichtbild definiert werden und die in Kapitel 6.2 vorgestellten Algorithmen könnten die im Prozess vorhandenen Lichtbilder über ihre Vererbungshierarchie erkennen.
Der Vorteil dieser Lösung liegt darin, dass sich am zugrundeliegenden Konzept nichts ändert. Die bisherige Modellierung und Änderung von Prozessen basiert nach wie vor auf dem etablierten Identifierkonzept, lediglich die Verwaltung der Identifier muss angepasst und erweitert werden. Die Nutzung bleibt durch das serviceorientierte Konzept jedoch gleich: Identifier können einfach über ihre ID referenziert werden.

6.3.2 Staffassignment

Ein Hauptaspekt der Konfiguration von Aktivitäten ist die Festlegung einer Staffassignment-Regel. Eine solche Regel setzt sich zusammen aus ausgewählten Entitäten des Orgmodells, sowie speziellen Bedingungen, die diese Menge an Entitäten beschränken oder an Bedingungen binden kann. Abbildung 7.8 zeigt den Modellierungsdialog für eine solche Regel. Im unteren Teil ist die resultierende Regel, basierend auf der getroffenen Auswahl, abgebildet.

Entitäten können dabei entweder über interne Nummern oder Namen referenziert werden. Auch die Nennung eines exakten Namen ist möglich.
Das hier gezeigte Konzept ermöglicht es, grundsätzliche Bedingungen für Bearbeiterzu-

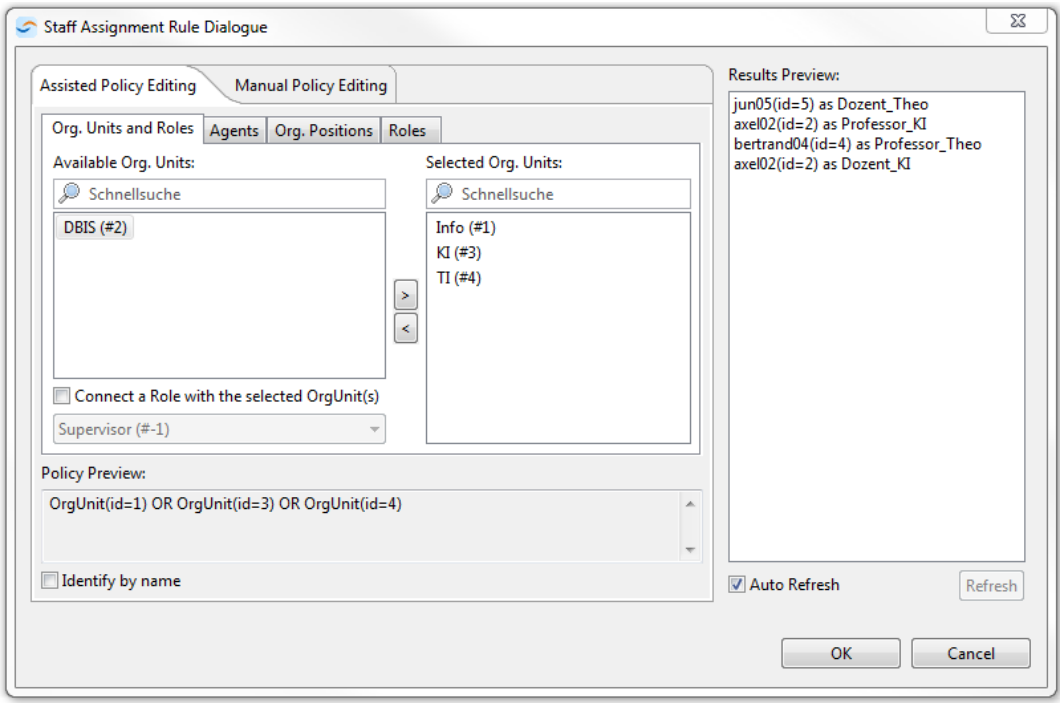


Abbildung 6.6: Dialog zur Erstellung einer Staffassignment-Rule

ordnungen festzulegen. Für die Anwendung in einem Flexclient wird diese Meta-Sprache durch zusätzliche Artefakte erweitert:

- Die definierte Anforderung, Bezug auf den Initiator einer Instanz zu nehmen, muss in Regeln abzubilden sein. Eine normale Regel, welche sich auf einen speziellen Bearbeiter bezieht, würde der Form: **Agent(id=4)** entsprechen. Für die gewünschte Referenz führen wir einen neuen Bezeichner ein: **InstanceInitiator.ID**. Die resultierende Regel lautet: **Agent(id=InstanceInitiator.ID)**. Der in Kapitel 6.2.4 definierte Algorithmus würde diese Referenz dann auflösen und durch die ID des Instanzinitiator ersetzen.

- Die wichtigste Erweiterung ist jedoch die Möglichkeit, einen bestimmten Kontext in einer Regel zu definieren. Wie im letzten Kapitel erläutert ist es notwendig, Daten hinsichtlich ihres Entstehungskontextes abbilden zu können. Ebenso notwendig kann es sein, Bearbeiter aufgrund der bisher im Prozess übernommenen Aufgaben auszuwählen. Adressierte Anforderungen sind hierbei etwa die Sicherstellung, dass nur ein Bearbeiter der schon einen Schritt mit der gleichen Kontextzuordnung wie der einzufügende Schritt, diesen neuen Schritt übernehmen darf. Ebenfalls ist die

Verhinderung einer solchen Zuordnung denkbar, im Rahmen einer Separation-of-Duties. Für diese Abbildungen führen wir ein weiteres Artefakt ein: **HasBeenInContextOf("Keyword")**. Dieses Artefakt lässt sich dann mit einer normalen Regel verknüpfen: **(OrgUnit(id=1) OR (OrgUnit(id=3)) AND !HasBeenInContextOf(Kredit)**. Diese Regel würde alle Bearbeiter der OrgUnits 1 und 3 erfassen, die in keiner Aktivität aus dem Kontext "Kredit"als Bearbeiter tätig waren. Der Flexclient wäre in der Lage diese Regel aufzulösen, da wir im vorigen Kapitel die Markierung von Aktivitäten in einem bestimmten Kontext definiert haben.

Die aufgeführten Beispiele können noch erweitert werden. Durch eine geeignete Definition von Artefakten können so auch komplexere Bedingungen über Kontextreferenzen abgebildet werden. Der Kernaspekt ist hierbei die Möglichkeit, einen solchen Kontext erstellen und referenzieren zu können.

6.4 User Interface Konzepte

Die Entwicklung eines benutzerfreundlichen Clienten für flexible Änderungen ist nicht nur eine Frage der benötigten Funktionalität, sondern auch eine Frage der geforderten und geleisteten Usability. Mächtige Werkzeuge in der IT-Landschaft zeichnen sich fast immer durch eine hohe Einarbeitungszeit aus. Es ist fast nie möglich, solche Programme nur kurz zu bedienen um ein schnelles Ergebnis zu erzielen. Auch auf dem Feld der informationsgestützten Prozessverwaltung ist es unumgänglich, ein hohes Maß an Wissen zu besitzen um ein Ergebnis zu erzielen, welches nicht nur halbwegs sinnvoll, sondern vollständig verwertbar ist. Die Entwicklung eines Flexclient ist darauf ausgelegt, die Lernkurve des Benutzers deutlich abzuflachen. Das Anwendungsfeld eines solchen Clienten ist es, kurzfristig notwendige Änderungen, von den Benutzern umsetzen zu lassen, die auch kurzfristig mit dem Resultat arbeiten müssen. So soll eine umständliche Behebung von Problemen durch Experten vermieden werden. Damit der Benutzer schnell und einfach Ergebnisse produzieren kann, werden für die Entwicklung des User Interfaces folgende grundlegenden Konzepte definiert:

- Entwicklung einer einfachen und sprechenden Oberfläche. Nur die gewünschte und auch benötigte Funktionalität soll abgebildet werden.

- Eine klare Strukturierung. Jede abgebildete Information soll auf Anhieb klarmachen, warum sie existiert. Der Benutzer soll nicht über Elemente rätseln, die er nicht nutzen oder im Extremfall nicht einmal verstehen kann.
- Der Benutzer wird von Anfang bis Ende geführt. Es soll nicht möglich sein, Teile eines Vorganges abzuschließen um dann Teile eines anderen Vorganges dazwischen zu schieben. Eine Operation am Flexclient wird immer vollständig oder gar nicht abgeschlossen werden.
- Ebenfalls als Grundprinzip sei die schon im Kontext der ADEPT1 Forschung erwähnte "Correctness by Construction", genannt. Durch die vorgestellten Funktionen und Konzepte soll es dem Benutzer quasi unmöglich sein, einen Fehler zu modellieren.

Die schlussendlich für unseren Flexclient umgesetzten User Interface Komponenten lassen sich durch anschauliche Beispiele sehr viel besser verdeutlichen und werden im nächsten Kapitel, der Implementierung des Prototypen, ausführlich dokumentiert.

7 Implementierung eines Prototyps

Zu Beginn dieser Arbeit stand der Wunsch nach der Entwicklung eines Clienten für flexibles, benutzerfreundliches Einfügen. Schnell war klar, das eine konkrete Realisierung nur auf Basis eines bestehenden WfMS geschehen konnte. Durch die jahrzehntelange Forschungsarbeit im Institut für Datenbanken und Informationssystem (DBIS) an der Universität Ulm, welche in der bereits erwähnten Ausgründung der AristaFlow Plattform resultierte, lag es nahe, AristaFlow für die Entwicklung des Flexclients heranzuziehen. Im Rahmen der Diplomarbeit wurden prototypisch Teile der in der vorliegenden Arbeit definierten Konzepte implementiert. Diese Arbeit wurde parallel dazu durch eine wissenschaftliche Hilfstätigkeit um Aspekte der Integration in die AristaFlow Plattform ergänzt.

7.1 Einordnung des Prototypen

Der Flexclient wurde konzipiert als eigenständige Anwendung in der AristaFlow Prozess-Landschaft und befindet sich auf Ebene der Benutzerinteraktion, dem sogenannten "User interaction layer"(Abbildung 7.1). Es handelt sich um eine Laufzeitkomponente, die vom Benutzer aufgerufen werden kann. Bei der Entwicklung wurde Wert darauf gelegt, bereits vorhandene Konzepte der Plattform für benötigte Funktionalitäten zu verwenden. So werden alle in Kapitel 6 und 7 definierten funktionalen und algorithmischen Konzepte durch den Flexclient als geschlossenes System realisiert. Für die Umsetzung der grundlegenden Clienten-Umgebung wurde die Eclipse-RCP (Rich Client Platform) Plattform gewählt, welche auch bei den bisherigen Komponenten von AristaFlow eingesetzt wird.

Die vorgestellten Konzepte zur Speicherung von Aktivitäten betreffen die Activity Repository Komponente. Diese Konzepte sind jedoch zum jetzigen Zeitpunkt des Prototyps noch nicht vollständig umgesetzt (siehe dazu Abschnitt 3, Ausblick).

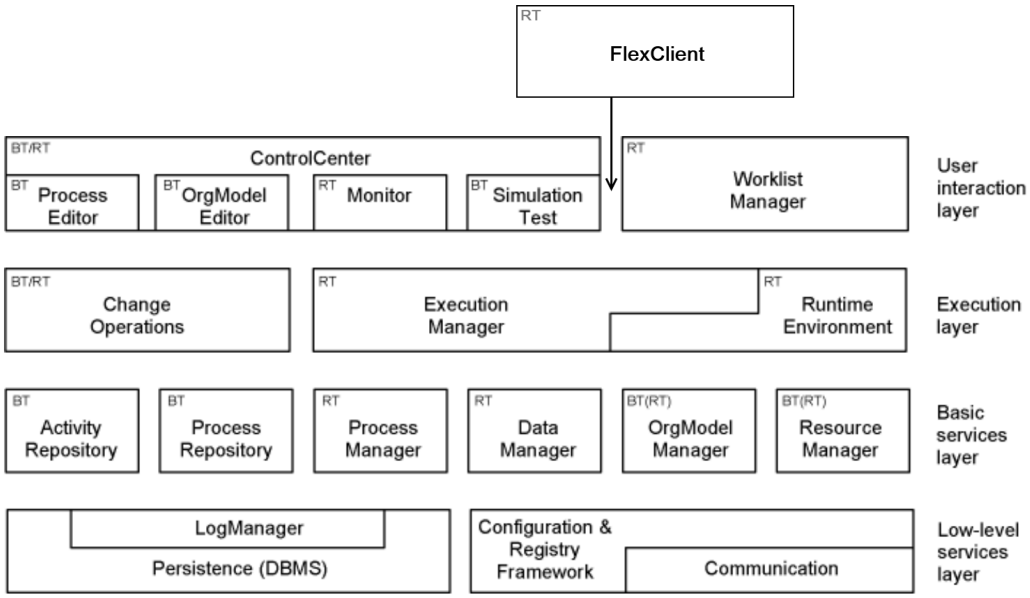


Abbildung 7.1: Einordnung des Flexclient in die bestehenden Architektur.

7.2 Interaktion des Prototypen

Dieser Abschnitt soll den Flexclient in seinem aktuellen Zustand vorstellen. Hierbei wird Schritt für Schritt gezeigt, wie sich das System dem Benutzer präsentiert und wie dieser mit dem System interagieren kann. Kenntlich gemacht wird ebenfalls, zu welchem Zeitpunkt der Flexclient mit den einzelnen anderen Komponenten der AristaFlow Plattform interagiert. Als Beispiel wird ein einfacher, linearer Prozess verwendet. Dieser Prozess ist frei konstruiert und dient lediglich dem Zweck der Veranschaulichung. Nach Starten des Flexclients bekommt der Benutzer das Startcenter präsentiert (Abbildung 7.2).

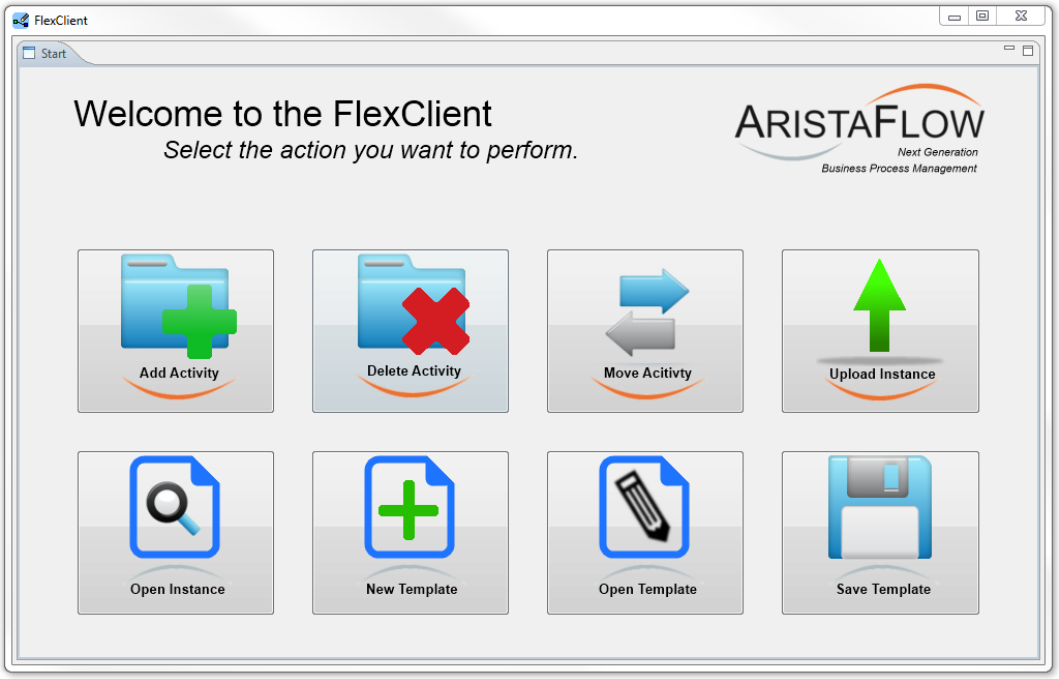


Abbildung 7.2: Das Flexclient Startcenter.

In dieser Übersicht sind alle grundlegenden Funktionen aufgelistet, die der Benutzer auswählen kann. Er hat die Möglichkeit, eine laufende Instanz für Anpassungen zu laden oder eigene Templates zu erstellen. Alle angebotenen Auswahloptionen sind möglichst einfach und sprechend illustriert. Der Benutzer kann auf den ersten Blick erkennen, welche Optionen für ihn bereitstehen. Das Startcenter bleibt über die gesamte Programmlaufzeit verfügbar. Die jeweiligen Funktionen stehen jedoch auch zu einem späteren Zeitpunkt über das Kontextmenü oder die Actionbar zur Auswahl bereit.

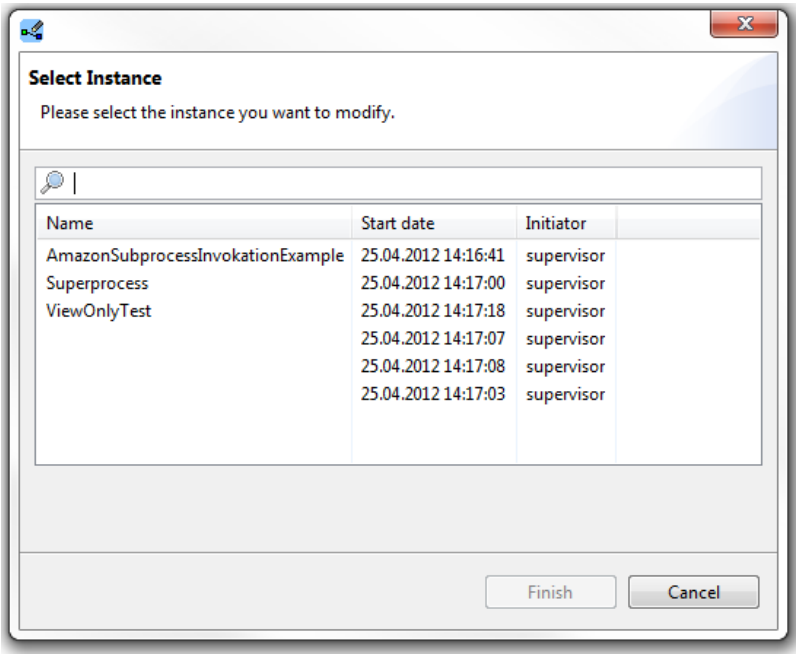


Abbildung 7.3: Benutzer kann Instanz auswählen.

In dieser Demonstration gehen wir davon aus, dass der Benutzer eine Instanz zur Bearbeitung öffnen möchte. Ein Klick auf die entsprechende Schaltfläche öffnet den Dialog der zur Verfügung stehenden Instanzen (Abbildung 7.3). Aufgeführt sind alle zum aktuellen Zeitpunkt ausgeführten Instanzen. Diese Instanzen werden über eine Anfrage an den die InstanzManager Komponente des Servers erhalten. Der Benutzer wählt in diesem Beispiel die Instanz mit dem Namen “Superprocess“. Der Flexclient stellt nun eine Lock-Anfrage an den InstanzManager-Service. Sollte die gewünschte Instanz bereits in einem anderen Editor geöffnet sein, bekommt der Benutzer eine Nachricht, die ihn auf diesen Umstand hinweist. Ist dies jedoch nicht der Fall, kann die Instanz nun über den erwähnten Service gesperrt werden. Lokale Änderungen an der Instanz sind jetzt möglich. Zu beachten ist dabei: Änderungen, die im Editor vorgenommen werden, sind vorerst auch nur für diesen gültig. Eine endgültige Änderung der Instanz kann nur durch einen entsprechenden Service vorgenommen werden, welcher die Änderungen an einer lokalen Instanz auf die originale Instanz auf dem Server abbildet.

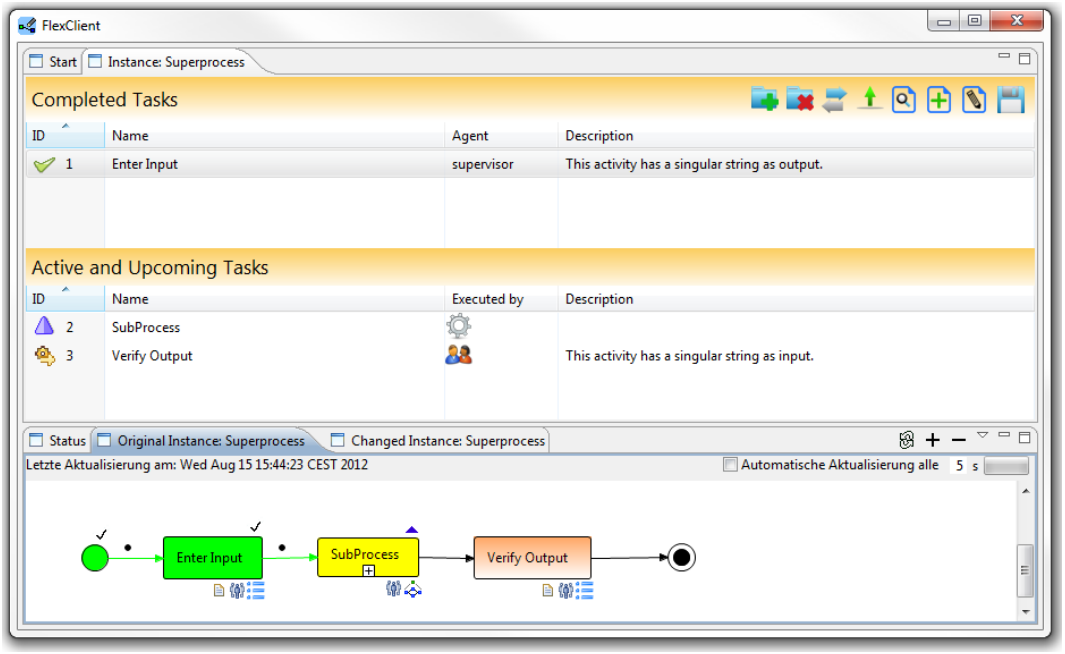


Abbildung 7.4: Ansicht einer geöffneten Instanz.

Nachdem eine Instanz erfolgreich geladen wurde, wird die Flexclient Editor Ansicht mit dem Kontext der jeweiligen Instanz geladen. Abbildung 7.4 zeigt diese initiale Ansicht einer geladenen Instanz. Drei Hauptkomponenten werden dem Benutzer präsentiert: Die bereits abgeschlossenen Schritte der Instanz, die noch kommenden Schritte sowie eine vereinfachte Prozessansicht in BPMN-Notation. Alle Informationen, die dem Benutzer hier präsentiert werden, sind die Namen der Schritte, ihre Beschreibung und wer diese Schritte jeweils ausführen darf. Auch in der grafischen Prozessansicht bekommt der Benutzer nur die für ihn verständlichen Schritte angezeigt (in diesem trivialen Beispiel sind keine Verzweigungen vorhanden). Auf expliziten Wunsch des Benutzers können jedoch zusätzliche Optionen, wie die Anzeige von Datenelementen, zugeschaltet werden. Über die Actionbar in der oberen rechten Leiste wählt der Benutzer nun die Option “Aktivität einfügen“.

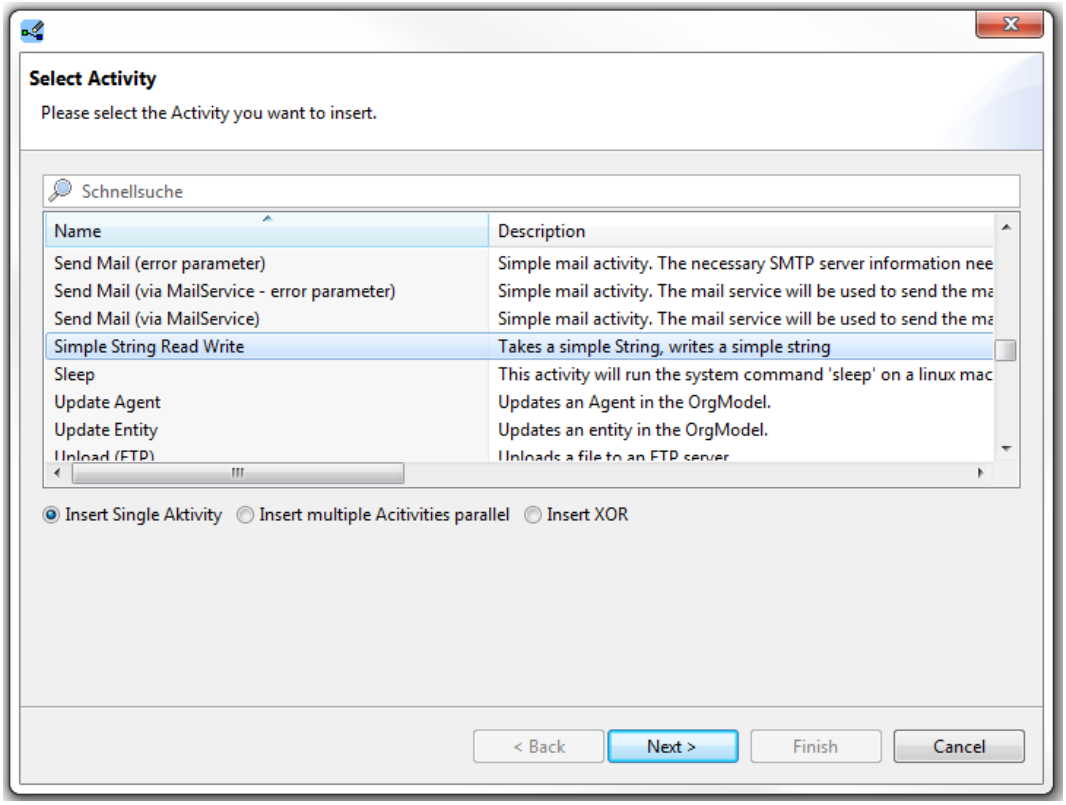


Abbildung 7.5: Auswahl einer einzufügenden Aktivität.

Es öffnet sich die erste Seite des Einfügedialogs (Abbildung 7.5). Der Flexclient hat alle über den Activity Repository Service verfügbaren Aktivitäten geladen. Über die implementierten Funktionen wurden diese Aktivitäten gefiltert. Beim Starten des Einfügevorgangs wurde der Kontext der aktuellen Instanz überprüft und festgestellt, dass alle verfügbaren Aktivitäten gewünscht sind. Das System erkennt, dass es sich nicht um eine Nachforderung von Daten oder um eine Neukomposition eines Prozesses handelt. Die erhaltenen Aktivitäten werden dem Benutzer mitsamt einer Beschreibung präsentiert. Zusätzlich zur Aktivitätenauswahl hat der Benutzer noch die Möglichkeit, den Modus des Einfügens zu spezifizieren. Zum aktuellen Zeitpunkt sind einzelnes und paralleles Einfügen realisiert. Über dies Maske soll jedoch auch später die Einfügemöglichkeit einer XOR-Verzweigung angeboten werden. In unserem Beispiel entscheidet sich der Benutzer dazu, die Aktivität “Simple String Read Write“, welche für dieses Beispiel erstellt wurde, einzufügen.

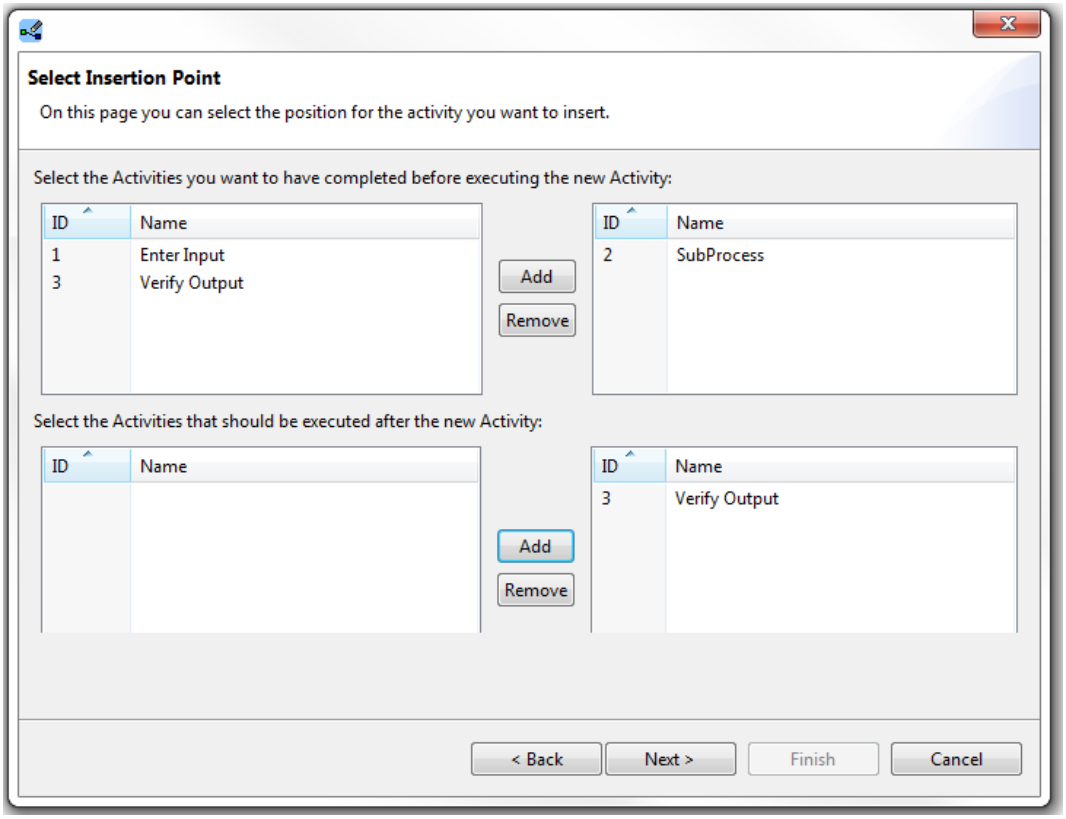


Abbildung 7.6: Auswahl des Vor- und Nachbereichs.

Die nächste Seite des Dialoges behandelt die Festlegung des Einfügezeitpunktes. Der Benutzer bekommt eine für Vor- und Nachbereich initiale Menge an möglichen Schritten geliefert (bestimmt durch den in Abschnitt 6.2.2 vorgestellten Algorithmus). Aus diesen Listen kann er jetzt einerseits die Schritte auswählen, die er vor der einzufügenden Aktivität beendet haben möchte als auch andererseits die Schritte, deren Start erst nach Abschluss der Aktivität gestartet werden sollen. Das User Interface überprüft hierbei schon bei der Auswahl der Bereiche, ob ein Einfügen an dieser Stelle möglich ist. Eine überschneidende Auswahl ist nicht möglich, Schritte des Vorbereichs können nicht im Nachbereich verwendet werden (und vice versa). Nur wenn eine korrekte Auswahl getroffen wurde, kann der Benutzer voranschreiten.

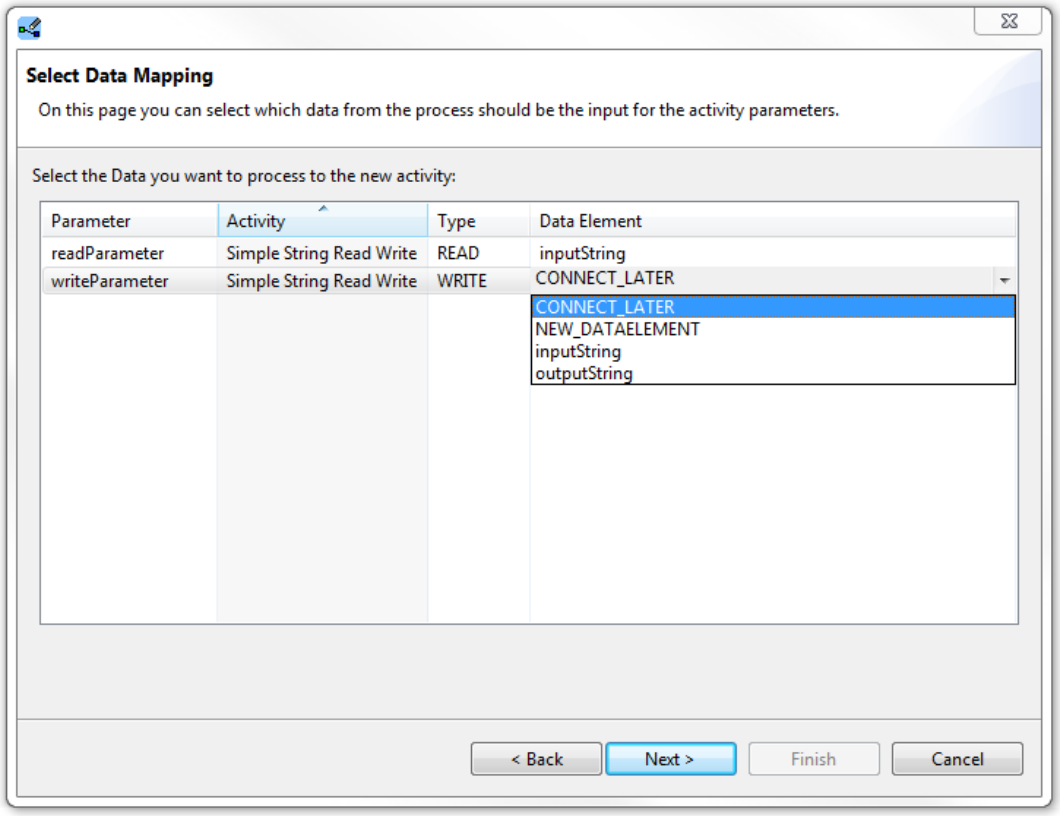


Abbildung 7.7: Festlegung des Datenmappings.

Nachdem der Benutzer die Einfügeposition gewählt hat, kann das System ermitteln, welche Daten zu diesem Zeitpunkt im Prozess versorgt sein werden. Abbildung 7.7 zeigt wie der Benutzer diese Daten verknüpfen kann. Für jeden Parameter einer einzufügenden Aktivität wird in der Liste ein Eintrag vorgenommen. Für jeden Eintrag kann der Benutzer jetzt eine passende Zuordnung vornehmen. Das System bietet ihm hierbei nur solche Daten an, die nach Überprüfung durch die entsprechenden Algorithmen (Kapitel 6.2.3) in Frage kommen. Die vorliegende Aktivität besitzt einen lesenden Parameter “readParameter“, für welchen der Benutzer das im System vorhandene Datum “inputString“auswählt. Für den Ausgabeparameter wählt er die Option CONNECT LATER. Diese Zuordnung kann dann zu einem späteren Zeitpunkt noch geändert werden. Damit der Benutzer die ihm angebotenen Datenelemente unterscheiden kann, sollten diese nicht über sprechende Namen verfügen. Er kann er jederzeit auf die grafische Darstellung des Prozesses zurückgreifen um sich über den Ursprung eines Datums in Kenntnis zu setzen.

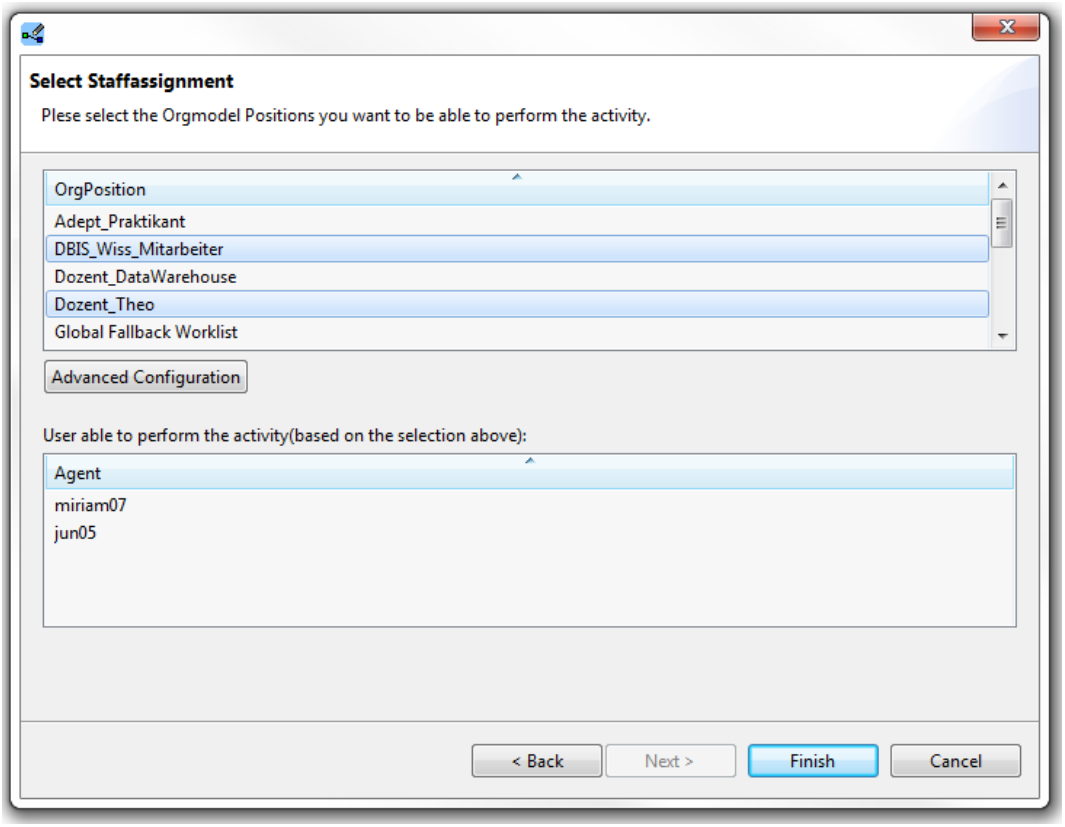


Abbildung 7.8: Bearbeiterzuordnung.

Wurde für jeden Parameter ein korrektes Mapping ausgewählt, bekommt der Nutzer die letzte Maske angezeigt, die Festlegung der Bearbeiterregel (Abbildung 7.8). Die gewählte Aktivität besitzt keine fest vorgegebene Regel für die Bearbeiterzuordnung. Der Benutzer kann nun über geeignete Vorschläge entscheiden, welche Entitäten des Organisationsmodells für die einzufügende Aktivität in Frage kommen. Der Flexclient ermittelt alle im Prozess vorkommenden Orgmodell Positionen und bietet dem Benutzer diese an. Über eine Multiselection kann der Benutzer diese Stellen auswählen und bekommt als Resultat die Bearbeiter angezeigt, welche auf Basis der Auswahl später für den Schritt in Frage kommen. Bestätigt der Benutzer diese letzte Maske, beginnt der eigentliche Einfügevorgang. Unter Verwendung des ChangeOperations-Service beantragt der Flexclient das Einfügen eines neuen, leeren Schrittes. Über entsprechende Methodenaufrufe wird diesem leeren Schritt dann die gewählte Aktivität mitsamt ihrer festgelegten Eigenschaften zugewiesen.

Sollte ein Einfügen nicht möglich sein, wird dem Benutzer mitgeteilt warum dies so ist. Die Oberfläche springt dann zu dem Punkt zurück, an dem ein Fehler behoben werden kann.

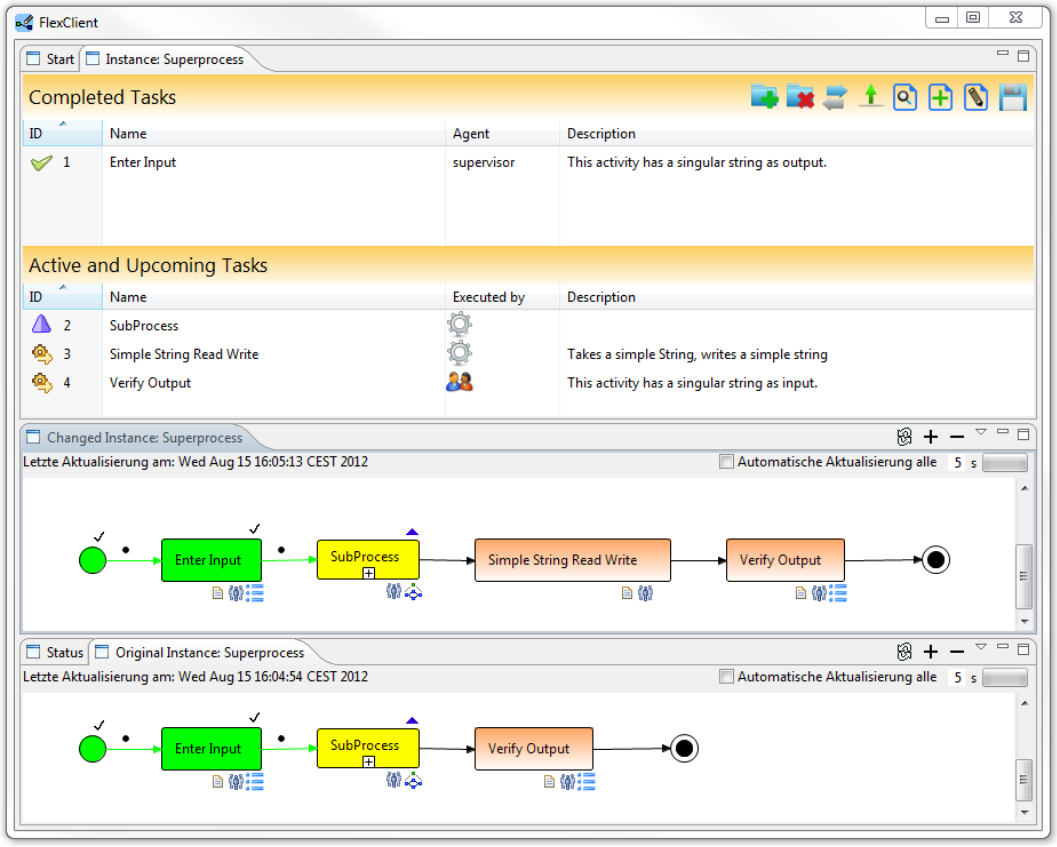


Abbildung 7.9: Ansicht nach Änderung der Instanz.

Sollte das Einfügen, wie in unserem Beispiel, erfolgreich gewesen sein, bekommt der Benutzer wieder die Instanzübersicht des Flexclient angezeigt (Abbildung 7.9). Hierbei werden ihm zwei verschiedene grafische Ansichten angeboten: Die Darstellung der Prozessinstanz in ihrer ursprünglichen Form, als auch nach der jeweiligen Veränderung. Der Benutzer bekommt so ein direktes Ergebnis vermittelt. Er kann nun je nach Belieben weitere Änderungsoperationen anstoßen oder das Editieren beenden und die Prozessinstanz wieder freigeben. Sollten noch unversorgte Daten im Prozess vorhanden sein, bekommt der Benutzer dies mitgeteilt und kann diesen Umstand beheben, indem er entweder einen weiteren Schritt einfügt (Nachforderung), oder das Datenmapping des betroffenen Schrittes ändert. Sind alle Probleme behoben, kann der Flexclient die endgültige Änderung der

Instanz beim InstanceManager von AristaFlow beantragen. Nach Durchführen der nativen Validierungen wird die Instanz dann entsperrt und auf dem Server zur erneuten Ausführung gebracht.

7.3 Ausblick

Der vorgestellte Prototyp stellt eine erste Umsetzung der im Rahmen dieser Arbeit definierten Konzepte dar. Der Fokus lag bei dieser Implementierung darauf, die identifizierten Funktionalitäten als eigenständige Komponente umzusetzen. Ziel war es ebenfalls, diese Komponente so in einem bestehenden Umfeld eines WfMS zu platzieren, dass die bestehende Mächtigkeit voll ausgenutzt werden kann. Es sind noch nicht alle geforderten Konzepte umgesetzt worden. Die Arbeit am vorgestellten Flexclient soll jedoch in der Zukunft im Rahmen der Entwicklung der AristaFlow Plattform weiter voran getrieben werden, damit dann eine vollständige Integration eines solchen Flexclients ein das Endbenutzersystem vorgenommen werden kann.

8 Zusammenfassung

Die vorliegende Arbeit hatte sich zum Ziel gesetzt herauszufinden, welche Schritte notwendig sind, um flexible Ad-hoc Änderungen in einer einfachen, für den Endbenutzer relevanten Art zu präsentieren. Hierzu wurde in grundlegenden Beispielen gezeigt, welche Anforderungen an Funktionalität und Präsentation von Seiten des Benutzers gestellt werden. Danach wurde gezeigt, was ein technisches System leisten muss, um diese Anforderungen umzusetzen. Im letzten Teil wurden konkrete Konzepte vorgestellt, auf Basis deren eine prototypische Implementierung umgesetzt wurde.
Die Eingangs gestellte Frage nach dem notwendigen Aufwand, um ein solches System zu realisieren, kann nicht durch Angabe eines exakten Wertes beantwortet werden. Vielmehr hängt der zu erwartende Aufwand davon ab, in welchem Kontext eine solche Funktionalität geleistet werden soll. Generell ist es definitiv möglich, dem Benutzer flexible Ad-hoc Änderungen verfügbar zu machen. Die Umsetzung wird jedoch in jedem Fall konkret auf ein spezielle Domäne zugeschnitten werden müssen. Damit mit der Aufbereitung für den Endbenutzer kein Verlust von Möglichkeiten einhergeht, wird ein Flexclient perfekt auf ein unterliegendes System abgestimmt und in dessen Umfeld eingegliedert werden. Was ein Benutzer endgültig an einem System leisten können wird, ist stets davon abhängig, wie der Balanceakt zwischen Erhaltung der Mächtigkeit und Vereinfachung der Bedienung bewältigt werden kann. Aus diesem Grund wird es für flexible Ad-hoc Änderungen keine “beste Lösung“, im Sinne eines komplett eigenständigen Informationssystems, geben, sondern jeweils eine kontextorientierte, spezielle Lösung, die auf bestehenden Kompetenzen aufbaut und diese für den jeweiligen Zweck präsentiert.

Die Frage nach dem benötigten Aufwand ist also eine Frage nach dem Aufwand, den man zu leisten bereit ist. Da jedoch die Bedürfnisse nach immer mehr Flexibilität und immer weiter abstrahierten Lösungen auch in Zukunft voraussichtlich nicht abnehmen werden, wird dieser Aufwand geleistet werden müssen, um den Anforderungen der Benutzer begegnen zu können.

Literaturverzeichnis

[1] *IBM Info Center*. http://publib.boulder.ibm.com, Abgerufen am 10.08.2012

[2] *Aristaflow BPM Suite*. http://www.uni-ulm.de/einrichtungen/aristaflow-forum/, Abgerufen am 21.06.2012

[3] DADAM, Peter: *ADEPT2-Ad-hoc-Abweichungen*. Powerpoint Präsentation, 2008

[4] DADAM, Peter ; REICHERT, Manfred: The ADEPT project: a decade of research and development for robust and flexible process support. In: *Springer: Vol.23, No. 2, 2009, special issue on 'Flexible Process-aware Information Systems', S.81-98*

[5] DADAM, Peter ; REICHERT, Manfred ; RINDERLE, Stefanie ; ATKINSON, Colin: Auf dem Weg zu prozessorientierten Informationssystemen der nächsten Generation - Herausforderungen und Lösungskonzepte. In: *Aktuelle Trends in der Softwareforschung - Tagungsband zum doIT-Forschungstag, Juni 2005, Karlsruhe, S.47-67*

[6] DADAM, Peter ; REICHERT, Manfred ; RINDERLE-MA, Stefanie ; GÖSER, Kevin ; KREHER, Ulrich ; JURISCH, Martin: Von ADEPT zu Aristaflow BPM Suite - Eine Vision wird Realität. In: *Ulmer Informatik-Berichte, Nr.2009-02, Januar 2009*

[7] GÖSER, Kevin: *Plug&Play-Aspekte und Integration Zustandsbehafteter Daten in Prozess-Management-Systeme*, Universität Ulm, Diplomarbeit, 2005

[8] MAURONER, Christoph: *Ausführung von variantenbehafteten Workflow-Modellen in Abhängigkeit vom Prozesskontext*, Universität Ulm, Diplomarbeit, 2009

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

[10] REICHERT, Manfred ; DADAM, Peter: Enabling Adaptive Process-Aware Information Systems with ADEPT2. In: *Handbook of Research on Business Process Modeling; J.Cardoso, W.M.P. van der Aalst(Eds.); IGI Global Publ., 2009, Kapitel 8*

[11] REICHERT, Manfred ; STOLL, Dietmar: Komposition, Choreographie und Orchestrierung von Web Services - Ein Überblick. In: *EMISA Forum, Band 24, Heft 2 2004, S.21-32*

[12] STOLL, Dietmar: *Komposition von Web Services*, Universität Ulm, Diplomarbeit, 2004

Abbildungsverzeichnis

Name: Florian Rapp Matrikelnummer: 627856

Erklärung

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

Ulm, den .....

Florian Rapp