



Diplomarbeit 2007

Studiengang Wirtschaftsinformatik

BPMS webMethods Fabric 7.1 für KMU



Student : Josef Schaller

Dozent : Laurent Bagnoud

Vorwort

Die vorliegende Diplomarbeit wurde im Rahmen des Abschlusses der Ausbildung zum Wirtschaftsinformatiker FH an der HES-SO geschrieben.

In den zwölf Wochen meiner Tätigkeit wurde das Business Process Management System von webMethods 7.1 im Zusammenhang mit der Entwicklung einer serviceorientierten Architektur näher untersucht.

In der heutigen Zeit wird vermehrt der Versuch unternommen, die Verbindung zwischen Management und Informationstechnologie zu schaffen. Der angeblich hervorragende Return on Investment lässt viele Managerherzen höher schlagen.

Doch was ist an diesen Vermutungen? Ist dies alles nur ein Trend der durch einen nächsten wieder abgelöst wird? Wie können mit Hilfe von Business Process Management Systemen serviceorientierte Architekturen unterstützt und aufgebaut werden? Wie viel Potential liegt darin, wie kann dies alles in bestehende Strukturen integriert werden und welchen Nutzen können kleine bis mittlere Unternehmungen davon haben?

Dies sind nur einige der Fragen die im Verlaufe dieser Arbeit beantwortet werden. Es wird erklärt, wie das Business Process Management System von webMethods für die Akteure einer bestehenden Logistikkette integriert und eine serviceorientierte Architektur aufgebaut werden kann.

Ich möchte es hier nicht unterlassen Dank an meine Familie auszusprechen, die mir während meiner Studienzeit, in jeder Hinsicht, tatkräftig zur Seite stand.

Siders, Dezember 2007

Zusammenfassung

Ein Business Process Management System ermöglicht die Modellierung, Integration, Simulation und Überwachung von Geschäftsprozessen.

Die Frage stellt sich hinsichtlich der Integration eines solchen Business Process Management Systems und eines Aufbaus einer serviceorientierten Architektur.

Das Problem liegt in der Natur der Vielfältigkeit der eingesetzten Softwareapplikationen der informationsaustauschenden Partner. Sei dies nun inter- oder intraspezifisch.

Wie kann ein Händler angebotene Dienste, wie Web Services eines Lieferanten, nutzen, im Wissen der unterschiedlich benutzten Software und wie ist dies zu vereinen mit dem Business Process Management?

Eine mögliche Beantwortung der Frage bietet der Einsatz eines Business Process Management Systems und einer serviceorientierten Architektur.

Das Business Process Management System ermöglicht die Administration von Prozessen im Geschäftsablauf und die serviceorientierte Architektur bietet eine Art Kommunikationsoberfläche für diese Prozesse, in Form von Web Services oder anderen eingesetzten Diensten.

Zum Aufzeigen eines Lösungsansatzes wird das Business Process Management System webMethods 7.1 für drei Unternehmungen eingeführt. Die nötigen Geschäftsprozesse modelliert, Programmlogik erweitert, integriert und Prozesse überwacht werden.

Das Resultat der Integration des Business Process Management Systems zeigt Vorteile hinsichtlich Optimierung, Automation und Kommunikation von Geschäftsprozessen und weist ganz klare Vorteile gegenüber bisherigen Methoden auf.

Weiter wird durch den Einsatz eines Business Process Management Systems die Neuentwicklung von bereits bestehender Softwarelogik verhindert.

Inhaltsverzeichnis

1	KONVENTIONEN.....	6
2	EINFÜHRUNG.....	7
3	STAND DER TECHNIK.....	9
4	ARCHITEKTUR WEBMETHODS 7.1.....	13
4.1	<i>BROKER</i>	13
4.2	DER INTEGRATIONSSERVER (IS)	15
4.3	<i>TRADING NETWORKS</i>	16
4.4	<i>MY WEBMETHODS SERVER</i>	18
4.5	<i>MONITOR</i>	19
4.6	<i>OPTIMIZE FOR INFRASTRUCTURE</i>	20
4.7	<i>OPTIMIZE FOR PROCESS</i>	21
4.8	<i>TASK ENGINE</i>	23
5	KURZÜBERBLICK WEBMETHODS 7.1 WERKZEUGE	24
6	HANDHABUNG WEBMETHODS 7.1.....	25
6.1	DER <i>DESIGNER 7.1</i>	25
6.2	MODELLIERUNG.....	26
6.3	DER DEVELOPER 7.1.....	31
6.4	<i>MONITORING</i>	32
6.5	INTEGRATION.....	32
7	VERWALTUNG WEBMETHODS 7.1	34
	<i>Einführung</i>	34
7.1	SYSTEMVORAUSSETZUNGEN	34
7.2	DATENBANKEN	35
7.3	DATENRETTUNG UND <i>BACKUPS</i>	36
7.4	AKTUALISIERUNG UND SUPPORT.....	37
7.5	BENUTZERVERWALTUNG.....	38
7.6	PROZESSMANAGEMENT	39
7.7	<i>TASK</i> AUSFÜHRUNG IN <i>MY WEBMETHODS</i>	40
8	BEISPIELPROZESS <i>TPROCESS3</i>.....	41
8.1	EINLEITUNG	41
8.2	ZUSAMMENFASSUNG	41
8.3	DAS MODELL.....	41
	<i>Receive Step</i>	41
	<i>hInput</i>	42
	<i>wsEncPHP</i>	42

	<i>saveOrderDB</i>	42
	<i>Notification</i>	42
8.4	DIE ENTWICKLUNG.....	42
	<i>retAlarmDoc</i>	42
	<i>hlInput Task</i>	43
	<i>wsEncPHP</i>	44
	<i>saveOrderDB</i>	44
	<i>Notification</i>	45
8.5	<i>BUILD AND UPLOAD</i> BEISPIELPROZESS <i>TPROCESS3</i>	45
8.6	GESCHÄFTSPROZESS <i>TPROCESS3</i> STARTEN	46
8.7	<i>MONITORING</i> BEISPIELPROZESS <i>TPROCESS3</i>	47
9	SZENARIO	49
10	BPMS HÄNDLER	50
10.1	EINFÜHRUNG	50
	<i>Bemerkung</i>	50
	<i>Überblick</i>	50
10.2	ARCHITEKTUR HÄNDLER.....	51
10.3	DAS MODELL	52
10.4	DIE EINZELNEN PROZESSSCHRITTE	52
	<i>orderInput</i>	52
	<i>wsNewOrder</i>	53
	<i>newOrderCheck</i>	53
	<i>receiveNoticeOfDelivery</i> und <i>receiveLoadingLeft</i>	54
	<i>confirmDelivery</i>	54
	<i>wsSendConfirmationAoD</i> und <i>wsSendConfirmationOfDelivery</i>	54
	<i>confirmationCheck</i>	54
10.5	ENTWICKLUNG.....	54
11	BPMS ENCAVEUR.....	57
11.1	EINFÜHRUNG	57
	<i>Überblick</i>	57
11.2	ARCHITEKTUR ENCAVEUR.....	57
	<i>Presentation Layer</i>	59
	<i>Business Layer</i>	59
	<i>Data Layer</i>	60
11.3	SOA ENCAVEUR – <i>PHP</i> SERVER UND <i>WEB SERVICE</i>	60
11.4	DAS MODELL	62
11.5	DIE EINZELNEN PROZESSSCHRITTE	63
	<i>receiveOrder</i>	63
	<i>infoCollector</i>	63
	<i>checkConfirmationTransport</i>	64
	<i>wsRequestTransport</i>	64
	<i>receiveConfirmation</i>	64
	<i>wsSendNoticeOfDelivery</i>	64
	<i>checkNOF</i>	64

<i>receiveDeliveryConfirmation</i>	64
12 BPMS TRANSPORTEUR	65
12.1 EINFÜHRUNG	65
<i>Bemerkungen</i>	65
<i>Überblick</i>	65
12.2 ARCHITEKTUR TRANSPORTEUR	65
12.3 DAS MODELL	65
13 SERVICE LEVEL AGREEMENTS (SLA)	67
14 DIE NOTWENDIGKEIT EINES BPMS FÜR KMU'S	70
15 PROBLEME WÄHREND DER DIPLOMARBEIT	71
15.1 VERSPÄTUNG DER SOFTWARE	71
15.2 RESSOURCEN	71
<i>Vorgaben</i>	71
<i>Virtuelles System</i>	72
<i>GSX Server</i>	72
<i>Auslagerung</i>	72
<i>Neues Image</i>	73
<i>Adapter</i>	73
<i>Fazit</i>	74
16 SYNTHESE	75
17 SCHLUSSWORT	77
18 EHRENWÖRTLICHE ERKLÄRUNG	78
19 ABBILDUNGSVERZEICHNIS	79
20 TABELLENVERZEICHNIS	82
21 CODEVERZEICHNIS	83

1 Konventionen

Aufgrund des teilweise besseren Verständnisses englischsprachiger Begriffe in den Bereichen der Informatik werden diese in der vorliegenden Arbeit nicht ins Deutsche übersetzt, oder nur in wenigen Fällen.

Solche Begriffe werden durch einen *kursiven Stil* kenntlich gemacht. In derselben Weise werden *Kapitelangaben*, *Variablen* und *spezielle Ausdrücke* behandelt.

Ausgeschlossen von dieser Konvention sind Begriffe im Vorwort, der Zusammenfassung, in Zitaten, Fussnoten, Quellenangaben und Tabellen.

Sehr häufig ist der Begriff *webMethods* anzutreffen. *webMethods* ist die Unternehmung die die *Business Process Management System (BPMS)* Software entwickelt.

Auf den offiziellen Webseiten und vielen Dokumenten ist das *BPMS* von *webMethods* als *webMethods 7.1* oder *webMethods Fabric 7.1* gekennzeichnet.

Zur Vereinfachung wurde in den Kapiteln der Arbeit der Begriff *webMethods* und *webMethods 7.1* verwendet. Ob die Unternehmung oder die Software gemeint ist, ist aus dem Zusammenhang erkennbar.

webMethods wurde am 1. Juni 2007 von der Software AG übernommen.

Fussnoten im Titel eines Kapitels bedeuten, dass Referenzdokumente zur Erarbeitung für das Kapitel hinzugezogen worden sind. Die Quellen sind am Seitenende aufgelistet.

2 Einführung

Ein wichtiger und nicht zu unterschätzender Trend in weiten Teilen der Informationstechnologie ist der vermehrte Einsatz von *Business Process Management Systemen* zur Modellierung, Simulation, Optimierung und Überwachung von Geschäftsprozessen.

In diesem Zusammenhang sind das Anbieten und Konsumieren von Diensten wie *Web Services* ein wichtiger Punkt.

Prozesse können so weit aufgespaltet werden, dass den Prozessschritten einzelne Services zugrunde gelegt werden können und somit ein modularer Aufbau vollzogen wird.

Ein grosser Vorteil von solchen Services ist die Möglichkeit, dass viele Partner diese Services nutzen können, ohne grossartige Änderungen an bestehende Softwareapplikationen vornehmen zu müssen.

Die oben erwähnte Thematik steckt noch in den Kinderschuhen, Standards sind teilweise noch nicht definiert, *Business Process Management Systeme* und serviceorientierte Architekturen weisen ein enormes Potential für die nächsten Jahre auf.

Nur schon aus diesen Gründen lohnt sich eine nähere Betrachtung von solchen Systemen und Architekturen.

Die Herausforderung liegt nun darin, ein solches *Business Process Management System (BPMS)* für drei kleine und mittlere Unternehmungen zu installieren.

Es gilt herauszufinden was das *BPMS webMethods 7.1* kann, wie eine Integration und Implementierung von statten geht, welche Rolle eine servicorientierte Architektur einnimmt und welchen Nutzen dies für kleine und mittlere Unternehmungen hat.

Zur Bewältigung der oben erwähnte Herausforderung und Klärung der aufgeworfenen Fragen wird auf nachfolgende Punkte näher eingegangen:

- Der aktuelle Stand der Technik
- Architektur des *BPMS* von *webMethods 7.1*
- Entwicklung und Untersuchung eines einfachen Geschäftsprozesses in *webMethods 7.1*
- Integration des *BPMS* von *webMethods 7.1* für die verschiedenen Akteure einer bestehende Logistikkette
- Resultate

Zum besseren Verständnis werden im nächsten Abschnitt generelle Informationen zu der Logistikkette aufgeführt.

Es handelt sich um drei potentielle Unternehmungen die miteinander interagieren. Ein Weinproduzent, ein Weinhändler und ein Transporteur. Durch das Anbieten von verschiedenen *Web Services* sollen die Akteure untereinander kommunizieren können, d.h. zum Beispiel soll der Händler eine Bestellung über den *Web Service* des Encaveurs machen können oder der Encaveur stellt eine Transportanfrage an den Transporteur um Ware auszuliefern.

Diese Services sollen alle in die Geschäftsprozesse der Akteure integriert und mit Hilfe des *BPMS* von *webMethods* überwacht werden.

Anmerkung: Detaillierte Informationen zum Szenario und den technischen Schnittstellen finden sich im Anhang.

3 Stand der Technik

Das Angebot von *Business Process Management Systemen (BPMS)* auf dem Markt ist relativ unübersichtlich. Standards sind noch in vielen Teilgebieten dieses Bereichs undefiniert. Auch wird oft im Zusammenhang mit Geschäftsprozessen das *Workflow Management (WFM)* erwähnt. Es lohnt sich den Unterschied zwischen *BPM* und *WFM* Systemen zu machen, denn der Softwaremarkt für beide dieser Managementsysteme ist sehr vielfältig.

„Workflow Management Systeme sollten aus drei Elementen bestehen, einem Designtool zur Darstellung des Ablaufmodells, der Workflow Engine zur Steuerung der Prozesse zur Laufzeit und einer Sammlung aller prozessrelevanten Daten in den AUDIT-Trails.

Business Process Management Systeme wurden darüber hinaus zum einen um die Elemente Modellentwurf zu Analysezwecken und zur Simulation und zum anderen um Reportingmechanismen zur Auswertung der AUDIT TRAILS erweitert, um mit der Rückführung der Ergebnisse in die Analyse zur Verbesserung der Modelle den Lifecycle zu schließen.“¹

Ein weiteres Schlagwort im Zusammenhang mit Business Process Management ist *SOA (Service Oriented Architecture)*. „Durch SOA können benötigte Funktionalitäten oder Dienste dynamisch zur Laufzeit eingebunden und aufgerufen werden.“²

BPMS bieten die Möglichkeit eine *SOA* aufzubauen. Services die mit *BPMS* entwickelt wurden können beispielsweise ausgegliedert und für andere zugänglich gemacht werden.

Die meisten *BPMS* auf dem Markt ermöglichen mit den gegebenen Funktionalitäten einen guten Aufbau für eine *SOA*, vor allem was die Handhabung von *Web Services* betrifft.

Ein wichtiger Punkt bei einem *BPMS* sind auch die Anbindungsmöglichkeiten an bestehende Komponenten die in der Unternehmung eingesetzt werden.

Bestehende Datenbanken sollen nicht ersetzt werden müssen, aus dem Grund müssen *BPMS* gute Adaptionmöglichkeiten aufweisen. Unterstützung von

¹ Quelle: Renate Karl, Dr. Ulrich Kampffmeyer. (05/2006). Einführung von Business Process Management- und Workflow Systemen [Online]. Erreichbar unter: http://www.documanager.de/magazin/artikel_1023_einfuehrung_von_business_process_management-.html [Letzter Zugriff: 04.12..2007]

² Melzer Ingo et al. Service –orientierte Architekturen mit Web Services. Konzepte-Standards-Praxis. 2. Auflage. Elsevier GmbH, 2007.

JDBC (Java Database Connectivity) das „Equivalent zu *ODBC (Open Database Connectivity)* in der Java Welt“³ gilt als eine Mindestvoraussetzung.

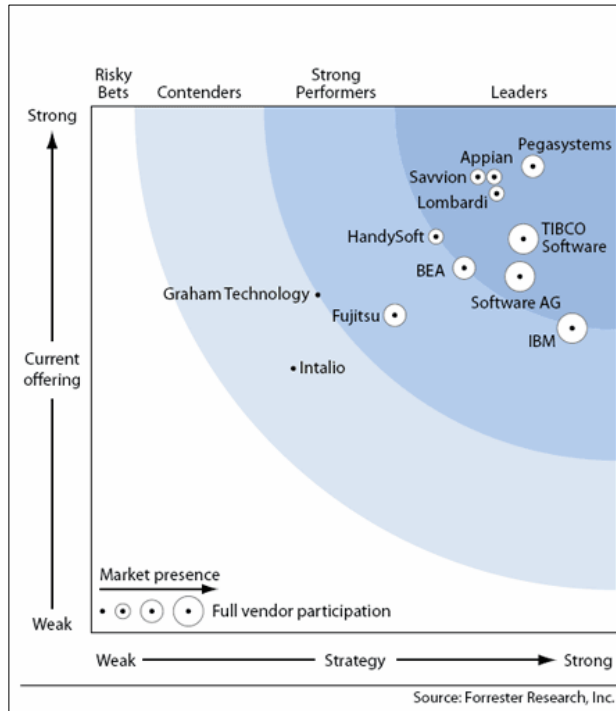


Abbildung 1 : Anbieter BPMS Software und ihre Stellung auf dem Markt. Quelle: The Forrester Wave: Human-Centric BPM for Java Platforms, Q3 2007

Aktuell gibt es verschiedene Software Applikationen die als *BPMS* Werkzeuge eingesetzt werden können. So zum Beispiel *BEA Aqualogic*, *Agentflow* von *Flowring* oder *IBM WebSphere BPM Suite*. Sogar Open Source Lösungen wie *BPM 2.0* von *Intalio* stehen zur Verfügung. Spezifikationen und Bemerkungen zu den erwähnten Systemen sind im Kapitel *Diverse BPMS Produkte* im Anhang enthalten.

Aber so viele verschiedene BPMS erhältlich sind, so viele Unterschiede bestehen, vor allem was die

Standardisierung der einzelner Komponenten betrifft. Bei der Integration eines *BPMS* sollte daher unbedingt auf die benutzten Standards Wert gelegt werden. Was dies die Prozessausführungssprache betrifft ist *BPEL (Business Process Execution Language)* ein Quasistandard, was für die Notation *BPMN (Business Process Management Notation)* ist.

Eine native Unterstützung von *BPEL* und *BPMN* ermöglicht das Open Source Produkt von *Intalio*.

Von vielen BPMS wird die eigene Einfachheit gelobt, dass *Business Analysten* Prozesse modellieren können und dann dem Entwickler übergeben. Diese Differenzierung besteht beispielsweise beim *webMethods BPMS*.

Man muss sich aber im Klaren sein, dass eine solche Differenzierung nur schwer zu verwirklichen ist, da ein Grundverständnis für die Entwicklung

³ James F. Chang, Business Process Management Systems, Strategy and Implementation. Auerbach Publications, 2006.

bestehen sollte. Ansonsten könnte es sein, dass die Arbeit des *Business Analysten* erheblich durch den Entwickler überarbeitet werden muss.

Dies kann man mit den modellierten Prozessen vergleichen: Ist der *Output* des Vorgängerschritts nicht mit dem *Input* des darauf folgenden Schritts konform, hat man ein Problem im Prozess.

Bevor man sich für ein Produkt auf dem Markt entscheidet ist es wesentlich abzuklären wie und ob man überhaupt ein *Business Process Management* und *Business Process Management System* einführen kann oder muss.

Grundsätzlich ist dies für die meisten Unternehmungen möglich und auch sinnvoll, aber nur schon in Anbetracht der meist hohen Kosten für die *BPMS* kann der vielbeschworene *ROI (Return on Investment)* geschwächt wirken, vor allem für kleine und mittlere Unternehmungen (KMU).

Weiter dürfen je nach Komplexität die Implementierung für das *BPMS* nicht unterschätzt werden. Dies führt zu der Frage ob es eventuell Alternativen für *BPMS* gibt, vor allem für *KMU's*.

Bei kleineren Unternehmungen bestehen Möglichkeiten für eine indirekte Optimierung von Prozessen. Denkt man an die Präsenz von Unternehmungen im Internet, kann schon das Anbieten von Produkten über eine Webseite kleinere Prozesse optimieren. Dies vielleicht eher als Zusatz zu den bestehenden Möglichkeiten. *E-Commerce* und *E-Shop* Lösungen gibt es zahlreiche auf dem Markt und sind mit weniger Implementierungsaufwand verbunden.

Auch elektronische Marktplätze sind eine Option, so genannte *E-Marketplaces*. Ein *E-Marketplace* ist ein elektronischer Handelsplatz auf dem sich Käufer und Verkäufer registrieren und dadurch kommunizieren und Geschäfte abwickeln können.⁴

IBM bietet eine *Hosting* Dienst für elektronische Marktplätze an. "Diese Hosting-Services bieten Unterstützung von Anwendungen für die Aktivierung von E-Procurement- und E-Marketplace-Lösungen."⁵

Egal ob es sich um eine kleine, mittlere oder grosse Unternehmung handelt. Bevor man sich für ein *BPMS* entscheidet sollte man sich über einige Faktoren wie die folgenden Gedanken machen:

- Welche Geschäftsstrategie verfolgt man
- Anzahl und Analyse aktueller Prozesse
- Bestehende Automation

⁴ Quelle: E-business guide, an australian guide to doing business online. [Online]. Erreichbar unter: <http://www.e-businessguide.gov.au/improving/e-marketplaces> [Letzter Zugriff: 04.12.2007]

⁵ Quelle: IBM. Evolving Technology. [Online]. Erreichbar unter: <http://www-05.ibm.com/e-business/ch/evolving/integration/component/scm.html> [Letzter Zugriff: 04.12.2007]

- Benutzte Informationstechnologie
- Aktueller und erwarteter *ROI*
- Verfügbares Budget
- *SOA* Integration

Man kann sagen, dass es für quasi alle nötigen Erfordernisse gute Lösungen gibt. Aber auch das beste *BPMS* kann für einige Unternehmungen die Abläufe unter Umständen komplizieren und keinen Nutzen mit sich bringen, insbesondere für kleine und mittlere Unternehmungen.

4 Architektur webMethods 7.1

4.1 *Broker*⁶

Der *Broker* stellt das Nachrichtenrückgrat einer *webMethods* Implementierung dar. Der ihm zugrunde liegende Informationsaustausch ist durch das *publish and subscribe* Modell beschrieben.

Beim *publish and subscribe* Modell werden Nachrichten zwischen den einzelnen Komponenten über den *Message Broker* verteilt. Der *Broker* erhält die Informationen in der Form von Dokumenten von demjenigen der es publizieren will, also dem *Publisher*, danach verteilt der *Broker* diese Informationen an den, der die Informationen will, den *Subscriber*.

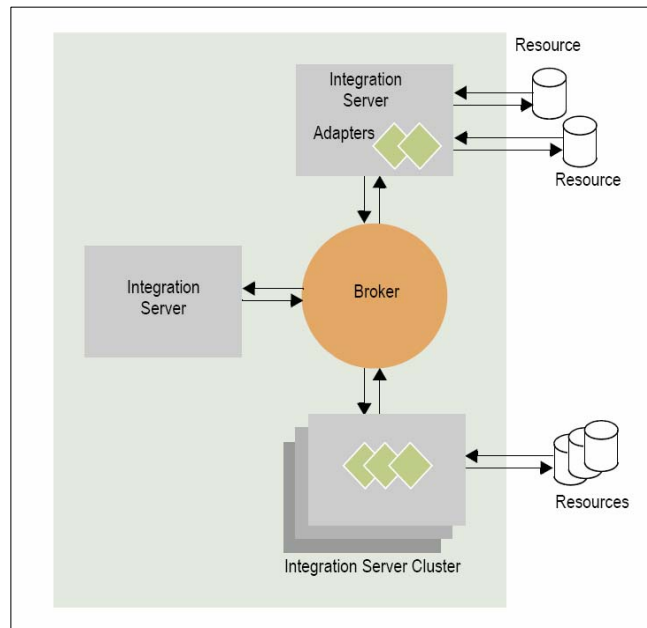


Abbildung 2 : Interaktion Broker mit Integrationsservern.
Quelle: Publish-Subscribe_Developer's_Guide_7_1.pdf

Der *Broker* erkennt aufgrund der Registrierung der Dokumententypen welche Nachrichten er weiterleiten kann. Auch die Konsumenten der Informationen sind im Broker präsent.

Erhält dieser ein akzeptables Dokument kommt es in die Warteschlange und der *Subscriber* kann es dort abholen.

Es ist durchaus möglich und in vielen Fällen anzutreffen, dass mehrere *Broker*

⁶ Quelle: webMethods. webMethods Broker Administrator's Guide.webMethods_Broker_Administrator's_Guide_7_1.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 30.11.2007]

Quelle: webMethods. Publish and Subscribe Developer's Guide.Publish-Subscribe_Developer's_Guide_7_1.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 30.11.2007]

vorhanden sind, beispielsweise in verschiedenen Gruppen, auch *Territories* genannt.

Dokumente erhalten, in die Warteschlange stellen und Dokumente ausliefern, charakterisieren die Aufgaben des *Brokers*.

Ohne einen *Broker* besteht die Möglichkeit eines lokalen Publizierens, dabei können nur Services auf dem lokalen Integrationsserver benutzt werden. In diesem Fall übernimmt der Integrationsserver die Rollen des *Publishers* und des *Subscribers*.

Weiter kann der *Broker* als *JMS (Java Messaging Service) Provider* fungieren und erlaubt den Nachrichtenaustausch von *JMS*-konformen Nachrichten.

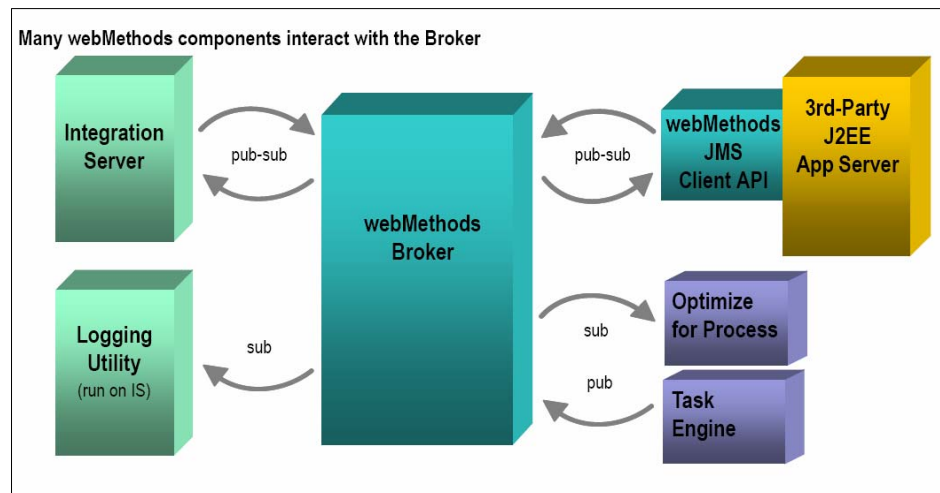


Abbildung 3 : Interaktion des Brokers mit verschiedenen Komponenten.

Quelle: webMethods_Broker_Administrator's_Guide_7_1.pdf

Die in der *Task Engine* gesammelten Instanzwerte aus der Laufzeitumgebung werden mittels Broker an die Komponente *Optimize for Process* weitergeben. Die *Task Engine* ist *Publisher* und *Optimize for Process* der *Subscriber* dieser Informationen.

Auch werden mit Hilfe des *Brokers* Logdaten, je nach Konfiguration in die Datenbank gespeichert.

Eine weitere Aufgabe des *Brokers* ist das Auslösen von Prozessschritten in der *Process Engine*. Die *Process Engine* befindet sich auf dem Integrationsserver.

Triggers erstellen eine Subskription eines zu publizierenden Dokuments und spezifizieren Services die diese Dokumente weiterverarbeiten.

Die Brokeradministration erfolgt über die grafische Schnittstelle von *My webMethods*.

4.2 Der Integrationsserver (IS)⁷

Der Integrationsserver stellt eine zentrale Instanz bei jedem *webMethods* Integrationszenario dar. Eine der wohl wichtigsten Funktionalitäten bildet die Verwaltung der Services. Die Services werden in Paketen verwaltet. Die

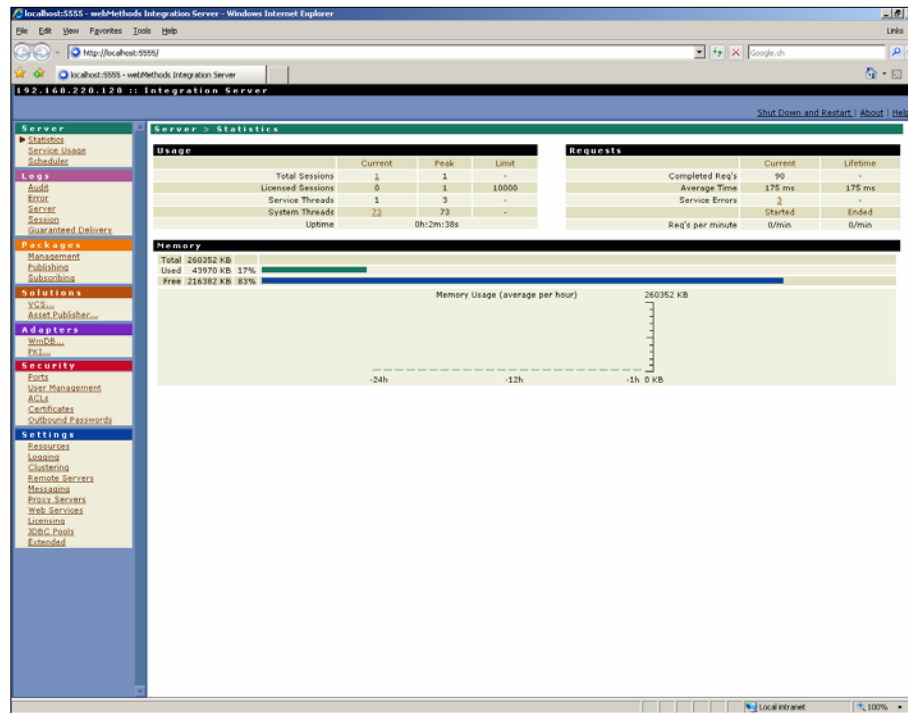


Abbildung 4 : Administrationsoberfläche Integrationsserver.

Quelle: Schaller Josef

Verwaltung der Pakete sowie sämtliche Konfigurationseinstellungen betreffend der Aktion und Interaktion des IS sind über einen Browser möglich.

Der Austausch der Informationen geschieht über einen oder mehrere *Broker*.

Der IS ist eine *Service Engine*. Der *Service Input* wird an den IS gesendet, der die Logik durchführt und den *Output* generiert.

⁷ Quelle: Ladson Richard. What is webMethods. [Online]. Erreichbar unter:

http://www.wmusers.com/knowledge/whitepapers/wMUsers-WhatIsWebMethods_v2.pdf
[Letzter Zugriff: 04.12.2007]

Quelle: webMethods. Integration Server Administration Guide.

webMethods_Integration_Server_Administrator's_Guide_7_1.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 25.11.2007]

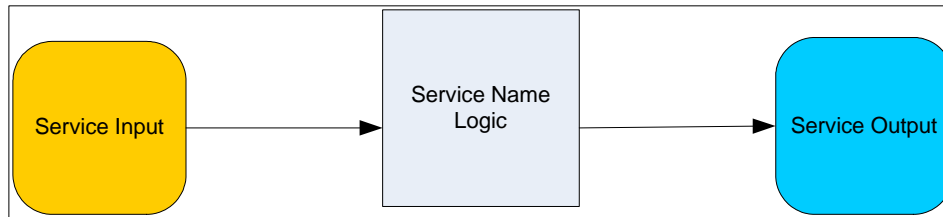


Abbildung 5 : Serviceverarbeitung

Quelle: Schaller Josef in Anlehnung an wMUsers What is webMethods v2

Dabei können innerhalb des IS, verschiedene Services kombiniert und modular zusammengestellt sein.

4.3 Trading Networks⁸

Trading Networks ist ein Service und läuft auf dem IS. Über *Trading Networks* können interne und externe Partner zusammengeführt werden um einen Marktplatz zu formen.

Der *My webMethods Server* ist ein Laufzeitcontainer und stellt Funktionen der einzelnen *webMethods* Komponenten zur Verfügung.

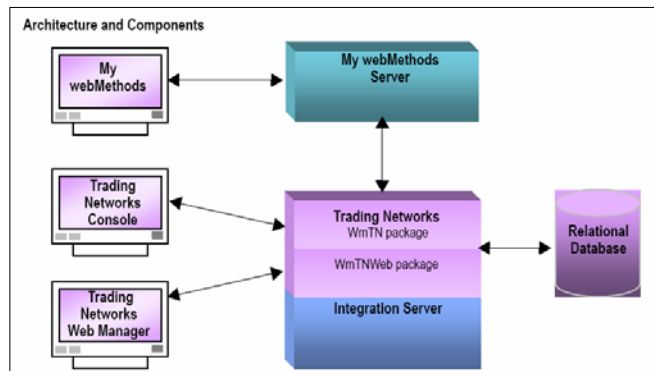


Abbildung 6 : Architektur und Komponenten.

Quelle : Trading Networks Concept Guide 7.1

So stellt der *My webMethods Server* die Funktionen des *Trading Networks*, welches auf dem IS läuft zur Verfügung. Beispielsweise die Überwachung und Verwaltung von Transaktionen.

Die Pakete *WmTN* und *WmTNWeb* erlauben die Verwaltung von Partnern im Netzwerk und den Dokumentaustausch.

Die Verwaltung von *Trading Networks* kann durch drei Varianten verwaltet werden: *My webMethods*, *Trading Networks Console* oder den *Trading Networks Web Manager*.

⁸ Quelle: webMethods. Trading Networks Concept Guide 7.1.

webMethods_Trading_Networks_Concepts_Guide_7_1.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 20.11.2007]

Quelle: Ladson Richard. What is webMethods. [Online]. Erreichbar unter:

http://www.wmusers.com/knowledge/whitepapers/wMUsers-WhatIsWebMethods_v2.pdf [Letzter Zugriff: 04.12.2007]

Komponenten von webMethods können durch *Trading Networks* viele *E-Business* Standards unterstützen, wie: *RosettaNet*, *EDI*, *ebXML Messaging Service*, *SWIFT*, *FIX* und *CIDX*.

4.4 My webMethods Server

Der *My webMethods Server* kann als eine Art Container angesehen werden, in dem die Funktionen der verschiedenen Komponenten in Laufzeit ausgeführt



Abbildung 7 : Loginseite für Administratoren und Benutzer.
Quelle: Schaller Josef

werden können. Die Funktionen können aufgrund von Zugriffsrechten kontrolliert werden.

Für weiterführende Informationen können auch die Kapitel:

- *Broker*
- *Trading Networks*
- *Monitor*
- *Optimize for Process*
- *Optimize for Infrastructure*
- *Task Engine*

konsultiert werden.

Administrator und Benutzer könne sich über einen Browser auf den *My webMethods Server* einloggen.

4.5 Monitor⁹

Durch den *Monitor* kann man Daten, die von anderen Komponenten erstellt oder geloggt wurden, anschauen.

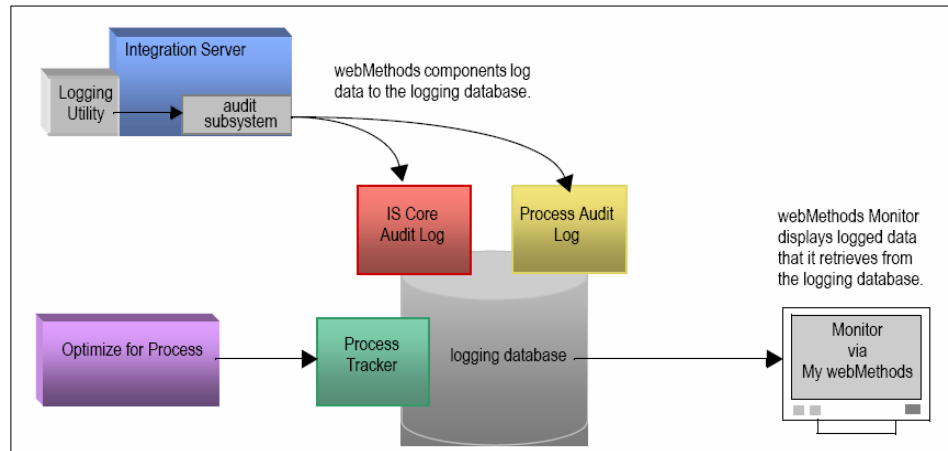


Abbildung 8 : Architektur mit Monitor.

Quelle Monitor User's Guide 7.1

Solche Daten können aus Geschäftsprozessen, Services oder Dokumenten stammen. Zusätzlich können solche Daten bearbeitet und wieder vorgelegt werden.

Monitor kann auch dazu benutzt werden, Daten aus dem *IS Core Audit Log* und dem *Process Audit Log* zu löschen.

⁹ Quelle: webMethods. Monitor's User Guide 7.1. webMethods_Monitor_User's_Guide_7_1-3.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 25.11.2007]

4.6 *Optimize for Infrastructure*¹⁰

Optimize for Infrastructure ermöglicht die Überwachung sämtlicher Komponenten von *webMethods* in Echtzeit. *Optimize for Infrastructure* besteht selber aus den Komponenten: *Infrastructure Data Collector* und *Optimize*.

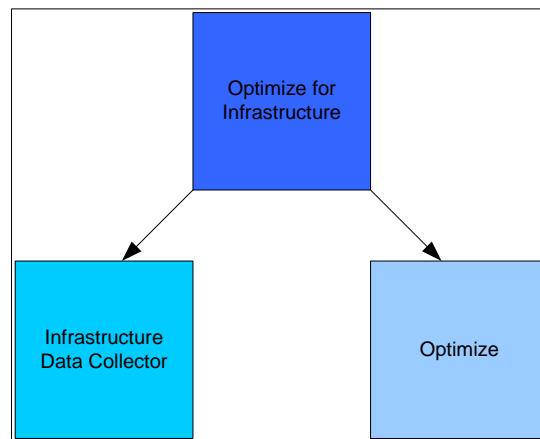


Abbildung 9 : Komponenten von *Optimize for Infrastructure*.

Quelle: Schaller Josef

Komponente	Beschreibung
Infrastructure Data Collector	Zur System- und operationellen Datenüberwachung der <i>webMethods</i> Laufzeitkomponenten. Diese Informationen werden an <i>Optimize for Infrastructure</i> weitergeben.
Optimize	Ermöglicht den Zugriff auf Ressourceninformationen die vom "Infrastructure Data Collector" gesammelt wurden.

Tabelle 1 : Komponentenbeschreibung *Optimize for Infrastructure*.

Quelle: Schaller Josef

¹⁰ Quelle: *webMethods. Optimize for Infrastructure Administrators Guide 7.1*.

webMethods_Optimize_for_Infrastructure_Administrator's_Guide_7_1-3.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 15.11.2007]

4.7 Optimize for Process¹¹

Optimize for Process dient der Analyse der in echtzeitablaufenden Prozesse und liefert darüber Informationen, wie *Performance* der Prozesse, Fehler die aufgetreten sind und *Key Performance Indicators*. „Key Performance Indicators are quantifiable measurements, agreed to beforehand, that reflect the critical success factors of an organization.“¹²

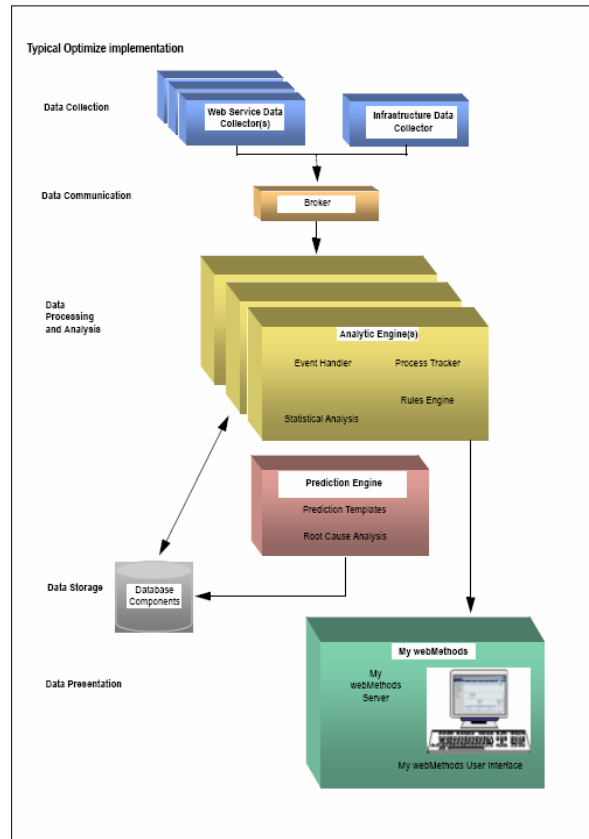


Abbildung 10 : Typische Optimize Infrastruktur
Quelle: Optimize for Infrastructure Administrator's Guide 7.1

¹¹ Quelle: webMethods. Optimize for Process User's Guide 7.1. webMethods_webMethods_Optimize_for_Process_User's_Guide_7_1.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 15.11.2007]

Quelle: webMethods. Optimize for Infrastructure Administrators Guide 7.1. webMethods_Optimize_for_Infrastructure_Administrator's_Guide_7_1-3.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 15.11.2007]

¹² Quelle: Reh John F. Key Performance Indicators (2007) [Online]. Erreichbar unter: <http://management.about.com/cs/generalmanagement/a/keyperfindic.htm> [Letzter Zugriff: 06.11.2007]

Schicht	Beschreibung
Data Collection	Optimize sammelt Daten von anderen webMethods Applikationen über den Web Service Data Collector oder Infrastructure Data Collector
Data Communication	Datensammlung über Broker durch Java Messaging System. Daten vom Data Collector gehen zur Optimize Analytic Engine.
Data Processing and Analysis	Die Analytic Engine erhält Geschäfts-, Prozess- und Systemdaten vom Data Collector Die Komponenten einer Analytic Engine sind in der Abbildung 11 enthalten.
Data Storage	Datenspeicherung
Data Presentation	Optimize bereitet die Daten auf, welche über die My webMethods Benutzerschnittstelle angezeigt werden. Auch administrative Handlungen können hier vorgenommen werden. Beispielsweise Rollen, Benutzer etc.

Tabelle 2: Beschreibung der Schichten der Optimize Infrastruktur in Abbildung 10

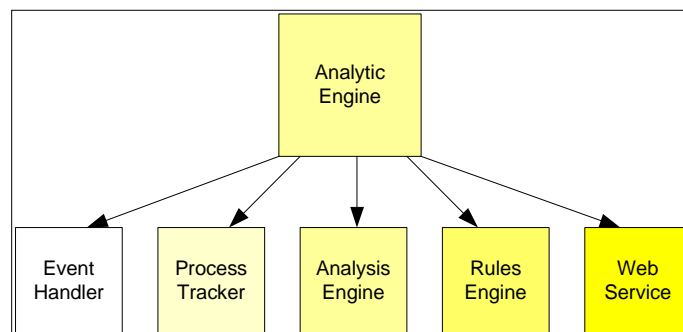


Abbildung 11 : Aufbau Analytic Engine

Quelle: Schaller Josef

Komponente	Aufgabe
Event Handler	Erhält Daten der Data Collectors Informationen über Ausstattung oder Applikationen wie beispielsweise der aktuelle Status einer Komponente: on- oder offline Meldung. Diese Informationen werden in die Datenbank geschrieben, in einem Format, dass eine Business Intelligence Software lesen kann.
Process Tracker	Verfolgung aktuell ausgeführter Prozesse und Alarmmeldung bei etwaigen Problemen.
Analysis Engine	Algorithmenausführung auf den Daten.
Rules Engine	Vergleich der Daten von der Analytic Engine mit gesetzten Regeln. Meldung falls Prozesse eine Benutzerinteraktion benötigen.
Web Service Layer	Resultat der Analyse wird zum My webMethods Server geschickt. Diese Resultate werden in My webMethods angezeigt.

Tabelle 3: Analytic Engine Komponenten und ihre Aufgaben.

4.8 Task Engine

Die *Task Engine* dient der Verwaltung von *Tasks*. Bei *Tasks* unter webMethods handelt es sich um Interaktionen mit Benutzern. Entweder um Daten zu bestätigen oder solche beizufügen.

Der *Business Analyst* oder auch der Entwickler können im webMethods *Designer 7.1 Tasks* erstellen, diese werden auf den *My webMethods Server* geladen und wo sie verwaltet werden können.

5 Kurzüberblick *webMethods 7.1* Werkzeuge

In nachstehender Tabelle stehen die Werkzeuge die von der *webMethods BPMS Suite* zur Verfügung gestellt werden. Sie soll nur einen kurzen Überblick darstellen, detaillierter Informationen sind in den nachfolgenden Kapiteln erläutert.

Werkzeug	Aufgabe
Designer 7.1	Modellierung, Simulieren und Testen von Prozessen.
Developer 7.1	Entwicklung von Services die in den Prozessschritten des Designers verwendet werden können.
Blaze Advisor 6.5	Business Rules Management System zur Entwicklung von Regeln.

Tabelle 4: Werkzeuge webMethods 7.1

6 Handhabung webMethods 7.1

6.1 Der Designer 7.1

Der *Designer* wird benutzt um Geschäftsprozesse zu modellieren. Er unterstützt verschiedene Rollen, dies äussert sich in den Benutzeroberflächen. Beispielsweise stehen einem *Business Analysten* andere Optionen zur Verfügung als einem Entwickler.

Der Benutzer kann hier die Geschäftsprozesse modellieren. Die Bedienung ist sehr intuitiv und wer schon mit *Eclipse* gearbeitet hat, findet sich relativ schnell zurecht.

Die wichtigsten Eigenschaften während des Modellierungsvorgangs sind wohl die Eigenschaften Perspektiven der einzelnen Schritte und der angebotene *Debugging* Modus.

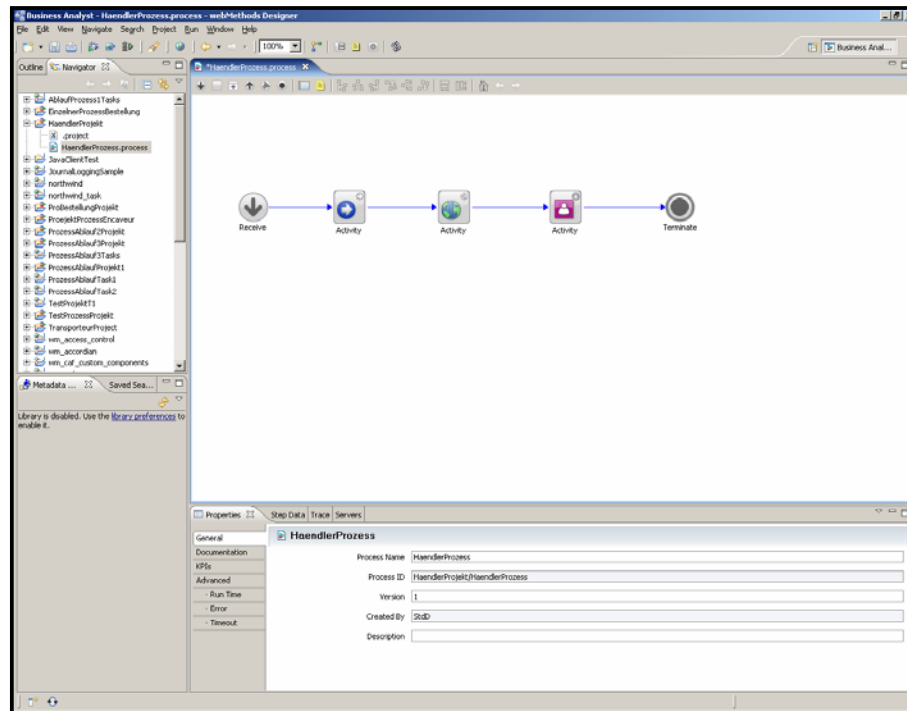


Abbildung 12 : Designer Benutzeroberfläche.

Quelle :Schaller Josef

Detaillierte Einblicke in die Funktionen des *Designers* finden sich in den nächsten Kapiteln wieder, in denen es sich um die Prozessmodellierung dreht.

6.2 Modellierung¹³

Die Modellierung der Geschäftsprozesse geschieht mit Hilfe des *Designers* 7.1, er wird standardmässig bei der Installation der BPMS Palette mit installiert.

Die grafische Oberfläche ist die von *Eclipse*. Beim *Designer* kann man zwischen verschiedenen Oberflächen, welche entsprechend der Benutzer konzipiert sind, auswählen. Beispielsweise die Benutzeroberfläche mit sämtlichen Optionen für den *Business Analyst*, eine für den Entwickler und eine Art *My webMethods* Ansicht.

Die Bedienung ist allgemein sehr intuitiv gehalten und es kann mittels *Drag and Drop* die gewünschten Prozesse auf die Arbeitsfläche gebracht werden.

Es können diverse Aktivitäten modelliert werden wie, Regeln, *Tasks* für eine menschliche Interaktion, *Web Services* zum Abrufen von externer und interner Funktionen und Informationen, *Integration Services* die schon standardmässig vorgegeben sind oder individuell im *Developer* entwickelt werden können.

Wichtig bei der Modellierung sind Dokumente und Variablen, die von einem Prozessschritt zum anderen mitgegeben werden können oder teilweise auch müssen. Verlangt ein Service einen gewissen *Input* muss dieser vom vorherigen Schritt mitgegeben werden, damit die Informationen zur Verfügung stehen. Auch der *Output*, welcher der Service erstellt ist möglicherweise für die nächste Aktivität erforderlich.

Dokumente können im *Developer*, einzelne Variablen im *Designer*, erstellt werden. Zu berücksichtigen sind die Typen von Werten welche von einem Prozessschritt zum nächsten weitergegeben werden. Meist handelt es sich um *Strings*. Es können aber auch Objekte mitgegeben werden, Objekte wählt man dann, falls es sich um andere Typen als um *Strings* handelt.

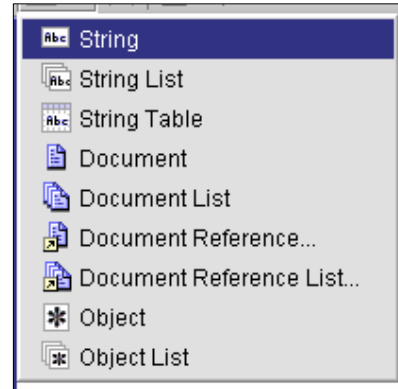


Abbildung 13 : Datentypen.
Quelle: Schaller Josef

Im *Developer* können diese Objekte zu den Typen *gewrapped* werden die man benötigt, beispielsweise *Integer*. Allgemein stehen die Typen zur Verfügung die man beim Programmieren unter Java auch kennt.

Zwischen dem Anfangsprozessschritt (*Receive Step*) und dem Abschluss eines Prozesses (*Terminate Step*) sind quasi alle Aktivitäten möglich. Wie oben schon erwähnt können diverse Aktivitäten definiert werden. Eine der wohl

¹³ Quelle: Designer. Hilfesiten des Designers 7.1

grössten Möglichkeiten Dienste einzufügen sind *IS Services*, welche mit Hilfe des *Developers* erstellt werden.

Besteht vielleicht ein Bedarf gewisse Programme oder Skripte in einer Aktivität auszuführen, müssen diese mittels *Services* im *Developer* entwickelt werden. Beispielsweise kann man ein Skript über einen individuellen Java Service aufrufen und dadurch auch im Prozessschritt benutzen.

Ein wichtiger Aspekt ist die menschliche Aktivität bei Prozessschritten, die so genannten *Task* Aktivitäten. *Tasks* sind Schritte im Prozessablauf die eine menschliche Interaktion voraussetzen und bestehen aus so genannten *Portlets* mit den *Views*, wobei die *Portlets* die Funktionalität und die *Views* die Ansichten definieren. Es besteht eine breite Palette an Formularen und Feldern die für das Design verwendet werden können.

Da die *Tasks* nicht aus *Services* auf dem *Integration Server* bestehen, ist es standardmässig nur mit dem *Designer* möglich solche Formulare für den Benutzer zu erstellen. Wobei anzumerken ist, dass die Funktions- und Designmöglichkeiten sehr vielfältig sind.

Für jeden Prozessschritt ist eine Vielzahl an einstellbarere Möglichkeiten vorgesehen. Es können nicht nur die Arten von Aktivitäten ausgewählt werden wie *Tasks*, *IS Services* etc. sondern auch die Bestimmung von *Key Performance Indicators*, welche das Erreichen bestimmter Grenzen oder *Performance* signalisieren. Sie sind wichtig für das Überwachen der Prozesse in *My webMethods*. Alarme und Warnungen werden bei Erreichen einer bestimmten Toleranzlimite bei einem Prozess ausgelöst.

Was bei Erreichen von solchen Toleranzlimiten passiert kann definiert werden, Email an den Verantwortlichen etc.

Verwendet man *IS Services* oder *Web Services* kann dadurch vielfach der Prozess automatisiert werden, beispielsweise eine Bestellung über einen *Web Service* kann direkt Informationen in die Datenbank des Produzenten speichern und diesen Prozessschritt abschliessen ohne dass eine Interaktion seitens des Benutzer erforderlich ist.

Es ist hier anzumerken, dass erheblich viele *Services* standardmässig bei der Installation des *Integration Servers* enthalten sind und nur mehr in so

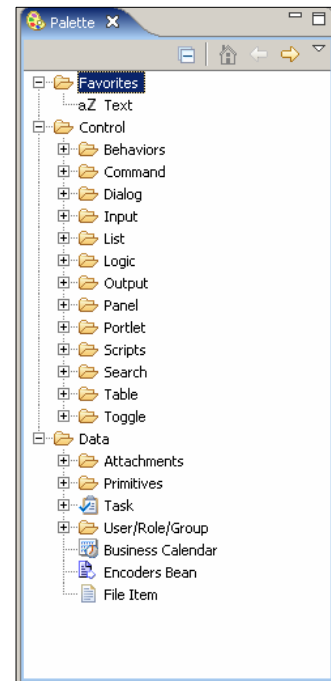


Abbildung 14 : Kontrollpalette für Tasks im Designer 7.1.

Quelle: Schaller Josef

genannten *Flow Services* zusammengesetzt werden müssen (*Developer*), danach können diese ohne weiteres im *Designer* benutzt werden.

Eine Auswahl an Möglichkeiten für *Services* und Anbindung an Datenbanken ist im Abschnitt *Developer* enthalten.

Weitere Einstellungen ermöglichen die Konfiguration von Konditionen wie *if*, *else* und *timeouts* die zwischen die einzelnen Prozessschritte geschaltet werden können. Denkbar in diesem Zusammenhang ist die Modellierung von Unterprozessen aufgrund der vorangehenden Konditionen.

Diese Konditionen sollten vorgängig bekannt sein, um Auswirkungen im bei der Entwicklung im *Developer* zu kennen.

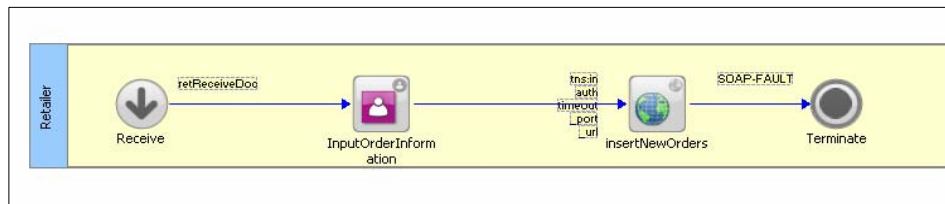


Abbildung 15 : Einfacher Prozess im Designer.

Quelle: Schaller Josef

Bezüglich zur zeitlichen Ausführung der Prozesse können diese mit einem *Business Calendar* konfiguriert werden.

Natürlich gehört zum Modellieren von Geschäftsprozessen die Ausführung zweier Schritte zur gleichen Zeit, wichtig hierbei sind die richtige Konfiguration der Bedingungen und die *Outputs* der Aktivität für die nächsten Schritte.

Individuell zu jedem Schritt kann man die maximale Anzahl der durchzuführenden Schritte einstellen, was die Prozessschritte anbelangt. Auch hier ist es wiederum möglich in gewissen Ereignissen im Prozessschritten, welcher aus einem *Flow Service* besteht, Schleifen und Wiederholungen zu erstellen.

Wie man bisher unschwer erkennen konnte sind die Prozessschritte sehr abhängig von den benutzten Diensten auf dem *Integration Server*. Um die Prozessmodellierung im *Designer* übersichtlicher zu gestalten kann es sinnvoll sein, einzelne Schritte die man im *Designer* machen möchte, durch einen *IS Service* zu ersetzen.

Umgekehrt können aber auch die Schritte ziemlich weit aufgespaltet werden, dies hängt sehr von den individuellen Prozessen und dem Abläufen innerhalb der Unternehmung ab.

Auch im Zusammenhang mit Schleifen die man modellieren möchte, diese können schon im *Developer* gemacht werden, ansonsten ist es durchaus möglich Rückschritte im Prozess selber zu konfigurieren. In jedem Fall ist es

unerlässlich die *In-* und *Outputs*, sowie die Konditionen dieser Schritte sorgfältig zu wählen.

Erwähnenswert ist die Benutzung gleicher Variablen oder Felder in zwei parallel ablaufenden Prozessschritten, dies ist hinsichtlich des nötigen *Inputs* für den nächsten Prozessschritt wichtig

Weitere Möglichkeiten beim Design der Geschäftsprozesse sind Unterprozesse (*Subprocesses*), sie sind fast analog zu den anderen Prozessschritten mit dem Unterschied, dass am Ende des Subprozess nur ein einziger Weg herausführt und mit einem *And join* versehen werden muss. Innerhalb des Unterprozesses können wiederum verschiedenen Prozessschritte stehen.

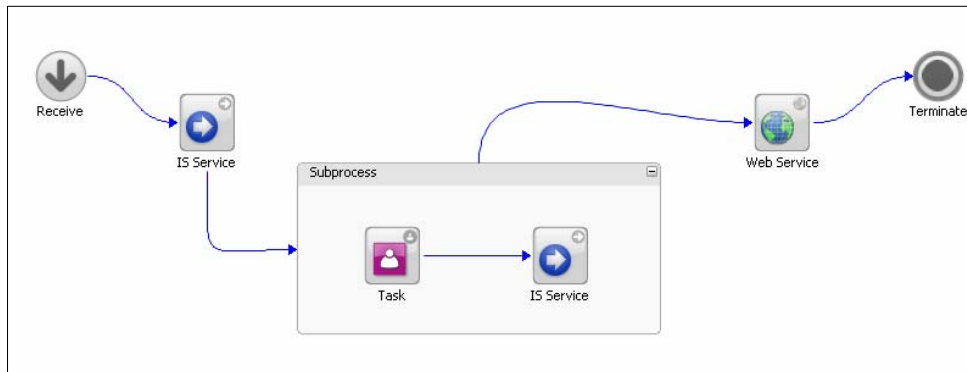


Abbildung 16 : Prozess mit Unterprozess.

Quelle: Schaller Josef

Ein weiterer Unterschied besteht darin, dass der Unterprozess selber keinen *In-* und *Output* haben kann, dies ist in den Prozessschritten innerhalb des Unterprozesses zu konfigurieren.

Die im *Designer* erstellten Prozesse werden auf den *My webMethods* Server *deployed*.

Die Wiederverwendbarkeit der modellierten Prozesse ist einerseits durch herkömmliches Kopieren der erstellten Projektlösungen möglich, andererseits ist darauf zu achten wie und für wen diese Prozesse modelliert wurden, die Wiederverwendbarkeit ist in diesem Sinne auch durch die Einstellungen von Gruppen und Benutzern zu beachten.

Die Services der Prozesse sind in der Paketverwaltung auf dem *Integration Server* gelistet und können auch wieder verwendet werden, natürlich aufgrund der individuellen Rechtevergaben an Benutzer und Gruppen.

Eine andere Möglichkeit der Wiederverwendbarkeit bietet den Import von Prozessen, wie man sich das von *Eclipse* gewöhnt ist. Es können Prozesse von *webMethods Designer* ab Version 6.x und *BPEL (Business Process Execution Language)* importiert werden.

Designer unterstützt den Import und Export von *BPEL 2.0* um *Prozess templates* zu konstruieren. Nach einem allfälligen Import oder Export sind noch einige Prozessdefinitionen zu verfeinern.

Prozesse des *Business Analysten* können nur zu Analysezwecken auf den *My webMethods Server* geladen werden, diejenigen des *Process Developers* können kompiliert und ausgeführt werden. Die Analyse auf dem *My webMethods Server* erlaubt eine Fehlersuche und mögliche Verbesserungen der Prozesse.

Sobald der Prozess modelliert ist, kann er simuliert und getestet werden. Damit ein *Debugging* erfolgen kann, muss der Prozess erstellt und auf den Server geladen werden. Hier muss beachtet werden, dass in der Rolle des *Business Analysten* nur ein Hochladen zu Analysezwecken möglich ist, das *Debugging* ist dem Entwickler vorbehalten.

Anmerkung: Die Rollenansichten innerhalb des Designers können per Mausklick gewechselt werden. Es sind keine Benutzerrechte damit verknüpft.

Beim *Debugging* wird der Prozess Schritt für Schritt durchgearbeitet und verfolgt, natürlich mit den notwendigen Informationen.

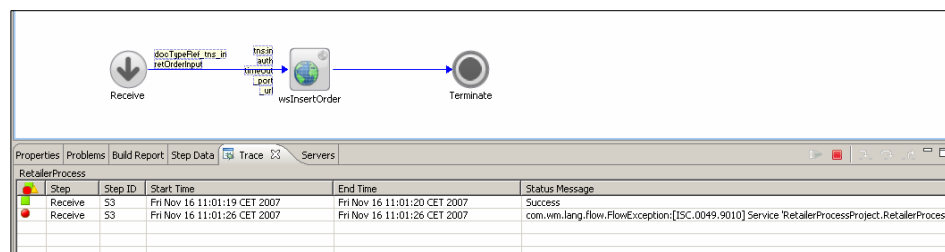


Abbildung 17 : Debugging im Designer.

Quelle: Schaller Josef

Sämtliche Prozesse sind wie die *Services* in Paketen organisiert. WebMethods bietet keine direkte Versionskontrolle von Prozessen. Es besteht aber die Möglichkeit die Pakete beispielsweise mit *CVS (Concurrent Versions System)* zu versionieren.

Beim der Modellierung im *Designer* benötigt es immer einen Start Schritt (*Receive Step*) damit ein Prozess starten kann. Dieser Schritt muss aber nicht manuell gestartet werden. Externe Ereignisse können diesen Prozess auslösen. So ist es möglich eine Datenbanktabelle zu überwachen und bei gegebenen Veränderungen ein Dokument zu publizieren, welches einen gewünschten Prozess auslöst. *Adapter Notifications* weisen diese Fähigkeit auf.

6.3 Der Developer 7.1

In Abbildung 18 ist die Benutzeroberfläche des *Developers* 7.1 wiedergegeben. Mit dem *Developer* werden *Services* für die Prozessschritte erstellt und konfiguriert. Diese sind in Paketen auf dem *Integration Server* gespeichert und können im Modell des *Designers* den Prozessen zugewiesen werden.

Anmerkung: Die Vielfältigkeit der Möglichkeiten im Developer werden hier nicht alle wiedergegeben, stattdessen wird auf die anderen Kapitel, in denen die Entwicklung der Prozesse geschieht, verwiesen. Weiterführende Information erhält man in den technischen Kapiteln im Anhang.

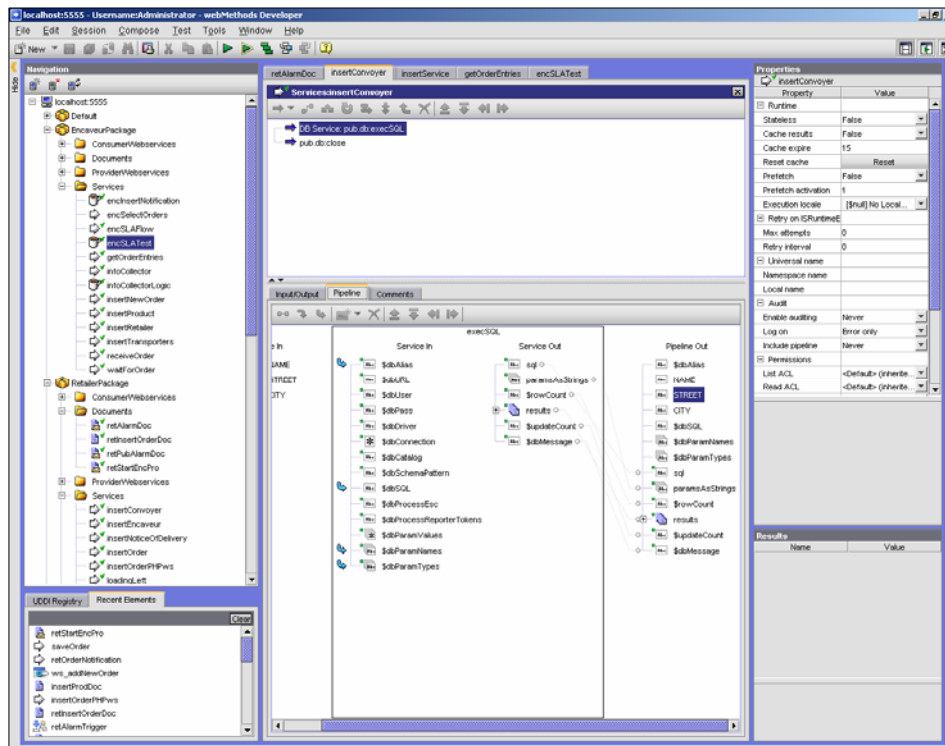


Abbildung 18 : Oberfläche Developer 7.1.

Quelle: Schaller Josef

6.4 Monitoring¹⁴

webMethods bietet Komponenten mit denen man Geschäftsprozesse überwachen kann. Siehe hierfür auch das Kapitel über die Architektur von *webMethods*.

Das *Monitoring* ist über die *My webMethods* Benutzerschnittstelle geregelt und zugänglich.

Damit ein Benutzer die Prozesse und *Tasks* überwachen kann benötigt er ein Benutzerkonto. Das Benutzerkonto wird von einem Administrator angelegt und einer Gruppe und Rollen zugewiesen.

Die Benutzer und Rollenverwaltung in *webMethods* sind sehr vielfältig. Es kann jedem Benutzer die gewünschten Rechte zugeordnet werden. Welche Prozesse können welche Benutzer wie überwachen und welchen Einfluss darauf nehmen. Somit kann eingestellt werden, dass jeder Benutzer nur Zugriff auf Prozesse hat die für ihn vorgesehen sind.

Informationen über Prozesse und *Tasks* können als CSV Dateien exportiert werden.

Ein praktisches Beispiel des *Monitoring* ist im Beispielprozess *tprocess3* in Kapitel 8 zu finden.

6.5 Integration¹⁵

Eine Integration in eine bestehende IT-Infrastruktur kann auf mehrere Arten geschehen.

Es gibt viel Adapter mit denen ein Informationsaustausch mit *webMethods* gemacht werden kann. In Abbildung 19 kann man anhand der Adapterdokumentation die verschiedenen Adaptermöglichkeiten erkennen.

¹⁴ Quelle: *webMethods*. *webMethods Monitor Users' Guide*. *webMethods_Monitor_User's_Guide_7_1-4.pdf* [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 30.11.2007]

¹⁵ Quelle: *webMethods*. *Gear 6.5 Standards Protocols Best Practices*. *.webMethods_Monitor_User's_Guide_7_1-4.pdf* [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 20.11.2007]

Adapters Documentation
Adapter Development Kit
Ariba Supplier OnRamp
AS/400
BroadVision
Character Replacer
Clarify
CommerceOne MarketSite OnRamp
Enterprise JavaBeans
J.D. Edwards WorldSoftware
JDBC
JMS
Lotus Notes
MSMQ
Multibyte File Parser
Oracle Applications
PeopleSoft
Remedy
SAP
Siebel
Tuxedo
WebSphere MQ
XSLT
Zengin TCP/IP Adapter
Adapters in End of Support Stages

Nicht nur mit Adaptern werden Verbindungen zu bestehender Softwarekomponenten gemacht, sondern auch die Generierung von Code zur Einbindung von Applikationen ins *BPMS*, somit sind die *Services* die in webMethods entwickelt werden auch für *Clients* benutzbar. Siehe auch Kapitel *Code Generierung* im Anhang.

Im Dokument *Gear 6.5 Standards and Protocols Best Practices* sind sämtliche Standards auf denen webMethods basiert und welches es unterstützt ausführlich beschrieben. Standards die im erwähnten Dokument beschrieben sind, sind in Tabelle 5 enthalten.

Abbildung 19 : Adapter anhand der Dokumentation.

Quelle:

<http://advantage.webmethods.com>

Transport Layer Protocols and Standards

XML Standards

Web Service Standards

Security Standards

B2B Standards

Vertical Standards

Application and Programming Standards

Vendor-Based Standards

Tabelle 5: Standards in Gear 6.5 Standards and Protocols Best Practices.

Quelle: webMethods. Gear 6.5 Standards Protocols Best Practices.

7 Verwaltung webMethods 7.1

Einführung

In dem nun folgenden Kapitel werden Informationen über Systemvoraussetzungen, *Backups* und weitere verwaltungstechnische Informationen gegeben. Die Informationen sind nicht abschliessend und sollen einen generellen Einblick über die Verwaltung von *webMethods* geben.

7.1 Systemvoraussetzungen¹⁶

Product	Hard Drive Space (MB)	RAM (MB)	Virtual/ Swap (MB)	CPUs
Infrastructure (shared code, such as JDKs)	200			
Blaze Advisor	200	1024 (2048)		1
webMethods Broker	750 (1250)*	512 (2048)		1
Database Component Configurator	60			
Deployer	20	Nothing beyond the host Integration Server		
Designer	700	1024 (2048)		1
Developer	100 (200)	128 (256)		1
Documentation				
Adapters and eStandards Modules	60			
Core Product Guides	160			
readmes	1			

Tabelle 6: Systemvoraussetzungen für webMethods Komponenten.

Quelle: Integration Server Administrator's Guide

¹⁶ Quelle: webMethods. Integration Server Administrator's Guide. webMethods_Integration_Server_Administrator's_Guide_7_1.pdf. Seite 20 und 21.[Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 20.11.2007]

Product	Hard Drive Space (MB)	RAM (MB)	Virtual/ Swap (MB)	CPUs
Integration Server	200 (350)	256 (512)		1
Metadata Library	100 (500)**	512 (2048)	1024	1
Monitor	Nothing beyond the host Integration Server.			
My webMethods Server	300	1024 (2048)		1
Optimize				
Analytic Engine	500	1024 (2048)	1024	1
Prediction Engine	500	1024 (2048)	1024	1
Infrastructure Data Collector	300	2048 (4096)	1024	1
Web Service Data Collector	100**	128 (256)	128	1
Process Engine	Nothing beyond the host Integration Server.			
Task Engine	Nothing beyond the host My webMethods Server.			
Trading Networks				
Console	25	128 (256)		
Server	50			1

Tabelle 7: Fortsetzung Tabelle 6.
Quelle: Integration Server Administrator's Guide

7.2 Datenbanken¹⁷

Als Datenbanksysteme kommen mehrere Typen in Frage. Eine Migration auf eine andere unterstützte Datenbank ist mit Hilfe des *Database Component Configurators*

ohne weiteres möglich.

Database Component and Version	Oracle		SQL Server		DB2	
	9.2.0.6, 9.2.0.6 RAC	10g, 10g RAC	2000 SP3	2006	8.2 LUW	iSeries V5R4
Analysis 30	•	•	•	•	•	
Process Tracker 30						
IS Core Audit Log 11, when using Monitor						
Process Audit Log 30						
Process Engine 30	•	•	•	•	•	
Archive 20						
Reporting 10						
Staging 10						
Cross Reference 10						
Document History 10	•	•	•	•	•	•
IS Internal 20*						
IS Core Audit Log 11, when not using Monitor						
Trading Networks 20	•	•	•	•	•	•
Trading Networks Archive 20						
Metadata Reasoner 30	•	•	•	•	•	
Metadata Repository 30						
My webMethods Server 20	•	•	•	•	•	

Tabelle 8: Unterstützte Datenbanksysteme.
Quelle: webMethods System Requirements

17 Quelle: webMethods. webMethods System Requirements. webMethods_System_Requirements_7_1.pdf. Seite 2. [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 30.11.2007]

Siehe auch *Datenrettung und Backups* im nächsten Kapitel. *webMethods* unterstützt eine Vielzahl an Betriebssystemen. Eine Migration auf andere Systeme ist auch hier möglich.

7.3 Datenrettung und Backups¹⁸

Sämtliche Informationen die *webMethods* benötigt sind in der Datenbank gespeichert. Ausgenommen die Modelle im *Designer*, diese sind im *Eclipse Workspace* gespeichert.

Damit bei einem Totalausfall/-verlust die Daten wieder geholt werden können, ist es notwendig regelmässig ein *Backup* der benutzten Datenbank zu machen.

Mit Hilfe des Integrationsservers ist es möglich gewisse Pakete zu archivieren. Dateien die für ein bestimmtes Paket nötig sind werden als Archive im *webMethods* Verzeichnis gespeichert.

In Abbildung 20 sind die Archivierungsmöglichkeiten für Pakete aufgezeigt. Die Archivierung ist über das Paketmanagement in der Integrationsserver Administration zu erledigen. Die Wiederherstellung geschieht dadurch, dass entsprechende Pakete wieder auf den Integrationsserver geladen werden.

Specify Files for the Archive

Files available in EncaveurPackage:

- code/
- code/classes/
- code/classes/Services.class
- code/jars/
- code/source/
- code/source/Services.java
- config/
- doc/
- lib/
- manifest.v3
- ns/
- ns/ConsumerWebservices/

Files to Include:

- ☒ All files
- ☐ Selected files
- ☐ All except selected files
- ☐ Files specified by filter:
- ☐ All except files specified by filter:
- (ex: *.java, *.class)

Files to Delete from Target Package (For distributing upgrades only): One file name per line. Use ";" to separate multiple file names.

Package Archive Information

Archive Type	<input checked="" type="radio"/> Full (Including all files is recommended)	
	<input type="radio"/> Patch (For distributing upgrades)	
Archive Name	EncaveurPackage	(.zip will be appended)
Brief Description		
Version	1.0	(current version: 1.0)
Build Number		(ex: 6)
Previous Patches Included		(ex: 4, 5)

Recommendations for Subscriber Install

webMethods Integration Server	7.1.0.0	(ex: 3.1)
Minimum Version of JVM	1.5.0_12	(ex: 1.2)

Required Settings for Creating a Patch

Version of Target Package	1.0	(ex: 1.0)
---------------------------	-----	-----------

Create Archive

Abbildung 20 : Archivierungsmöglichkeit auf dem Integrationsserver.

Quelle: Schaller Josef

Informationen über Prozesse werden geloggt. Es gibt eine mehrere Varianten wie und welche Daten geloggt werden. Standardmässig werden diese Logdateien in

¹⁸ Quelle: webMethods. webMethods Integration Server Administrator's Guide.

webMethods_Integration_Server_Administrator's_Guide_7_1.pdf. [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 30.11.2007]

Flat Files gespeichert und nicht in die Datenbank, von dieser Variante ist aber abzuraten, es ist besser die Logdateien in die Datenbank zu speichern. Siehe auch Kapitel *Architektur Optimize for Infrastructure*.

In Tabelle 9 sind Unterverzeichnisse und wichtigen Dateien die für ein *Recovery* nötig sind aufgelistet. Es wird empfohlen diese Dateien regelmässig zu sichern.

Das Backup dieser Dateien sollte nicht im laufenden Betrieb gemacht werden. Der Server sollte keine Aktivität aufweisen oder muss heruntergefahren sein. Sind mehrere IS und Broker in der IT-Infrastruktur sind die Backups auch für die Broker zu machen.

Pfad	Wichtige Dateien
./audit/data	AuditStore.data0000000 AuditStore.log0000000
./DocumentStore	ISResubmitStoredata0000000 ISResubmitStorelog0000000 ISTransStoredata0000000 ISTransStorelog0000000 TriggerStoredata0000000 TriggerStorelog0000000
./WmRepository4	FSDdata0000000 FSDlog0000000

Tabelle 9: Pfad der wichtigen Dateien für regelmässige Backups.

Quelle: webMethods. Integration Server Administrator's Guide.

7.4 Aktualisierung und Support

Aktualisierungen der Softwareversion sind nur über die logingesicherte Webseite <http://advantage.webmethods.com> von webMethods zu erreichen. Loginname und Passwort erhält man als Käufer der *webMethods BPMS Suite*.

Fixes und Aktualisierungen können dort herunter geladen werden, dies aber nur wenn man als technischer Kontakt eingetragen ist, d.h. jede Unternehmung die *webMethods* benutzt, muss einen Benutzer als technischen Kontakt eintragen lassen um *Fixes* herunterladen zu können.

Wie die Aktualisierungen und *Fixes* installiert und welche Auswirkungen sie haben, sind in den einzelnen *Readmes* sehr gut beschrieben. Je nach Art der Aktualisierung sind einzelne Komponenten von *webMethods* herunterzufahren.

Zusätzlich können Supportanfragen an *webMethods* gesendet werden.

Ein Forum und eine Vielzahl von Dokumenten über die Installation, Administration und Handhabung von *webMethods BPMS* sind auch über <http://advantage.webmethods.com> verfügbar.

Neben dieser Möglichkeit erreicht man über den Link <http://www.wmusers.com> ein englischsprachiges Forum welches einzig Fragen und Diskussionen über *webMethods* beinhaltet. Es ist sehr gut strukturiert und viele kompetente Benutzer helfen gerne weiter.

Wie bei allen grossen Softwareanbietern gibt es zahlreiche Kurse für *webMethods*. Entweder in der Unternehmung die eine Integration von *webMethods* machen oder zu Schulungszwecken in Kurszentren. Es gibt auch eine *E-Learning* Plattform. Allerdings sind alle Kurse kostenpflichtig.

7.5 Benutzerverwaltung¹⁹

Benutzer, Gruppen und Rollen sind über die Administrationsoberfläche für den Integrationsserver und in *My webMethods* zu konfigurieren. Es besteht aber faktisch eine Trennung zwischen einem Benutzer des Integrationsservers und dem des *My webMethods Servers*.

Es können aus externen Quellen Benutzer importiert werden. Externe Quellen können sein: Ein *Central User Management* oder *Lightweigh Directory Access Protocol (LDAP)*.

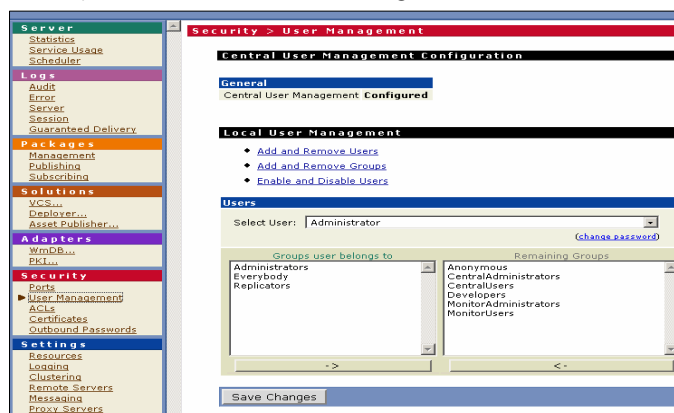


Abbildung 21 : Benutzerverwaltung Integrationsserver.

Quelle: Schaller Josef

Sind für bestimmte Servicezugriffe auf dem *IS* bestimmte Rechte nötig müssen diese mit Hilfe der *Access Control List* konfiguriert werden, so dass ein Benutzer der in *My webMethods* erstellt wurde auch Zugriff auf die Ressourcen des *IS* hat. Detaillierte Beschreibung sind in den Administrator Handbüchern der beiden

¹⁹ Quelle: webMethods. webMethods Integration Server Administrator's Guide.

webMethods_Integration_Server_Administrator's_Guide_7_1.pdf. [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 30.11.2007]

Server zu finden. In der Abbildung 21 ist die Benutzerverwaltung des Integrationsservers erkennbar

7.6 Prozessmanagement²⁰

Folgende Einflüsse können auf die Prozesse genommen werden.

Einfluss
Löschen von unbenutzten Prozessen
Deaktivieren von Prozessmodellen
Aktivieren von Prozessmodellen
Unterbrechung von Prozessen
Wiederaufnahme eines Prozesses beim gewünschten Schritt
Unterbrochener Prozess wieder laufen lassen

Tabelle 10: Prozesseinflüsse.

Quelle: webMethods. Getting Started with Business Process Management.

Die Einflussnahme auf die Prozesse, wie auch auf *Tasks*, kann aufgrund der zugewiesenen Rechte variieren.

Damit ein Prozess verwaltet werden kann ist eine Suchfunktion für Prozesse in My webMethods verfügbar. In Abbildung 23 ist die Suche nach Prozessinstanzen ersichtlich.

Abbildung 22 : Suchmöglichkeit nach Prozessen in My webMethods.

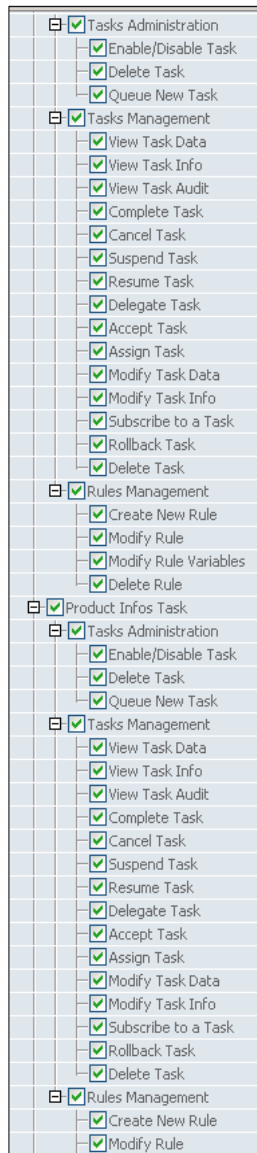
Quelle: Schaller Josef

Im Allgemeinen kann die Suche nach beliebigen Objekten gemacht werden, nicht nur nach Prozessen und Benutzern.

²⁰ Quelle: webMethods.Getting Started with Business Management 7.1.

Getting_Started_with_Business_Process_Management_7_1.pdf. [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 30.11.2007]

7.7 Task Ausführung in My webmethods



Welche Benutzer welche *Tasks* wie ausführen können kann der Administrator konfigurieren. Dies geschieht über die Rollenverwaltung. Diese kann hinsichtlich der Datenlevelsicherheit, Zugriffsprivilegien, Funktionsprivilegien, dynamischer Attribute und Mitglieder editiert und individuell konfiguriert werden. Es ist möglich die Prozesse den Rollen zuzuordnen, so dass nur die Mitglieder in dieser Rolle die Prozesse ausführen können.

Regeln für *Tasks* definieren die Umstände der Ausführung.

Log-Einträge für *Tasks* steuern zur Verfolgung der nachfolgenden Punkte bei:

- Wann ein *Task* eingereicht wird (*queued*)
- Wenn der Benutzer einen *Task* akzeptiert, startet, unterbricht, wieder aufnimmt, beendet oder abbricht
- Ob der *Task* erfolgreich beendet, abgebrochen oder eine Zeitüberschreitung erfahren hat

Siehe auch Abschnitt *Task Erstellung – Eigenständige* und *Task Erstellung – Im Prozess* in Kapitel *Technische Aspekte* im Anhang.

Abbildung 23 : Task Verwaltung für einzelne Benutzer.
Quelle: Schaller Josef

8 Beispielprozess *tprocess3*

8.1 Einleitung

In diesem Kapitel wird anhand eines einfachen Geschäftsprozesses die nötigen Schritte vollzogen, wie die Werkzeuge von *webMethods 7.1* benutzt werden können und wie Prozesse entwickelt, *deployed* und überwacht werden können.

8.2 Zusammenfassung

Es handelt sich um einen Geschäftsprozess eines Händlers. Der Händler hat die Möglichkeit eine Bestellung an den Encaveur zu schicken. Der Encaveur bietet hierfür seinen *Web Service* an.

Der Web Service speichert die Bestellung in die Datenbank des Encaveurs.

Für den *Web Service* sind die Identität des Händlers (*Retailers*), Identität des gewünschten Produkts und die Menge als Input mitzugeben. Weiter wird im Prozess die ausgeführte Bestellung in die Datenbank des Händlers gespeichert.

Sobald diese Schritte fertig sind, wird der Encaveur mittels einer Emailnachricht informiert, dass er eine neue Bestellung erhalten hat.

Danach ist der Prozess beendet.

8.3 Das Modell



Abbildung 24 : Bestellprozess des Händlers.

Quelle: Schaller Josef

Receive Step

Der *Receive Step* wartet auf ein Dokument. In diesem Beispiel ist dies eine Nachricht aus dem Lager, dass der Lagerbestand so weit gesunken ist, dass eine neue Bestellung erforderlich ist. Sobald dieses Dokument publiziert wurde, startet der Prozess.

hlInput

Human interaction input. Es handelt sich um einen *Task*, der eine Eingabe vom Benutzer erfordert. Im *Task* kann der Benutzer die benötigten Werte für die Bestellung eingeben, hier: *idproduct*, *idretailer* und *quantity*.

Sobald der *Task* komplettiert ist, werden die Daten an den *Web Service wsEncPHP* geschickt.

wsEncPHP

wsEncPHP ist der *Web Service* des Encaveurs. Der die Informationen der Bestellung über die *PHP* Applikationslogik in die *MySQL* Datenbank speichert. Ein Schema der Architektur ist in Abbildung 48, in der Beschreibung der Architektur des Encaveur's, dargestellt.

saveOrderDB

Damit der Händler weiss, welche Bestellung er an den Encaveur verschickt hat, speichert dieser Prozessschritt die gesendeten Daten zusätzlich in seine eigene *MS SQL* Datenbank.

Notification

Sobald die Bestellungen gespeichert wurden, wird dem Encaveur eine Email geschickt, dass er eine neue Bestellung erhalten hat. Danach ist der Bestellprozess abgeschlossen.

8.4 Die Entwicklung

Nachdem der Modellierungsprozess abgeschlossen ist, werden die nötigen *Services* entwickelt und Einstellungen getroffen, damit der Prozess später ausgeführt werden kann, dies geschieht mit Hilfe des *Developers 7.1* und teilweise noch mit dem *Designer 7.1*.

Anmerkung: Zur Wahrung der Übersicht werden in diesem Kapitel Verweise auf technische Kapitel im Anhang gegeben.

retAlarmDoc

retAlarmDoc ist ein simples Dokument, das im *Developer* entwickelt wurde. Es beinhaltet die Variable: *alert*. Sie löst später mit den erforderlichen Konditionen den Geschäftsprozess aus.

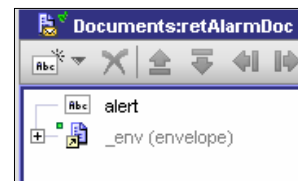


Abbildung 25 :
Dokumentvariable alert.
Quelle: Schaller Josef

Es wird ein neues Dokument mit der Variable *alert* (String) erstellt, danach in dessen Eigenschaften der Wert *publishable* auf *true* gesetzt.

Das im *Developer* erstellte Dokument wird nun im *Designer* dem *Receive Step* zugewiesen und die Kondition: *alert = low* im *Tab Transitions* gesetzt.

Der *Receive Step* ist nun der *Subscriber* dieses Dokuments. Sobald dies publiziert wird, wird der Prozess gestartet. Ist die Bedingung *low* erfüllt, wird der Task im nächsten Schritt aufgerufen.

Correlation Service	
Logical Server	Default
Step Retry Count	0
Step Retry Interval	60000
Generated Service Name	
Receive Protocol	Subscription (For Broker Documents)
Receive Document	Documents:retAlarmDoc

Abbildung 26 : Eigenschaften Receive Step.
Quelle: Schaller Josef

Field Name	Operator	Comparison Value/Field	AND/OR
Alert	=	low	

Abbildung 27 : Transitions Receive Step mit der Kondition für alert Variable
Quelle: Schaller Josef

hInput Task

Der *Task* kann mit dem *Wizard* erstellt werden. Für detailliertere Informationen

Name	Type
idproduct	String
idretailer	String
quantity	String

Abbildung 28 : Business Data für den Task.
Quelle: Schaller Josef

kann das Kapitel *Task Erstellung –Im Prozess* konsultiert werden.

Als *Output* des *Tasks* sind die benötigten Werte für den Web Service *wsEncPHP* konfiguriert. Ersichtlich in Abbildung 29.

wsEncPHP

Es handelt sich hier um den *Web Service* der für die *SOA* und das *BPMS* des Encaveurs entwickelt wurde. Im *Developer* wurde ein *Consumer Web Service* erstellt und hier in diesem Schritt als *Web Service* deklariert.

Siehe auch Kapitel: *BPMS* Encaveur und *Web Service* Erstellung im Anhang.

saveOrderDB

Der *saveOrderDB* Service speichert die Werte *idproduct* und *quantity* in die Datenbank des Händlers, der Wert *idretailer* ist hier nicht mehr erforderlich.

In der Abbildung 30 ist das *Mapping* der Werte in der *Pipeline* ersichtlich. In der *Pipeline* sind die Werte *idproduct*, *idretailer* und *quantity* (Werte aus dem *hInput Task*). Diese Werte werden durch den *Flow* Schritt *MAP* den nächsten benötigten Variablen zugeordnet.

- *idproduct* = *PRODUCTCODE*
- *quantity* = *AMOUNT*.

Der blaue Pfeil bei der Variable *ENCAVORID* symbolisiert, dass dieser Wert fix zugewiesen wurde. Er trägt den Wert 2007. Diese Daten werden dann im *Flow* Schritt DB Service: *pub.db.execSQL* in die lokale Datenbank geschrieben.

Der Schritt wurde auf gleiche Weise erstellt wie im Kapitel *Service Generation* im Anhang beschrieben. Danach wird mit *pub.db.close* die Verbindung zur Datenbank geschlossen.

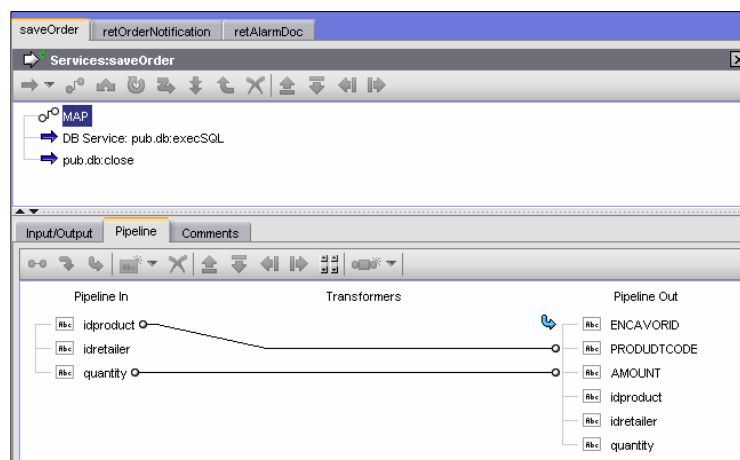


Abbildung 29 : Flow Service und Pipeline.
Quelle: Schaller Josef

Notification

Der Service im *Flow Schritt pub:client.smtp* ist einer der vielen *Built-in Services* von webMethods.

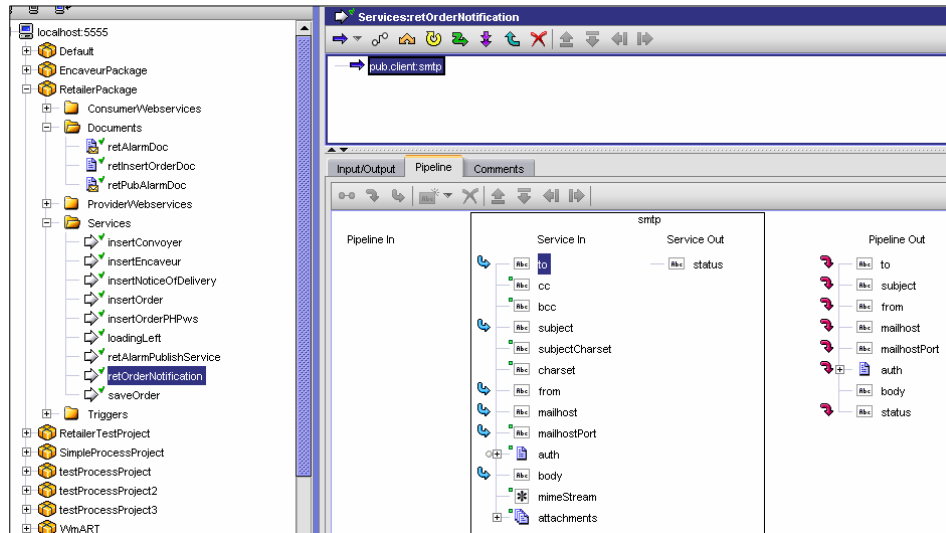


Abbildung 30 : Flow Step im retOrderNotification Service.

Quelle: Schaller Josef

Die deklarierten blauen Pfeile im *SMTP* Service sind einerseits Login Informationen für den Bluewin *SMTP* Server und andererseits die Email Werte wie Betreff etc.

8.5 Build and upload Beispielprozess tprocess3

Im *Designer* kann der Prozess auf den *My webMethods Server* geladen werden. Damit dies geschehen kann muss von der Ansicht des *Business Analysten* in die Ansicht des *Developers* gewechselt werden.

Ist der *Build* Prozess erfolgreich wird als nächstes in die Administrationsoberfläche des *My webMethods Servers* gewechselt und der Prozess aktiviert, so dass er gestartet werden kann.

Anmerkung: Hat man in der Rolle des Business Analysten im Designer den Prozess nur zu Analysezwecken hochgeladen erscheint dies in der Spalte Analysis enabled.

Die Aktivierung kann über: *Administration -> Business-> Business Processes* erreicht werden.

PROCESS NAME	MODEL VERSION	TYPE	EXECUTION ENABLED	ANALYSIS ENABLED	USED
DARetailerProcess	1	webMethods	✓	✓	Yes
RetailerProcess	1	webMethods	✓	○	Yes
RetailerTestProcess	1	webMethods	✓	○	Yes
SimpleProcess	1	webMethods	✓	✓	Yes
testProcess	1	webMethods	✓	○	Yes
tprocess1	1	webMethods	✓	○	Yes
tprocess2	1	webMethods	✓	○	Yes
tprocess3	1	webMethods	✓	✓	Yes

Abbildung 31 : Aktivierung von Prozessen für Ausführung und Analyse.

Quelle: Schaller Josef

8.6 Geschäftsprozess tprocess3 starten

Normalerweise würde der Prozess durch einen möglichen *Trigger* oder ein anders publizierendes Dokument gestartet. Zu Demonstrationszwecken geschieht dies in diesem Fall manuell im *Developer*. Wie im *Receive Step* des Geschäftsprozess (siehe Modell oben) wurde die Kondition der Variable *alert* genommen. Der Prozess wird nur ausgeführt fall dieser Wert *low* ist. Dieser kann hier mitgegeben werden (Abbildung 34). Der Wert *low* kann bedeuten, dass es im Lager zu Engpässen kommt, sprich zuwenig Artikel eines bestimmten Produktes vorhanden sind und die Notwendigkeit besteht eine neue Bestellung auszuführen.

Abbildung 32 : Ausführung des Dokuments im Developer 7.1.

Quelle: Schaller Josef

Integrationsserver vorhanden ist, wird das Dokument lokal publiziert. Hätte man zusätzliche IS und Broker würde man das Ziel des zu publizierenden Dokuments wählen, auch ein *Client* könnte das Ziel sein.

Danach wird das Resultat des Vorgangs angezeigt.

Abbildung 33 : Publikation auf dem lokalen IS.

Quelle: Schaller Josef

8.7 Monitoring Beispielprozess tprocess3

Der *Receive Step* des Prozess erhält nun das Dokument mit dem Wert *alert=low*. Dies führt dazu, dass der Prozess gestartet wurde. Nun besteht die Möglichkeit den Prozess in der *My webMethods* Benutzeroberfläche zu überwachen.

Im unteren Teil der Abbildung 29 ist ersichtlich, dass ein Prozess aktiv ist, es handelt sich um den gestarteten Prozess *tprocess3*, welcher mit Hilfe des *Developer* gestartet wurde. Wie im Model oben zu erkennen ist, ist als erste Aktivität des Prozesses eine Eingabe zu machen (*Task*). Aus dem Grund ist im oberen Teil der Abbildung 29 auch der Task 7679 *HIInput* erschienen. Bei der Prozessinstanz können nun die Details des Prozesses eingesehen werden.

Task List

TASK ID	TASK TYPE	PRIORITY	CREATED DATE
7470	Activity	None	26.11.2007 19:35
7471	Activity	None	26.11.2007 19:36
7678	H Input	None	30.11.2007 15:57
7679	H Input	None	02.12.2007 14:17

Process Instances

LAST UPDATED	START DATE / TIME	PROCESS NAME	VERSION	PROCESS INSTANCE ID	STATUS	DURATION	DETAIL
02.12.2007 14:16:59.265	02.12.2007 14:16:59.265	tprocess3	1	dcb6cd60a0d811dca0e98e3c9f4b03ed	Started	0d 00:13:18.516	

Abbildung 34 : Task Liste und Prozess Instanzen in My webMethods.
Quelle: Schaller Josef

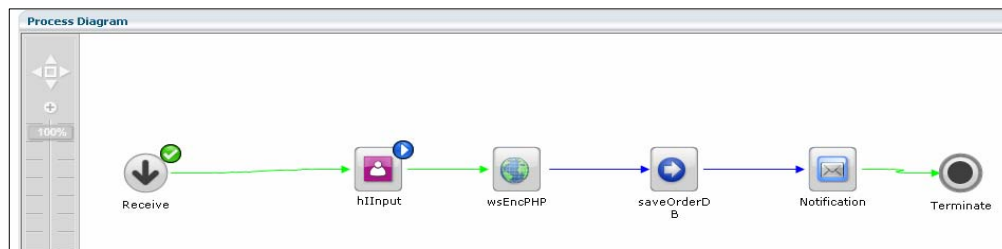


Abbildung 35 : Prozessdiagramm vor Komplettierung des Tasks hInput.
Quelle: Schaller Josef

In Abbildung 30 sieht man, dass der Prozess initialisiert wurde und nun eine menschliche Eingabe erfordert. Der Prozess wartet so lange, bis diese Eingaben gemacht und komplettiert wurden. Erst danach geht's weiter.

Der *Task* wird aufgerufen und die benötigten Informationen eingetragen. Hierfür klickt man in der *Task* Liste auf die gewünschte *Task ID*.

Abbildung 36 : Task Details. Eingabemaske für die Bestellung.

Quelle: Schaller Josef

Durch das Abschicken und Kompletieren des *Tasks* geht der Prozess weiter und ruft den *Web Service* des Encaveurs auf, danach erfolgt die Speicherung in die lokale Datenbank und schlussendlich wird eine Information an den Encaveur verschickt, wie beschrieben im Model des Beispielprozesses.

Nimmt man nun erneut Einsicht in die Details der Prozess Instanzen ist erkenntlich, dass der ganze Prozess abgearbeitet und abgeschlossen wurde.

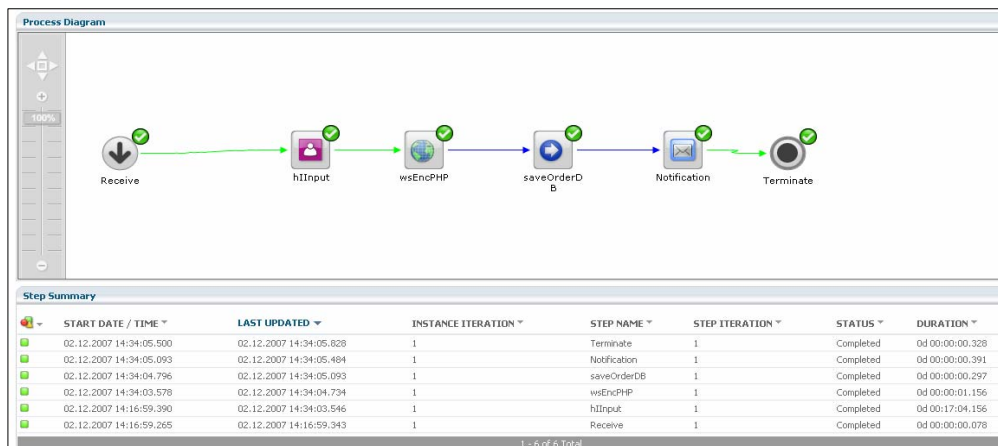


Abbildung 37 : Prozessdiagramm und ausgeführte Schritte.

Quelle: Schaller Josef

Um untersten Teil der Abbildung 32 sieht man die Zusammenfassung mit Informationen der ausgeführten Schritte.

Sind alle Schritte erfolgreich ausgeführt worden, hat der Encaveur eine neue Bestellung in der Datenbank und eine Meldung per Email erhalten. Auch der Händler hat in seiner lokalen Datenbank die Bestellung gespeichert. Damit ist der Geschäftsprozess abgeschlossen.

9 Szenario

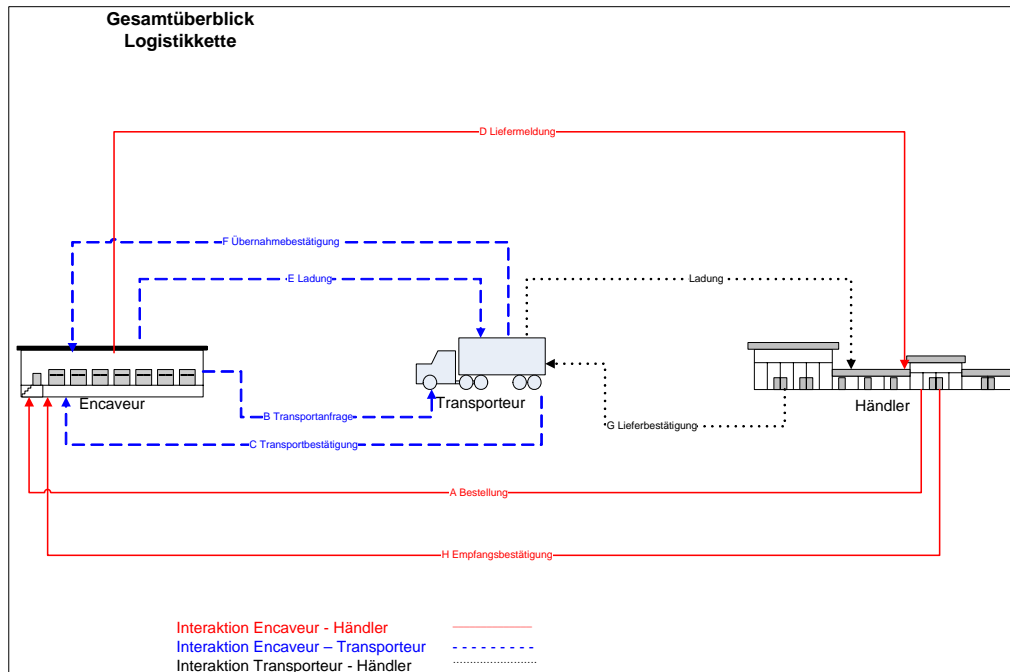


Abbildung 38: Logistikkette
Quelle : Schaller Josef

In der Abbildung 35 ist die Logistikkette dargestellt. Es handelt sich um drei Akteure: Encaveur, Transporteur und Händler. Alle drei interagieren miteinander auf verschiedene Weisen.

Der Händler braucht Produkte vom Encaveur, der Transporteur liefert diese an den Händler. Es finden Kommunikationen in denen relevante Informationen für die jeweiligen Partner fließen. Der Ablauf einer Bestellung verläuft in den meisten Fällen auf die gleiche Art und Weise.

Dieses Szenario bildet eine der Grundlage der vorliegenden Arbeit. Detaillierte Informationen über die stattfindenden Aktionen zwischen den Akteuren und welche Zielvorgaben gestellt sind, werden Pflichtenheft, dem Beschrieb der technischen Schnittstellen und Definition des Szenarios im Anhang genauer erläutert.

10 BPMS Händler

10.1 Einführung

Bemerkung

Damit das Kapitel übersichtlich bleibt und sich nicht in allzu technischen Erläuterungen verliert, wurde versucht die Implementierung in allgemeinen Schritten zu präsentieren. Es werden jedoch Verweise auf Kapitel gegeben, in denen die technische Umsetzung detaillierter erläutert wird. Dies trifft auch auf die Kapitel *Business Process Management System Encaveur* und *Business Process Management System Transporteur* zu.

Die Details der technischen Schnittstellen und Beschreibung des Händlers sind im Anhang enthalten. Zusammenfassend soll hier die Grundidee des Ablaufs des Geschäftsprozesses des Händlers wiedergegeben werden, damit die Prozessmodellierung nachvollzogen werden kann.

Es wird angenommen, dass der Händler über keine IT-Infrastruktur verfügt und sämtliche Implementierungen für den Händler im *BPMS* geschehen.

Anmerkung: Die Bezeichnung Händler wurde in den Bereichen der Implementierung nicht benutzt, anstelle der Bezeichnung Händler wurde der englische Begriff Retailer verwendet.

Überblick

Neben der detaillierten Beschreibung der Interaktionen in den technischen Schnittstellen und der Beschreibung des Szenarios sind im Allgemeinen folgende Punkte wissenswert:

- Der Händler kann über den Webservice *newOrder* des Encaveurs eine Bestellung aufgeben
- Der Transporteur schickt eine Meldung, dass die Ware unterwegs ist *loadingLeft*.
- Der Encaveur sendet über den Web Service des Händlers eine Lieferungsbestätigung *noticeOfDelivery*.

Der Händler bietet die Web Services *loadingLeft* für den Transporteur und *noticeOfDelivery* für den Encaveur an.

10.2 Architektur Händler

Wie schon erwähnt verfügt der Händler über noch keine bestehende Softwareapplikation. Aus dem Grund wird alles im BPMS von webMethods entwickelt. Als Datenspeicher dient eine *MS SQL 2005* Datenbank.

Sämtliche Services des Händlers liegen auf dem Integrationsserver. Auch die Logik ist in den Services integriert.

Bei diesem Akteur gilt es zu beachten, dass auch Services die nicht für nachfolgenden Prozess nötig sind, gemacht werden müssen. Das will heissen, dass beispielsweise das Einfügen von Transporteuren oder neuen Lieferanten möglich sein muss. Dies kann mit *standalone Tasks* gemacht werden. Alternativ kann auch ein Prozess für das Einfügen neuer Datensätze gemacht werden. Siehe auch Kapitel *Task Erstellung* im Anhang.

Das *BPMS* stellt gemäss den Beschreibungen in den technischen Schnittstellen diverse Web Services zur Verfügung, die die *SOA* ausmachen. Diese Web Services können von anderen *BPMS* oder *Clients* gemäss Vorgaben konsumiert werden.

Auch der Händler kann gebrauch von seiner *SOA* machen.

Angenommen er hat mehrere Prozesse, können diese die Services in der *SOA* benutzen.

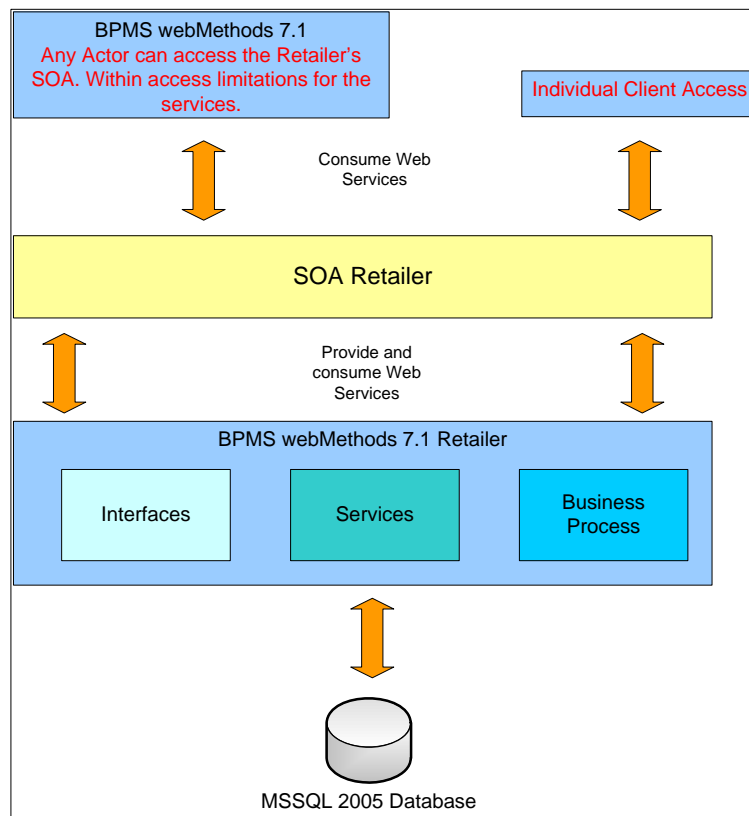


Abbildung 39 : Architektur Händler.

Quelle: Schaller Josef

10.3 Das Modell

Zuerst wird der Prozess für den Händler im *Designer* modelliert. Das Modell wird in der Rolle des *Business Analysten* erstellt und dem Modell liegt im ersten Schritt noch keine logische Implementierung zu Grunde.

Die Informationen für die Bestellung werden vom Benutzer eingegeben und danach zum Web Service *wsNewOrder* des Encaveurs geschickt. Mit dem IS-

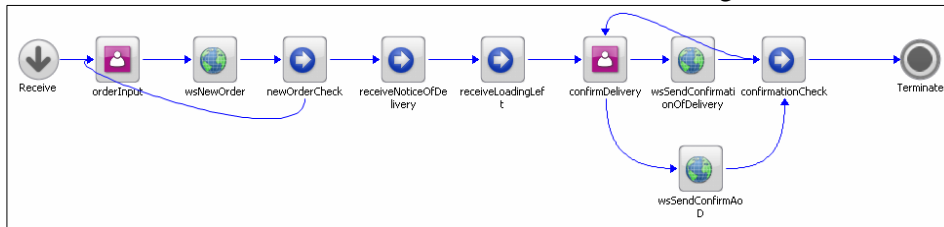


Abbildung 40 : Prozessmodell des Händlers.

Quelle: Schaller Josef

Service *newOrderCheck* wird geprüft ob die Bestellung in Ordnung ist, gemäss den Rückgabeparametern des Web Services. Sollte die Bestellung nicht gültig sein, muss sie neu eingegeben werden, der Prozess führt in diesem Fall zurück zum *orderInput* Schritt.

Sobald die Bestellung erfolgt ist, wird auf die Lieferbestätigung des Encaveurs gewartet *receiveNoticeOfDelivery*, danach auf die Bestätigung, dass der Transporteur die Ware geladen hat.

Sobald nun die Ware beim Händler eintrifft, wird dies bestätigt. Einmal geht eine Bestätigung an den Encaveur mittels *wsSendConfirmAoD* und einmal an den Transporteur mittels *wsSendConfirmationOfDelivery*. Bei den letzten zwei Diensten handelt es sich um Web Services der jeweiligen Akteure. Ein Check ob die Bestätigungen erfolgreich waren wird mit *confirmationCheck* ausgeführt. Danach ist der Prozess beendet.

10.4 Die einzelnen Prozessschritte

orderInput

Der Prozessschritt *orderInput* ermöglicht den Benutzer seine Bestellung aufzugeben. Es handelt sich um einen *Task*. Damit erhält später der Benutzer die Möglichkeit den Produktcode *idproduct*, seine Identität *idRetailer* und die Menge *quantity* der Artikel an den Encaveur weiterzuleiten. Wie der *Task* erstellt werden kann ist im Kapitel *Task Erstellung* ersichtlich.

wsNewOrder

In diesem Schritt wird der *Web Service* des Encaveurs konsumiert. Wie im Kapitel *Web Service Erstellung* beschrieben ist, kann mit Hilfe des Endpunkts des Web Service darauf zugegriffen werden. Siehe auch Kapitel *Architektur des Encaveurs*.

Damit der Web Service in den Prozess eingebunden werden kann, wird ein *Consumer* Web Service im *Developer* erstellt. Alternativ kann mit Hilfe der *WSDL* Datei der Web Service direkt im *Designer* gemacht werden. Der *wsNewOrder* Web Service erwartet die Produktidentität, Händleridentität und die Menge der gewünschten Artikel. Der Input wird in einem Dokument festgelegt *retOrderInputDoc*.

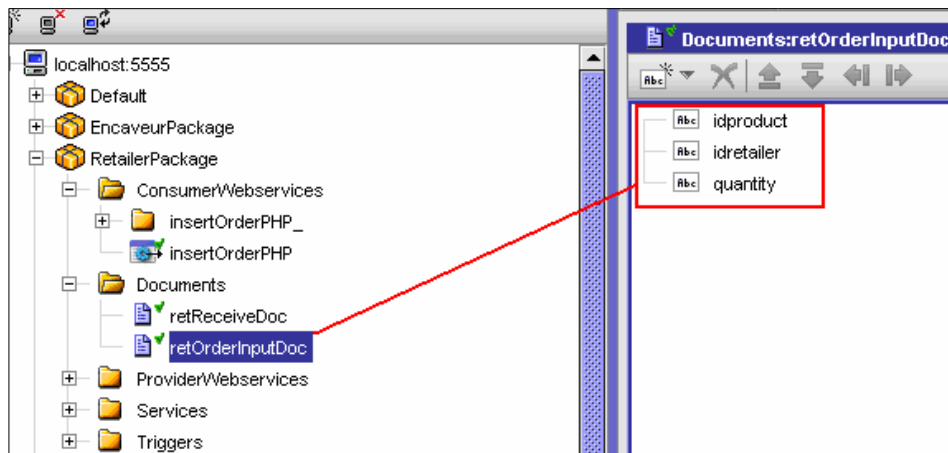


Abbildung 41 : Dokument mit nötigen Variablen für die Bestellung.
Quelle: Schaller Josef

Anmerkung: Um einen besseren Überblick zu erhalten sollten generell Dokumente als In- und Output verwendet werden, anstelle der Zuweisung einzelner Variablen.

newOrderCheck

Ist der Rückgabewert des Web Service *wsNewOrder* > 0 geht der Prozess zum nächsten Schritt über, andernfalls zurück zum *orderInput*.

Der Service *newOrderCheck* nimmt das Resultat des Web Service *wsNewOrder* entgegen. Er prüft ob die Bestellung gültig ist. Wie in den technischen

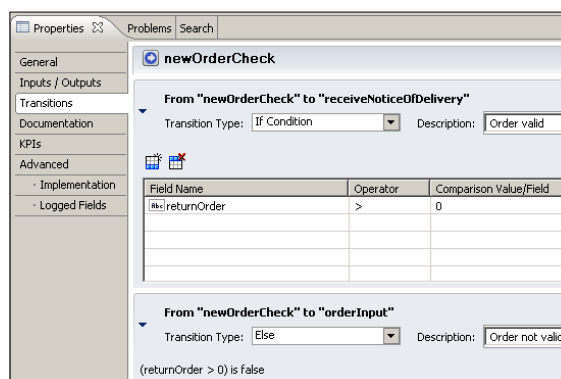


Abbildung 42 : Konditionen im Service *newOrderCheck*.
Quelle: Schaller Josef

Schnittstellen beschrieben ist, erwartet er einen *int* Wert. Ist dieser 0 gab's einen Fehler und die Bestellung muss erneut aufgeben werden. In den Eigenschaften des *newOrderCheck* im *Designer* werden dafür so genannte *Transitions* gemacht.

receiveNoticeOfDelivery* und *receiveLoadingLeft

Wurde die Bestellung aufgeben wartet nun das System auf die Bestätigung der Bestellung. Eine *Update Notification* auf der Bestelltabelle stellt sicher, dass beim entsprechenden Eintrag das System darüber benachrichtigt wird. Danach kann der nächste Schritt ausgeführt werden.

confirmDelivery

Beim *confirmDelivery* handelt es sich um einen *Task*. Dies aus dem Grund, weil eine menschliche Interaktion notwendig ist. Sobald die Ware beim Händler ankommt, werden die nötigen Daten für die folgenden beiden Web Services mit Hilfe dieses *Tasks* eingegeben und weitergeleitet

wsSendConfirmationAoD* und *wsSendConfirmationOfDelivery

Beide Prozesse können parallel ausgeführt werden. Die *Input*-Informationen für diese Services sind entweder noch in der *Pipeline* oder werden im *Task* hinzugefügt.

confirmationCheck

Analog zum *newOrderCheck* werden hier die Rückgabewerte validiert. Mit Hilfe von *Transitions* ist dies möglich. Ist alles gültig wird der Prozess abgeschlossen.

10.5 Entwicklung

Anmerkung: Die Web Services der anderen Akteure müssen für die Implementierung vorhanden sein. Es wird angenommen diese bestehen schon für sämtliche Akteure. Wie man zur Erstellung eines Web Service vorgeht wird im Kapitel Web Service Erstellung im Anhang erklärt.

Neben der Entwicklung der *Web Services* müssen noch andere Dienste/Services vorhanden sein. Beispielsweise das Einfügen eines neuen *Encaveurs* oder auch *Transporteurs*. Grundsätzlich sind auch die *Web Services*

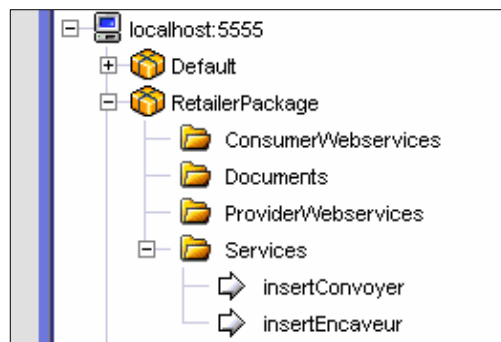


Abbildung 43 : Pakete im Developer 7.1

Quelle: Schaller Josef

nichts anderes als normale Dienste, der *Web Service* wird für einen normalen *IS-Service* erstellt, während für das Einfügen eines neuen Transporteurs kein *Web Service* erstellt werden muss.

Damit die Übersicht innerhalb der Pakete klar bleibt, wurde eine Struktur wie in der Abbildung 43 gewählt. Natürlich kann das nach Belieben geändert werden.

Der Ordner *Services* enthält sämtliche erstellten *Services*, wie der *Service* zum Einfügen eines neuen Transporteurs. Im Dokumenten Ordner werden die Dokumente gelagert, welche später zwischen den Prozessschritten zirkulieren und Variablen für diese enthalten.

In den untersten beiden *Web Service* Ordnern sind einerseits die *Web Services* enthalten die für andere Akteure angeboten werden, andererseits diejenigen die aufgerufen werden um Informationen an andere Akteure weiterzuleiten.

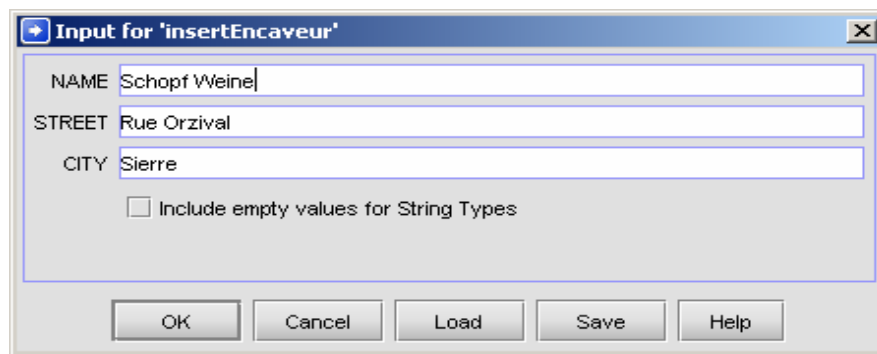


Abbildung 44 : Eingabemaske für Input im Developer 7.1.

Quelle: Schaller Josef

Über die Verwaltung des Integrationsservers wird eine neuer *Service* der Datensätze in die Datenbank des Händlers (*retailerDatabase*) speichern kann, konfiguriert. Ein Beispiel zur Erstellung eines *Service* mit Hilfe des Integrationsservers ist im Kapitel *Service Generation* beschrieben.

Im *Developer* ist es direkt möglich Dienste auszutesten. Als Ergebnis sind in der entsprechenden Datenbank die neuen Informationen vorhanden. Der *Developer* generiert eine Eingabemaske in die man den geforderten *Input* eintragen kann.

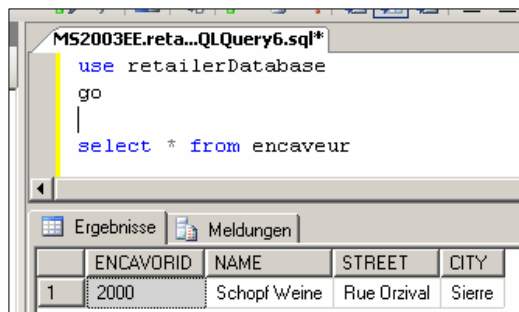


Abbildung 45 : Select Abfrage im MSSQL Management Studio.

Quelle: Schaller Josef

Bei Ausführung des Dienstes kann der Benutzer die benötigten Informationen in die Felder eingeben. Diese werden in der Datenbank gemäss der Service Generation gespeichert. Eine *Select-Abfrage* im *MS SQL Server Management Studio* verdeutlicht das Ergebnis. (Abbildung 45)

Da man für diese Dienste eine Verbindung aufbaut, ist es sinnvoll diese auch wieder zu schliessen. Im *Developer* stehen eine Menge

vordefinierter Dienste, *Flow* oder *Java Services*, zur Verfügung. Auch ein Dienst um die Verbindung zu schliessen.

Der erste Schritt *DB Service: pub.db:execSQL* wurde automatisch mit der *Service Generation* erstellt. Zusätzlich wurde ein weiterer Schritt *pub.db:close* angehängt um die Verbindung zur Datenbank zu beenden.

Die nächsten beiden Services dienen für das spätere Anbieten der *Web Services noticeOfDelivery* und *loadingLeft*. Analog dem vorher beschriebenen Vorgehen erstellt man zuerst die Dienste die die nötigen Werte in die Datenbank speichern. Wir berücksichtigen die Vorgaben in der Beschreibung der technischen Schnittstellen.

Der IS-Service *insertOrder* vom Händler speichert parallel zum Aufruf des *Web Service insertNewOrder* vom Encaveur die Informationen betreffend der neuen Bestellung in die lokale Datenbank.

Der Service vom Encaveur *insertNewOrder* wird nun zu einen *Provider Web*



Abbildung 46 : Einzelne Services im Service insertEncaveur

Quelle: Schaller Josef

Service, mit dessen Hilfe wir eine neue Bestellung in die Datenbank des Encaveurs speichern können. Im Kapitel *Service Level Agreements* ist die Einbindung der *SLA* detaillierter beschrieben.

11 *BPMS* Encaveur

11.1 Einführung

Überblick

Der Encaveur ist ein weiterer Akteur in der Logistikkette. Es sind folgende Annahmen in diesem Zusammenhang zu erwähnen:

Er besitzt eine Applikation mit der er seine Daten wie Bestellungen, Kunden und Produkte pflegen kann. Die Applikation wurde in *PHP* entwickelt. Um ein *Business Process Management* anzuwenden wurden die Geschäftsprozesse mit Hilfe eines Analysten analysiert und er ist nun bereit das *Business Process Management System* von webMethods in seine IT-Infrastruktur zu integrieren.

Diese Punkte gilt es zu beachten:

- Der Encaveur bietet *Web Service newOrder* für Bestellungen an.
- Der Transporteur kann durch den *Web Service confirmTransport* die Transportanfrage des Encaveurs bestätigen.
- Der Encaveur wird durch den *Web Service confirmAOR* vom Transporteur informiert, dass die Ladung übernommen wurde.
- Der Händler bestätigt beim Encaveur den Ladungserhalt mit Hilfe des *Web Service confirmAoD*.

11.2 Architektur Encaveur

Folgend sehen wir einen genaueren Überblick wie die verschiedenen Komponenten wie Applikation in *PHP*, *BPMS* und *SOA* des Encaveurs aussehen.

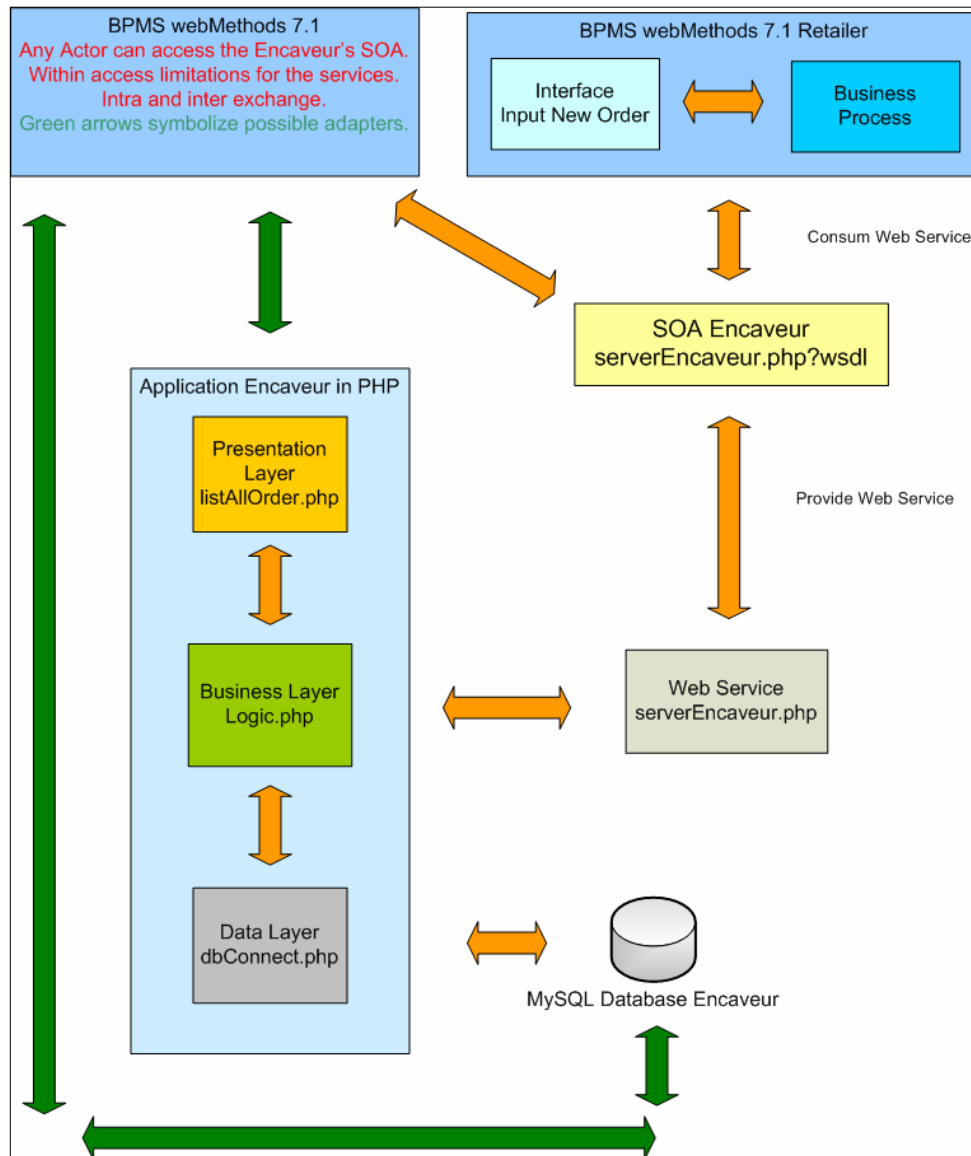


Abbildung 47 : Architektur Encaveur.

Quelle: Schaller Josef

Wie bereits erwähnt besitzt der Encaveur eine Applikation in *PHP*. Mit dieser kann er Bestellungen in die Datenbank speichern. In dieser Ausführung und zur Vereinfachung werden nur die Bestellungen genauer erklärt. Analog geht es mit den anderen Services die er zur Verfügung stellt.

Presentation Layer

Bisher konnte der Encaveur seine Bestellungen mit Hilfe des *Presentation Layer*, also der grafischen Schnittstelle verwalten. Diese grafische Schnittstelle fällt mit der Integration des BPMS weg, da diese später im *Designer* neu gemacht wird.

Id Product	Id Retailer	Quantity	Order Total Price	Id Transporter	Id Transport	Id Vehicle	Date of Loading	Loading OK	Date of Product Reception	Reception OK
9001	1002	450	state open	state open	state open	state open	state open	state open	state open	state open
9001	1002	4502	state open	state open	state open	state open	state open	state open	state open	state open

Abbildung 48 : Auflistung der Bestellungen mit Hilfe von PHP.

Quelle: Schaller Josef

Man sieht dass der Encaveur die Möglichkeit hat, sämtliche Bestellungen die er selber eingetragen hat, auflisten kann. Dies geschieht mit Hilfe der Datei *listAllOrder.php*.

Anmerkung: Die Datei *listAllOrder.php* wurde nur zum Zweck der Kontrolle der Einträge der Tabelle *Orders* entwickelt.

Business Layer

Bei der Applikation des Encaveur hat man eine *3tier* Architektur. Die Logik ist von der der grafischen Schnittstelle getrennt. Dies ist im Allgemeinen für ein späteres *Code Reusing* sehr von Vorteil und wünschenswert. Die Funktion *addNewOrder*

```
<?php
include("dbConnect.php");

function addNewOrder($idproduct,$idretailer,$quantity)
{
    $abfrage = "insert into orders(idproduct,idretailer,quantity)
    values('$idproduct','$idretailer','$quantity')";
    mysql_query($abfrage) or die(mysql_error());
}
?>
```

Code 1 : Funktion *addNewOrder* in der Logikdatei *logic.php*.

Quelle: Schaller Josef

ermöglicht eine neue Bestellung in die Datenbank zu speichern. Die Logik die hier präsent ist, ist unabhängig von den anderen Komponenten. Diese bietet nun die Möglichkeit die Funktion sehr gut wieder zu verwenden.

Data Layer

Im *Data Layer* wird die Verbindung zur *MySQL* Datenbank gemacht, damit die Daten der neuen Bestellung dort persistent gespeichert werden können.

Die *dbConnect.php* Datei enthält verbindungsspezifische Informationen. Der Quellcode kann im Anhang eingesehen werden.

Die bisher beschriebenen Schichten sind der eigentliche Status quo des Encaveurs, bevor irgendeine Integration oder ein *SOA* Aufbau gemacht wurde.

11.3 SOA Encaveur – PHP Server und Web Service

Mit Unterstützung der *nusoap.php*²¹ Klasse wird ein *PHP* Server erstellt auf dem der Web Service *ws_addNewOrder* gehostet wird.

Beim *ws_addNewOrder* handelt es sich um den Web Service *newOrder* der in der Beschreibung der technischen Schnittstelle erwähnt ist. Wichtig ist hier die Benutzung der Funktion aus dem *Business Layer* (*logic.php*) *addNewOrder*.

In der Logik gibt es keine Veränderungen. Es muss hier nur die Funktion *addNewOrder* aufgerufen werden.

Anmerkung: Je nachdem ob man schon eine Applikation für die Verwaltung seiner Daten besitzt oder ob man eine Neue komplett im BPMS entwickelt, können die Service Level Agreements gemacht werden. Es macht Sinn diese dort zu integrieren, wo die Web Services gehostet werden. Im Fall des Encaveurs in der serverEncaveur.php oder logic.php, beim Händler der noch keine Applikation besitzt, direkt als Service im BPMS.

²¹ Nusoap ist ein SOAP Toolkit für PHP. Weitere Informationen und der Download ist verfügbar unter: http://sourceforge.net/project/showfiles.php?group_id=57663 [Letzter Zugriff: 05.12.2007]

Die WSDL Datei kann durch Aufruf des Servers eingesehen werden.

Anmerkung: Dies hängt vom Installationsverzeichnis des Web Servers ab.

```
<?php
require_once("nusoap.php");
require_once("logic.php");

$ns="http://localhost/";
$server = new soap_server();
$server->configureWSDL('Order',$ns);
$server->wsdl->schemaTargetNamespace=$ns;
$server->register('ws_addNewOrder',array('idproduct'=>'xsd:integer','idretailer'
=>'xsd:integer','quantity'=>'xsd:integer'),
    array('result'=>'xsd:string'),
    $ns,"$ns#ws_addNewOrder");
function ws_addNewOrder($idproduct,$idretailer,$quantity)
{
    return new soapval('return','xsd:string',addNewOrder($idproduct, $idretailer,
$quantity));
}
$server->service($_HTTP_RAW_POST_DATA);
?>
```

Code 2 : Datei serverEncaveur.php.

Quelle: Schaller Josef

Nun hat man einen *Web Service* der neue Bestellungen in die Datenbank speichert. Dieser Service ist ein Teil der *SOA*. Auch alle anderen Services die der Encaveur anbieten will, sind Bestandteil dieser (seiner) serviceorientierten Architektur.

Anmerkung: Beim Aufbau des BPMS macht es Sinn, dass auch andere Funktionen wie Kundenpflege usw. in die SOA aufgenommen werden. Je nach dem in welcher Form das Business Process Management dies benötigt und wie die Eigenschaften der Geschäftsprozesse konstituiert sind.

Es besteht aber nicht nur die Möglichkeit eines Zugriffs mittels Web Services, auch Adapter können benutzt werden. Die Benutzung von Adaptern ist aber weitaus komplexer, da für Applikationen wie die des Encaveurs eigene Adapter geschrieben/entwickelt werden müssen. Es sind in diesem Sinne keine Adapter für Zugriffe auf die Logik von PHP Applikationen vorhanden.

11.4 Das Modell

Bis auf zwei *Web Services* *wsRequestTransport* und *wsSendNoticeOfDelivery* sind alles *IS-Services*. Dies darf aber nicht darüber hinwegtäuschen, dass innerhalb der *Services (Flow Services)* *Web Services* liegen. Die darunter liegenden *Web Services* werden alle vom *PHP* Server des Encaveurs angeboten.

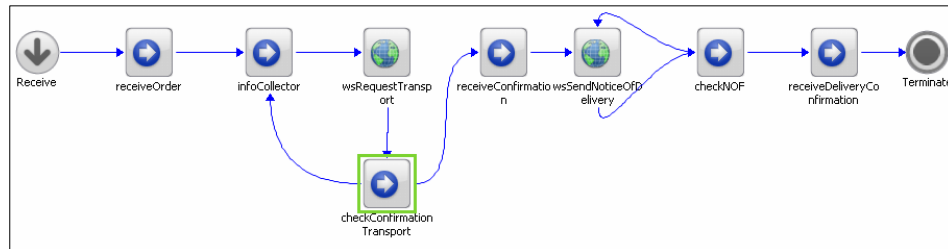


Abbildung 49 : Prozessmodell des Encaveur.

Quelle: Schaller Josef

Der erste Schritt im Prozess ist das Warten auf eine neue Bestellung. Sobald die Bestellung eingetroffen ist werden zusätzlich benötigte Informationen für den *wsRequestTransport* Web Service vom *infoCollector* geholt. Ist der Aufruf des *Web Services* nicht gültig oder liegt ein anderes Problem vor, führt der Schritt zurück zum *infoCollector*. Beispielsweise kommt im *infoCollector* die Identität für den Encaveurs hinzu.

Da es sich in diesem Prozess immer um denselben Encaveur handelt kann dieser Wert fix mitgegeben werden. Wie bei den anderen Variablen die dem *Web Service* übergeben werden ist es notwendig die Identitäten für den Transporteur kompatibel zu halten, will also heissen das die Identität des Encaveurs beim Transporteur diesem bekannt sein muss.

Da es in der Natur einer Bestellung liegt diese so bald als möglich auszuliefern wird für das gewünschte Auslieferungsdatum der heutige Tag plus eins weitergegeben.

Der *Output* des *Web Service wsRequestTransport* wird an den Service *checkConfirmation* weitergeleitet, der entscheidet ob die Rückgabe gültig ist und fortgefahren werden kann. Angenommen das Auslieferungsdatum ist nicht möglich leitet dies der *checkConfirmation* Service an den *infoCollector* weiter, der den Tag um eins erhöht und wiederum abschickt.

Dies geschieht solange bis ein freier Termin für die Auslieferung gefunden wurde. Die schrittweise Erhöhung der Tage stellt sicher, dass der Auslieferungstermin der nächstmögliche ist.

Sobald die Bestätigung des Transports beim Encaveur eintrifft wird dem Händler über seinen *Web Service wsSendNoticeOfDelivery* mitgeteilt, dass die Ware

geliefert werden kann. Wiederum ist hier ein Check für die Rückgabewerte des *Web Service* der den *Output* validiert.

Bis der Händler seine Bestellung vom Transporteur erhält und die Lieferung durch den *Web Service confirmAoD* vom Encaveur meldet, passiert nichts mehr.

Danach wird der Prozess beendet.

Anmerkung: Es ist ersichtlich, dass im ganzen Prozess keine menschliche Interaktion notwendig ist. Alternativ besteht die Möglichkeit Bestätigungen in den Prozess zu integrieren. Beispielsweise bestätigt der Encaveur zuerst den Transport oder dergleichen. Dies sind Entscheidungen die vom Management und dem Business Analysten getroffen werden.

11.5 Die einzelnen Prozessschritte

receiveOrder

Es wird auf einen Bestelleingang des Händlers oder eines anderen zugriffsberechtigten Akteurs gewartet. Zur Prüfung des Bestelleingangs wird ein *Flow Service* erstellt, der in gewünschten Zeitabständen die Datenbank des Encaveurs nach neuen Einträgen überprüft. Es sind zwei Komponenten im *Flow Service* nötig: Ein Service für die Intervallberechnung und ein *Web Service* in *PHP* der die Abfragen in der *MySQL* Datenbank des Encaveurs macht. Somit kann die Tabelle *Orders* des Encaveurs überwacht werden und allfällig neue Bestellungen gemeldet werden.

infoCollector

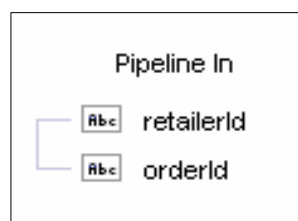


Abbildung 50 : Pipeline Input.

Quelle: Schaller Josef

Identität der Bestellung und das Lieferdatum.

Der *infoCollector* hat als Input die Werte der Bestellung, die neu in der Datenbank eingetragen wurden. Um eine Transportanfrage zu machen benötigt er folgende Informationen: Identität des Encaveurs, Identität des *Retailers*,

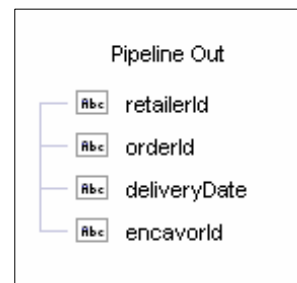


Abbildung 51 : Pipeline Output.

Quelle: Schaller Josef

Die Identität des Encaveurs ist immer dieselbe und kann fix im *Flow Service* deklariert werden.

Zur Berechnung des Lieferdatums wird das heutige Datum genommen und weiter geschickt. Das aktuelle Datum kann mit dem *Build-in Service pub.date:getCurrentDateString* ermittelt werden. Wird das Lieferdatum vom *Web*

Service des Transporteurs abgelehnt kommt die Information vom *checkConfirmationTransport* zurück und es wird um einen Tag erhöht.

Die Identität des Händlers ist bereits in der Prozesspipeline und muss nicht extra aus anderen Quellen geholt werden. Einzig die Identität der Bestellung muss mit Hilfe eine Tabellenabfrage geholt werden. Siehe auch Kapitel *Service Generation* im Anhang.

checkConfirmationTransport

Hier geschieht die Validierung der Rückgabewerte des *Web Service wsRequestTransport*. Falls diese in Ordnung sind kann zum nächsten Schritt weitergegangen werden, ansonsten zurück zum *infoCollector*. Die Rückgabewerte des *Web Service wsRequestTransport* des Transporteurs sind 0 oder > 0.

wsRequestTransport

Es wird der *Web Service* des Transporteurs aufgerufen um einen Transport für die Ware zu erhalten.

receiveConfirmation

In diesem Schritt erhält man die Rückgabewerte des *wsRequestTransports*, die zuvor vom *checkConfirmationTransport* validiert wurden. Laut Beschreibung der technischen Schnittstellen (siehe Anhang) wird nur ein Parameter zurückgegeben, welcher besagt ob der Transport in Ordnung ist oder nicht.

wsSendNoticeOfDelivery

Sobald die Werte in der Datenbanktabelle Bestellungen des Encaveurs über die Transportbestätigung eingegangen sind, wird der Händler informiert, dass die Ladung unterwegs ist und am Datum X bei ihm eintrifft.

Die benötigten Parameter sind bis auf die Identität des Transports noch in der *Pipeline* und können als *Input* weitergeleitet werden. Aus der Beschreibung der technischen Schnittstelle ist nicht ersichtlich woher diese Identität kommt.

checkNOF

checkNOF steht für *checkNoticeOfDelivery*. Hier geschieht die Validierung ob die Übermittlung der *Notice of Delivery* erfolgreich war. War sie nicht erfolgreich wird gemäss Rückgabewerten des *Web Service wsSendNoticeOfDelivery* die Werte korrigiert.

receiveDeliveryConfirmation

Bis der Händler über den *Web Service* des Encaveurs *confirmAoD* den Warenerhalt zurückmeldet ist der Prozess des Encaveurs ausgesetzt. Erst wenn die Tabellen mit den Werten der Auslieferungsbestätigung gefüllt ist kann der Prozess abgeschlossen werden. Dies geschieht analog zum Schritt *receiveOrder*.

12 BPMS Transporteur

12.1 Einführung

Bemerkungen

Aus zeitlichen Gründen und den aufgetretenen Problemen während der Arbeit wird hier nur das mögliche Prozessmodell wiedergeben und zusammenfassend geschildert wie der Prozess abläuft.

Die Gründe und aufgetretenen Probleme sind im Kapitel *Probleme während der Diplomarbeit* geschildert.

Überblick

Der Transporteur ist der dritte Akteur in der Logistikkette. Er besitzt eine Applikation zur Verwaltung seiner Daten wie Kundenpflege, ausgeführte Transporte. Diese Applikation kann in Java oder C# geschrieben sein.

12.2 Architektur Transporteur

Vergleich möglicher Architekturtypen anhand der Architekturen des Händlers und des Encaveurs in den entsprechenden Kapiteln.

12.3 Das Modell

In Abbildung 52 ist das Prozessmodell des Transporteurs wiedergegeben. In der Tabelle 12 finden sich zusammenfassend die Beschreibung der einzelnen Prozessschritte wieder.

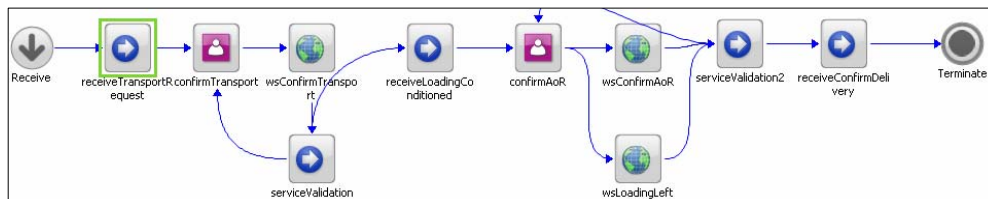


Abbildung 52 : Prozessmodell des Transporteurs.

Quelle: Schaller Josef

Schritt	Beschreibung
receiveTransportRequest	Transporteur erhält eine Transportanfrage durch seinen Web Service.
confirmTransport	Manuelle Bestätigung des Transports.

wsConfirmTransport	Daten der manuellen Bestätigung werden mit Hilfe des Web Services vom Encaveur gemacht. Validierung der Rückgabewerte mit serviceValidation.
receiveLoadingConditioned	Sobald die Ware beim Encaveur geladen wurde, wird der Transporteur informiert, hierfür dient sein Web Service loadingConditioned.
confirmAoR	Dem Encaveur wird bestätigt, dass die Ware geladen wurde.
wsLoadingLeft	Der Händler wird informiert, dass die Ladung unterwegs ist.
serviceValidation2	Die Rückgabewerte der Web Services werden validiert. Falls nicht in Ordnung zurück zum Schritt confirmAoR
receiveConfirmDelivery	Der Händler meldet die Ankunft der Ware.

Tabelle 11: Prozessschritte und Beschreibung.

Quelle: Schaller Josef

13 Service Level Agreements (SLA)²²

Jeder *Web Service* unterliegt gewissen Anforderungen wann und wie auf diesen zugegriffen werden kann. Beispielsweise soll der *Web Service newOrder* des Encaveurs von Montag bis Freitag von 0800h – 0500h verfügbar sein.

Beim Händler muss dieser 24h zur Verfügung stehen und bis zu 10 Bestellungen pro Minute annehmen können, die Antwortzeit beträgt maximal 5 Sekunden.

Die *SLA's* sind mit Konditionen vergleichbar und müssen geprüft werden ob sie in Ordnung sind. Damit diese *SLA's* integriert werden können wird ein neuer *Java Service* erstellt, der diese Konditionen prüft.

Obwohl dieser *Java Service* als eigener Schritt in den Geschäftsprozess kommen könnte, wird dies vermieden, da es nur ein Test ist für einen bestimmten Prozessschritt ist. Aus dem Grunde wird der *Java Service* in den *Flow Service* des *Web Services* gehängt. Besser gesagt in den dem *Web Service* zugrunde liegenden Service.

Natürlich kommt es drauf an wie man die Integration absolviert. Der Händler beispielsweise hat die *Service Level Agreements* im *BPMS*, in der Logik, integriert, da er die *Web Services* mit dem *BPMS* entwickelt. Der Encaveur kann diese in der Logik der Applikation integrieren, oder im Bereich der den *Web Service* anbietet.

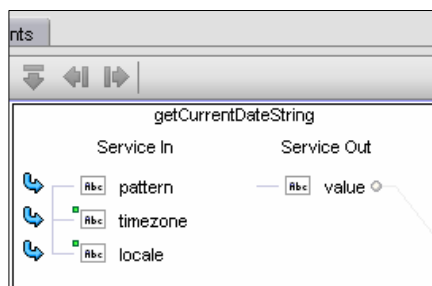


Abbildung 53 : Service Input Parameter.
Quelle: Schaller Josef

Anhand eines Beispiels soll gezeigt werden wie die *SLA* für die *Web Services* zu integrieren sind, anhand des *Web Services newOrder* des Encaveur.

Folgende Implementierung eines *Service Level Agreements* soll als Veranschaulichung dienen.

Der *Web Service newOrder* soll nur von Montag bis Freitag verfügbar sein.

Dafür erstellt man als erstes einen neuen *Flow*

Service hier: *encSLAFlow*.

Mit Hilfe des vordefinierten Service: *pub.date:getCurrentDateString* ermittelt man das aktuelle Datum. In der Ansicht der *Pipeline* sieht man, dass dieser Service drei

²² Quelle: webMethods. webMethods Integration Server Built in Services Reference 7.1. webMethods_webMethods_Integration_Server_Built-In_Services_Reference_7_1-1.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 15.11.2007]

Quelle: webMethods. webMethods Developer User's Guide 7.1. webMethods_Developer_User's_Guide_7_1.pdf [Online]. Erreichbar als registrierter Benutzer unter: <http://advantage.webmethods.com> [Letzter Zugriff: 15.11.2007]

Werte benötigt, obligatorisch ist nur *pattern*, die anderen Werte wie *timezone* und *local* werden bei Nichtdeklaration vom System übernommen.

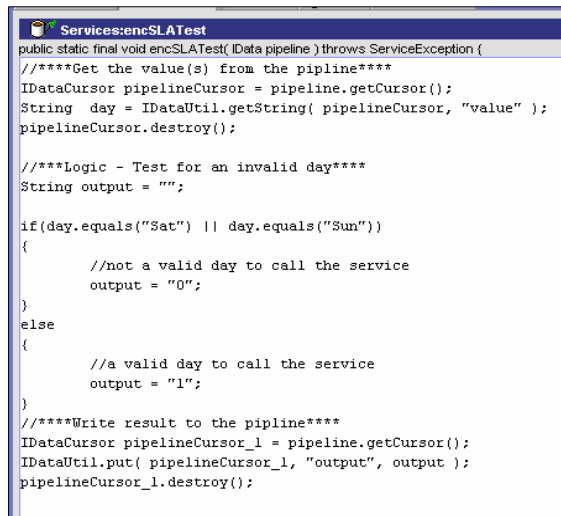
pattern bedeutet in diesem Fall welchen Teil des Datum gewünscht wird. Was für *patterns* für diesen Service bestehen, kann man im Dokument: *webMethods Integration Server Built in Services Reference* detaillierter nachlesen.

Hier wurde für das *pattern* der Tag des Monats gewählt. Die blauen Pfeile deuten darauf hin, dass die Werte fix zugewiesen wurden, so dass der Service nicht bei jeder Ausführung nach den obligatorischen Feldern fragt. Für die Zeitzone wurde *ECT* (*European Central Time*) und die Lokale (*en*) für Englisch gewählt. Die Ausgabe *value* des Service liefert uns den String vom Wochentag in der Englischen Abkürzung *Sat* für *Saturday* etc.

Nun hat man die Ausgabe des Strings in der *Pipeline*. Jetzt kommt der *Java Service* zum Zug. Er nimmt den *String* und testet ob es sich um einen Tag handelt der ungültig ist (Samstag und Sonntag).

Ist der Tag nicht ein Samstag oder ein Sonntag wird dem *Output* der *Wert=1* zugewiesen. Das Resultat wird danach wieder in die *Pipeline* geschrieben, damit es für den nächsten Schritt vorhanden ist.

Dieser *Java Service* wird nun in den *Flow Service* *encSLAFlow* eingehängt. Nun sind bereits zwei Dienste im *Flow Service* derjenige der das Datum ermittelt und der der



```

public static final void encSLATest( IData pipeline ) throws ServiceException {
    //****Get the value(s) from the pipeline****
    IDataCursor pipelineCursor = pipeline.getCursor();
    String day = IDataUtil.getString( pipelineCursor, "value" );
    pipelineCursor.destroy();

    //****Logic - Test for an invalid day****
    String output = "";

    if(day.equals("Sat") || day.equals("Sun"))
    {
        //not a valid day to call the service
        output = "0";
    }
    else
    {
        //a valid day to call the service
        output = "1";
    }

    //****Write result to the pipeline****
    IDataCursor pipelineCursor_1 = pipeline.getCursor();
    IDataUtil.put( pipelineCursor_1, "output", output );
    pipelineCursor_1.destroy();
}

```

Abbildung 54 : Java Service zum Testen der SLA.
Quelle: Schaller Josef

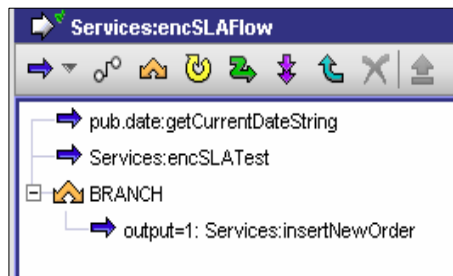


Abbildung 55 : Flow Service mit Branch Step.

Quelle: Schaller Josef

dieses aufgrund eines *Strings* testet. Der dritte Service speichert die Daten in die Datenbank. Dieser muss aber wissen ob es sich um einen gültigen Tag also *Wert=1* handelt oder um einen ungültigen. Hierfür erstellen wir im *Flow Service* einen so genannten *Branch Step* mit dem man die Bedingung testen kann.

Damit die Bedingung getestet werden kann, wird in den Eigenschaften des *Branch Steps* die Option *Evaluate* auf *True* gesetzt.

Danach stellt man die Bedingung im *insertNewOrder* in den Eigenschaften ein. *Label: output=1*. Wichtig ist den Name

der Variable in der *Pipeline* zu benutzen. *Output* heisst die Variable die vom *encSLATest* Service in die *Pipeline* geschrieben wird.

Damit keine *NULL* Werte in die Datenbank geschrieben werden, muss der *Input* für den Service definiert werden. Dafür wird ein neues Dokument erstellt, welches die nötigen Variablen für das Einfügen einer neuen Bestellung enthält. Hier: *IdRetailer*, *IdProduct*, *Quantity*.

Als letzten Schritt ist es nun notwendig die Variablen richtig zuzuweisen.

Anmerkung: Heissen die Variablen eines Outputs gleich wie die des Inputs ist keine Zuweisung notwendig. Ansonsten kann man im Tab Pipeline diese Zuweisung machen. Siehe Abbildung 56.

Man weist die Variablen einfach mit der Maus zu. Da das Einfügen der Daten in die Datenbank der letzte *Flow Step* ist, können die Werte im Bereich *Pipeline Out* verworfen werden, da sie nicht weiter gebraucht werden.

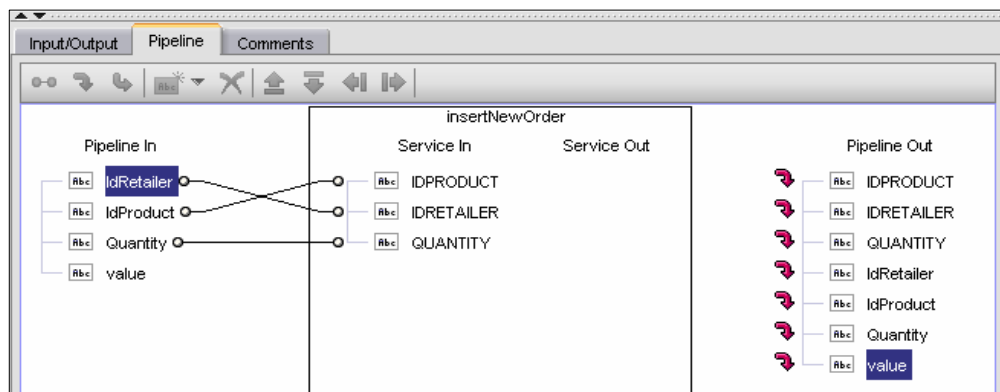


Abbildung 56 : Pipeline im Service insertNewOrder.

Quelle: Schaller Josef

Nun ist es nur an Tagen von Montag – Freitag möglich neue Bestellungen mit diesem Service zu tätigen. Bis jetzt ist dieser Service nur ein IS-Service. Dieser kann aber wie jeder andere Service bei der Erstellung eines *Provider Web Services* ausgewählt werden, siehe hierfür Abschnitt *Web Service Erstellung* im Anhang.

14 Die Notwendigkeit eines *BPMS* für KMU's

Als Voraussetzung zur Integration eines *BPMS* gehört ein aktives *Business Process Management*. Es ist sinnlos oder nicht ratsam ein *BPMS* integrieren zu wollen ohne dass das Management für diesen Schritt besteht.

Die Umsetzung eines solchen Projektes bringt einige Kosten mit sich. Der Preis für die Software wird in diesem Fall nicht unbedingt so sehr relevant sein, da der zeitliche Aufwand erheblich höher und mit hohen Arbeitsstunden verbunden ist.

Für einen kleineren Betrieb mit kleinem Budget für IT ist dies wohl eher nicht ratsam, zumal in diesem Fall mehrere Alternativen bestehen. Um eine serviceorientierte Architektur umzusetzen ist ein komplexes *BPMS* nicht notwendig.

Alternativ zur Integration des *BPMS* beim Encaveur besteht durchaus die Möglichkeit eine serviceorientierte Architektur aufzubauen ohne ein wuchtiges *BPMS* umsetzen zu wollen. Ein einfacher Java Applikationsserver auf dem Web Services laufen wäre hier die wohl sinnvollere Alternative.

Die Softwarekosten sind hier nicht existent und der zeitliche Aufwand die geforderten Web Services zu integrieren, inklusive der Installation eines Applikationsservers, ist im Vergleich zur Implementierung eines *BPMS* von webMethods, gering.

Natürlich muss auch in Betracht gezogen werden wie weit es möglich ist bestehende Aktivitäten in Prozesse aufzugliedern und welchen *Return on Investment (ROI)* man schlussendlich erhält. Man muss sich die Frage stellen ob eine Aufspaltung in viele kleine Prozessschritte nicht eine zusätzliche unnötige Komplexität mit sich bringt.

Ein weiterer Aspekt spielt die Struktur der Unternehmung, wie sieht es im Management aus oder besteht überhaupt annähernd so etwas wie ein Management in *KMU's* und ist dies fähig ein solches *BPMS* zu meistern und damit ist nicht die Implementierung gemeint?

Abschliessend muss man erwähnen, dass eine eindeutige Ablehnung oder Zustimmung für ein *BPMS* wie das von webMethods, hinsichtlich des Nutzens für eine *KMU*, nicht gegeben werden kann.

Tendenziell kann man aber sagen, dass der Nutzen und Sinn parallel zur Grösse der Unternehmung steigt, also eher im oberen Bereich der mittleren Unternehmungen liegt.

15 Probleme während der Diplomarbeit

In diesem Kapitel sind die Probleme die während der Diplomarbeit aufgetreten sind erläutert. Es handelt sich hier ausschliesslich um Probleme die eine zeitliche Verzögerung mit sich gebracht haben oder als kritisch angesehen werden kann. Probleme welche während der Installation und Konfiguration von *webMethods* aufgetaucht sind, konnten die meisten mit Hilfe der Dokumentation von *webMethods* gelöst werden.

15.1 Verspätung der Software

Bei Beginn der Diplomarbeit am 17. September 2007 war die Software noch nicht verfügbar.

Dies war insofern noch nicht unbedingt ein Problem weil in den ersten zwei Wochen die Erstellung des Pflichtenhefts geplant waren. Es war aber doch relevant, als dass für gewisse Punkte im Pflichtenheft die Software nötig gewesen wäre. Da der Link mit den Logininformationen für den *Download* des *webMethods Installers* erst am Dienstag 2. Oktober 2007 zur Verfügung stand, war schon eine kleine zeitliche Verspätung vorprogrammiert.

Dieser Sachverhalt war aber noch kontrollier- und korrigierbar.

15.2 Ressourcen

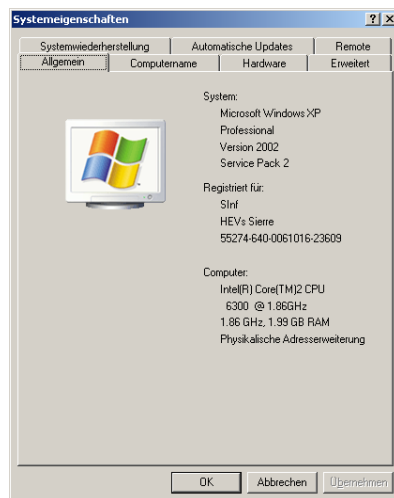


Abbildung 57 :
Systemeigenschaften
Arbeitscomputer.
Quelle: Schaller Josef

Die Systemeigenschaften des Computers für die Diplomarbeit sind in Abbildung 57 wiedergegeben.

Vorgaben

Laut Vorgaben sollte die Installation des *BPMS* auf einer virtuellen Maschine geschehen. Hierfür wurde das *Image* welches von der Schule zur Verfügung gestellt wurde benutzt. Es handelte sich hierbei um ein Windows Server 2003 System, inklusive *.Net Framework* mit *Visual Studio* und *MSSQL Server 2005*. Weiter sind auf diesem *Image* eine Vielzahl an Softwarekomponenten integriert, so dass das *Image* eine Grösse von rund 15.8 GB aufweist.

Ein Vergleich mit den vorgeschlagenen Werten im Kapitel Systemvoraussetzung lässt ziemlich schnell erkennen, dass unter Benutzung einer *VM*

Ware Workstation für das *Image* die Ressourcen nicht genügen. In Abbildung 57 sieht man, dass das System 2 GB Arbeitsspeicher aufweist. Diese sind unter dem *Host* und dem virtuellen System zu verteilen. Damit man nur schon mit dem *Host* System vernünftig arbeiten kann braucht es 1 GB Arbeitsspeicher, der Rest wurde dem virtuellen zugewiesen.

Virtuelles System

Da für das zu integrierende Szenario vorgesehen war, dass für jeden der drei Akteure ein eigenes *BPMS* erstellt werden sollte wären drei virtuelle Systeme nötig gewesen. Da die Ressourcen nicht einmal für ein virtuelles System gereicht haben, fragte ich nach einer Möglichkeit für weitere PC's für die Arbeit. Diese konnten mir aber nicht gewährt werden und es sollte ein einziges *BPMS* für alle drei Akteure genügen.

Also startete ich die Installation der *webMethods BPMS Suite 7.1* auf dem *VM Ware Image*.

Nach der Installation und Konfiguration die schon erheblich länger als geplant gedauert haben, wurde ich darin bestätigt, dass die die Komponenten alle zusammen zu viele Ressourcen brauchen. Da nicht mehrere PC's zur Verfügung standen, mussten alle Server und Komponenten auf dem gleichen System laufen. Wie in der Architektur ersichtlich ist, sind einige Server die gleichzeitig laufen müssen, dies hat dem entsprechend Ressourcen gefressen und ein Arbeiten war äusserst schwierig, quasi nicht möglich, zumal der *My webMethods Server* dadurch regelmässig abstürzte.

GSX Server

Auf weitere Nachfrage nach mehr Ressourcen wurde mir am 6. November 2007 ein Platz auf einem *GSX Server* der Schule zur Verfügung gestellt und das *Image* wurde vom Informatikdienst der Schule aufgespielt. (Vier Wochen vor Abgabe der Diplomarbeit)

Mit Hilfe der *VM Ware Console* konnte auf das *Image* zugegriffen werden.

Leider war das nicht die Lösung meines Problems. Die virtuelle Maschine wies ständig eine Beanspruchung des Systems von 100% auf. Ein Arbeiten war nicht möglich. Zudem beeinträchtigte die Auslastung die Benutzung der Systeme von anderen Benutzern.

Auslagerung

Eine weitere Möglichkeit den *GSX Server* doch noch benutzen zu können, wäre eine Auslagerung gewisser Server auf den *Host* gewesen. Aufgrund der Netzwerktopologie wurde das einzig mögliche Szenario die Auslagerung des *MS SQL Servers* auf den lokalen *Host* und das *BPMS* auf dem *GSX Server* gewählt.

Dies hat aber nicht viel geändert, ein Arbeiten war weiterhin nicht möglich.

Neues Image

Aus dem Grund habe ich mich entschlossen ein neues *VM Ware Image* aufzusetzen, auf dem nur die nötigsten Programme installiert sind.

Es wurde ein *MS 2003 EE Server* genommen, inklusive *MS SQL Server 2005*. Zusätzlich wurden die Java Komponenten installiert.

Andere Software wie *.Net Framework* und *Visual Studio* wurden nicht integriert. Dadurch konnte das *Image* auf rund 66% des vorherigen *Images* reduziert werden. Dabei hat die Neuinstallation des gesamten *BPMS* inklusive der nötigen Konfiguration wiederum enorm Zeit in Anspruch genommen.

Am 13. November 2007 war die Installation und Konfiguration des *BPMS* auf dem neuen *Image* abgeschlossen. Das Resultat war, dass es etwas besser als zuvor lief. Der *Designer* und *Developer* konnten gut benutzt werden. Ein Benutzen aller Komponenten zur gleichen Zeit war aber auch hier äusserst schwierig.

Adapter

Der *JDBC Adapter* ist ein Zusatzmodul für die *webMethods BPMS Suite*. Der *JDBC Adapter* ermöglicht den Datenaustausch mit einer Datenbank mit Hilfe des *JDBC* Treibers.

Bei der Version 7 von *webMethods* kann der *JDBC Adapter* als Zusatzmodul während dem Installationsprozess installiert werden. Bei der Version 7.1 ist dies möglich.

Sämtliche Dokumentation für den *JDBC Adapter* sind nur für die Versionen 6.x verfügbar. Das Paket *wmJDBCAdapter* ist in der Administrationsoberfläche unter *Adapter* nicht ersichtlich, auch besteht kein Paket *wmJDBCAdapter* bei den anderen Paketen. Obwohl beim *Installer* sämtliche Komponenten zur Installation ausgewählt wurden.

Eigenartigerweise kann dieses Zusatzmodul beim Installieren nicht angewählt werden, ob dies aus Gründen der Lizenzierung oder anderer Restriktionen nicht möglich ist, bleibt ungewiss.

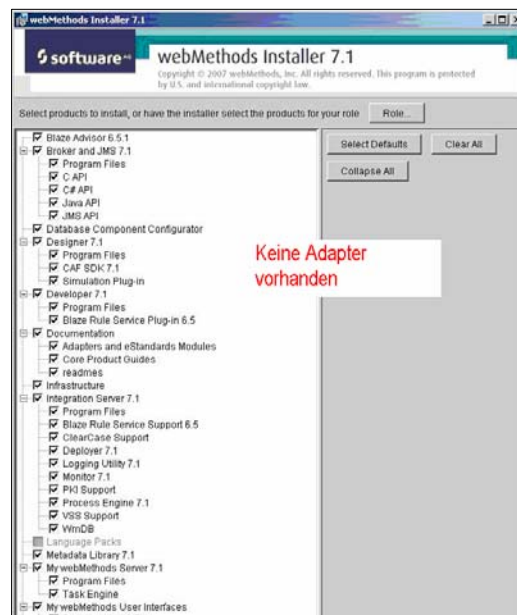


Abbildung 58 : Komponentenauswahl, keine Adapter vorhanden.

Quelle: Schaller Josef

In Abbildung 58 und 59 sind zwei *Printscreens* im Vergleich. Es ist erkennbar, dass die *Adapter* in der Version die für die Diplomarbeit benutzt wurde, nicht existieren.

Da aber die *Adapter* für einige Aspekte der Arbeit und Implementierung wichtig sind, wurde darauf während der Arbeit eingegangen, lediglich die Entwicklung für die Adapter und den damit eng zusammenhängenden Komponenten wurde nicht gemacht.

Fazit

Zusammenfassend muss hier erwähnt werden, dass die Arbeit mit dem *BPMS* unter den zur Verfügung gestellten

Ressourcen mehr als schwierig war und dementsprechend viel Verzögerung mit sich gebracht hat. Es mussten gewisse Prioritäten getroffen werden und Teile des Pflichtenhefts gestrichen werden.

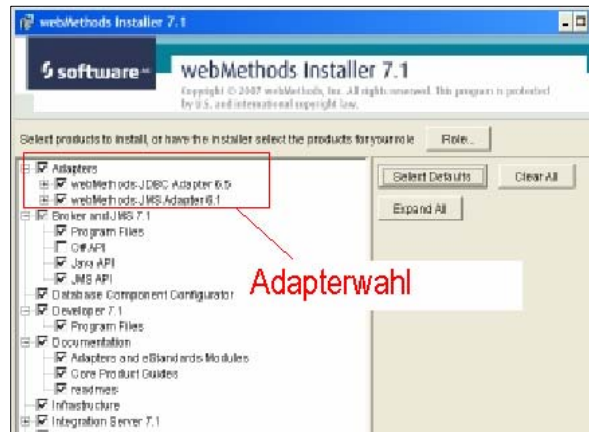


Abbildung 59 : Komponentenauswahl, Adapter vorhanden.

Quelle: Carlson Mark, <http://www.wmusers.com>

16 Synthese

Wie in der Einführung beschrieben sind einige Aufgaben für die Abwicklung dieser Diplomarbeit angefallen.

Es waren Aufgaben zu erledigen die eine weite Spanne aufweisen. Es wurden Aufgaben in den Bereichen *Business Analyse*, Forschung, Entwicklung und Architektur gemacht. Die Schwerpunkte lagen darin, ein *BPMS* zu integrieren und die Möglichkeiten einer serviceorientierten Architektur (SOA) aufzuzeigen. Wobei die Auseinandersetzung mit dem *BPMS* von *webMethods* als grösster Brocken hervor trat.

Es war eine gewisse Zeit nötig erstmals einen Überblick über sämtliche Komponenten zu gewinnen. Auch die Konfiguration nach der Installation bedingte viel Nachlesen in der Dokumentation. Die vielen Verweise zwischen den Dokumenten der einzelnen Komponenten haben schon manchmal den Überblick verlieren lassen.

Der Punkt der Applikationsentwicklung für die drei verschiedenen Akteure ist während der Arbeit etwas in den Hintergrund gerückt, dies aufgrund der Feststellung, dass es kein primäres Ziel darstellt und die zeitlichen Verzögerungen diesen Entscheid notwendig machten. Die Applikationen sollte dazu herangezogen werden damit ein Aufzeigen der Implementierung eines bestehenden Systems möglich sein soll. Dies wurde anhand des Beispiels des Encaveurs aufgezeigt, es stellt aber nur eine Variante von vielen dar.

Einerseits handelt es sich um ein *BPMS*, andererseits um eine serviceorientierte Architektur und dann wiederum um die Vereinigung beider Komponenten. Eine SOA kann durchaus ohne ein *BPMS* verwirklicht werden, beispielsweise durch Anbieten von *Web Services* für die Akteure.

Ein *BPMS* ohne *eigene SOA* ist in diesem Sinne nicht möglich, zumal eine *SOA* nicht nur aus *Web Services* bestehen muss, obwohl dies im Moment wohl einer der vielversprechendsten Ansätze ist. Um es ein bisschen zu abstrahieren besteht das *BPMS* von *webMethods* nur aus Services. Alles ist ein Service. Die Interaktionen zwischen Prozessen und Prozessschritten sind Services.

Dies allein stellt schon eine Art *SOA* dar. Ob die nun „öffentlich“ zugänglich ist oder nicht, ist nicht relevant.

Anhand der verschiedenen Akteure konnte sehr gut aufgezeigt werden, dass *Web Services* sehr einfach in ein *BPMS* zu integrieren sind, auch dass solche angeboten werden können. Obwohl aufgrund der zeitlichen Verzögerungen wegen der Ressourcenknappheit nicht alles einwandfrei umgesetzt werden konnte sind meiner Meinung nach die Vorteile eines *BPMS* und *SOA* sehr gut zur Geltung gekommen, vielleicht nicht im Bereich der kleinen Unternehmungen.

Auch die Feststellung, dass ein *BPMS* nicht für alle Unternehmungen vor allem im KMU Bereich von Vorteil ist hat sich sehr gut herauskristallisiert.

Mit jedem Schritt der Forschung für das *BPMS* von *webMethods* wuchs auch das Verständnis vom Geschäftsprozessemanagement und serviceorientierter Architekturen.

17 Schlusswort

Während der Arbeit konnten folgende Punkte erläutert werden:

- Integrationsmöglichkeiten in verschiedene IT-Infrastrukturen, mit oder ohne bestehende Applikationen
- Hoher Stellenwert von *Code Reusing*
- Sinn und Unsinn der Implementierung eines *BPMS* in einer *KMU*
- Verschmelzung und Trennung von *BPMS* und *SOA*

In dieser Arbeit wurde aufgezeigt wie anhand eines Szenarios ein *Business Process Management System (BPMS)* in bestehende IT-Architekturen integriert werden kann. Weiter wurde aufgezeigt, dass für die Erreichung dieses Ziel keine bestehende Software vorhanden sein muss, alles kann im *BPMS* entwickelt werden, eine Art jungfräulicher Ansatz.

Vor allem aber die Möglichkeit des Wiedergebrauchs von Software, insbesondere der Logik ist ein Hauptresultat der Arbeit.

Eine serviceorientierte Architektur (*SOA*) muss nicht zwangsläufig nur aus *Web Services* bestehen, es ist sehr gut ersichtlich dass zwischen Prozessen auch ganz andere Services zwischen geschaltet werden können.

Auch wurde ersichtlich, dass als Voraussetzung eines *BPMS* ein intaktes *Business Process Management* vorhanden sein sollte.

Die Verschmelzung zwischen einem *Business Process Management System* und einer serviceorientierten Architektur sind vielfach nicht zu differenzieren, ist auch nicht unbedingt nötig, da der Vorteil klar auf der Hand liegt. Dies hat sich auf oft während dieser Arbeit gezeigt.

18 Ehrenwörtliche Erklärung

Ich bestätige hiermit, dass ich die vorliegende Diplomarbeit alleine und nur mit den angegebenen Hilfsmitteln realisiert habe und dass ich ausschliesslich die erwähnten Quellen benutzt habe. Ohne Einverständnis des Leiters des Studienganges und des für die Diplomarbeit verantwortlichen Dozenten Herrn Bagnoud Laurent werde ich dieses Dokument an niemanden verteilen.

Siders, Dezember 2007

Der Autor, Schaller Josef

19 Abbildungsverzeichnis

ABBILDUNG 1 : ANBIETER BPMS SOFTWARE UND IHRE STELLUNG AUF DEM MARKT. QUELLE:	
THE FORRESTER WAVE: HUMAN-CENTRIC BPM FOR JAVA PLATFORMS, Q3 2007	10
ABBILDUNG 2 : INTERAKTION BROKER MIT INTEGRATIONSSERVERN.	13
ABBILDUNG 3 : INTERAKTION DES BROKERS MIT VERSCHIEDENEN KOMPONENTEN.....	14
QUELLE: WEBMETHODS_BROKER_ADMINISTRATOR'S_GUIDE_7_1.PDF	14
ABBILDUNG 4 : ADMINISTRATIONSOBERFLÄCHE INTEGRATIONSSERVER.	15
QUELLE: SCHALLER JOSEF	15
ABBILDUNG 5 : SERVICEVERARBEITUNG	16
QUELLE: SCHALLER JOSEF IN ANLEHNUNG AN WMUSERS WHAT IS WEBMETHODS V2.....	16
ABBILDUNG 6 : ARCHITEKTUR UND KOMPONENTEN.	16
QUELLE : TRADING NETWORKS CONCEPT GUIDE 7.1	16
ABBILDUNG 7 : LOGINSEITE FÜR ADMINISTRATOREN UND BENUTZER.....	18
QUELLE: SCHALLER JOSEF	18
ABBILDUNG 8 : ARCHITEKTUR MIT MONITOR.....	19
QUELLE MONITOR USER'S GUIDE 7.1	19
ABBILDUNG 9 : KOMPONENTEN VON OPTIMIZE FOR INFRASTRUCTURE.....	20
QUELLE: SCHALLER JOSEF	20
ABBILDUNG 10 : TYPISCHE OPTIMIZE INFRASTRUKTUR	21
QUELLE: OPTIMIZE FOR INFRASTRUCTURE ADMINISTRATOR'S GUIDE 7.1.....	21
ABBILDUNG 11 : AUFBAU ANALYTIC ENGINE.....	22
QUELLE: SCHALLER JOSEF	22
ABBILDUNG 12 : DESIGNER BENUTZEROBERFLÄCHE.....	25
QUELLE :SCHALLER JOSEF	25
ABBILDUNG 13 : DATENTYPEN.....	26
QUELLE: SCHALLER JOSEF	26
ABBILDUNG 14 : KONTROLLPALETTE FÜR TASKS IM DESIGNER 7.1.	27
QUELLE: SCHALLER JOSEF	27
ABBILDUNG 15 : EINFACHER PROZESS IM DESIGNER.	28
QUELLE: SCHALLER JOSEF	28
ABBILDUNG 16 : PROZESS MIT UNTERPROZESS.....	29
QUELLE: SCHALLER JOSEF	29
ABBILDUNG 17 : DEBUGGING IM DESIGNER.	30
QUELLE: SCHALLER JOSEF	30
ABBILDUNG 18 : OBERFLÄCHE DEVELOPER 7.1.	31
QUELLE: SCHALLER JOSEF	31
ABBILDUNG 19 : ADAPTER ANHAND DER DOKUMENTATION.	33
QUELLE: HTTP://ADVANTAGE.WEBMETHODS.COM	33
ABBILDUNG 20 : ARCHIVIERUNGSMÖGLICHKEIT AUF DEM INTEGRATIONSSERVER.	36
QUELLE: SCHALLER JOSEF	36
ABBILDUNG 21 : BENUTZERVERWALTUNG INTEGRATIONSSERVER.....	38
QUELLE: SCHALLER JOSEF	38
ABBILDUNG 22 : SUCHMÖGLICHKEIT NACH PROZESSEN IN MY WEBMETHODS.....	39
QUELLE: SCHALLER JOSEF	39
ABBILDUNG 23 : TASK VERWALTUNG FÜR EINZELNE BENUTZER.....	40
QUELLE: SCHALLER JOSEF	40
ABBILDUNG 24 : BESTELLPROZESS DES HÄNDLERS.	41
QUELLE: SCHALLER JOSEF	41
ABBILDUNG 25 : DOKUMENTVARIABLE ALERT.....	42
QUELLE: SCHALLER JOSEF	42

ABBILDUNG 26 : EIGENSCHAFTEN RECEIVE STEP.	43
QUELLE: SCHALLER JOSEF	43
ABBILDUNG 27 : TRANSITIONS RECEIVE STEP MIT DER KONDITION FÜR ALERT VARIABLE	43
QUELLE: SCHALLER JOSEF	43
ABBILDUNG 28 : BUSINESS DATA FÜR DEN TASK.....	43
QUELLE: SCHALLER JOSEF	43
ABBILDUNG 29 : FLOW SERVICE UND PIPELINE.	44
QUELLE: SCHALLER JOSEF	44
ABBILDUNG 30 : FLOW STEP IM RETORDERNOTIFICATION SERVICE.	45
QUELLE: SCHALLER JOSEF	45
ABBILDUNG 31 : AKTIVIERUNG VON PROZESSEN FÜR AUSFÜHRUNG UND ANALYSE.	46
QUELLE: SCHALLER JOSEF	46
ABBILDUNG 32 : AUSFÜHRUNG DES DOKUMENTS IM DEVELOPER 7.1.....	46
QUELLE: SCHALLER JOSEF	46
ABBILDUNG 33 : PUBLIKATION AUF DEM LOKALEN IS.....	46
QUELLE: SCHALLER JOSEF	46
ABBILDUNG 34 : TASK LISTE UND PROZESS INSTANZEN IN MY WEBMETHODS.	47
QUELLE: SCHALLER JOSEF	47
ABBILDUNG 35 : PROZESSDIAGRAMM VOR KOMPLETTIERUNG DES TASKS HINPUT.	47
QUELLE: SCHALLER JOSEF	47
ABBILDUNG 36 : TASK DETAILS. EINGABEMASKE FÜR DIE BESTELLUNG.	48
QUELLE: SCHALLER JOSEF	48
ABBILDUNG 37 : PROZESSDIAGRAMM UND AUSGEFÜHRTE SCHRITTE.....	48
QUELLE: SCHALLER JOSEF	48
ABBILDUNG 38: LOGISTIKKETTE.....	49
QUELLE : SCHALLER JOSEF	49
ABBILDUNG 39 : ARCHITEKTUR HÄNDLER.	51
QUELLE: SCHALLER JOSEF	51
ABBILDUNG 40 : PROZESSMODELL DES HÄNDLERS.	52
QUELLE: SCHALLER JOSEF	52
ABBILDUNG 41 : DOKUMENT MIT NÖTIGEN VARIABLEN FÜR DIE BESTELLUNG.....	53
QUELLE: SCHALLER JOSEF	53
ABBILDUNG 42 : KONDITIONEN IM SERVICE NEWORDERCHECK.	53
QUELLE: SCHALLER JOSEF	53
ABBILDUNG 43 : PAKTE IM DEVELOPER 7.1	54
QUELLE: SCHALLER JOSEF	54
ABBILDUNG 44 : EINGABEMASKE FÜR INPUT IM DEVELOPER 7.1.....	55
QUELLE: SCHALLER JOSEF	55
ABBILDUNG 46 : EINZELNE SERVICES IM SERVICE INSERTËNCAVEUR	56
QUELLE: SCHALLER JOSEF	56
ABBILDUNG 47 : ARCHITEKTUR ENCAVEUR.....	58
QUELLE: SCHALLER JOSEF	58
ABBILDUNG 48 : AUFLISTUNG DER BESTELLUNGEN MIT HILFE VON PHP.	59
QUELLE: SCHALLER JOSEF	59
CODE 1 : FUNKTION ADDNEWORDER IN DER LOGIKDATEI LOGIC.PHP.....	59
QUELLE: SCHALLER JOSEF	59
CODE 2 : DATEI SERVERËNCAVEUR.PHP.....	61
QUELLE: SCHALLER JOSEF	61
ABBILDUNG 49 : PROZESSMODELL DES ENCAVEUR.	62
QUELLE: SCHALLER JOSEF	62
ABBILDUNG 50 : PIPELINE INPUT.....	63
QUELLE: SCHALLER JOSEF	63
ABBILDUNG 51 : PIPLINE OUTPUT.	63

QUELLE: SCHALLER JOSEF	63
ABBILDUNG 52 : PROZESSMODELL DES TRANSPORTEURS.....	65
QUELLE: SCHALLER JOSEF	65
ABBILDUNG 53 : SERVICE INPUT PARAMETER.	67
QUELLE: SCHALLER JOSEF	67
ABBILDUNG 54 : JAVA SERVICE ZUM TESTEN DER SLA.....	68
QUELLE: SCHALLER JOSEF	68
ABBILDUNG 55 : FLOW SERVICE MIT BRANCH STEP.....	68
QUELLE: SCHALLER JOSEF	68
ABBILDUNG 56 : PIPELINE IM SERVICE INSERTNEWORDER.....	69
QUELLE: SCHALLER JOSEF	69
ABBILDUNG 57 : SYSTEMEIGENSCHAFTEN ARBEITSCOMPUTER.	71
QUELLE: SCHALLER JOSEF	71
ABBILDUNG 58 : KOMPONENTENAUSWAHL, KEINE ADAPTER VORHANDEN.....	73
QUELLE: SCHALLER JOSEF	73
ABBILDUNG 59 : KOMPONENTENAUSWAHL, ADAPTER VORHANDEN.....	74
QUELLE: CARLSON MARK, HTTP://WWW.WMUSERS.COM	74

20 Tabellenverzeichnis

TABELLE 1 : KOMPONENTENBESCHREIBUNG OPTIMIZE FOR INFRASTRUCTURE.	20
TABELLE 2: BESCHREIBUNG DER SCHICHTEN DER OPTIMIZE INFRASTRUKTUR IN ABBILDUNG 10	22
TABELLE 3: ANALYTIC ENGINE KOMPONENTEN UND IHRE AUFGABEN.	23
TABELLE 4: WERKZEUGE WEBMETHODS 7.1	24
TABELLE 5: STANDARDS IN GEAR 6.5 STANDARDS AND PROTOCOLS BEST PRACTICES.	33
TABELLE 6: SYSTEMVORAUSSETZUNGEN FÜR WEBMETHODS KOMPONENTEN.	34
TABELLE 7: FORTSETZUNG TABELLE 6.	35
TABELLE 8: UNTERSTÜTZTE DATENBANKSYSTEME.	35
TABELLE 9: PFAD DER WICHTIGEN DATEIEN FÜR REGELMÄSSIGE BACKUPS.	37
TABELLE 10: PROZESSEINFLÜSSE.	39
TABELLE 11: PROZESSSCHRITTE UND BESCHREIBUNG.	66

21 Codeverzeichnis

CODE 1 : FUNKTION ADDNEWORDER IN DER LOGIKDATEI LOGIC.PHP.....	59
CODE 2 : DATEI SERVERENCAVEUR.PHP.....	61