HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

AdNovum

Enterprise Service Bus zur firmeninternen Anwendungsintegration – Systematische Evaluation und Einführungsunterstützung

Studienarbeit

Technischer Bericht

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2014

Autoren: Corina Honegger & Emre Avsar
Betreuer: Prof. Dr. Olaf Zimmermann
Projektpartner: AdNovum Informatik AG, Zürich

18. Dezember 14

Aufgabenstellung

Aufgabenstellung Studienarbeit
Emre Avsar, Corina Honegger

Enterprise Service Bus zur firmeninternen Anwendungsintegration – Systematische Evaluation und Einführungsunterstützung

1. Auftraggeber und Betreuer

Diese Studienarbeit wird in Zusammenarbeit mit der Firma AdNovum durchgeführt.

Betreuer HSR:

Prof. Dr. Olaf Zimmermann, Institut für Software, ozimmerm@hsr.ch

Ansprechpartner Auftraggeber:

Der Hauptansprechpartner bei AdNovum ist: Christian Fritz, Head of Internal Applications.

Der Industriepartner hat die Hinweise für externe Auftraggeber http://www.hsr.ch/Hinweise-fuer-externe-Auftragg.7263.0.html zur Kenntnis genommen und ist insbesondere mit der Gebührenregelung einverstanden (ein unterschriebenes Formular dazu liegt bereits vor).

2. Ausgangslage

AdNovum Informatik AG hat Interesse, einen unternehmensweiten Enterprise Service Bus (ESB) zu etablieren. Mit einem derartigen ESB können Services einfacher gefunden und genutzt werden. Zusätzlich besteht seitens AdNovum Informatik AG Interesse, die Erkenntnisse und Erfahrungen bei der Einführung eines ESBs für spätere Consulting-Projekte zu nutzen.

3. Ziele der Arbeit und Liefergegenstände

Thematik der Arbeit (Arbeitsaufträge):

a) ESB-Produkte vergleichen: Auswahl und Analyse von zwei bestehenden ESB-Produkten und deren Kategorisierung, so dass die Geschäftsleitung eine Übersicht der Produkte und ihrer Fähigkeiten erhält. Dabei soll die Implementierung an Hand des spezifischen Use-Cases analysiert werden: Ein internes Zeiterfassungs-Tool benötigt das Datum des Stellenantritts (und ev. Austritts) eines Mitarbeiters. Diese Information ist im HR-Management-Tool gespeichert und soll via ESB übertragen werden.

b) Spezifikation und Analyse: Leistungsumfangs und der Konfigurationsmöglichkeiten der ausgewählten ESB-Produkte (z.B. Discovery, Routing, Data Mappings, Message Transport) dokumentieren.

c) Vergleich und Empfehlung: Produkte an Hand eines Kriterienkatalogs gegenüberstellen und idealen Anwendungsfall bestimmen.

d) Checkliste für allgemeine ESB-Readiness der IT-Landschaft einer Unternehmung erstellen.

4. Unterstützung

Die erwartete und effektiv erhaltene Unterstützung wird durch die Studenten in Sitzungsprotokollen definiert und im SA-Bericht dokumentiert.

5. Zur Durchführung

Mit dem HSR-Betreuer finden in der Regel wöchentlich Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf zu veranlassen. Besprechungen mit dem Auftraggeber werden nach Bedarf durchgeführt.

Alle Besprechungen, bei denen eine Vorbereitung durch den Betreuer nötig ist, sind von den Studenten mit einer Traktandenliste vorzubereiten. Beschlüsse sind in einem Protokoll zu dokumentieren.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. Arbeitszeiten sind zu dokumentieren.

Die Spezifikation der Anforderungen geschieht durch die Studenten in Absprache mit dem Betreuer. Bei Disputen entscheidet der Betreuer in Rücksprache mit den Studenten über die definitiv für die Bachelorarbeit relevanten Anforderungen.

Vorstudie, Anforderungsdokumentation und Architekturdokumentation sollten im Laufe des Projektes mittels Milestone mit dem Auftraggebern und dem Betreuer in einem stabilen Zustand abgenommen werden. Zu den abgegebenen Arbeitsresultaten wird ein vorläufiges Feedback abgegeben. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

Die Rechte an den Ergebnissen der Bachelorarbeit werden in einer separaten Vereinbarung definiert (Standard-Regelung lt. Sitzung mit AdNovum am 15.9.2014).

6. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in zwei Exemplaren abzugeben. Bei der Projektdokumentation sind die Anleitungen des Studienganges inklusive Anhängen zu beachten, siehe https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html. Zudem ist eine kurze Projektresultatdokumentation für das Wiki von Prof. Zimmermann erwünscht (ggfs. kurzes Video).

7. Termine

Siehe auch HSR-Webseiten, https://www.hsr.ch/Termine-Bachelor-und-Studiena.5142.0.html

15.09.14	Beginn der Studienarbeit, Ausgabe der Aufgabenstellung durch den Betreuer
15.12.14	Die Studierenden senden folgende Dokumente der Arbeit per Email zur Prüfung an ihre Betreuer: - Kurzfassung - A0-Poster Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
19.12.14	Die Studierenden senden die vom Betreuer abgenommene und freigegebene Kurzfassung als Word-Dokument an das Studiengangsekretariat (cfurrer(at)hsr.ch).
19.12.14	Abgabe des Berichtes an den Betreuer bis 17.00 Uhr.

Allfällige weitere Termine sind am Sekretariat der Abteilung Informatik zu erfragen und sollten entsprechend in einem Sitzungsprotokoll dokumentiert werden.

8. Beurteilung

Eine erfolgreiche Studienarbeit zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS-Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert.

Siehe http://studien.hsr.ch/allModules/23498_M_SAI.html für die Modulbeschreibung der Studienarbeiten.

Für die Beurteilung ist der HSR-Betreuer verantwortlich unter Einbezug des Feedbacks des Projektpartners.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte (Abstract, Management Summary, technische u. persönliche Berichte) sowie Gliederung, Darstellung und Sprache der gesamten Dokumentation.	1/5
3. Inhalt*)	3/5

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit festgelegt.

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Studienarbeiten.

Rapperswil, den 03. 10. 2014

Prof. Dr. Olaf Zimmermann
Institut für Software
Hochschule für Technik Rapperswil

Abstract

Seit über zehn Jahren kursiert das Architekturmuster Enterprise Service Bus (ESB) nun schon in der IT-Welt, speziell im Bereich Integration von Unternehmensanwendungen. In der Praxis haben sich ESBs vor allem in komplexen Service Oriented Architecture (SOA)-Umgebungen durchgesetzt. Dennoch herrscht bis heute eine gewisse Unwissenheit über den Sinn und Zweck von ESBs.

Mit dieser Arbeit wollen wir Klarheit über den Mythos ESB schaffen und eine Hilfestellung für Evaluationen bieten. Die Bewertung von ESBs und die Klärung der Grundsatzfrage, ob deren Einsatz sinnvoll ist oder nicht, stehen dabei im Mittelpunkt.

Hierfür erstellen wir in einem ersten Schritt mit den Use Case-Anforderungen unseres Industriepartners und unserem bisherigen Wissen einen Kriterienkatalog. Als praktische Anwendung und Verifikation folgen anschliessend die Resultate der Bewertung von zwei ESB-Produkten. Im Kontext dieser Produkte setzen wir dann den Use Case um. Es gibt diverse Anbieter sowohl im Open-Source- wie auch im kommerziellen Bereich. Wir haben aus beiden Kategorien jeweils ein Produkt evaluiert (WSO2 Enterprise Service Bus und Oracle Service Bus).

Das Ergebnis unserer Arbeit ist ein in fünf Kategorien gegliederter Kriterienkatalog und eine ESB-Readiness-Checkliste. Diese beiden Arbeitsmittel ermöglichen eine strategische Bewertung von ESB-Produkten und erleichtern die Beurteilung, ob die Einführung eines solchen ökonomisch ist. Ausserdem werden in unserer Arbeit die beiden Produkte von WSO2 und Oracle detailliert analysiert und bewertet.

Bei unserer Arbeit haben wir festgestellt, dass sich die auf dem Markt erhältlichen bekannten ESB-Produkte in ihrer Funktionalität und Eignung stark unterscheiden. Wir zeigen auf, dass ein ESB nicht zwingend ein nicht anpassbarer Monolith mit Hunderten von Funktionen sein muss. Bevor man mit der Evaluation beginnt, sollte man deshalb abklären, welche Applikationen und Protokolle über den ESB realisiert werden sollen.

Vorwort und Danksagung

Die vorliegende Studienarbeit ist im Rahmen des Bachelor-Studiengangs an der Hochschule für Technik in Rapperswil entstanden.

Unser erster Dank geht an Herrn **Prof. Dr. Olaf Zimmermann** für die Betreuung der Studienarbeit.

Ein besonderer Dank richtet sich an **Andrea Duttwiler und Verena Müdespacher** von der AdNovum Informatik AG für die Verifikation, grammatische Korrektur sowie die lyrisch konstruktiven Vorschläge, die wir unter anderem auch in die Arbeit einfliessen liessen.

Inhaltsverzeichnis

1 Management Summary

1.1 Ausgangslage

Seit 2002 hat der Begriff Enterprise Service Bus (ESB) sich in der IT-Welt etabliert und spaltet die Fachspezialisten in zwei Lager. Auf der einen Seite stehen die Befürworter, während sich auf der anderen die Gegner aufgestellt haben. Es gibt keine generelle Antwort auf die Frage, ob ein ESB nun gut oder schlecht ist. Fakt ist aber, dass viel Unwissenheit über Definition und Eigenschaft des ESBs herrscht. Unternehmen sind sich unsicher, ob ein ESB eine Bereicherung oder Belastung ist. Oftmals fehlen ihnen auch die Grundlagen für die Beurteilung und Evaluierung der auf dem Markt erhältlichen Produkte.

Ein ESB wird hauptsächlich in Service Oriented Architecture (SOA)-Landschaften eingesetzt, um mehrere Applikationen über einen zentralen Bus miteinander kommunizieren zu lassen. Dabei liegt die dazu notwendige Logik anstatt bei den Anwendungen im ESB. Er regelt das Routing und Mapping, während er gleichzeitig für die Sicherheit des Nachrichtentransports und Umsetzung von Quality of Service (QoS) Anforderungen sorgt.

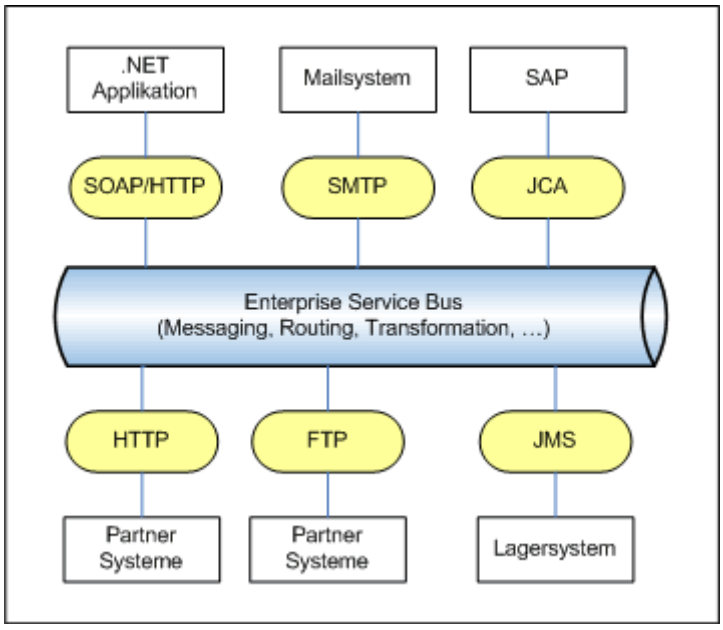


Abbildung 1 Logische Ansicht eines ESBs [1]

Unser Industriepartner möchte herausfinden, wie die Marktsituation der ESBs aussieht und welche Produkte sich für den Einsatz in der Firma oder bei Kunden eignen. Dazu sollen zwei Produkte im Kontext eines Use Cases verglichen und bewertet werden. Aus den gemachten Erfahrungen sollen anschliessend Konsequenzen und Anforderungen für die Einführung eines ESB erarbeitet werden.

Im Rahmen des Use Cases soll das interne Zeiterfassungstool des Industriepartners diverse Arbeitsvertragsdaten aus den beiden Personalverwaltungssystemen abfragen. Der ESB übernimmt dabei Routing und Filterung der Daten, sowie die Konvertierung in das JSON Format.

1.2 Vorgehen und Technologien

Mit Hilfe des Internets und Anmerkungen unseres Betreuers verschafften wir uns zuerst einen Überblick der aktuellen ESB Produkte. Dazu informierten wird uns über zwölf Produkte (kategorisiert nach Open Source und kommerziell). Daraus wählten wir jeweils eines für die Bewertung aus. Wir entschieden und für den Open Source ESB von WSO2. Bei der Wahl des kommerziellen Produktes

wurde aufgrund der Partnerschaft mit Oracle der Oracle Service Bus (OSB) bevorzugt und ausgewählt.

Abgeleitet aus den funktionalen und nichtfunktionalen Anforderungen des Use Cases, Recherchen aus dem Internet und aus fachspezifischer Literatur erarbeiteten wir einen Kriterienkatalog für die Bewertung von ESBs. Diesen verifizierten wir anschliessend durch die praktische Anwendung beim WSO2 ESB und beim OSB.

Die Umsetzung des Use Cases in den beiden ESBs gestaltete sich zu Beginn jeweils zeitaufwendiger als geplant, da uns die Erfahrung mit den Produkten fehlte. Dies konnten wir nachholen, indem wir diverse Samples der Hersteller durchgearbeitet haben. Dadurch waren wir schlussendlich in der Lage, den Use Case in beiden Produkten zu realisieren.
Da das Zeiterfassungstool eine sehr komplexe Architektur hat und dessen Anpassung den Umfang unsere SA gesprengt hätte, entwickelten wir einen Simulator, der nur die notwendigen Funktionen beinhaltet.

1.3 Ergebnisse

Unsere Bewertungen haben gezeigt, dass beide ESBs ihre eigenen Stärken und Schwächen haben.

Der WSO2 ESB zeichnet sich vor allem durch seine Leichtgewichtigkeit, Intuivität und gute Unterstützung beim Anschluss von Webapplikationen aus. Ausserdem existieren in einer offiziellen Guide des Herstellers Lösungsvorschläge für die Umsetzung aller Enterprise Integration Patterns (EIP). Negativ fällt lediglich die fehlende Funktion zur Versionierung von Konfigurationseinstellung und Komponenten auf.
Die Filterung der Daten in unserem Use Case realisierten wir mit einer XSLT-Transformation. Für die Translation von XML zu JSON existiert ein integrierter Message Formatter, welcher sich mittels eines einfachen Parameters aufrufen lässt.

Die Stärken des OSB zeigen sich vor allem beim Einsatz in grösseren Systemlandschaften mit vielen unterschiedlichen Kommunikationstechnologien. Er besitzt ein eigenes Transport Framework für die Erstellung von Custom Adapters und besitzt viele wichtige Management Funktionen. Dazu gehören unter anderem der implementierte Cluster und Load Balancing Service sowie die umfangreichen Alerting- und Monitoring-Funktionen. Nur die teuren Anschaffungskosten sowie die Tatsache, dass Oracle Produkte-Know-How zwingend notwendig ist, fallen wirklich negativ auf.
Gefiltert haben wir die Daten für unseren Use Case ebenfalls mit einer XSLT-Transformation. Die Translation zu JSON mussten wir jedoch selber programmieren und mittels Java Callout während der Nachrichtenverarbeitung im ESB aufrufen. Es existiert keine integrierte Funktionalität für diesen Schritt.

Um unsere gesamten Erfahrungen mit ESBs zusammenzufassen, erarbeiteten wir zum Abschluss eine ESB Readiness-Checkliste. Sie beinhaltet wichtige Fragen die sich Unternehmen stellen sollten, um zu entscheiden, ob die Einführung eines ESB sinnvoll und rentabel wäre. Die Checkliste ist gegliedert in drei Kategorien:

- ESB-Bedarf (Müssen) – Welche Voraussetzungen müssen erfüllt sein, dass eine ESB-Einführung in Frage kommt?
- ESB-Kompetenzen (Können) – Welche Kompetenzen werden für die Einführung und den Unterhalt des ESBs gebraucht?
- ESB-No-Gos (Nicht-Dürfen) – Welche Fakten weisen darauf hin, dass die Einführung eines ESBs unterlassen werden sollte?

1.4 Ausblick

Mit unseren erarbeiteten und geprüften Kriterien und der ESB-Readiness-Checkliste sollten auch Personen ohne ESB-Erfahrung in der Lage sein, ein solches Produkt zu bewerten und eine erste Aussage über die Wirtschaftlichkeit einer Einführung zu machen.

Es existieren jedoch viele Produkte auf dem Markt, die sich auf eine spezifische Technologie oder Sparte spezialisiert haben. Beispielsweise der SAP NetWeaver PI oder der Microsoft BizTalk Server. Betrachtet man solche spezialisierten Produkte, verändern sich auch die Anforderungen an die Schnittstellen und Nachrichtenverarbeitung des ESBs. Der Kriterienkatalog könnte in einer Folgearbeit so erweitert werden, dass diese neuen Anforderungen in die Bewertung einbezogen werden.

Auch die Readiness-Checkliste ist aktuell sehr oberflächlich gehalten, da sie auf möglichst alle Unternehmen anwendbar sein soll. Jedoch hat ein Technologiekonzern sicherlich eine andere Ausgangslage als vergleichsweise eine Schule, obwohl beide normalerweise viele unterschiedliche Applikationen in ihrer IT-Systemlandschaft betreiben. Ausgehend von den unterschiedlichen Ausgangslagen könnte die Checkliste so angepasst werden, dass sich diese auch in den Kompetenzen und dem Bedarf wiederspiegeln.

2 Problemanalyse

In diesem Kapitel wird das Ziel unserer Studienarbeit (SA) genauer vorgestellt und die originale Aufgabenstellung ergänzt und abgerundet.

2.1 Definitionen

In diesem Bericht ist der Begriff ESB das Kernthema. Dazu existieren zahlreiche Definitionen. Um eine klare Ausgangslage zu schaffen, ist hier die für die Arbeit allgemein gültige Definition eines ESB beschrieben.

2.1.1 Wozu braucht es einen ESB?

In vielen Unternehmen herrscht eine grosse Vielfalt an SOA-Applikationen. Typischerweise müssen diese untereinander kommunizieren. Dies verursacht einen regen Nachrichtenfluss, welcher historisch bedingt oftmals durch Point-to-Point-Schnittstellen realisiert ist. Diese Art von Schnittstellen haben jedoch enorme Nachteile, da sie physisch an einen Endpunkt und dessen Konfiguration gebunden sind.

Hier setzt nun ein ESB an [2].

2.1.2 Definition eines Enterprise Service Bus

Lange hat man geglaubt, die Kommunikation zwischen Applikationen mit der Enterprise Application Integration (EAI) Architektur „Hub and Spoke“ endgültig gelöst zu haben. Tatsache war jedoch, dass die meisten Projekte scheiterten. Als Antwort auf diese Problematik erschienen um das Jahr 2002 herum einige Softwarelösungen auf dem Markt, welche die Projekte zum Erfolg führen sollten.
Die Firma Gartner fasste diese neuartigen Anwendungen in ihren Reports in einem Begriff zusammen: Enterprise Service Bus. Damit trugen sie massgeblich zur Publikation und Prägung des Begriffs bei [1].

Ein Enterprise Service Bus erlaubt, den Nachrichtenaustausch zwischen heterogenen Systemen zentral zu realisieren und zu verwalten. Typischerweise sind dies Applikationen in einer serviceorientierten Landschaft. Bei der Implementierung eines ESB werden die Systeme über Adapter angeschlossen. Der Nachrichtenfluss wird innerhalb des ESB abgebildet und geroutet. Das Abbilden beinhaltet auch Filterung, Data Translation und Transformation. Diese Prozesse werden dabei durch die Integration von Standards und EIPs unterstützt.

David A. Chappell[1] hat den Begriff in seinem weltbekannten ESB Buch folgendermassen zusammengefasst:

> *“The ESB provides a highly distributed, event-driven Service Oriented Architecture (SOA) that combines Message Oriented Middleware (MOM), web services, intelligent routing based on content, and XML data transformation. ESBs are used to solve integration challenges in many unique ways […]. “*
>
> *[3, p. xi]*

Zitat 1 Definition eines ESBs nach David A. Chappell

[1] David A. Chappell hat 2004 das erste Buch über ESBs veröffentlich und damit massgeblich zur Prägung und Bekanntmachung des Begriffs beigetragen.

2.1.3 Data Translation vs. Data Transformation

In der Fachsprache gibt es oft Vermischungen der Begriffe und sie werden nicht einheitlich verwendet. In den folgenden Abschnitten ist daher unter Data Translation die Umwandlung in ein anderes Message-Format zu verstehen, ohne dabei die Rohdaten zu verändern. Als Data Transformation wird die Daten-Manipulation bezeichnet, worunter zum Beispiel auch Grössenkonvertierungen fallen.

Kodierung in ein anderes Zeichenformat (z.B. von ISO-8859-1 zu UTF-8) wird in diesem Sinne als Data Transformation angesehen. Der Grund ist, dass das Format gleich bleibt und die Daten innerhalb des Formats verändert werden (übersetzt werden in ein anderes Zeichenformat).

2.1.4 Apache License Version 2.0

Open-Source-Produkte werden generell mit GNU General Public License [4], MIT License [5] oder Apache License Version 2.0 [6] veröffentlicht. Das ausgewählte Open-Source-Produkt WSO2 Enterprise Service Bus verwendet die Apache License in der zweiten Version (Apache License Version 2.0). Die Konsequenzen beim Gebrauch dieses Produktes sind:

- **Erforderlich**
  Es muss eine Kopie der Lizenz sowie die Copyright-Erwähnung im Endprodukt enthalten sein.
- **Erlaubt**
  Das Endprodukt darf zu kommerziellen Zwecken verwendet werden. Vertrieb und Veränderungen an der Software sind erlaubt.
- **Verboten**
  Bei Schäden durch Nutzung der Software können die Urheber des Produktes nicht verantwortlich gemacht werden. Ausserdem ist es verboten, die Namen (inkl. Logos und Marken) der Mitwirkenden zu nutzen.

2.1.5 Microservices - Yet Another Acronym?

Martin Fowler beschreibt auf seiner Webseite [7] Microservices als einen Architekturstil, in welchem eine Applikation aus kleinen Services besteht. Diese Services laufen in eigenen Prozessen und kommunizieren mit leichtgewichtigen Mechanismen (meist HTTP). Diese Services werden als Business-Logic-Komponenten gebaut und sollen unabhängig von der Applikation verteilbar designed werden. Im Gegensatz zu traditionellen Applikationen sind Microservices also, wie der Name schon verrät, kleinere Teile einer Applikation und nicht die Applikation selbst.

Der Vorteil besteht darin, dass einzelne Komponenten stark skalierbar sind. So ist es möglich ein Microservice mehrfach zu verteilen und somit einzelne Teile einer Applikation unabhängig skalieren zu können. Ausserdem ist das Testen einfacher, so sagt es Martin Fowler.

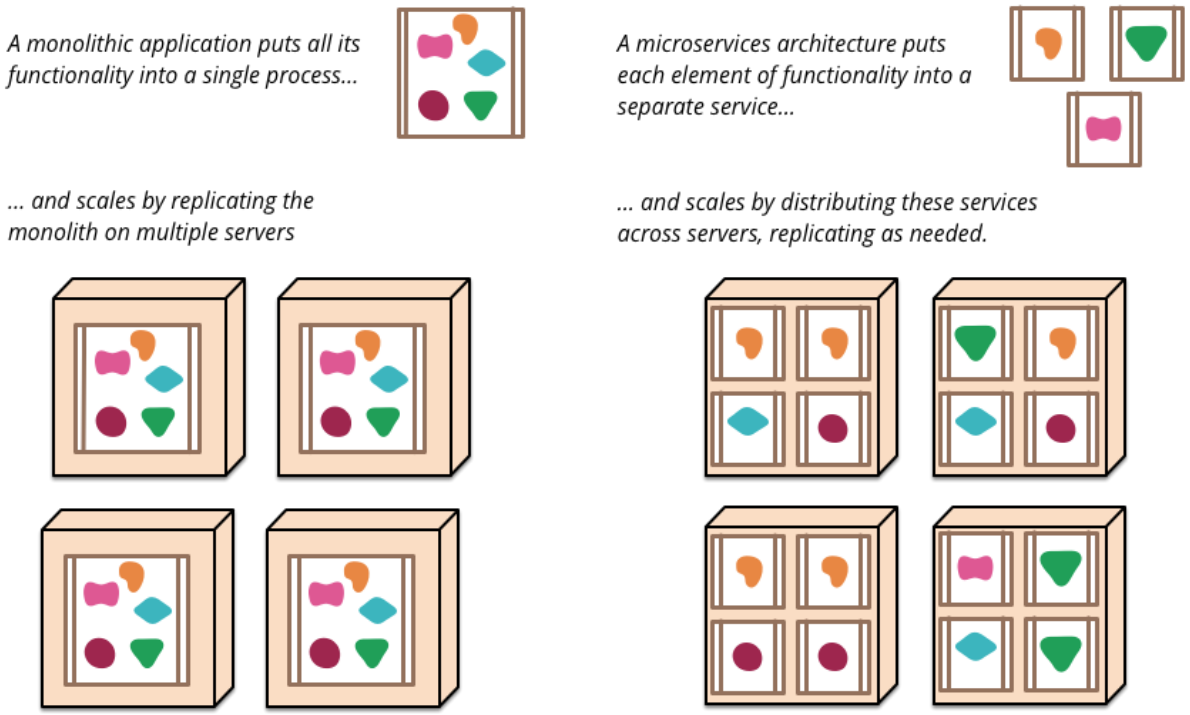


Abbildung 2 Vergleich zwischen traditionelle Applikationen (links) und Microservices (rechts) [7]

Ein weiterer Vorteil bieten Microservices durch die Unabhängigkeit einer Programmiersprache. Einzelne Microservices können dementsprechend in verschiedenen Programmiersprachen implementiert werden, da sie nur eine Schnittstelle als Service anbieten und unabhängig von einem Applikationskontext aufgerufen werden können.

Die interessante Frage ist nun, wie sich Microservices von ESBs unterscheiden. Im Grunde ist ein ESB eine zentrale Schnittstelle zu Services. Diese können im Idealfall die gleichen Charakteristiken wie Microservices aufweisen. Denn sie dienen ebenfalls als Services, welche über einen leichtgewichtigen Kommunikationsweg (z.B. HTTP) aufgerufen werden können. Auch können solche Services in verschiedenen Programmiersprachen entwickelt werden.

Wir sehen den Unterschied zwischen Microservices und ESBs jedoch im Anwendungsfall. Dieser Unterscheidet sich wie folgt:

- Microservices sind kleine Services, welche zusammen eine Applikation komplettieren.
- ESBs hingegen können Schnittstellen von Applikationen vereinheitlichen, bzw. durch zusätzliche Funktionen (z.B. Message Content Transformation oder Message Translation) die Aufrufe so verändern, dass sie für den Aufrufer ideal sind.
  Für den Service Konsument ist also ein Aufruf einfacher zu finden und zu benutzen.
- ESBs verändern die Architektur von Applikationen nicht und können nicht wie Microservices einzelne Services skalierbar machen. Jedoch können Aufrufe im ESB verteilt und gegebenenfalls zwischengespeichert werden, um diese später abzuarbeiten.
- Microservices alleine verfügen über keine zusätzlichen Funktionen wie sie der ESB hat. So ist es auch nicht möglich, Microservices mit in einer Registry nachzuschauen.

Somit sehen wir Microservices als ein Architekturstil für den Bau von Applikationen und ESBs als ein Integrationstool von Services einer Applikation.

2.2 Anforderungsanalyse

In diesem Kapitel wird unsere detaillierte Analyse der Aufgabenstellung zusammengefasst. Dazu führten wir mehrfach Rücksprache mit unserem Betreuer und dem Industriepartner um sicherzustellen, dass wir ein gemeinsames Verständnis der konkreten Aufgabe haben. Nebst den nachfolgenden Abschnitten erstellten wir daraus auch die Projektplanung gemäss Kapitel **Error! Reference source not found.**.

2.2.1 Projektziele

Aus der Aufgabenstellung haben wir die nachfolgenden Ziele abgeleitet, welche sich in drei Bereiche gruppieren lassen.

2.2.1.1 Vergleich von zwei ESB-Produkten

Es müssen zwei Produkte verglichen, getestet und bewertet werden. Vorzugsweise ist eines davon kommerziell und das andere Open Source. Daraus soll eine Empfehlung für geeignete Anwendungsfälle erstellt werden.

2.2.1.2 Anwendungskontext – Use Case

Als Kontext dient der Use Case „Arbeitszeit_Übermittlung“. Die zu evaluierenden ESB-Produkte werden mit diesem konkreten Use Case geprüft.

2.2.1.3 Checkliste für ESB Readiness

Im Rahmen der Projektarbeit soll eine Checkliste entwickelt werden. Diese kann später dazu benutzt werden, eine Systemumgebung auf eine ESB-Einführung vorzubereiten. Zudem soll sie bei der Beurteilung helfen, ob ein ESB einsetzbar und überhaupt sinnvoll wäre.

2.2.2 Use Case

Das Zeiterfassungstool „Instrumentum Temporis Computando“, kurz ITC, möchte den aktuellen Status (Arbeitsvertrag-Start und -Ende) der Verträge einzelner Mitarbeiter aus den HR-Management-Systemen Umantis und Abacus erhalten. Diese Daten werden benötigt, um das restliche Ferienguthaben von Mitarbeitern, welche im aktuellen Jahr angefangen haben oder aufhören werden, zu berechnen.

Der ESB dient als Koordinator und verwaltet die Anfragen so, dass die Anfrage zum richtigen HR-Management-Tool weitergeleitet wird, und ist auch dafür verantwortlich, dass die Daten im richtigen Format gesendet werden.

Der ESB soll somit aufgrund der Anfrage von ITC auswählen können, zu welchem System er den Request weiterleiten sollte, und erkennen, welche zusätzlichen Informationen oder Formatänderungen benötigt werden, um diese gleich vorzunehmen.

Das Zeiterfassungstool erhält dann die neusten Vertragsdaten, ohne die HR-Management-Tools zu kennen oder zu entscheiden müssen, in welchem System die Daten enthalten sind.

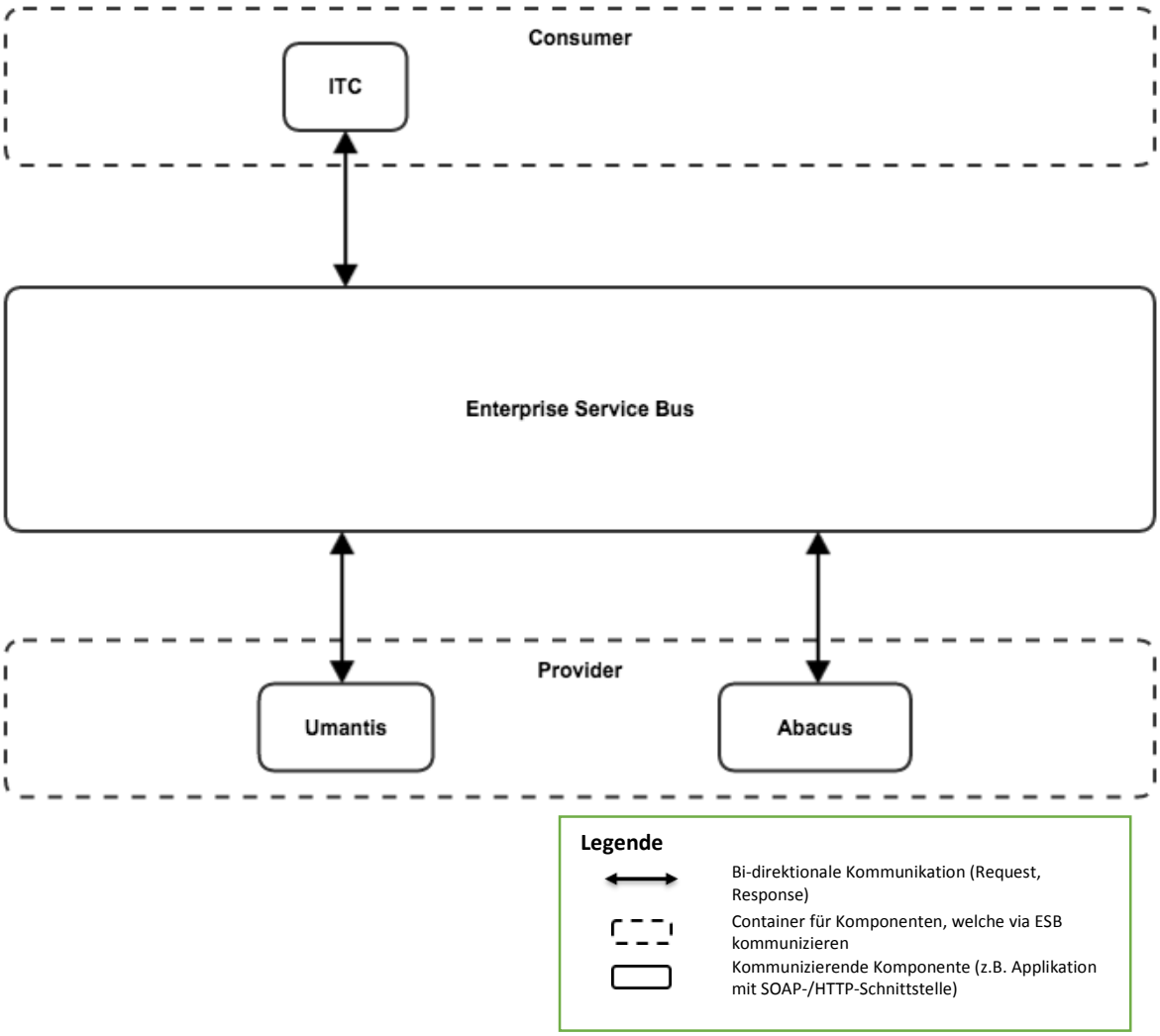


Abbildung 3 Zusammenspiel der Applikationen mit Hilfe eines ESB

2.2.2.1 Fully Dressed

Um sicherzustellen, dass wir den Use Case für den Anwendungskontext richtig verstanden haben, wird er nachfolgend noch im „Fully Dressed"-Format dargestellt.

ID	**UC-1**
Title	Vertragsdaten in Zeiterfassungstool importieren
Actors	Zeiterfassungstool ITC Koordinator ESB HR-Management-Tool Umantis HR-Management-Tool Abacus
Preconditions	Es bestehen Vertragsdaten zum Mitarbeiter in einem der beiden HR-Systeme sowie im ITC.
Postconditions	Das Zeiterfassungstool hat die gewünschten Vertragsdaten erhalten und kann mit diesen Daten weiterarbeiten.
Main Success Scenario	1. ITC fordert aktuelle Vertragsdaten an, indem es einen Request auf den ESB stellt 2. ESB entscheidet, welche Routen er benutzen muss 3. ESB sucht das richtige Mapping des Formats (z.B. REST over HTTP zu SOAP over HTTP) und leitet den Request im neuen Format weiter 4. HR-Management-System sucht intern nach Vertragsdaten und antwortet dem ESB mit diesen Daten 5. ESB sucht das richtige Mapping des Formats als Antwort und leitet diese an das Zeiterfassungstool weiter 6. Zeiterfassungstool zeigt Vertragsdaten an
Extensions	4. Keine Daten zum Vertrag gefunden: Es wird ein leerer Datensatz zurückgesendet
Frequency of Use	Mehrmals täglich
Priority	P1 – Maximum

Tabelle 1 Use Case "Fully Dressed"

2.2.2.2 Sequenzdiagramm

Im unten abgebildeten Sequenzdiagramm haben wir die Abläufe des Use Cases visuell dargestellt. Der Benutzer der ITC Applikation startet mit generateReport() den Use Case. Um einen Report generieren zu können, muss ITC nun die Daten mit den HR-Systemen synchronisieren und erstellt eine Anfrage beim ESB. Der ESB entscheidet welches Backend angesprochen werden soll und formatiert die Nachrichten so, dass sie dem ausgewählten HR-System passen. Dann wird das Backend aufgerufen. Die Antwort wird wiederrum so umformatiert, sodass ITC die Nachrichten entgegennehmen kann. ITC kann dann mit den Vertragsdaten ein Report erstellen.

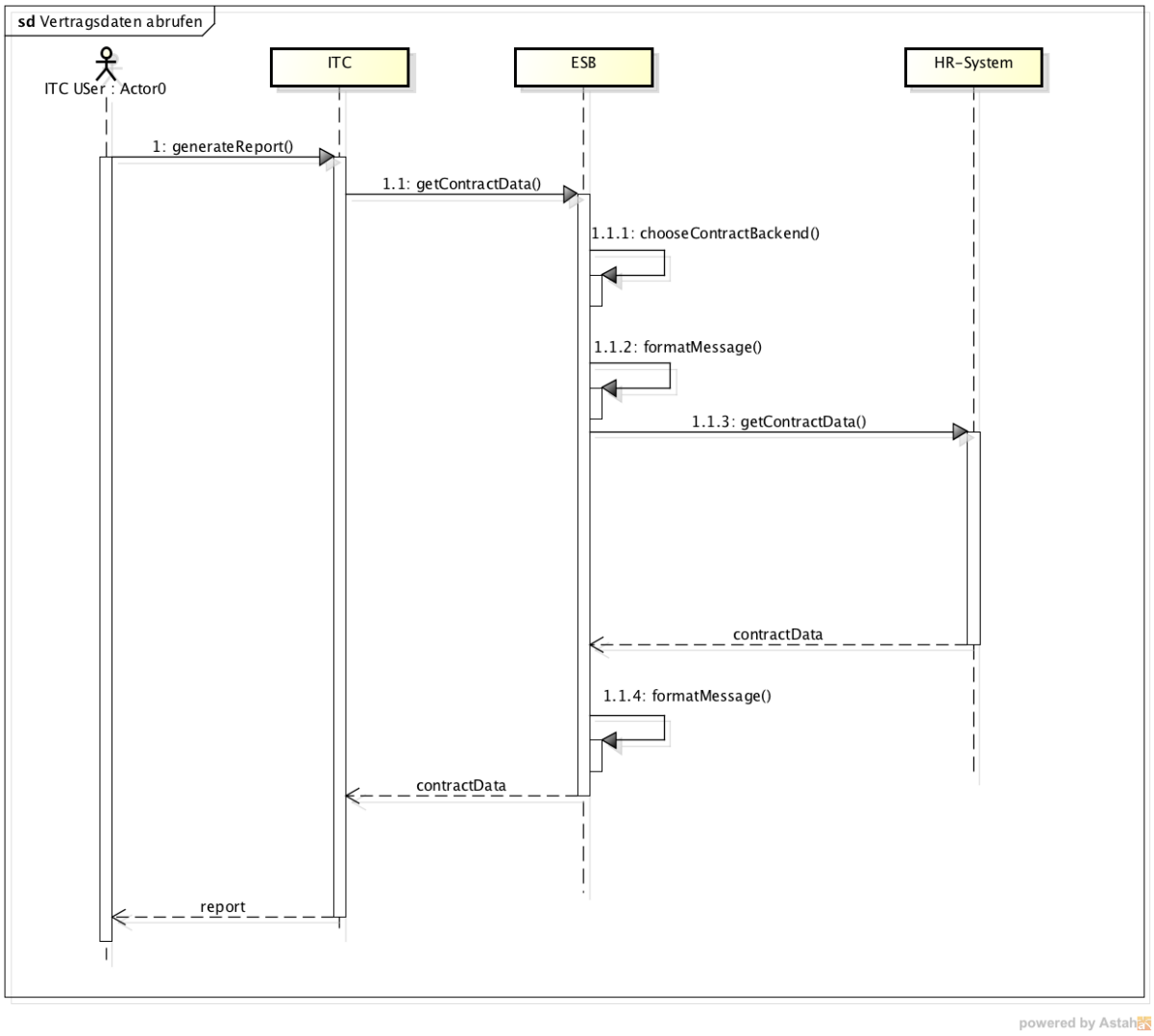
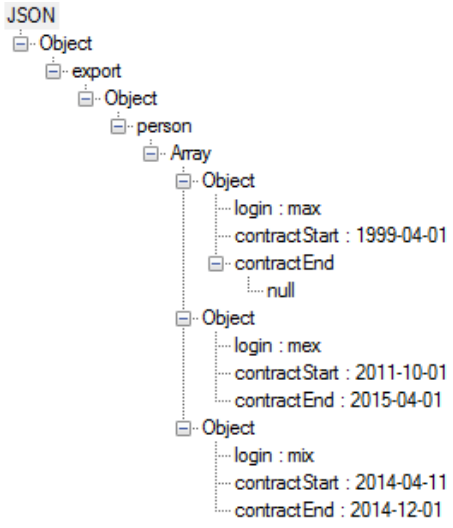


Abbildung 1 Sequenzieller Ablauf des Anwendungsfalls

2.2.2.3 Technische Details ITC

Die ITC-Software verlangt für den Import die Daten im JSON-Format. Benötigt werden Login sowie Start- und Enddatum des Vertrags (falls vorhanden, sonst *null)*. Das folgende Bild zeigt Beispieldaten.




```
{
    "export": {
        "person": [{
            "login": "max",
            "contractStart": "1999-04-01",
            "contractEnd": null
        },
        {
            "login": "mex",
            "contractStart": "2011-10-01",
            "contractEnd": "2015-04-01"
        },
        {
            "login": "mix",
            "contractStart": "2014-04-11",
            "contractEnd": "2014-12-01"
        }]
    }
```

Code 1 ITC-Importformat Beispieldaten in JSON

2.2.2.4 Technische Details Umantis

Die Umantis-Daten können über einen direkten HTTP-Aufruf einer Webseite bezogen werden. Für die Umsetzung relevante Sicherheitsmechanismen gibt es keine zu beachten, da der Zugriff durch Umantis auf die öffentlichen IPs der AdNovum beschränkt ist. Die Daten werden als XML mit der folgenden Struktur zur Verfügung gestellt:

```
<?xml version="1.0" encoding="UTF-8"?>
<export name="AdNovum employees" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <person system_key="umantis" person_key="max">
        <Person_Schluessel>max</Person_Schluessel>
        <Stelle_Schluessel>max</Stelle_Schluessel>
        <Quellsystem_Schluessel>umantis</Quellsystem_Schluessel>
        <Person_Nummer>max</Person_Nummer>
        <Person_Login>max</Person_Login>
        <Person_Anrede>Herr</Person_Anrede>
        <Person_Vorname>Max</Person_Vorname>
        <Person_Nachname>Muster</Person_Nachname>
        <Person_Rolle>MitarbeiterIn</Person_Rolle>
        <Person_Sprache>Deutsch</Person_Sprache>
        <Person_Geburtsdatum>1980-01-22</Person_Geburtsdatum>
        <Person_Antrittsdatum>1999-04-01</Person_Antrittsdatum>
        <Person_Austrittsdatum/>
        <Person_AdressePrivat>Musterstrasse 37</Person_AdressePrivat>
        <Person_PLZPrivat>1234</Person_PLZPrivat>
        <Person_OrtPrivat>Zurich</Person_OrtPrivat>
        <Person_LandPrivat>CH</Person_LandPrivat>
        <Person_TelPrivat>+41 450 65 73</Person_TelPrivat>
        <Person_TelMobilPrivat/>
        <Person_AdresseGeschaeft>Musterfirmastrasse 77</Person_AdresseGeschaeft>
        <Person_PLZGeschaeft>8005</Person_PLZGeschaeft>
        <Person_OrtGeschaeft>Zürich</Person_OrtGeschaeft>
        <Person_LandGeschaeft>CH</Person_LandGeschaeft>
        <Person_TelDirektGeschaeft>+41 44 111 11 11</Person_TelDirektGeschaeft>
        <Person_TelZentralGeschaeft>+41 44 111 11 10</Person_TelZentralGeschaeft>
        <Person_TelMobilGeschaeft/>
        <Person_TelInternGeschaeft>111</Person_TelInternGeschaeft>
        <Person_HomepageGeschaeft>http://www.firma.ch</Person_HomepageGeschaeft>
        <Person_EmailGeschaeft>max.muster@firma.ch</Person_EmailGeschaeft>
        <Person_Autokennzeichen/>
        <Person_Raumnummer/>
        <Person_AbacusNr>10</Person_AbacusNr>
        <Person_Kuerzel>MAX</Person_Kuerzel>
        <Person_Bemerkungen/>
        <Person_geaendert_von>Import</Person_geaendert_von>
        <Person_geaendert_am>2014-09-19 13:34:24</Person_geaendert_am>
        <Stelle_Nummer>max</Stelle_Nummer>
        <Stelle_Bezeichnung>Software Engineer</Stelle_Bezeichnung>
        <Stelle_Schluesselstelle/>
        <Stelle_Antrittsdatum>1999-04-01</Stelle_Antrittsdatum>
        <Stelle_Austrittsdatum/>
        <Stelle_Aktualitaet>PositionType_Position</Stelle_Aktualitaet>
        <Stelle_VorgesetzterPersonenSchluessel>mmm</Stelle_VorgesetzterPersonenSchluessel>
        <Stelle_Status>Aktuelle Stelle</Stelle_Status>
        <Stelle_Funktionsbereich>ProdEng</Stelle_Funktionsbereich>
        <Stelle_Organisationseinheit>Firma Informatik AG</Stelle_Organisationseinheit>
        <Stelle_Fuehrungsverantwortung>Keine</Stelle_Fuehrungsverantwortung>
        <Stelle_Stufe>Mitarbeiter/in</Stelle_Stufe>
        <Stelle_Stellenart/>
        <Stelle_Arbeitsregion/>
        <Stelle_Arbeitsort>Zürich</Stelle_Arbeitsort>
        <Stelle_Jobsprache>Deutsch</Stelle_Jobsprache>
        <Stelle_Beschaeftigungsgrad>100%</Stelle_Beschaeftigungsgrad>
        <Stelle_geaendert_von>Import</Stelle_geaendert_von>
        <Stelle_geaendert_am>2014-09-19 13:34:24</Stelle_geaendert_am>
    </person>
</export>
```

Code 2 Personaldaten-Export aus Umantis in XML

2.2.2.4.1 Daten-Translation und -Transformation

Um die Daten von Umantis für die ITC-Applikation verwendbar zu machen, müssen sie im ESB zuerst gefiltert und anschliessend in JSON umgewandelt werden.

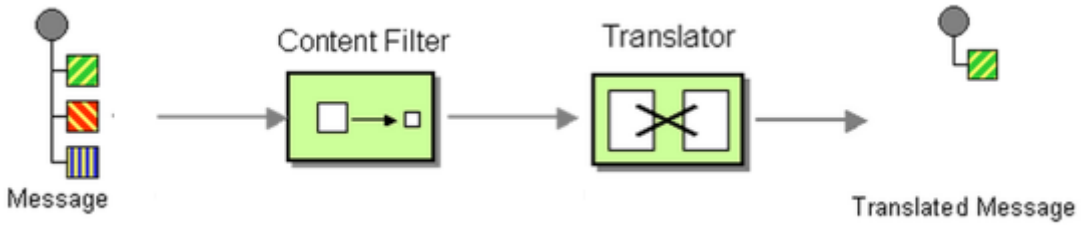


Abbildung 4 Umantis-Datenverarbeitung in ESB

Das Mapping der Attributfelder zwischen den beiden Applikationen sollte wie folgt aussehen:

Umantis	ITC-Applikation
Person_Login	login
Person_Antrittsdatum	startContract
Person_Austrittsdatum	endContract

Tabelle 2 Mapping der Attributfelder zwischen Umantis und ITC

2.2.2.5 Technische Details Abacus

Abacus kann via Browser mit dem Öffnen einer "Java Network Launching Protocol(JNLP)"-Datei gestartet werden. Die Java-Applikation bietet gemäss Hersteller keine Schnittstelle via URLs an, sondern lediglich Webservices.
Wir konnten im Rahmen der Studienarbeit dieses Backend nicht weitergehend analysieren und einbinden. Wir haben uns deshalb auf vom Industriepartner höher priorisierte Inhalte konzentriert.

2.2.3 Funktionale Anforderungen

Aus dem Use Case und Gesprächen mit den Projektbetreuern bei AdNovum können die folgenden funktionalen Anforderungen definiert werden:

Nr.	Prio [1 = required] [2 = nice to have] [3 = optional]	Beschreibung	Zugehörige Kriterien
FA1	1	SOAP (über HTTP) und RESTful HTTP Adapter Support	FK1
FA2	2	Parallele Nachrichtenverarbeitung im ESB aus Sicht des Anwendungsprogramms (mehrere gleichzeitige Streams) → Competing Consumer Pattern	DK3
FA3	2	Die Nachrichten müssen mindestens mit einer 128-Bit Advanced-Encryption-Standard(AES)-Verschlüsselung übertragen werden.	FK3
FA4	1	Der Consumer/Producer muss authentifiziert werden.	FK3
FA5	1	Java-Unterstützung für Programmierung des ESB (welche Java Edition?	FK2
FA6	1	Daten sollen im Format Unicode Transformation Format-8 (UTF-8) an ITC geliefert werden.	DK2

FA7	1	Daten sollen so transformiert werden, dass sie im ITC ohne weitere Konvertierung eingepflegt werden können (z.B. Eintrittsdatum im Format DD.MM.JJJJ).	DK2

Tabelle 3 Funktionale Anforderungen

Die nachfolgende Tabelle enthält alle nichtfunktionalen Anforderungen. Sie wurde mit Christian Fritz diskutiert.

Nr.	Prio **[1 = required] [2 = nice to have] [3 = optional]**	Beschreibung	Zugehörige Kriterien
NFA1	1	Die ESB-Komponenten sollen modular aufgebaut sein. Der Kunde soll nur das kaufen / installieren / lizenzieren müssen, was er auch möchte.	AK3
NFA2	2	Der ESB kann die Daten von Verträgen für mindestens 400 Mitarbeiter in einem Request abhandeln. Ein Datensatz beinhaltet zwei Datumsfelder (Start und Ende des Vertrags) und das Login des Mitarbeiters und ist kleiner als eins bis zwei Kilobyte gross.	FK6
NFA3	2	Der ESB kann mindestens 100 Anfragen (UC-1) vom System ITC pro Sekunde ohne Verluste verarbeiten.	FK6
NFA4	2	Der ESB antwortet erfolgreich auf 99,99% der Anfragen (UC-1) innerhalb einer Sekunde. Auf die restlichen Anfragen (0,01%) wird innerhalb der nächsten 12 Stunden eine Antwort geliefert.	FK6
NFA5	3	Verbindungen sollen in maximal 500ms aufgebaut werden. Der ESB soll Connection Pooling anbieten, so dass nicht alle Verbindungen neu geöffnet werden müssen. Der Aufwand für den Verbindungsaufbau soll dadurch verringert werden.	
NFA6	1	Der ESB soll bei der Implementation von eigenen Artefakten die Programmiersprache Java (mindestens Java 7 Update 60) unterstützen.	FK2

Tabelle 4 Nichtfunktionale Anforderungen

3 Evaluation eines Enterprise Service Bus

Um zwei konkrete Produkte für unsere Studienarbeit zu evaluieren, haben wir zuerst zwölf der bekanntesten Produkte oberflächlich analysiert. Im folgenden Abschnitt werden diese Produkte vorgestellt. Im Anschluss haben wir dann einen Kriterienkatalog erstellt, um die zwei ausgewählten ESB-Produkte bewerten zu können.

3.1 Übersicht aktueller ESB-Produkte

Für die Produkteevaluation wurden im Vorfeld die unten aufgeführten Produkte ausgewählt. Bei der Auswahl wurde besonders auf die unterschiedlichen Einsatzgebiete/Spezialisierungen der ESBs geachtet. Da eine Anforderung darin besteht, ein Open-Source- und ein kommerzielles Produkt zu evaluieren, wurde darauf geachtet, in beiden Kategorien eine Auswahl anzubieten.

3.1.1 Kommerzielle Produkte

3.1.1.1 Oracle Service Bus

Beschreibung	Ehemals AquaLogic Service Bus (von BEA Systems übernommen). Bietet eine Integration an, sodass komplexe Architekturen agiler werden. Besonderen Wert wird auf Performance und Skalierbarkeit gelegt.
Auswahlgrund	• Gute Unterstützung für Java • Auf dem Markt etabliert (OSB-Einführung bei der PostFinance durch [ipt[2]]) • AdNovum hat Partnerstatus bei Oracle
Quelle	http://www.oracle.com/technetwork/middleware/service-bus/overview/index.html

3.1.1.2 Tibco ActiveMatrix Service Bus

Beschreibung	Leichtgewichtiger ESB, welcher zur Schliessung der Lücke bei der Übermittlung (Mediation Gap) bei SOA-Infrastrukturen dient.
Auswahlgrund	• Orientierung Richtung Configuration over Code • Breite Enterprise-Integration-Patterns-Unterstützung
Quellen	http://www.tibco.com/products/automation/application-integration/enterprise-service-bus/activematrix-service-bus https://docs.tibco.com/products/tibco-activematrix-service-bus-3-3-0

3.1.1.3 SAP NetWeaver Process Integration

Beschreibung	Eine Komponente von SAP NetWeaver, welche den Datenaustausch zwischen SAP und fremden Systemen ermöglicht. Unterstützt Messaging und verhindert Datenverlust durch Persistenz der Nachrichten.
Auswahlgrund	• SAP-Produkte sind in der Unternehmenswelt gut integriert und werden viel gebraucht • Grosse Community • Sehr viel Funktionalität vorhanden
Quelle	http://help.sap.com/saphelp_nwpi711/helpdata/en/c0/3930405fa9e801e10000000a155106/frameset.htm

[2] [ipt] ist ein Informatik Unternehmen (http://www.ipt.ch/de/)

3.1.1.4 IBM WebSphere Enterprise Service Bus

Beschreibung	Verbindet flexibel Anwendungen und Services (im SOA-Umfeld). Vermindert somit die Komplexität der Integration.
Auswahlgrund	• Grosse Firma steht hinter dem Produkt • Hoffnung auf einfache Integration mit anderen IBM-Produkten wie Application Servers • Bietet für viele Mainframe-Systeme Adapter an
Quelle	http://www-03.ibm.com/software/products/de/wsesb

3.1.2 Open-Source-Produkte

3.1.2.1 JBoss ESB

Beschreibung	Bietet Business Process Monitoring, Integrated Development Environment, Human Workflow User Interface, Business Process Management, Connectors, Transaction Manager, Security, Application Container und noch vieles mehr ohne Vendor Lock-in.
Auswahlgrund	• Gute Unterstützung für Java • Gute Kompatibilität mit JBoss Application Server • Grosse Community und guter Support
Quelle	http://jbossesb.jboss.org/

3.1.2.2 Apache ServiceMix

Beschreibung	Flexibler Open Source Integration Container, welcher die Funktionalitäten von Apache ActiveMQ, Camel und CXF sowie Karaf bündelt, um eine Integrationsumgebung zu schaffen.
Auswahlgrund	• Open-Source-Projekt (Apache) • Sehr Java-orientiert • Benutzt mehrere Apache-Projekte, welche in der Branche bekannt sind (Apache Camel, Apache ActiveMQ)
Quelle	http://servicemix.apache.org/

3.1.2.3 Apache Synapse

Beschreibung	Leichtgewichtiger und hoch performanter ESB. Zusätzlich zu XML und SOAP bietet Synapse noch Plain Text, Binary, Hessian und JSON als Austauschformate an.
Auswahlgrund	• Open-Source-Projekt (Apache) • Gute Beispiele, viele Enterprise-Integration-Patterns werden abgedeckt • Wird von WSO2 Enterprise Service Bus (1. ausgewähltes Produkt) gebraucht
Quelle	http://synapse.apache.org/

3.1.2.4 WSO2 Enterprise Service Bus

Beschreibung	Leichtgewichtiger und hoch performanter ESB basierend auf Apache Synapse ESB. Referenziert auf bekannte Enterprise Integration Patterns.
Auswahlgrund	• Wird von grossen Unternehmen wie eBay eingesetzt • Bestehende Möglichkeit der Integration in der Cloud • Dokumentation basierend auf Enterprise Integration Patterns

Quelle	http://wso2.com/products/enterprise-service-bus/

3.1.2.5 Mule ESB

Beschreibung	Leichtgewichtiger Java-basierter ESB und Integrationsplattform. Ermöglicht schnelle und einfache Anbindung von Applikationen und deren Datenaustausch.
Auswahlgrund	• Gut etabliertes Open-Source-Produkt • Viel Dokumentation verfügbar • Eigene IDE mit visuellem Editor
Quellen	http://www.mulesoft.com/ http://www.mulesoft.org/what-mule-esb

3.1.2.6 Sun Microsystems OpenESB

Beschreibung	OpenESB ist das einfachste und effizienteste ESB-Produkt für die Integration von SOA-Applikationen. OpenESB wird durch die Community weiterentwickelt. Bietet Tools für Design, Development, Test und Deployment von SOA-Applikationen an.
Auswahlgrund	• Gute Unterstützung für Java • Interessant, ob längerfristig geeignet (da Sun von Oracle übernommen wurde) • Grosse Community (Forum mit über 3700 Themen, Wiki)
Quelle	http://www.open-esb.net/

3.1.2.7 Spring Integration

Beschreibung	Benötigt das Spring Framework und ermöglicht so die Programmierung von Enterprise Integration Patterns im Spring-Umfeld. Bietet ein leichtgewichtiges Messaging für Spring-basierte Applikationen an.
Auswahlgrund	• Gute Unterstützung mit Java (Spring) • Grosse Community • Es werden konkrete Enterprise Architecture Patterns wie Aggregator, Transformer und Channel erwähnt
Quelle	http://projects.spring.io/spring-integration/

3.1.2.8 Talend ESB

Beschreibung	Talend ist ein zuverlässiger und skalierbarer ESB und vereinfacht die Verbindung von SOA-Applikationen und die Verwaltung solcher Services.
Auswahlgrund	• Einziges Produkt mit deutscher Dokumentation • Hat eigene Eclipse-ähnliche IDE für Modellierung der Integration • Unterstützt laut Website Messaging, Webservices, Data Services Routing, sowie Datentransformation
Quellen	http://de.talend.com/products/esb http://www.talend.com/resource/apache-esb.html

3.2 Auswahl der zu evaluierenden Produkte

Aufgrund der organisatorischen Rahmenbedingungen (Kooperation zwischen Produktehersteller Oracle und Industriepartner AdNovum Informatik AG) konnten wir das erste Produkt, welches kommerziell ist, einfach und schnell identifizieren.

Bei den Open-Source-Produkten war es notwendig, dass wir uns zuerst oberflächlich in die einzelnen ESBs einarbeiten. Dann analysierten wir nochmals die Anforderungen und das Unternehmensprofil der AdNovum Informatik AG, um die Auswahl einzugrenzen. Ausschlaggebend war vor allem die in den Anforderungen aufgeführte Unterstützung von Java [NFA6]. Weiter legten wir Wert darauf, dass eine gute Dokumentation verfügbar ist und die Entwicklung noch aktiv läuft. Zur Auswahl standen noch die folgenden Produkte:

- Mule ESB
- Spring ESB
- Talend ESB
- WSO2 Enterprise Service Bus

Nach einiger Recherche fanden wir Illustrationen der letzten Forrester Wave über Enterprise Service Bus und der Magic Quadrants von Gartner für „Application Infrastructure for Systematic Application Integration Projects“. Bis auf den Spring ESB kommen alle drei Produkte in mindestens einem der Reporte vor.

Auffallend in den Berichten ist, dass der WSO2 – nebst FuseSource, welches jedoch nicht zur Wahl steht – die beste Bewertung von Forrester erhielt (bei den Open-Source-Produkten). Beim Vergleich der beiden Gartner Magic Quadrants von 2010 und 2012 fällt zudem auf, dass WSO2, Talend und MuleSoft sich signifikant verbessern konnten und es alle in den Visionaries Quadrant geschafft haben. Auch bei der Überprüfung des Lizenzmodels (Apache License Version 2.0 [6]) von WSO2 waren wir vom Produkt überzeugt. Aufgrund dieser Fakten kann man sagen, dass WSO2 ein innovatives, interessantes und zukunftsversprechendes Unternehmen ist. Der WSO2 ESB wurde bislang jedoch in sehr wenigen ESB Evaluationen und Vergleichen detailliert betrachtet. Aus diesen Gründen haben wir uns für dieses Produkt entschieden.

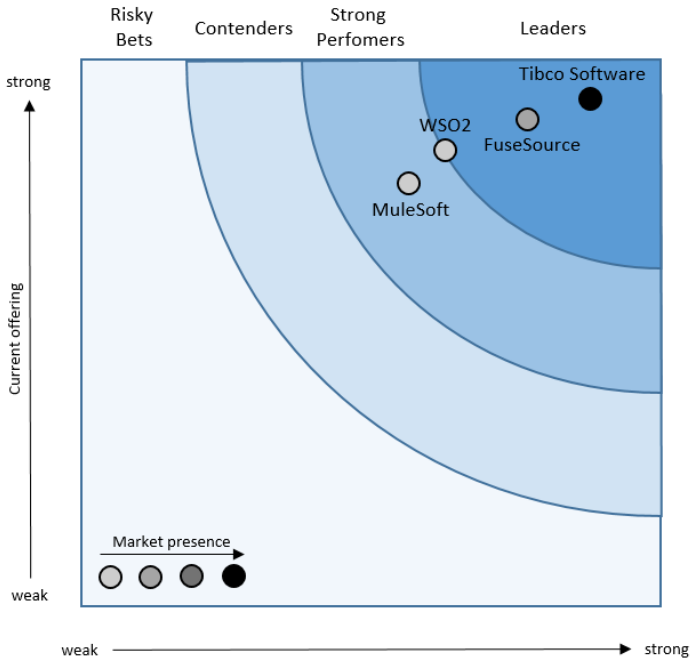


Abbildung 5 Nachbildung der Forrester Wave™: Enterprise Service Bus, Q2'11

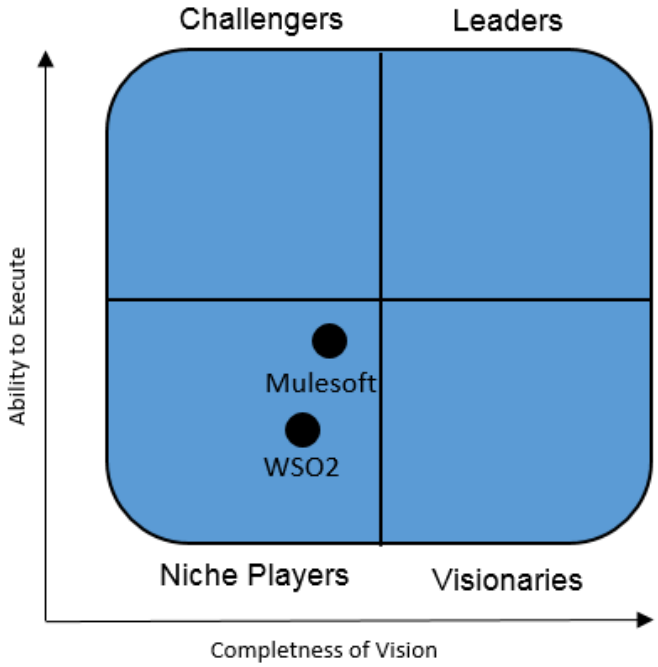


Abbildung 6 Nachbildung des Gartner Magic Quadrant for Application Infrastructure for Systematic Application Integration Projects, 2010

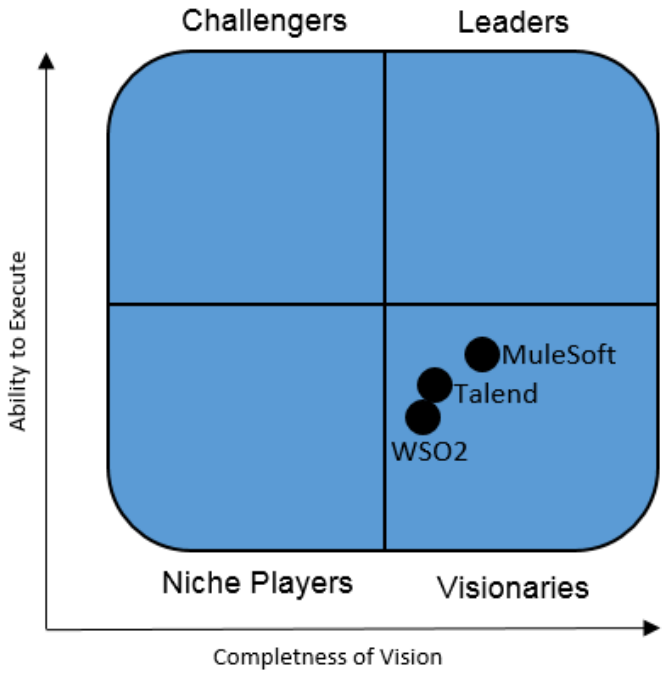


Abbildung 7 Nachbildung des Gartner Magic Quadrant for Application Infrastructure for Systematic Application Integration Projects, 2012

3.3 Bewertungskriterien

Nach der Auswahl der Produkte haben wir einen Bewertungskriterienkatalog für die ESBs erstellt. Mit Hilfe dieser Kriterien erhält man eine gute Übersicht über die ESBs selbst sowie deren Funktionsumfang. Die Kriterien sind bewusst nicht auf die notwendige Anforderung aus dem Use Case beschränkt, da der Use Case nur als Beispielfall gilt und für einen produktiven Einsatz weit mehr Anforderungen zu erfüllen hat. In diesem Kapitel stellen wir die Bewertungskriterien gegliedert nach Kategorien vor.

Kategorie	Kriterien
Funktionale Kriterien	• FK1 – Unterstützte Protokolle/Technologien und Adapter [FA1] • FK2 – Custom Adapter Framework [FA5] • FK3 – Security [FA3, FA4] • FK4 – Message Translation • FK5 – Datenstrukturtiefe • FK6 – Message-Durchsatz / Concurrent Streams • FK7 – Betreibbarkeit in Cloud • FK8 – Discovery and Registry • FK9 – Workflow Engine and Service Composition • FK10 – Versionierung der Komponenten
Design-Kriterien	• DK1 – Message Routing Patterns • DK2 – Message Content Transformation Pattern [FA6, FA7] • DK3 – Message Consumer Patterns [FA2]
Support- und Troubleshooting-Kriterien	• STK1 – Reporting and Statistic • STK2 – Alerting and Monitoring • STK3 – Failover / Load Balancing • STK4 – Verfügbare Dokumentation Community • STK5 – Support
"Look & Feel"-Kriterien	• LFK1 – Manageability • LFK2 – Entwicklungsumgebung • LFK3 – Aufwand Installation / Konfiguration
Allgemeine Kriterien	• AK1 – Configuration over Coding • AK2 – HW-/SW-Anforderungen • AK3 – Modularer Aufbau der Applikation [NFA1] • AK4 – Kosten

Tabelle 5 Übersicht der Bewertungskriterien

Die folgenden Abschnitte beschreiben die einzelnen Bewertungskriterien im Detail. Alle Kriterien werden dreistufig bewertet. Grundsätzlich gilt dabei Kriterium erfüllt/übertroffen, mittelmässig erfüllt, nicht erfüllt. Als Einteilungshilfe ist jeweils eine Beschreibung der Stufen aufgeführt.

3.3.1 Funktionale Kriterien

3.3.1.1 FK1 – Unterstützte Protokolle/Technologien und Adapter

Welche Standardprotokolle werden für die Nachrichtenübermittlung angeboten und wie sind sie implementiert?

Welche Technologien und Protokollverschachtelungen werden unterstützt? Gibt es zudem vorgefertigte Adapter zu bekannten Third-Party-Applikationen/Services wie JIRA, SAP oder Salesforce?

Zu den von uns als Standard definierten Protokollen/Technologien gehören:

- Database Connector mit TCP-Unterstützung
- E-Mail (SMTP/POP3)
- File Import/Export over S/FTP
- SOAP over HTTPS/S und TCP
- REST over HTTPS/S

Bewertung:

- ***Standardprotokolle plus zusätzliche Adapter/Konnektoren zu Third-Party Services***
- ***Nur Standardprotokolle***
- ***Nicht alle Standardprotokolle***

3.3.1.2 FK2 – Custom Adapter Framework

Wie sieht die Unterstützung von Custom Adapters aus? Welche Libraries werden unterstützt (z.B. Java Class Library)? Können Java-Komponenten (Servlets, Beans) in den Application Server deployed werden?

Bewertung:

- ***Gute Unterstützung***
  Implementation eines Custom Adapter ist einfach und es wird eine gängige Library zur Verfügung gestellt
- ***Mittlere Unterstützung***
  Implementation ist umständlich, eine gängige Library wird unterstützt
- ***Schlechte Unterstützung***
  Schlechte Custom-Adapter-Framework-Unterstützung

3.3.1.3 FK3 – Security

Es werden die Security Objectives gemäss NIST [8] unterstützt, sodass der ESB die Security gewährleisten kann.

a) Authentication
   a. Was für Authentifizierungsmethoden werden von den Adaptern unterstützt?
   b. Unterstützte Zertifikate?
   c. Security-Engineering-Probleme
      Was passiert bei mehrfachem Login durch technischen User des ESB? Problematik von technischen Usern einbeziehen.

b) Authorization
Wie stellt der ESB sicher, dass das Backend, welches angeschlossen ist, tatsächlich dieses ist, welches angibt es zu sein?

c) Confidentiality
Wird die Vertraulichkeit der Daten sichergestellt? Wird Verschlüsselung angeboten? Welche Verschlüsselungsalgorithmen werden unterstützt (z.B. durch AES)?

d) Data Integrity
Wird die Integrität der Daten sichergestellt (z.B. mit MAC oder einer vergleichbaren Checksummenfunktion)?
(End-User vs. technischer User)

Bewertung:

- ***Hoher Security-Standard***
  Das Produkt bietet in allen vier Bereichen eine Security-Lösung an, welche die heutigen Standards übertrifft.

- ***Standard-Security***
  Das Produkt bietet in allen vier Bereichen eine Security-Lösung an, welche den heutigen Standards entspricht.
- ***Mangelnde Security-Funktionen***
  Das Produkt bietet nicht in allen Bereichen eine Security-Lösung an und/oder die Funktionen entsprechen nicht dem heutigen Standard.

  Anmerkung: Da Security sehr weitreichend ist und auf jeder Schnittstelle andere Implementationen und Funktionen umfasst, wird hier nur die Security für die von uns gemäss Use Case untersuchten HTTP-Schnittstellen bewertet.

  Als Standards haben sich zurzeit Verschlüsselungen mit RSA und AES oder Camelia (mind. 128-Bit), Übertragung via SSL (mind. TLS 1.0) und Überprüfung der Integrität mittels Einweg-Hashfunktion durchgesetzt.

3.3.1.4 FK4 – Message Translation

Welche Formate werden für die Message Translation angeboten?

a) Binary
b) CSV (text-based)
c) JSON
d) XML

Bewertung:

- ***Gute Unterstützung***
  Alle vier Formate werden unterstützt.
- ***Mittlere Unterstützung der Formate***
  Es werden nicht alle Formate unterstützt, jedoch reichen in der Regel die angebotenen Message-Translation-Formate aus.
- ***Schlechte Unterstützung***
  Es werden keine oder sehr wenige Formate unterstützt. In der Praxis reichen diese nicht aus.

3.3.1.5 FK5 – Datenstrukturtiefe

Wie tiefe Datenstrukturen werden vom ESB noch unterstützt respektive können noch z.B. durch Filterung ausgewertet werden?

Bewertung:

- ***Mehrfach verschachtelte Strukturen***
- ***Maximal zweifach verschachtelte Strukturen***
- ***Keine verschachtelten Strukturen***

3.3.1.6 FK6 – Message-Durchsatz / Concurrent Streams

Wie viele Messages kann ein ESB gleichzeitig verarbeiten?

Bewertung:

- ***Erweiterte synchrone Verarbeitung***
  Es können mehr als drei Messages gleichzeitig verarbeitet werden
- ***Einfache synchrone Verarbeitung***
  Es können zwei bis drei Messages gleichzeitig verarbeitet werden
- ***Sequenzielle Verarbeitung***
  Keine Concurrent Streams

3.3.1.7 FK7 – Betreibbarkeit in Cloud

Lässt sich das Produkt in einer Cloud betreiben oder gibt es gar bereits Anbieter, die das Produkt als Cloud-Lösung bereitstellen? Werden die IDEAL-Eigenschaften [9, p. 6] vom ESB eingehalten?

a) **Isolated State**
   Das Produkt ist stateless in Bezug auf Session und Application State.
b) **Distribution**
   Das Produkt lässt sich in mehrere Komponenten aufteilen, welche unabhängig deployed werden können.
c) **Elasticity**
   Das Produkt lässt sich gut skalieren.
   a. **Scale Out**
      Performanz steigt mit der Anzahl Ressourcen (mehrere Knoten mit ESB-Installation)
   b. **Scale Up**
      Performanz steigt mit steigender Performanz der Ressourcen (mehr CPU, RAM für einen Knoten mit ESB-Installation)
d) **Automated Management**
   Runtime Tasks müssen schnell abgearbeitet werden.
e) **Loosely Coupling**
   Komponenten innerhalb des ESB sind lose gekoppelt und hängen nicht voneinander ab.

Bewertung:

- ***IDEAL-Eigenschaften werden erfüllt***
- ***Stateless und Cloud-Erfahrung vorhanden***
- ***Keine Angaben verfügbar***

3.3.1.8 FK8 – Discovery and Registry

Discovery and Registry sind umfangreiche Funktionen und für den Use Case nicht relevant. Sie werden deshalb im Rahmen der Studienarbeit nicht evaluiert.

3.3.1.9 FK9 – Workflow Engine and Service Composition

Workflow Engine and Service Composition sind umfangreiche Funktionen und für den Use Case nicht relevant. Sie werden deshalb im Rahmen der Studienarbeit nicht evaluiert.

3.3.1.10 FK10 – Versionierung der Komponenten

Lassen sich die Komponenten (z.B. Endpunkte, Connections, Sequenzen) versionieren? Können mehrere Versionen gleichzeitig betrieben werden? Ist es möglich, eine neue Version einer Komponente zu entwickeln und diese zu einem bestimmten Zeitpunkt zu veröffentlichen / aktivieren?

Bewertung:

- ***Versionierung der Komponenten und Betrieb von multiplen Versionen unterstützt***
- ***Versionierung unterstützt, keine multiplen Versionen möglich***
- ***Keine Versionierung unterstützt***
  Was konfiguriert wird, ist immer in der Produktionsumgebung live installiert.

3.3.2 Design-Kriterien

3.3.2.1 DK1 – Message Routing Patterns

Welche Routing Patterns werden wie unterstützt [10, p. 224]?

a) **Simple Router**
   Content-Based Router – Routing anhand des Nachrichteninhalts. Ist es möglich, hierzu den Message-Inhalt und den Header zu durchsuchen?

   Message Filter – Können Nachrichten bedingt weitergeleitet und wenn nötig verworfen werden? Was passiert mit verworfenen Messages?

   Recipient List – Weiterleiten der Nachrichten an mehrere Channels. Kann z.B. eine Message in eine Logdatenbank und an den effektiven Empfänger versendet werden?

   Splitter – Grössere Nachrichten aufteilen (z.B. wenn mehre Elemente enthalten sind) und einzeln routen. Nach welchen Kriterien kann eine Message gesplittet werden?

   Aggregator – Gesplittete Nachrichten zusammenfügen. Nach welchen Kriterien ist dies möglich?

   Resequencer – Nachrichtensequenzen in die korrekte Reihenfolge überführen und entsprechend weiterleiten. Welche Art von Sequenzen (z.B. TCP-Sequenzen) werden unterstützt? Wie hoch ist die maximal unterstützte Message-Grösse?
b) **Composed Routers**
   Können mehrere Routingvarianten kombiniert werden? Welche Einschränkungen bestehen?
c) **Dynamic Routing**
   Kann dynamisch geroutet werden? Falls z.B. mehrere potentielle Empfänger vorhanden sind, kann der ESB an den routen, der am wenigsten ausgelastet ist?

Bewertung:
- ***Alle sechs Patterns beliebig kombinierbar und dynamisches Routing***
- ***Teil der Patterns und beliebige Kombinationsmöglichkeit***
- ***Teil der Patterns ohne Kombinationsmöglichkeit***

3.3.2.2 DK2 – Message Content Transformation Patterns

Welche Möglichkeiten gibt es, um den Inhalt der Messages von einem Format in ein anderes Format transformieren zu können? Mit welchem Aufwand können diese Patterns angewendet werden? Wie modelliert man diese Patterns?

a) **Content Enricher** [10, p. 336]
   Kann man die Messages durch Hinzufügen von zusätzlichen Informationen (z.B. zusätzliche Datenfelder) erweitern?
b) **Content Filter** [10, p. 342]
   Kann man den Inhalt der Messages durch gezieltes Entfernen von nicht brauchbaren Informationen filtern?
c) **Extensible Stylesheet Language Transformation (XSLT)**
   Die Patterns können mit XSLT modelliert werden.
d) **XQuery**
   Die Patterns können mit XQuery modelliert werden.
e) **Domain Specific Language (DSL)**
   Die Patterns können mit einer DSL modelliert werden. Somit fällt zusätzliche Einarbeitungszeit an, um die DSL zu lernen.
f) **Eigenes Tool**
   Die Patterns können mit einem eigenen Tool modelliert werden. Somit fällt zusätzliche Einarbeitungszeit an, um das Tool kennenzulernen.

Bewertung:

- ***Content Enricher und Content Filter werden unterstützt. Die Modellierungssprache ist standardisiert und somit nicht herstellerspezifisch.***
- ***Content Enricher und Content Filter werden unterstützt. Die Modellierungssprache ist herstellerspezifisch. Zusätzlicher Aufwand für das Erlernen der Modellierungssprache fällt an.***
- ***Es werden keine Message Content Transformation Patterns unterstützt.***

3.3.2.3 DK3 – Message Consumer Patterns

Endpoints stellen die Schnittstellen zu den angeschlossenen Services dar. Sie werden nach ihren Eigenschaften und Funktionen bewertet.

- **Message Consumer Patterns**
  Welche Message Consumer Patterns der folgenden Kategorien können mit dem ESB abgebildet werden [10, p. 464]?
  - **Polling Consumer**
    Kann der Consumer Endpoint einen expliziten Call machen, wenn er Messages erhalten will?
  - **Event Driven Consumer**
    Was passiert mit nicht angeforderten Messages? Können diese ebenfalls verarbeitet werden?
  - **Competing Consumer**
    Können mehrere gleichwertige Consumer verwendet werden, um mehrere Messages gleichzeitig zu verarbeiten?
  - **Message Dispatcher**
    Ist es möglich, zwischen mehreren gleichwertigen Consumer die Messages strategisch zu verteilen (z.B. durch Load Balancing)?
  - **Selective Consumer**
    Können Nachrichten anhand ihres Inhalts gefiltert werden? Wie kann der Filter spezifiziert werden (z.B. XML Schema Validation)? Was passiert mit Messages, die den Kriterien nicht entsprechen?
  - **Durable Subscriber**
    Können Messages für einen bestimmten Consumer auch noch zu einem späteren Zeitpunkt weitergeleitet werden, wenn der Consumer aktuell nicht erreichbar ist?
  - **Idempotent Receiver**
    Wie behandelt der ESB doppelt erhaltene Nachrichten? Kann er diese überhaupt erkennen?
  - **Service Activator**
    Können nur einzelne Services einer angeschlossenen Applikation aufgerufen werden, basierend auf Messaging- und Non-Messaging-Technologien? Kann der ESB sozusagen als Proxy fungieren?

Bewertung:

- ***Alle 8 Patterns***
- ***5-7 Patterns***
- ***Weniger als 5 Patterns***

3.3.3 Support- und Troubleshooting-Kriterien

3.3.3.1 STK1 – Reporting and Statistic

Werden die Aktivitäten im ESB benutzerfreundlich und gut lesbar geloggt? Was wird geloggt (Message Flow, Konfigurationsänderungen usw.)? Können daraus direkt im ESB Reports und Statistik abgerufen werden?

Bewertung:

- ***Message Flow und Konfigurationsänderungen, nützliche Aufbereitung***
- ***Message Flow und/oder Konfigurationsänderungen, mässige Aufbereitung***
- ***Kein Reporting***

3.3.3.2 STK2 – Alerting and Monitoring

Können für den Betrieb des ESB Meldungen definiert werden und besteht die Möglichkeit zu einem Live Monitoring? Interessante Überwachungsgegenstände sind der Message Flow und die laufenden Dienste. Unterstützt der ESB ausserdem SNMP?

Bewertung:

- ***Alerting und Monitoring***
- ***Alerting oder Monitoring***
- ***Keine der Funktionen***

3.3.3.3 STK3 – Failover / Load Balancing

Bietet der ESB eine Möglichkeit für Load Balancing und Failover an? Wie gut lässt sich ein ESB verteilen?

a) **Verlust von Nachrichten**
   Beim Ausfall oder der Lastverteilung werden eine oder mehrere Nachrichten nicht zugestellt, weil sie zum Beispiel beim ausgefallenen ESB bereits in Verarbeitung waren.
   Wird das Timeout-based Delivery[3] Pattern angewandt?
b) **Doppelte Zustellung von Nachrichten**
   Beim Ausfall oder der Lastverteilung werden eine oder mehrere Nachrichten mehrfach zugestellt, weil die Koordination zwischen den einzelnen Instanzen nicht erfolgreich abläuft.

Bewertung:

- ***Eingebaut***
  Es wird vom Produkt bereits Load Balancing und Failover angeboten, sodass keine zusätzliche Software oder Hardware notwendig ist.
- ***Unterstützt***
  Das Produkt bietet zwar selbst kein Load Balancing und Failover an, jedoch kann der ESB mit einem Failover oder mit Load Balancing so umgehen, dass die Messages auf dem Bus korrekt übermittelt werden.
- ***Nicht unterstützt***
  Das Produkt unterstützt kein Load Balancing und Failover oder ist nicht geeignet dafür, da zum Beispiel die Startzeit zu hoch ist.

[3] Timeout-based Delivery ist ein Cloud Computing Pattern http://www.cloudcomputingpatterns.org/Timeout-based_Delivery (zuletzt aufgerufen am 15.12.2014)

3.3.3.4 STK4 – Verfügbare Dokumentation Community

Wie einfach lässt sich das Produkt anhand einer Anleitung installieren und konfigurieren? Wie gut sind Funktionen dokumentiert? Gibt es eine Community (z.B. in Form eines Forums oder Questions and Answers)?

a) **Installationsanleitung**
   Der Hersteller bietet eine ausreichende Dokumentation zum Produkt, sodass die Installation einfach ist.
b) **Verständlichkeit der Dokumentation**
   Die Funktionen des Produkts werden mit Beispielen verständlich erläutert und mit Konfigurationsmöglichkeiten erklärt.
c) **Community**
   Es werden gängige Use Cases in Foren oder auf ähnlichen Plattformen von anderen Benutzern diskutiert.

Bewertung:

- ***Umfangreiche Dokumentation und Community***
  Es besteht eine gute und verständliche Installationsanleitung und Dokumentation vom Hersteller sowie eine grosse Community.
- ***Dokumentation vorhanden***
  Es besteht eine Installationsanleitung und Dokumentation, jedoch besteht keine Community oder sie ist sehr klein.
- ***Mangelhafte Dokumentation***
  Es gibt keine ausreichende Dokumentation. Man muss sich durch die Funktionalitäten durchkämpfen.

3.3.3.5 STK5 – Support

Besteht die Möglichkeit von „Premium Support“ durch den Produkteanbieter? Wie gut ist dieser Support? Entstehen dadurch zusätzliche Kosten?

a) **Support-Angebot**
   Der Hersteller bietet bei Schwierigkeiten Kundensupport an.
b) **Qualität des Support-Services**
   Der Support ist verständlich und hilfreich.
c) **Kosten für den Support**
   Der Support-Service ist kostenlos oder kostenpflichtig.

Bewertung:

- ***Individueller, preiswerter Support***
  Für das Produkt wird qualitativ hochwertiger Support angeboten. Bei Problemen wird man individuell betreut. Der Support ist inbegriffen oder die Kosten sind nicht zu hoch.
- ***Standard Support***
  Es wird grundsätzlich bei Problemen Support angeboten. Auf individuelle Kundenwünsche wird nur begrenz eingegangen.
- ***Mangelhafter oder kein Support***
  Es wird kein oder nur schlechter Support angeboten oder die Kosten sind extrem hoch.

3.3.4 "Look & Feel"-Kriterien

3.3.4.1 LFK1 – Manageability

Wie gestaltet sich die Konfiguration mittels Managementkonsole? Ist sie intuitiv und lässt sich darüber auf alle wichtigen Funktionen zugreifen? Ist auch das Systemmanagement in der Konsole implementiert?

a) **Command Line Interface**
   Lässt sich das Produkt über eine Konsole bedienen? Kann ein Systemadministrator neue Konfigurationen erfassen, bestehende mutieren und löschen?
b) **Web-Konsole**
   Existiert eine Web-Konsole für das Produkt? Wie benutzerfreundlich ist diese Web-Konsole? Lassen sich einfach Konfigurationen erstellen, mutieren und löschen?
c) **Logging**
   Das Einloggen und andere Aktionen werden in einer Log-Datei festgehalten.

Bewertung:

- ***Sehr gute Usability und integrierte Systemmanagement-Funktionen (für Command Line Interface und Web-Konsole) sowie Logging der Aktivitäten***
  Alle wichtigen Optionen sind schnell und einfach via Command Line Interface oder Web-Konsole zugreifbar, zudem können die ESB-Systemtasks durchgeführt werden. Aktionen werden geloggt.
- ***Mittelgute Usability und integrierte Systemmanagement-Funktionen, inkl. Logging der Aktionen, kein Command Line Interface***
  Die meisten ESB- und Systemmanagement-Funktionen sind mehr oder weniger gut in der Web-Konsole auffindbar. Aktionen werden geloggt. Ein ausreichend funktionierendes Command Line Interface existiert nicht.
- ***Schlechte Usability, keine Systemmanagement-Funktionen, kein Logging***
  Schlechte Aufbereitung der Managementkonsole und/oder keine integrierten Systemmanagement-Funktionen. Aktionen werden nicht geloggt.

3.3.4.2 LFK2 – Entwicklungsumgebung

Gibt es eine IDE im ESB für die Konfiguration der Adapter und des Nachrichtenflusses? Ist eine grafische Umgebung verfügbar?

Bewertung:

- ***IDE und gute grafische Unterstützung***
- ***IDE, aber keine grafische Unterstützung***
- ***Keine IDE, keine grafische Unterstützung***

3.3.4.3 LFK3 – Aufwand Installation / Konfiguration

Wie einfach gestaltet sich der Weg von der Installation bis zu einer laufenden Connection, auf welche eine Applikation zugreifen könnte? Wie simpel kann man eine Connection einrichten bzw. konfigurieren?

a) **Aufwand Installation**
   Das Produkt lässt sich einfach installieren, sodass keine grösseren Komplikationen auftreten.
b) **Aufwand Konfiguration**
   Ein simpler Use Case lässt sich schnell und unkompliziert einrichten.

Bewertung:

- ***Verständlich und simples Setup***
  Die Installation sowie die Konfiguration des Produkts sind verständlich aufgebaut und einfach einzurichten.
- ***Ausreichend verständlich, aber aufwändigeres Setup***
  Für die Installation und Konfiguration braucht man im Durchschnitt länger, jedoch grundsätzlich verständlich.
- ***Kompliziertes und anspruchsvolles Setup***
  Es ist sehr schwer, das Produkt zu installieren und eine Connection zu konfigurieren. Man muss sich zuerst durch die Dokumentation kämpfen.

3.3.5 Allgemeine Produktkriterien

3.3.5.1 AK1 – Configuration over Coding

Wie gut konfigurierbar ist der ESB? Lassen sich Nachrichten-Flow, Adapter, Endpoints und Transformationen konfigurieren? Welche Techniken (Konfiguration in einer Datei, als DSL, CLI oder in der Web-Konsole) werden für die Konfiguration dabei verwendet?

Bewertung:

- ***Umfassend konfigurierbar***
  Alles lässt sich konfigurieren, man muss nicht programmieren können, um die Funktionalität des ESB zu verändern. Die Komponenten lassen sich mindestens via Datei, CLI oder Web-Konsole konfigurieren.
- ***Teilweise konfigurierbar***
  Es lassen sich Teile des ESB konfigurieren, jedoch muss trotzdem programmiert werden. Für die Konfiguration muss eine herstellerspezifische DSL erlernt werden.
- ***Nicht konfigurierbar***
  Es ist nicht möglich, den ESB über Konfigurationsdateien einzustellen.

3.3.5.2 AK2 – HW-/SW-Anforderungen

Welche Hardwareanforderungen sind notwendig? Gibt es zusätzliche Software, die beschafft werden muss, z.B. eine nicht in die ESB-Installation integrierte DB?

Bewertung:

- ***Keine speziellen Anforderungen***
- ***Erweiterte HW-Anforderungen oder externe Software werden vorausgesetzt***
  Erweiterte HW-Anforderungen sind zwingend und/oder es wird zusätzlich vorinstallierte Software vorausgesetzt (z.B. externe DB)
- ***Erweiterte HW-Anforderungen und externe Software werden vorausgesetzt***

3.3.5.3 AK3 – Modularer Aufbau der Applikation

Wie modular ist der ESB aufgebaut? Ist es möglich, einen Teil bei der Lizenzierung und Installation auszuwählen?

a) **Auswahlmöglichkeiten bei der Lizenzierung**
   Bei der Lizenzierung ist es möglich, nur einen Teil zu lizenzieren. Dies erlaubt die Reduktion der Lizenzkosten.
b) **Auswahlmöglichkeiten bei der Installation**
   Bei der Installation ist es möglich, nur einen Teil zu installieren. Dies erlaubt es, die Installation schlank zu halten und die Betriebskosten zu reduzieren.

Bewertung:

- ***Komplett modularer Aufbau mit Teillizenzierung***
  Die Applikation ist modular aufgebaut und lässt somit Lizenzierung und Installation der einzelnen Teile zu.
- ***Teilweise modularer Aufbau mit Teillizenzierung***
  Die Applikation ist teilweise modular aufgebaut und es lassen sich mindestens einzelne Teile davon individuell lizenzieren.
- ***Kein modularer Aufbau und/oder keine Teillizenzierung***
  Die Applikation ist nicht modular aufgebaut und lässt somit keine Auswahl bei der Lizenzierung oder Installation zu.

3.3.5.4 AK4 – Kosten

Wie hoch sind die Kosten für die Lizenzierung der Software, inkl. allfälliger Kosten für notwendige Zusatzprodukte (z.B. DB, falls nicht integriert)?

Bewertung:

- ***Keine Kosten***
  Es entstehen keine Kosten beim Einsatz des Produkts, da das Produkt eine Open-Source-Lizenz benutzt und keine Zusatzprodukte benötigt.
- ***Moderate Kosten***
  Die Lizenzkosten sind moderat im Verhältnis zu vergleichbaren Produkten.
- ***Hohe Kosten***
  Die Lizenzkosten sind hoch im Quervergleich.

3.4 WSO2 Enterprise Service Bus

Dieser Abschnitt beschreibt das bewertete Produkt WSO2 Enterprise Service Bus. Aus Gründen der Einfachheit haben wir das Produkt in der Fortsetzung des Dokuments als WSO2 ESB bezeichnet.

Das Produkt haben wir für die Beurteilung am 6. Oktober 2014 in der Version 4.8.1 heruntergeladen und installiert. Als Installationsumgebung wählten wir eine virtuelle Maschine Windows 7 SP1 x64 mit allen aktuellen Windows Patches und dem Java SE Development Kit 7u67 aus.

Die Bewertung der Kriterien beruht auf Informationen aus der Online Dokumentation[4], Hinweisen aus der anwendungsinternen Hilfe und unserer persönlichen Erfahrung mit dem Produkt. Um das Produkt kennen zu lernen haben wir ein Tutorial[5] sowie einige Samples durchgespielt und den Use Case umgesetzt.

Die Informationen wurden ggf. aus den jeweiligen Webseiten [11] von WSO2 entnommen.

3.4.1 Produktbeschreibung

> *"We've taken a fresh look at old-style, centralized ESB architectures, and designed our unique WSO2 Enterprise Service Bus from the ground up as the highest performance, lowest footprint, and most interoperable SOA and integration middleware today. Relying on our innovative Carbon technology, the ESB delivers a smooth start-to-finish project experience that you cannot find anywhere else.*
> *The feature rich and standards compliant WSO2 ESB delivers high performance within a lean footprint. For example, a deployed WSO2 ESB often fits within a 160 MB memory space"*
>
> *[12]*

Zitat 2 Produktebeschreibung des Herstellers zum WSO2 Enterprise Service Bus

WSO2 ESB wird mit einem Apache Tomcat Applikations-Server mitgeliefert und unterstützt alle 61 Enterprise Integration Patterns [10].

WSO2 Carbon[6] ist ebenfalls ein Produkt von WSO2 und wird im WSO2 ESB verwendet.

[4] Die komplette Dokumentation ist einsehbar unter https://docs.wso2.com/display/ESB481/WSO2+Enterprise+Service+Bus+Documentation (zuletzt aufgerufen am 15.12.2014).

[5] WSO2 bietet online ein Einführungs-Tutorial an https://docs.wso2.com/display/ESB481/Tutorial (zuletzt aufgerufen am 15.12.2014).

[6] WSO2 Carbon http://wso2.com/products/carbon/ (zuletzt aufgerufen am 17.12.2014)

3.4.2 Architektur

Die logische Architektur (Message Flow des ESB) wird in der folgenden Grafik illustriert:

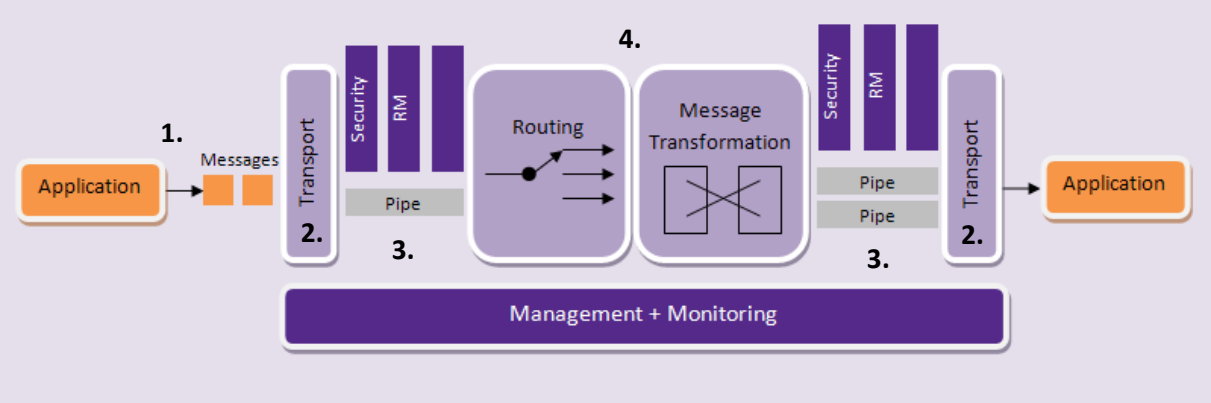


Abbildung 8 WSO2 Message-Architektur [11]

Die Übermittlung beinhaltet die folgenden Schritte:

1. Eine Applikation verschickt eine Nachricht an den ESB (Request oder Response).
2. Der ESB nimmt die Message mit Hilfe eines *Transports* entgegen.
3. Der Transport legt die Message nun auf eine Pipe, welche mit einer Apache Axis2 Engine (Einsatz von Apache CFX ist nicht möglich) realisiert ist und die Quality-of-Service(QoS)-Aspekte umsetzt. Dies betrifft den Eingangs- sowie Ausgangfluss. An dieser Stelle sind zwei Betriebsmodi anwendbar:
    - Mediating Messages – eine Single Pipe
    - Proxy Services – separate Pipes, welche den *Transport* zu verschiedenen *Proxy Services* verbinden
4. Nun wird die Message geroutet und transformiert. Die beiden Schritte können in beliebiger Reihenfolge durchgeführt werden. Es kann sowohl vor als auch nach dem Routing transformiert werden. Diese beiden für die Transformation zuständigen Units werden auch als *Mediation Framework* zusammengefasst und basieren auf der Apache Synapse[7] Implementation.
5. Anschliessend werden die Messages, abhängig von ihrer Destination, unter Berücksichtigung der QoS-Aspekte an die entsprechenden Pipes geschickt.
6. Zuletzt sorgt wieder der Transport dafür, dass die Message den ESB mit dem richtigen Transportprotokoll verlässt.

WSO2 verwendet dabei diverse Begriffe, die in der IT-Literatur nicht eindeutig verwendet werden. Die Definitionen dazu sind zu ungenau.

Da in der Bewertung des WSO2 ESB diese produktspezifischen Begriffe aber regelmässige Verwendung finden, folgt hier eine Erläuterung gemäss Hersteller.

[7] Apache Synapse ist der Name eines leichtgewichtigen und hoch performanten ESBs der Apache Software Foundation. Dieser ESB ist das Ergebnis einer engen Zusammenarbeit zwischen WSO2 und Apache. Viele frühere und aktuelle Mitarbeiter von WSO2 sind aktive Apache Comitters oder Project Management Committee (PMC) Mitglieder. Detaillierte Informationen sind unter http://wso2.com/apache/ verfügbar (zuletzt aufgerufen am 15.12.2014).

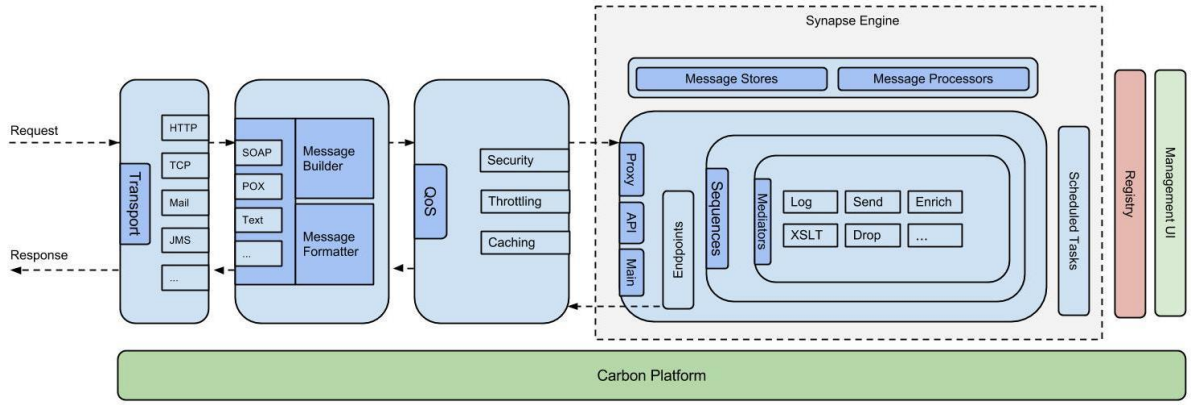


Abbildung 9 WSO2 Komponenten-Architektur [11]

Komponente	Beschreibung
Transport	Ein Transport leitet Messages in einem speziellen Format wie zum Beispiel HTTP/S, JMS, SMTP oder auch TCP weiter. Er besteht aus zwei Komponenten: • **Message Builder** Dieser identifiziert die Message anhand des Content Type und konvertiert sie ins XML-Format. Der Content kann in textbasiertem oder Binary-Format sein. • **Message Formatter** Der Formatter konvertiert die Messages beim Verlassen des ESB wieder von XML in das Originalformat zurück.
QoS-Komponenten	Es existieren diverse QoS-Module, um die zuverlässige Nachrichtenübermittlung sicherzustellen. Dazu gehören neben dem Security-Modul auch Message Relay und Caching.
Endpoints	Ein Endpoint definiert eine externe Destination für eine Message, vergleichbar mit einem Serviceendpunkt. Es gibt verschiedene Typen wie zum Beispiel den WSDL, HTTP, Address, Load Balancing oder Failover Endpoint. Diese sind unabhängig vom verwendeten Transport und können daher mehrfach gebraucht werden.
Sequences	Sequences sind die Konfigurationskomponenten des Mediators und ermöglichen diesem die Implementation von Pipes und Filtern.
Mediators	Als Mediators wird ein ganzer Katalog von Prozess-Einheiten bezeichnet, welche alle spezifischen Funktionen ausführen. Solche Funktionen können unter anderem Senden, Aufteilen, Filtern, Austauschen, Aggregieren oder Klonen von Messages sein. Diese Einheiten ermöglichen auch das Implementieren der Enterprise Integration Patterns.
Scheduled Tasks	Scheduled Tasks ermöglichen das Planen und Ausführen von Jobs für ESB-interne und -externe Mediation-Operationen.

Tabelle 6 WSO2 Komponentendefinitionen [11]

3.4.3 Produktbewertung

In den anschliessenden Abschnitten wird der WSO2 ESB an Hand der in Kapitel 3.3 zusammengestellten Bewertungskriterien überprüft und beurteilt.

3.4.3.1 Bewertung der funktionalen Kriterien

3.4.3.1.1 FK1 – Unterstützte Protokolle/Technologien und Adapter

WSO2 bietet mehrere Transports zur Nachrichtenübermittlung, Konnektoren zu Third-Party Services und Unterstützung für ganze Third-Party-Adapter an.

Für unsere Bewertung interessant ist jedoch nur eine Handvoll dieser Optionen.

WSO2 Transport	Beschreibung und unterstützte Protokolle
HTTP/HTTPS Servlet	Diese beiden Transporter unterstützen die Übertragung von Messages. **SOAP** (Version 1.1 und 1.2) Messages sowie Messages via **RESTful** HTTP/HTTPS können übertragen werden.
VFS	Dieser Transporter basiert auf der „Commons Virtual File System“ API von Apache und wird dazu verwendet, Files aus einem Local- oder Remote-File-System zu lesen und zu schreiben. Die dabei unterstützten Protokolle sind **SFTP**, **FTP,** FTPS (FTP over SSL), ZIP/TAR/GZ, File, WebDAV, CIFS und einige Weitere[8], welche nicht ganz so populär sind. Unter anderem wird er auch dazu benutzt, um mit Hilfe des Datenbank-Mediators auf eine Datenbank zu schreiben und zu lesen.
MailTo	Unterstützt wir damit das Versenden von E-Mails via **Simple Mail Transfer Protocol (SMTP)** sowie das Empfangen via **Post Office Protocol (POP3)** oder **Internet Message Access Protocol (IMAP)**.
TCP	**TCP** Transporter können für Übertragung von **SOAP** Messages verwendet werden.
Adapters zu Third-Party Services	Neben den Transportern sind auch vorkonfigurierte Konnektoren zu den Cloud Services JIRA, Google Spreadsheet, Salesforce, Twitter und Twilio verfügbar. Ausserdem lassen sich weitere Mediators Plug&Play-ähnlich implementieren, wie zum Beispiel die von SAP oder PayPal angebotenen Standard-Adapters.

[8] eine komplette Liste ist zu finden unter http://commons.apache.org/proper/commons-vfs/filesystems.html (zuletzt aufgerufen am 20.10.2014).

Weitere unterstützte Protokolle/Technologien	• FIX (Financial Information eXchange) • HL7 (Health Level 7 International) • HTTP PassThrough, HTTP-NIO, HTTPS-NIO • JMS (Java Message Service) – unterstützt werden alle JMS Provider sowie verschiedenen Message Brokers im Parallelbetrieb. • Local Transport (Message-Austausch lokal auf dem ESB Server, u.a. in Verbindung mit einem Proxy Service) • Multi-HTTPS • MSMQ • UDP • RabbitMQ AMQP

Tabelle 7 WSO2 Transports: Unterstützte Protokolle und Technologien

Bewertung:
Positive Bewertung (+): Standardprotokolle plus zusätzliche Adapter/Konnektoren zu Third-Party Services

3.4.3.1.2 FK2 – Custom Adapter Framework

Um den WSO2 ESB zu erweitern, gibt es mehrere Möglichkeiten. Einerseits werden die Entwicklung und die Implementierung sogenannter Artefakte unterstützt, andererseits können Mediators selbst entwickelt und angeschlossen werden. WSO2 bietet dazu eine Entwicklungsumgebung an. Diese basiert auf Eclipse Java EE IDE for Web Developers (Version: Kepler Service Release 2).

Es können Endpoints, Sequence Templates, Sequences, Proxy Services, Priority Executors, Message Stores, Message Processors, APIs sowie Mediators und Scheduled Tasks als Artefakte entwickelt und einfach via Upload-Funktion in den ESB integriert werden. Grundsätzlich kann in Java programmiert werden, es werden aber auch die Spring und Apache Scripting Frameworks unterstützt.

Im Development Studio gibt es für alle Objekte eine Design View, womit die Properties praktisch und einfach geändert werden können. Mittels Wizard können neue Artefakte erstellt oder kopiert werden. Ausserdem ist der Building-Prozess mittels Apache Maven direkt integriert und es besteht die Möglichkeit zur einfachen Verwendung eines Command Line Tools wie Jenkins.

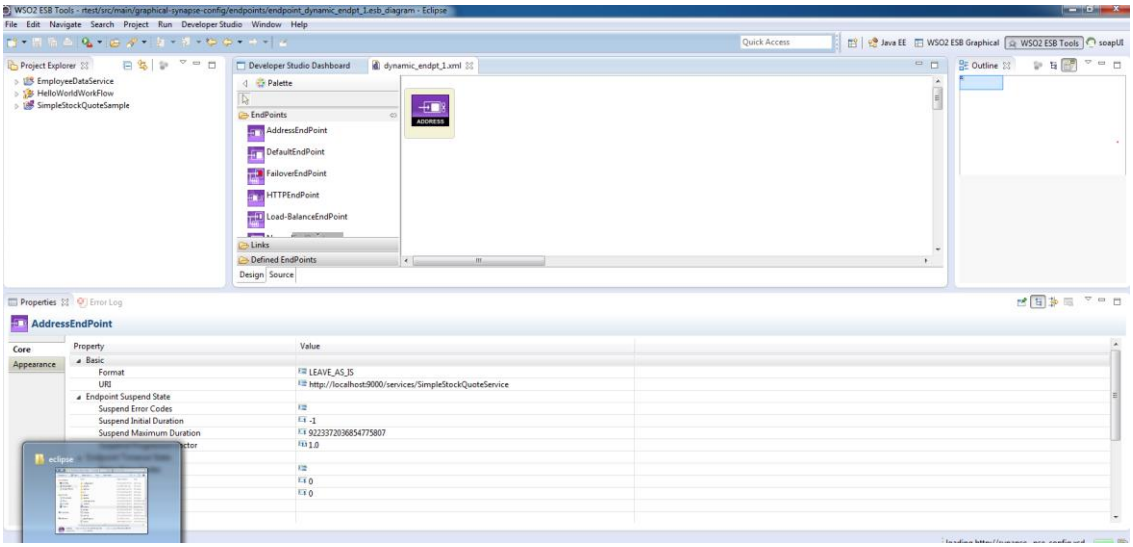


Abbildung 10 WSO2 Developer Studio: Configure properties of an AddressEndPoint

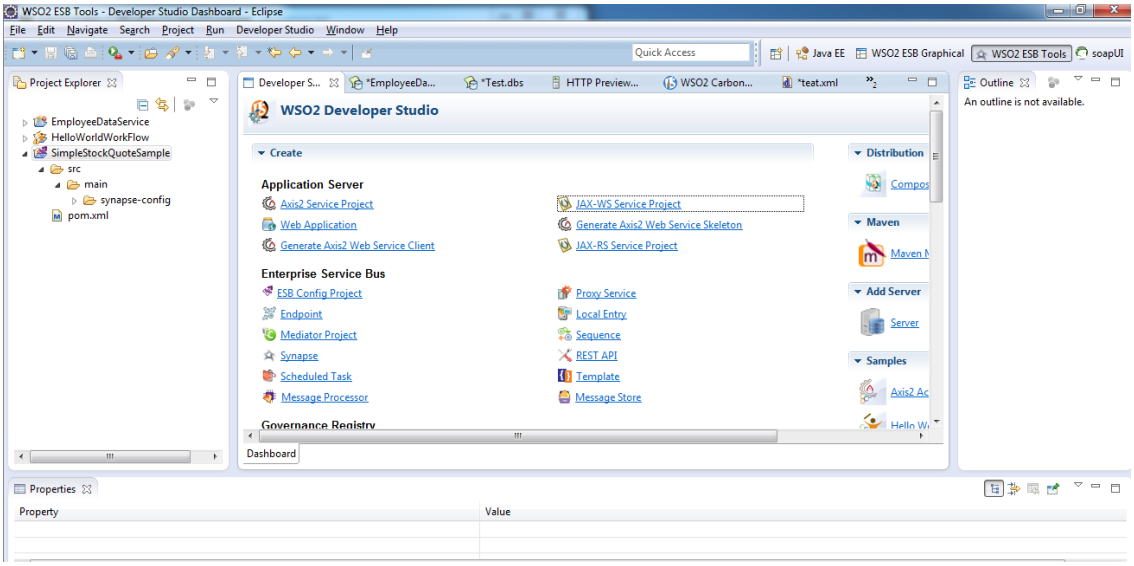


Abbildung 11 WSO2 Developer Studio Dashboard

Bewertung:
Positive Bewertung (+): Gute Unterstützung
Standardprotokolle plus zusätzliche Adapter/Konnektoren zu Third-Party Services

---

Anmerkung: Ein „Adapter“ besteht bei WSO2 aus verschiedenen Teilen, weshalb bei der Bewertung auf diverse Artefakte verwiesen wird.

3.4.3.1.3 FK3 – Security [FA3, FA4]

WSO2 ESB implementiert die grundlegenden Security-Konzepte von WS-Security-, WS-Policy- und WS-Security-Policy-Spezifikationen. Dadurch ist der komplette X.509-Zertifikatstandard unterstützt. Der Authentication-Prozess wird auch mittels SAML 1.1 oder 2.0, Service as „Security Token Services“ (STS) und Bootstrap Policy unterstützt. Applikationsintern werden zwei verschiedene Key Store Typen unterstützt: JKS (Java Key Store) und PKCS12 (Public Key Cryptography Standards) [13].

a) Authentication

Es werden alle Authentifizierungsmethoden gemäss WS-Security unterstützt. Darunter gehört unter anderem die Anmeldungsverifizierung mittels Benutzername und Passwort, aber auch X.509-Zertifikate.

Eine Lösung für die Problematik mit den technischen Usern wird nicht angeboten. Dieses Problem muss vollumfänglich in den Client- und/oder Recipient-Applikationen gehandhabt werden.

b) Authorization

Für die Authorization werden der applikationsinterne User Store und Key Store verwendet. Alle Zertifikate und Benutzer-Credentials werden bereits dort überprüft.

c) Confidentiality

Die Confidentiality kann ebenfalls mittels X.509-Zertifikat auf Clients- und Recipient-Seite erfolgen. Besitzt der Client jedoch kein Zertifikat, kann alternativ die Verschlüsselung mit einem vom Client zur Verfügung gestellten symmetrischen Key erfolgen. Dieser Key wird mit dem Public Key vom Service verschlüsselt und innerhalb der SOAP-Nachricht mitübermittelt. Der Service kann dann mit seinem Private Key zuerst den Schlüssel und anschliessend mit dem Schlüssel die Nachricht entschlüsseln.

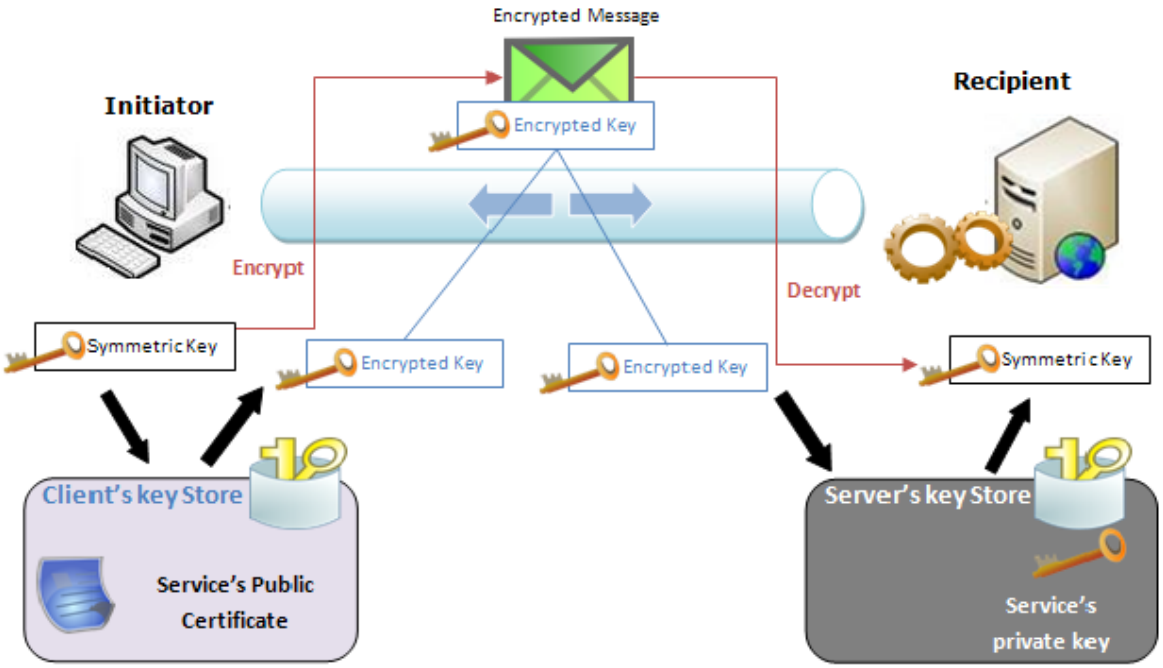


Abbildung 12 WSO Prozess für das Verschlüsseln mit Zertifikaten [13]

d) Data Integrity

Auch die Datenintegrität kann primär mit X.509-Zertifikaten abgedeckt werden. Jedoch wird auch hier die Option unterstützt, dass der Client kein Zertifikat besitzt. In diesem Fall erfolgt das Signieren analog dem Verschlüsseln ohne Zertifikat: signieren durch Client mit eigenem Key, welcher mittels Service Public Key verschlüsselt innerhalb der SOAP-Nachricht übermittelt wird.

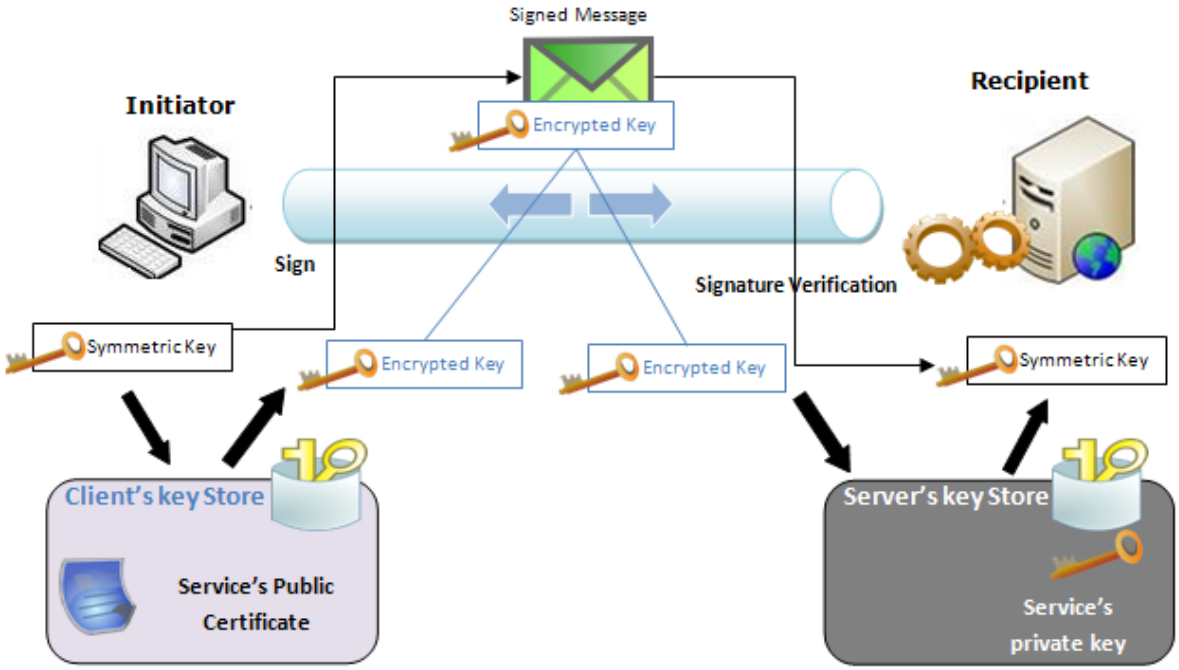


Abbildung 13 WSO2 Prozess für das Signieren mit Zertifikaten [13]

Bewertung:

Positive Bewertung (+): Hoher Security-Standard

Das Produkt bietet in allen vier Bereichen eine Security-Lösung an, welche die heutigen Standards übertrifft. Dies wird vor allem mit der Zertifikatsunterstützung sichergestellt. Wie einfach die Security-Aspekte zu realisieren sind, wird in dieser Bewertung jedoch nicht berücksichtigt, mangels Erfahrung mit dem Produkt. Es werden aber ausreichend Möglichkeiten geboten, die Kommunikation nach bestem Gewissen abzusichern.

3.4.3.1.4 FK4 – Message Translation

Für die Message Translation werden sogenannte Message Builders und Message Formatters eingesetzt. Beim Eingang einer Nachricht werden die rohen Payload-Daten verarbeitet und anhand ihres Content Type in SOAP konvertiert. Beim Verlassen des ESB kommt ein Message Formatter zum Einsatz.

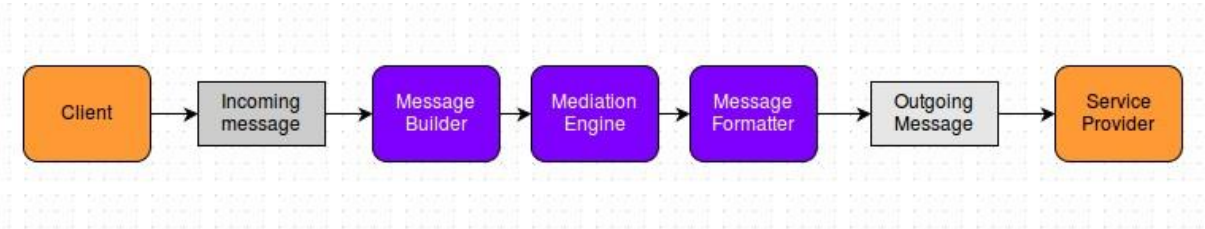


Abbildung 14 Verarbeitung der Nachrichten im WSO2 ESB [14]

Diese Verarbeitungsblocks lassen sich einfach via Konfigurationsfile aktivieren/deaktivieren.

```
    <!-- ================================================= -->
    <!-- Message Formatters -->
    <!-- ================================================= -->
    <!-- Following content type to message formatter mapping can be used to implement support -->
    <!-- for different message format serializations in Axis2. These message formats are -->
    <!-- expected to be resolved based on the content type. -->
- <messageFormatters>
    <messageFormatter class="org.apache.axis2.transport.http.XFormURLEncodedFormatter" contentType="application/x-www-form-urlencoded"/>
    <messageFormatter class="org.apache.axis2.transport.http.MultipartFormDataFormatter" contentType="multipart/form-data"/>
    <messageFormatter class="org.apache.axis2.transport.http.ApplicationXMLFormatter" contentType="application/xml"/>
    <messageFormatter class="org.apache.axis2.transport.http.SOAPMessageFormatter" contentType="text/xml"/>
    <messageFormatter class="org.apache.axis2.transport.http.SOAPMessageFormatter" contentType="application/soap+xml"/>
    <messageFormatter class="org.apache.axis2.format.PlainTextFormatter" contentType="text/plain"/>
        <!--JSON Message Formatters-->
        <!--messageFormatter contentType="application/json" class="org.apache.synapse.commons.json.JsonFormatter"/-->
    <messageFormatter class="org.apache.synapse.commons.json.JsonStreamFormatter" contentType="application/json"/>
        <!--messageFormatter contentType="application/json" class="org.apache.axis2.json.JSONMessageFormatter"/-->
        <!--messageFormatter contentType="application/json" class="org.apache.axis2.json.JSONStreamFormatter"/-->
    <messageFormatter class="org.apache.axis2.json.JSONBadgerfishMessageFormatter" contentType="application/json/badgerfish"/>
        <!--messageFormatter contentType="application/x-www-form-urlencoded" class="org.wso2.carbon.relay.ExpandingMessageFormatter"/-->
        <!--messageFormatter contentType="multipart/form-data" class="org.wso2.carbon.relay.ExpandingMessageFormatter"/-->
        <!--messageFormatter contentType="application/xml" class="org.wso2.carbon.relay.ExpandingMessageFormatter"/-->
        <!--messageFormatter contentType="text/html" class="org.wso2.carbon.relay.ExpandingMessageFormatter"/-->
        <!--messageFormatter contentType="application/soap+xml" class="org.wso2.carbon.relay.ExpandingMessageFormatter"/-->
        <!--messageFormatter contentType="text/xml" class="org.wso2.carbon.relay.ExpandingMessageFormatter"/-->
        <!--messageFormatter contentType="x-application/hessian" class="org.apache.synapse.format.hessian.HessianMessageFormatter"/-->
        <!--messageFormatter contentType="" class="org.apache.synapse.format.hessian.HessianMessageFormatter"/-->
        <!--messageFormatter contentType="application/edi-hl7" class="org.wso2.carbon.business.messaging.hl7.message.HL7MessageFormatter"/-->
  </messageFormatters>
    <!-- ================================================= -->
    <!-- Message Builders -->
    <!-- ================================================= -->
    <!-- Following content type to builder mapping can be used to implement support for -->
    <!-- different message formats in Axis2. These message formats are expected to be -->
    <!-- resolved based on the content type. -->
- <messageBuilders>
    <messageBuilder class="org.apache.axis2.builder.ApplicationXMLBuilder" contentType="application/xml"/>
    <messageBuilder class="org.apache.synapse.commons.builders.XFormURLEncodedBuilder" contentType="application/x-www-form-urlencoded"/>
    <messageBuilder class="org.apache.axis2.builder.MultipartFormDataBuilder" contentType="multipart/form-data"/>
    <messageBuilder class="org.apache.axis2.format.PlainTextBuilder" contentType="text/plain"/>
        <!--JSON Message Builders-->
        <!--messageBuilder contentType="application/json" class="org.apache.synapse.commons.json.JsonBuilder"/-->
    <messageBuilder class="org.apache.synapse.commons.json.JsonStreamBuilder" contentType="application/json"/>
        <!--messageBuilder contentType="application/json" class="org.apache.axis2.json.JSONBuilder"/-->
        <!--messageBuilder contentType="application/json" class="org.apache.axis2.json.JSONStreamBuilder"/-->
    <messageBuilder class="org.apache.axis2.json.JSONBadgerfishOMBuilder" contentType="application/json/badgerfish"/>
        <!--messageBuilder contentType="application/xml" class="org.wso2.carbon.relay.BinaryRelayBuilder"/-->
        <!--messageBuilder contentType="application/x-www-form-urlencoded" class="org.wso2.carbon.relay.BinaryRelayBuilder"/-->
        <!--messageBuilder contentType="multipart/form-data" class="org.wso2.carbon.relay.BinaryRelayBuilder"/-->
        <!--messageBuilder contentType="multipart/related" class="org.wso2.carbon.relay.BinaryRelayBuilder"/-->
        <!--messageBuilder contentType="application/soap+xml" class="org.wso2.carbon.relay.BinaryRelayBuilder"/-->
        <!--messageBuilder contentType="text/plain" class="org.wso2.carbon.relay.BinaryRelayBuilder"/-->
        <!--messageBuilder contentType="text/xml" class="org.wso2.carbon.relay.BinaryRelayBuilder"/-->
        <!--messageBuilder contentType="x-application/hessian" class="org.apache.synapse.format.hessian.HessianMessageBuilder"/-->
        <!--messageBuilder contentType="" class="org.apache.synapse.format.hessian.HessianMessageBuilder"/-->
        <!--messageBuilder contentType="application/edi-hl7" class="org.wso2.carbon.business.messaging.hl7.message.HL7MessageBuilder"/-->
  </messageBuilders>
```

Code 3 WSO2 Message-Formatter-/Message-Builder-Konfiguration

Nach Herstellerangaben kann der WSO2 ESB Daten im Binary-, CSV-, JSON- und XML-Format verarbeiten.

Zusätzlich besteht die Möglichkeit, durch Message Relay die Nachrichten eines bestimmten Content Type ohne Konvertierung durch den ESB zu transportieren.

Bewertung:
Positive Bewertung (+): Gute Unterstützung
Alle vier Formate werden unterstützt. Zusätzlich können beliebige Formate via Message Relay durchgeschleust werden.

3.4.3.1.5 FK5 – Datenstrukturtiefe

Es werden mehrfach verschachtelte Strukturen unterstützt. Bei einem Test mit 100 Verschachtelungen konnte der WSO2 ESB die Filterung mit XSLT erfolgreich durchführen.
Die dazu verwendete XML-Datei sowie die Konfiguration sind als Anhang im Dokument aufgeführt.

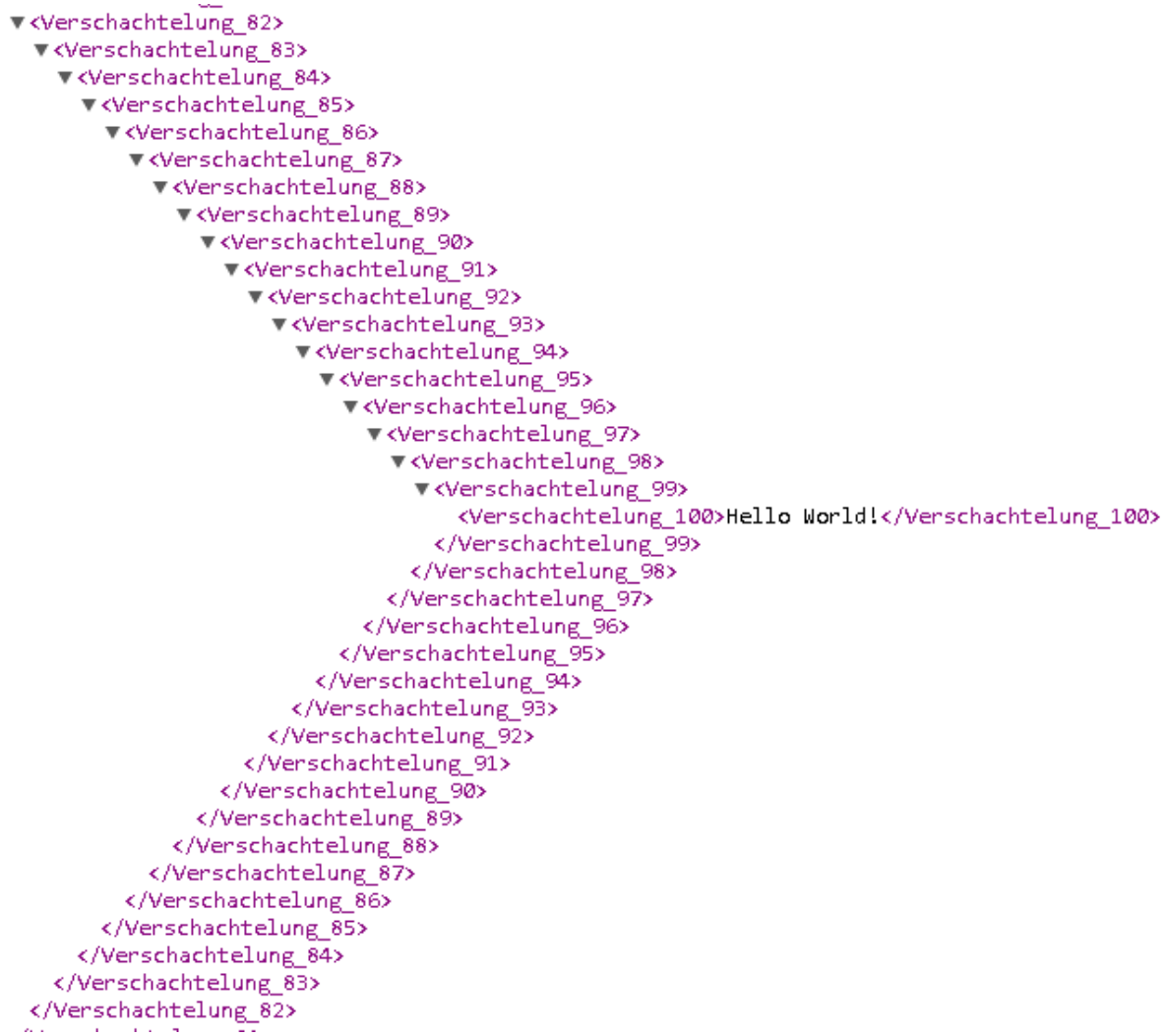


Abbildung 15 Ausschnitt aus dem XML-Dokument mit 100-facher Verschachtelung

Das Resultat wird richtig angezeigt

```
{
  - export: {
      - person: {
            login: "paolo",
            contractStart: "1990-04-01",
            contractEnd: null,
            fk5: "Hello World!"
        }
    }
}
```

Abbildung 16 100-faches XML nach einer XSLT Filterung

Bewertung:
Positive Bewertung (+): Mehrfach verschachtelte Strukturen

3.4.3.1.6 FK6 – Message-Durchsatz / Concurrent Streams

Die Messages durchlaufen im WSO2 ESB diverse Stufen. Daher können sich auch jederzeit mehrere Messages gleichzeitig im ESB befinden. Der Message-Durchsatz kann daher nicht eindeutig beurteilt werden. Allerdings gibt es die folgenden Fakten, die darauf hinweisen, dass Concurrent Streams unterstützt werden:

- Es werden bis zu 1000 Concurrent-Non-Blocking-HTTP-/HTTPS-Verbindungen per ESB Server unterstützt
- In Message Stores können Nachrichten zwischengespeichert werden, um sie beispielsweise zu einem bestimmten Zeitpunkt zu versenden
- Es können mehrere Message Processors an einen Message Store angeschlossen werden. Der Message Sampling Processor kann auch mehrere Messages zur gleichen Zeit aus dem Store holen (via Parameter konfigurierbar)
- Der Message Iterator erlaubt, eine Nachricht zu splitten und die einzelnen Teile parallel zu verarbeiten

---

Bewertung:
Positive Bewertung (+): Erweiterte synchrone Verarbeitung
Es können mehr als drei Messages gleichzeitig verarbeitet werden.

3.4.3.1.7 FK7 – Betreibbarkeit in Cloud

Die IDEAL-Kriterien lassen sich nur teilweise auf das Produkt anwenden (siehe Auflistung unten). Trotzdem ist der ESB eindeutig tauglich für einen Betrieb in der Cloud. WSO2 bietet das Produkt auch in ihrer eigenen Cloud als ESB-as-a-Service an, somit gibt es auch bereits praktische Erfahrungen damit. Des Weiteren bietet der WSO2 ESB einen integrierten Cloud Service Gateway (CSG). Dieser erlaubt, die ESB-Funktionen ohne grossen Aufwand durch eine Firewall in anderen (z.B. public) Netzwerken anzubieten. Realisiert wird dies durch einen Proxy Service mit der CSG-Komponente, welche direkt auf dem ESB und als Frontend für das private Netzwerk läuft.

a) **Isolated State** ✓
   Der WSO2 ESB ist gemäss Herstellerangaben stateless. [12]
b) **Distributed** ✗
   Das Produkt lässt sich nicht ohne weiteres in verteilte Komponenten aufteilen und betreiben.
c) **Elastic – Scale Out / Scale Up** ✓
   Das Produkt lässt sich sehr gut skalieren, vor allem in der Scale-Out-Variante. Mehrere Instanzen können via Load Balancer zu einem Cluster gebündelt werden. Ausserdem gibt es in der Konfiguration Performance-Parameter, die helfen, beim Scale Up das Produkt entsprechend zu skalieren, beispielsweise die Anzahl Threads für die Transports und diverse Memory(RAM)-Werte.

d) Automated Management ✗
Die Konfiguration manifestiert sich in Dateien. Es existiert kein Command Line Interface (CLI) zur Administration der Konfiguration. Somit ist Automated Management schwer zu erreichen und dementsprechend nicht erfüllt.

e) Loosely Coupled ✓
Komponenten sind lose gekoppelt im ESB. Es laufen beispielsweise eigene Services mit eigenen Endpunkten, die einander nicht tangieren oder blockieren. Ausserdem können einzelne Artefakte jederzeit hinzugefügt, entfernt oder gewartet werden, ohne dass der ganze ESB dadurch funktionsunfähig wird.

Bewertung:
Neutrale Bewertung (o): Stateless und Cloud-Erfahrung vorhanden
Das Produkt lässt sich problemlos in der Cloud betreiben. Trotzdem konnten nicht alle IDEAL-Kriterien (D, A) eindeutig erfüllt werden, weshalb die Bewertung nur durchschnittlich ausfällt.

3.4.3.1.8 FK8 – Discovery and Registry

Keine Bewertung für dieses Kriterium.

3.4.3.1.9 FK9 – Workflow Engine and Service Composition

Keine Bewertung für dieses Kriterium.

3.4.3.1.10 FK10 – Versionierung der Komponenten

Es wird keine Versionierung der Komponenten in der Web-Konsole unterstützt.

Es ist jedoch möglich, in der Entwicklungsumgebung die jeweilige Konfiguration als Artefakt zu exportieren. Sie kann anschliessend in der Web-Konsole importiert werden. Somit ist es möglich, eine neue Version vorzubereiten, ohne die aktuelle Konfiguration verändern zu müssen. Es wäre denkbar, die Artefakte nach Versionen zu archivieren, um somit eine eigene Versionierung implementieren zu können.

Bewertung:
Negative Bewertung (–): Keine Versionierung unterstützt

3.4.3.2 Bewertung der Design-Kriterien

3.4.3.2.1 DK1 – Message Routing Patterns

WSO2 legt viel Wert auf die Enterprise Integration Patterns. Daher stellt der Hersteller auch einen kompletten Implementation Guide für alle einzelnen Patterns zur Verfügung. Gemäss diesem werden alle vorhandenen Message Routing Patterns unterstützt. Ein Auszug der Message Routing Patterns befindet sich im Anhang[9].

Bewertung:
Positive Bewertung (+): Alle sechs Patterns beliebig kombinierbar und dynamisches Routing

---

3.4.3.2.2 DK2 – Message Content Transformation Pattern

WSO2 unterstützt alle sechs Enterprise Integration Patterns für Message Content Transformation: Envelope Wrapper, Content Enricher, Content Filter, Claim Check, Normalizer und Canonical Data Model [15].

---

[9] Der Auszug kann auch online abgerufen werden unter folgender URL (zuletzt aufgerufen am 24.10.2014): https://docs.wso2.com/display/IntegrationPatterns/Message+Routing

Message Content Transformation Pattern	**Beschreibung**
Content Enricher:	Mittels Enrich Mediator können Messages um Daten „erweitert“ werden. Die Daten können beispielsweise in einem einfachen Textfile hinterlegt sein. Alternativ kann auch ein XSLT, XQuery oder Smooks Mediator dazu verwendet werden.
Content Filter:	Mittels XSLT, XQuery und Smooks[10] Mediator können Transformationen auf einzelne Elemente einer Message angewendet werden. Mediator Eigenschaften: • Standardmässig werden die als Key definierten Elemente im SOAP Body gesucht und transformiert • Es können überflüssige Daten aus der Message gelöscht werden • Der Mediator wird ausschliesslich in XSLT, XQuery oder Smooks erstellt, folglich können nur XML Files bearbeitet werden

Tabelle 8 Von WSO2 ESB unterstützte Message Content Transformation Patterns

Bewertung:
Positive Bewertung (+): Content Enricher und Content Filter werden unterstützt. Die Modellierungssprache ist standardisiert und somit nicht herstellerspezifisch

XSLT wird als Modellierungssprache angeboten.

3.4.3.2.3 DK3 – Message Consumer Patterns [FA2]

Wie die Message Routing Patterns sind die Message Consumer Patterns im EIP Implementation Guide ausführlich beschrieben und im Anhang sowie über die folgende Website zu finden:

https://docs.wso2.com/display/IntegrationPatterns/Messaging+Endpoints
zuletzt aufgerufen am 24.10.2014

Bewertung:
Positive Bewertung (+): Alle 8 Patterns werden unterstützt

3.4.3.3 Bewertung der Support- und Troubleshoot-Kriterien

3.4.3.3.1 STK1 – Reporting and Statistic

Mit dem implementierten log4j Framework kann jede Komponente im ESB geloggt werden. Es werden System Logs, Access Logs (u.a. Management-Konsolenzugriffe) und Application Logs (z.B. laufende Webservices und Applikationen) fortlaufend geschrieben. Diese sind bequem über die Management-Konsole zugreif- und auswertbar.

[10] Smooks ist ein Java-basiertes Framework für die Manipulation und Transformation von XML. Detailliertere Informationen sind auf der Hersteller-Website http://www.smooks.org verfügbar.

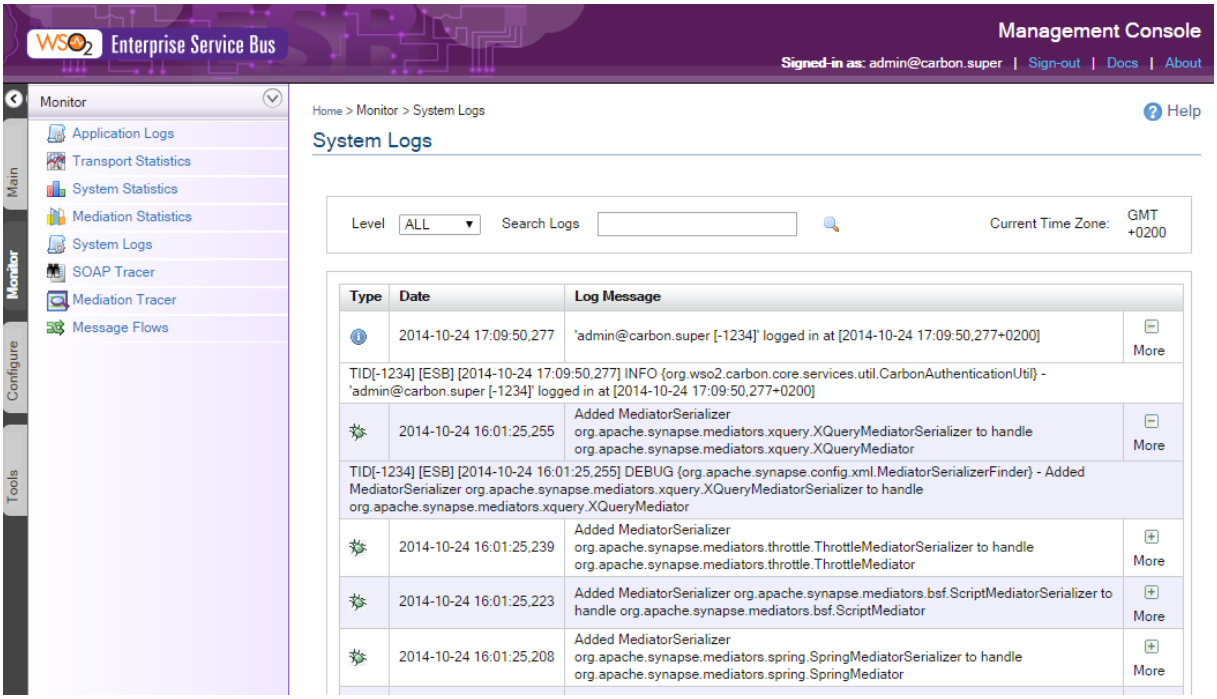


Abbildung 17 WSO2 System Log View

Der Message Flow kann in Echtzeit in den einzelnen Komponenten beobachtet werden. Dazu stehen die folgenden vier Kanäle zur Verfügung:

Abbildung 18 WSO 2 Message Flows

Ebenfalls können die fliessenden SOAP Messages aufgezeichnet und gefiltert werden:

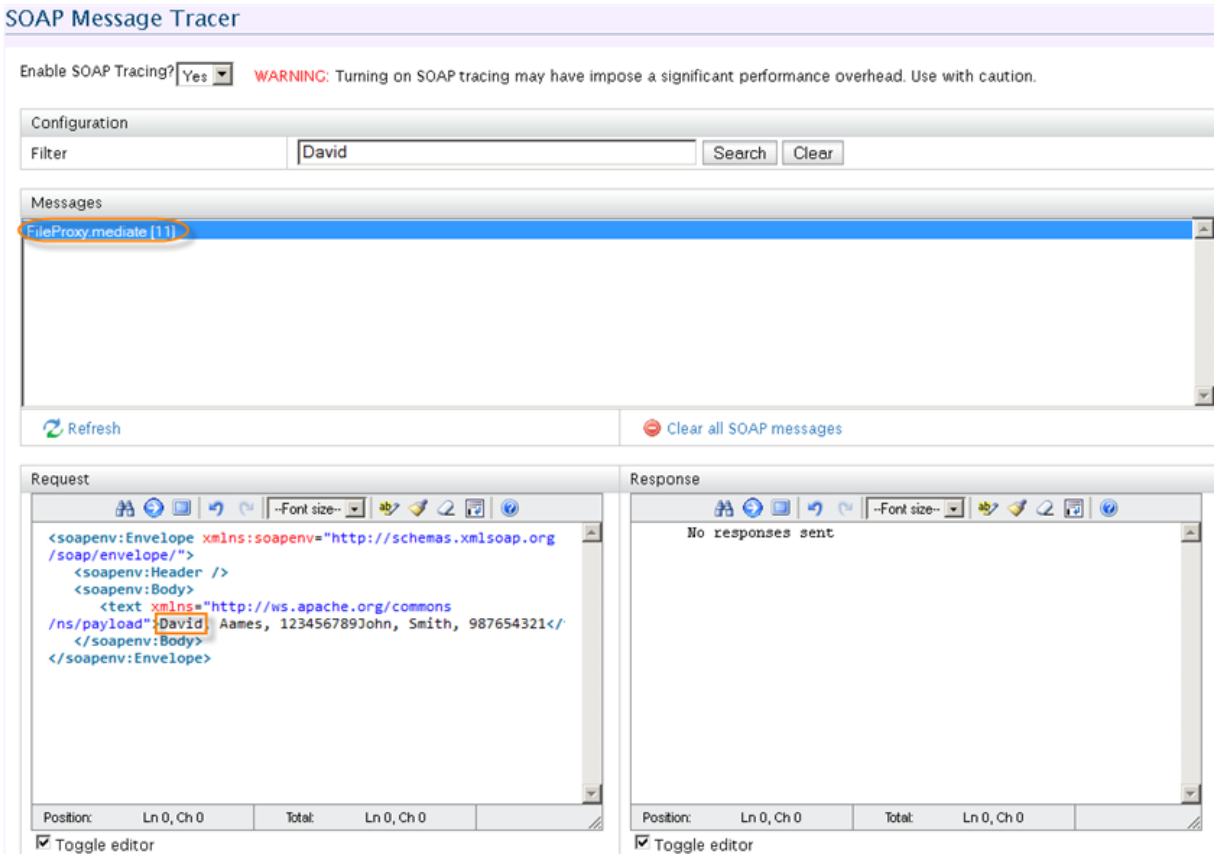


Abbildung 19 WSO2 SOAP Message Tracer

Auch Transport-, System- und Mediation-Statistiken können direkt in der Management-Konsole aufgerufen werden. Alle Services haben zudem ihre eigene Statistik, welche jeweils separat konfiguriert und aktiviert werden kann.

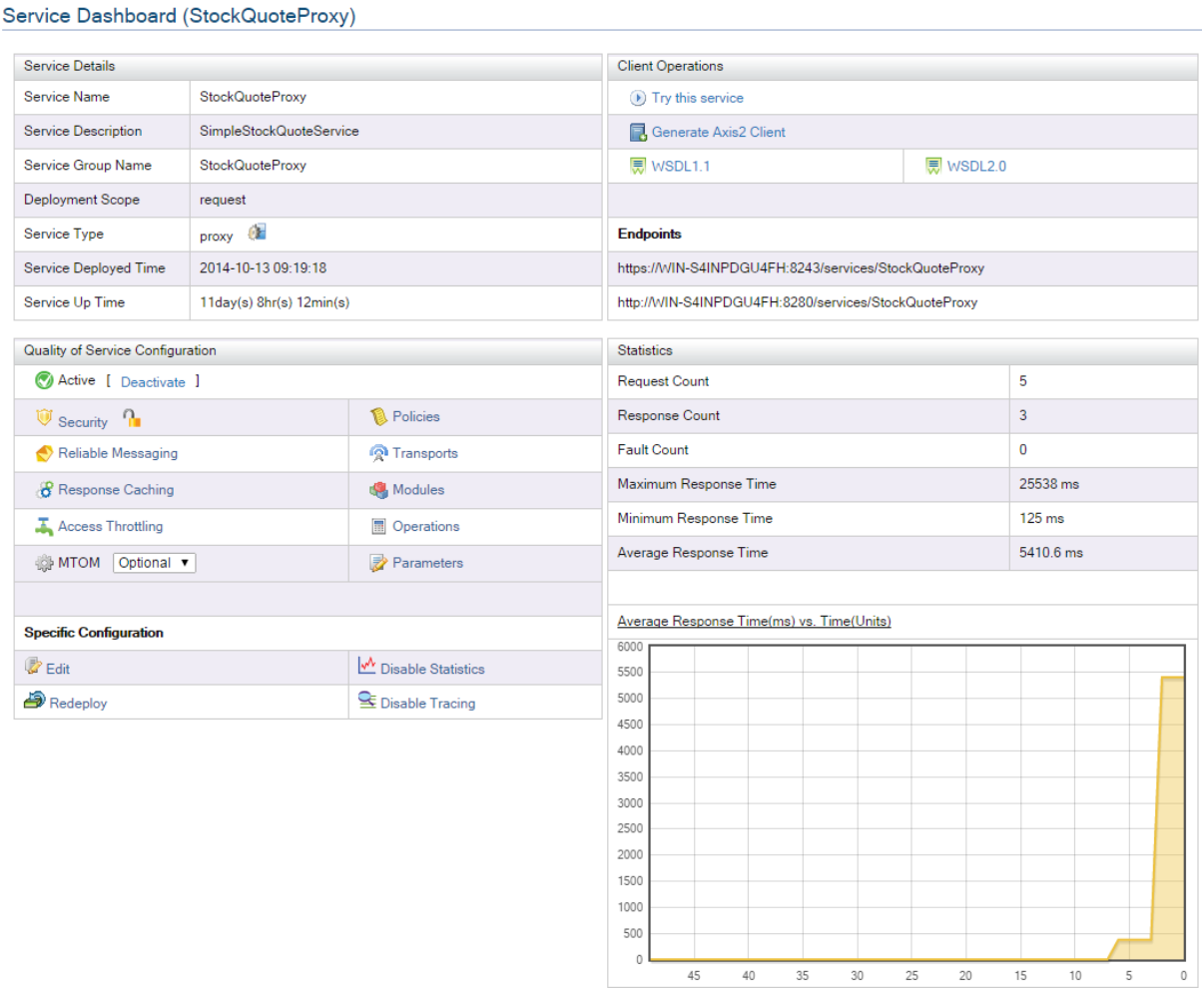


Abbildung 20 WSO2 Service Dashboard mit Statistik

Bewertung:
Positive Bewertung (+): Message Flow und Konfigurationsänderungen, nützliche Aufbereitung

3.4.3.3.2 STK2 – Alerting and Monitoring

WSO2 ESB unterstützt vor allem Java Management Extensions (JMX) und Simple Network Management Protocol (SNMP) für das Monitoring. Ein Alerting konnte jedoch nicht gefunden werden. Alternativ könnte man mit SNMP Traps oder mit Komponenten aus WSO2 ESB Anhang einer Anleitung aus dem Internet [11]ein Alerting konfigurieren.

3.4.3.3.2.1 JMX

Management-Ressourcen können via MBeans überwacht und in einer JConsole beobachtet werden. Beim Starten des WSO2 ESB Servers wird die JMX URL im Log ausgegeben. Es gibt bereits vordefinierte MBeans für Transport, Latenz, NttpConnections, Threads und Proxy Service.

[11] Alerting Nachbau mit WSO2 ESB Komponenten https://buddhimawijeweera.wordpress.com/2012/11/11/basic-alert-system-with-wso2-esb/ (zuletzt Aufgerufen am 17.12.2014).

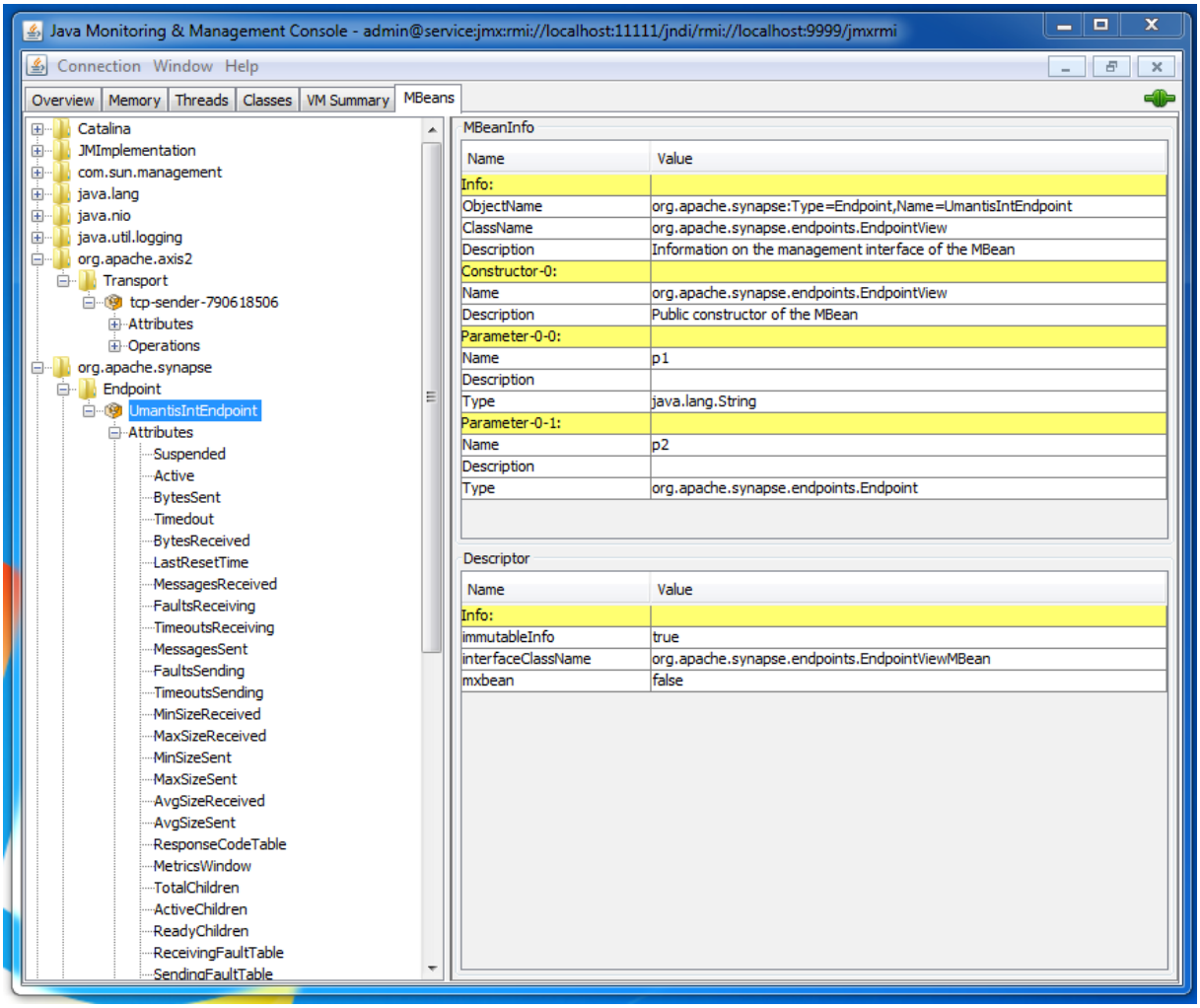


Abbildung 21 MBean für einen konfigurierten Endpoint

Es können vor allem Statistiken, welche auch in der Web-Konsole sichtbar sind, überwacht werden.

3.4.3.3.2.2 SNMP

Alternativ zur JConsole können MBeans via SNMP überwacht werden. Hierzu muss zuerst SNMP gemäss Dokumentation [16] aktiviert werden.

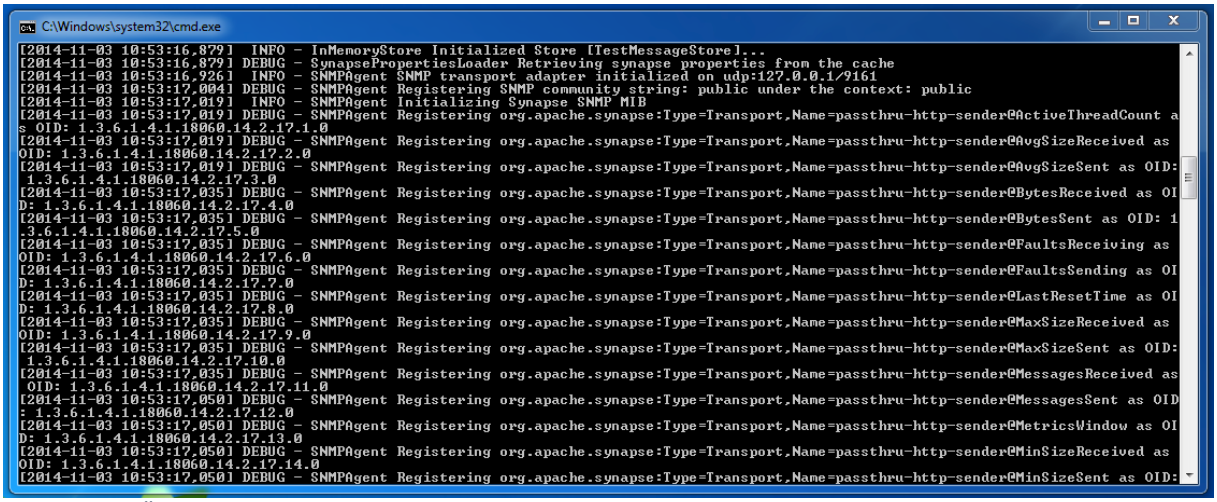


Abbildung 22 Überwachte MBeans via SNMP

Bewertung:
Positive Bewertung (+): Alerting und Monitoring

3.4.3.3.3 STK3 – Failover / Load Balancing

Der komplette WSO2 ESB lässt sich mit Hilfe eines Third-Party Load Balancer problemlos clustern. Dazu bietet der Hersteller auch selber ein Produkt an, welches sich WSO2 Elastic Load Balancer nennt. Es ermöglicht die Unterteilung von Manager und Worker Nodes, welche jeweils im Aktiv/Aktiv- oder Aktiv/Passiv-Modus betrieben werden können. Dadurch wird ein Failover und Load Balancing zwischen mehreren ESBs ermöglicht. Weitere Informationen sind in der Dokumentation [17] verfügbar.

Der ESB selbst bietet jedoch die Möglichkeit, Load Balance oder Failover Endpoints zu erstellen. Die effektiven Producer können auf beliebigen Nodes verteilt sein. Der Verlust von Nachrichten oder eine doppelte Zustellung kann somit verhindert werden.

Endpoint	**Beschreibung**
Load Balance Endpoint:	Es gibt einen dynamischen und einen statischen Load Balance Endpoint. Beide ermöglichen, nach dem Round-Robin-Prinzip Anfragen auf ein Set von Endpoints zu verteilen. Optional kann auch hier schon ein Property „Failover“ konfiguriert werden. Dieses sorgt dafür, dass Nachrichten automatisch an den nächsten Endpoint gesendet werden, falls der zuerst gewählte ausfällt. Die Ermittlung der beteiligten Endpoints geschieht entweder durch eine statische Konfiguration oder dynamisch mit Hilfe eines zusätzlichen Parameters auf den Endpoints.
Failover Endpoint:	Dieser Endpoint funktioniert nach dem Primary-/Slave-Prinzip. Aus einer Liste mit verfügbaren Endpoints wird der erste als Primary gewählt. Alle Anfragen werden ausschliesslich an den Primary Endpoint weitergeleitet. Im Fehlerfall wird einfach der nächste aus der Liste gewählt und als Primary markiert.

Tabelle 9 Übersicht WSO2 ESB Load Balancing und Failover

Bewertung:
Neutrale Bewertung (o): Unterstützt
Das Produkt bietet zwar selbst kein Load Balancing und Failover an, jedoch kann der ESB mit einem Failover oder mit Load Balancing so umgehen, dass die Messages auf dem Bus korrekt übermittelt werden. Es sollte aber auch beachtet werden, dass die Load Balance und Failover Endpoints ein zusätzliches Plus sind und immerhin partiell die gewünschten Funktionen produktintern auf dedizierte Verbindungen ermöglichen.

3.4.3.3.4 STK4 – Verfügbare Dokumentation Community

a) **Installationsanleitung**
   Eine Installationsanleitung existiert für Linux, Solaris und Windows. Sie ist gut verständlich und, weil die Installation sehr einfach ist, sehr simpel gehalten.
b) **Verständlichkeit der Dokumentation**
   Es existiert eine qualitativ hochwertige Dokumentation im Internet. Sie entspricht dem Standard der heutigen Zeit (Atlassian Confluence) und ist gut verständlich. Eine Offline-Dokumentation kann mit der Export-Funktion erstellt werden. Die Funktionen werden mit Beispielen (inkl. Konfigurationsmöglichkeiten) beschrieben.

c) **Community**
Im herstellereigenen Blog[12] werden aktuelle Themen behandelt. Zusätzlich existieren bereits 1553 Fragen mit dem Tag „wso2esb“ und 3571 Fragen mit dem Tag „wso2“ auf Stack Overflow[13] (Stand 05.12.2014).

Bewertung:
Positive Bewertung (+): Umfangreiche Dokumentation und Community

---

3.4.3.3.5 STK5 – Support

Besteht die Möglichkeit von „Premium Support“ durch den Produkteanbieter? Wie gut ist dieser Support? Entstehen dadurch zusätzliche Kosten?

1) **Support-Angebot**
Der Hersteller bietet vier Support-Modelle an.
   a. **Community Support**
   Als Community Support wird der kostenfreie Support bezeichnet. Diese Fälle werden von der Community gelöst. Die Kommunikation findet in der Regel auf Stack Overflow statt.
   b. **Evaluation Support**
   Evaluation Support ist für den Endkunden auch kostenfrei. Der Unterschied zum Community Support ist, dass WSO2 den Support direkt übernimmt und probiert, so neue Kunden zu gewinnen.
   c. **Development**
   Hilfestellungen und Beratung bei der Entwicklung und Integration, beim Performanz-Tuning und bei Proof-of-Concept-Implementationen. Zusätzlich wird unlimitierter WSO2 Carbon Studio Support geleistet.
   d. **Production Support**
   Production Support kümmert sich um Feature Upgrades, Product Patches und Service Packs. Der Hersteller bietet bei Schwierigkeiten Kundensupport an, insbesondere bei Evaluation, Development und Production Support [18].
2) **Qualität des Support-Services**
Die Mitarbeiter im Support hinterliessen einen professionellen Eindruck. Grundsätzliches Know-How und Verständnis für den Kunden war bei der Support-Hotline vorhanden.
3) **Kosten für den Support**
(Angaben gemäss Preisliste 1. Oktober 2014, im Anhang)

Bewertung:
Neutrale Bewertung (o): Standard-Support
Der Support ist nicht inbegriffen. Das Unternehmen generiert Einnahmen durch Integrations- und Supportdienstleistungen. Die Qualität des Supports ist dementsprechend hochwertig.

3.4.3.4 Bewertung der "Look & Feel"-Kriterien

3.4.3.4.1 LFK1 – Manageability

1) **Command Line Interface**
Ein Command Line Interface existiert nicht.
2) **Web-Konsole**
Eine Web-Konsole existiert. Sie ist intuitiv und simpel aufgebaut. Konfigurationen lassen sich mit Hilfe von Wizards einfach erstellen. Sie können ebenfalls gelöscht und mutiert werden.

---

[12] Der Blog befindet sich unter http://wso2.com/blogs/ (zuletzt aufgerufen am 24.10.2014).

[13] Stack Overflow http://www.stackoverflow.com ist eine klassische Frage&Antwort-Seite für Programmierer (zuletzt aufgerufen am 18.12.2014).

3) **Logging**
   Das Produkt bietet eine grosszügige Anzahl von Log-Dateien an:
   a. **HTTP Access Log**
      Jegliche HTTP-Zugriffe werden im HTTP Access Log zusammengefasst.
   b. **WSO2 Carbon Log**
      Logs über den WSO2 ESB Server
   c. **WSO2 ESB Error Log**
      Error und Warning Logs aus WSO2 Carbon Log werden hier noch einmal zusammengefasst.
   d. **WSO2 ESB Service Log**
      Alle Logs, welche in Sequenzen via Log Mediator geloggt worden sind, werden hier zusammengefasst.

Bewertung:
Neutrale Bewertung (o): Kein Command Line Interface vorhanden
Der Support ist nicht inbegriffen. Das Unternehmen generiert Einnahmen durch Integrations- und Supportdienstleistungen. Die Qualität des Supports ist dementsprechend hochwertig.

3.4.3.4.2 LFK2 – Entwicklungsumgebung

Für WSO2 ESB existiert ein Eclipse Plug-In. Somit können Artefakte in der IDE vorbereitet werden. Das Plug-In unterstützt das Konfigurieren mit Grafiken.

Der Benutzer kann zum Beispiel Endpoints gleich konfigurieren wie in der Web-Konsole:

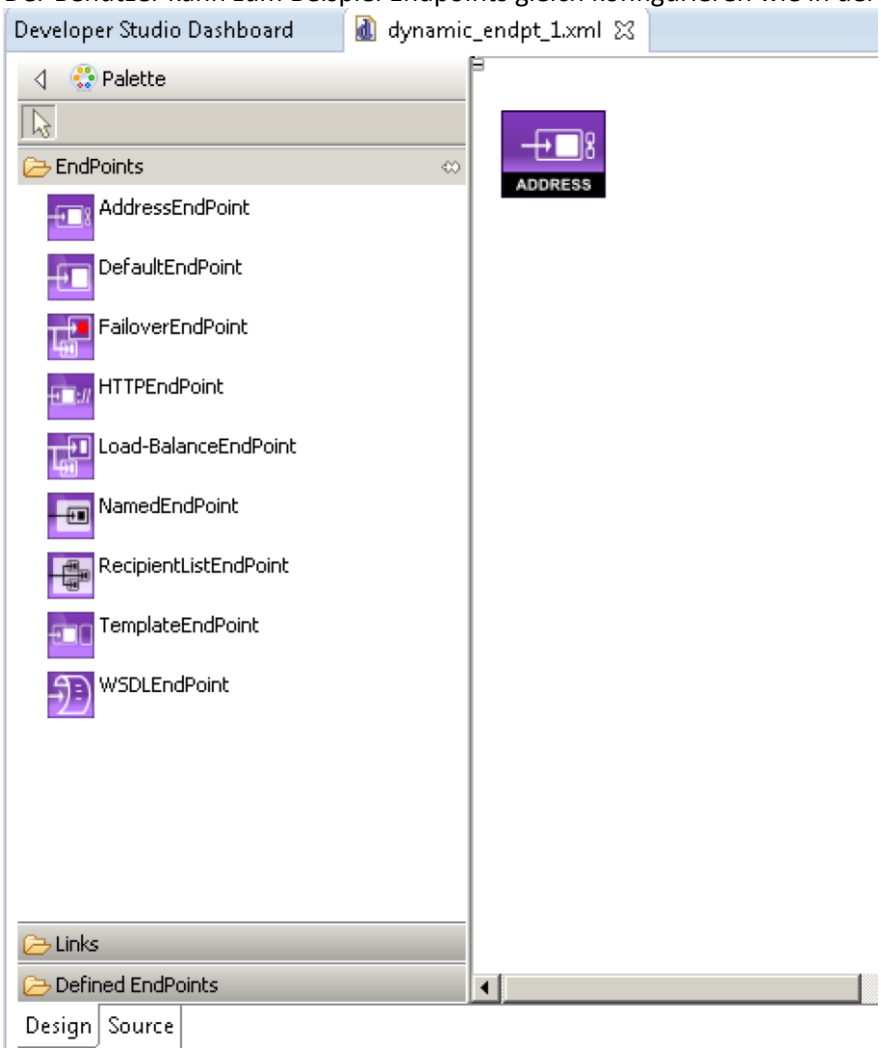


Abbildung 23 WSO2 Eclipse Plugin mit EndPoint View

Abbildung 24 WSO2 Eclipse Plugin mit Property View

Bewertung:
Positive Bewertung (+): IDE und gute grafische Unterstützung

3.4.3.4.3 LFK3 – Aufwand Installation / Konfiguration

a) **Aufwand Installation**
   a. **Download**
      Die Webseite [19] ist gut strukturiert, sodass man das Produkt schnell herunterladen kann.
   b. **Installationsvorgang**
      Die Installation [20] des Produkts gestaltet sich sehr einfach und ist Schritt für Schritt dokumentiert
      i. Herunterladen des Produkts
      ii. Archiv entpacken
      iii. Java-Umgebung (JAVA_HOME) setzen
      iv. Starten des WSO2 Servers

b) **Aufwand Konfiguration**
   Um das Produkt zu starten und zu betreiben, sind keine weiteren Konfigurationen nötig. Es kann sozusagen out-of-the-box verwendet werden.

Bewertung:
Positive Bewertung (+): Verständliches und simples Setup

3.4.3.5 Bewertung der allgemeinen Kriterien

3.4.3.5.1 AK1 – Configuration over Coding

Mit Hilfe der Web-Konsole und der diversen Konfigurationsdateien können so gut wie alle Funktionen des ESB ohne Programmierkenntnisse konfiguriert werden. Es besteht in der Web-Konsole auch jederzeit die Möglichkeit, in die Source View zu wechseln. Dort kann die Konfiguration zusätzlich in XML bearbeitet werden. Alle Parameter sind ausserdem sehr detailliert beschrieben.

Bei Standardprozessen wie Translation und Transformation (z.B. Filtern von Content) wird jedoch vorausgesetzt, dass Kenntnisse in XSLT 1.0/2.0, XPath, XQuery oder Smooks vorhanden sind. Hier wäre ein grafischer Editor wünschenswert.

Bewertung:
Neutrale Bewertung (o): Teils konfigurierbar
Es lassen sich Teile des ESB konfigurieren, jedoch muss trotzdem programmiert werden. Für die Konfiguration muss aber keine herstellerspezifische DSL erlernt werden. Die Konfiguration folgt via Web-Konsole.

3.4.3.5.2 AK2 – HW-/SW-Anforderungen

Hardware-Anforderungen:	
Memory	Keine Mindestanforderungen
Harddisk	Mindestanforderungen von ca. 125 Megabyte für die Installation
Software-Anforderungen:	
Betriebssystem	Linux Solaris MS Windows – XP / Vista
Java	JDK 1.6.x Nicht kompatibel mit JDK 1.8.x

Tabelle 10 WSO2 ESB HW-/SW-Anforderungen gemäss [21]

Bewertung:
Neutrale Bewertung (o): Erweiterte HW-Anforderungen oder externe Software wird vorausgesetzt
Externe Software JDK 1.6.x wird vorausgesetzt.

3.4.3.5.3 AK3 – Modularer Aufbau der Applikation [NFA1]

Bei der Installation konnten wir keine Module oder Funktionen explizit auswählen, jedoch sind sehr viele Funktionen standardmässig ausgeschaltet und lassen sich einzeln aktivieren. Ein gutes Beispiel hierfür sind die verschiedenen Message Builders und Message Formatters, welche über die entsprechende Konfigurationsdatei zuschaltbar sind.

Bewertung:
Positive Bewertung (+): Komplett modularer Aufbau mit Teillizenzierung
Da der ESB selber schon auf die Grundfunktionalität beschränkt ist und viele zusätzliche Funktionen wie Clustering oder Live Monitoring (mittels SNMP oder JMX) durch externe Produkte unterstützt werden, genügt die Möglichkeit zum Aktivieren/Deaktivieren von den einzelnen Modulen, um eine positive Bewertung zu erhalten. Zudem ist eine Teilinstallation aus Sicht von Kosten und Ressourcen bei diesem Open-Source-Produkt – welches bereits mit 125MB Festplattenspeicher für die Installation klarkommt – hinfällig.

3.4.3.5.4 AK4 – Kosten

Der WSO2 Enterprise Service Bus ist Open Source und mit Apache License Version 2.0 lizenziert. Daher entstehen keine Mehrkosten beim Einsatz des Produkts. Es werden auch keine Zusatzprodukte benötigt.

Bewertung:
Positive Bewertung (+): Keine Kosten

3.4.4 Implementation

3.4.4.1 Use Case-Setup

Da WSO2 die Enterprise Integration Patterns unterstützt, haben wir den Use Case *Vertragsdaten in Zeiterfassungstool importieren* auf Grundlage des Content Filter Patterns der „Enterprise Integration

Patterns with WSO2 ESB“[14] Guide umgesetzt. Für den Translator konnten wir eine einfachere Option verwenden, da wir die Daten bereits im Content Filter bearbeiten und unser Translator ausschliesslich für die Umwandlung zuständig ist.

3.4.4.1.1 Proxy-Service-Konfiguration

Die komplette Definition eines Message Flow wird bei WSO2 in einem Proxy zusammengefasst. Demnach erstellen wir für unsere Umantis-Verbindung einen solchen mit den folgenden Eigenschaften:

WSO2 Objekt/Einstellung	Bezeichnung/Attribut	Eigenschaft
Proxy Service → Custom Proxy	UmantisVertragsdaten	Objekt, welches als Service zur Verfügung steht und den kompletten Nachrichtenfluss innerhalb des ESB regelt.
Transport Settings	HTTP, HTTPS	Der Service nimmt Verbindungen über diese beiden Schnittstellen entgegen.
In Sequence	-	Die Client-Anfrage muss nicht bearbeitet werden, da eingehend ein simpler HTTP/GET-Befehl ausgeführt wird.
Endpoint → Address Endpoint	URL Umantis Server	Definiert das Backend des Services, hier kann die direkte URL von Umantis verwendet werden ohne Modifikationen, z.B. *https://192.168.70.129.*
Out Sequence	XSLT Mediator Property Mediator Log Mediator Send Mediator	Der XSLT Mediator ist zuständig für das Content Filtering der Nachricht, während der Property Mediator die Umwandlung von XML in JSON übernimmt. Der Log Mediator wird danach dazwischen geschaltet. Dies hilft, bei Problemen einen Anhaltspunkt zu erhalten, in welchem Zustand sich die Daten zu dem Zeitpunkt befinden. Der Sendmediator ist zum Schluss notwendig, damit die Daten an den Aufrufer zurückübermittelt werden.
Fault Sequence	-	Für unseren Use Case nicht notwendig.

Tabelle 11 WSO2 Use Case Setup

Die Konfiguration kann im grafischen Design Editor oder als XML bearbeitet und erstellt werden. Zur Zusammenfassung sei hier nur der XML-Auszug aufgezeigt:

[14] Die komplette Guide ist auf der Herstellerhomepage aufrufbar und kann heruntergeladen werden [15] .

```
<?xml version="1.0" encoding="UTF-8"?>
<proxy xmlns="http://ws.apache.org/ns/synapse"
       name="UmantisVertragsdaten"
       transports="https,http"
       statistics="disable"
       trace="disable"
       startOnLoad="true">
   <target>
      <outSequence>
         <xslt key="split"/>
         <property name="messageType" value="application/json"
scope="axis2"/>
         <log level="full"/>
         <send/>
      </outSequence>
      <endpoint>
         <address uri="https://192.168.70.129"/>
      </endpoint>
   </target>
   <description/>
</proxy>
```

Code 4 WSO2 Proxy-Konfiguration

Im Property Tag kann über den MessageType das gewünschte Content-Type-Format angegeben werden. Für den Use Case ist dies „application/json“.

3.4.4.1.2 XSLT Transformation

Der Verweis „key=split“ im XSLT Tag bezieht sich auf einen Eintrag in der lokalen Registry des ESB, welche den Code für die XSLT Transformation beinhaltet.

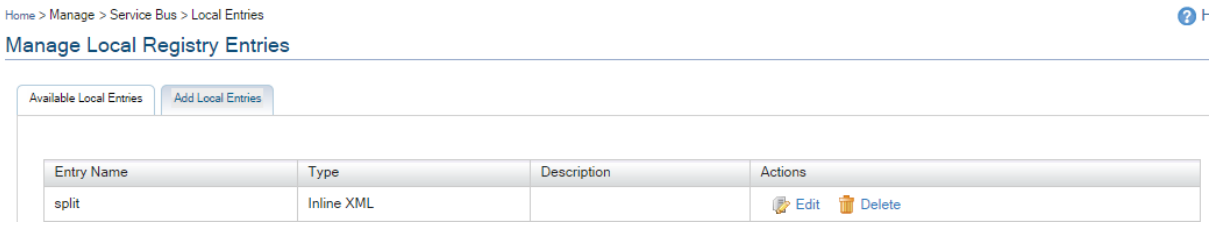


Abbildung 25 WSO2 Local Registry Entries

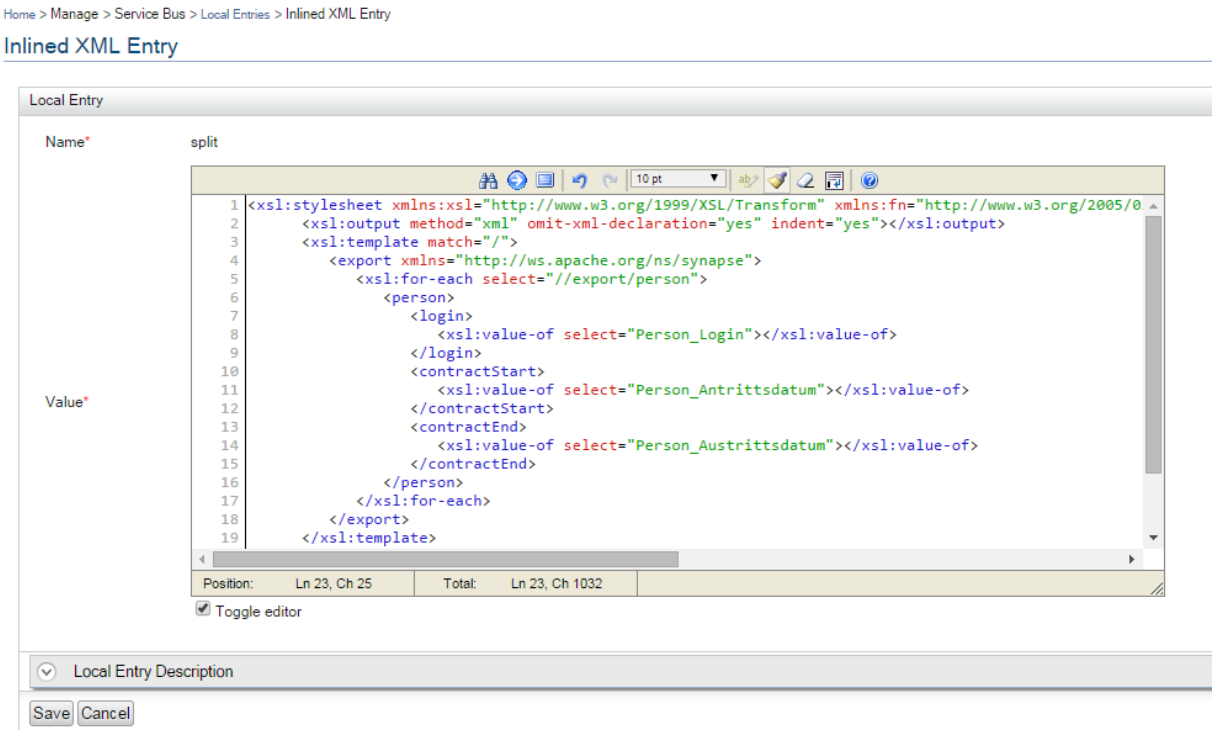


Abbildung 26 WSO2 Local Registry Entry Editing

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fn="http://www.w3.org/2005/02/xpath-functions"
xmlns:m0="http://services.samples" version="2.0" exclude-result-prefixes="m0 fn">
      <xsl:output method="xml" omit-xml-declaration="yes"
indent="yes"></xsl:output>
      <xsl:template match="/">
         <export xmlns="http://ws.apache.org/ns/synapse">
            <xsl:for-each select="//export/person">
               <person>
                  <login>
                     <xsl:value-of select="Person_Login"></xsl:value-of>
                  </login>
                  <contractStart>
                     <xsl:value-of select="Person_Antrittsdatum"></xsl:value-of>
                  </contractStart>
                  <contractEnd>
                     <xsl:value-of select="Person_Austrittsdatum"></xsl:value-of>
                  </contractEnd>
               </person>
            </xsl:for-each>
         </export>
      </xsl:template>
   </xsl:stylesheet>
```

Code 5 WSO2 XSLT Transformation File

3.5 Oracle ESB

Oracle bietet für Hochschulen ein Programm an, welches den Studenten die Möglichkeit bietet, kostenlos das Produkt Oracle ESB zu verwenden. Die Beurteilung basiert auf Release 12c (12.1.3.0.0), heruntergeladen und installiert am 02.11.2014. Als Datenbank wird MySQL 5.6.21 verwendet.

3.5.1 Produktbeschreibung

> *"Oracle Service Bus is a configuration-based, policy-driven enterprise service bus designed for SOA life cycle management. It provides foundation capabilities for service discovery and intermediation, rapid service provisioning and deployment, and governance. Service Bus provides scalable and reliable service-oriented integration, service management, and traditional message brokering across heterogeneous environments. It combines intelligent message brokering with routing and transformation of messages, along with service monitoring and administration. Service Bus leverages industry standards to connect services and support a high level of heterogeneity, connecting your existing middleware, applications, and data sources, and protecting existing investments."*
>
> *[22, p. 63]*

Zitat 3 Produktebeschreibung des Herstellers zum Oracle Service Bus

3.5.2 Architektur

Die Architektur von Oracle Service Bus ist in der Abbildung unten illustriert.

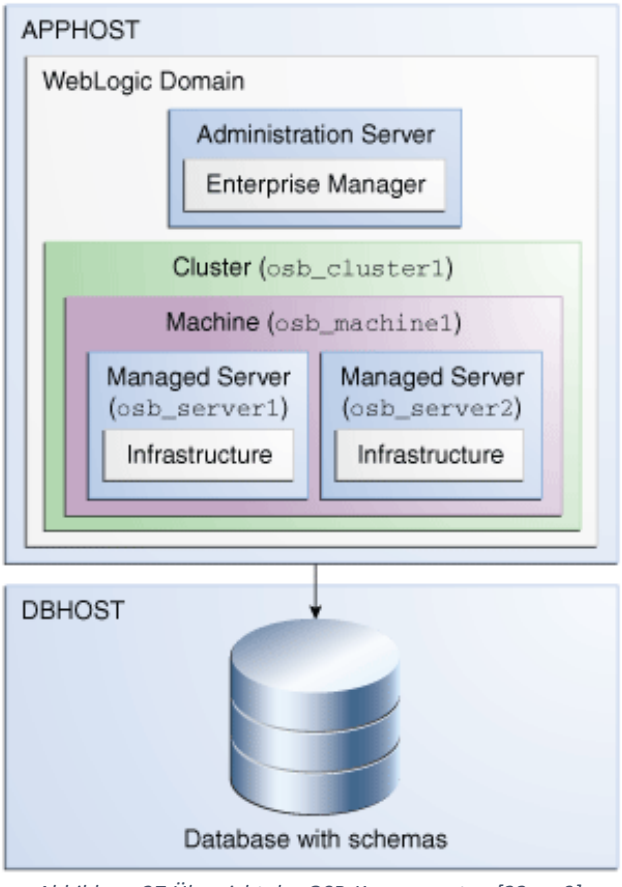


Abbildung 27 Übersicht der OSB-Komponenten [23, p. 9]

Der Oracle Service Bus besteht aus den folgenden Komponenten:

Komponente	Beschreibung
Administration Server	Web-Applikation für die Administration des WebLogic-Application-Servers, in welcher unter anderem der ESB läuft. Hier wird auch alles für das Management des Oracle Service Bus eingestellt.
Managed Server	Der Oracle Service Bus läuft in einem Managed Server. Dieser Managed Server ist ein Teil einer WebLogic Domain.
Cluster	Ein (oder mehrere) Cluster kann (können) in der WebLogic Domain laufen.

Tabelle 12 Komponenten des Oracle Service Bus

3.5.2.1 Service-Bus-Komponenten

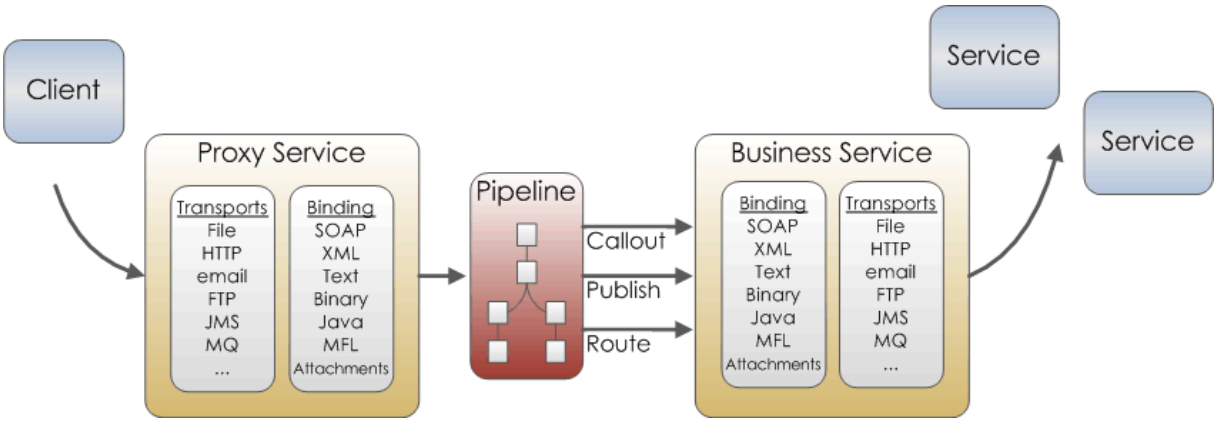


Abbildung 28 Nachrichtenfluss durch den Oracle Service Bus [24, p. 67]

3.5.2.1.1 Service-Bus-Komponenten

3.5.2.1.1.1 Proxy Services

Proxy Services sind die externen Schnittstellen zum ESB. Ein Proxy Service wird für die interne Weiterleitung der Messages zu Business Services gebraucht.

3.5.2.1.1.2 Business Services

Business Services sind die Schnittstellen zu den angebotenen Webservices. Der ESB interagiert in diesem Falle als Client. Business Services sind auch für Transformationen und Translationen von Nachrichten.

3.5.2.1.1.3 Message Flows

Message Flows definieren, wie Nachrichten geroutet, validiert und transformiert werden, bevor sie zum nächsten Business Service weitergeleitet werden. Message Flows können sequenziell mit einer Pipeline oder parallel mit Split-Join ablaufen.

3.5.2.1.1.4 Transports

Transports sind die verfügbaren Protokolle, welche zwischen den Services gebraucht werden können.

3.5.2.1.1.5 JCA Adapters

J2EE ConnectorArchitecture (JCA) Adapter ermöglichen die Interaktion mit Proxy und Business Services wie Datenbanken, Dateisystemen, FTP-Servern, Messaging-Systemen, E-Mails, Lightweight Directory Access Protocol (LDAP), Cloud Services und der Oracle E-Business Suite.

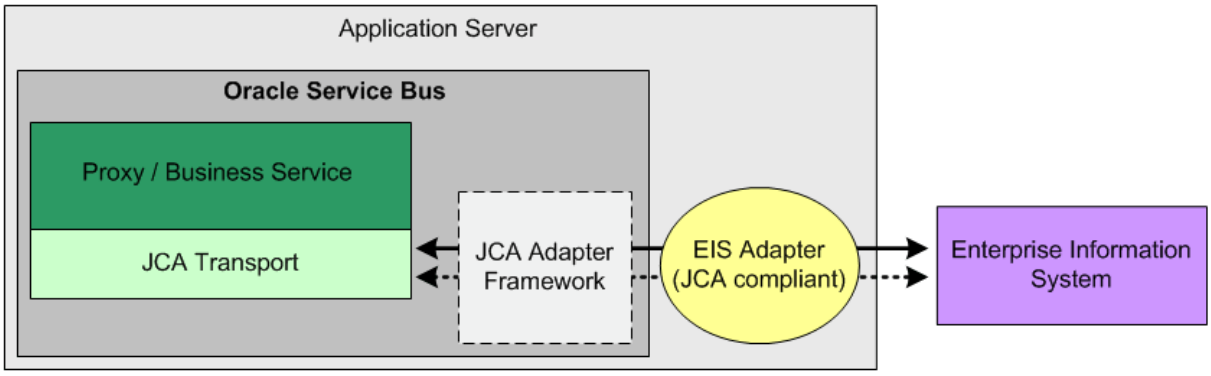


Abbildung 29 Interaktion zwischen Oracle Service Bus Services und einem Enterprise Information System via JCA Adapter [24, p. 537]

3.5.2.1.1.6 Bindings

Bindings sind die vordefinierten Typen von Nachrichten. Für jedes Binding sind Funktionen definiert wie Ver- und Entpacken von Nachrichten und Security Handling.

3.5.3 Produktbewertung

Installiert auf einer virtuellen Maschine Windows 7 SP1 x64 mit allen aktuellen Windows Patches und dem Java SE Development Kit 7u67.

3.5.3.1 Bewertung der funktionalen Kriterien

3.5.3.1.1 FK1 – Unterstützte Protokolle/Technologien und Adapter

Transport-Protokoll	Beschreibung und unterstützte Protokolle
E-Mail	• POP • SMTP • IMAP
File	Um Files von einem lokalen File-System zu lesen und zu schreiben, wird ein File-Adapter benutzt.
FTP / SFTP	Der OSB unterstützt FTP und SFTP als Transporttechnologie sowohl auf der Proxy- wie auch auf der Business-Service-Seite.
HTTP / HTTPS	Diese beiden Transports unterstützen die Übertragung von Messages. **SOAP** (Version 1.1 und 1.2) Messages sowie Messages via **RESTful** HTTP/HTTPS können übertragen werden.
JMS	(Including MQ using JMS, and JMS/XA) WebSphere MQ
Adapters zu Third-Party Services	Der OSB stellt einen komplett vorkonfigurierten SAP- und Salesforce-Adapter zur Verfügung. Weitere Applikationen können über den sogenannten Third-Party Adapter angeschlossen werden. Er erlaubt, WSDL- und JCA-Files von externen Softwareherstellern wie PeopleSoft zu importieren und so auf einfache Weise einen Customized JCA Adapter zu erstellen.

Weitere unterstützte Protokolle/Technologien	• Database Connector • DSP (Oracle Data Service Integrator) • EJB / RMI • JCA • JEJB • Local (proprietär für Inter-ESB-Kommunikation) • SB (RMI Support) • SOA-DIRECT (Oracle SOA Suite) und BPEL • Tuxedo (Oracle Tuxedo) • WS (Web Services)

Tabelle 13 OSB Protokollunterstützung

Bewertung:
Positive Bewertung (+): Standardprotokolle plus zusätzliche Adapter/Konnektoren zu Third-Party Services

3.5.3.1.2 FK2 – Custom Adapter Framework

Es ist möglich, Custom Adapters für den Oracle Service Bus zu entwickeln. Das Oracle J2EE Connector Architecture Adapter Framework [24, p. 537] unterstützt die Entwicklung von eigenen Adaptern.

In einer solchen Architektur interagiert der JCA Transport mit dem Backend Enterprise Information System (EIS) und erlaubt diesem die Kommunikation mit dem OSB. Dazu erfolgt die Verbindung über das integrierte JCA Adapter Framework und einen EIS Adapter. Dies wird im nachfolgenden Bild illustriert, in welchem die gestrichelten Linien eine Response und die durchgezogenen Linien einen Request darstellen. Der EIS Adapter fungiert als zentrale Drehscheibe.

Zusätzlich bietet Oracle die „Transport SDK“[15] an, um eine Abstraktionsschicht zwischen dem Service Bus und dessen Komponenten zu implementieren. Die Abstraktion ermöglicht die einfache Entwicklung von neuen Transports, ohne sich um den Nachrichtenfluss kümmern zu müssen.

Bewertung:
Positive Bewertung (+): Gute Unterstützung

3.5.3.1.3 FK3 – Security [FA3, FA4]

Der OSB unterstützt die Security [24, p. 903] in verschiedenen Levels, die auf den Oracle Platform Security Services und/oder dem Oracle Web Services Manager (OWSM) basieren:

- Transport-level Security: SSL, Basic Authorization und Custom Security Credentials
- Message-level Security: WS-Security, SAML, Benutzername und Passwort, X.509, Signierung und Verschlüsselung sowie Custom Security Credentials
- Console security: Single-Sign-on und Role-based-Zugriff
- Policy Security

Um eine Bewertung zu erstellen, schauen wir die Security-Aspekte in den nachfolgenden Abschnitten etwas detaillierter an:

a) Authentication
Der OSB unterstützt eine Authentifizierung auf dem Transport und auf dem Message Level [24, p. 915]. Auf dem Transport Level wird dies durch einen übermittelten Custom Token im HTTP(S) Header oder durch SSL realisiert, beim Message Level kann der Custom Token oder

[15] Weitere Informationen über die „Transport SDK“ sind unter http://docs.oracle.com/middleware/1213/osb/develop/osb-transport-design.htm#OSBDV1237 verfügbar (zuletzt aufgerufen am 01.12.2014)

Benutzername/Passwort im SOAP Header oder bei Non-SOAP Messages im XML-Payload übermittelt werden. In diesem Level wird auch die Security mit WS-Policies unterstützt.

Als Custom Token werden unter anderem X.509-Zertifikate und SAML Tokens unterstützt, aber auch beliebige andere, welche an eine entsprechende Verifizierungsstelle weitergeleitet werden können, wie im folgenden Bild illustriert.

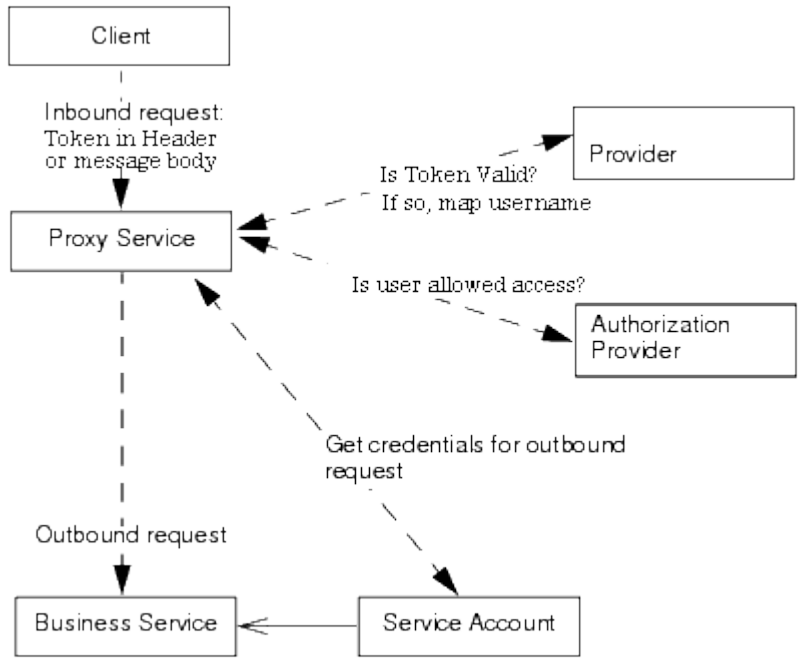


Abbildung 30 OSB Authentication [24, p. 915]

b) Authorization
Für die Authorization werden der WebLogic User Store und Key Store verwendet. Alternativ kann auch ein externer Store angeschlossen werden.

c) Confidentiality
Die Vertraulichkeit respektive Verschlüsselung kann ebenfalls mit X.509-Zertifikaten sichergestellt werden. Dazu kann auf einen beliebigen Service Key Provider innerhalb des Netzwerks zugegriffen werden.

d) Data Integrity
Auch die Datenintegrität kann mit X.509-Zertifikaten und einer Public Key Infrastructure sichergestellt werden.

Bewertung:
Positive Bewertung (+): Hoher Security-Standard
Das Produkt bietet in allen vier Bereichen eine Security-Lösung an, welche die heutigen Standards übertrifft. Dies wird vor allem mit der Zertifikatsunterstützung sichergestellt. Es gilt jedoch zu beachten, dass die Security grösstenteils mit externen Lösungen realisiert wird (z.B. mit einer PKI- oder SAML-Authentifizierungsstelle). Mit der Möglichkeit zur Message-Level und Transport-Level Security kann man aber durchaus sagen, dass die heutigen Standards sehr gut implementiert und übertroffen werden können. Wie einfach die Security-Aspekte zu realisieren sind, wird in dieser Bewertung jedoch nicht berücksichtigt, mangels Erfahrung mit dem Produkt.

3.5.3.1.4 FK4 – Message Translation

Der OSB unterstützt die Konvertierung von verschiedenen Formaten. Dies betrifft sowohl die Translation von Text (z.B. CSV) und Binary in das XML-Format als auch umgekehrt. Dazu wird ein

Message Format Language (MFL)[16] File benutzt. Der JDeveloper beinhaltet einen Format Builder zur grafischen Unterstützung, um ein solches File zu erstellen.

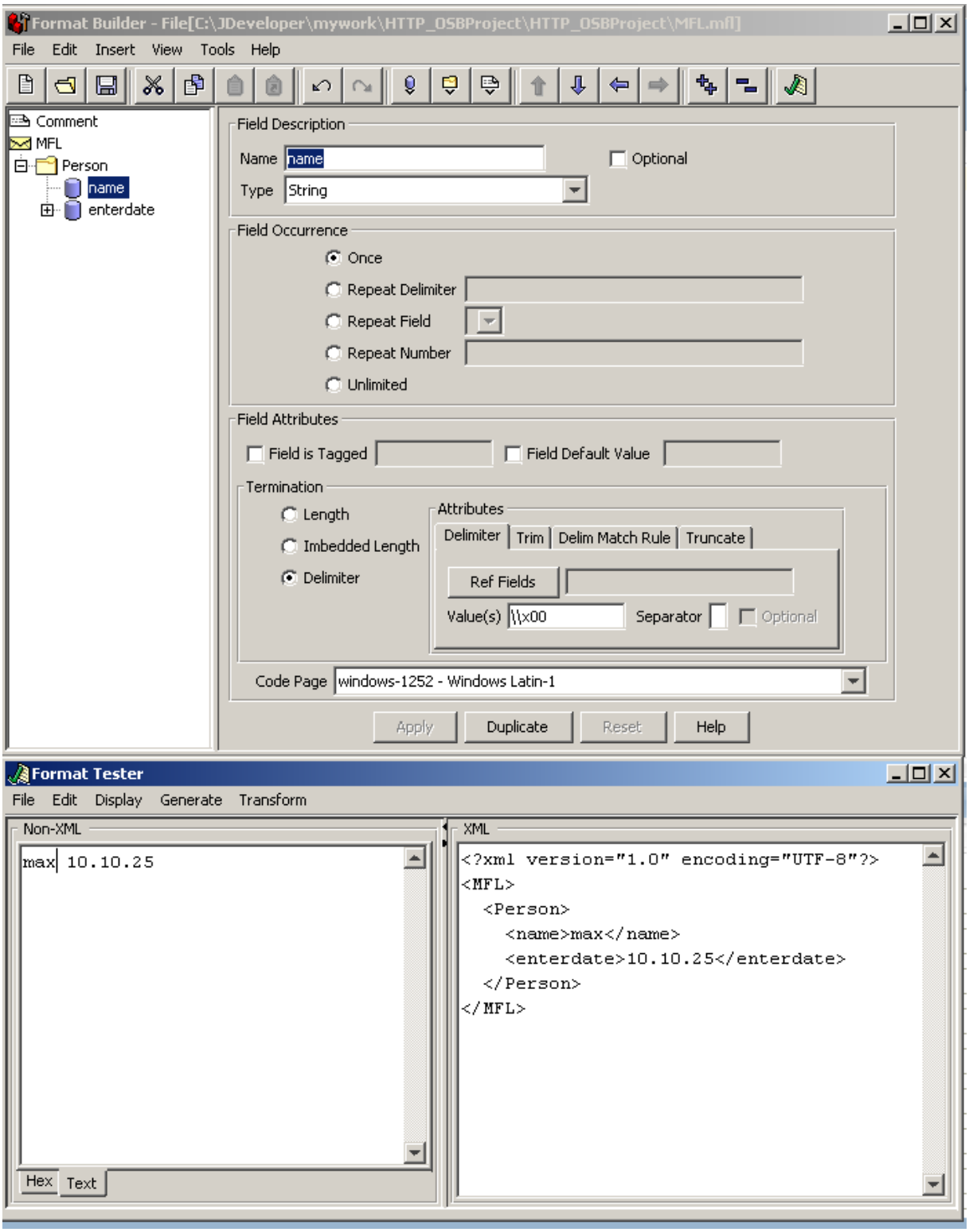


Abbildung 31 OSB - MFL Format Builder and Tester

Die Konvertierung von und zu JSON wird leider nicht unterstützt. Es besteht jedoch die Möglichkeit, selber einen Konverter zu schreiben und diesen dann während der Verarbeitung in der Pipeline

[16] MFL ist eine proprietäre Format- und Transformationssprache der Firma BEA System welche 2008 von Oracle übernommen wurde.

mittels Java Call aufzurufen oder einen Native Format Builder zu erstellen, jedoch sind beide Varianten sehr umständlich.

Bewertung:
Neutrale Bewertung (o): Mittlere Unterstützung der Formate
JSON wird nicht unterstützt, obwohl dieses Format in den letzten Jahren an Popularität gewonnen hat und gerade die Verarbeitung in Java durch seine kompakte Form erleichtert.

3.5.3.1.5 FK5 – Datenstrukturtiefe

Dieses Bewertungskriterium haben wir mit dem gleichen Setup wie bei der Bewertung des Produkts WSO2 ESB überprüft.

Der Oracle Service Bus konnte in unserem Test die Anforderungen erfüllen.

Bewertung:
Positive Bewertung (+): Mehrfach verschachtelte Strukturen

3.5.3.1.6 FK6 – Message-Durchsatz / Concurrent Streams

Die Messages durchlaufen im OSB diverse Stufen (Proxy Service, Pipeline, Business Service). Daher können sich auch jederzeit mehrere Messages gleichzeitig im ESB befinden. Der Message- Durchsatz kann daher nicht eindeutig beurteilt werden und ist auch abhängig von den darunterliegenden Hardwarekomponenten. Allerdings gibt es die folgenden konfigurierbaren Parameter, die nahelegen, dass Concurrent Streams unterstützt werden:

- Die Concurrent Requests in einem Business Server (Outbound Requests) können limitiert werden.
- Die Anzahl Threads, welche einen Request für einen Proxy Service verarbeiten, können limitiert werden.

Positive Bewertung (+): Erweiterte synchrone Verarbeitung
Es können mehr als drei Messages gleichzeitig verarbeitet werden.

3.5.3.1.7 FK7 – Betreibbarkeit in Cloud

Der Oracle Service Bus erfüllt fast alle IDEAL-Eigenschaften [9]:

a) Isolated State ✓
Die Architektur des Oracle Service Bus ist stateless und lightweight [24, p. 63].

b) Distributed ✓
Das Produkt besteht aus einem Administration Server, mehreren Managed Servers und der Datenbank. Diese lassen sich verteilt betreiben.

c) **Elastic – Scale Out / Scale Up** ✓
Scale Out ist durch Clustering [22, p. 327] erreichbar. Scale Up ist auch möglich. Hierbei kann der Benutzer unter anderem die Anzahl Threads konfigurieren [25, p. 200].

d) Automated Management ✗
Die Konfiguration manifestiert sich in Dateien. Es existiert kein Command Line Interface (CLI) zur Administration der Konfiguration. Somit ist Automated Management schwer zu erreichen und dementsprechend nicht erfüllt.

e) Loosely Coupled ✓
Komponenten sind lose gekoppelt im Oracle Service Bus. Es laufen beispielsweise eigene Services mit eigenen Endpunkten, die einander nicht tangieren oder blockieren. Ausserdem können einzelne Artefakte jederzeit hinzugefügt, entfernt oder gewartet werden, ohne dass der ganze ESB dadurch funktionsunfähig wird.

Bewertung:
Neutrale Bewertung (o): Stateless und Cloud-Erfahrung vorhanden
Das Produkt lässt sich problemlos in der Cloud betreiben. Trotzdem konnten nicht alle IDEAL Kriterien (A) eindeutig erfüllt werden, weshalb die Bewertung nur durchschnittlich ausfällt.

3.5.3.1.8 FK8 – Discovery and Registry

Keine Bewertung für dieses Kriterium.

3.5.3.1.9 FK9 – Workflow Engine & Service Composition

Keine Bewertung für dieses Kriterium.

3.5.3.1.10 FK10 – Versionierung der Komponenten

Alle Änderungen müssen in Sessions erstellt und dann aktiviert (wahlweise mit einem Kommentar) werden. Der Oracle Service Bus hat eine Übersicht aller Sessions.

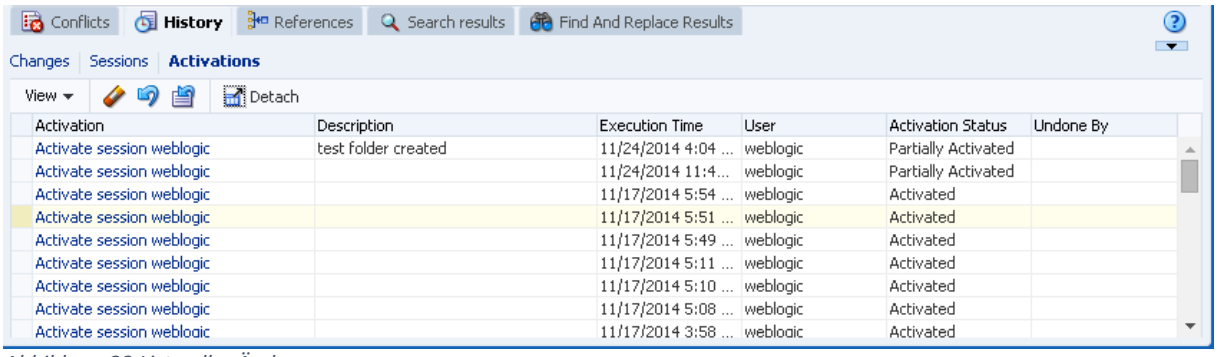


Abbildung 32 Liste aller Änderungen

Es ist dann möglich, einzelne Sessions rückgängig zu machen. Leider ist es nicht möglich, spezifische Komponenten mit Versionen zu versehen, sondern nur, eine ganze Session zurückzusetzen.

Was wir auch als hilfreich empfunden haben, war die Möglichkeit, alle Änderungen inkl. der Session zu verwerfen.

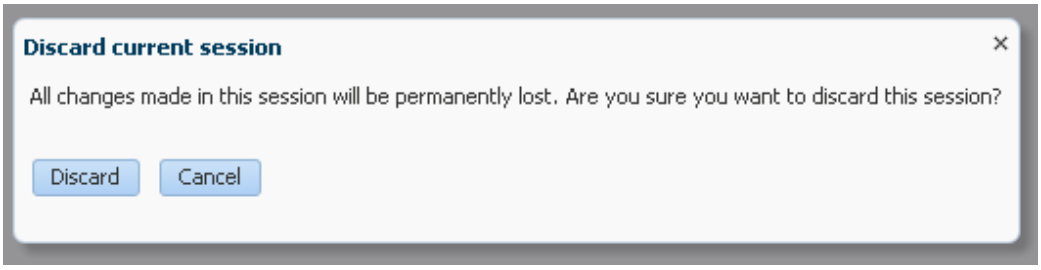


Abbildung 33 Dialog mit der Frage, ob wir tatsächlich die Änderungen in der Session verwerfen möchten

Um eine separate Version einer Komponente zu betreiben, muss diese kopiert werden. Sie ist dann logisch unabhängig.

Bewertung:
Positive Bewertung (+): Versionierung der Komponenten und Betrieb von multiplen Versionen unterstützt

3.5.3.2 Bewertung der Design-Kriterien

3.5.3.2.1 DK1 – Message Routing Patterns

Es gibt keine offizielle Liste mit den vom OSB unterstützten EI Patterns. Wir konnten trotzdem eine Anleitung in einem Blog [26] finden. Daher gehen wir nachfolgend auf Möglichkeiten ein, wie die

Patterns implementiert werden könnten. Dies sind jedoch Annahmen und keine Praxiserfahrungen und sie beziehen sich auf Konfigurationen innerhalb der Pipeline.

a) **Simple Router**

Content-Based Router ✓

Es kann ein „Conditional Branch“ zum Message Flow hinzugefügt werden. Danach können beliebig viele Branches mit eigenen Routing Conditions erstellt werden. Als Input für den If-Else Operator kann eine Variabel oder direkt eine XPath-Beschreibung aus der Message verwendet werden.

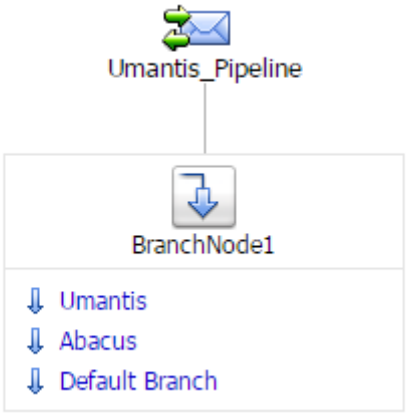


Abbildung 34 OSB Content-Based Routing

Alternativ ist es auch möglich, innerhalb einer Stage mit einer If-Else-Anweisung zu arbeiten. Der Vergleich wird dort ebenfalls mit Hilfe einer Variabel oder einer XPath-Beschreibung direkt aus der Message gemacht.

Message Filter ✓

Die Nachrichten können innerhalb des OSB auf zwei Levels gefiltert werden:

Entry Level (Proxy Service)	Nur Nachrichten, welche einem definierten XSD- oder MFL-Schema entsprechen, werden überhaupt vom Proxy Service angenommen.
Processing Level (Pipeline)	Mittels If-Else-Anweisung innerhalb des Message Flow werden nur die gewünschten Nachrichten weiterverarbeitet (sonst kann z.B. ein Error generiert werden).

Recipient List ✓

Mit einer For-Each-Anweisung kann aus einem externen File die Liste aufgerufen werden, welche alle Recipients beinhaltet. Innerhalb dieser Schlaufe ruft dann ein Service Callout eine Dynamic Route zu den jeweiligen Recipients auf.

Splitter ✓

Der OSB unterstützt ein Konstrukt namens Split Join, um Nachrichten aufzuteilen und zu vereinen. Dazu wird ein mitgegebenes WSDL File verwendet.

Resequencer ✗

Der OSB selber unterstützt dieses Pattern leider nicht direkt. Allerdings könnte indirekt mit einer Datenbank gearbeitet werden, wo die Messages jeweils durch einen Polling Proxy Service zwischengespeichert werden. Die Logik wäre dann in der Datenbank mittels StoredProcedure gesteuert und der OSB würde durch einen Java Callout auf die Nachrichten zugreifen, sobald alle Messages einer kompletten Gruppe eingetroffen sind.

Aggregator ✗
Die Implementierung ist nicht direkt unterstützt. Die Realisation könnte jedoch ähnlich wie beim Resequencer aussehen mit dem Unterschied, dass die Nachrichten mit einem Split-Join-Konstrukt in der Pipeline zusammengeführt werden.

b) **Composed Routers**
Da die Pipeline beliebig aufgebaut werden kann mit verschiedenen Routen und Branches, sind auch die verwendeten Routing Patterns beliebig kombinierbar.

c) **Dynamic Routing**
Dynamisches Routen wird direkt in der Pipeline unterstützt. Es gibt dazu extra ein Konstrukt namens Dynamic Routing. Als Expression für das Routing wird eine Variabel oder eine XPath-Beschreibung direkt aus der Message verwendet.

Abbildung 35 OSB Dynamic Routing

Bewertung:
Neutrale Bewertung (o): Teil der Pattern und beliebige Kombinationsmöglichkeit

3.5.3.2.2 DK2 – Message Content Transformation Pattern

Der Oracle Service Bus unterstützt Message Content Transformation mit XQuery (W3C spezifizierte Sprache) und XSLT. Es wird nicht zwischen Content Enricher und Content Filter unterschieden. Somit werden beide Patterns unterstützt.

Für XQuery existiert eine Benutzeroberfläche, um Transformationen visuell zu gestalten:

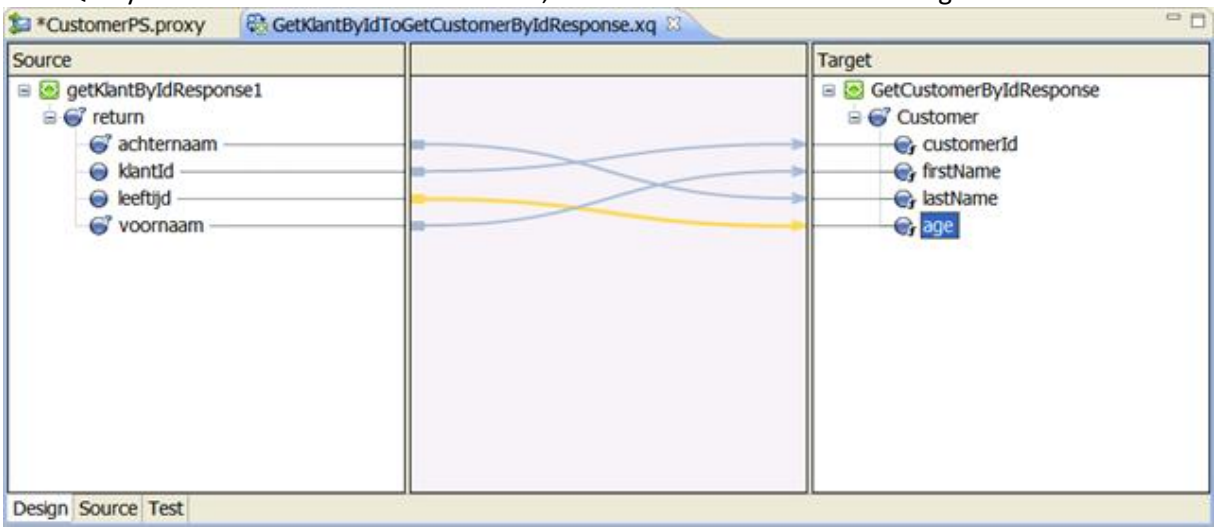


Abbildung 36 XQuery-Benutzeroberfläche mit Mappings in Oracle's JConsole [27]

Bewertung:
Positive Bewertung (+): Content Enricher und Content Filter werden unterstützt. Die Modellierungssprache ist standardisiert und somit nicht herstellerspezifisch.
XSLT und XQuery (inkl. grafischer Unterstützung) werden als Modellierungssprache angeboten.

3.5.3.2.3 DK3 – Message Consumer Patterns [FA2]

In der Auflistung unten haben wir die Message Consuming Patterns des OSB getestet. Zusätzlich haben wir herausgefunden, dass das Split-Join Pattern [24, p. 227] angewendet werden kann, um die Nachrichten in mehrere Sub-Nachrichten aufzuteilen und diese unabhängig abzuarbeiten. Dadurch kann die Performanz gesteigert werden.

- **Message Consumer Patterns**
  Welche Message Consumer Patterns der folgenden Kategorien können mit dem ESB abgebildet werden?
  - **Polling Consumer** ✓
    Mit Poller Transports [24, p. 587] können Nachrichten aus der JMS Queue (Oracle Advanced Queuing wird auch unterstützt) gelesen und verarbeitet werden. Die Nachrichten werden „at least once" prozessiert.
  - **Event Driven Consumer** ✓
    Dieses Pattern kann im Sinne von Publish-Subscribe implementiert werden. Die Subscriber sind Consumer Proxies.
  - **Message Dispatcher** ✓
    Der OSB erlaubt, bei der Konfiguration des Business Service mehrere Targets zu hinterlegen und zwischen verschiedenen Load-Balancing-Strategien zu wählen. Die verfügbaren Strategien sind Round Robin, Random und Random-Weighted.
  - **Competing Consumer** ✓
    Dieses Message Pattern ist analog dem Message Dispatcher via Load-Balancing-Strategie im Business Service realisierbar.
  - **Selective Consumer** ✓
    Die Nachrichten können problemlos anhand von vordefinierten Schemas (XML oder WSDL) selektiert werden. Der weitere Verlauf der ungültigen Nachrichten kann in einem Error-Handling-Ablauf in der Pipeline definiert werden. Oder die ungültigen Nachrichten werden ins Nirwana geschickt.
  - **Durable Subscriber** ✓
    Dieses Pattern ist realisierbar, allerdings müssen die Nachrichten dazu innerhalb des OSB via JMS Queue übermittelt werden. Die Logik wird dann durch die JMS Queue auf dem WebLogic Server realisiert.
  - **Idempotent Receiver** ✓
    Da der OSB stateless ist, kann er von sich aus keine duplizierten Nachrichten erkennen. Er nimmt jede Nachricht eigenständig entgegen und kennt keine Historie. Jedoch kann die Pipeline als JMS Queue verwaltet werden, wodurch sich die Zustellung einer Nachricht auf einmal begrenzen lässt. Alternativ kann dies umgangen werden, indem die Nachrichten – oder mindestens die MessageID und Timestamps – extern gesichert werden. Kommt eine neue Nachricht in den OSB, wird ein Check durchgeführt, ob diese Nachricht schon einmal entgegengenommen wurde.
  - **Service Activator** ✓
    Durch Filtern der Nachrichten nach einem bestimmten Schema (XML oder WSDL) können Anfragen von Clients überprüft und beispielsweise nur die „Gültigen" (=Anfragen eines bestimmten Services) verarbeitet werden. Ungültige Anfragen können demnach im OSB abgefangen und verworfen oder mit einer Error Message beantwortet werden, auch wenn sie vom Replier aus zulässig wären.

Bewertung:
Positive Bewertung (+): Alle 8 Patterns werden unterstützt

3.5.3.3 Bewertung der Support- und Troubleshoot-Kriterien

3.5.3.3.1 STK1 – Reporting and Statistic

Der Oracle Service Bus kann Informationen zu allen Typen von Events aufzeichnen sowie zusätzlich Startup- und Shutdown-Informationen, Fehler und Warnungen loggen. Die Logs werden gemäss dem Standard von Oracle Diagnostic Logging (ODL) erstellt.

Konfigurationsänderungen werden aufgezeichnet und versioniert. Message Flow wird standardmässig nicht geloggt. Die Begründung des Herstellers ist, dass eine Aufzeichnung der ganzen Message Flows zeitaufwändig wäre. Jedoch kann der Benutzer wenn nötig jeweils Report-Actions einfügen, um so loggen zu können.

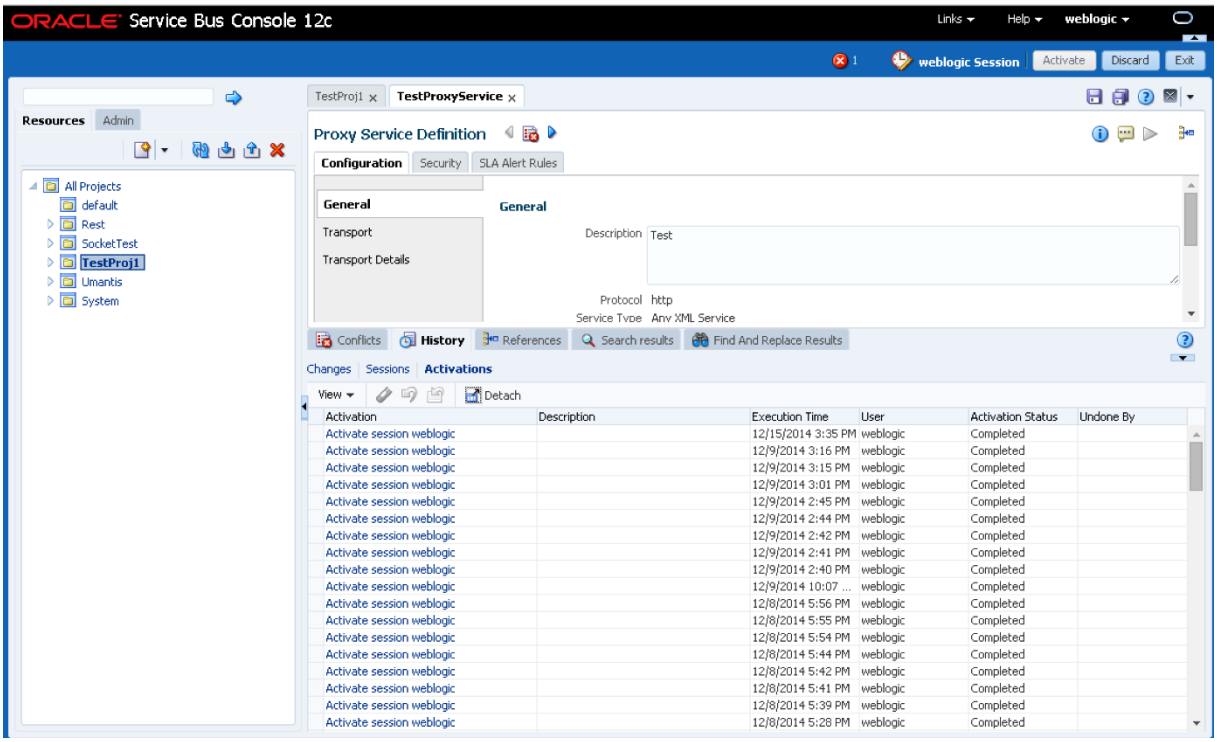


Abbildung 37 History mit Activations im Oracle Service Bus

Bewertung:
***Positive Bewertung (+):* Message Flow und Konfigurationsänderungen, nützliche Aufbereitung**

3.5.3.3.2 STK2 – Alerting and Monitoring

Der Oracle Service Bus bietet Service Monitoring an. Diese Funktion zeigt an, wie viele Nachrichten erfolgreich abgearbeitet wurden und wie viele fehlerhaft waren. Durchschnittszeiten für die Abarbeitung einer Nachricht werden auch berechnet.

Via JMX Monitoring APIs kann auf Statistiken zugegriffen werden. Zusätzlich werden SNMP (in den Versionen 1 und 2) unterstützt.

Für Alerting können sogenannte „Alert Destinations“ definiert werden. Wenn Alerts auftreten, werden diese an die definierten „Alert Destinations“ weitergeleitet. Alert Destinations sind in der Regel E-Mail-Adressen oder JMS Destinations. Es ist aber auch möglich, „Reporting Destinations“ zu

definieren. Diese werden gegen Oracle-spezifische Schnittstellen implementiert und können dann in Applikationen von Drittanbietern (oder eigenem Code) aufgerufen werden.

Bewertung:
***Positive Bewertung (+):* Alerting and Monitoring unterstützt**

3.5.3.3.3 STK3 – Failover / Load Balancing

In der hier getesteten Version 12 ist der OSB fester Bestandteil der Oracle SOA Suite, welche zur Oracle Fusion Middleware Plattform gehört. Dies hat zur Folge, dass das ganze Setup auf dem WebLogic Server basiert. Ein nicht zu vernachlässigender Vorteil ist jedoch, dass während der Konfiguration der Oracle Service Bus Domain direkt eine Cluster-Umgebung aufgebaut werden kann. Zu den normalen OSB-Servern kann zusätzlich ein Proxy Server konfiguriert werden, der als Load Balancer dient. Somit kann bereits während des Setup eine komplette Load-Balanced- und Failover-abgesicherte Umgebung aufgebaut werden. Voraussetzung ist, dass die Services so konfiguriert werden, dass sie über den Load Balancer auf den OSB zugreifen. Alternativ können Anfragen immer noch direkt an die jeweiligen OSB-Server gesendet werden, jedoch besteht dann keine Ausfallsicherheit.

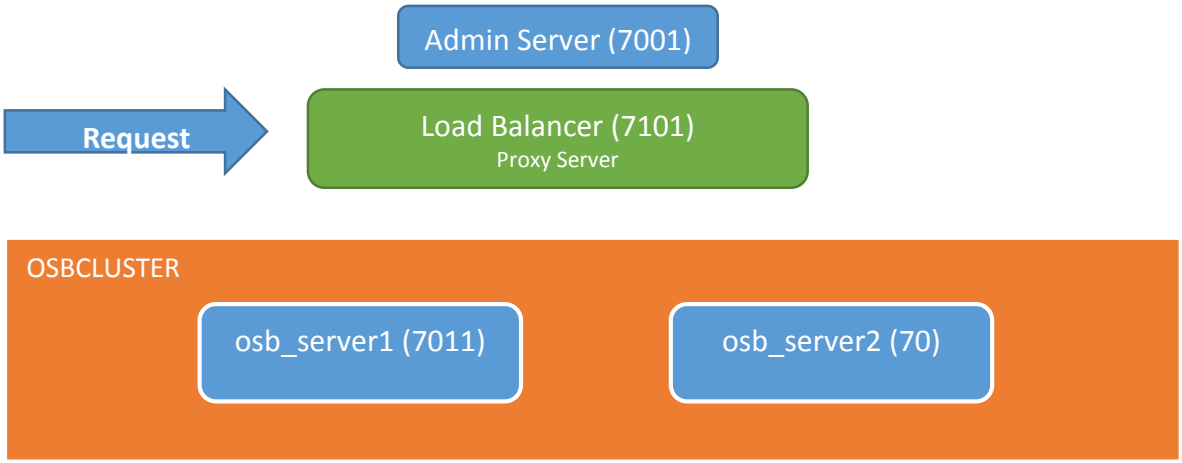


Abbildung 38 OSB Load Balancing und Cluster

Bewertung:
Positive Bewertung (+): Eingebaut
Der Hersteller unterstützt Load Balancing und Ausfallsicherheit direkt innerhalb des Produkts.

3.5.3.3.4 STK4 – Verfügbare Dokumentation Community

Die Installation des Produkts ist aufwändig und zeitintensiv. Die Installation ist jedoch gut dokumentiert. Es existiert eine sehr detaillierte Dokumentation auf der Website des Herstellers[17].

a) **Installationsanleitung**
   Die Installationsanleitung ist umfangreich und trotzdem verständlich. Die Installation ist aufwändig, aber simpel.
b) **Verständlichkeit der Dokumentation**
   Das Produkt hat sehr viele Funktionen, welche im Regelfall dokumentiert sind. Die Dokumentation ist dementsprechend sehr umfangreich. Die einzelnen Kapitel sind verständlich geschrieben.

---

[17] Alle Dokumentation zum OSB sind unter https://docs.oracle.com/middleware/1213/osb/index.html verfügbar (zuletzt aufgerufen am 12.12.2014).

c) **Community**
Oracle bietet ein Forum im Format eines Questions & Answers (Q&A) an. Zusätzlich wird ein Blog aufgeführt.

Ausserdem sind bei Stack Overflow ca. 500 Resultate mit dem Tag „OSB“ und 40 Resultate mit dem Tag „Oracle Service Bus“ zu finden (Stand 05.12.2014).

Bewertung:
Positive Bewertung (+): Umfangreiche Dokumentation und Community

---

3.5.3.3.5 STK5 – Support

Der Hersteller präsentiert auf der Webseite den Support als eine eigene Dienstleistung. Dementsprechend ist das Angebot sehr generisch gehalten und unübersichtlich für Laien. Um dieses Kriterium bewerten zu können, haben wir den Hersteller via Kontaktformular[18] angefragt. Wir haben zwar schnell eine Antwort erhalten, jedoch wurden wir darauf angewiesen, dass wir an der falschen Stelle nachgefragt haben. Wir wurden dann weitergeleitet und zum Schluss darauf hingewiesen, dass wir alle Informationen auf der Webseite finden könnten. Somit haben wir bereits einen Einblick in die Qualität des Supports erhalten. Die restlichen Punkte haben wir recherchiert:

a) **Support-Angebot**
Wir konnten das Support-Angebot nicht verstehen. Jedoch ist auf der Webseite erkennbar, dass mit grosser Wahrscheinlichkeit Support für dieses Produkt existiert.
b) **Qualität des Support Service**
Uns wurde auf die Anfrage nicht hilfreich geantwortet. Dieses Kriterium wurde nicht erfüllt.
c) **Kosten für den Support**
Der Support Service ist kostenpflichtig. Die Kosten sind wahrscheinlich hoch (der Hersteller hat im Durchschnitt hohe Preise in allen Segmenten), wir konnten dies aber nicht verifizieren.

Bewertung:
Negative Bewertung (–): Mangelhafter oder kein Support
Schlechte Erfahrung mit dem Support, wahrscheinlich hohe Kosten

---

3.5.3.4 Bewertung der "Look & Feel"-Kriterien

3.5.3.4.1 LFK1 – Manageability

Der Oracle Service Bus bietet als Web-Konsole die „Oracle Service Bus Console“ [24, p. 91] an. Diese gestaltet sich im Funktionsumfang mächtig und unterstützt viele Funktionen, welche auch in der Entwicklungsumgebung gebraucht werden können.

In der „Oracle Service Bus Console“ gibt es „Service Bus Sessions“. Diese Sessions sind für die Multiuser-Fähigkeit notwendig und verhindern das Überschreiben von Änderungen. Jeder Benutzer arbeitet in einer Sandbox Session und kann dann die Konfiguration einchecken.

---

[18] Online Kontaktformular unter https://apps.instantservice.com/OracleChat/ (zuletzt aufgerufen am 12.12.2014)

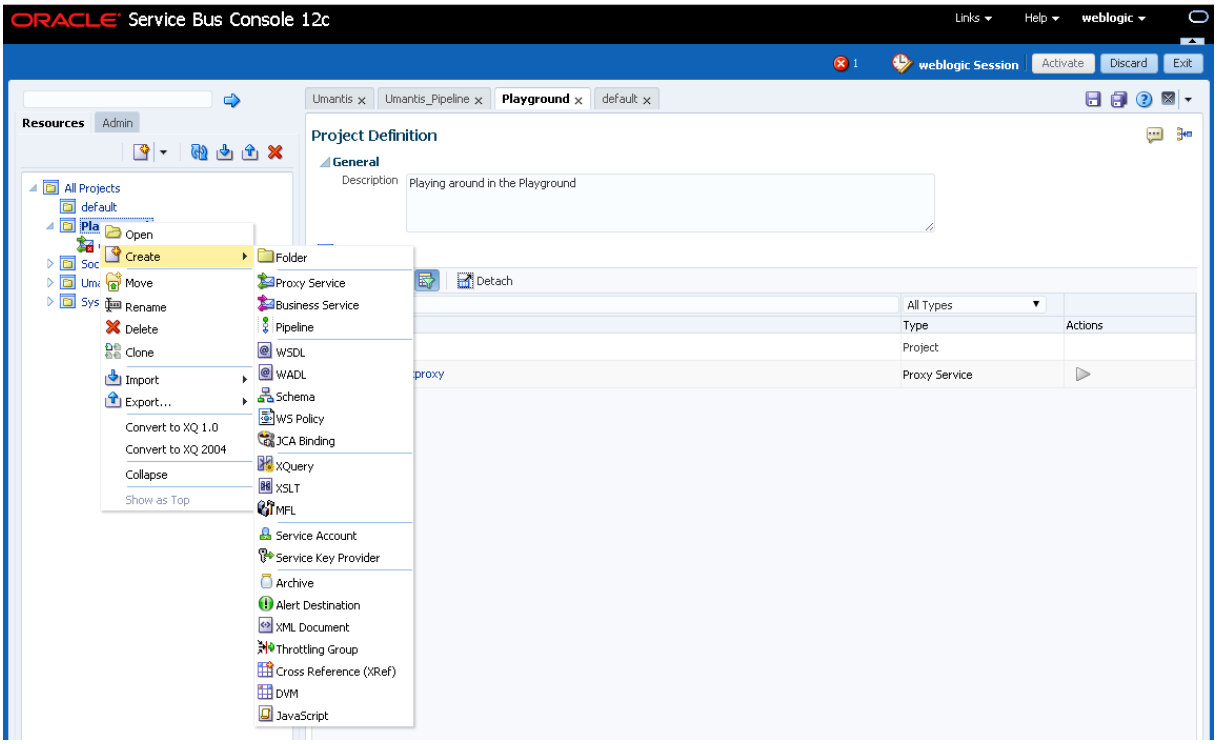


Abbildung 39 Oracle Service Bus Console

Neben der „Oracle Service Bus Console“ gibt es die „WebLogic Server Administration Console“. Da das Produkt in einer WebLogic-Umgebung installiert ist, können die Informationen in dieser Konsole administriert werden.

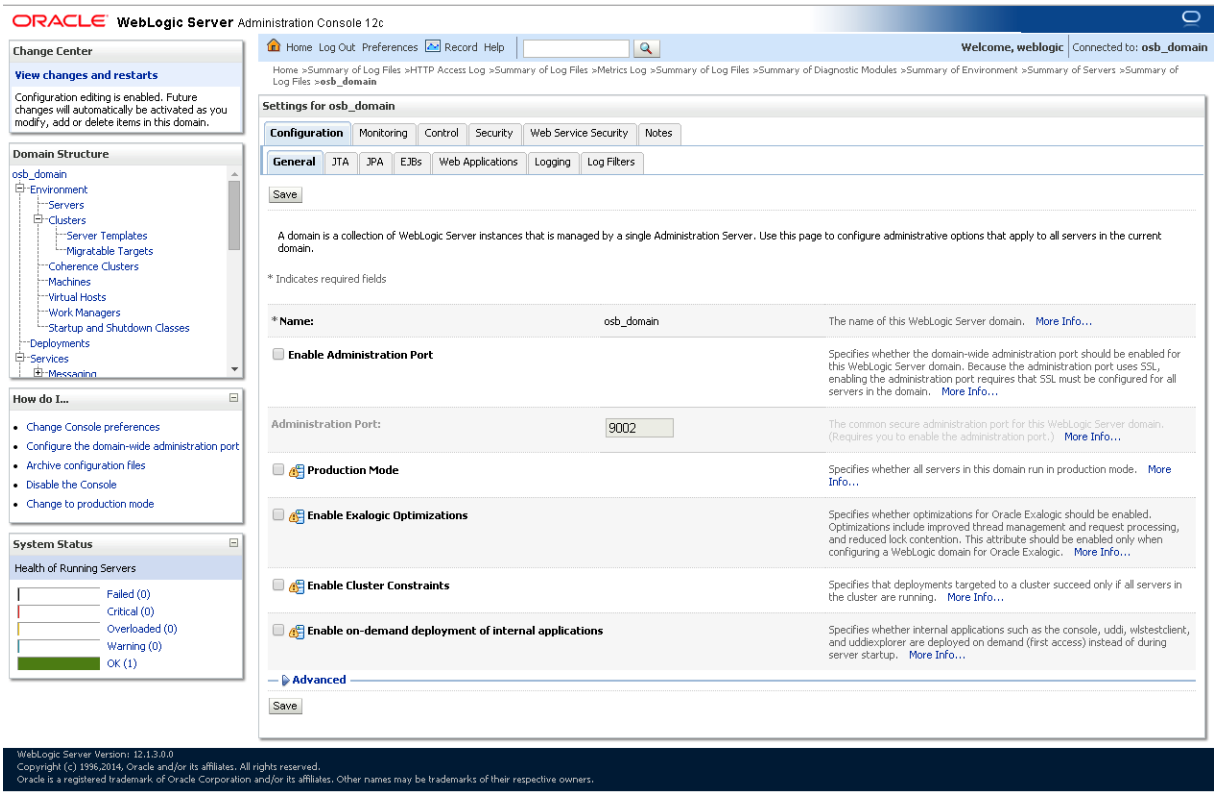


Abbildung 40 WebLogic Server Administration Console

a) Command Line Interface

Es ist möglich, mit Command Line Tools die Konfiguration des Oracle Service Bus zu

exportieren und importieren. Oracle bietet für Exports und Imports Java-Klassen an, welche wie gewohnt aufgerufen werden können.

Oracle hat zudem eine Anleitung[19] in der Dokumentation, wie man mit einem Ant-Target einen Export einrichten kann.

```
java -Xms384m -Xmx768m
-Dosgi.bundlefile.limit=500
-Dosgi.nl=en_US
-Dosb.home=D:/oracle/Oracle_OSB1
-Dweblogic.home=D:/oracle/wlserver_10.3
-Dharvester.home=${osb.home}/harvester
-Dsun.lang.ClassLoader.allowArraySyntax=true
-jar D:/oracle/oepe_11gR1PS1/eclipse/plugins/org.eclipse.equinox.launcher_1.0.201.R35x_v20090715.jar
-data D:/oracle/user_projects/myWorkspace
-application com.bea.alsb.core.ConfigExport
-configProject config
-configJar sbconfig.jar
-configSubProjects OSB Project 1,OSB Project 2
-includeDependencies true
```

Abbildung 41 Beispiel aus der Dokumentation: Export auf CLI

b) Web-Konsole

Es existiert eine Web-Konsole für das Produkt. Diese ist funktionstechnisch mächtig. Konfigurationen lassen sich erstellen, verändern und löschen. Die Web-Konsole ist Multiuser-fähig.

c) Logging

Die Änderungen werden jeweils geloggt und sind sichtbar in der Service Bus Console:

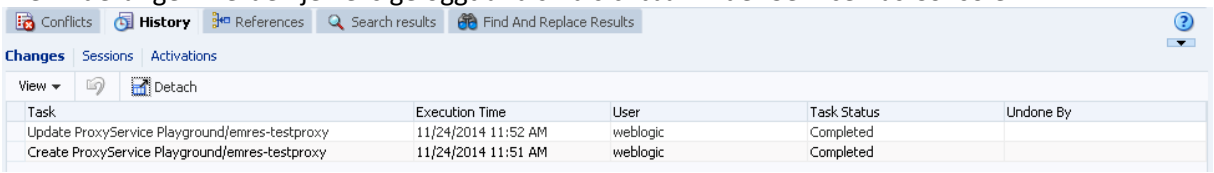


Abbildung 42 Logging der Änderungen

Bewertung:

Positive Bewertung (+): Sehr gute Usability und integrierte Systemmanagement-Funktionen (für Command Line Interface und Web-Konsole) sowie Logging der Aktivitäten

Da die Web-Konsole viele Funktionen anbietet, ist es generell schwierig, eine gute Übersicht anzubieten. Unserer Meinung nach hat dies Oracle aber geschafft. Sie unterstützen ausserdem Export und Import von Konfigurationen. Aktionen werden geloggt.

3.5.3.4.2 LFK2 – Entwicklungsumgebung

Als Entwicklungsumgebung bietet Oracle den JDeveloper an. Die Entwicklungsumgebung ist im gleichen Stil aufgebaut wie andere Tools vom Hersteller (z.B. Oracle SQL Developer).

Die IDE kann in verschiedenen Modi gestartet werden. So kann ein Entwickler die Applikation im „Studio Developer“ starten und hat eine andere Ansicht als ein „Customization Developer“ in der „Customization Developer“-Mode.

---

[19] http://docs.oracle.com/cd/E23943_01/dev.1111/e15866/tasks.htm#OSBDV128 (zuletzt aufgerufen am 12.12.2014)

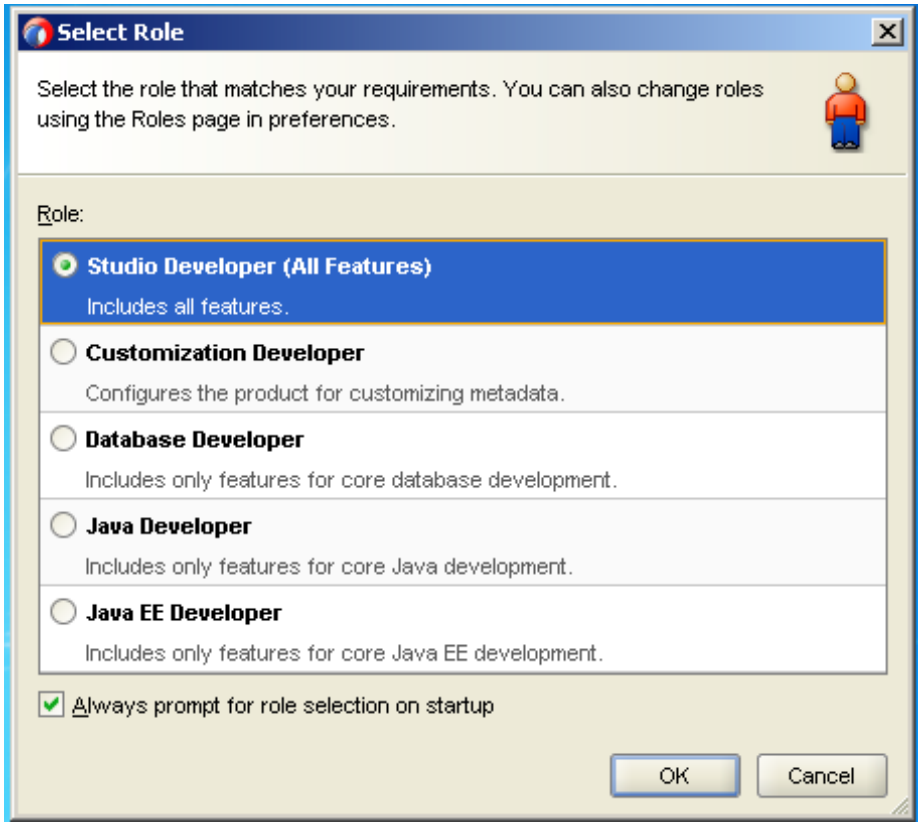


Abbildung 43 Rollenauswahl beim Start von Oracle's JDeveloper

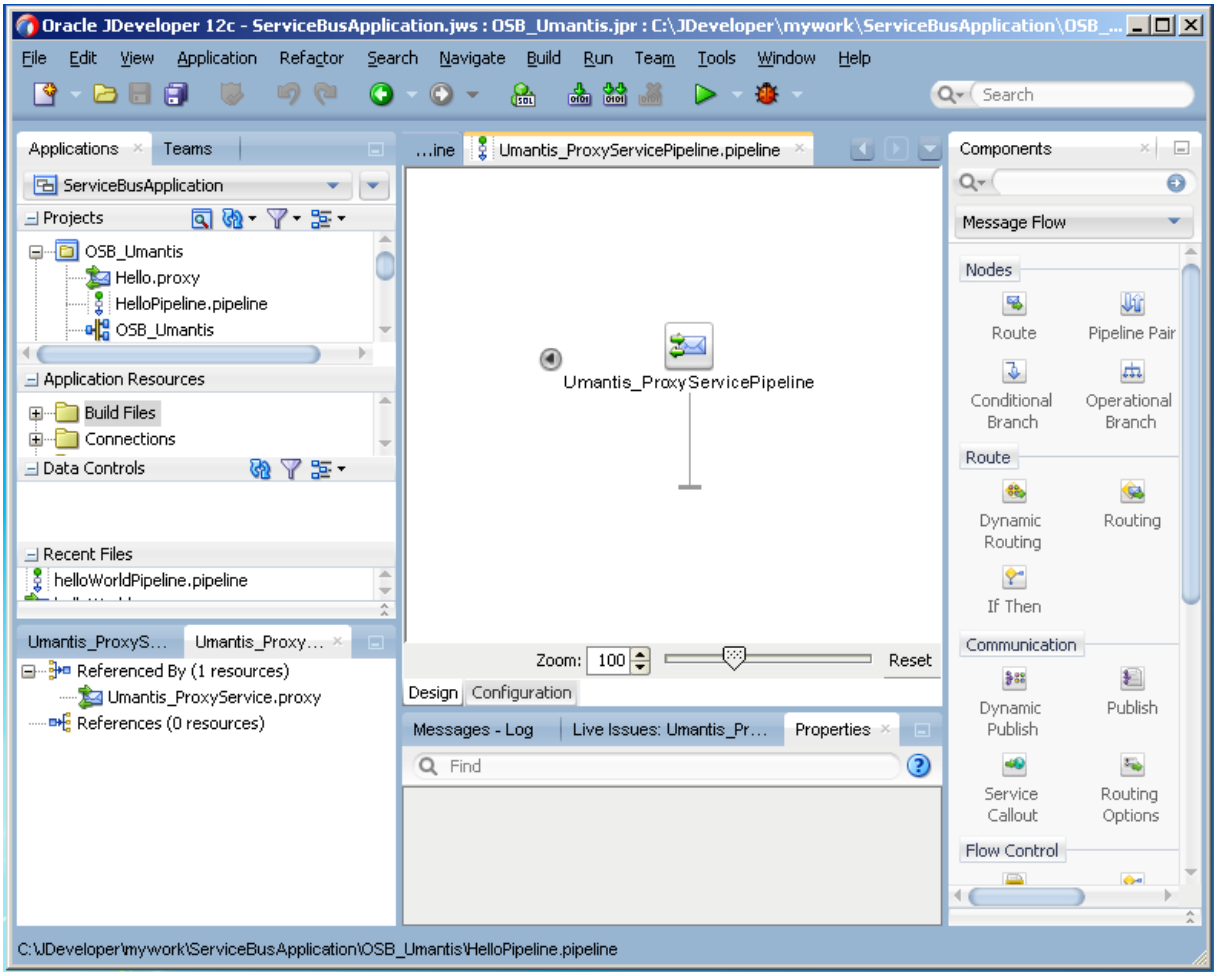


Abbildung 44 JDeveloper in Aktion

Analog zur Web-Konsole ist es auch hier möglich, einzelne Komponenten grafisch zu editieren.

Bewertung:
Positive Bewertung (+): IDE und gute grafische Unterstützung
Grafische Unterstützung wird zusätzlich verbessert mit verschiedenen Ansichten.

3.5.3.4.3 LFK3 – Aufwand Installation / Konfiguration

a) **Aufwand Installation**
   a. **Download**
      Die Webseite ist gut strukturiert, sodass man das Produkt schnell herunterladen kann.
   b. **Installationsvorgang**
      Die Installation gemäss Anleitung [23, p. 17] des Produkts gestaltet sich sehr zeitaufwändig, aber simpel
      i. Herunterladen des Produkts (komplette SOA Suite)
      ii. Archiv entpacken
      iii. Installation der Suite durchführen (keine Komponentenauswahl)

b) **Aufwand Konfiguration**
   Damit der OSB jedoch überhaupt gestartet werden kann, muss nach der Installation eine aufwändige und komplizierte Konfiguration[20] durchgeführt werden:

---

[20] Eine Anleitung zur Konfiguration ist unter http://docs.oracle.com/middleware/1213/core/INOSB/configure_osb.htm#INOSB377 verfügbar (zuletzt aufgerufen am 15.11.2014).

a. *JAVA_HOME* Variable setzen
b. Durchführen der RCU (Repository Creation Utility), um die vorgängig installierte Datenbank vorzubereiten (hier können die entsprechenden Komponenten der SOA ausgewählt werden)
c. Konfigurieren der Oracle Service Bus Domain

Bewertung:
Negative Bewertung (–): Kompliziertes und anspruchsvolles Setup
Ohne Erfahrung mit Produkten von Oracle ist man zwingend auf die Installationsanleitung angewiesen und das Setup ist mit ca. 2 Stunden doch eher zeitaufwändig. Einen weiteren Minuspunkt ergibt die Tatsache, dass zwar mehrere Datenbanksysteme für die SOA-Suite-Installation unterstützt werden, aber nur mit einer Oracle DB die OSB-Komponenten installierbar sind. Während der Installation und in der Anleitung gibt es keinen Hinweis auf diese Tatsache. Erst wenn man sich über ein paar verschachtelte Links auf der Website zu den Requirements durchgeklickt hat, wird diese Einschränkung ersichtlich.

3.5.3.5 Bewertung der allgemeinen Kriterien

3.5.3.5.1 AK1 – Configuration over Coding

Der OSB lässt sich über die Web-Konsole sehr umfangreich konfigurieren, ohne auch nur eine Zeile Code verändern zu müssen. Der Code ist gar nicht erst einsehbar. Um XSLT oder MFL Mappings Files komfortabel zu erstellen, werden im JDeveloper Tools angeboten, welche grafische Unterstützung bieten. Ein Beispiel der angebotenen Tools wurde bereits in FK4 und DK2 aufgezeigt. Auch die Erstellung der Pipeline sowie der Proxy und Business Services wird im JDeveloper etwas ausführlicher grafisch unterstützt als in der Web-Konsole.

Sämtliche Performance- und programminternen Parameter lassen sich über verteilte Konfigurationsfiles verändern. Jedoch sind diese nicht zentral verwaltet und man muss sich mit der Architektur auskennen, da vieles über den WebLogic Server gesteuert wird.

Bewertung:
Positive Bewertung (+): Umfassend konfigurierbar
Alles lässt sich konfigurieren, man muss nicht programmieren können, um die Funktionalität des ESB zu verändern. Der Hersteller bietet eine umfassende Web-Konsole an. Jedoch müssen wir vor dieser Bewertung Warnen, da der OSB auch nur sehr wenige Möglichkeiten für Konfigurationen lässt. Die vielen fehlenden Funktionen müssen extern programmiert und anschliessend über einen JavaCallout in der Pipeline aufgerufen werden. Letztlich wird man nicht um die Programmierung herumkommen, wenn man einen OSB produktiv einsetzen will.

3.5.3.5.2 AK2 – HW-/SW-Anforderungen

Hardware-Anforderungen:	

Memory	Minimal-Anforderung:
	4 GB physisch, 8 GB allgemein
	
	Allgemein gilt die Formel gemäss Oracle:
	
	`3 GB of available memory for the operating system and other software`
	`+ 3 GB of available memory for each Managed Server`
	`6 GB Total required available memory`

Harddisk	1.4 GB Fusion Middleware Infrastructure + 582 MB WebLogic Server Installation --- 2 GB Total required harddisk space Anmerkung: für die Installation ist ca. 2,5-mal so viel Speicherplatz notwendig.
CPU	Mindestens 300MHz
Software-Anforderungen:	
Betriebssystem	Linux x86-64 (Oracle und RedHat) Microsoft Windows x64 Apple Mac OS X (Intel, 64-bit)
Java	JDK 1.7.0_51+ aufwärts
Datenbank	Für komplette Unterstützung[21] der Oracle Fusion Middleware Infrastructure Server Features: Oracle Database (12.1.0.1+, 11.1.0.7+, 11.2.0.3+,)

Tabelle 14 Oracle Service Bus HW-/SW-Anforderungen [23, p. 11]

Bewertung:
Neutrale Bewertung (o): Erweiterte HW-Anforderungen und externe Software werden vorausgesetzt

3.5.3.5.3 AK3 – Modularer Aufbau der Applikation [NFA1]

Der OSB selber lässt sich nicht modular installieren. Die einzige Auswahl, die man während der Installation hat, ist die zwischen den SOA-Suite-Komponenten, da der OSB ein Teil davon ist. Jedoch kann er nur als Ganzes ausgewählt werden.

Bewertung:
Negative Bewertung (–): Kein modularer Aufbau und/oder Lizenzierung

3.5.3.5.4 AK4 – Kosten

Der OSB setzt sich aus mehreren Komponenten zusammen, die zwingend lizenziert werden müssen. Folglich kann auch bei der Lizenzierung nicht modular ausgewählt werden.

Bewertung:
Negative Bewertung (–): Hohe Kosten

3.5.4 Implementation

Die Implementation des Use Case war etwas anspruchsvoll, da die Vielfalt an Templates für den OSB 12g noch nicht sehr umfangreich ist und man sich doch zuerst mit dem Produkt vertieft auseinandersetzen muss. Letztlich hat für das Verständnis das standardmässig mitgelieferte Beispiel geholfen, in dem vor allem auf die Testmöglichkeiten eingegangen wurde. Auf der Basis dieses Beispiels konnten wir dann unsere Verbindung zu Umantis schrittweise aufbauen.

3.5.4.1 Use Case-Setup

3.5.4.1.1 Projektumsetzung

[21] Für den Datenbankteil „Application Data Access“ wird ferner auch eine JavaDB oder beliebige Datenbank mit einem JDBC Treiber unterstützt.

Für die Erstellung und die Konfiguration aller folgenden Komponenten verwendeten wir die Web-Konsole des OSB.

Ein Service kann innerhalb des OSB als Projekt zusammengefasst werden. Wir haben entsprechend ein Projekt mit dem Namen Umantis und darin die vier notwendigen Komponenten erstellt:

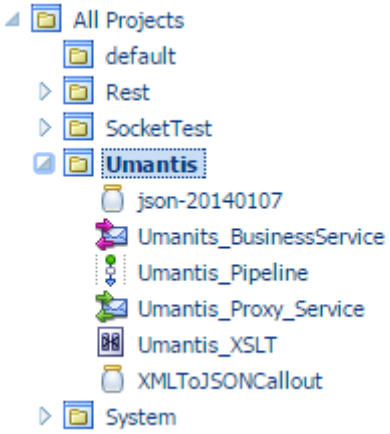


Abbildung 45 OSB-Umantis-Projekt

Komponente	Beschreibung
Umantis_ProxyService	Der Proxy Service ist als Endpoint für den Aufruf vom ITC Client zuständig. Er nimmt die Anfrage entgegen und sendet die modifizierte Antwort wieder an den Client. Wir verwendeten die Standard-Parameter mit den folgenden Änderungen: *General* Protocol: HTTP Service Type: Any XML Service Target: Umantis_Pipeline *Transport* Protocol: HTTP Endpoint URI[22]: /Umantis/Umantis_Proxy_Service
Umantis_Pipeline	Die Pipeline regelt den Verkehr innerhalb des OSB. Sie leitet die Anfrage an den richtigen Business Service, nimmt sie wieder entgegen und transformiert sie. Die verwendete Pipeline läuft wieder mit der Standard-Konfiguration mit der folgenden Ergänzung: *Service Type* Service Type: Any XML Service *Message Flow* Die Pipeline besteht aus einem Route Node, welcher fix zum Umantis_BusinessService routet und den Body der Antwort mittels XSLT entsprechend Use Case transformiert.

[22] Der Service ist demnach unter http://*hostname*:7003/Umantis/Umantis_Proxy_Service aufrufbar.

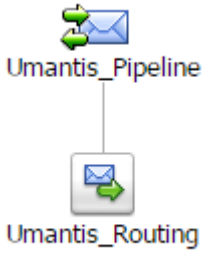


Abbildung 46 OSB Message Flow

Es ist nur nötig, Response Actions zu definieren. Für den Request ist keine Bearbeitung notwendig. Folgende Actions werden in der Pipeline durchlaufen:

Replace

Über den XPath-Ausdruck „//.“ wird der komplette Inhalt aus dem XML Body ausgewählt, welcher komplett mit dem XSLT-transformierten Text ersetzt wird (replace node contents).

Java Callout

Über den Java Callout wird die Methode *convertToJSON* der XMLToJSONCallout Library aufgerufen. Übergeben wird der Parameter *fn-bea:serialize($body).* Dieser serialisiert den Inhalt des kompletten Bodys und wird als String an die Java Methode übergeben. Das Resultat wird in der Variabel *$calloutJSON* als String zurückgegeben.

Set Transport Header

Hier werden die beiden Header Parameter manuell angepasst. Dies ist notwendig, weil sonst der Content-Type immer noch als Text übergeben werden würde und nicht als JSON.

Content-Type	*application/json*
Content-Encoding	*UTF-8*

Replace

Der Node Inhalt des Bodys der Response Message wird mit dem Inhalt der Variabel *$calloutJSON* ersetzt.

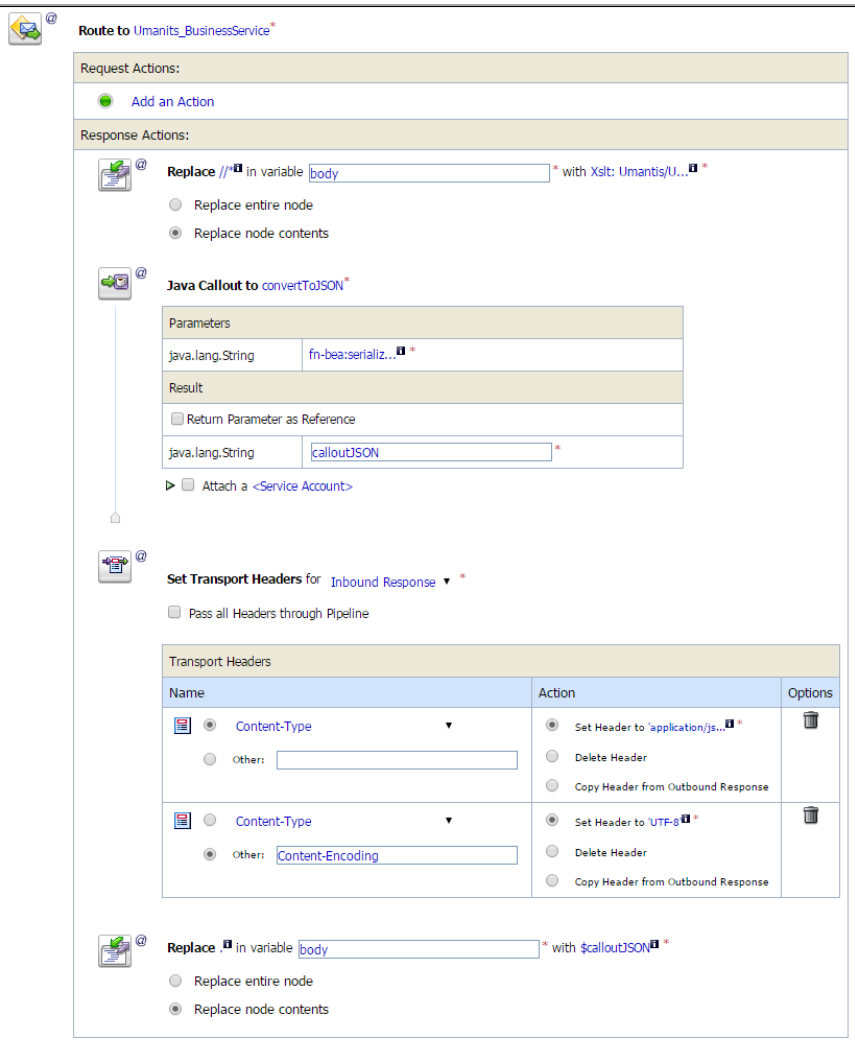


Abbildung 47 OSB Routing Details

Umantis_BusinessService

Der Business Service regelt die Verbindung zum Producer Endpoint, dem Umantis-System. Er nimmt die Anfragen von der Pipeline entgegen und leitet sie an Umantis weiter. Dasselbe gilt für den umgekehrten Transportweg von Umantis zur Pipeline. Wir verwendeten die Standard-Parameter mit den folgenden Änderungen:

General

Protocol:	HTTP
Service Type:	Any XML Service
Target:	Umantis_Pipeline

Transport

Protocol:	HTTP
Load Balancing:	None
Endpoint URI[23]:	http://10.10.10.10/index.xml

Transport Details

Read Timeout	10
Connection Timeout	10
HTTP Request Method	POST
Authentication	None

[23] Hier wird zu Testzwecken auf einen lokalen Server zugegriffen, welchen wir für unsere Projektarbeit als VM erstellten. Diese URI kann einfach mit der produktiven Umantis URI ausgetauscht werden.

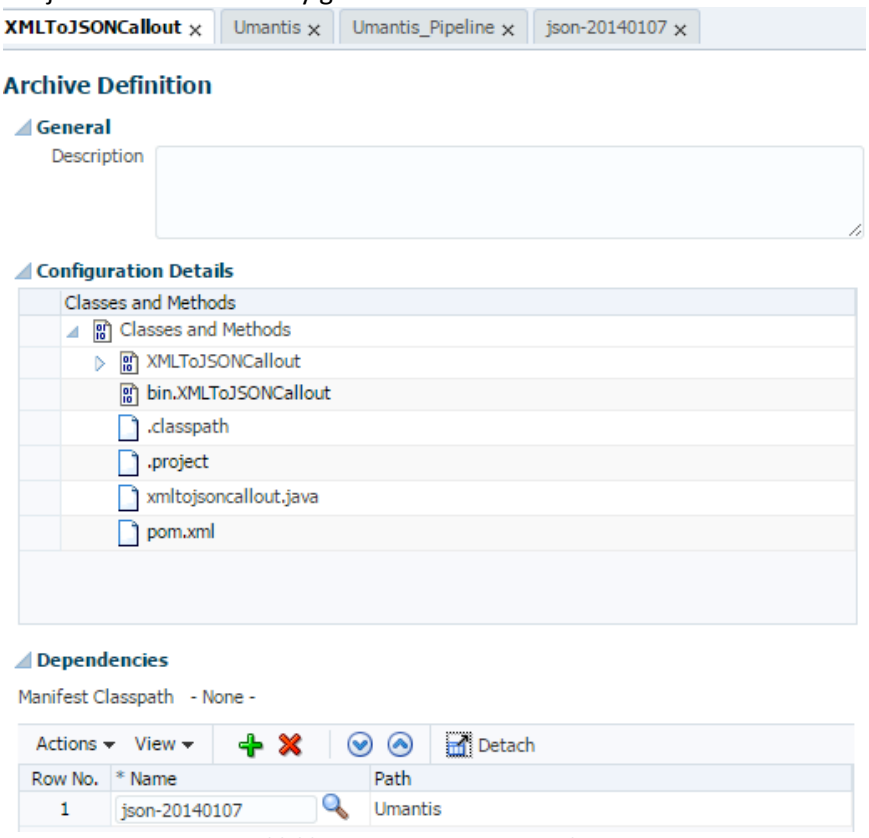
Umantis_XSLT	Um von der Pipeline auf ein XSLT-File zugreifen zu können, wird zuerst in der Admin-Konsole ein XSLT-Objekt erstellt. Dort kann der Code in Plain Text eingegeben oder aus einem Textfile geladen werden.
XMLToJSONCallout **json-20140107**	Das XMLToJSONCallout Archiv beinhaltet das Java-Programm für die Konvertierung von XML zu JSON. Zusätzlich muss eine Dependency auf die json-20140107 Library gemacht werden.  *Abbildung 48 OSB XMLToJSON Library*

Tabelle 15 Konfiguration des Use Case im Oracle Service Bus

3.5.4.1.2 XSLT Transformation

Für die XSLT Transformation konnten wir denselben Code wie bereits beim WSO2 verwenden. Dies hat uns viel Zeit erspart. Der Code wird hier darum nicht noch einmal aufgeführt.

Von der Pipeline aus kann der Expression Builder aufgerufen werden. Dort wird über den Unterpunkt XSLT Resources das XSLT-Objekt eingebunden und der zu transformierende Teil aus dem Dokument ausgewählt ($body).

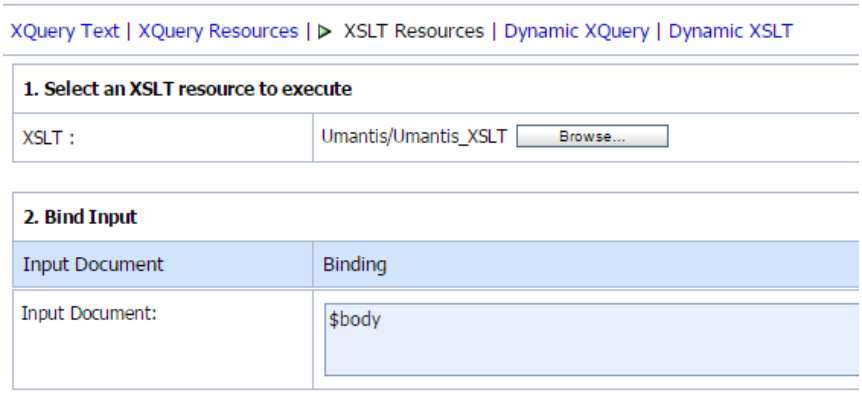


Abbildung 49 OSB XSLT Resource Selection

3.5.4.1.3 JSON-Konvertierung

Um von XML nach JSON zu konvertieren haben wir ein eigenes kleines Programm geschrieben mit der Klasse *XMLToJSONCallout*. Dieses verwendet eine Open Source JSON for Java Library[24]. Mittels Java Callout wird aus der Pipeline heraus die Methode *convertToJson* aufgerufen.

```
import java.io.Serializable;

import org.json.JSONException;
import org.json.JSONObject;
import org.json.XML;

/*
 * This class is needed to convert xml to json as callout
 * It also removes all xml wrapped around <person>
 */

public class XMLToJSONCallout implements Serializable {
    private static final long serialVersionUID = 1L;

    public static int PRETTY_PRINT_INDENT_FACTOR = 4;

    public static String convertToJSON(String xml) {
        String toReturn = "";

        try {
            JSONObject json = XML.toJSONObject(xml).getJSONObject("Body").getJSONObject("export");
            json.remove("xmlns");
            toReturn = json.toString(PRETTY_PRINT_INDENT_FACTOR);
        }
        catch (JSONException je) {
            System.out.println(je.toString());
        }

        return toReturn;
    }
}
```

Code 6 Java-Programm zur Konvertierung von XML zu JSON

Damit der OSB während der Laufzeit auch auf die JSON Library zugreifen kann, muss diese zusätzlich im <Domain_Home>/lib Verzeichnis[25] der OSB Domain abgelegt werden.

[24] JSON Library für Java http://www.json.org/java/ (zuletzt aufgerufen am 15.12.2014)

[25] Anmerkung: Dies war in keiner Dokumentation oder bei den Samples erwähnt und hat uns sehr viel Zeit gekostet. Nur per Zufall stiessen wir auf diese Lösung in einem Forum.

4 Auswertung

4.1 Vergleich der ESB-Lösungen

Um einen Quervergleich der beiden ESB-Lösungen zu erhalten, sind nochmals alle Bewertungsergebnisse zusammengefasst und gewichtet. Die Gewichtung orientiert sich an der Bewertungsbeschreibung der jeweiligen Kriterien in Kapitel 3.3.

+ Positive Bewertung: Kriterium voll erfüllt
o Neutrale Bewertung: Kriterium teilweise erfüllt
– Negative Bewertung: Kriterium nicht erfüllt

4.1.1 Bewertungsübersicht

Kriterium	WSO2 ESB	Bewertung	Oracle ESB	Bewertung
Funktionale Kriterien				
FK1 – Unterstützte Protokolle/Technologien und Adapter [FA1]	Standardprotokolle plus zusätzliche Adapter/Konnektoren zu Third-Party Services	+	Standardprotokolle plus zusätzliche Adapter/Konnektoren zu Third-Party Services	+
FK2 – Custom Adapter Framework [FA5]	Gute Unterstützung	+	Gute Unterstützung	+
FK3 – Security [FA3, FA4]	Hoher Security-Standard	+	Hoher Security-Standard	+
FK4 – Message Translation	Gute Unterstützung	+	Mittlere Unterstützung der Formate	o
FK5 – Datenstrukturtiefe	Mehrfache Verschachtelungen werden unterstützt	+	Mehrfache Verschachtelungen werden unterstützt	+
FK6 – Message-Durchsatz / Concurrent Streams	Erweiterte synchrone Verarbeitung	+	Erweiterte synchrone Verarbeitung	+
FK7 – Betreibbarkeit in Cloud	Stateless- und Cloud-Erfahrung vorhanden	o	Stateless- und Cloud-Erfahrung vorhanden	o
FK8 – Discovery and Registry	*Nicht bewertbar*			
FK9 – Workflow Engine and Service Composition	*Nicht bewertbar*			
FK10 – Versionierung der Komponenten	Keine Versionierung unterstützt.	–	Versionierung der Komponenten und Betrieb von multiplen Versionen unterstützt	+
Design-Kriterien				
DK1 – Message Routing Patterns	Alle sechs Patterns beliebig kombinieren und dynamisches Routing	+	Teil der Pattern und beliebige Kombinationsmöglichkeit	o
DK2 – Message Content Transformation Pattern	Content Enricher und Content Filter werden unterstützt. Die Modellierungssprache ist standardisiert und somit nicht herstellerspezifisch (Nur XSLT)	+	Content Enricher und Content Filter werden unterstützt. Die Modellierungssprache ist standardisiert und somit nicht herstellerspezifisch (XSLT und XQuery)	+
DK3 – Message Consumer Patterns [FA2]	Alle 8 Patterns werden unterstützt	+	Alle 8 Patterns werden unterstützt	+
Support- und Troubleshooting-Kriterien				

STK1 – Reporting and Statistic	Message Flow und Konfigurations-änderungen, nützliche Aufbereitung	+	Message Flow und Konfigurations-änderungen, nützliche Aufbereitung	+
STK2 – Alerting and Monitoring	Monitoring und Alerting (SNMP Traps oder Nachbau) vorhanden	+	Monitoring und Alerting (sehr gut) unterstützt	+
STK3 – Failover / Load Balancing	Unterstützt durch externe Software	o	Eingebaut	+
STK4 – Verfügbare Dokumentation Community	Umfangreiche Dokumentation und Community	+	Sehr umfangreiche Dokumentation und gute Erläuterung der Beispiele (Tendiert fast zu Überdokumentation)	+
STK5 – Support	Standard-Support	o	Mangelhafter oder kein Support Schlechte Erfahrung mit dem Support, hohe Kosten	–
"Look & Feel"-Kriterien				
LFK1 – Manageability	Solide Web-Konsole vorhanden, gutes Logging, kein Command Line Interface	o	Sehr gute Usability und integrierte Systemmanagement-Funktionen (für Command Line Interface und Web-Konsole) sowie Logging der Aktivitäten	+
LFK2 – Entwicklungsum-gebung	IDE und gute grafische Unterstützung	+	IDE und gute grafische Unterstützung (Grafische Unterstützung wird zusätzlich verbessert mit verschiedenen Ansichten)	+
LFK3 – Aufwand Installation / Konfiguration	Einfaches und simples Setup	+	Kompliziertes und anspruchsvolles Setup	–
Allgemeine Kriterien				
AK1 – Configuration over Coding	Teils konfigurierbar	o	Umfassend konfigurierbar	+
AK2 – HW-/SW-Anforderungen	Externe Software (JDK 1.6.x) wird vorausgesetzt	o	Erweiterte HW-Anforderungen und externe Software werden vorausgesetzt	o
AK3 – Modularer Aufbau der Applikation [NFA1]	Komplett modularer Aufbau	+	Kein modularer Aufbau und/oder Lizenzierung	–
AK4 – Kosten	Open Source, keine Zusatzprodukte nötig	+	Hohe Kosten	–

Tabelle 16 Vergleich der Bewertungskriterien beider Produkte

4.2 Änderungsbedarf bei einem Wechsel

Wenn man sich für ein Produkt entscheidet, bindet man sich in der Regel stark an den Hersteller. Dies ist in den meisten Fällen nicht unerwünscht und nicht problematisch. Jedoch müssen die Konsequenzen trotzdem bei der Wahl klar bedacht werden. In diesem Kapitel betrachten wir den Vendor Lock-in und untersuchen zwei konkrete Fälle: die Wechsel zwischen den beiden Systemen (bi-direktional).

Bei einem Wechsel ist zu beachten, dass es sich beim WSO2 um einen leichtgewichtigen ESB und beim OSB um einen schwergewichtigen (viele Features und viele Möglichkeiten, aber nicht überschaubar) handelt.

Zusätzlich zur herstellerspezifischen Konfiguration, die angepasst werden muss, haben wir die jeweiligen Eigenschaften aus dem Kriterienkatalog überprüft. Dabei haben wir uns auf die Unterschiede zwischen den zwei Produkten konzentriert. Die Resultate sind in der Tabelle im zugehörigen Kapitel zu finden. In der Tabelle werden lediglich die Konsequenzen eines Wechsels (z.B. nicht mehr unterstützte Patterns oder Funktionen) aufgeführt.

Aus unserer Use Case-Erfahrung mit den beiden Produkten können wir sagen, dass grundsätzlich nur das XSLT wiederverwendet werden konnte. Da die Architektur der beiden Produkte komplett unterschiedlich ist, mussten alle Objekte und Bausteine neu erstellt werden.

Wiederverwendung finden daher nur ein paar wenige Komponenten, welche allgemein sind (Third-Party-Deklarationen):

- XML-, Query- und WSDL-Schemas/Deklarationen
- XSLT-Files
- XPath-Deklarationen
- WS Security Policies
- Externe Java-Programme und Schnittstellen

Nicht zu vernachlässigen bei einem solchen Wechsel sind auch die viel höheren Anforderungen an die Hardware und die Lizenzkosten. Ebenfalls ist es notwendig, einiges an Produkte-Know-how für den OSB und Oracle allgemein aufzubauen, da auch eine WebLogic Domain und eine Oracle DB betrieben werden müssen.

Aus logischer Sicht sind Überlegungen zu Anpassungen bei den folgenden Kriterien erforderlich:

Legende für die Spalte „Braucht Anpassung“:
x = zwingend
o = optional (implementationsabhängig oder zusätzliche Features sind möglich)
✓ = keine Änderungen nötig

4.2.1 WSO2 ESB zu Oracle Service Bus

In der unten ersichtlichen Tabelle haben wir die Änderungen aufgelistet, welche beim Wechsel vom Open Source Produkt WSO2 ESB zum kommerziellen Produkt Oracle Service Bus zu beachten sind.

Kriterium	Braucht Anpassung	Änderung / Kommentar
FK1 – Unterstützte Protokolle/Technologien und Adapter	x	Adapter zu Third-Party Services wie Cloud Services JIRA, Google Spreadsheet, Salesforce, Twitter und Twilio können im Oracle Service Bus nicht standardmässig konfiguriert werden. Sie müssten beim Einsatz entweder selber implementiert oder weggelassen werden. Zusätzlich sind die folgenden Protokolle / Technologien nicht standardmässig vom Oracle Service Bus unterstützt: • FIX (Financial Information eXchange) • HL7 (Health Level 7 International)
FK2 – Custom Adapter Framework	x	Custom Adapter müssen auf den OSB migriert und folglich neu geschrieben werden, da der Aufbau eines Adapters sich in der Architektur unterscheidet. Java-Klassen, welche keine WSO2 ESB-spezifischen Bibliotheken brauchen, können wiederverwendet werden.
FK3 – Security	o	Der Oracle Service Bus unterstützt ebenfalls Zertifikate. Diese können im Idealfall übernommen werden.
FK4 – Message Translation	✓	XSLT kann wiederverwendet werden.
FK5 – Datenstrukturtiefe	✓	Kein Änderungsbedarf
FK6 – Message-Durchsatz / Concurrent Streams	✓	Kein Änderungsbedarf
FK7 – Betreibbarkeit in Cloud	o	Der Oracle Service Bus eignet sich besser (bessere Unterstützung der IDEAL-Eigenschaften) für den Betrieb in der Cloud. Beim Wechsel kann die Eigenschaft „Distributed“ zusätzlich konfiguriert werden. Bestehende Eigenschaften erfüllt der Oracle Service Bus.
FK8 – Discovery and Registry		*Nicht anwendbar, da nicht im Funktionsumfang der Studienarbeit*
FK9 – Workflow Engine and Service Composition		*Nicht anwendbar, da nicht im Funktionsumfang der Studienarbeit*
FK10 – Versionierung der Komponenten	o	Zusätzliches Feature des OSB, welches vom WSO2 ESB nicht unterstützt wird.
DK1 – Message Routing Patterns	x	Im Oracle Service Bus können die meisten Message Routing Patterns nachgebaut werden. Beim Wechsel müssen diese angepasst werden. Nicht unterstützt werden hingegen Resequencer und Aggregator. In diesem Fall muss bei der Benutzung auf alternative Message Routing Patterns, welche unterstützt werden, zurückgegriffen werden.
DK2 – Message Content Transformation Pattern	o	Nur Änderungsbedarf, falls mit dem Enrich Mediator gearbeitet wurde, nicht aber beim XQuery oder XSLT Mediator.
DK3 – Message Consumer Patterns	o	Kein Änderungsbedarf in der Logik, da alle Patterns unterstützt werden. Jedoch muss die Programmierung neu gemacht werden.

STK1 – Reporting & Statistic	o	Nur optionaler Änderungsbedarf, da beide Produkte detaillierte Auswertungen liefern.
STK2 – Alerting and Monitoring	x	Beide Produkte unterstützen für das Monitoring und Alerting JMX und SNMP. Da MBeans und SNMP Straps aber unterschiedliche Daten beinhalten können, muss die Implementation angepasst werden.
STK3 – Failover / Load Balancing	x	Für Failover und Load Balancing wird im WSO2 ESB ein eigenes Produkt (WSO2 Elastic Load Balancer) angeboten. Dieses Produkt ist mit grosser Wahrscheinlichkeit nicht kompatibel mit dem Oracle Service Bus oder interagiert nicht ideal damit. Dieses Produkt muss also nicht mehr gebraucht werden und kann vom Ziel entfernt werden. Der Oracle Service Bus unterstützt einen integrierten Load Balancer. Diese Komponente der Software muss nach der Installation konfiguriert werden.
STK4 – Verfügbare Dokumentation Community		*Nicht anwendbar*
STK5 – Support		*Nicht anwendbar*
LFK1 – Manageability		*Nicht anwendbar, da spezifisch für das Produkt.* *Es wird ein anderes Command Line Interface, Web-Konsole und Logging angeboten. Jedoch stellen beide Hersteller Konfigurationsmöglichkeiten in allen drei Bereichen zur Verfügung.*
LFK2 – Entwicklungsumgebung	x	Die Entwickler müssen die JDeveloper-Entwicklungsumgebung kennenlernen.
LFK3 – Aufwand Installation / Konfiguration		*Nicht anwendbar*
AK1 – Configuration over Coding	o	Der OSB bietet mehr Konfigurationsmöglichkeiten.
AK2 – HW-/SW-Anforderungen	x	Der Oracle Service Bus hat höhere Hardware-Anforderungen (mind. 6GB RAM und 2GB Harddisk-Speicher). Diese müssen auf jeden Fall eingehalten werden, um Performanceeinbussen zu meiden. Der Oracle Service Bus benötigt ausserdem eine Instanz von Oracle Database. Diese muss ebenfalls installiert werden. Zusätzlich ist ein Update auf Java Version 1.7.0_51 (oder höher) nötig. OSB: RAM Total 6GB, HDD Total 2GB
AK3 – Modularer Aufbau der Applikation		*Nicht anwendbar, da komplett unterschiedliche Architektur*
AK4 – Kosten	x	Beim Wechsel auf den Oracle Service Bus müssen die Verantwortlichen der IT-Abteilung mit zusätzlichen Kosten rechnen. Diese entfallen bei WSO2 ESB, da die Software Open Source ist.

Tabelle 17 Änderungsbedarf beim Wechsel von WSO2 ESB zu Oracle Service Bus aufgeteilt nach Kriterien

4.2.2 Oracle Service Bus zu WSO2 ESB

Bei einem Wechsel vom Oracle Service Bus zum WSO2 ESB können grundsätzlich die gleichen Komponenten wie in Kapitel 4.2.1 erwähnt übernommen werden. Auch in diesem Fall müssen alle Komponenten einer Verbindung neu erstellt werden, da die Architektur in den beiden Systemen jeweils komplett unterschiedlich ist. In der Tabelle unten ist der Änderungsbedarf bei einem Wechsel aufgelistet.

Kriterium	Braucht Anpassung	Änderung / Kommentar
FK1 – Unterstützte Protokolle/Technologien und Adapter		
FK2 – Custom Adapter Framework	x	Custom Adapter müssen auf den WSO2 ESB migriert und folglich neu geschrieben werden, da der Aufbau eines Adapters sich in der Architektur unterscheidet. Java-Klassen können grundsätzlich wiederverwendet werden.
FK3 – Security	o	Der WSO2 ESB unterstützt ebenfalls Zertifikate. Diese können im Idealfall übernommen werden.
FK4 – Message Translation	o	XSLT-Files können übernommen werden. Jedoch unterstützt der WSO2 ESB keine MFL-Files. Textfiles wie CSV müssen neu mit einem Smooks-Konfigurationsfile übersetzt werden.
FK5 – Datenstrukturtiefe	✓	Kein Änderungsbedarf
FK6 – Message-Durchsatz / Concurrent Streams	✓	Kein Änderungsbedarf
FK7 – Betreibbarkeit in Cloud	x	Beide Produkte unterstützen die Betreibbarkeit in der Cloud. Jedoch bietet der WSO2 ESB im „Distributed“-Kriterium eingeschränktere Möglichkeiten.
FK8 – Discovery and Registry		*Nicht anwendbar, da nicht im Funktionsumfang der Studienarbeit*
FK9 – Workflow Engine and Service Composition		*Nicht anwendbar, da nicht im Funktionsumfang der Studienarbeit*
FK10 – Versionierung der Komponenten	x	Die Versionierung der Komponenten kann mit dem WSO2 ESB nicht mehr umgesetzt werden.
DK1 – Message Routing Patterns	o	Beim Wechsel stehen nun alle Enterprise Integration Patterns für das Routing zur Verfügung. Dementsprechend müssen die Routings nur nachgebaut, jedoch logisch nicht angepasst werden.
DK2 – Message Content Transformation Pattern	✓	Kein Änderungsbedarf, da XQuery und XSLT Transformationen unterstützt werden.
DK3 – Message Consumer Patterns	✓	Kein Änderungsbedarf in der Logik, da alle Patterns unterstützt werden.
STK1 – Reporting and Statistic	o	Nur optionaler Änderungsbedarf, da beide Produkte detaillierte Auswertungen liefern.
STK2 – Alerting and Monitoring	x	Beide Produkte unterstützen für das Monitoring und Alerting JMX und SNMP. Da MBeans und SNMP Straps aber unterschiedliche Daten beinhalten können, muss die Implementation angepasst werden. Da der OSB zusätzliche Alerting-Funktionen direkt über die Pipeline ermöglicht, muss die Planung diesbezüglich angepasst werden.
STK3 – Failover / Load Balancing	x	Der WSO2 hat keine integrierte Lösung. Es muss auf ein externes Produkt zurückgegriffen werden. Empfohlen wird der herstellereigene WSO2 Elastic Load Balancer.

STK4 – Verfügbare Dokumentation Community		*Nicht anwendbar*
STK5 – Support		*Nicht anwendbar*
LFK1 – Manageability		*Nicht anwendbar, da spezifisch für das Produkt.* *Es wird ein anderes Command Line Interface, Web-Konsole und Logging angeboten. Jedoch stellen beide Hersteller Konfigurationsmöglichkeiten in allen drei Bereichen zur Verfügung.*
LFK2 – Entwicklungsumgebung	x	Die Entwickler müssen die auf Eclipse basierende IDE von WSO2 kennenlernen.
LFK3 – Aufwand Installation / Konfiguration		*Nicht anwendbar*
AK1 – Configuration over Coding	x	Der WSO2 bietet weniger Konfigurationsmöglichkeiten als der OSB.
AK2 – HW-/SW-Anforderungen	✓	Da die HW-Anforderungen viel geringer sind und keine zusätzliche Software benötigt wird, müssen keine Änderungen vorgenommen werden.
AK3 – Modularer Aufbau der Applikation		*Nicht anwendbar, da komplett unterschiedliche Architektur*
AK4 – Kosten	✓	Da der WSO2 auf Open-Source-Lizenzbasis läuft, können Kosten eingespart werden und es sind keine Änderungen notwendig.

Tabelle 18 Änderungsbedarf beim Wechsel von Oracle Service Bus zu WSO2 ESB aufgeteilt nach Kriterien

4.3 Empfehlung

4.3.1 Anwendungsfall für Einsatz des Produkts WSO2 ESB

Bei unserer ersten Erfahrung mit dem WSO2 ESB konnten wir feststellen, dass dieser sich am besten für kleine bis mittelgrosse Systemumgebungen eignet. Dies gilt speziell, wenn dort die ersten Erfahrungen mit einem ESB gemacht und hauptsächlich Webapplikationen angeschlossen werden sollen. Der Grund dafür ist, dass man sich ohne vertieftes Produkte-Know-how schnell zurechtfindet und einem eine sehr grosse Online-Community hilfreich zur Seite steht.

Auch stellt der WSO2 sehr viele Beispiele sowie eine EIP-Integrationsanleitung zur Verfügung. Viele Mitarbeiter führen dazu Blogs, in welchen sie auf weitere Spezialprobleme und Implementationsszenarien eingehen. Für mittelgrosse Systemumgebungen stellt der WSO2 zusätzlich erweiterte Lösungen für Load Balancing, Failover und Monitoring bereit. Alle diese Komponenten laufen unter der Apache License Version und sind somit kostenfrei erhältlich.

Technisch sehen wir die Stärken dieses ESB vor allem in der Anbindung von Web- und SOAP Services, dank guter Unterstützung von XML- und JSON-Format sowie von RESTful HTTP Services. Ebenfalls ist deren Transformation und Translation ohne viel Aufwand und Wissen durchführbar und die Webservices lassen sich mittels vordefinierten Security-Szenarien einfach absichern.

Der WSO2 stellt für grössere Kunden auch kostenpflichtig Consulting- und Development-Unterstützung zur Verfügung und bietet somit auch für Grosskunden eine solide ESB-Lösung an. Diesen Bereich haben wir nicht bewertet, da wir im Rahmen dieser Arbeit keine Erfahrungen machen konnten. Wir können lediglich auf die Referenzen und auf die auf der Homepage aufgeführten Case Studies[26] verweisen.

4.3.2 Anwendungsfall für Einsatz des Produkts Oracle Service Bus

Da es sich beim OSB um einen mächtigen ESB mit vielen Features handelt, ist er besser geeignet für grosse Systemumgebungen. Also für den Fall, dass eine grosse Anzahl Verbindungen darüber realisiert werden soll. Vorteilhaft ist auch, wenn bereits Know-how über Oracle-Produkte besteht und solche im Einsatz sind. Ein guter Grund für den OSB können auch viele anzuschliessende Anwendungen sein, welche keine standardisierte Schnittstelle haben und viel Customizing in der Kommunikation erfordern.

Ein weiterer Grund dafür, dass der OSB vor allem für grössere Systemumgebungen geeignet ist, sind die hohen Kosten. Nicht nur der OSB selbst, sondern auch eine Oracle DB und ein WebLogic Server müssen installiert und lizenziert werden. Ausserdem kommen höhere Anforderungen an die Hardware dazu. Eine weitere Feststellung ist, dass man ohne Oracle Produkte-Know-how sehr schnell an seine Grenzen stösst – dies wurde uns bereits bei der Installation klar. Dafür gibt es bei der Installation die Option, eine Cluster-Umgebung aufzubauen und die Load-Balancing- und Failover-Problematik direkt zu lösen.

Ein technischer Vorteil liegt darin, dass auf dem WebLogic Server Client- oder Host-Applikationen, welche am ESB angeschlossen werden, direkt betrieben werden können. Der OSB bietet auch gute Unterstützung für Java-Applikationen und erlaubt mittels Java Callout sogar den Aufruf solcher

[26] Zum Beispiel die Case Study der SUVA - http://wso2.com/casestudies/suva-streamlines-insurance-processes-with-wso2-esb oder von eBay http://wso2.com/casestudies/ebay-uses-100-open-source-wso2-esb-to-process-more-than-1-billion-transactions-per-day (zuletzt aufgerufen am 12.12.2014).

Applikationen während der Verarbeitung in der Pipeline. Dies verschafft dem Entwickler eine grosse Anzahl von Möglichkeiten bei der Transformation und Translation mittels Java-Programmierung. Mit dem JDeveloper existiert ein zusätzliches Tool, welches das Customizing unterstützt. Dies beinhaltet auch grafische Unterstützung für das Erstellen und Transformieren von XML- und WSDL-Schemas und MFL-Files. Die Pipeline ermöglicht auf verständliche Art die Implementation der Routing Patterns mittels selbsterklärender grafischer Darstellung. Auch sicherheitstechnisch kann der OSB durch die Möglichkeit der Implementation auf verschiedenen Layers hohen Ansprüchen gerecht werden.

Wird ein ESB für eine grössere Systemumgebung gebraucht, ist die Wahl eines OSB berechtigt. Für kleine Unternehmen mit begrenztem Budget wäre er jedoch definitiv überdimensioniert.

4.4 ESB Readiness-Checkliste

In diesem Kapitel haben wir eine Readiness-Checkliste vorbereitet. Diese hilft einem Unternehmen beim Entscheid für oder gegen einen ESB. Die Checkliste kann zur Überprüfung der aktuellen Situation verwendet werden. Hierbei ist die Unterscheidung zwischen Müssen- und Können-Kriterien besonders interessant. Zusätzlich wird mit der Kategorie Nicht-Dürfen von nicht nötigen ESB Einsätzen gewarnt.

Abbildung 50 Kategorisierung der ESB Readiness-Checkliste in Müssen, Können und Nicht-Dürfen

Diese Kategorisierungen verhalten sich nicht exklusiv zueinander. Somit wäre es denkbar, dass ein Unternehmen gemäss Checkliste einen ESB integrieren und betreiben kann, jedoch aufgrund der niedrigen Komplexität nicht muss (Beibehaltung der tieferen Kosten).

In den nachfolgenden Kapiteln ist der Begriff „Aufwand“ als Kosten zu verstehen. Diese können sich aus der Zeit für die Einarbeitung, Integration und Konfiguration eines ESB, aber auch aus den Lizenzkosten zusammensetzen.

4.4.1 ESB-Bedarf (Müssen)

Diese Gruppierung beantwortet die Fragestellung:

„Ab wann ist der Aufwand für ein Unternehmen ohne ESB grösser als mit (inkl. Integration) ESB?“

Um diese Frage zu beantworten, müssen die Verantwortlichen der IT-Abteilung eines Unternehmens Informationen zu den folgenden Thematiken beschaffen:

- Wie gross ist die Enterprise-Komplexität?
  - Anzahl Services
  - Anzahl Komponenten
  - Anzahl verschiedener Protokolle
  - Anzahl externer Schnittstellen
  - Anzahl Aufrufe
- Existiert ein Bedarf an zusätzlicher Logik bei Service-Aufrufen?

    - Müssen Nachrichten transformiert werden?
    - Müssen Nachrichten übersetzt (Translate) werden?
    - Müssen semantische Unterschiede in Nachrichten normalisiert zu einem standardisierten Nachrichtenformat überführt werden?
- Bedarf an Auslagerung der Vermittlung des Nachrichtenflusses (z.B. durch die Benutzung des Mediator Pattern)

4.4.2 ESB-Kompetenzen (Können)

Diese Gruppierung beantwortet die Frage:

„Ab wann ist ein Unternehmen in der Lage, einen ESB zu integrieren und zu benutzen?“

Um diese Frage zu beantworten, müssen die Verantwortlichen der IT-Abteilung eines Unternehmens Informationen zu den folgenden Thematiken beschaffen:

- Ist die technische Umsetzung möglich?
    - Internes Know-how für Integration vorhanden?
    - Skills für eine qualitativ hochwertige Integration vorhanden?
- Ist die aktuelle IT-Landschaft geeignet für einen ESB-Einsatz?
    - Systeme und Prozesse müssen identifiziert werden
    - Integrationsprofile müssen erstellt werden
    - Gibt es nichtfunktionale Anforderungen, die eingehalten werden müssen (Performance, Security, andere QoS-Attribute)?
- Erlaubt die IT Governance eine Integration überhaupt?
    - Wer ist zuständig für den Betrieb und die Wartung?
    - Projektteam vs. Integrationsteam?

4.4.3 ESB-No-Gos (Nicht-Dürfen)

Diese Gruppierung beantwortet die Frage:

„In welcher Situation ist dem Unternehmen davon abzuraten, einen ESB für die Integration zu verwenden?“

Diese Frage konzentriert sich im Gegensatz zu den vorherigen Fragen auf Argumente gegen eine Integration eines ESB im Unternehmen. Vielfach wird ein ESB in nicht benötigten Situationen eingesetzt, was unnötig Kosten generiert.

- Kein Bedarf
    - Bestehen im Unternehmen überhaupt Anwendungsfälle, in denen es sinnvoll wäre, einen ESB zu brauchen?
    - Gibt es Fälle, in denen es ökonomischer ist, auf einen ESB zu verzichten?

4.4.4 Konsequenzen

Die Integration eines ESB im Unternehmen hat Konsequenzen. Die Benutzung eines ESB kann diese Konsequenzen hervorrufen, muss aber nicht. Zusätzlich zu den vom Industriepartner gewünschten Checklisten haben wir die positiven und negativen Konsequenzen nach Einsatz eines ESBs aufgelistet.

4.4.4.1.1 Positive Konsequenzen

- Architektur der Integration ist gut dokumentiert und visualisiert

- Neue Systeme können besser integriert werden
- Stärkere Entkopplung der Komponenten
- Zentrale Lösung innerhalb des Unternehmens, welche stark skalierbar ist

4.4.4.1.2 Negative Konsequenzen

- Beim Einsatz eines ESB, der eigentlich nicht benötigt wird, ist der Aufwand für Integration zwischen Applikationen grösser als ohne ESB
- Eigenes Team für Integration wird benötigt
- Eigenes Team für Betrieb (inkl. Wartung) des ESB wird benötigt
- ESB-Know-how (inkl. produktspezifisches Know-how) muss erhöht werden
- Hohe Ausgaben für den initialen Aufwand
- Komplexität wird erhöht, da nun eine Schicht mehr existiert

4.4.5 Checklisten

In diesem Kapitel haben wir Checklisten erarbeitet und stellen diese vor.

Die Checklisten sollen als Basis für die Entscheidung dienen und können beliebig auf den Kontext angepasst und erweitert werden. Die Einträge in den Checklisten werden mit den Kriterien aus dem Kriterienkatalog begründet und referenzieren jeweils auf die entsprechenden Kriterien. Sie können zudem auch auf Empfehlungen aus der Fachliteratur [28] [29] basieren.

Wie die Antworten zu den Checklisten interpretiert werden können, ist in der folgenden Matrix beschrieben:

	Wahr	Falsch
ESB-Bedarf (Müssen)	Ein mögliches Anzeichen, dass der Aufwand mit ESB **kleiner** ist als ohne ESB.	Ein mögliches Anzeichen, dass der Aufwand mit ESB nicht **kleiner** (aber nicht unbedingt grösser) ist als ohne ESB.
ESB-Kompetenzen (Können)	Die Anforderung für die Integration eines ESB ist **erfüllt**.	Die Anforderung für die Integration eines ESB ist **nicht erfüllt**.
ESB-No-Gos (Nicht-Dürfen)	Die Integration eines ESB ist in diesem Fall **nicht empfehlenswert**.	Die Integration eines ESB **kann** in diesem Fall **sinnvoll sein**, muss es aber nicht.

Tabelle 19 Matrix mit Interpretationshilfen zu den ESB Readiness-Checklisten

Wenn ein Element der Checkliste mit „*nicht anwendbar*“ beantwortet wird, soll dazu eine klare Begründung dokumentiert werden.

Jede Checkliste kann idealerweise ausgedruckt und ausgefüllt werden.

4.4.5.1 ESB-Bedarf (Müssen)

ID	Typ	Beschreibung / Element	Wahr	Falsch	Nicht anwendbar
1	Enterprise-Komplexität	Es existieren mindestens drei Komponenten (zum Beispiel Applikationen), welche miteinander kommunizieren.			
2	Enterprise-Komplexität	Es existiert mehr als ein Webservice pro zu integrierender Komponente.			
3	Enterprise-Komplexität	Es werden mehr als zwei Protokolle für den Nachrichtenaustausch benötigt.			
4	Enterprise-Komplexität	Es existieren Services, welche oft (mehrmals täglich) aufgerufen werden.			
5	Heterogenität	Beim Konsum eines Services muss zusätzlicher Aufwand betrieben werden, um den Inhalt des Aufrufs in das richtige Format zu übersetzen. *(*FK4 – Message Translation*)*			
6	Heterogenität	Beim Konsum eines Services muss zusätzlicher Aufwand betrieben werden, um den Inhalt der Antwort in das richtige Format zu übersetzen. *(*FK4 – Message Translation*)*			
7	Heterogenität	Beim Konsum eines Services muss zusätzlicher Aufwand betrieben werden, um den Inhalt des Aufrufs mit Daten (z.B. aus einer zentralen Datenquelle) zu bereichern. (*DK2 – Message Content Transformation Patterns*)			
8	Heterogenität	Beim Konsum eines Services muss zusätzlicher Aufwand betrieben werden, um den Inhalt des Aufrufs zu filtern. (*DK2 – Message Content Transformation Patterns*)			
9	Heterogenität	Es existiert mindestens ein Service, der in zwei semantisch verschiedenen Aufrufen (verschiedene Daten) aufgerufen wird. Der Service sollte diese Aufrufe äquivalent behandeln. *(Enterprise Integration Patterns: Normalizer Pattern)*			
10	Vermittlung des Nachrichtenflusses	Kommunikation soll in einer zentralen Instanz gemanagt werden. Kommunikationspartner sollen sich nicht direkt kennen.			

Tabelle 20 Übersicht aller Fragen, die gestellt werden müssen in der Kategorie ESB-Bedarf (Müssen)

4.4.5.2 ESB-Kompetenzen (Können)

ID	Typ	Beschreibung / Element	Wahr	Falsch	Nicht anwendbar
1	Machbarkeit (technisch)	Innerhalb des Unternehmens hat mindestens ein Mitarbeiter bereits fortgeschrittene[27] Erfahrungen mit der ESB-Integration gemacht.			
2	Machbarkeit (technisch)	Der Begriff ESB kann in eigenen Worten erklärt werden, sodass ein Laie die Grundsätze eines ESB versteht.			
3	Machbarkeit (aktuelle Architektur)	Es existiert eine Liste mit allen Komponenten, die mit dem ESB integriert werden sollen.			
4	Machbarkeit (aktuelle Architektur)	Prozesse im Unternehmen sind klar dokumentiert. Eine Integration in die ESB-Umgebung ist somit verständlich.			
5	Machbarkeit (aktuelle Architektur)	Es existiert eine Liste aller nichtfunktionalen Anforderungen (z.B. Security, Performance) für jede Komponente, welche integriert wird.			
6	IT Governance	Es existiert (oder wird erstellt) ein für den ESB verantwortliches Team im Unternehmen.			
7	IT Governance	Es existiert eine Policy, dass für Änderungen am ESB (z.B. neue Komponenten hinzufügen) das Integrationsteam zuständig ist.			
8	IT Governance	Es existiert eine Policy, dass für Änderungen am ESB (z.B. neue Komponenten hinzufügen) das Projektteam zuständig ist.			

Tabelle 21 Übersicht aller Fragen, die gestellt werden müssen in der Kategorie ESB-Kompetenzen (Können)

[27] Als "fortgeschrittene Erfahrungen" zählt mindestens ein Integrationsprojekt (mind. drei Komponenten und drei verschiedene Protokolle)

4.4.5.3 ESB-No-Gos (Nicht-Dürfen)

ID	Typ	Beschreibung / Element	Wahr	Falsch	Nicht anwendbar
1	Kein Bedarf	Es sind zum aktuellen Zeitpunkt nur zwei Komponenten bekannt, die miteinander kommunizieren sollen. Dies wird sich in naher Zukunft nicht ändern. (Ein ESB ist hier überdimensioniert. Alternative wäre eine Point-to-Point-Kommunikation.)			
2	Kein Bedarf	Es ist zum aktuellen Zeitpunkt nicht bekannt, ob in der Zukunft neue Komponenten integriert werden sollen. (Design auf Vorrat)			
3	Kein Bedarf	Kommunikation basiert nur auf einem oder zwei Protokollen. In der Zukunft könnten jedoch neue Protokolle hinzukommen. Dies ist aber zum aktuellen Zeitpunkt nicht bekannt. (Design auf Vorrat)			
4	Kein Bedarf	Es ist zum aktuellen Zeitpunkt nicht nötig, Nachrichten zu übersetzen oder anzupassen (Daten filtern oder anreichern). In der Zukunft könnten jedoch die Funktionen gebraucht werden. Dies ist aber zum aktuellen Zeitpunkt nicht bekannt. (Design auf Vorrat)			

Tabelle 22 Übersicht aller Fragen, die gestellt werden müssen in der Kategorie "Nicht-Dürfen"

5 ITCSimulator-Applikation

Da ITC eine sehr komplexe Applikation darstellt, haben wir mit dem Kunden und dem Betreuer vereinbart, dass die ITC-Applikation analysiert und in einer einfacheren Version simuliert wird. Die Simulations-Applikation deckt funktional nur den Use Case ab.

5.1.1 Anforderungen

Für den ITCSimulator haben wir funktionale und nichtfunktionale Anforderungen definiert. Diese richten sich nach der original Applikation ITC, welche simuliert wird.

5.1.1.1 Funktionale Anforderungen

In der Tabelle unten sind die wichtigsten Funktionalen Anforderungen, welche der ITCSimulator erfüllen muss.

Nr.	Prio **[1 = required]** **[2 = nice to have]** **[3 = optional]**	Beschreibung
FA1	1	ITCSimulator soll via ESB Vertragsdaten erhalten können (gemäss Use Case-Beschreibung)
FA2	1	ITCSimulator soll eine Ansicht anbieten, in welcher die erhaltenen Daten als Liste angezeigt werden können
FA3	1	ITCSimulator soll die erhaltenen Daten in die eigene Datenbank einpflegen können

Tabelle 23 Funktionale Anforderungen an ITCSimulator

5.1.1.2 Nichtfunktionale Anforderungen

Der ITCSimulator muss sich gleich verhalten wie die ITC-Applikation. Dies gilt vor allem für Datenstrukturen (zum Beispiel Formatierung von Ein- und Austrittsdatum). Ausserdem wird ein ähnliches Setup gewählt (Application Framework, Application Server und Presentation Framework). Ein identisches ist nicht notwendig.

5.1.1.3 Datenstrukturen

ITC bewirtschaftet eine Oracle Datenbank, in welcher die Vertragsdaten enthalten sind. Die Tabellen in der Datenbank sehen folgendermassen aus:

5.1.1.3.1 Tabelle für Mitarbeiter

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EMPLOYEE_ID	NUMBER	No	(null)	1	(null)
2	CTL_TCN	NUMBER(6,0)	No	0	2	(null)
3	LOGIN_NAME	VARCHAR2(15 BYTE)	No	(null)	3	(null)
4	FIRST_NAME	VARCHAR2(20 BYTE)	No	(null)	4	(null)
5	LAST_NAME	VARCHAR2(20 BYTE)	No	(null)	5	(null)
6	CTL_ACT	NUMBER(1,0)	Yes	1	6	(null)
7	CTL_CRE_UID	VARCHAR2(30 BYTE)	No	(null)	7	(null)
8	CTL_CRE_TS	TIMESTAMP(6)	No	(null)	8	(null)
9	CTL_MOD_UID	VARCHAR2(30 BYTE)	No	(null)	9	(null)
10	CTL_MOD_TS	TIMESTAMP(6)	No	(null)	10	(null)
11	POOL_MANAGER	NUMBER	Yes	(null)	11	(null)
12	SORTING_PREFERENCE	NUMBER(1,0)	No	0	12	(null)

Tabelle 24 Tabellenschema (Tabelle mit Mitarbeitern) der ITC-Applikation

5.1.1.3.2 Formatierung Datum

Formatierung des Datums kann auf DD.MM.YY vereinfacht werden.

TIME_ENTRY_ID	CTL_TCN	STARTPOINT
9980	0	26.09.02
9981	0	26.09.02
9982	0	27.09.02
9983	0	27.09.02
9984	0	27.09.02

Abbildung 51 Beispieldaten (Formatierung des Datums) aus ITC Datenbank

5.1.1.4 Application Framework und Application Server

Framework / Server	**Beschreibung**
Application Framework	ITC nutzt das AdNovum-eigene Application Framework MOAP3 (Mother Of All Projects, Version 3) und basiert auf Java EE.
Application Server	GlassFish 3.1.0
Presentation Framework	Apache Struts 1.3

Tabelle 25 Von ITC verwendete Frameworks

5.1.2 Architektur ITCSimulator

Der ITCSimulator ist analog ITC auf einer Two-Tier- und Three-Layer-Architektur aufgebaut. Die Architektur ist eine stark vereinfachte Version der ITC-Architektur.

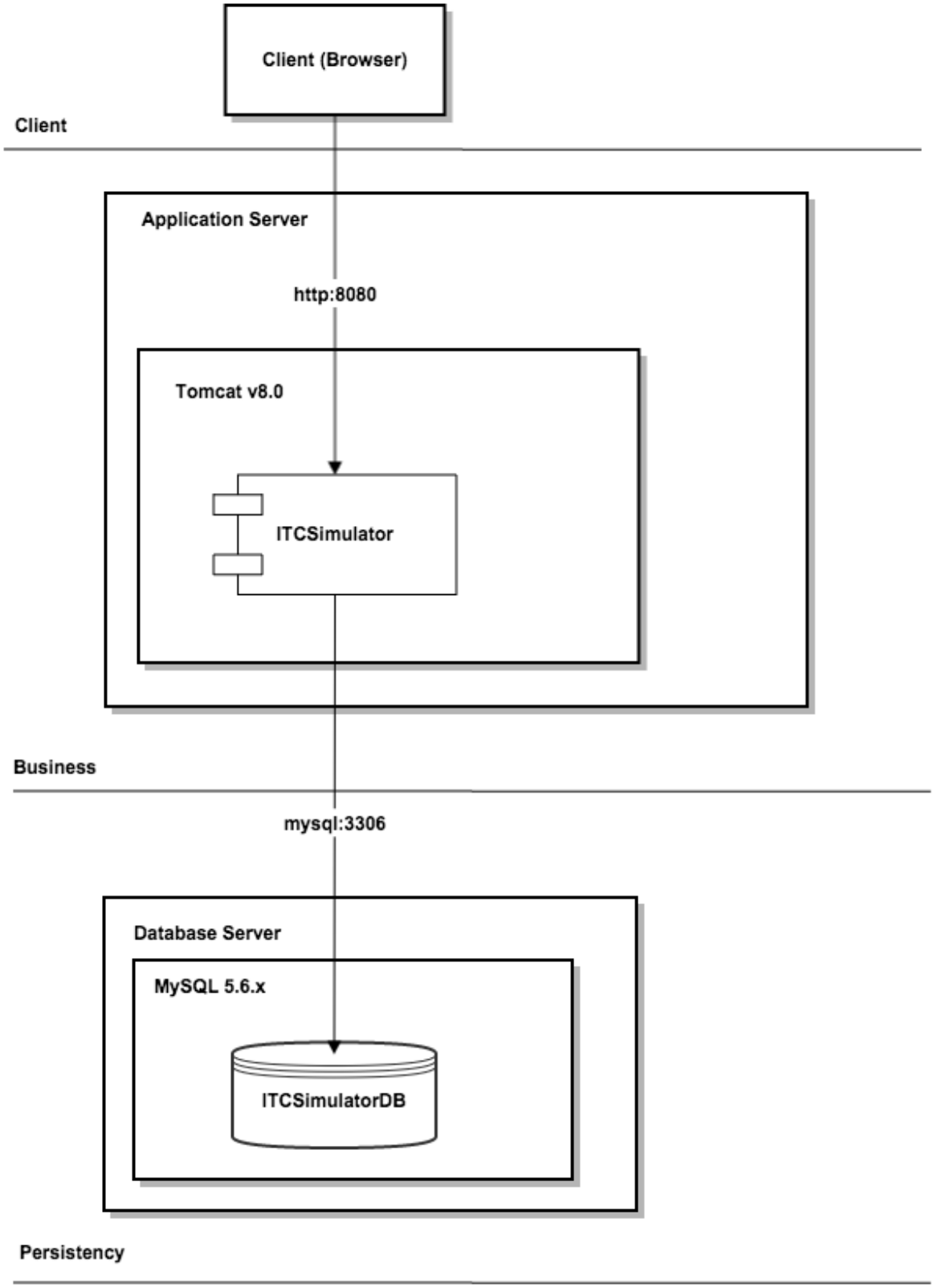


Abbildung 52 ITCSimulator-Architektur

5.1.2.1 Client Layer

Als User Interface Framework haben wir uns für Java Server Faces (Version 2.2) entschieden. Zudem wurde das Bootstrap Framework und JQuery (inkl. DataTables Plug-In) gewählt, um so ein schönes Frontend erstellen zu können. Vom Client Layer greifen ManagedBeans (JSF 2.2 Komponenten) auf die Business-Logik im Business Layer zu.

5.1.2.2 Business Layer

Im Business Layer kommunizieren Plain Old Java Objects (POJOs) mit dem Client. Zusätzlich wurde Spring für Dependency Injection gebraucht. Ausserdem wird für die HTTP-Anfragen, welche zu den ESB Backends führen, die HTTP-Client-Bibliothek von Apache verwendet.

5.1.2.3 Persistence Layer

Als Datenbank-Management-System wurde MySQL (Version 5.6.x) gewählt. Die Datenbank besteht aus einer Tabelle Employee und ist analog der Employee-Tabelle von ITC aufgebaut. Es wurden nur die Felder Login, Firstname und Lastname übernommen sowie zwei zusätzliche Felder definiert für die Vertragsdaten (ContractStart, ContractEnd).

```
mysql> show columns from employee;
+---------------+-------------+------+-----+-------------------+----------------+
| Field         | Type        | Null | Key | Default           | Extra          |
+---------------+-------------+------+-----+-------------------+----------------+
| id            | bigint(20)  | NO   | PRI | NULL              | auto_increment |
| login         | varchar(255)| YES  |     | NULL              |                |
| firstname     | varchar(255)| YES  |     | NULL              |                |
| lastname      | varchar(255)| YES  |     | NULL              |                |
| contractStart | datetime    | YES  |     | NULL              |                |
| contractEnd   | datetime    | YES  |     | NULL              |                |
| lastupdate    | datetime    | YES  |     | CURRENT_TIMESTAMP |                |
+---------------+-------------+------+-----+-------------------+----------------+
7 rows in set (0.00 sec)
```

Abbildung 53 Tabellenbeschreibung Employee der ITCSimulator-Datenbank

Als Implementation der Java Persistence API (JPA) wurde Hibernate (Version 4.3.6) verwendet.

6 Verzeichnisse

6.1 Tabellenverzeichnis

6.2 Abbildungsverzeichnis

6.3 Codeverzeichnis

6.4 Literaturverzeichnis

[1] M. Demolsky, «JAXenter,» 08 04 2008. [Online]. Available: http://jaxenter.de/artikel/Enterprise-Service-Bus. [Zugriff am 15 12 2014].

[2] Jürgen Kress, Berthold Maier, Hajo Normann, Danilo Schmeidel, Guido Schmutz, Bernd Trops, Clemens Utschig-Utschig, Torsten Winterberg, «Oracle,» July 2013. [Online]. Available: http://www.oracle.com/technetwork/articles/soa/ind-soa-esb-1967705.html. [Zugriff am 26 09 2014].

[3] D. A. Chappell, Enterprise Service Bus, 2004.

[4] Free Software Foundation, «GNU General Public License,» 29 Juni 2007. [Online]. Available: http://www.gnu.org/copyleft/gpl.html.

[5] Massachusetts Institute of Technology, «The MIT License (MIT),» [Online]. Available: http://opensource.org/licenses/MIT.

[6] The Apache Software Foundation, «Apache License, Version 2.0,» The Apache Software Foundation, Januar 2004. [Online]. Available: http://www.apache.org/licenses/LICENSE-2.0.html. [Zugriff am 15 12 2014].

[7] M. Fowler, «Microservices,» 25 März 2014. [Online]. Available: http://martinfowler.com/articles/microservices.html. [Zugriff am 15 12 2014].

[8] National Institute of Standards and Technology, Standards for Security Categorization of Federal Informaton and Information Systems, 2004.

[9] C. Fehling, F. Leymann, R. Retter und P. Arbitter, Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications, Springer Science & Business Media, 2014.

[10] G. Hophe und B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison Wesley, 2003.

[11] WSO2, «Architecture - Enterprise Service Bus 4.8.0 - WSO2 Documentation,» WSO2, 17 November 2014. [Online]. Available: https://docs.wso2.com/display/ESB480/Architecture. [Zugriff am 15 12 2014].

[12] WSO2, «WSO2 Enterprise Service Bus,» 2014. [Online]. Available: http://wso2.com/products/enterprise-service-bus/. [Zugriff am 15 12 2014].

[13] WSO2, «Service Level Security Implementation,» WSO2, 13 10 2014. [Online]. Available: https://docs.wso2.com/display/AS521/Service-Level+Security+Implementation. [Zugriff am 16 12 2014].

[14] WSO2, «Working with Message Builders and Formatters - Enterprise Service Bus 4.8.1,» 10 Juli 2014. [Online]. Available: https://docs.wso2.com/display/ESB481/Working+with+Message+Builders+and+Formatters.

[15] WSO2, «Enterprise Integration Patterns with WSO2 ESB,» 16 Mai 2014. [Online]. Available: https://docs.wso2.com/display/IntegrationPatterns/Enterprise+Integration+Patterns+with+WSO2+ESB. [Zugriff am 15 12 2014].

[16] WSO2, «SNMP Monitoring - Enterprise Service Bus 4.8.1 - WSO2 Documentation,» WSO2, 29 Januar 2014. [Online]. Available: https://docs.wso2.com/display/ESB481/SNMP+Monitoring. [Zugriff am 15 12 2014].

[17] WSO2, «Overview - Clustering Guide 4.2.0 - WSO2 Documentation,» 16 Dezember 2014. [Online]. Available: https://docs.wso2.com/display/CLUSTER420/Overview#Overview-clusterexampleClusteringexample . [Zugriff am 17 12 2014].

[18] WSO2, «WSO2 Support,» WSO2, [Online]. Available: http://wso2.com/support/#quickstart. [Zugriff am 14 10 2014].

[19 WSO2, «Installing ESB on Linux and Solaris from Binary Distribution - Enterprise Service Bus 4.0.3

] - WSO2 Documentation,» WSO2, 12 Januar 2012. [Online]. Available: https://docs.wso2.com/display/ESB403/Installing+ESB+on+Linux+and+Solaris+from+Binary+Distribution. [Zugriff am 15 12 2014].

[20] WSO2, «Starting ESB Management Console on Linux and Solaris - Enterprise Service Bus 4.0.3 - WSO2 Documentation,» WSO2, 23 März 2012. [Online]. Available: https://docs.wso2.com/display/ESB403/Starting+ESB+Management+Console+on+Linux+and+Solaris. [Zugriff am 15 12 2014].

[21] WSO2, «ESB Installation Prerequisites - Enterprise Service Bus 4.0.3 - WSO2 Documentation,» WSO2, 20 Mai 2014. [Online]. Available: https://docs.wso2.com/display/ESB403/ESB+Installation+Prerequisites. [Zugriff am 15 12 2014].

[22] Oracle Corporation, Oracle Enterprise Manager Lifecycle Management Administrator's Guide, 12c Release 4 (12.1.0.4), 2014.

[23] Oracle Corporation, Oracle Fusion Middleware Installing and Configuring Oracle Service Bus, 12c (12.1.3), 2014.

[24] Oracle Corporation, Oracle® Fusion Middleware - Developing Services with Oracle Service Bus 12c (12.1.3), 2014.

[25] Oracle Corporation, Oracle Fusion Middleware Tuning Performance Guide 12c (12.1.3), 2014.

[26] A. Pareek, «Routing Message Interaction Patterns - Reference,» März 2011. [Online]. Available: http://beatechnologies.files.wordpress.com/2011/03/message-interaction-patterns.pdf.

[27] R. Van Luttikhuizen und E. Elzinga, «Jumpstart for Oracle Service Bus Development,» Oktober 2009. [Online]. Available: http://www.oracle.com/technetwork/articles/jumpstart-for-osb-development-page--097357.html. [Zugriff am 17 12 2014].

[28] R. Butenuth, «Was ist ein ESB und wofür kann man ihn nutzen?,» 30 November 2012. [Online]. Available: https://blog.codecentric.de/2012/11/was-und-wofur-esb/. [Zugriff am 15 10 2014].

[29] R. Mason, «To ESB or not to ESB,» 6 Juli 2009. [Online]. Available: http://blogs.mulesoft.org/to-esb-or-not-to-esb/. [Zugriff am 15 10 2014].

[30] T. Arnuphaptrairong, «Top Ten Lists of Software Project Risks,» 16-18 März 2011. [Online]. Available: http://www.uio.no/studier/emner/matnat/ifi/INF5181/h14/pensumliste/microsoft-word---iaeng-top-ten-lists-of-software-project-risk1---imecs2011_pp732-737.pdf. [Zugriff am 22 09 2014].

[31] Apache Software Foundation, «Apache HTTP Server Project,» 21 Juli 2014. [Online]. Available: http://httpd.apache.org/. [Zugriff am 16 12 2014].

[32] Stoneburner, Gary, «Underlying Technical Models for Information Technology Security,» 12 2001. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf. [Zugriff am 10 12 2014].

6.5 Glossar

Abkürzung / Begriff	Beschreibung
AES	Advanced Encryption Standard
CLI	Command Line Interface
EAI	Enterprise Application Integration
ECTS	European Credit Transfer System
EIP	Enterprise Integration Patterns von Hohpe und Woolf
EJB	Enterprise JavaBeans
ESB	Enterprise Service Bus
FA	Funktionale Anforderung
FIFO	First-In-First-Out
FTP	File Transport Protocol
FTPS	File Transfer Protocol over SSL
HTTP(S)	Hypertext Transfer Protocol (Secure)
IMAP	Internet Message Access Protocol
JCA	J2EE ConnectorArchitecture
JMS	Java Message Services
JPA	Java Persistence API
MFL	Message Format Language
MOM	Message Oriented Middleware
NFA	Nichtfunktionale Anforderung
NIST	National Institute of Standards and Technology
OSB	Oracle Service Bus
POP	Post Office Protocol
QoS	Quality of Service
RMI	Remote Method Invocation
Round Robin	Sequenzielle Ausführung nach dem FIFO-Prinzip
SA	Studienarbeit
SFTP	SSH File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SSH	Secure Shell
SSL	Secure Sockets Layer
UTF-8	Unicode Transformation Format-8
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformation

Tabelle 26 Glossar mit Abkürzungen und Beschreibungen