Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D – 70569 Stuttgart

Fachstudie Nr. 57

Vergleich von kommerziellen Implementierungen eines Enterprise Service Bus

Jan Trautvetter, Necati Aydin, Felix Billau

Studiengang: Softwaretechnik

Prüfer: Prof. Dr. Frank Leymann

Betreuer: Tobias Unger

begonnen am: 01.02.2006

beendet am: 08.05.2006

CR-Klassifikation: H.4.m

Fachstudie „Vergleich von kommerziellen Implementierungen eines Enterprise Service Bus“

Version:	2.0
Autoren:	Jan Trautvetter, Necati Aydin, Felix Billau
erstellt am:	31. Januar 2006
letzte Änderung:	8. Februar 2006

Universität Stuttgart
Fakultät Informatik, Elektrotechnik und Informationstechnik – IAAS

Inhaltsverzeichnis

1

Allgemein

1.1 Vision des Enterprise Service Bus

Klärung des Begriffs

Weil der Begriff des Enterprise Service Bus (ESB) unterschiedlich definiert wird und die gestellten Anforderungen an ihn oft nicht übereinstimmen, erfolgt in diesem Abschnitt eine kurze Einführung in das Konzept des ESB inklusive der Aufstellung der Fähigkeiten, die ein ESB-Produkt aus unserer Sicht kennzeichnen.

Der ESB kann zunächst einmal als **Konzept** und damit als zu implementierende Lösung verstanden werden, die es, als Infrastruktur für die Informationsweitergabe eines Unternehmens, aufzubauen gilt.
Der ESB kann aber auch als **Produkt** verstanden werden, da inzwischen zahlreiche Softwarehersteller über eine Lösung in ihrem Produktportfolio verfügen, die diese Funktionalität verspricht.
Für diese Arbeit ist diese Unterscheidung nicht von Interesse, da alle Produkte gegen Kriterien evaluiert werden, die wir als elemtare Fähigkeiten des ESB identifiziert haben.

Wir betrachten deshalb den ESB, wie es auch in der Literatur üblich ist, unabhängig von der Art seiner Realisierung, als Softwareinfrastruktur und Middlewareschicht mit folgenden Eigenschaften und Fähigkeiten:

- Ein ESB soll die service-orientierte Architektur, also die Integration heterogener Anwendungen in ein Dienstmodell, ermöglichen.
- Ein ESB soll mit allen angeschlossenen Anwendungen kommunizieren können, unabhängig davon, welche Formate und Protokolle sie unterstützen.
- Ein ESB soll auf Verteilung und Skalierung ausgelegt sein.
- Die Administration eines ESB soll zentral erfolgen.

Zur Verdeutlichung der wichtigsten Bestandteile (und deren Zusammenhang) eines ESB siehe Abbildung 1.1. Tiefergehende Informationen über das Konzept und die Implementierung eines ESB sind in [CHA] zu finden.

In der angesprochenen Abbildung ist deutlich zu erkennen, dass sich ein ESB aus beliebig vielen Knoten zusammensetzen kann, die jeweils als externe Schnittstelle fungieren. Diese Knoten können beispielsweise Anwendungsserver oder Message Orientaded Middleware Systeme sein und es ist macht für die angeschlossenen Dienste ausserhalb des ESB keinen Unterschied welcher der Knoten angesprochen wird.
Ein ebenfalls wichtiges Kennzeichen des ESB ist seine nach aussen hin homogene Administrationsschnittstelle, die eine zentrale Konfiguration des gesamten Systems und damit aller angeschlossenen Knoten ermöglicht. Dies zeigt die Abbildung in Form der alleinstehenden Konfigurationskomponente.

Inwieweit es mit einem Produkt nun möglich ist die skizzierte Infrastruktur aufzubauen, wird Ergebnis dieser Fachstudie sein. Dabei stehen die Ergebnisse aber natürlich in direktem Zusammenhang mit den Kriterien und können deshalb auch nur als Evalierung auf diese Kriterien hin verstanden werden. Mit dieser Arbeit wird also keine allgemeingültige Aussage darüber getroffen, welches Produkt der Beste ESB ist.

Produkteigenschaften

Da der ESB als Infrastruktur verwendet werden kann auf der eine SOA aufbaut gibt es zahlreiche Produkte dieser Kategorie, die den Untenehmen eine Lösung ihrer Integrationsproblematik versprechen.

Die Produkte unterscheiden sich vor allem durch ihre Entwicklungsgeschichte.
So gibt es auf der einen Seite Firmen, die sich speziell auf den ESB spezialisiert haben und ihre Middelware-Komponenten, sowie die Konfigurationsmöglichkeiten, speziell auf seine Anforderungen abgestimmt haben, und damit im Verhältnis eher leichtgewichtige Container als Knoten des ESB anbieten. Mit Container ist hiebei eine Sammlung von Anwendungen und Funktionen gemeint, die in einer Komponente gesammelt vorliegen und von aussen her abrufbar sind. Sonic, Iona und andere Hersteller haben diesen Weg verfolgt. Auf der anderen Seite gibt es aber auch die Spezialisten im Bereich der Integration, die den ESB als Aufbau auf ihr bisheriges Lösungsportfolio implementiert haben. Hierzu wären beispielsweise IBM oder BEA zu nennen. Diese Lösungen basieren auf einem Anwendungungsserver, der um die ESB-Funktionalität erweitert wurde und verfügen somit über eher schwergewichtige Container, die dafür aber vielfältige Funktionen unterstützen.

Aus diesen grundlegenden Ansätzen zum Aufbau eines ESB-Produktes, das natürlich noch auf die konkreten Bedürfnisse eines Unternehmens zugeschnitten werden muss, ergeben sich bereits charackteristische Stärken und Schwächen. So hat das Produkt von IBM zwar einen großen Funktionsumfang, ist daher aber auch eher als schwergewichtig anzusehen. Die spezialisierten Produkte hingegen unterstützen mitunter weniger Standards, bilden dafür aber die zentralen Anforderungen an den ESB effizient ab.

Zum Test der Produkte

Um die Produkte detailiert betrachten zu können, erfolgt zuerst eine Produktvorauswahl in Kapitel 2. Anschließend erfolgt die eingehende Analyse dieser Produkte, die sich in mehrere Teile gliedert. In Kapitel 3 wird das Loan-Broker-Szenario vorgestellt, das zur Evaluierung der praktischen Fähigkeiten der Produkte herangezogen wird. In Kapitel 4 wird die Kriterienliste vorgestellt, nach der der Leistungsumfang der Produkte bewertet wird. In Kapitel 5 schließlich geschieht die Evaluierung selber und in Kapitel 6 steht dann die abschließende Betrachtung.

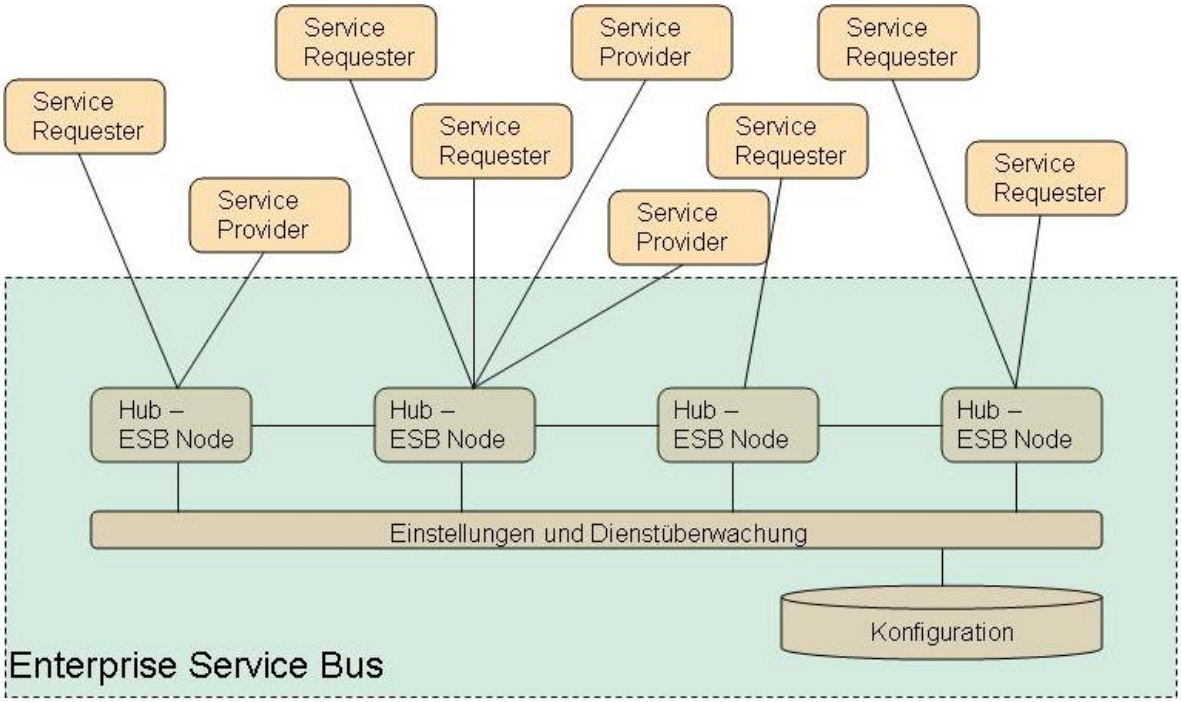

Abbildung 1.1: Realisierung eines ESB

2

Produktauswahl

2.1 Kriterienliste zur Vorauswahl

Die Evaluation wird maßgeblich durch die Auswahl der ESB-Produkte bestimmt. Es kamen daher für die Analyse nur Produkte in Frage, die verfügbar, dokumentiert und weit verbreitet waren und sind. Deswegen ergeben sich folgende Kriterien zur Produktauswahl:

- Verfügbarkeit der Software - Software-Paket und ein Evaluierungsschlüssel waren verfügbar.
- Verfügbarkeit und Güte der Dokumentation - Sowohl eine Beschreibung als auch Tutorials waren verfügbar.
- Relevanz des Produktes - Das Produkt hatte nach unserem Eindruck und in Absprache mit dem Betreuer einen guten Marktauftritt und stellte eine ernstzunehmende Größe im ESB-Umfeld dar.

2.2 Produkte in der Vorauswahl

Nach erfolgter Recherche im Web wurden folgende elf Produkte identifziert, die ESB-Funktionalität versprechen.

1. „ESB for Websphere“ der Firma Cape Clear
2. „ESB 2006“ der Firma Fiorano
3. „Artix“ der Firma Iona
4. „AquaLogic Service Bus“ der Firma BEA
5. „SONIC ESB“ der Firma Sonic
6. „EntireX“ der Firm Software AG
7. „WebSphere ESB“ der Firma IBM
8. „ServiceMix“, ein Open-Source-Projekt
9. „Business Integration Engine“ der Firma Brunswick
10. „Integration Suite“ der Firma Polarlake
11. „Business Works“ der Firma Tibco

Nach Kontaktaufnahme mit den Herstellern und vertiefter Ansicht ihres Lösungsportfolios ergab sich folgende Situation:

Produkt	Verfügbarkeit	Dokumentation	Relevanz
Cape clear (ESB for Websphere)	Software ohne Evaluationsschlüssel (15 Tage)	Doku und Tutorials (sehr umfangreich)	EAI-Spezialist/ Nischenabieter
Fiorano (ESB 2006)	Evaluationsschüssel ohne Software	Doku und Tutorials (sehr umfangreich)	EAI-Spezialist/ Nischenabieter
Iona (Artix)	Software mit Evaluationsschlüssel 30 Tage)	Doku und Tutorials (sehr umfangreich)	EAI-Spezialist/ Nischenabieter
BEA (AquaLogic Service Bus)	Software mit Evaluationsschlüssel (30 Tage)	Doku und Tutorials (sehr umfangreich)	EAI-Spezialist/ Starke Stellung im Markt/ nutzt den eigenen Anwendungsserver
Sonic (ESB)	Software mit Evaluationsschlüssel (30 Tage)	Doku und Tutorials (umfangreich)	EAI-Spezialist/ Das ESB-Produkt ist bereits weit verbreitet und etabliert
Software AG (EntireX)	Mail-Kontakt mit Herr B. wegen eines Vertrages mit IAAS	Doku und Tutorials	Unternehmen mit umfangreichem Portfolio/ ESB ist ein Bundle aus bestehenden Lösungen
IBM (WebSphere ESB)	Durch IAAS lizensiert	Doku und Tutorials (sehr umfangreich)	Unternehmen mit sehr umfangreichem Portfolio/ ESB ist ein Bundle aus bestehenden Lösungen
Open Source (ServiceMix)	Open Source	Doku (vorhanden)	Das einzige Open Source Projekt
Brunswick (Business Int. Engine)	Open Source	für 195 Euro verfügbar	EAI-Spezialist/ Nischenabieter
Polarlake (Integration Suite)	nicht verfügbar	nicht verfügbar	EAI-Spezialist/ Nischenabieter
Tibco (Business Works)	nicht verfügbar	nicht verfügbar	EAI-Spezialist/ Nischenabieter

2.3 Ergebnis der Vorauswahl

Folgende Produkte wurden nicht näher betrachtet, weil für die Evaluation in unserer Fachstudie Software, Dokumentation und Tutorials nicht vollständig und kostenlos auf Anhieb verfügbar waren.

- Cape clear (ESB for Websphere) - Kontaktaufnahme war nicht erfolgreich.
- Fiorano (ESB 2006) - Kontaktaufnahme war nicht erfolgreich.
- Software AG (EntireX) - Hier wäre ein Vertragsabschluss für die Evaluation erforderlich.
- Brunswick (Business Integration Engine) - Dokumentation und die Tutorials sind kostenpflichtig.
- Polarlake (Integration Suite) - Kontaktaufnahme war nicht erfolgreich.
- Tibco (Business Works) - Kontaktaufnahme war nicht erfolgreich.

Ebenfalls nicht näher betrachtet wurde das folgende Produkt:

- Open Source (ServiceMix) - Auch wenn dieses Produkt über einige interessante Eigenschaften (Implementierung des JBI-Standards JSR 208 und Nutzung des Spring-Frameworks) verfügt, wurde es nicht näher betrachtet.
  Dies liegt zum einen daran, dass das Produkt noch keine Marktreife erreicht hat, weil neben den passenden Entwicklungswerkzeugen auch die Tutorials fehlen. Zum anderen ist die Fachstudie aber auch auf den Vergleich von kommerziellen Produkten im ESB Bereich ausgelegt.

Damit blieben vier Anbieter übrig, deren Produkte für die Analyse zur Verfügung standen.
Der „Sonic ESB“ von Sonic wurde gesetzt, da diese Lösung erstmalig im März 2002 eingeführt wurde und somit über die längste Historie der Verbesserungen und Erweiterungen verfügt.
Der „WebSphere ESB“ von IBM wurde ebenfalls gesetzt, da IBM als der größte Anbieter für den Aufbau von Softwareinfrastrukturen über eine herausragende Marktstellung verfügt.
Die Entscheidung, ob im folgenden „Artix“ von Iona oder der „AquaLogic Service Bus“ von BEA evaluiert wird, fiel zugunsten von BEA aus, da BEA als Integrationsspezialist weithin bekannt ist und die Dokumentation sowie die Tutorials zu dem neuen ESB-Produkt sehr aussagekräftig sind.
Übrig blieben somit folgende Produkte, die im Rahmen dieser Fachstudie untersucht wurden:

- IBM (WebSphere ESB)
- Sonic (ESB)
- BEA (AquaLogic Service Bus)

3

Das Loan-Broker-Szenario

Das Loan-Broker-Szenario ist ein bekanntes Szenario zur Verdeutlichung der unternehmenstypischen Situation von verteilt arbeitenden Diensten, die in ein übergeordnetes Dienstmodell aggregiert werden sollen.
Eine Einführung in den Ablauf und die daran beteiligten Dienste geschieht in Kapitel 3.1, eine detailliertere Beschreibung der Schnittstellen erfolgt in Kapitel 3.2.

3.1 Einführung

In diesem Szenario spricht ein Kunde über einen Client den Dienst eines Brokers an, um für seinen Kreditwunsch das beste Angebot zu erhalten.
Der Broker ist der Dienst, dessen Funktionalität in diesem Szenario näher betrachtet wird. Seine Aufgabe gliedert sich in mehrere Stufen.

1. Entgegennahme der Anfrage des Clients.
2. Kommunikation mit einer Rating-Agentur und Feststellen der Kreditwürdigkeit.
3. Kommunikation mit allen Banken, deren Leistungsportfolio dem Kreditwunsch entspricht.
4. Auswahl des besten Angebots.
5. Senden einer Antwort an den Client, in der das beste Angebot enthalten ist.

Zur Verdeutlichung der einzelnen Schritte siehe hierzu auch Abbildung 3.1 aus [HOH]

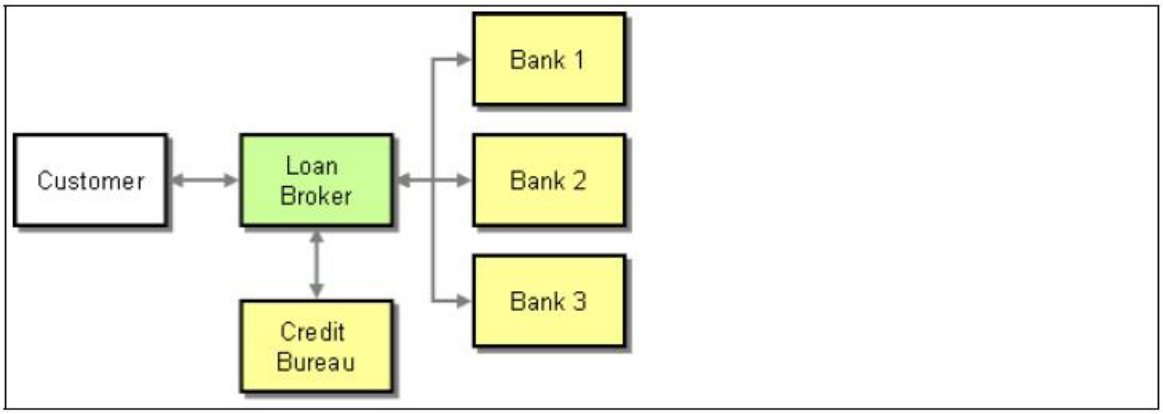

Abbildung 3.1: Das Loan-Broker-Szenario

3.2 Detailierte Beschreibung des Loan-Brokers

Im Folgenden werden die bereits aufgezählten Schritte (die der Loan-Broker durchführt, um dem Kreditsuchenden das bestmögliche Angebot präsentieren zu können) nocheinmal detailiert beschrieben.

1. Entgegenahme der Anfrage des Clients - Der Broker erhält eine Nachricht, in der die **SSN**(Sozialversicherungsnummer), der **LoanAmount**(Kredithöhe) und der **LoanTerm**(Kreditlaufzeit) des Kreditsuchenden enthalten sind.

2. Kommunikation mit einer Rating-Agentur und Feststellen der Kreditwürdigkeit - Die **SSN** der Anfrage sendet der Broker nun an die Rating Agentur und erhält von dieser eine Antwort, die die Kreditwürdigkeit des Anfragers widerspiegelt. Die Antwort der Rating Agentur enthält dabei die **SSN**, den **CreditScore** (Der Status, den der Kreditsuchende hat) und die **HistoryLength** (Die Länge der Kredithistorie des Anfragenden).

3. Kommunikation mit allen Banken, deren Leistungsportfolio dem Kreditwunsch entspricht - Hier werden die Daten zum gewünschten Kredit und die Daten der Rating Agentur zusammengestellt und darauf hin untersucht, ob sie den Bedingungen der einzelnen Banken genügen.
   Die *Bedingung für Bank1* ist, dass der CreditScore mindestens 500 ist und die History Length mindestens 5 Jahre beträgt.
   Die *Bedingung für Bank2* ist, dass der CreditScore mindestens 700 ist und die History Length mindestens 10 Jahre beträgt.
   Bank3 stellt keinerlei Anforderungen.
   Jede Bank, die angesprochen wird, erhält vom Broker als Daten die **SSN**, den **LoanAmount**, den **LoanTerm**, den **CreditScore** und die **HistoryLength**.
   Nach der Verarbeitung durch die Bank erhält der Broker eine Antwort mit der **SSN**, dem **LoanAmount**, der **InterestRate** und der **QuoteID** zurück.

4. Auswahl des besten Angebotes - Der Broker wählt die Bank mit der besten **InterestRate** aus.

5. Senden einer Antwort an den Client in der das beste Angebot enthalten ist. - Der Broker leitet das ausgewählte Angebot an den Anfragenden zurück. Es enthält die **SSN**, den **LoanAmount**, die **InterestRate** und die **QuoteID**.

Die Anfragen an die Banken sollen dabei parallel ausgeführt werden.

3.3 Überblick über die Architektur des Loan-Brokers

Um die Produkte anhand des Loan-Broker-Szenarios evaluieren zu können, wurde eigens ein Webservice-Server in C# geschrieben, der die Funktion der externen Dienste (Kreditbüro

und die drei Banken) übernahm. Diese Dienste waren als Webservices mit SOAP/HTTP-Bindung erreichbar. Im Anhang sind exemplarisch Anfragen an den Loan-Broker-Dienst selbst, aber auch an Kreditbüro und eine der drei Banken samt den Antworten aufgeführt. Der Loan-Broker-Dienst selbst wurde mit den zu evaluierenden Produkten implementiert. Die nachfolgende Abbildung fasst die Architektur des Loan-Broker-Szenarios nochmals kurz zusammen:

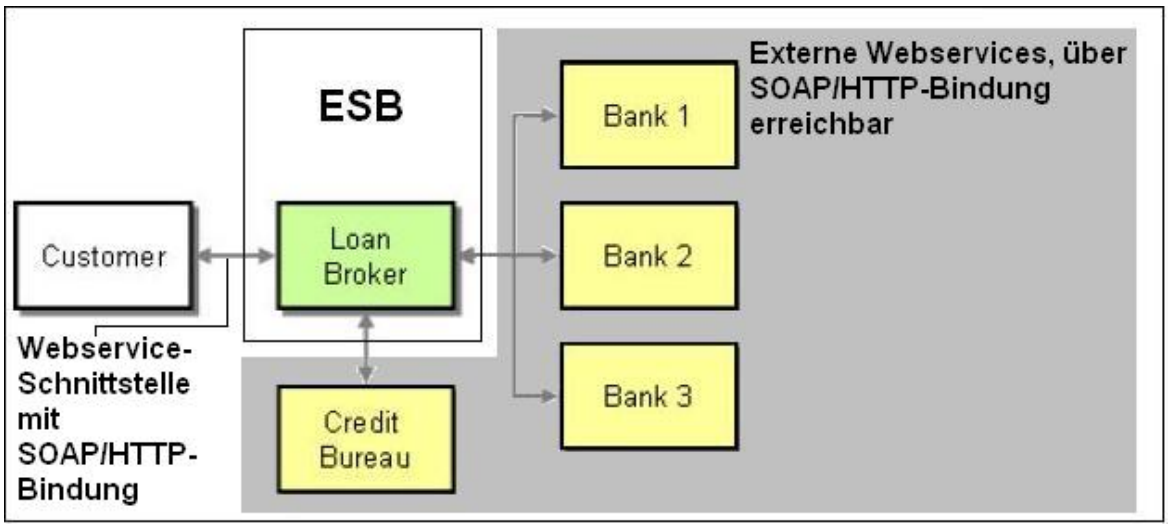

Abbildung 3.2: Die Architektur des Loan-Brokers

4

Bewertungsgrundlage

In diesem Kapitel werden die Kriterien definiert, auf die die Produkte untersucht wurden. Die Kriterienliste spiegelt dabei detaillierter die vier folgenden Bereiche wieder:

- Erfüllbarkeit des Loan-Broker-Szenarios - Kann das Loan-Broker-Szenario mit dem Produkt modelliert und gelöst werden?
- Nachrichtenverarbeitung - Ist der Bus in der Lage, die Nachrichten, durch Routing und Veränderung, in der gewünschten Form zu verarbeiten?
- Features - Werden Transaktionen, Standards, Sicherheitskonzepte, Skalierbarkeit etc. unterstützt?
- Benutzerfreundlichkeit der Entwicklungsumgebung - Lässt sich das Produkt gut bedienen?

Die Performance der Systeme ist von uns nicht objektiv feststellbar. Wir hätten uns auf die Angaben des Herstellers verlassen müssen und verzichten im Rahmen dieser Fachstudie daher für die Evaluierung auf dieses Kriterium.

4.1 Kriterienliste

Erfüllbarkeit des Loan-Broker-Szenarios

Dies ist das gewichtigste Kriterium. Es wird überprüft, ob alle geforderten Eigenschaften des Loan-Brokers mit den (ESB-)Mitteln des untersuchten Produktes umgesetzt und nachgebildet werden können.

Nachrichtenverarbeitung

Mit diesem Oberkriterium wird die Fähigkeit des Produktes festgehalten, inwieweit es die Kernfunktionalität eines ESB, nämlich die Möglichkeiten zur Nachrichtenverarbeitung und Nachrichtenweiterleitung, beherrscht. Nähere Informationen dazu sind in den beiden Unterpunkten Routing und Transformationen enthalten.

- Routing - Es ist möglich, die Nachrichten auf dem Bus zu verschiedenen Zeitpunkten auszuwerten und anhand der festgestellten Werte die nächsten Verarbeitungsschritte auszuwählen.
  Diese Funktionalität kann beispielsweise dazu benutzt werden, die an den Bus angeschlossenen Dienste nur dann zu benachrichtigen, wenn die Nachricht bestimmte Kriterien erfüllt, oder die Nachrichtentransformation situationsbedingt zu steuern.
  Eine gute Unterstützung dieser Funktionalität zeigt sich einerseits in der Möglichkeit, an verschiedenen Punkten Verzweigungen, beziehungsweise Routing-Schritte einfügen zu können, und andererseits in der Möglichkeit, alle Informationen der Nachrichten extrahieren und auswerten zu können.

- Transformationen - Es ist möglich die Nachrichten auf dem Bus zu verschiedenen Zeitpunkten in ihrer Struktur oder ihrem Inhalt zu verändern. Das heißt, dass das Einfügen, das Löschen und das Ändern von Elementen der Nachricht, beispielsweise bevor Routingschritte durchgeführt werden, jederzeit möglich ist.

Transaktionsschutz

Es können Transaktionskontexte für Bestandteile eines Prozesses, für ganze Prozesse oder auch applikationsübergreifend definiert werden.

Sicherheit

Für einen ESB ist neben der Zugriffseinschränkung für die Ressourcen vor allem ein sicherer Nachrichtentransport von Interesse, so dass sich folgende drei zentrale Sicherungskriterien ergeben.

- Authentifizierung - Der Benutzer und seine Gruppe können festgestellt werden.

- Authorisierung - Die Ressourcen des ESB können für Benutzer und Gruppen freigegeben, beziehungsweise gesperrt werden.

- Sicherer Nachrichtentransport - Sowohl die Kommunikation von Benutzern mit der Anwendung, als auch die Kommunikation der Anwendung mit anderen Anwendungen lässt sich durch Verschlüsselungsmechanismen absichern.

Neben der Frage, ob sich diese Sicherungen einrichten lassen, interessieren hier auch die Möglichkeiten dazu. Beispielsweise sollte eine gute Authentifizierungsstrategie die Einbindung bestehender Kataloge (Active Directory, LDAP, etc.) ermöglichen, während eine gute Nachrichtenverschlüsselungsstrategie sowohl symmetrische, als auch asymmetrische Verfahren unterstützen sollte. Auch ist interessant, ob sich Sicherheitskontexte über Anwendungsgrenzen hinweg definieren lassen.

Integrationsspektrum

Da in den Nachrichtenfluss eines ESB bestehende Dienste eingebunden werden, ist der Umfang der in das Produkt integrierten Adaptoren ein wichtiges Kriterium.
Für weit verbreitete Software wie beispielsweise J2EE, Corba, .NET, MQ Series sollten deshalb Adaptoren existieren, um diese Anwendungen ohne Probleme einbinden zu können.

Unterstützung von Standards

Im Zusammenhang mit Web-Services haben sich einige Protokoll-Standards etabliert. Diese sogenannten WS*-Standards sollte auch ein ESB weitgehend unterstüzen, damit die Interoperabilität mit den Diensten und Anwendungen der Umgebung gegeben ist.

Skalierbarkeit

Für den Fall, dass die Leistung eines Server nicht ausreicht, um die Anforderungen an ihn abzudecken, können bei Bedarf Resourcen hinzugefügt, bzw. auch wieder entfernt werden. Replikationsmechanismen, damit Entwicklungen und Veränderungen propagiert werden können, sind dabei ebenso integriert wie eine zentrale Verwaltungsschnittstelle für den gesamten Bus.

Registrierung

Es ist möglich, eine Registrierung der Dienste vorzunehmen und diese Registrierung programmatisch anzusprechen, um entdeckungsbasiert die Dienste aufrufen zu können. Dies erlaubt die Migration von Diensten ohne eine Änderung der betroffenen Nachrichtenflüsse.

Benutzerfreundlichkeit der Entwicklungsumgebungs

Die Bedienung der Anwendung ist durchdacht und einfach. Genauer wird dies beispielsweise durch die sieben Kriterien der Ergonomie-Norm DIN EN ISO 9241, Teil 10 „Grundsätze der Dialoggestaltung“ definiert. Nähere Informationen hierzu befinden sich beispielsweise in [HOFF].
Die sieben Kriterien sind:

- Aufgabenangemessenheit - Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgbe effektiv und effizient zu erledigen.
- Selbstbeschreibungsfähigkeit - Ein Dialog ist selbstbeschreibungsfähig, wenn jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer auf Anfrage erklärt wird.
- Steuerbarkeit - Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist.
- Erwartungskonformität - Ein Dialog ist erwartungskonform, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, z.B. seinen Kenntnissen aus dem Arbeitsgebiet, seiner Ausbildung und seiner Erfahrung sowie den allgemein anerkannten Konventionen.
- Fehlertoleranz - Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann.
- Individualisierbarkeit - Ein Dialog ist individualisierbar, wenn das Dialogsystem Anpassungen an die Erfordernisse der Arbeitsaufgabe sowie an die individuellen Fähigkeiten und Vorlieben des Benutzers zuläßt.
- Lernförderlichkeit - Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen des Dialogsystems unterstützt und anleitet.

4.2 Anmerkung zur Durchführung

Die Auswertung der in 4.1 aufgeführten Kriterien für ein konkretes Produkt geschieht zu einem Teil über die Erfahrungen, die bei der Durchführung des Szenarios gemacht worden sind. Dies betrifft speziell die Kriterien zur Bedienbarkeit und zur Nachrichtenverarbeitung.
Der andere Teil, also die Auswertung der Features, ergibt sich aus den Produktspezifikationen, beziehungsweise der Produktbeschreibung durch den Hersteller. An dieser Stelle vertrauen wir auf die Richtigkeit der gemachten Angaben, da sie zusammengenommen für einen Test innerhalb der Szenarien zu umfangreich sind.

5

Evaluierung

In diesem Kapitel werden die Produkte BEA „AquaLogic Service Bus“, IBM „Websphere ESB“ (mit der IDE „WebSphere Integration Developer“) und „SONIC ESB“ von Sonic auf ihre Leistungsfähigkeit geprüft, indem das zuvor beschriebene Loan-Broker-Szenario modelliert wird und die Kriterien überprüft werden.
Die einzelnen Unterkapitel sind dabei jeweils wie folgt aufgebaut:

- Einführende Produktbeschreibung
- Durchführung des Szenarios
- Auswertung der Kriterien
- Abschließende Betrachtung

5.1 BEA AquaLogic Service Bus (Ver. 2.1)

5.1.1 Einführende Produktbeschreibung

Der AquaLogic Service Bus ist Teil einer umfangreichen Suite von Produkten, mit denen die verschiedenen Aspekte der Integrationsproblematik abgedeckt werden. Außer dem Service Bus enthält sie noch folgende weitere Lösungen, die zur Ergänzung und Erweiterung ebenfalls eingesetzt werden können:

- BEA AquaLogic Service Registry - Komponente, um Dienste zu katalogisieren und entdeckungsbasierte Aufrufe zu ermöglichen.
- BEA AquaLogic User Interaction - Komponente, um unter Einsatz der Portaltechnologie die Erstellung und das Management von Benutzerschnittstellen zu vereinfachen.
- BEA AquaLogic Data Services Platform Product Center (früher BEA Liquid Data) - Komponente, die die Datenintegration heterogener Quellen ermöglicht. Durch eine virtuelle Datenintegration und den Aufbau einer Datenabstraktionsschicht werden dabei die Daten gekapselt.
- BEA AquaLogic Enterprise Security Product Center (früher WebLogic Enterprise Security) - Komponente, die die Implementierung und die Einbindung von (domainübergreifenden) Sicherheitsmodellen unterstützt.

Diese Produkte bauen jeweils auf dem WebLogic Server auf (ebenfalls ein BEA Produkt) und bieten zusammengenommen alle Bestandteile für den Aufbau einer komplexen Integrationslösung, wobei es aber auch möglich ist, jeweils nur ein Produkt, oder aber eine Kombination der Produkte, einzusetzen.
Für den zu untersuchenden Service Bus bedeutet dies, dass einige Funktionalität, wie beispielsweise die Transaktionsunterstützung, nicht direkt in das Produkt integriert ist, aber durch den Einsatz eines anderen Produktes der Suite erreicht werden kann. Um bei dem Beispiel der Transaktionssicherheit zu bleiben, wäre es hier möglich, durch den Einsatz der Data Service Plattform die gewünschte Funktionalität zu erreichen.
Die Implementierung des in Kapitel 3 beschriebenen Loan-Broker-Szenarios ist dadurch nicht gefährdet und gelingt ohne Probleme (siehe hierzu Kapitel 5.1.2), da das Szenario die Kernfunktionalität eines ESBs überprüft und diese vollständig in das Produkt integriert worden ist.
Bei der Auswertung der Kriterien für das Produkt in Kapitel 5.1.3 zeigt sich allerdings, dass Funktionalität fehlt, die in andere Produkte dieser Kategorie integriert ist. Bewertet wurde deshalb, um die Vergleichbarkeit zu gewährleisten, lediglich die Funktionalität des Service Busses, wobei aber auf die ergänzenden Produkte der Suite verwiesen wird.
Weitere Informationen zu BEA und den bereitgestellten Produkten sind unter [BEA1] zu finden, während die Dokumentation des AquaLogic Service Bus unter [BEA2] zu finden ist.

5.1.2 Durchführung des Szenarios

Sowohl der BEA AquaLogic Service Bus an sich, als auch die Entwicklungsumgebung für die Definition seiner Dienste bauen auf einem Anwendungsserver auf, so dass die Modellierung der Nachrichtenflüsse mit Hilfe eines Browsers als Benutzerschnittstelle geschieht. Dies wirkt zunächst ungewöhnlich, weil keine komplexe Entwicklungsumgebung mit zahlreichen Menüpunkten zur Verfügung steht, sondern eine HTML-Seitenbasierte Webanwendung, bereitet aber im Folgenden keine Probleme. Im Gegenteil zeigt sich während der Arbeit mit dem System, dass die damit einhergehende Beschränkung auf die wesentlichen Bedienelemente gut gelungen ist.

Beginn - Anlegen einer Domäne

Um ein Projekt anzulegen, muss zunächst innerhalb des Anwendungsservers eine Domain angelegt werden, die die Funktion eines Containers übernimmt, auf dem alle globalen Einstellungen vorgenommen werden können. Für das Anlegen der Domain steht ein Wizard zur Verfügung, der in Abbildung 5.1 zu sehen ist.

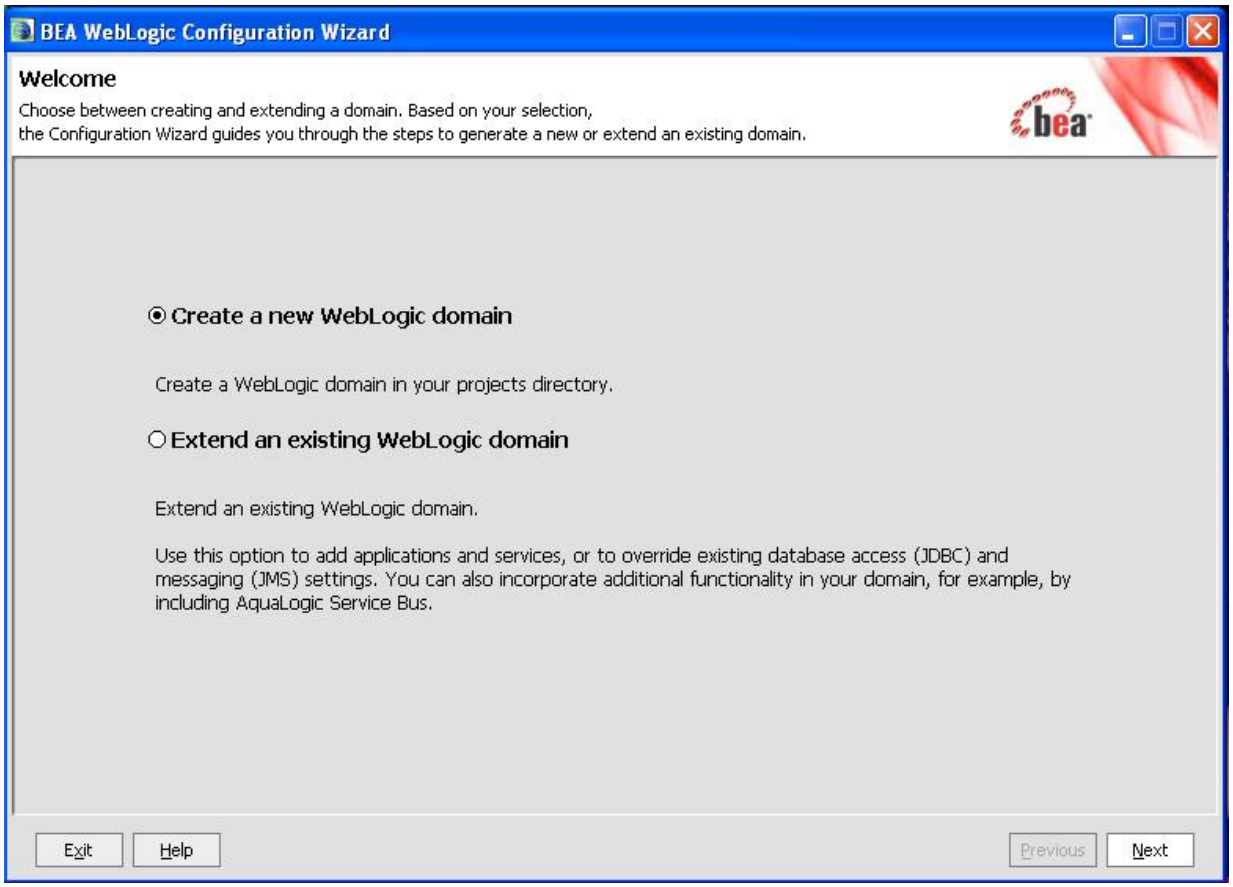

Abbildung 5.1: Anlegen einer Domain

Anlegen eines Projektes

Nachdem der Anwendungscontainer definiert worden ist, wird das Projekt selbst angelegt und die Ordnerstruktur festgelegt. Dies geschieht weitgehend intuitiv. Einen ersten Eindruck vermittelt Abbildung 5.2.

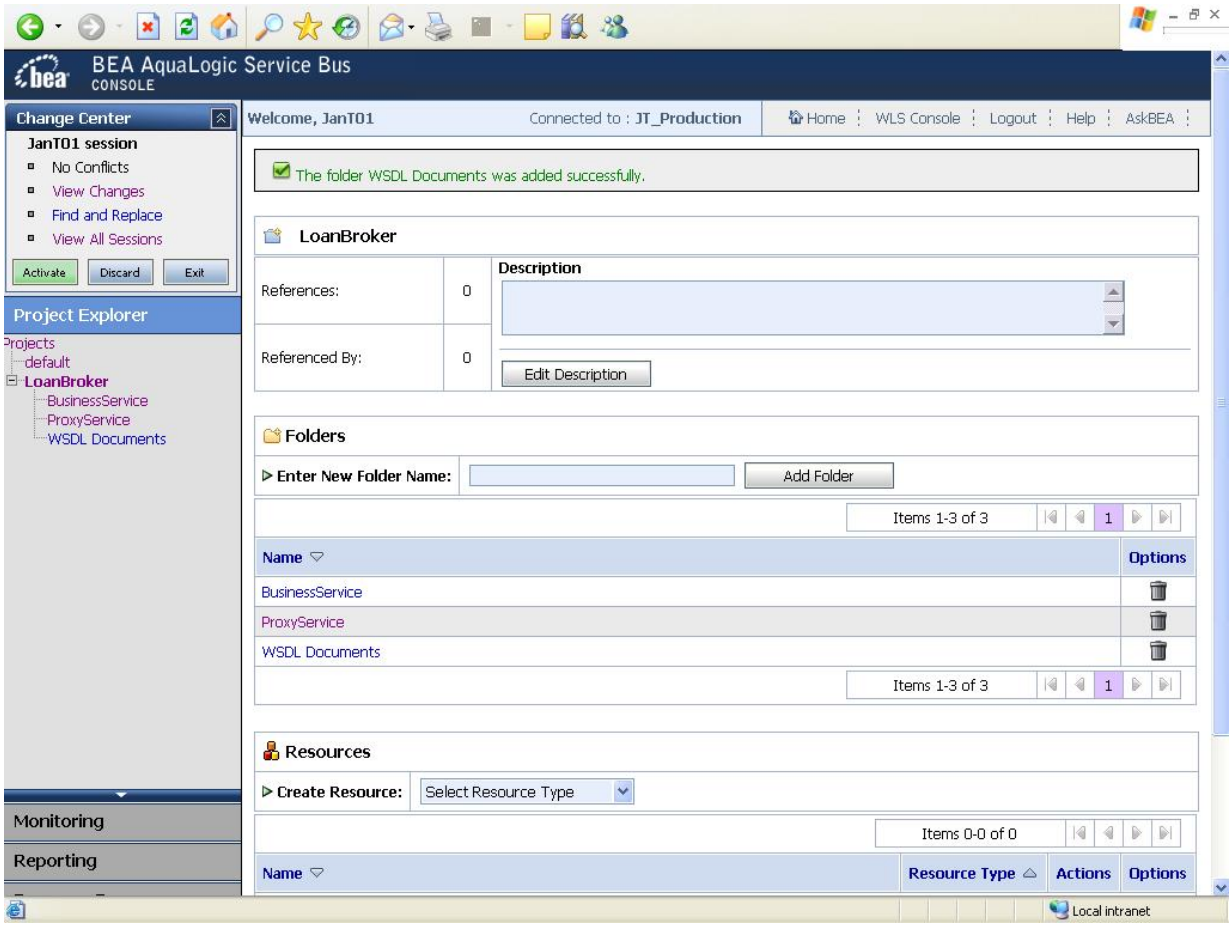

Abbildung 5.2: Anlegen der Ordnerstruktur

Anlegen der Dienste

Jetzt, da die Ordnerstruktur des Projektes aufgesetzt wurde, werden die Dienste aufgebaut. Dies geschieht in zwei Schritten.
Zuerst werden die Schnittstellendefinitionen der Ressourcen dem Projekt hinzugefügt. Dies kann mit typisierten Variablen durch die Einbindung einer WSDL-Datei oder einer MFL-Datei (Message Format Language - proprietäres Format zur Nachrichtendefinition) geschehen oder untypisiert durch die Auswahl von Text oder Binary.
In dieser Szenariodurchführung geschieht die Definition der Dienste mit Hilfe von WSDL-

Dateien. Mit dem Hinzufügen dieser Dateien sind auch bereits alle extern benötigten Ressourcen für das Aufsetzen des Nachrichtenflusses vorhanden.
Den nächsten Schritt kennzeichnet die Ableitung von Diensten aus den WSDL-Dateien, die später in den Ablauf eingebunden werden können.
Dazu werden, aufbauend auf den WSDL-Dokumenten, wiederum Ressourcen angelegt. Diese Ressourcen sind vom Typ BusinessService (alle eingebundenen Dienste), beziehungsweise ProxyService (externe Schnittstellen des Integrationsszenarios). Hierbei gilt es besonders zu beachten, dass die eingebundenen Operationen in der WSDL-Datei vom Typ „RPC-Style“ sind, da dieser das Verwenden der Variablen im Nachrichtenfluss stark vereinfacht, indem diese in den nächsten Schritten direkt abgefragt werden können. Aus demselben Grund empfiehlt es sich auch die einzelnen Variablen in der Deklaration der Nachrichten explizit zu definieren.
Zum Hinzufügen von Ressourcen siehe auch Abbildung 5.3.

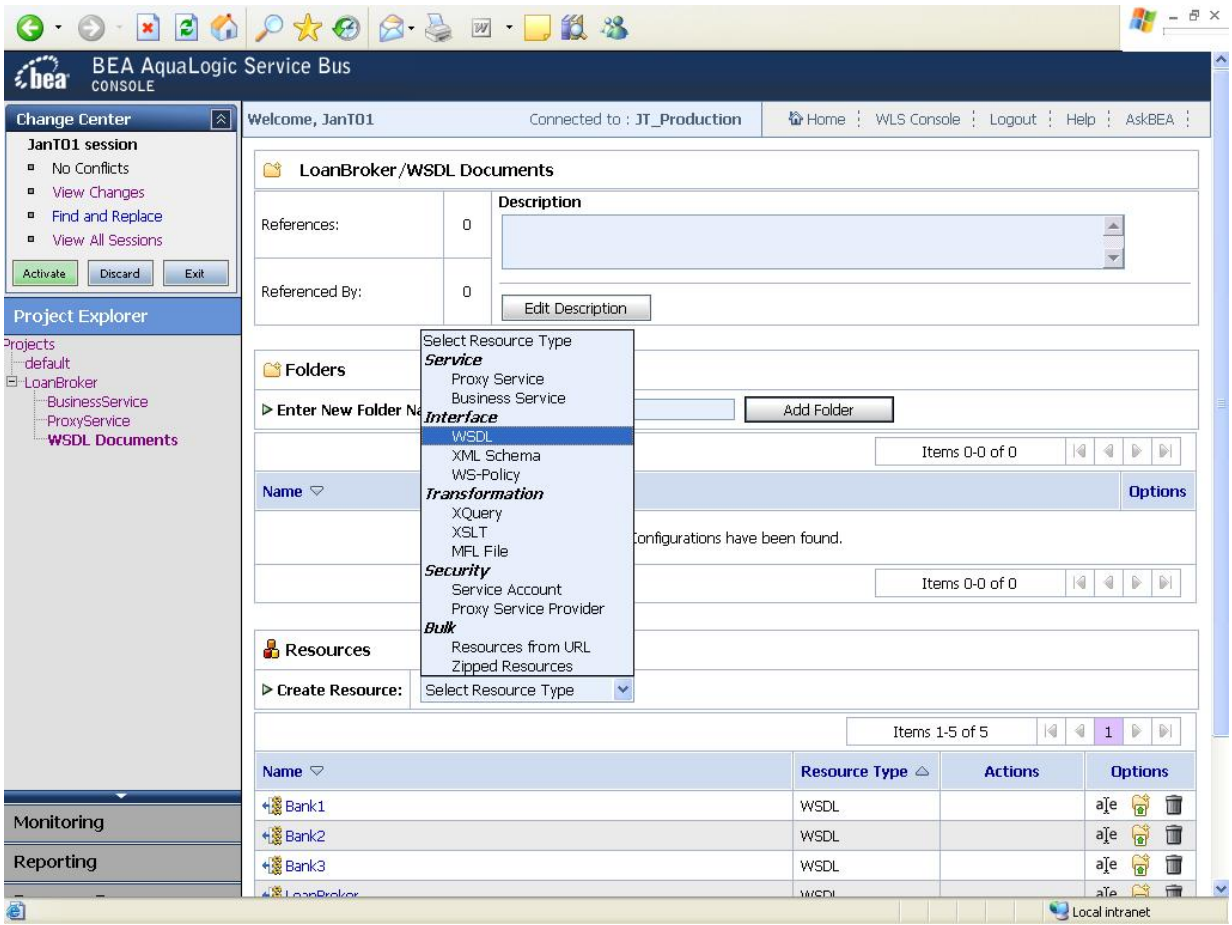

Abbildung 5.3: Anlegen von Ressourcen

Definition des Nachrichtenflusses

Zu diesem Zeitpunkt sind alle Dienste verfügbar. Sie müssen nun orchestriert werden, weswegen nun die Modellierung des Nachrichtenflusses erfolgt.
Der Zeitpunkt der Einbindung der Dienste, notwendige Bedingungen für einen Aufruf, Nachrichtenänderungen sowie einige andere Elemente, die diesen Ablauf kennzeichnen, werden dabei definiert. Die Definition erfolgt wiederum in zwei Schritten.

Der erste Schritt beinhaltet die Definition von grobgranularen Verarbeitungsblöcken, den sogenannten Stages. Sie strukturieren den Nachrichtenfluss und dienen letztendlich der Sammlung von elementaren Aktionen. Siehe hierzu auch Abbildung 5.4.

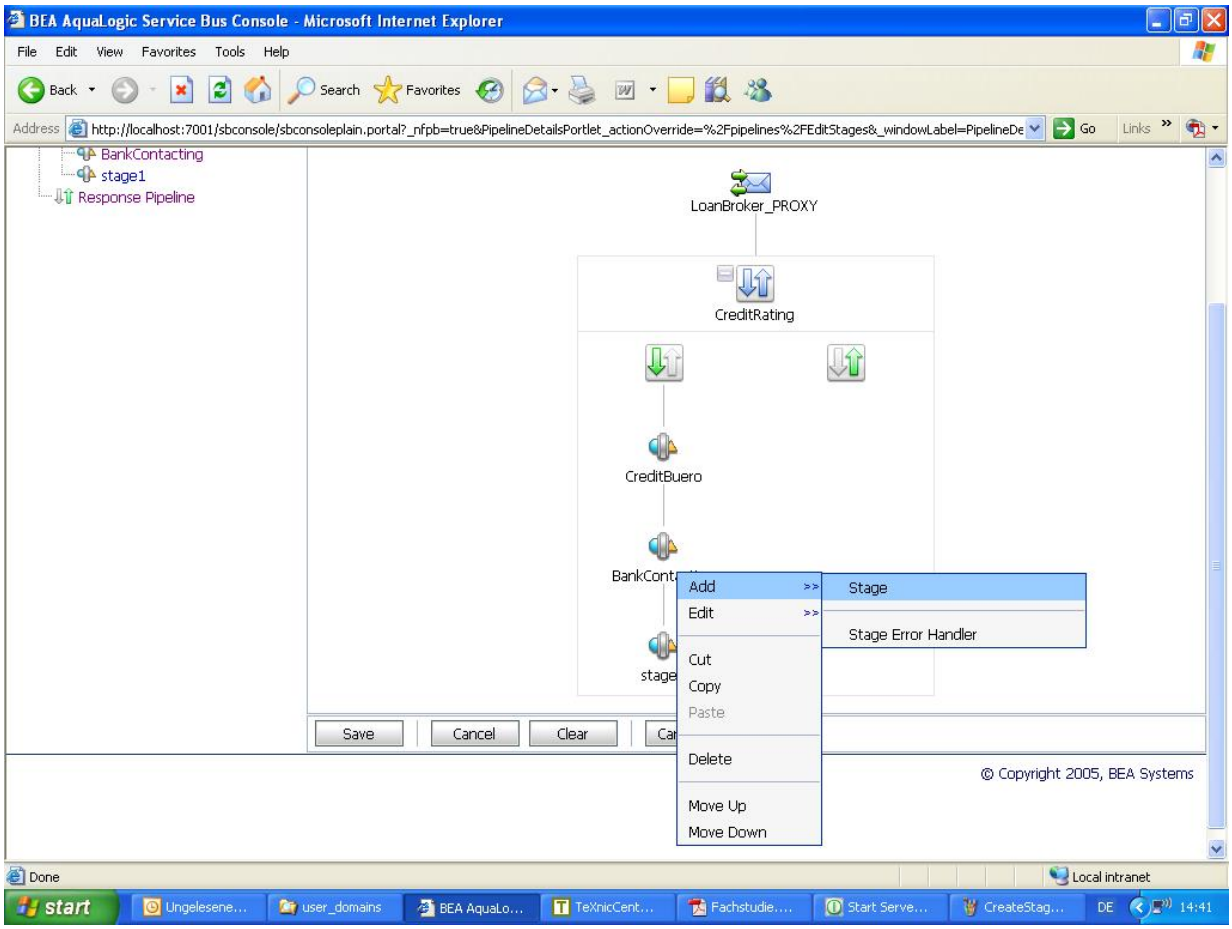

Abbildung 5.4: Anlegen von Stages

Im zweiten Schritt werden die eigentlichen Aktionen definiert. Das Spektrum der Aktionen ist dabei sehr weitreichend und deckt den Umgang mit Nachrichten vollständig ab. Beispielsweise können hier andere Dienste aufgerufen werden und Nachrichten in jeder Form manipuliert und angepasst werden. Siehe hierzu auch Abbildung 5.5.

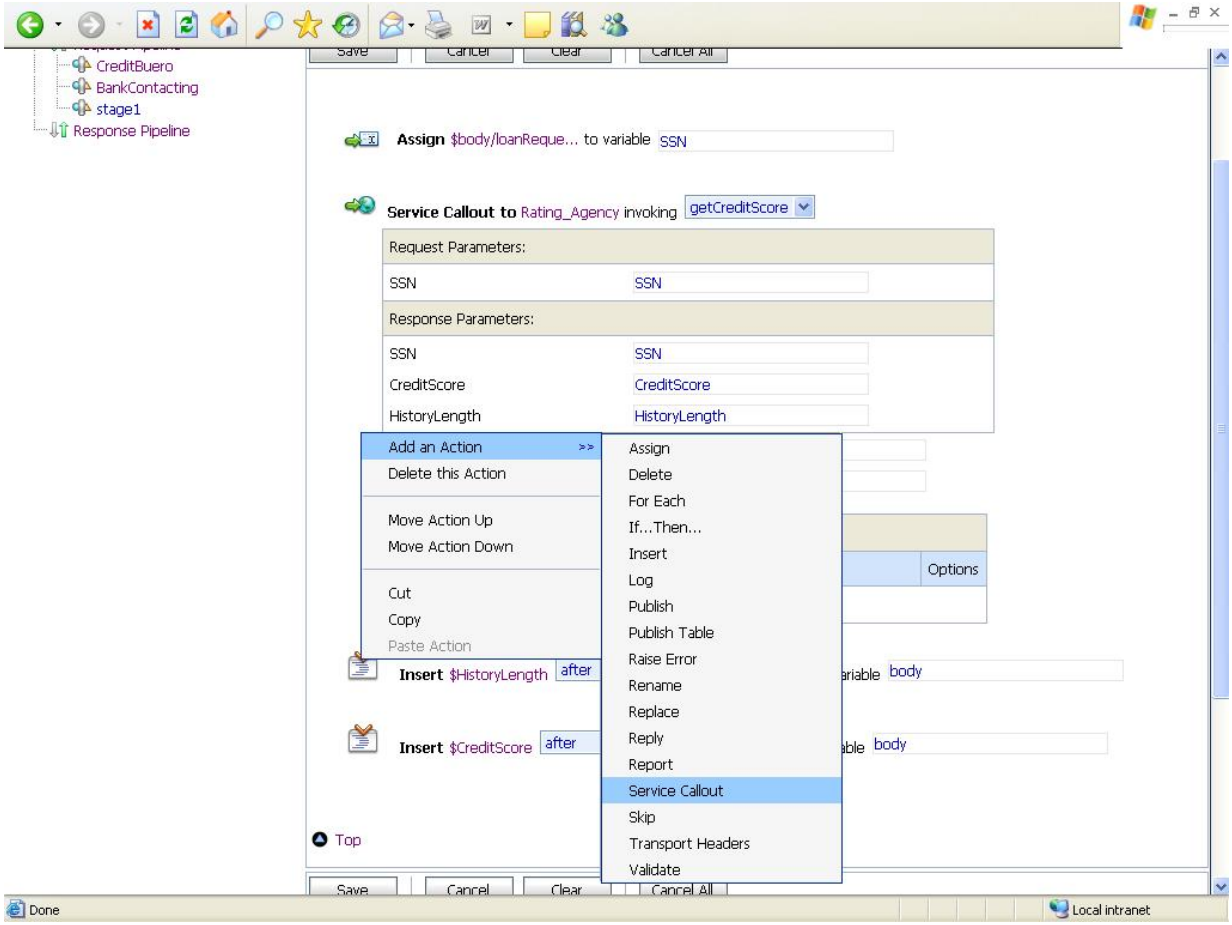

Abbildung 5.5: Anlegen von Aktionen

Überwachung des Nachrichtenflusses

In das Produkt ist ein leistungsfähiges Werkzeug zum Testen und zur Überwachung des Nachrichtenflusses integriert. Es ermöglicht das Nachvollziehen von Nachrichtenänderungen und die Richtigkeit der Aufrufe von externen Diensten. Siehe hierzu auch Abbildung 5.6.

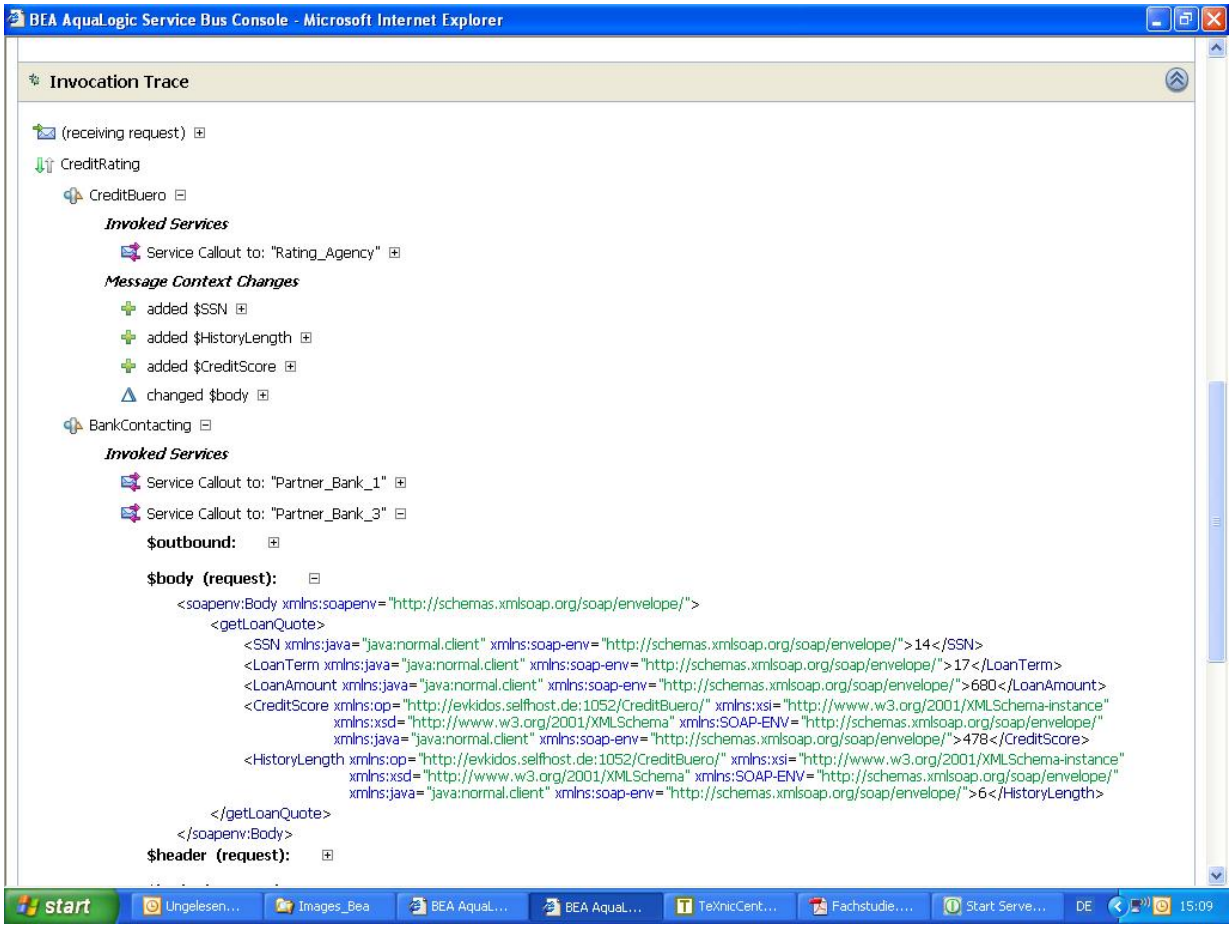

Abbildung 5.6: Überwachung des Nachrichtenflusses

Abweichungen zum Szenario

Das Szenario konnte durch die Anwendung gelöst werden, jedoch existiert eine Abweichung. Das Ansprechen der Banken geschieht in dem Szenario parallel und am Ende erfolgt eine Aggregation der zurückgesendeten Nachrichten.

Eine parallele Verarbeitung jedoch, wie sie hierzu erforderlich wäre, wird durch den AquaLogic Service Bus nicht untersützt.
Es existieren letztendlich folgende drei Möglichkeiten, um Verzweigungen in den Nachrichtenfluss einzubauen.

- „operational branching“ - Routing aufgrund der aufgerufenen Funktionen (wenn innerhalb der WSDL-Datei mehrere Operationen definiert sind).
- „conditional branching“ - Routing aufgrund von Entscheidungskriterien.
- „routing table“ - Routing aufgrund von Entscheidungskriterien.

Das „operational branching“ löst das Problem nicht, weil hier kein Durchlauf paralleler Verzweigungen geschieht, sondern lediglich der Pfad durchlaufen wird, der in Verbindung mit der aufgerufenen Operation innerhalb des Port Types der WSDL-Datei steht.
Mit dem „conditional branching“ und der „Routing Table“ lassen sich zwar, wie beim „operational branching“ auch, die zu durchlaufenden Zweige bestimmen. Es besteht aber das Problem, dass, sobald ein konkrete Verzweigung ausgewählt wurde, die anderen alternativen Zweige nicht mehr duchlaufen werden.

Es lassen sich also nur eindeutige und nicht parallelisierte Nachrichtenpfade definieren, so dass der Schritt der Parallelisierung und der Aggregation innerhalb des Szenarios auf eine Sequenz abgebildet werden musste.

5.1.3 Auswertung der Kriterien

Der AquaLogic Service Bus erreicht *25,5 von 33 Punkten*. Dies spiegelt zum einen die Vollständigkeit des Lösungsportfolios, als auch die erfolgreiche und einfache Umsetzung des Loan-Broker-Szenarios mit diesem Produkt wieder.

Erfüllbarkeit des Loan-Broker-Szenarios

8 von 10 Punkten
Das Szenario lässt sich ohne Probleme mit dem AquaLogic Service Bus lösen. Der Abzug von zwei Punkten ergibt sich aus der fehlenden Möglichkeit zur Parallelisierung der Aufrufe an die Banken (siehe hierzu auch 5.1.2).

Nachrichtenverarbeitung

4 von 4 Punkten
Die Funktionalität zur Nachrichtenverarbeitung ist umfangreich und bis auf sehr komplexe Abläufe, wie zum Beispiel die Agreggation parallelisierter Nachrichten, bieten die angebotenen Konstrukte vielfältige Möglichkeiten, um einen strukturierten Nachrichtenflusses aufzubauen.
Ein definierter Nachrichtenfluss wird dabei in einem dem AquaLogic Service Bus entsprechenden XML-Format gespeichert. In dieser Datei sind alle aufgerufenen Services, Bedingungen, Zuweisungen und Nachrichtenänderungen enthalten.
Die Punkte im Einzelnen:

- Routing
  Kriterienerfüllung: +
  Um die Nachrichten regelbasiert an die angeschlossenen Systeme übergeben zu können oder sie an ihnen vorbeizuleiten gibt es mehrere Möglichkeiten:

  - „conditional branching“ - Verzweigung durch Auswertung von Nachrichtenparametern.
  - „operational branching“ - Verzweigung durch die aufgerufene Operation innerhalb des Porttypes.
  - „Routing Table“ - Verzweigung durch Auswertung von Nachrichtenparametern.

  Diese Strukturierungselemente auf der höchsten Ebene der Flussdefinition bieten die Möglichkeit, Zweige für die Nachrichtendurchläufe zu definieren, wobei allerdings während eines konkreten Nachrichtendurchlaufs jeweils nur eine Verzweigung durchlaufen werden kann.
  Innerhalb einer Stage (Sammlung von Aktionen, die auf der Nachricht ausgeführt werden) oder eines Routingschrittes können dann jeweils wieder Bedingungen überprüft und entsprechend die sogenannten Service Callouts durchgeführt werden.

- Transformationen
  Kriterienerfüllung: +
  Transformationen sind ohne Einschränkung möglich. Die Definition einer Transformation kann auf drei Wegen erfolgen:

  – Durch Einbindung von externen XML-Style-Sheets.

  – Durch direkte Definition innerhalb der Anwendung, bei der die Dokumentstruktur grafisch angezeigt und dann modelliert wird. Dies geschieht auf der Grundlage von XPath-Expressions.

  – Durch Einbindung von XQuery-Sprachkonstrukten.

  Die Schnittstelle der Anwendung hat sich hier als sehr intuitiv und einfach zu bedienen herausgestellt hat.

Transaktionsschutz

0 von 2 Punkten
Der AquaLogic Service Bus selbst unterstützt keine Transaktionen. Es können lediglich Role-backs in Form von selbstgeschriebenen Error-Handlern definiert werden, um die Gewährleistung von Konsistenz aufrechtzuerhalten.
Auf der anderen Seite ist der Service Bus darauf ausgelegt mit anderen Produkten aus der Integrationssuite von BEA eng zusammenzuarbeiten.
Zum Beispiel liefert die AquaLogic Data Services Platform die gewünschte Funktionalität, indem hier transaktionsgeschützte Bereiche festgelegt werden können. Aber auch der Tuxedo Anwendungsserver mit seinen zahlreichen Adaptoren (Cics, Corba, etc.) kann dazu benutzt werden, da dieser Server als Transaktionsmonitor eingesetzt werden kann.

Sicherheit

2 von 2 Punkten
Authentifizierung (Erkennen des Benutzers), Authorisierung (Rechtevergabe für die Benutzer), Verschlüsselungsmechanismen sowie eine Zertifizierung für den abgesicherten Nachrichtentransport sind in den Weblogic Server integriert und damit auch für den AquaLogic Service Bus verfügbar. Sie können bei Bedarf eingebunden werden.
Authentifizierung und Authorisierung der Benutzer werden dabei über einen integrierten LDAP-Server (Export und Import möglich / Anbindung an externen Server möglich) gelöst, wobei der Schutz der einzelnen Anwendungen durch Policies geschieht. Die Berechtigungen für die Benutzer selbst lassen sich für einzelne Personen oder für Gruppen einstellen und es gibt bereits vordefinierte Rollen, die mit einem individuellen Rechtekatalog den Nutzern oder Gruppen zugewiesen werden können.
Zur Absicherung der Kommunikation mit dem Server werden verschiedenste Möglichkeiten zur symmetrischen und asymmetrischen Verschlüsselung der Nachrichten angeboten.

Beispielsweise HTTPS mit SSL-Verschlüsselung (symmetrische Verschlüsselung - der Client weist sich gegenüber dem Server aus), Zwei-Wege-SSL-Verschlüsselung (asymmetrische Verschlüsselung - die Parteien authentifizieren sich gegenseitig), X.509 Tokens und zahlreiche weitere Möglichkeiten werden dazu angeboten. Auch WS-Policy und WS-Security werden hier als Möglichkeit zur Definition von Sicherheitsbestimmungen unterstützt.
Für die domainübergreifende Authentifizierung und damit für eine globale Verwaltung der Benutzer wird von BEA das Enterprise Security Product Center zur Verfügung gestellt.

Integrationsspektrum

1 von 2 Punkten
Die eingebundenen Dienste können über verschiedenste Schnittstellen aufgerufen werden. Der Service Bus unterstützt FTP, HTTP, HTTPS, JMS (unter anderem geeignet für die Kommunikation mit „Message oriented Middleware“-Systeme) und die Email-Protokolle (POP/SMTP/IMAP) als bekannte Transportmechanismen direkt.
Alle anderen Protokolle oder Typen von XML-Nachrichten können durch ein Schema spezifziert werden. Für die Schemaerstellung wird das Werkzeug „Format Builder“ mitgeliefert, welches das Mapping in beliebige Protokolle vereinfacht.
Auch wenn mit Hilfe der Schemadefinition alle Schnittstellen abgebildet werden können, wäre die Bereitstellung einiger weiterer Adaptoren wie zum Beispiel Corba/IIOP wünschenswert gewesen. Den kompletten Satz an bereits durch BEA vorgefertigten Adaptoren erhält man aber erst, wenn weitere Lösungen von BEA, beispielsweise Tuxedo (siehe auch das Kriterium Transaktionsschutz) oder WebLogic Integration, eingesetzt werden, die dann als Proxy fungieren.

Die wichtigsten Standards

2 von 2 Punkten
WS-Interoperability, WS-Security, WS-Policy, WS-Addressing und WS-ReliableMessaging werden durch den Weblogic Server unterstützt, auf dem der AquaLogic Service Bus aufbaut.

Skalierbarkeit

2 von 2 Punkten
Da der AquaLogic Service Bus auf dem Weblogic Server aufbaut, können alle Skalierungs- und Deploymentmöglichkeiten des BEA Anwendungsservers benutzt werden. Dies bedeutet, dass Domains definiert werden und innerhalb dieser Domains Replizierung und Routing eingesetzt werden, um die Skalierung, die Verteilung und den Einsatz von Clustern zu ermöglichen.

Registrierung

0 von 2 Punkten
Der AquaLogic Service Bus selbst enthält keine Funktionalität zur Registrierung.
In der AquaLogic Produktsuite ist jedoch die AquaLogic Service Registry enthalten. Sie bietet die Möglichkeit, Dienste in einen Katalog, der den UDDI-Standard 3 unterstützt, aufzunehmen. Die Komponente arbeitet sowohl mit LDAP als auch mit Active Directory zusammen.

Benutzerfreundlichkeit

6,5 von 7 Punkten
Die Bedienung der Anwendung ist exakt auf die Aufgabe zugeschnitten. Da sie zudem stabil und robust arbeitet ist der Umgang mit der Anwendung sehr angenehm.
Die Punkte im Einzelnen:

- Aufgabenangemessenheit
  Kriterienerfüllung: +
  Die Bedienung der Anwendung ist exakt auf die Definition von Nachrichtenflüssen zugeschnitten. Der Aufbau einer Problemlösung wird unterstützt, indem die zentralen Aufgaben, also das Einbinden von Diensten und das Bearbeiten von Nachrichten, klar gegliedert und die dazu notwendigen Eingaben auf ein Minimum reduziert sind.

- Selbstbeschreibungsfähigkeit
  Kriterienerfüllung: +
  Die Bedienung der Anwendung ist, unter anderem durch die Selbstbeschreibungsfähigkeit, nach erfolgter Einarbeitung selbsterklärend.

- Steuerbarkeit
  Kriterienerfüllung: +
  Die Steuerbarkeit der Anwendung ist durch eine klare Gliederung und das Übertragen von Konzepten aus der Programmierung auf den Umgang mit Nachrichten gelungen. Es werden beipielsweise if-then-else- oder for-Konstrukte unterstützt.

- Erwartungskonformität
  Kriterienerfüllung: +
  Die Ergebnisse entsprechen den Erwartungen.

- Fehlertoleranz
  Kriterienerfüllung: +
  Die Anwendung reagiert nicht sensibel auf fehlerhafte Eingaben. Häufige Warnungen und die Möglichkeit einen Bearbeitungsstand jederzeit wieder laden zu können, vereinfachen zudem die Arbeit.

- Individualisierbarkeit
  Kriterienerfüllung: o
  Eine Individualisierbarkeit ist nicht vorgesehen, weil die Anwendung nicht Nutzerbezogen, sondern Aufgabenbezogen aufgebaut ist.

- Lernförderlichkeit
  Kriterienerfüllung: +
  Die Kenntnisse zum richtigen Umgang mit der Anwendung, also wie Nachrichtenflüsse aufgebaut werden können und wie die Anwendung effizient zu bedienen ist, verbessern sich durch die Nutzung der Anwendung.

5.1.4 Abschließende Betrachtung

BEA hat mit dem AquaLogic Service Bus ein ausgereiftes Produkt in der Kategorie ESB vorgelegt. Die Entwicklungsumgebung ist gut strukturiert aufgebaut, intuitiv zu bedienen und läßt wenige Wünsche bezüglich der Möglichkeiten zur Nachrichtenbearbeitung offen. Gut gelöst ist auch die Struktur der entwickelten Anwendungen, da ein Nachrichtenfluss vollständig in XML-Dateien abgelegt wird und damit eine gute Nachvollziehbarkeit der eigenen Definitionen möglich ist.
Für einen professionellen Einsatz ist allerdings zuerst zu überprüfen, ob der AquaLogic Service Bus ohne Erweiterungen eingesetzt werden kann, oder ob weitere Produkte von BEA für den Aufbau einer passenden Integrationslösung benötigt werden.

5.2 IBM WebSphere ESB (Ver. 6.0.1)

5.2.1 Einführende Produktbeschreibung

IBM WebSphere ESB stellt zwei Server zur Verfügung: den WebSphere Process Server und den WebSphere ESB Server. Als Entwicklungsumgebung dient der WebSphere Integration Developer (WID). Auf dem WepSphere ESB Server können nur Mediation-Module ausgeführt werden, auf dem WebSphere Process Server sowohl Mediation-Module, als auch Module, die auch BPEL-Prozesse, Maps, etc. beinhalten können.
Die ESB-Funktionalität soll durch Mediation-Module bereitgestellt werden, wobei es derzeit nicht möglich ist, dass eine Anfrage an mehrere Stellen, bzw. Dienste gleichzeitig geht, denn die Aggregierung der Antworten ist innerhalb eines Mediation-Moduls nicht möglich. Zwar ist es möglich mehrere Antworten in einen Filter oder in ein „Custom Mediation“-Objekt münden zu lassen. Sie können aber innerhalb des Filters oder des Custom-Mediation-Objektes nicht mehr unterschieden werden.

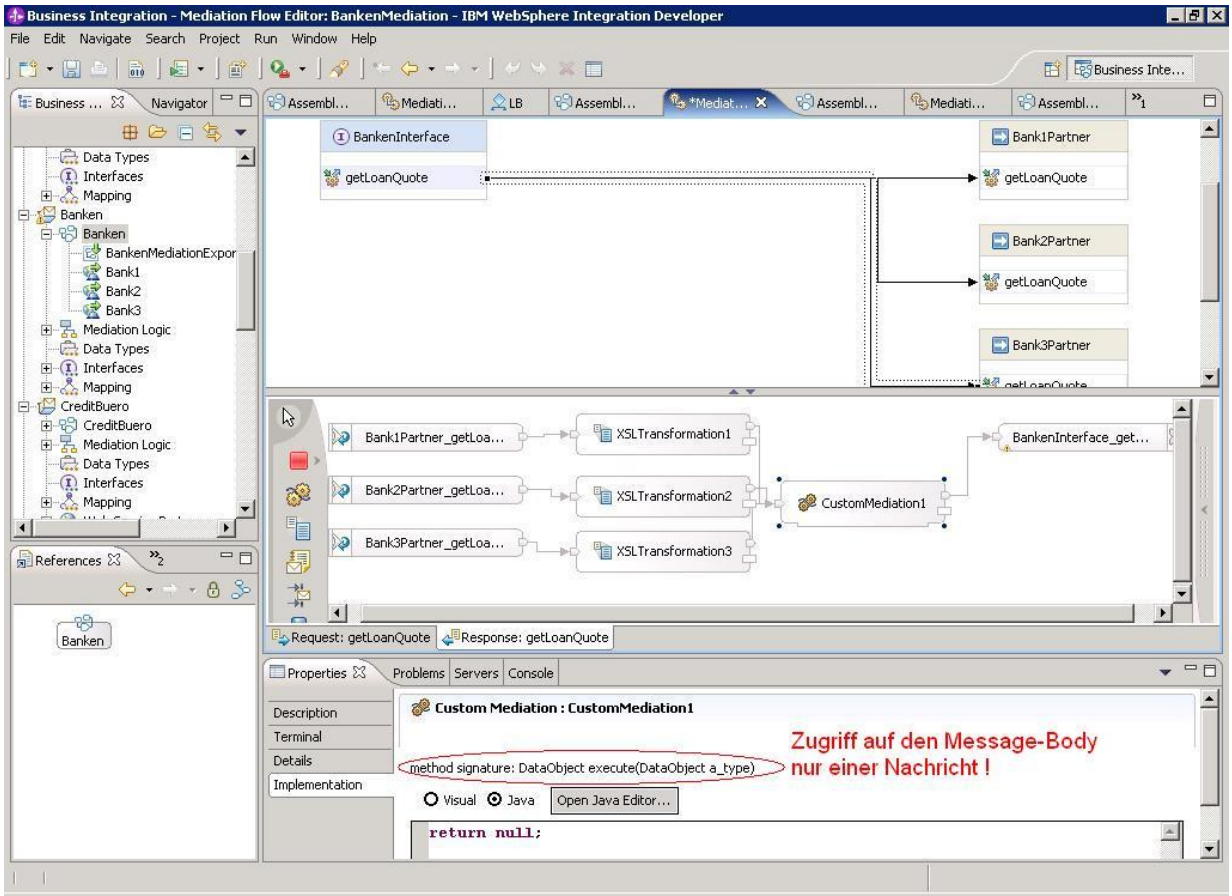

Abbildung 5.7: Aggregationsproblem innerhalb eines Mediation-Moduls

Somit ist ein Vergleich der Antworten und Auswahl der besten Alternative, wie vom Loan-

Broker-Szenario gefordert innerhalb eines Mediation-Moduls nicht möglich. Getetstet wurde der WID 6.0.1 mit Fixpack „Interim Fix 003“.
Um die Möglichkeiten aufzuzeigen, die ein Mediation-Modul bietet, sei an dieser Stelle das „StockQuote-Sample“ (Aktienkurs-Beispiel) aus der IBM „WebSphere ESB“-Dokumentation angeführt. Es ist das einzige dort aufgeführte Beispiel zu WebSphere ESB überhaupt, und wurde für die Evaluierung von IBM WebSphere ESB im Rahmen dieser Fachstudie mit dem WID implementiert. Mit diesem Beispiel wird der ganze Funktionsumfang eines Mediation-Moduls abgedeckt. Durch die Beschreibung des „Stock Quote Samples“ werden die Unterschiede zum Loan-Broker-Szenario und damit auch die Defizite der Mediation-Module deutlich.

Das Stock Quote Sample[1]

Ein Finanzdienstleistungsunternehmen bieten seinen Kunden einen interaktiven webbasierten Aktienmarktdienst an. Das Unternehmen will sich von seinen Wettbewerbern dadurch unterscheiden, dass es abgestufte Ebenen des Dienstes anbietet. Seinen Standardkunden will es Aktienkurse zeitverzögert anbieten und seinen Premiumkunden (Kunden, die den Dienst abonniert haben) Aktienkurse in Echtzeit.

Das Unternehmen will:

- Die Dienste „zeitverzögerte Aktienkursabfrage“ und „Aktienkursabfrage in Echtzeit“ als einen einzigen Dienst anbieten, der dynamisch bestimmt, welche externen Dienste aufgerufen werden.
- Alle Anfragen an den Dienst protokollieren, um Buchprüfungsanforderungen gerecht zu werden.

[1] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.wbit.sample.appl.6.sib.doc/stockquote/topics/ssample.html

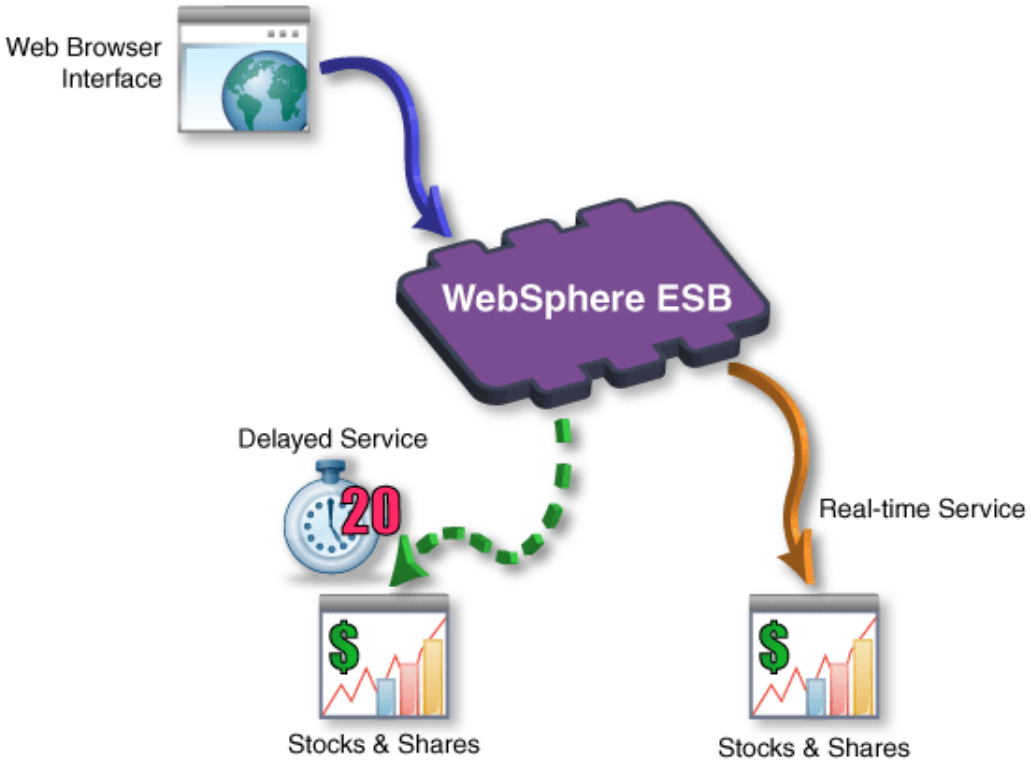

Abbildung 5.8: Übersicht über das Stock Quote Sample, Quelle: IBM WebSphere ESB Dokumentation

Entwurf des Beispiels

Der Mediationsdienst, der entweder auf dem WebSphere ESB Server oder auf dem WebSphere Process Server ausgeführt wird, ist in einem einzelnen Mediation-Modul mit Namen „StockQuote“ enthalten, das sich wie folgt zusammensetzt: Ein Export, der eine Schnittstelle für die Dienstanfrage des Clients zur Verfügung stellt, Importe, die Schnittstellen zu den externen Webservice-Anbietern zur Verfügung stellen, und eine Mediation-Flow-Komponente, die die Mediations-Implementierung definiert.

Das Mediation-Modul „StockQuote“ wird im Assembly-Editor zusammengestellt, und die Mediation-Flow-Komponente „StockQuote_MediationFlow“ wird im Meditation-Flow-Editor erstellt. Das folgende Bild zeigt die Beziehung zwischen den Schnittstellen und Referenzen im Assembly- und Mediation-Flow-Editor.

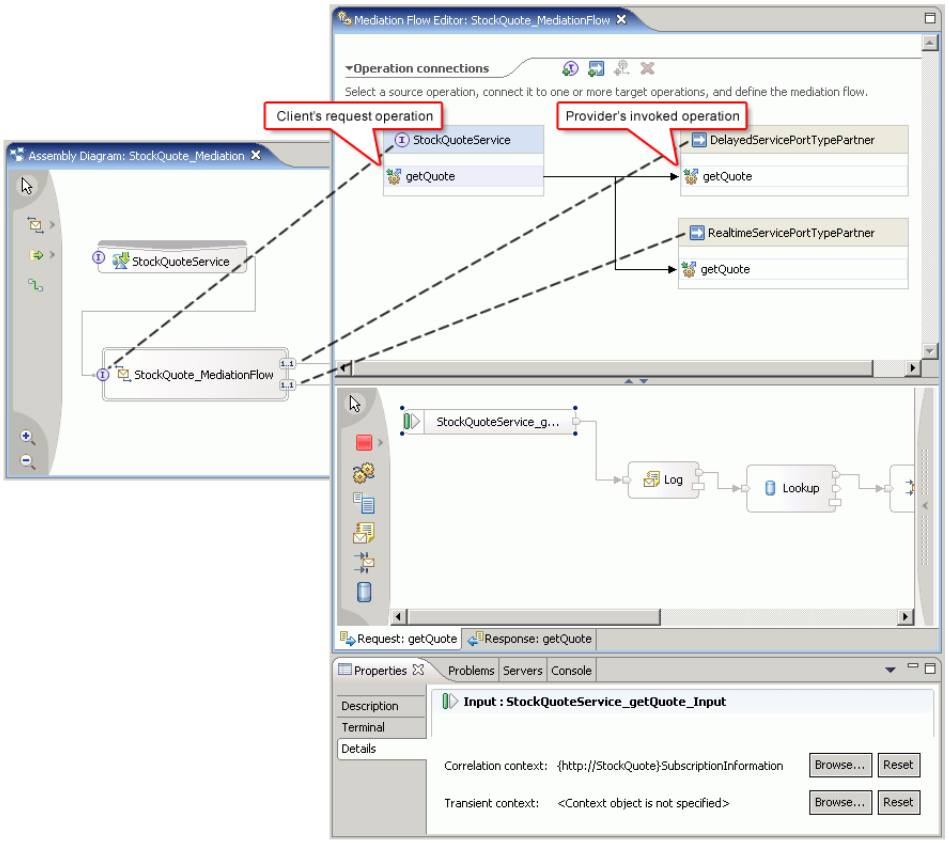

Abbildung 5.9: Beziehungen zwischen Schnittstellen und Referenzen im Assembly- und Mediation-Flow-Editor, Quelle: IBM WebSphere ESB Dokumentation

Das StockQuote-Modul

Das folgende Diagramm zeigt die Elemente des StockQuote-Moduls:

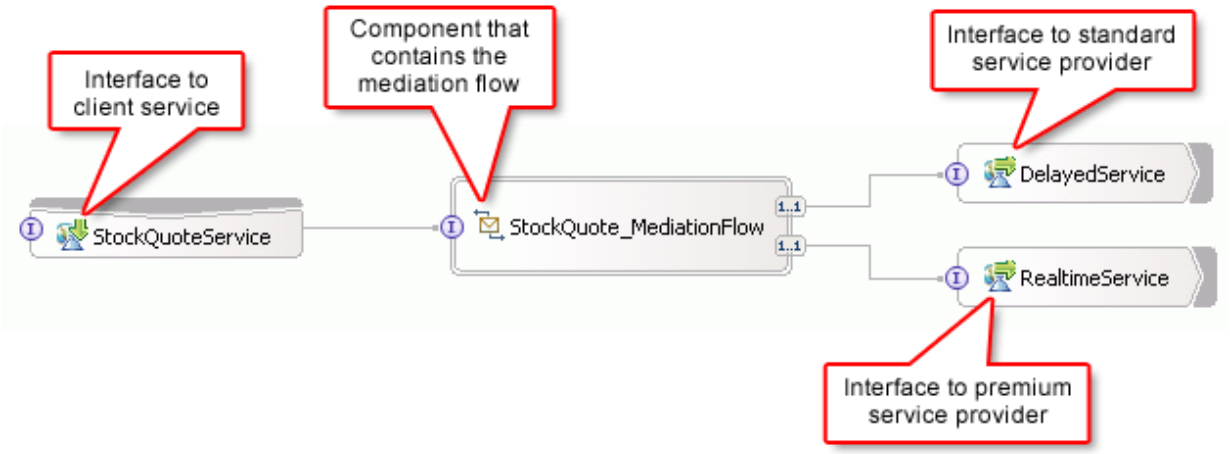

Abbildung 5.10: Das StockQuote-Modul, Quelle: IBM WebSphere ESB Dokumentation

- **StockQuoteService** hat eine WSDL-Schnittstelle mit dem Namen StockQuoteService, und verwendet eine SOAP/JMS-Webservice-Bindung, damit das Servlet-Front-End über JAX-RPC Verbindungen mit dem Mediation-Modul aufbauen kann.
- **StockQuote_MediationFlow** enthält den Mediationsfluss.
- **RealtimeService** besitzt eine Webservice-Bindung und eine Schnittstelle, die zum Echtzeitdienst passt.
- **DelayedService** besitzt eine Webservice-Bindung und eine Schnittstelle, die zum zeitverzögerten Dienst passt.

StockQuote_MediationFlow

Das folgende Diagramm zeigt den Fluss der Anfrage. Dieser Fluss definiert die Mediationslogik, die auf die Nachricht beim Durchwandern der „StockQuote_MediationFlow“-Komponente hin zu den Ziel-Dienstanbietern angewendet wird.

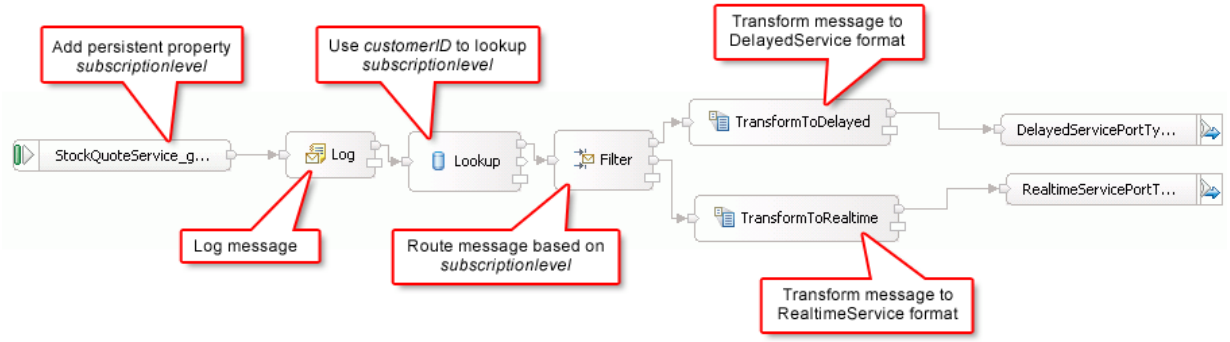

Abbildung 5.11: Die StockQuote_MediationFlow-Komponente, Quelle: IBM WebSphere ESB Dokumentation

1. Die Eigenschaft „subscriptionLevel“ wird im Correlation-Kontext der Nachricht gesetzt, so dass sie später im Antwortfluss verfügbar ist.

2. Die Anfrage wird unter Verwendung des „Message-Logger“-Primitivs, genannt „Log“, protokolliert.

3. Ein Datenbank-Such-Primitiv genannt „Lookup“ verwendet das „customerID“-Element im Nachrichten-Body, um zu bestimmen, ob der Kunde zur Benutzung des Premium- oder des Standarddienstes berechtigt ist, indem diese Information in der verfügbaren „CostumerDatabase“ nachgeschlagen wird. Diese Information wird zur späteren Verwendung der „subscriptionLevel“-Eigenschaft des Correlation-Kontexts der Nachricht hinzugefügt.

4. Die Anfrage wird dann von einem Nachrichtenfilter, genannt „Filter“, auf Grund der subscriptionLevel-Information im Correlation-Kontext entweder zum Echtzeit-Aktienkurs-Dienst oder zum zeitverzögerten Aktienkurs-Dienst weitergeleitet.

5. Auf dem Weg zum jeweiligen Dienst wird die Nachricht von den XSLT-Primitiven „TransformToDelayed“ bzw. „TransFormToRealtime“ umgeformt, so dass sie die Form haben, die der Dienst erwartet.

Das folgende Diagramm zeigt den Fluss der Antwort. Dieser Fluss definiert die Mediationslogik, die auf die zurückgegebene Nachricht beim Durchwandern der „StockQuote_MediationFlow“-Komponente von den Ziel-Dienstanbietern zurück zum Client angewendet wird.

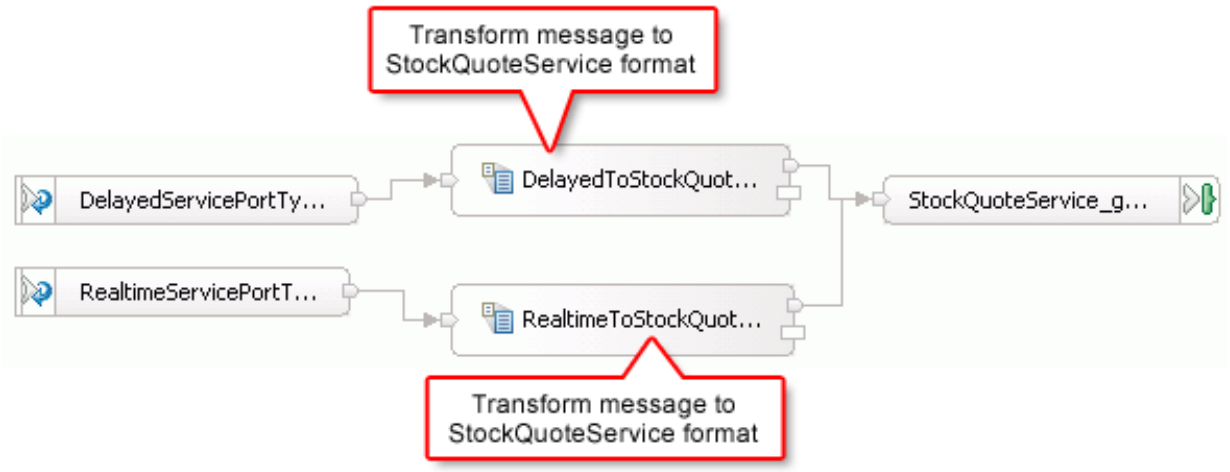

Abbildung 5.12: Der Antwortfluss in der StockQuote_MediationFlow-Komponente, Quelle: IBM WebSphere ESB Dokumentation

Die Antwort eines jeden Dienstes wird jeweils durch ein XSLT-Primitiv („DelayedToStockQuoteService & „RealtimeToStockQuoteService“) geleitet, um auf das vom „StockQuoteService“ benötigte Format gebracht zu werden. Zusätzlich werden die XSLT-Primitve auch verwendet, um den Wert von „subscriptionLevel“ im Correlation-Kontext auf die Eigenschaft „qualityOfService“ abzubilden. Der „qualityOfService“-Text zeigt bei einer Antwort, die vom Echtzeit-Dienst kommt, „Premium“ an, und „Standard“ bei einer Antwort, die vom zeitverzögerten Dienst kommt. Dieser Text kann im Client angezeigt werden, um anzugeben, welcher Dienstanbieter genutzt wurde.

Wie dieses Beispiel zeigt, ist das Routen und Transformieren von Nachrichten mit dem Mediation-Modul möglich, und auch recht komfortabel zu implementieren, solange es sich nur um eine Nachricht handelt, die geroutet, transformiert, ausgewertet werden muss. Auch ist es wohl möglich, mehrere externe Dienste (gleichzeitig) anzusprechen, solange nur ein Dienst eine Antwort zurückgibt. Sobald ein zweiter eine Nachricht zurückgibt, wie es im Loan-Broker-Szenario der Fall ist, ist ein sinnvolles Verarbeiten der Nachrichten nicht mehr möglich.

5.2.2 Durchführung des Loan-Broker-Szenarios

Um von den Entwicklungsumgebungen der untersuchten Produkte einen Eindruck zu gewinnen und vergleichen zu können, wie einfach, bzw. kompliziert das Erstellen des Loan-Broker-Szenarios ist, wird an dieser Stelle die Durchführung des Szenarios beschrieben.
Um also das Loan-Broker-Szenario mit dem WID zu erstellen zu können, muss auf ein „klassisches“ Modul zurückgegriffen werden, in das ein BPEL-Business-Process eingefügt

werden kann.
Zunächst wird ein Workspace für das WID-Projekt erstellt, in dem einfach mit dem Datei-Explorer des Betriebssystems ein leerer Ordner im gewünschten Verzeichnis erstellt wird. Daraufhin wird der WID gestartet - im nun folgenden Dialog wird der gerade erstellte Ordner als Workspace-Ordner angegeben und auf OK geklickt.
Nach dem Start muss der Willkomens-Bildschirm geschlossen werden. Damit kann das Erstellen des Szenarios beginnen, indem durch einen Rechtsklick in die „Business Integration"-Ansicht links oben eine neue Bibliothek angelegt wird, in der die verschiedenen Datentypen gespeichert werden, ebenso wie die verschiedenen Schnittstellen (Interfaces) in Form von WSDL-Dateien. Die Bibliothek wird „Ressourcen" genannt.
In dieser Bibliothek werden nun neue Datentypen (Business Objects) erstellt, die für die Kommunikation zwischen dem LoanBroker-Modul und den Mediation-Modulen, in denen der Aufruf der externen Dienste stattfindet, verwendet werden. Folgende Datentypen wurden erstellt:

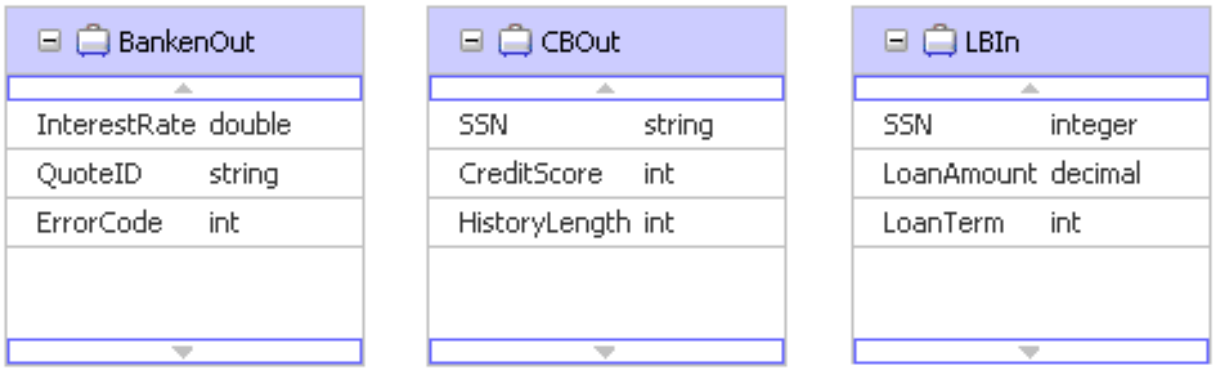

Abbildung 5.13: Die erstellten Datentypen

Um es Drittprogrammen (z.B. MS Visual Studio) zu erleichtern, aus den generierten WSDL-Dateien Stubs für das Ansprechen des Loan-Brokers zu generieren, wird LBIn nicht für die externe Schnittstelle des Loan-Brokers verwendet, sondern nur für die BankenInterface-Schnittstelle zur internen Kommunikation.
Jetzt werden die für die externe Kommunikation mit dem Kreditbüro und den Banken benötigten, zuvor erstellten WSDL-Dateien importiert. Zusätzlich werden die Schnittstellen neu erstellt, die der internen Kommunikation zwischen dem LoanBroker-Modul und den Mediation-Modulen dienen, in denen der Aufruf der externen Dienste stattfindet.

Folgende Schnittstellen wurden neu erstellt:

BankenInterface:

	Name	Type
getLoanQuote		
Input(s)	LBInput	LBIn
	CBOutput	CBOut
Output(s)	BankenOutput	BankenOut

CreditBueroInterface:

	Name	Type
getCreditScore		
Input(s)	CBInput	LBIn
Output(s)	CBOutput	CBOut

LoanBrokerInterface:

	Name	Type
getBestLoanQuote		
Input(s)	SSN	integer
	LoanAmount	decimal
	LoanTerm	int
Output(s)	SSN	integer
	LoanAmount	decimal
	InterestRate	double
	QuoteID	string

Abbildung 5.14: Neu erstellte Schnittstellen

Nun wird durch Rechtsklick in die „Business Integration“-Ansicht links oben und auf den Menüpunkt „New“ ein neues Modul angelegt, das LoanBroker-Modul, mit dem der Loan-Broker implementiert wird.
Dem Modul muss zunächst ein Verweis auf die Ressourcen-Bibliothek hinzugefügt werden, damit die dort definierten Datentypen und Schnittstellen im Modul verwendet werden können. Dies geschieht mit dem Dependency-Editor.
Nun wird im Assembly-Editor des LoanBroker-Moduls ein Business-Process hinzugefügt und „LB“ genannt.
Dem LB-Business-Process wird die LoanBrokerInterface-Schnittstelle hinzugefügt. Durch Rechtsklick auf den Business Process und Klick auf den Menüpunkt „Export“ ⇒ „Web Service Binding“ (Transport: „soap/http“) wird eine WSDL-Datei namens „LB_Export_LoanBrokerInterfaceHttpPort“ generiert. Diese WSDL-Datei importiert LoanBrokerInterface.wsdl. Liegen beide Dateien vor, kann über „LB_Export_LoanBrokerInterfaceHttpPort.wsdl der Loan-Broker-Webservice extern angesprochen werden.

Bevor mit dem LoanBroker-Modul fortgefahren werden kann, müssen nun die Mediation-Module erstellt werden, die für den Aufruf der externen Dienste (Credit-Buero und die drei Banken) eingesetzt werden. Beispielhaft wird hier nur die Erstellung des CreditBuero-Mediation-Moduls beschrieben. Die Mediation-Module für die Banken werden analog erstellt.
Analog zum Erstellen eines Moduls, wird ein Mediation-Modul durch Rechtsklick in die Business-Integration-Ansicht und Klick auf den Menüpunkt „New“ erstellt. Als Target Runtime wird der „WebSphere Process Server“ gewählt, das Mediation-Modul „CreditBuero“ genannt. Auch den Mediation-Modulen muss mit dem Dependency-Editor ein Verweis auf die Ressourcen-Bibliothek hinzugefügt werden.
Im Assembly-Editor des CreditBuero-Mediation-Moduls wird die Mediation-Komponente „MediationCB“ genannt, und entsprechend dem LB-Business-Process im LoanBroker-Modul wird ihr ebenfalls eine Schnittstelle hinzugefügt, und zwar die „CreditBueroInterface“-Schnittstelle.
Per Drag&Drop wird nun aus der Ressourcen-Bibliothek die „CreditBuero“-Schnittstelle in den Assembly-Editor des CreditBuero-Mediation-Moduls gezogen und im aufgehenden Dialog „Import with Web Service Binding“ gewählt. Der Import bekommt den Namen „CB“. Nun wird die Mediation-Komponente mit dem CB-Import verbunden und die Frage, ob eine passende Referenz am Quellknoten erstellt werden soll mit „OK“ beantwortet.
Damit ist im Assembly-Editor die Arbeit getan. Mit Rechtsklick auf die Mediation-Komponente und Klick auf den Menüpunkt „Generate Implementation“ wird der Mediation-Flow der Mediation-Komponente generiert und der Mediation-Flow-Editor öffnet sich.
Hier muss zunächst im oberen Bereich „Operation Connections“ die Quelloperation „getCreditScore“ mit der Zieloperation „getCreditScore“ verbunden werden.
Im unteren „Mediation-Flow“-Bereich muss nun die eingehende Anfrage auf die Anfrage an den externen Credit-Buero-Webservice abgebildet werden. Dazu wird ein „XSL Transformation“-Primitiv eingefügt und der „CreditBueroInterface_getCreditScore_Input“-Knoten mit diesem Primitiv verbunden und dann das XSL Tansformation“-Primitiv mit dem „CreditBueroPartner_getCreditScore_Callout“-Knoten verbunden.

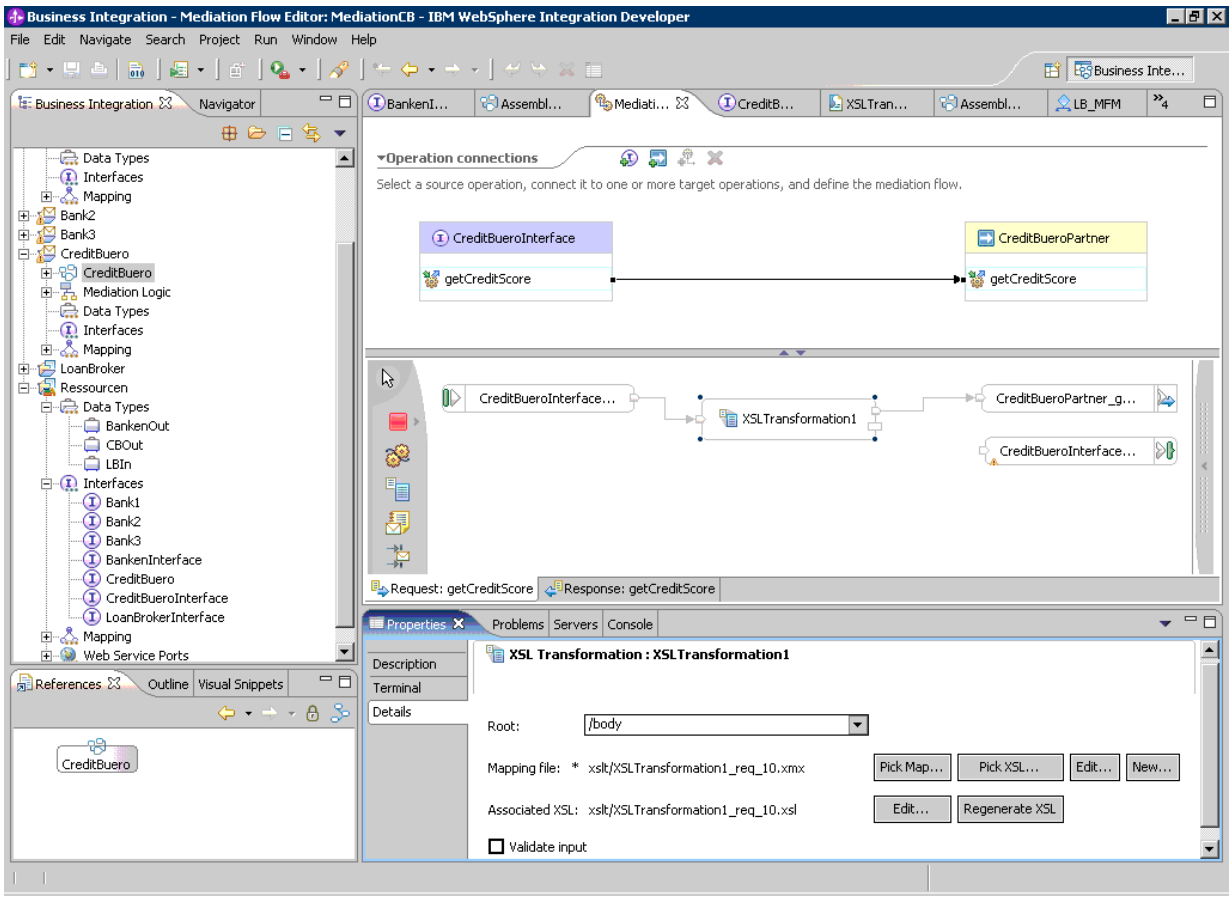

Abbildung 5.15: Der Mediation-Flow-Editor

Um die Transformation zu implementieren, muss das „XSL Transformation“-Primitiv markiert werden. Unter dem Tab „Details“ in der „Properties“-Ansicht nun in der Zeile „Mapping file“ auf die Schaltfläche „New...“ klicken. Im aufgehenden Dialog auf „Finish“ klicken - der „XSL Transformation“-Editor öffnet sich. Hier muss nun smo/body/getCreditScore/CBInput/SSN[0..1] der Quelle auf smo/body/getCreditScore/SSN des Ziels abgebildet werden, indem smo/body/getCreditScore/CBInput/SSN[0..1] (Quelle) markiert wird und dann mit Rechtsklick auf smo/body/getCreditScore/SSN (Ziel) das Kontextmenü aufgerufen wird. Hier muss nun der Menüpunkt „Create Mapping“ angeklickt werden. Damit wurde das Mapping für dieses Element erstellt. Für die CreditBuero-Anfrage müssen hier keine weiteren Elemente abgebildet werden. Die Änderungen müssen gespeichert und der „XSL Transformation“-Editor geschlossen werden. Auch im Mediation-Flow-Editor muss gespeichert werden. Durch Klick auf die Schaltfläche „Regenerate XSL“ wird die gerade erstellte Transformation angewendet.
Damit wurde der Nachrichtenfluss für die Anfrage komplettiert. Der Nachrichtenfluss für die Antwort muss genau so transformiert werden und wird im „Response: getCreditScore“-Reiter des Mediation-Flow-Editors analog zur Anfrage erstellt.

Die Mediation-Module der Banken werden entsprechend erstellt.

Wurden alle Mediation-Module erstellt, kann mit dem LoanBroker-Modul fortgefahren werden. Um nun die Mediation-Module aus dem LoanBroker heraus ansprechen zu können, müssen die die Schnittstellen „CreditBueroInterface“ (einmal) und „BankenInterface“ (dreimal) per Drag&Drop in den Assembly-Editor des LoanBroker-Moduls gezogen werden. Im dabei jeweils aufgehenden Dialog wird „Import with no binding“ gewählt. Die Imports werden entsprechend umbenannt (CB, Bank1, Bank2, Bank3). Für jeden Import ist das Vorgehen dann wie folgt: Rechtsklick auf den Import und den Menüpunkt „Generate Binding...“ und dann „SCA Binding“ wählen. Danach Rechtsklick auf den Import und den Menüpunkt „Select Service to Import“ wählen. Im aufgehenden Dialog können hier die Exporte der zuvor erstellten Mediation-Module gewählt werden. Dies wird für alle Importe gemacht. Der letzte Schritt im Assembly-Editor ist, die Importe mit dem LB-Business-Process zu verbinden - die Frage, ob eine Referenz zu erstellen ist, ist jedesmal mit „OK“ zu beantworten.
Der Assembly-Editor des LoanBroker-Moduls sollte jetzt ungefähr so aussehen:

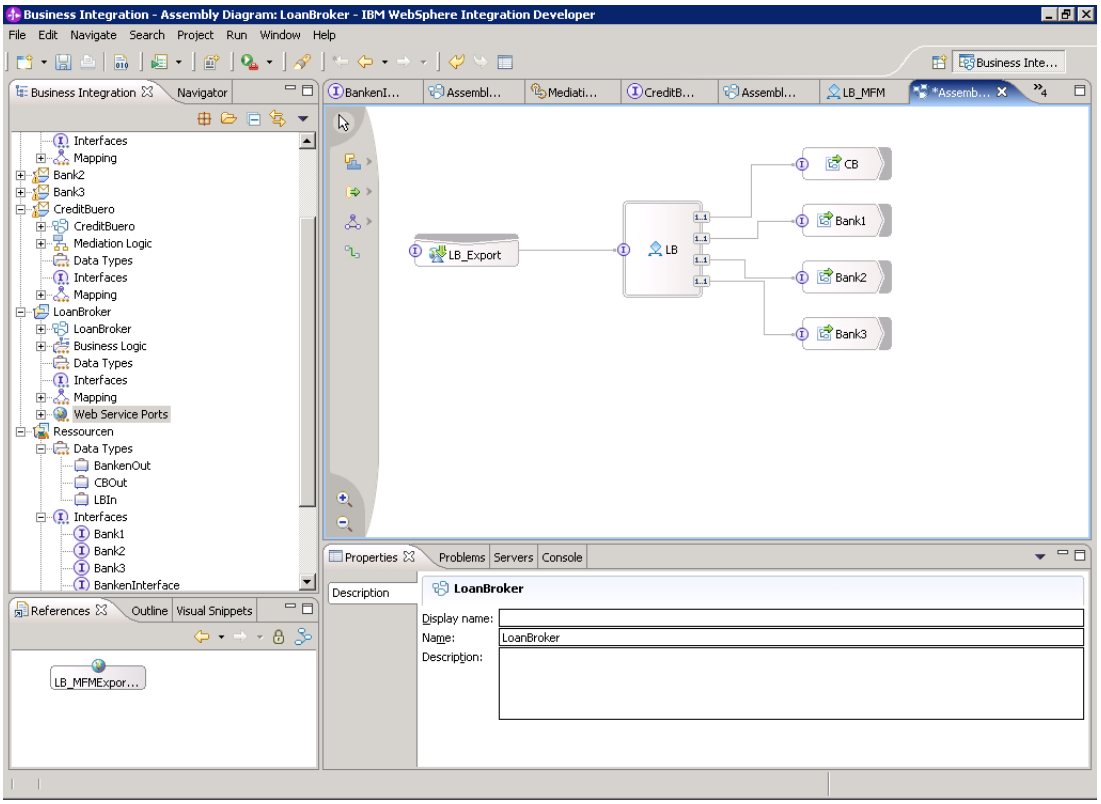

Abbildung 5.16: Das LoanBroker-Modul wurde im Assembly-Editor fertig gestellt

Mit Rechtsklick auf den LB-Business-Process und Klick auf den Menüpunkt „Generate Implementation“ wird die Implementierung generiert und es öffnet sich der Business-Process-Editor. Hier ist das Vorgehen nun wie folgt:

Zunächst werden folgende Variablen hinzugefügt:

- CBIn: Interace::CreditBueroInterface::getCreditScore::Input
- CBOut: Interace::CreditBueroInterface::getCreditScore::Output
- BankenIn: Interace::BankenInterface::getLoanQuote::Input
- B1Out: Interace::BankenInterface::getLoanQuote::Output
- B2Out: Interace::BankenInterface::getLoanQuote::Output
- B3Out: Interace::BankenInterface::getLoanQuote::Output

Nun werden folgende „Activities“ eingefügt:

- Assign-Activity „AssignCB“: Zuweisung der Variablen „CBIn“ für den Aufruf des Credit-Buero-Dienstes
- Invoke-Activity „CBInvoke“: Aufruf des Credit-Buero-Dienstes
- Assign-Activity „BankenAssign“: Zuweisung der Variablen „BankenIn“ für den Aufruf der Banken
- Parallel Activity: In diesen Container kommen folgende Activities:
  - Choice-Activity „ChoiceB1“: Bedingung für den Aufruf des Bank1-Dienstes
  - Choice-Activity „ChoiceB2“: Bedingung für den Aufruf des Bank2-Dienstes
  - Invoke-Activity „B1Invoke“: Aufruf des Bank1-Dienstes
  - Invoke-Activity „B2Invoke“: Aufruf des Bank2-Dienstes
  - Invoke-Activity „B3Invoke“: Aufruf des Bank3-Dienstes
- Assign-Activity „AssignOut“: Zuweisung der Elemente „SSN“ und „LoanAmount“ der „OutputVariable“
- Snippet „ProcessOut“: Code-Snippet, um das beste Kreditangebot anhand des Zinssatzes zu ermitteln und der „OutputVariable“ zuzuweisen.

Fertiggestellt sieht der LB-Business-Process wie folgt aus:

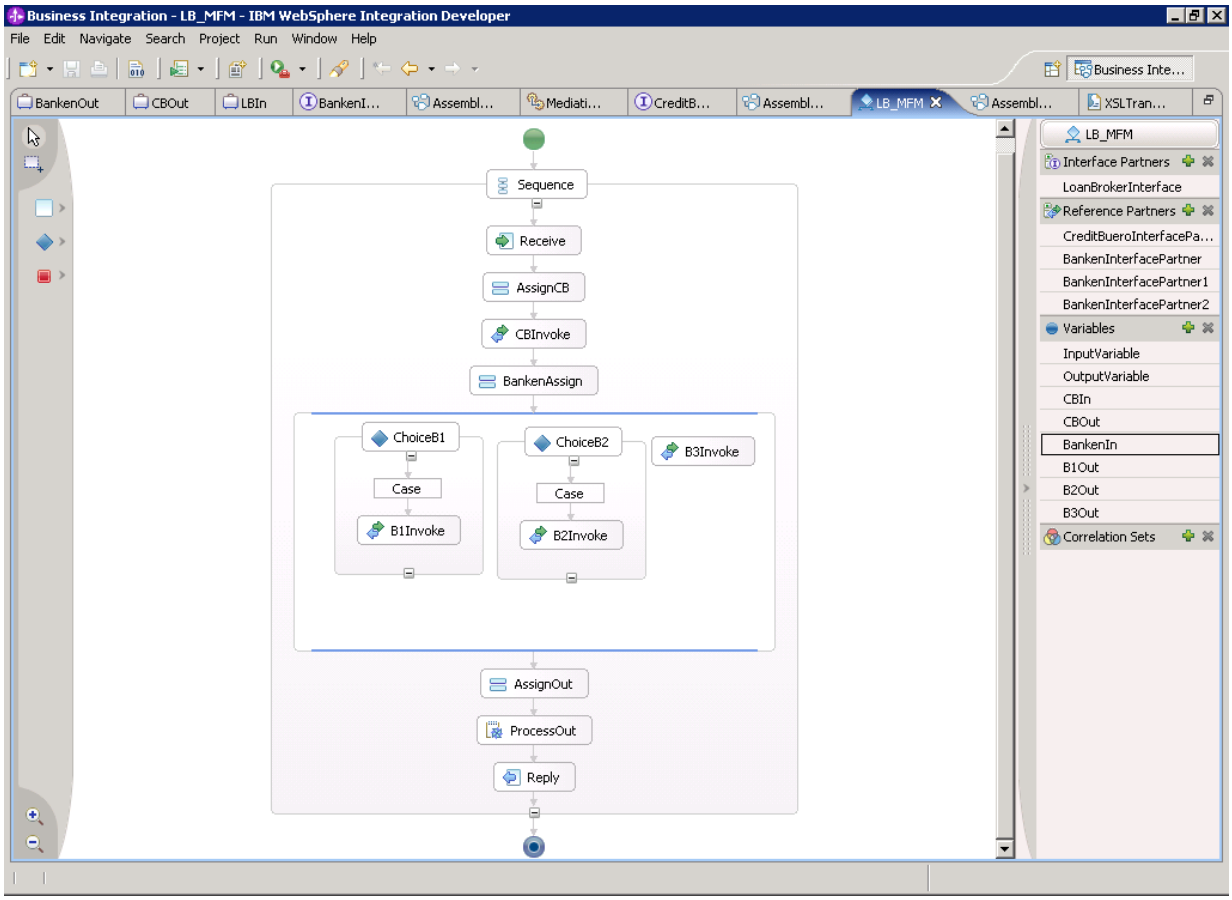

Abbildung 5.17: Der fertig gestellte LB-Business-Process

Damit wurde das Loan-Broker-Szenario vollständig mit dem WID erstellt. Um es zu testen, muss zunächst der Process-Server in der Server-Konsole gestartet werden.
Wurde der Server gestartet, können die Projekte (d.h. die Module/Mediation-Module LoanBroker, CreditBuero, Bank1, Bank2, Bank3) dem Server hinzugefügt werden, indem in der Server-Konsole mit Rechtsklick auf den Process-Server das Kontextmenü aufgerufen wird und dort „Add and remove projects...“ angeklickt wird.
Im folgenden Dialog müssen die Projekte „CreditBueroApp“, „Bank1App“, „Bank2App“, „Bank3App“ und „LoanBrokerApp“ über die Schaltfläche „Add >“ aus der Liste „Available projects“ der Liste „Configured projects“ hinzugefügt werden. Danach auf „Finish“ klicken. Um sicherzugehen, dass alle Projekte gestartet wurden, sollte nun im Kontextmenü des Process-Servers unter „Restart Project“ jedes Projekt nochmal neu gestartet werden. Wurde das für alle Projekte durchgeführt, sollte der Loan-Broker einsatzbereit sein und kann getestet werden.

5.2.3 Auswertung der Kriterien

WebSphere ESB erlangt 22 von 33 möglichen Punkten, und liegt damit trotz einer größeren Anzahl an verfügbaren Standards, und mehr Möglichkeiten zum Transaktionsschutz, zur Integration und zur Registrierung hinter dem „AquaLogic Service Bus“ von BEA und SONIC ESB. Zum Abzug führten vor allem der geringe Funktionsumfang der Mediation-Module und der überwiegend negative Eindruck des WebSphere Integration Developers.

Erfüllbarkeit des Loan-Broker-Szenarios

1 von 10 Punkten
Das Szenario ist mit Websphere ESB bedingt erfüllbar, d.h. mit ESB-Mitteln (Mediation-Modulen) nicht erfüllbar, da dort wichtige Funktionen, wie Aggregation mehrerer Nachrichten, Wiederverwendung einer Antwortnachricht für eine weitere Anfrage, bzw. die Transformation mehrer Nachrichten auf eine Nachricht fehlen.
Somit musste zur Umsetzung des Loan-Broker-Szenarios ein BPEL-Business-Process verwendet werden, der aber nicht zur originären ESB-Funktionalität zu zählen ist.

Nachrichtenverarbeitung

4 von 4 Punkten
Durch die an vielen Stellen einfügbaren Code-Snippets, bzw. Java-Komponenten sind der Nachrichtenverarbeitung kaum Grenzen gesetzt. Allerdings bieten auch Filter- und XSL Transformation“-Primitive (Mediation-Module), bzw. Choice-Activities (Business Process) bereits einige Möglichkeiten der Nachrichtenbearbeitung.
Die Punkte im Einzelnen:

- Routing
  Kriterienerfüllung: +
  Das Routen von Nachrichten wird unterstützt. In Mediation-Modulen sind im Mediation-Flow dafür die „Filter“-Primitive zuständig.
  In einem Business Process können hierfür „Choice“-Activities herangezogen werden.

- Transformationen
  Kriterienerfüllung: +
  Transformationen werden natürlich unterstützt. Innerhalb eines Mediation-Moduls sind im Mediation-Flow dafür die „XSL Transformation“-Primitive zuständig, mit deren Hilfe eine XSL-Datei generiert und auf die Nachricht angewendet werden kann. Der Anforderung, anhand von Merkmalen einer Nachricht eine XSL-Transformation aus einer Datenbank zu laden und diese auf die Nachricht anzuwenden, kann innerhalb eines Mediation-Moduls mit Hilfe eines „Custom Mediation“-Primitivs entsprochen werden, wenn das Laden und die Transformation von Hand in Java implementiert wird.
  Eventuell kann das Laden der XSL-Transformation aber auch durch ein vor dem

Custom-Mediation-Primitiv eingefügtes „Lookup“-Primitiv geleistet und die geladene XSL-Transformation im Correlation-Kontext der Nachricht gespeichert werden. Die Transformation selbst muss aber auf jeden Fall von Hand implementiert werden, d.h. es muss das Lesen der XSL-Transformation aus dem Correlation-Kontext der Nachricht und das Anwenden dieser auf die Nachricht in Java implementiert werden.

Transaktionsschutz[2]

2 von 2 Punkten
Bei der Verknüpfung von Komonenten im Assembly-Editor eines Mediation-Moduls besteht die Möglichkeit, Quality-of-Service-Merkmale (QoS-Merkmale) für Schnittstellen, Komponenten und Referenzen anzugeben. Diese Merkmale definieren, wie stark eine Komponente zur Laufzeit verwaltet werden muss.
Ein mögliches QoS-Merkmal ist „Transaction“. Es bestimmt die logische Arbeitseinheit, die von der Komponente während der Verarbeitung ausgeführt wird. Alle Änderungen an Daten, die während einer Transaktion durchgeführt werden, werden für eine logische Arbeitseinheit entweder zusammen als Einheit vorgenommen, oder zusammen als Einheit zurückgenommen. Mediation-Modulen steht also Transaktionsschutz zur Verfügung.
Des Weiteren wird auch der Webservice-Standard „WS-Atomic Transactions“ unterstützt (siehe auch 5.2.3).

Sicherheit

2 von 2 Punkten
Der WebSphere Process Server stellt eine Sicherheitsinfratruktur und -mechanismen mit folgenden Eigenschaften zur Verfügung:

- Authentifizierung (Authentication). Wenn die Sicherheit angeschaltet ist, müssen Clients authentifiziert werden. Wenn ein Client versucht, auf eine gesicherte Anwendung zuzugreifen, tritt eine Exception auf.

  Web-Clients (z.B. JSPs oder servlets) können mit einer HTTP-Basis-Authentifizierung eingerichtet werden, so, dass der Browser zur Eingabe von Benutzername und Passwort auffordert, wenn die URL aufgerufen wird.

  Java-Clients können JAAS für die Authentifizierung benutzen.

  Webservice-Clients können die Webservice/SOAP-Authentifizierung (WS-Security) benutzen.

  Einige WebSphere ESB-Komponenten haben Authentifizierungs-Aliase, die für den

[2] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.wiring.ui.doc/topicsib/csib_policies.html

Zugriff auf Datenbanken und Messaging-Engines zur Laufzeit verwendet werden. Es können bei der der Installation von WebSphere ESB für diese Authentifizierungs-Aliase Benutzer-ID und Passwort angegeben werden, oder auch später über die WebSphere-Verwaltungs-Konsole.

Einige Komponenten besitzen Nachrichten-getriebene Beans, die unter einer „Ausführen als“-Rolle konfiguriert wurden. Es können bei der Installation von WebSphere ESB für diese „Ausführen als“-Rolle Benutzer-ID und Passwort angegeben werden, oder auch später über die WebSphere-Verwaltungs-Konsole.

Unterpunkte sind:

- Authentifizierungs-Aliase modifizieren[3]
- Common-Event-Infrastructure-Authentifizierungs-Aliase[4]
- Service-Component-Architecture-Authentifizierungs-Aliase[5]

- Das Lightweight Directory Access Protocol (LDAP) wird als Benutzerverzeichnis unterstützt[6].

- Zugriffskontrolle (Access control)[7]. Zugriffskontrolle dient dazu, sicherzustellen, dass ein authentifizierter Benutzer die notwendige Erlaubnis hat, um eine bestimmte Operation auszuführen.

  Zugriffskontrolle kann für Komponenten, die entwickelt werden, eingerichtet werden, um diese sicher zu gestalten. Das geschieht unter Verwendung von Service-Component-Architecture-Merkmalen zur Entwicklungszeit.

- Das Sichern von Adaptern (Securing adapters)[8]. Als Teil des Sicherungsvorgangs der WebSphere ESB-Umgebung können auch die von WebSphere ESB verwendeten Adapter gesichert werden.

[3] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/doc/tsec_modifyauthalias.html

[4] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/doc/rsec_ceiauthent.html

[5] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/doc/rsec_scaauthent.html

[6] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/doc/tsec_ldap.html

[7] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/doc/csec_accesscontrol.html

[8] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/doc/tsec_adapters.html

WebSphere-Adapter ermöglichen eine verwaltete, bidirektionale Konnektivität zwischen Enterprise Service Systemen (EISen) und den von WebSphere ESB unterstützten Modulen und J2EE-Komponenten.

Für die eingehende Kommunikation in WebSphere ESB über einen Adapter gibt es keinen Authentifizierungsmechanismus, da die J2C-Funktionen, die von den WebSpphereESB-Adaptern benutzt werden, keine Sicherheitsunterstüzung für eingehende Nachrichten anbieten.

Um einen Adapter zu sichern, kann eine „Ausführen als“-Rolle für die Komponente, die Ziel des WebSphere-Adapter-Exports ist, definiert werden. Dies geschieht während der Entwicklung über das SCA-Merkmal „SecurityIdentity“.

Integrationsspektrum

2 von 2 Punkten
Für WebSphere existieren folgende Adapter (Stand 7. März 2006): Adapter for Enterprise JavaBean (EJB), Adapter Framework, Ariba Buyer, Clarify CRM, COM, CORBA, Data Handler for Complex Data, Data Handler for EDI, Data Handler for XML, Development Kit (Programmierschnittstellen in Java und C++), DTS Protocol, e-mail, eMatrix, Exchange, FastMRP for WebSphere Business Integration, FIX Protocol, Flat Files, Healthcare Data Protocols, HTTP Adapter, i2, iSeries, JD Edwards OneWorld, JDBC, JMS, LDAP, Lotus Domino, XA08: Adapter for MDSI Advantex Server V7.4, MetaSolv Applications, Oracle Applications, XA07: Adapter for Peace Energy V7.1, PeopleSoft Enterprise, Portal Infranet, UGS' Teamcenter or PTC's Windchill integration using IBM's WebSphere Business Integration, SAP Exchange Infrastructure, SAP Software, Selectica Configurator, Siebel Business Applications, SMARTEAM Gateway (GWY), SunGard FRONT ARENA, SWIFT, SynphonyRPM, Inc. for IBM WebSphere® Business Integration Adapter, TCP/IP, Web Services, WebSphere Adapter Toolkit, WebSphere Message Broker, WebSphere MQ Adapter, WebSphere MQ Workflow Adapter.

Die wichtigsten Standards

2 von 2 Punkten
Das Produkt unterstützt folgende Standards[9]:

- Java Message Service (JMS) 1.1, zur Verfügung gestellt durch den WebSphere Application Server, auf den WebSphere ESB aufbaut. Anwendungen können eine Vielzahl an Transportprotokollen nutzen, einschließlich TCP/IP, SSL, HTTP, und HTTPS.
- WebSphere ESB unterstützt folgende Webservices-Standards

[9] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.wesb.doc/concepts/cwesb_wesb.html

- SOAP/HTTP, SOAP/JMS, WSDL 1.1
- UDDI 3.0 Service Registry, Web Services Gateway
- WS*-Standards, einschließlich WS-Security und WS-Atomic Transactions.

Skalierbarkeit[10]

2 von 2 Punkten
Abhängig von den Anforderungen können verschiedene Bus-Topologien und Busumgebungen eingesetzt werden. Dabei sind folgende Topologien und Umgebungen möglich:

- Einen Einzelserver-ESB[11]
- Einen Mehrfachserver-ESB ohne Clustering[12]
- Einen Mehrfachserver-ESB mit Clustering[13]
- Einen ESB mit Verbindungen zu WebSphere MQ-Netzwerken[14]

Registrierung

2 von 2 Punkten
IBM WebSphere ESB besitzt ein eigenes UDDI-Verzeichnis-Komponente[15].

Benutzerfreundlichkeit der Entwicklungsumgebung

5 von 7 Punkten
Die Bedienung der Anwendung ist recht intuitiv und auch komfortabel. Negativ ist, dass nach jedem Schritt gespeichert und neu erstellt („Build Project“) werden muss, damit keine Probleme auftreten. Das Projekt am Ende zu speichern und zu erstellen führte immer zu Problemen, die nicht mehr behoben werden konnten, d.h. das gesamte Projekt musste neu angelegt werden.
Weiterhin fiel die für den Start der Server benötigte Zeit negativ auf, ebenso die Zeit, die für den Neustart eines Projektes benötigt wird.

[10] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.wesb.doc/concepts/cwesb_busenv.html
[11] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/tasks/tjj0070_.html
[12] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/tasks/tjj0071_.html
[13] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/tasks/tjj0072_.html
[14] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/concepts/cjj0005_.html
[15] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.nd.doc/info/ae/ae/twsu_ep.html

Auch das Erstellen der Projekte an sich dauert ziemlich lange.
Die Punkte im Einzelnen:

- Aufgabenangemessenheit
  Kriterienerfüllung: +
  Die Bedienung des WID ist der Aufgabe, nämlich Integrationsprojekte zu entwickeln angemessen.

- Selbstbeschreibungsfähigkeit
  Kriterienerfüllung: o
  Nach der Lektüre der Dokumentation ist die Benutzungsoberfläche des WID weitestgehend selbsterklärend.
  Treten während des Programmablaufes Exceptions auf, sind diese nicht sehr aussagekräftig und helfen nicht, den Fehler zu beheben.

- Steuerbarkeit
  Kriterienerfüllung: o
  Die Steuerbarkeit des WID ist teilweise recht gut. Andererseits lassen sich die Reiter einer Ansicht aber z.B. nicht mit der Tastenkombination STRG+Tab wechseln und Elemente in der Business-Integration-Ansicht lassen sich, einmal angelegt, nicht mehr umbenennen.

- Erwartungskonformität
  Kriterienerfüllung: +
  Der WID und die Dialoge des WID sind erwartungskonform, entsprechen also den Merkmalen des Benutzers in Bezug auf dessen Kenntnisse des Arbeitsgebietes, dessen Erfahrungen sowie den allgemein anerkannten Konventionen.

- Fehlertoleranz
  Kriterienerfüllung: -
  Die Fehlertoleranz des WID (Ver. 6.0.1, Fixpack „Interim Fix 003“) ist ungenügend. Teilweise erkennbare fehlerhafte Eingaben werden nicht direkt abgefangen. Evtl. werden beim Erstellen der Projekte die Fehler erkannt und angezeigt. Oftmals werden aber auch beim Erstellen erkennbare Fehler übersehen, so dass beim Porgrammablauf der Projekte Exceptions auftreten.

- Individualisierbarkeit
  Kriterienerfüllung: +
  Die Benutzungsoberfläche des WID ist in hohem Maße individualisierbar.

- Lernförderlichkeit
  Kriterienerfüllung: +
  Das System unterstützt die Erlernung des Umgangs mit dem WID und ist somit lernförderlich.

5.2.4 Abschließende Betrachtung

WebSphere ESB bietet rudimentäre ESB-Funktionalitäten, wie Routing und Transformation, das Loan-Broker-Szenario lässt sich allerdings nicht mit den ESB-Bordmitteln von WebSphere ESB nachbilden. Um dies bewerkstelligen zu können, fehlt den Mediation-Modulen

- die Möglichkeit, mehrere externe Dienste gleichzeitig anzusprechen,
- die Möglichkeit, Antworten der externen Dienste sinnvoll zu aggregieren und
- die Möglichkeit, Antwortnachrichten für die Anfrage an andere externe Dienst weiterzuverwenden.

Allerdings stehen für WebSphere ESB zahlreiche Adapter für die Integration von Drittprogrammen zur Verfügung (siehe 5.2.3), es können umfangreiche QoS-Einstellungen für die Kommunikation (Stichwort „Transaktionsschutz“) vorgenommen werden (siehe 5.2.3) und es werden zahlreiche Kommunkitations- und Webservice-Standards (siehe 5.2.3) unterstützt.
Betrachtet wurde auch das Entwicklungswerkzeug für WebSphere ESB, der WebSphere Integration Developer (WID). Dieser zeichnet sich nicht durch überragende Stabilität, absolute Fehlerrobustheit, hervorragende Benutzerfreundlichkeit und die Abwesenheit jeglicher Fehler aus.
Im Anhang ist eine WSDL-Datei angeführt, die mit dem WSDL-Editor des WID erstellt wurde, die also dieser daselbst generiert hat, mit dem das Ansprechen des betreffenden Dienstes aus dem WID heraus allerdings jedes Mal mit einer Exception endete.
Das MS Visual Studio konnte aus dieser WSDL-Datei einen funktionierenden Stub generieren, der im WID integrierte Web Services Explorer hatte keine Mühen, den Dienst anhand dieser WSDL-Datei aufzurufen, nur der Process Server, auf dem das CreditBuero-Mediation-Modul ausgeführt wurde, lieferte jedes Mal eine Exception.
Es bleibt festzuhalten, das der WID, zumindest teilweise, WSDL-Dateien generiert, die er selbst nicht verarbeiten kann.
Negativ aufgefallen ist auch die Tatsache, dass die IP-Adresse für die URL, über die die Kreditbüro- und Bankdienste erreichbar waren, vom WID nur einmal während des ersten Ausführens aufgelöst wurde, und dann angenommen wird, die IP-Adresse ändere sich nicht mehr und müsse also nicht mehr neu aufgelöst werden. Da es aber eine DynDNS-Adresse war, und die IP-Adresse sich daher alle 24 h änderte, waren die Webservices am nächsten Tag nicht mehr erreichbar. Dies konnte jeweils nur mit einem Neustart des WID behoben werden.
Zur Bedienung des WID ist negativ aufgefallen, dass sich die Reiter der verschiedenen Fenster nicht mit STRG+Tab wechseln lassen, und sich Elemente der Business-Integration-Ansicht nicht mehr umbenennen lassen.
Insgesamt ist die Bedienung aber dennoch recht intuitiv und, z.B. im Vergleich zu SONIC, recht komfortabel.

5.3 Sonic Enterprise Service Bus (Ver. 6.1)

5.3.1 Einführende Produktbeschreibung

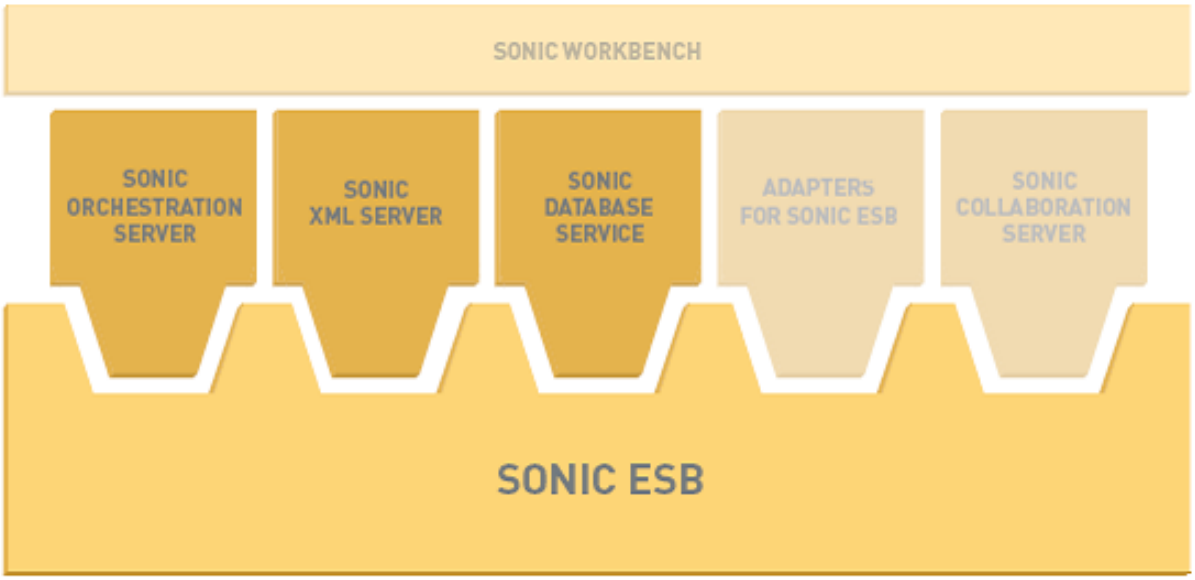

Abbildung 5.18: Der Sonic ESB als Fundament des Sonic SOA Suite

Der Sonic ESB ist das Fundament der Sonic SOA Suite. Die Sonic SOA Suite beinhaltet unter anderem den Sonic Orchestration Server, Sonic Collaboration Server, Sonic XML Server und Sonic Database Service. Sie bildet eine ESB-basierte Serviceplattform, welche die Verwaltung von Geschäftsprozessen und die Transparenz von Vorgängen im gesamten Unternehmen ermöglicht. Um die volle ESB-Funktionalität zu gewährleisten, werden die unten aufgeführten Komponenten für den Sonic ESB benötigt.

Sonic Orchestration Server - Ermöglicht das Management von Geschäftsprozessen innerhalb des Unternehmens. Der Sonic Orchestration Server erweitert die intelligenten Routing-Funktionen des Sonic ESB und ermöglicht die Modellierung, Automatisierung und das Management komplexer, zustandsabhängiger Geschäftsprozesse im gesamten Unternehmen. Der Orchestration Server integriert über den ESB alle teilnehmenden Services in einen koordinierten und geregelten Geschäftsprozess.

Sonic XML Server - Sorgt für eine skalierbare und optimierte Speicherung, Abfrage, Transformation und Verarbeitung von XML-Daten und nutzt XSLT- und XQuery-Standards, um Audit-, Protokoll- und Überwachungsfunktionen, Funktionen für die Bereitstellung und Zusammenfügung von Daten sowie zur Erfassung geschäftlicher Ereignisse im gesamten ESB zu unterstützen.

Sonic Database Service - Ermöglicht eine Integration relationaler Datenquellen in die service-orientierte Architektur. Mit dem Sonic Database Service lassen sich Abfragen, Updates und „stored procedures“ als verfügbare Services auf dem Sonic ESB allgemein konfigurieren und ausführen. Der Sonic Database Service führt die SQL-Abfrage automatisch auf der Grundlage eingehender XML-Messages aus und transformiert die Ergebnisse wieder in XML. Der ESB-fähige Sonic Database Service ermöglicht die Remote-Administration von Datenbankverbindungen und die Konfiguration über die Sonic Management Console.

Sonic Collaboration Server - Erweitert die Funktionalität von Sonic ESB um die Option, externe Geschäftspartner über B2B-Protokolle und Web-Services-Standards zu integrieren.

Adapter for Sonic ESB - Ermöglichen service-basierte Interaktionen mit über 200 Anwendungen und Systemen, einschließlich vorgefertigter Anwendungen (z.B. SAP, PeopleSoft), B2B Systemen (z.B. EDI, SWIFT, HIPAA), Mainframe-Applikationen und Legacy-Systemen.

Sonic Workbench - Ist eine umfassende Entwicklungsumgebung für standard-basierte Integrationsprojekte. Neben XQuery und einem visuellen XML-Transformation-Mapper und -Debugger enthält die Sonic Workbench intelligente Route-Builder sowie einen Business-Process-Modeler.

SonicMQ - Ist ein robustes und beständiges Enterprise-Messaging-System. Neben Verfügbarkeit und Performance zeichnet es sich durch umfangreiche Managementmöglichkeiten und ein hohes Maß an Skalierbarkeit und ständige Verfügbarkeit für komplexe Implementierungen aus. Cluster-Technologien sorgen zusammen mit der Dynamic Routing Architecture dafür, dass SonicMQ-Implementierungen grenzenlos erweitert werden können.

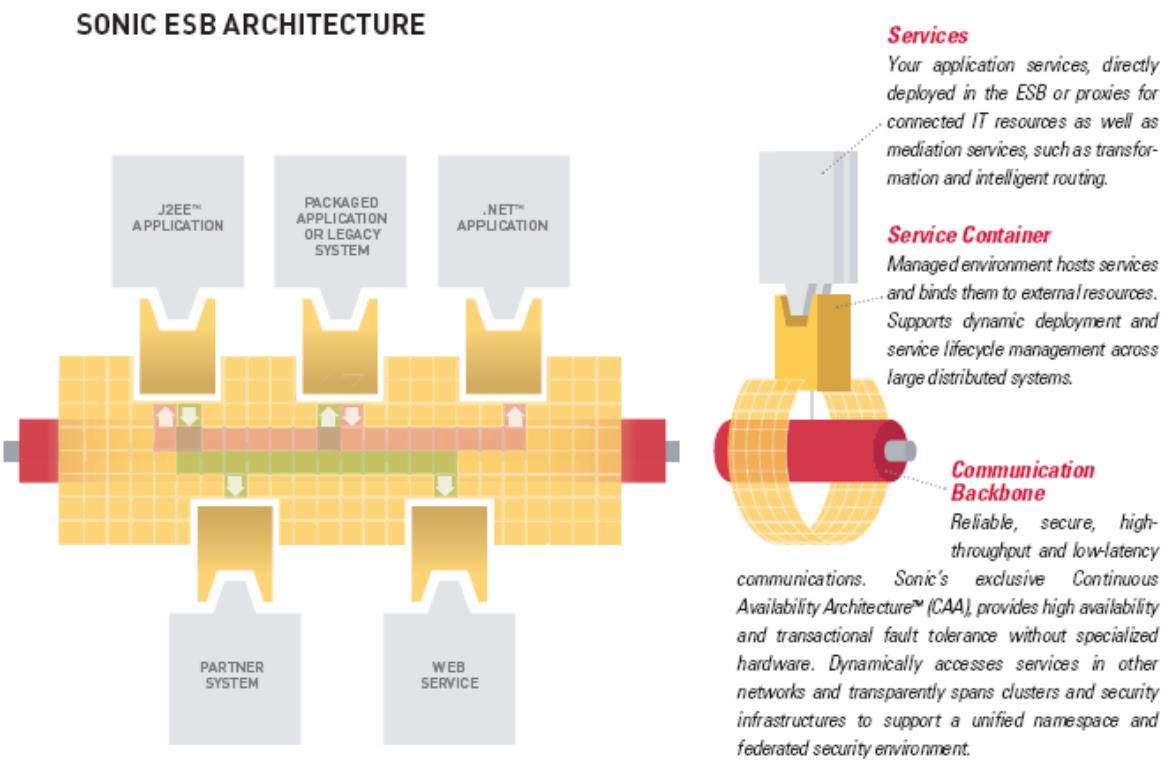

Abbildung 5.19: Architektur des Sonic ESB

Der Sonic ESB setzt auf dem SonicMQ als Messaging-System auf und verfügt dadurch über die Vorteile, die das System mit sich bringt. Auf dem Sonic ESB sind alle verbundenen Ressourcen, ob J2EE-Anwendungen, Web Services oder Legacy Messagebroker gleichberechtigt. Diese Services stehen allgemein für die dynamisch konfigurierte Interaktion mit anderen Services zur Verfügung. Die Services können ohne Unterbrechung anderer aktiver Services skaliert, verschoben oder aktualisiert werden. Die Servicekonfigurationen und Servicebeziehungen werden in einem global zugänglichen Speicher mit verteiltem Cache abgelegt. Laut eigenen Angaben ist das Repository schnell und steht ständig zur Verfügung, selbst bei Netzwerkausfällen. Zum Erstellen und zur Konfiguration von Containern, Services, Prozessen und Endpoints steht der ESB-Explorer zur Verfügung. Dieser ist in die Management Console integriert.

Weitere Informationen zum Sonic ESB und den anderen Sonic Produkten sind unter [SON1] und [SON2] zu finden.

5.3.2 Durchführung des Szenarios

Um das Loan-Broker-Szenario mit Sonic realisieren zu können, muss zunächst der SonicMQ DomainManager gestartet worden sein. Des Weiteren muss man die Sonic Management Console und innerhalb dieser Console den Sonic ESB Explorer starten. Beim Start der Sonic Management Console muss der Client erst eine Verbindung zum SonicMQ DomainManager aufbauen. Die Verbindung wird nur über eine Authentifizierung des Clients zugelassen (Abbildung 5.20). Um den Loan-Broker-Prozess zu modellieren, muss dann noch das Stylus Studio IE gestartet werden.

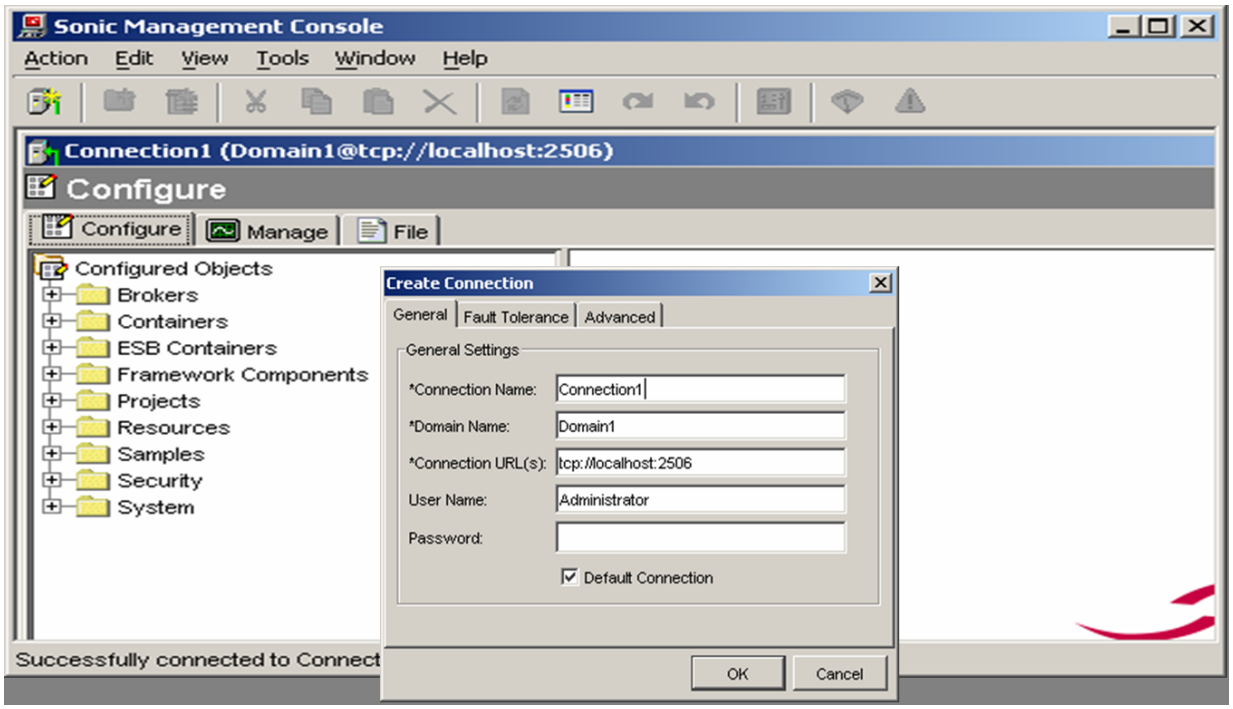

Abbildung 5.20: Sonic Management Console: Connection zum SonicMQ

Als erstes sollte man einen neuen Container in der Sonic Management Console für das Loan-Broker-Szenario anlegen. In diesen Container wird dann ein ESB-Container abgelegt, der alle Services und Prozesse, die für das Szenario nötig werden, beinhaltet. Man könnte aber auch einen schon vorhandenen Container für diesen Zweck nehmen.

Container erstellen

Zum Erstellen des Loan-Broker-Containers muss man in der Management Console (Abbildung 5.21) im Reiter „Configure“ den Ordner „Containers“ auswählen. Mit dem „Create new configuration“ Button kann man einen neuen Container erstellen. Für den neu erstellten Loan-Broker-Container wird nun ein Bootfile erstellt, indem im Kontextmenü des

Loan-Broker-Containers „Generate Boot File ...“ ausgewählt wird.

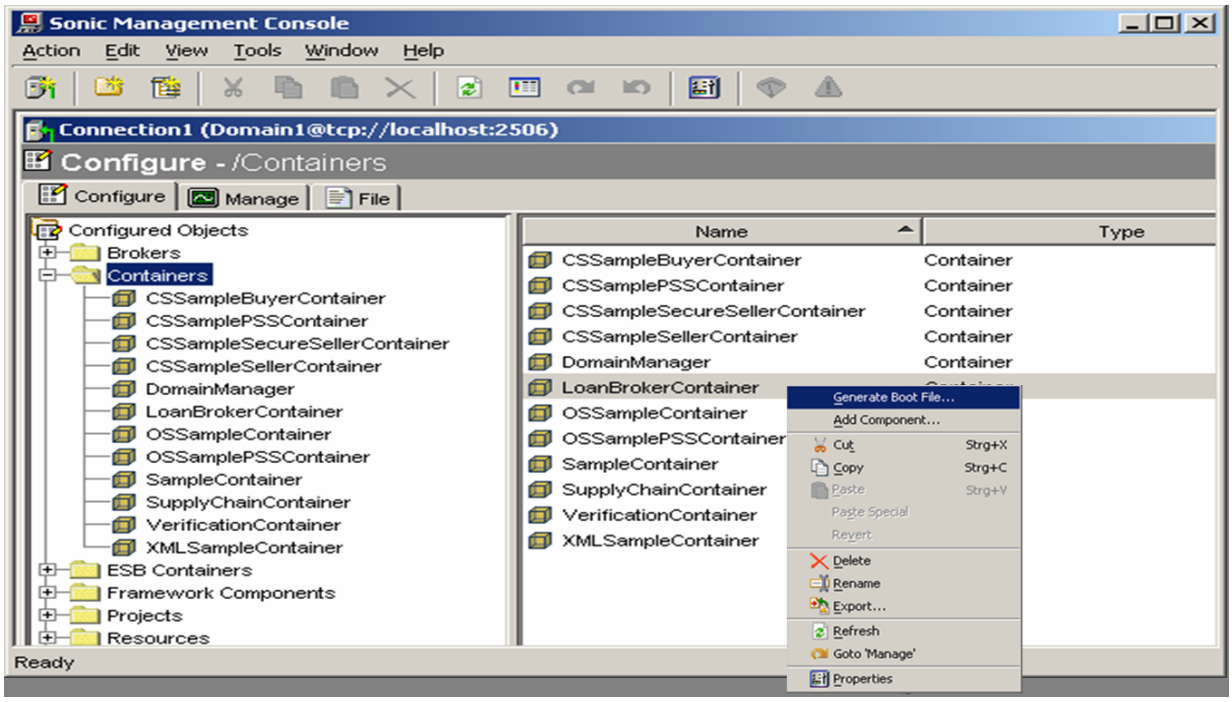

Abbildung 5.21: Sonic Management Console: Container bzw. Bootfile erstellen

Als Bootfile wird eine XML-Datei (LBcontainer.xml) generiert, welche als Parameter für “startcontainer.bat“ genommen wird. Jetzt kann der Loan-Broker-Container über eine Eingabekonsole gestartet werden.

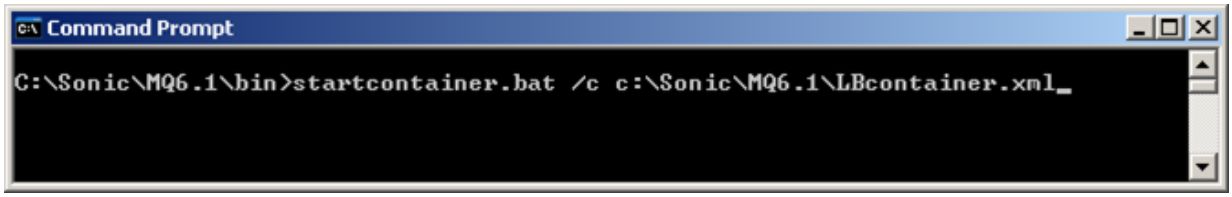

Abbildung 5.22: Starten des Loan-Broker-Containers

Nachdem der Loan-Broker-Container angelegt wurde, muss nun der ESB-Container erstellt werden, in dem die Services und Prozesse abgelegt werden. Das Erstellen eines ESB-Containers kann nur aus dem ESB-Explorer heraus geschehen. Dazu muss man im ESB-Explorer den Ordner „ESB-Containers“ wählen und mit „New“ den neuen Loan-Broker-ESB-Container anlegen.

Für das Szenario müssen folgende ESB-Prozesse (siehe S.65), Services, etc. im ESB-Explorer erstellt und dem Loan-Broker-ESB-Container hinzugefügt werden:

- ESB-Prozesse: *Loan-Broker, PartnerBank1, PartnerBank2, PartnerBank3*
  Die Prozesse für die Partnerbanken beinhalten jeweils den entsprechenden WebServiceCall für die Bank und dazugehörigen Transformationen auf die Ein- und Ausgabeparameter der WebServices. Der Prozess Loan-Broker wird als interner WebService implementiert und beinhaltet alles, was im folgenden aufgeführt wird.

- Services:

  - Content-Based Routing Services:
    ConditionBank1, ConditionBank2, ConditionResponse
    Die Services ConditionBank1 und ConditionBank2 beinhalten die Routingbedingungen (abhängig von CreditScore und HistoryLength) für die PartnerBank1 und PartnerBank2. PartnerBank3 benötigt keinen Content-Based Routing Service, da Bank3 unabhängig von CreditScore und HistoryLength einen Kredit vergibt. Der Service ConditionResponse wertet die Antworten der Banken aus, findet den günstigsten Kredit heraus und routet zu den entsprechenden Transformationen, die die XML-Nachricht auf die Ausgabeparameter des Loan-Broker-WebServices transformieren. Die Routingbedingungen sind in Javascript geschrieben und werden im Sonic Stylus Studio IE erstellt.

  - XML Transformation Services:
    1TransB3, 2TransB1, 3TransB2, ...
    Diese Transformationsservices haben die Aufgabe, die XML-Nachricht zu transformieren. Es gibt insgesamt 7 dieser Transformationen. Der Grund weshalb man so viele Transformationen benötigt, ist, dass die Nachricht entsprechend der Bankanfragen verschieden aussieht und für jeden möglichen Fall eine eigene Transformation nötig ist. Die Transformationen werden als XSLT-Mappings realisiert, die im Sonic Stylus Studio IE erstellt werden.

- Endpoints: für die Prozesse müssen jeweils ein Entry- und ein Exit-Endpoint definiert werden. Da der Loan-Broker-Prozess ein interner WebService ist, muss der Exit-Endpoint „ReplyTo“ sein. Außer den Entry- und Exit-Endpoints können noch Rejected Message Endpoints (RME) und Fault-Endpoints definiert werden, um Nachrichten, die fehlerhaft sind bzw. nicht richtig geroutet werden können, auffangen zu können. Es besteht auch die Möglichkeit, einen Event-Endpoint für einen Prozess zu definieren, um jeden Schritt des Prozesses verfolgen zu können. Services benötigen nicht unbedingt Entry- bzw. Exit-Endpoints, doch unerklärlicherweise benötigen die Transformationen 3TransB1, 3TransB2 und 3TransB3 keinen Exit-Endpoint, wohingegen die anderen vier Transformationen einen Exit-Endpoint benötigen, um die Nachricht richtig routen zu können.

Prozess erstellen

Ein ESB-Prozess ist eine Sequenz von Services, verschachtelten ESB-Prozessen und Endpoints, die einen Nachrichtenfluss ermöglichen. Zum erstellen eines ESB-Prozesses muss man den Ordner „Processes“ im ESB-Explorer wählen und mit „New“ einen neuen Prozess anlegen. Im „Process Maintenance“-Fenster können unter anderem die Endpoints ausgewählt werden. Falls ein gewünschter Endpoint noch nicht besteht, kann auch vorläufig ein Platzhalter erstellt werden. Wenn ein „Itinerary“, d.h. ein Routingverlauf (Abbildung 5.24) definiert wird, muss unter „Tracking“ der Tracking-Level und ein Event-Endpoint gewählt werden. Wenn der Prozess als interner WebService realisiert wird, muss die WSDL-URL angegeben werden und der Exit-Endpoint muss „ReplyTo“ sein.

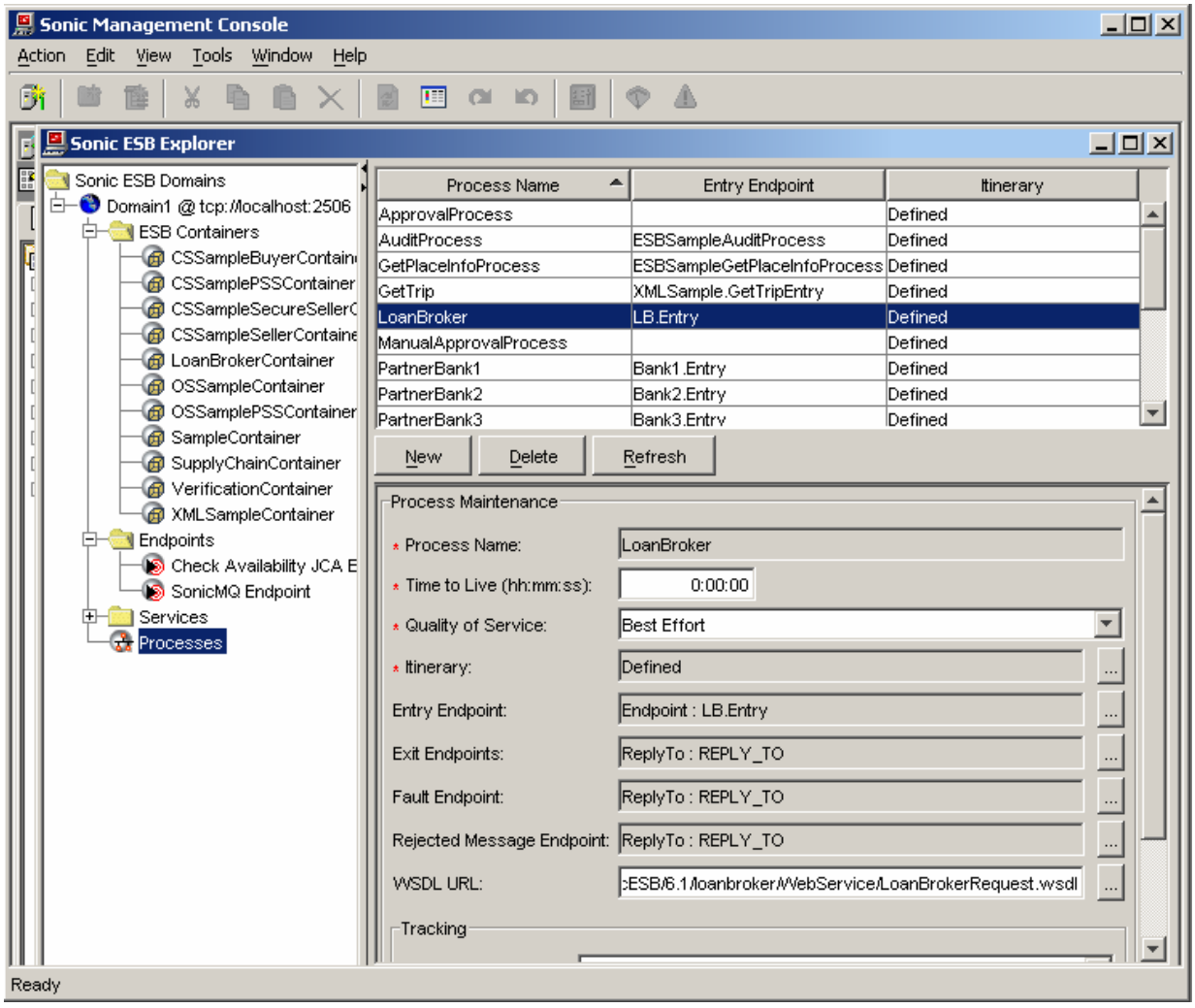

Abbildung 5.23: ESB-Explorer

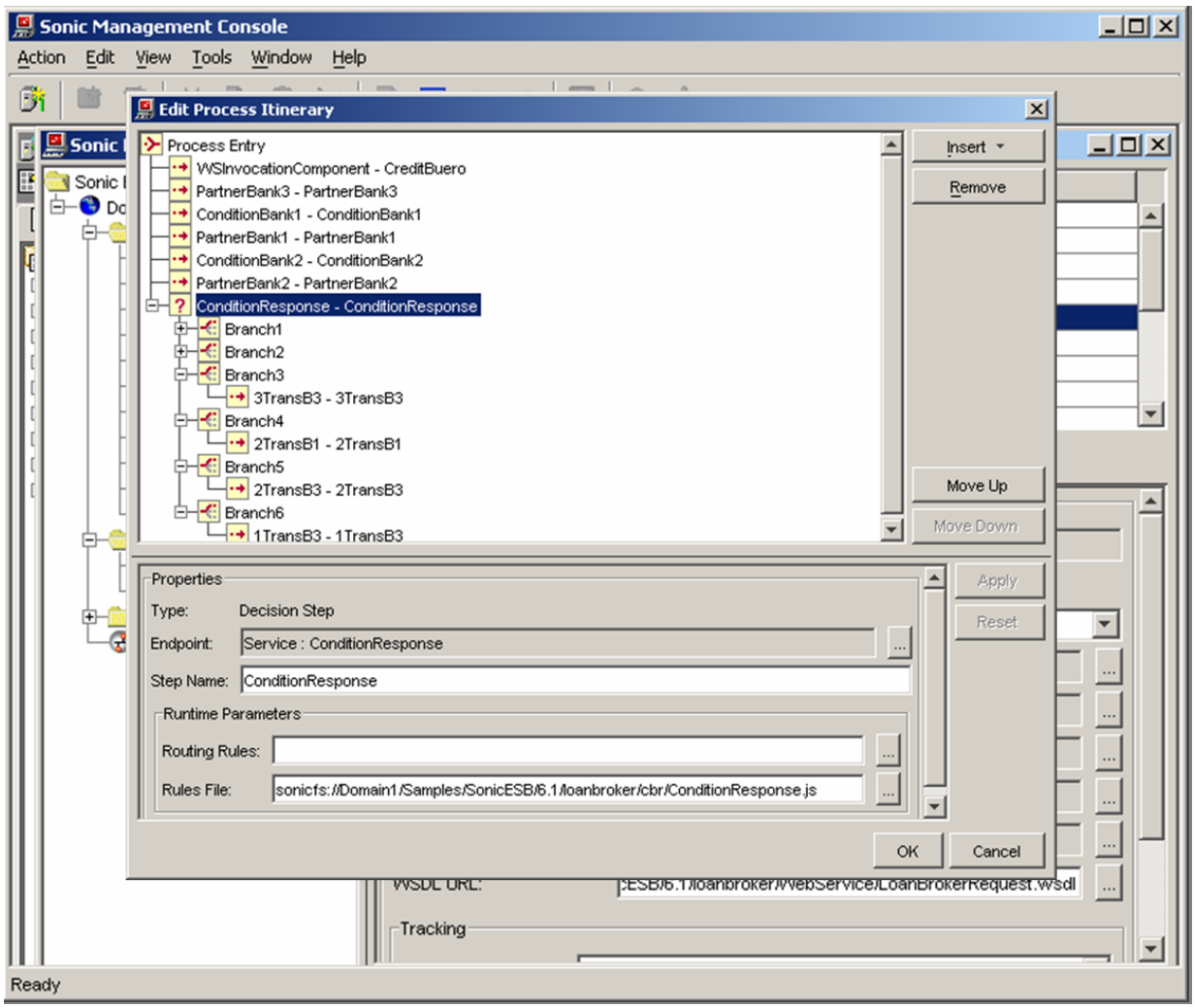

Abbildung 5.24: ESB-Explorer: Process-Itinerary

Service erstellen

Im ESB-Explorer unter dem Ordner „Services“ den gewünschten Servicetyp wählen und mit „New“ einen neuen Service anlegen. Im „Service Maintenance“-Fenster können wie schon im Kapitel „Prozess erstellen“ beschrieben die erforderlichen Einstellungen gewählt werden. Für die Transformationsservices muss hier die StyleSheet-URL angegeben werden. Man kann auch Javascript anhängen, um komplizierte Transformationen relisieren zu können. Der „Message Part Index“ gibt an, an welcher Stelle der Nachricht die Transformation erfolgt. Die XSLT-Datei für die Transformtion wird im Sonic Stylus Studio IE erstellt. Hier gibt es ein visuell unterstütztes Tool, um eine XML-Datei auf eine andere XML-Datei zu mappen (Abbildung 5.25).

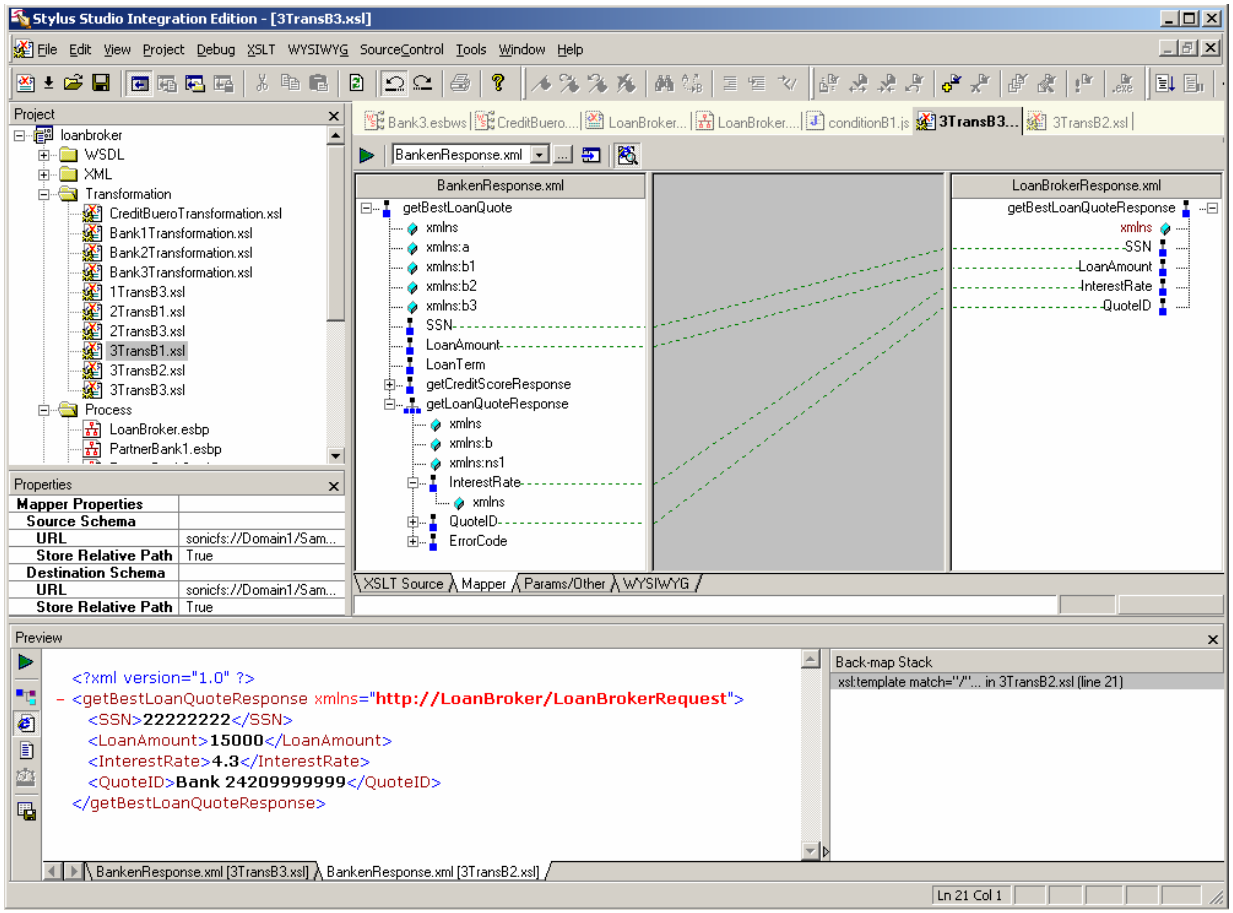

Abbildung 5.25: Sonic Stylus IE: XSLT-Mapping

Für die Routingservices muss entweder eine CBR-Datei oder eine Javascriptdatei, welche die Routingbedingungen beinhalten, angegeben werden. Diese Dateien sind im Stylus Studio IE zu erstellen, wo ein Editor für Java eingebunden ist und für die Erstellung einer CBR-Datei auch ein intuitiv bedienbares Tool vorhanden ist. In CBR's sind leider nur einfach bedingte Verzweigungen einsetzbar. Bei komplexeren Routingbedingungen muss man auf Javascript zurückgreifen. Die Javascriptdatei muss eine Funktion „rule()“ für die Routinganweisungen implementieren, wie zum Beispiel die folgende Datei (ConditionB2.js).

```
function rule()
{
var cs = XQ_getXPath("number(//CreditScore)", 0, "");
var hl = XQ_getXPath("number(//HistoryLength)", 0, "");

java.lang.System.out.println("B2::CreditScore: " + cs + "
HistoryLength: " + hl);
```

```
if(cs>=700 & hl>=10)
{
return XQ_getProcess("PartnerBank2");
}
return XQ_getService("ConditionResponse");
}
```

Endpoint erstellen

Im ESB-Explorer unter dem Ordner „Endpoints“ den Endpointtyp „SonicMQ Endpoint“ wählen und mit „New“ einen neuen Endpoint erstellen. Im „Endpoint Maintenance“ können nun verschiedene Einstellungen vorgenommen werden. Wie z.B. der Connection-Typ, QoS, etc. Es gibt die Möglichkeit, zwischen „Queue“ (Point-to-Point) und „Topic“ (Publish and Subscribe) zu wählen. Mit „New Queue“ können neue JMS Queues erstellt werden, die man dann mit dem JMS-Test-Client als Sender bzw. Reciever ansprechen kann, um den Nachrichtenfluss zu überprüfen.

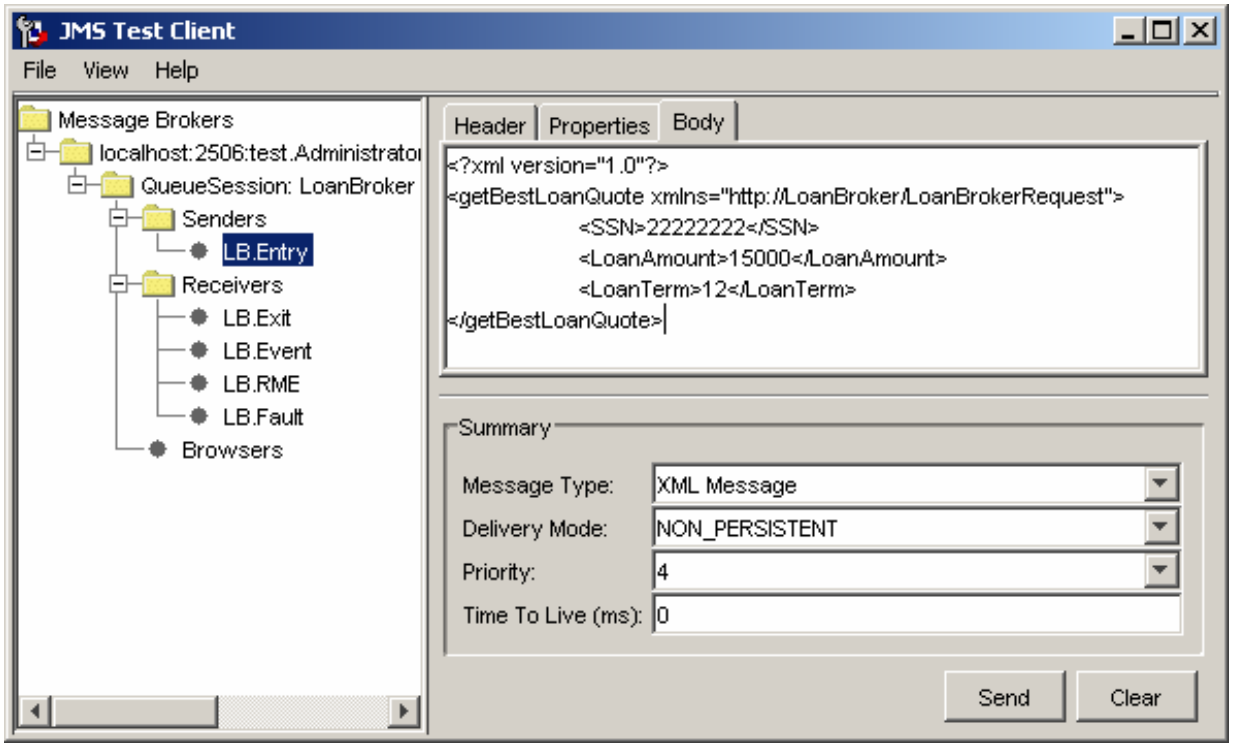

Abbildung 5.26: JMS-Test-Client

Prozess im Stylus Studio IE modellieren

Unter „File“, „New“ die Auswahl „ESB Process“ wählen. Der Designer wird geöffnet und der „Process Entry“ Knoten erscheint. An diesem Knoten können nun alle Elemente des Prozesses von der nebenstehenden Palette per Drag'n Drop angefügt werden. Im „Property“-Fenster können z.B. Endpoints, Service und interne Prozesse ausgewählt werden. Abbildung 5.27 zeigt den gesamten Loan-Broker-ESB-Prozess mit allen WebService Calls, Transformationsservices, Routingservices und internen Prozessen.

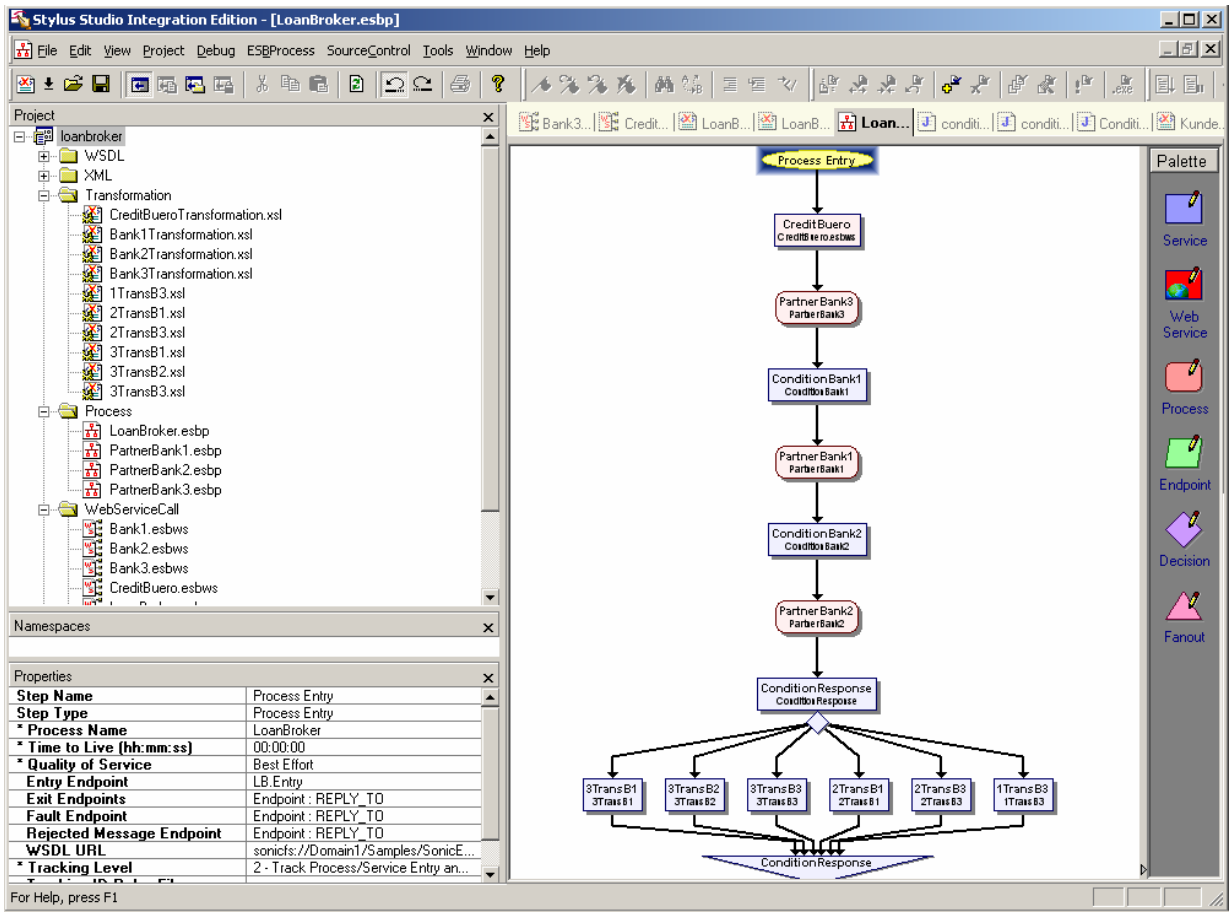

Abbildung 5.27: Sonic Stylus IE: Loan-Broker-Process

WebService-Call im Stylus Studio IE erstellen

Unter „File“, „New“ die Auswahl „Web Service Call“ wählen. Ein Tool wird geöffnet, in dem nun im Feld „WSDL“ die WSDL-Datei ausgewählt wird. Mit dem Einbinden der WSDL-Datei wird der WebService-Call automatisch generiert. Unter „Parameters“ gibt es verschiedene Möglichkeiten, die Eingangsnachricht auf die Eingangsparameter zu transformieren, bzw. die Ausgangsparameter auf die Ausgangsnachricht zu mappen. Der WebService-Call kann direkt im Stylus Studio IE getestet werden.

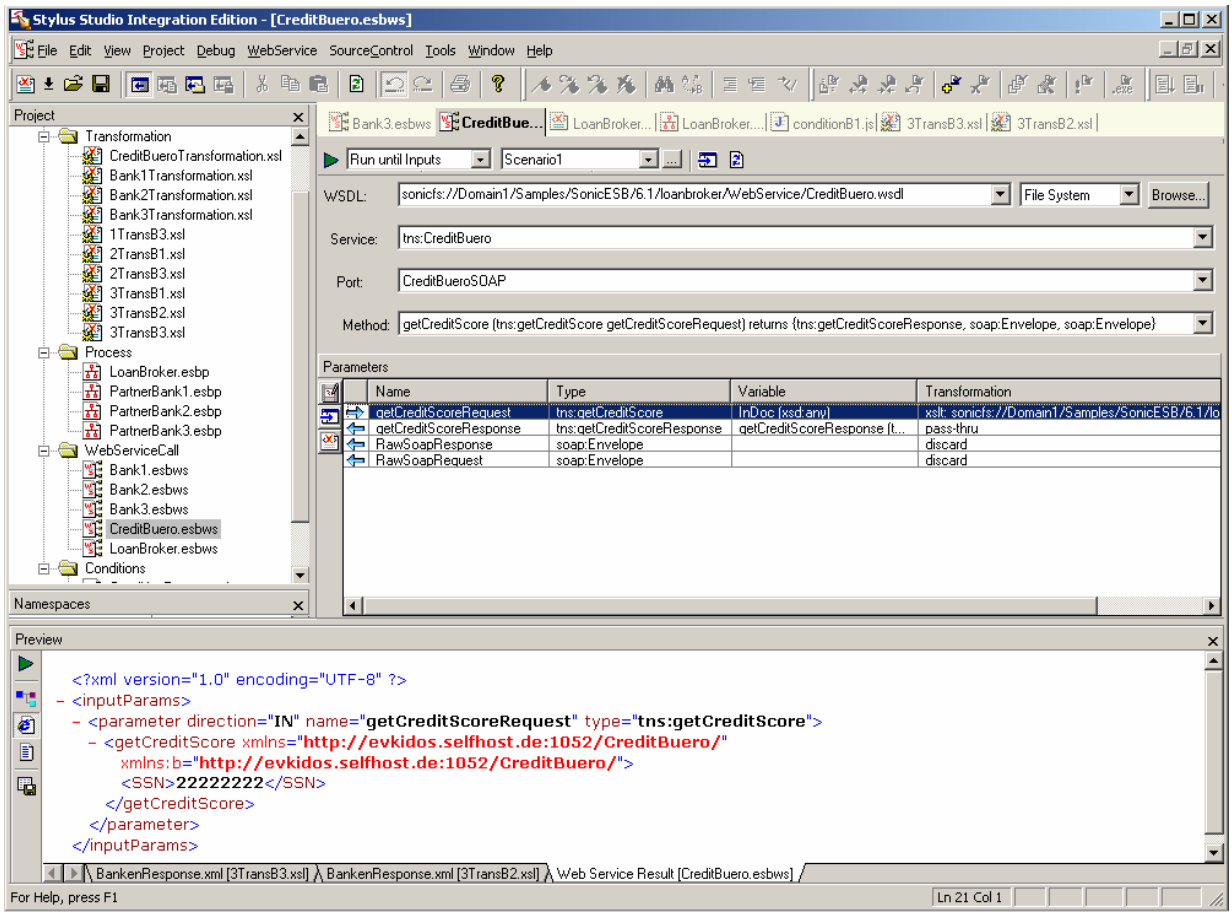

Abbildung 5.28: Sonic Stylus IE: WebService Call

5.3.3 Auswertung der Kriterien

Sonics Enterprise Service Bus erreicht *24,5 von 33 Punkten.*

Erfüllbarkeit des Loan-Broker-Szenarios

8 von 10 Punkten
Das Loan-Broker-Szenario wird mit ESB von Sonic ohne Probleme realisiert. Der Abzug von zwei Punkten ergibt sich, wie auch schon bei BEA, aus der fehlenden Möglichkeit zur Parallelisierung der Aufrufe an die Banken.

Nachrichtenverarbeitung

4 von 4 Punkten
Da der Sonic ESB auf dem SonicMQ, einem der ausgereiftesten Massaging Systeme aufbaut, sind auch die Nachrichtenverarbeitungsmöglichkeiten sehr vielfältig. Wie auch schon bei den Produkten von IBM und BEA ist die Agreggation parallelisierter Nachrichten bei Sonic nicht vorgesehen. Die Nachrichten für einen ESB-Prozess werden über den SonicMQ als Multipart-Messages gesendet. Der ESB-Prozess Entry-Endpoint wandelt die Nachricht in das Sonic ESB XQMessage-Format um und umgekehrt wird die Nachricht durch den Exit-Endpoint wieder auf das Multipart-Message-Format umgewandelt. Im XQMessage-Format, d.h. auf dem Bus, gibt es eine Vielzahl von Möglichkeiten, Nachrichten zu verarbeiten.
Näher betrachtet wurden die Routing- und Transformationsmöglichkeiten der Nachricht.

- Routing
  Kriterienerfüllung: +
  Um die Nachrichten regelbasiert an die angeschlossenen Systeme übergeben zu können oder sie an ihnen vorbeizuleiten gibt es mehrere Möglichkeiten.
  - „Decision Branches“ - Verzweigung durch Auswertung von Nachrichtenparametern mit Hilfe eines CBR-Services in einem ESB-Prozess. Hierbei kann immer nur ein Zweig verfolgt werden.
  - „Fanout Branches“ - Verteilung der Nachricht in einem ESB-Prozess an alle Zweige. Es werden alle Zweige verfolgt. Kopien der Nachricht werden dabei an alle Endpoints gesendet. Auch wenn die verarbeitete Nachricht von den verschiedenen Endpoints wieder an einen einzigen Endpoint weitergeleitet wird, hat man verschiedene Nachrichten, die nicht zusammengeführt werden können.
  - „Nested ESB Processes“ - in einem ESB-Prozess gibt es die Möglichkeit einen anderen EBS-Prozess aufzurufen, um dadurch eine verschachtelte Verarbeitung der Nachricht zu ermöglichen.
- Transformationen
  Kriterienerfüllung: +

Transformationen sind ohne Einschränkung möglich. Die Definition einer Transformation kann auf mehreren Wegen erfolgen, z.B. durch Einbindung von externen XML-Stylesheets oder durch die direkte Definition von XPath-Anweisungen. Weiterhin ist es möglich Javascript-Helper-Files einzubinden. Mit diesen Helper-Files können komplexere Transformationen oder andere Operationen (wie zum Beispiel Content-Based Routing) durchgeführt werden.

Transaktionsschutz

1 von 2 Punkten
Der Sonic ESB unterstützt nur lokale Transaktionen und keine externen Transaktionen. Sonic ESB hat zwei QoS (Quality of Service) Level. Mit dem Level „AT_LEAST_ONCE“ übernimmt der ESB-Container das Handling von „create, commit und rollback“ einer Transaktion. Mit „BEST_EFFORT“ wird nur ein Minimum an Transaktionsschutz gewährleistet, doch mit Anbindung von zusätzlichem Transaktionscode kann der Transaktionsschutz auf eigene Bedürfnisse angepasst werden.

Sicherheit

2 von 2 Punkten
Der Sonic ESB bietet umfassende Authentifizierungs-, Autorisierungs- und Verschlüsselungsmöglichkeiten über den gesamten ESB sowie flexible Möglichkeiten für den Einsatz vorhandener Unternehmenssicherheitsrichtlinien (Integrierte RSA-Verschlüsselung).

Integrationsspektrum

2 von 2 Punkten
Mit dem Paket „Adapter for ESB“ können Anwendungspakete (ERP, SCM, CRM,...), B2B-Systeme, Mainframe-Applikationen und Legacytechnologien als Services eingesetzt und integriert (z.B. SAP, Sibel, EDI, SWIFT, usw.) werden. Dieses Paket bietet mehr als 200 Adaptoren an.

Die wichtigsten Standards

0 von 2 Punkten
WS-Interoperability, WS-Security, WS-Policy, WS-Addressing und WS-ReliableMessaging werden durch den Sonic ESB 6.1 (aktuelle Version) nicht unterstützt. Diese Standards sind für den den Sonic ESB 7.0, der ab Mai 2006 verfügbar ist, angekündigt.

Skalierbarkeit

2 von 2 Punkten
Da der Sonic ESB auf dem SonicMQ aufbaut, können alle Skalierungs- und Deploy-

mentmöglichkeiten des SonicMQ benutzt werden. Der Sonic ESB ist daher nahezu „unbegrenzt“ skalierbar.

Registrierung

0 von 2 Punkten
Der Sonic ESB enthält keine Funktionalität zur Registrierung.

Benutzerfreundlichkeit

5,5 von 7 Punkten
Die Bedienung der Anwendung ist einigermaßen komfortabel. Jedoch stört das häufige Hin- und Herwechseln zwischen dem Sonic Stylus Studio IE, das recht instabil läuft, und der Sonic Management Console. Hinzu kommt, dass Änderungen im Stylus Studio nach dem Speichern von der Managenent Console immer übernommen wurde, umgekehrt jedoch das Stylus Studio die Änderungen in der Management Console nicht immer übernommen hat. Die Punkte im Einzelnen:

- Aufgabenangemessenheit
  Kriterienerfüllung: +
  Die Bedienung ist der Aufgabe, nämlich Integrationsprojekte zu entwickeln angemessen.

- Selbstbeschreibungsfähigkeit
  Kriterienerfüllung: o
  Eine Selbstbeschreibungsfähigkeit ist erst nach intensiver Einarbeitungsphase erkennbar. Eine Einarbeitung mit Handbuch, welches für jeden Produktteil vorhanden ist, ist unbedingt erforderlich. Da die Dokumentation aber kein richtiges Tutorial enthält, fällt es dem ungeschulten Anwender oft schwer, einen Durchblick zu bekommen.

- Steuerbarkeit
  Kriterienerfüllung: o
  Die Steuerbarkeit des Sonic ESB ist recht gut, wobei an mancher Stelle der Anwender intuitiv z.B. die Returntaste zum Bestätigen benutzen würde, doch dies in manchen Dialogen nicht unterstützt wird.

- Erwartungskonformität
  Kriterienerfüllung: o
  Die Ergebnisse entsprechen in der Regel den Erwartungen. Eine Ausnahme waren im Szenario die Endpoints der Transformationen, wobei einige ohne Exit-Endpoint richtig geroutet wurden und andere, die eine ähnliche Transformation an gleicher Stelle ausführenen sollten, nur mit einem explizit angegeben Exit-Endpoint richtig weiter geroutet wurden.

- Fehlertoleranz
  Kriterienerfüllung: +
  Der Sonic ESB und insbesondere SonicMQ sind robust auf fehlerhafte Eingaben. Wenn Exceptions auftraten, waren sie in der Regel auch nachvollziehbar. Sonic bietet auch zusätzliche Konsolen, um sich Fault- bzw. RME-Messages anzeigen zu lassen.

- Individualisierbarkeit
  Kriterienerfüllung: +
  Eine Individualisierbarkeit ist nur in einem kleinen Rahmen gegeben.

- Lernförderlichkeit
  Kriterienerfüllung: +
  Der Umgang mit den Möglichkeiten eines ESB verbessert sich durch die Nutzung der Anwendung.

5.3.4 Abschließende Betrachtung

Der Sonic ESB konnte in vielerlei Hinsicht überzeugen. Er ist robust und läuft stabil. Zu bemängeln gibt es drei Punkte:

- Die Aggregation parallel verarbeiteter Nachrichten: Dieser Mangel wiegt allerdings nicht allzu schwer, da er durch die serielle Verarbeitung in unserem Szenario wieder wet gemacht werden konnte.
- Probleme beim Auslesen von Nachrichtenelementen mit XPath-Anweisungen, wenn eine XSLT-Transformation zuvor stattgefunden hat. Nach der Transformation konnte zum Beispiel ein zuvor als Number ausgelesener Wert nicht mehr als Number ausgelesen werden.
- Endpoint-Routing-Probleme: Inkonsistenz bei den Exit-Endpoints für Services, d.h., dass in manchen Fällen ein Exit-Endpoint zwingend erforderlich ist, um die Nachricht richtig zu routen und in anderen Fällen nicht, wobei nicht klar wird, weshalb das so ist.

6

Zusammenfassung

6.1 Ergebnisse der Evaluierung

Die Evaluierungsergebnisse aus Kapitel sind in der folgenden Tabelle noch einmal zusammengefasst.

Kriterien:	BEA	Sonic ESB	IBM WebSphere ESB
Erfüllbarkeit Szenario	8/10 Punkte	8/10 Punkte	1/10 Punkte
Nachrichtenverarbeitung	4/4 Punkte	4/4 Punkte	4/4 Punkte
Transaktionsschutz	0/2 Punkte	1/2 Punkte	2/2 Punkte
Sicherheit	2/2 Punkte	2/2 Punkte	2/2 Punkte
Integrationsspektrum	1/2 Punkte	2/2 Punkte	2/2 Punkte
Standards	2/2 Punkte	0/2 Punkte	2/2 Punkte
Skalierbarkeit	2/2 Punkte	2/2 Punkte	2/2 Punkte
Registrierung	0/2 Punkte	0/2 Punkte	2/2 Punkte
Benutzerfreundlichkeit	6,5/7 Punkte	5,5/7 Punkte	5/7 Punkte
Summe:	25,5/33 Punkte	24,5/33 Punkte	22/33 Punkte

Welche Features die einzelnen Produkte dabei jeweils unterstützen, verdeutlicht die folgende Tabelle.

Feature	IBM Websphere	Sonic ESB	BEA Aqualogic Service Bus
Transaktions-schutz	Gegeben	nur lokaler Transaktionsschutz	nicht vorhanden
Sicherheit	WS-Security, Java 2 security LTPA, SAS, CSI, u.a.	SSL, LDAP, JAAS, u.a.	WS-Policy/WS-Security SSL, X.509, etc.
Integrations-spektrum	EJB, COM, CORBA, HTTP, JDBC, JMS, TCP/IP, u.a.	HTTP, HTTPS, JMS, COM, EDI, SWIFT, J2EE Connector, u.a.	FTP, HTTP, HTTPS, JMS SMTP, POP3, IMAP kein IIOP o. ä.
Standards	SOAP/HTTP, SOAP/JMS, WSDL 1.1, WS*, UDDI 3.0 Service Registry	WS-Standards werden erst mit der Version 7.0 unter-stützt	WS-Interoperability, WS-Security, WS-Policy, WS-Addressing, WS-ReliableMessaging
Skalierbarkeit	Gegeben	Gegeben	Gegeben
Registrierung	Gegeben	nicht vorhanden	rudimentär, Zukauf einer Registrierungs-Komponente möglich

Die beste Erfüllung der untersuchten Kriterien in der Evaluierung erreicht also der Aqualogic Service Bus von BEA mit 25,5 von 33 möglichen Punkten. Der Sonic ESB erreicht nachfolgend mit 24,5 Punkten ebenfalls eine gute Abdeckung während das Produkt WebSphere ESB von IBM mit 22 Punkten die Anforderungen zwar auch erfüllt, hier aber kleinere Abstriche in der Bedienbarkeit gemacht werden müssen.

Sowohl der Auqualogic Service Bus, als auch der Sonic ESB unterscheiden sich in den wichtigsten Kriterien Erfüllbarkeit des Szenarios, Nachrichtenverarbeitung und Benutzerfreundlichkeit kaum und erreichen hier dementsprechend auch eine ähnlich hohe Punktzahl Punktzahl. Durch die gute Abdeckung der Standards kann sich der BEA Aqualogic Service Bus aber noch vorteilhaft gegenüber dem Sonic ESB positionieren.
Der WebSphere ESB von IBM weist zwar in den Kriterien Transaktionsschutz und Registrierung Funktionalität auf, über die die beiden Konkurrenten nicht verfügen, jedoch traten im praktischen Test, also der Szenariodurchführung, mit diesem Produkt Probleme auf. Dies bedingt zum Einen die zum Zeitpunkt der Evaluierung noch schwach ausgebaute ESB-Funktionalität der Mediation-Module, mit denen das Loan-Broker-Szenario nicht modelliert werden kann. Zum anderen ergaben sich während der Arbeit mit der Entwicklungsumgebung IBM WebSphere Integration Developer häufiger Probleme, die eines Workarounds bedurften.

Auf der einen Seite konnte in der Evaluierung zwar keines der Produkte die volle Punkt-

zahl erreichen, da jeweils spezifische Schwachstellen bestehen. So verzichtet beispielsweise der Käufer des stärksten Produktes (Aqualogic Servic Bus) auf einen integrierten Transaktionsschutz beziehungsweise auf einen Satz speziellerer vorgefertigter Adaptoren (Corba, Cics, etc.). Auf der anderen Seite ließ sich aber das Szenario mit allen drei Produkten umsetzen und es entstand während der Ausarbeitung auch der Eindruck, dass die Produkte aller drei Teilnehmer flexibel genug gestaltet sind, um sie auch in komplexeren Situationen den eigenen Bedürfnissen entsprechend anpassen zu können.

Unsere abschließenende Empfehlung ist es daher mit einer Analyse der eigenen Anforderungen zu beginnen. Ist diese geschehen, kann das Produkt mit der höchsten Abdeckung der individuellen Ziele ausgewählt und realisiert werden. Die dabei noch fehlende Funktionalität wird aber im Folgenden eventuell noch durch eigene Anpassungen kompensiert werden müssen.

Literaturverzeichnis

[IBM] IBM Dokumentation: „WebSphere Enterprise Service Bus 6.0.1“. URL: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.wesb.doc/info/welcome.html [Stand: 21.03.2006].

[RED1] Keen, M. et al. (2004): „Implementing an SOA using an Enterprise Service Bus“. URL: http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg246346.html?Open [Stand: 25.07.2004]

[RED2] Nott, C. (2004): „Using Business Service Choreography In Conjunction With An Enterprise Service Bus“. URL: http://www.redbooks.ibm.com/abstracts/redp3908.html?Open [Stand: 11.10.2004]

[CHA] Chappel, Dave: Enterprise Service Bus, O'Reilly, 2004 5

[HOH] Hohpe, Gregor/ Tham, Hsue-Shen: „Enterprise Integration Patterns with Biztalk Server 2004“ - whitepaper. URL: http://www.eaipatterns.com/docs/integrationpatterns_biztalk.pdf [Stand: 21.03.2006] 13

[HOFF] Hoffmann, Britta: „Einführung in die ISO 9241-10“, 15.März 2005. URL: http://www.fit-fuer-usability.de/1x1/knigge/einfuehrung.html [Stand: 21.03.2006] 19

[BEA1] Startseite des Webauftritts von BEA. URL: http://www.bea.com [Stand: 21.03.2006] 22

[BEA2] Dokumentation des AquaLogic Service Bus von Bea. URL: http://edocs.bea.com/alsb/docs21 [Stand: 21.03.2006] 22

[SON1] Produktüberblick von Sonic Software. URL: http://www.sonicsoftware.com/products/index.ssp [Stand: 21.03.2006] 59

[SON2] Dokumentation des Sonic ESB. URL: http://www.sonicsoftware.com/products/sonic_esb/documentation/index.ssp [Stand: 21.03.2006] 59

Anhang A

Anhang

A.1 Problem der WID-generierten WSDL-Datei

Die betreffende WSDL-Datei, mit der WebSphere ESB Probleme hatte, ist die folgende:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://evkidos.selfhost.de:1052/CreditBuero/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
name="CreditBuero" targetNamespace="http://evkidos.selfhost.de:1052/CreditBuero/">
  <wsdl:types>
    <xsd:schema targetNamespace="http://evkidos.selfhost.de:1052/CreditBuero/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://evkidos.selfhost.de:1052/CreditBuero/">
      <xsd:element name="getCreditScore">
       <xsd:complexType>
       <xsd:sequence>
       <xsd:element name="SSN" type="xsd:string"></xsd:element>
       </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
      <xsd:element name="getCreditScoreResponse">
       <xsd:complexType>
       <xsd:sequence>
       <xsd:element name="SSN" type="xsd:string"></xsd:element>
       <xsd:element name="CreditScore" type="xsd:int"></xsd:element>
       <xsd:element name="HistoryLength" type="xsd:int"></xsd:element>
       </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
```

```
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="getCreditScoreResponse">
    <wsdl:part element="tns:getCreditScoreResponse" name="getCreditScoreResponse"/>
  </wsdl:message>
  <wsdl:message name="getCreditScoreRequest">
    <wsdl:part element="tns:getCreditScore" name="getCreditScoreRequest"/>
  </wsdl:message>
  <wsdl:portType name="CreditBuero">
    <wsdl:operation name="getCreditScore">
      <wsdl:input message="tns:getCreditScoreRequest"/>
      <wsdl:output message="tns:getCreditScoreResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="CreditBueroSOAP" type="tns:CreditBuero">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getCreditScore">
      <soap:operation
      soapAction="http://evkidos.selfhost.de:1052/CreditBuero/getCreditScore"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="CreditBuero">
    <wsdl:port binding="tns:CreditBueroSOAP" name="CreditBueroSOAP">
      <soap:address location="http://evkidos.selfhost.de:1052"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Es stellte sich heraus, dass die folgende Zeile das Problem war:

```
    <wsdl:part element="tns:getCreditScore" name="getCreditScoreRequest"/>
```

Nachdem hier statt `name="getCreditScoreRequest"` `name="getCreditScore"` stand, konnte der WID die WSDL-Datei verarbeiten.

A.2 Tips zum Umgang mit dem WID

Ein paar Tips an dieser Stelle sollen den Lesern dieser Fachstudie die Arbeit mit dem WID erleichtern.

- Tip für die Arbeit mit dem WID: nach jedem Schritt, und sei er noch so klein, Speichern (STRG+S) und erstellen („build Project“, STRG+B) - wird erst am Ende gespeichert und erstellt, ist die Wahrscheinlichkeit, dass etwas nicht funktioniert, dass Probleme angezeigt werden, oder, falls nicht, dass beim Ausführen „Exceptions“ auftreten, sehr hoch.

- Tip für die Arbeit mit den Testclients: Das gesamtes Modul muss gespeichert und erstellt, und zusätzlich einem Server (am besten dem Process Server) hinzugefügt und neu gestartet worden sein („Restart Project“ im Kontextmenü des betreffenden Servers). Erst dann sollte man den Testclient starten. War dies nicht der Fall und wurde das betreffende Modul vom Testclient auf einem Server eingesetzt (deployed), traten sehr oft„Exceptions“ auf. Unter Umständen reicht es auch, wenn das Modul nur gespeichert und erstellt wurde, dies allerdings ohne Garantie.

- Tip: Beim Beenden und wieder Starten des WID sollte man darauf achten, dass die javaw.exe-, java.exe- und eclipse.exe-Prozesse einer anderen WID-Instanz (insbesondere der gerade beendeten) auch tatsächlich beendet wurden. Andernfalls kann es zu Komplikationen kommen. Eine Garantie, dass diese Prozesse beim Beenden des WIDs auch wirklich beendet werden, besteht nicht!

- Das automatische Erstellen (Menüpunkt „Project“ ⇒ „Build Automatically“) sollte man ausschalten und stattdessen nach jedem Speichern von Hand erstellen (mit STRG + B). Während des Bauens sollte man warten und keine weiteren Aktionen vornehmen.

A.3 Anfragen an die Dienste

Anfrage an den Loan-Broker-Webservice:

```
POST /LoanBrokerWeb/sca/LB_MFMExport HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client
Protocol 1.1.4322.2032)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 427
Expect: 100-continue
Host: iaaswid1.informatik.uni-stuttgart.de:9080

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<getBestLoanQuote xmlns="http://Ressourcen/LoanBrokerInterface">
<SSN xmlns="">67868767</SSN>
<LoanAmount xmlns="">5000</LoanAmount>
<LoanTerm xmlns="">12</LoanTerm>
</getBestLoanQuote>
</soap:Body>
</soap:Envelope>
```

Antwort des Loan-Broker-Webservices (WebSphere ESB):

```
HTTP/1.1 200 OK
Date: Tue, 02 May 2006 14:33:40 GMT
Content-Type: text/xml; charset=utf-8
Content-Language: en-US
Server: WebSphere Application Server/6.0
Content-Length: 541

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header/>
<soapenv:Body>
<interface:getBestLoanQuoteResponse
xmlns:interface="http://Ressourcen/LoanBrokerInterface">
<SSN>67868767</SSN>
<LoanAmount>5000</LoanAmount>
<InterestRate>3.9</InterestRate>
<QuoteID>Bank 2707599999</QuoteID>
</interface:getBestLoanQuoteResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Anfrage an den Credit-Buero-Webservices:

```
POST / HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client
Protocol 1.1.4322.2032)
Content-Type: text/xml; charset=utf-8
```

```
SOAPAction: "http://evkidos.selfhost.de:1052/CreditBuero/getCreditScore"
Content-Length: 358
Expect: 100-continue
Host: evkidos.selfhost.de:1052

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
<getCreditScore xmlns="http://evkidos.selfhost.de:1052/CreditBuero/">
<SSN xmlns="">123133</SSN>
</getCreditScore>
</soap:Body>
</soap:Envelope>
```

Antwort des Credit-Buero-Webservices:

```
HTTP/1.1 200 OK
Server: Felix Billau FS-ESB-Server 1.0
Date: 02.05.2006 16:29:45
Content-Type: text/xml; charset=utf-8
Content-Length: 514
Connection: close

<?xml version="1.0" encoding="utf-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<buero:getCreditScoreResponse
xmlns:buero="http://evkidos.selfhost.de:1052/CreditBuero/">
<SSN xmlns="">67868767</SSN>
<CreditScore xmlns="">753</CreditScore>
<HistoryLength xmlns="">15</HistoryLength>
</buero:getCreditScoreResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Anfrage an einen Bank-Webservices:

```
POST /Bank3/getLoanQuote HTTP/1.1
Host: evkidos.selfhost.de:1052
```

```
Accept: application/soap+xml,multipart/related,text/*
User-Agent: IBM WebServices/1.0
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: "http://evkidos.selfhost.de:1052/Bank3/getLoanQuote"
Connection: Keep-Alive
Content-Type: text/xml; charset=utf-8
Content-Length: 526
Date: Tue, 02 May 2006 14:33:48 GMT

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header/>
<soapenv:Body>
<bank3:getLoanQuote xmlns:bank3="http://evkidos.selfhost.de:1052/Bank3/">
<SSN>67868767</SSN>
<LoanTerm>12</LoanTerm>
<LoanAmount>5000</LoanAmount>
<CreditScore>753</CreditScore>
<HistoryLength>15</HistoryLength>
</bank3:getLoanQuote>
</soapenv:Body>
</soapenv:Envelope>
```

Antwort des Bank-Webservices:

```
HTTP/1.1 200 OK
Server: Felix Billau FS-ESB-Server 1.0
Date: 02.05.2006 16:29:45
Content-Type: text/xml; charset=utf-8
Content-Length: 513
Connection: close

<?xml version="1.0" encoding="utf-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<bank3:getLoanQuoteResponse xmlns:bank3="http://evkidos.selfhost.de:1052/Bank3/">
<InterestRate xmlns="">6.4</InterestRate>
<QuoteID xmlns="">Bank 33248099999</QuoteID>
```

```
<ErrorCode xmlns="">0</ErrorCode>
</bank3:getLoanQuoteResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```