Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Projekt

Sven Stegelmeier

Eine Service Oriented Architecture mit Enterprise Service Bus für einen Ferienklub

Fakultät Technik und Informatik
Studiendepartment Informatik

Sven Stegelmeier

Thema der Ausarbeitung
Projekt

Eine Service Oriented Architecture mit Enterprise Service Bus für einen Ferienklub

Stichworte

ESB SOA MULE Muster Messaging

Kurzzusammenfassung

In der heutigen Welt, wo Software zu einem unabdingbaren Bestandteil von Unternehmen geworden ist, verlangen die vielen neuen Herausforderungen nach neuen Architekturansätzen. Eine seit wenigen Jahren bekannte Architektur ist die sog. Service Oriented Architecture (SOA), welche sich durch eine Aufteilung der Softwarelandschaft in Dienste auszeichnet. Durch dieses neue Paradigma sollen die heutigen Probleme wie z.B. die Wartbarkeit von Applikationslandschaften oder das schnelle Unterstützen von neuen Geschäftsprozessen durch Softwaresysteme gelöst werden. Neben der Neuentwicklung von Softwaresystemen ist u.a. auch die Integration von Altanwendungen ein wichtiges Ziel von SOA. Eine wichtige Voraussetzung für die Integration ist die lose Kopplung von Systemen zu größeren Gesamtsystemen. Dieser Grad der Kopplung wird durch das Versenden von Nachrichten als Kommunikationsmittel zwischen den einzelnen Systemen erreicht. Eine Infrastrukturkomponente einer SOA ist der sog. Enterprise Service Bus (ESB), welcher u.a. die Aufgabe der Steuerung des Nachrichtenflusses zwischen den einzelnen beteiligten Systemen hat. Damit ist er eine zentrale Steuerungskomponente für die Integration von einzelnen Diensten zu neuen Anwendungen. In dieser Ausarbeitung wird über die Realisierung einer SOA mit einem ESB, welcher auf dem Open-Sourceprodukt MULE basiert und der Begriff der losen Kopplung durch Nachrichtenübermittlung erläutert.

Inhaltsverzeichnis

1 Einleitung

In Unternehmen sind heutzutage eine Vielzahl von Anwendungen und Softwaresystemen im Einsatz. Diese dienen im Wesentlichen zur beschleunigten Ausführung von Geschäftsprozessen. Trotz weit entwickelter Vorgehensmodelle und Muster für die Softwareentwicklung steht ein Unternehmen im Allgemeinen vor dem großen Problem der Wartbarkeit ihrer Softwarelandschaft. Die vielen Applikationen sind meistens über die Zeit eng miteinander gekoppelt worden um eine gemeinsame Aufgabe zu verrichten. Die Schwierigkeiten beginnen, wenn diese eng gekoppelte Landschaft mit ihren vielen internen Abhängigkeiten abgeändert werden muss. Dies kann z.B. auf Grund von neuen Geschäftsprozessen oder der Änderung existierender eintreten. Da diese notwendigen Änderungen eine Menge Ressourcen verbrauchen würden, schrecken Unternehmen oft vor tiefgreifenderen Änderungen zurück und führen stattdessen Workarounds durch. Man spricht auch von einer Phase der Lähmung, da eine wirkliche Weiterentwicklung der Softwarelandschaft nicht erfolgen kann.

Die Lösung zu diesen Problemen bietet eine neue Architektur names Service Oriented Architecture (SOA). Sie hat sich eine sehr lose Kopplung und eine Aufteilung von Applikationen in Dienste zum Ziel gesetzt. Eine detaillierte Einführung und Diskussion von SOA bietet (Dirk Krafzig, 2005). Die weitere Ausarbeitung ist wie folgt strukturiert. In Abschnitt 2 wird auf die Integration von Softwaresystemen mittels Nachrichtenübermittlung (Messaging) eingegangen. Eine Middleware zur Realisierung dieser Konzepte wird in Abschnitt 3 kurz vorgestellt. Die Anwendung dieser Middleware zur Konstruktion einer SOA mit einem Enterprise Service Bus (ESB) wird in Abschnitt 4 erläutert. Diese Ausarbeitung schließt mit einer Zusammenfassung des Themas in Abschnitt 5 ab.

2 Integration von Softwaresystemen durch Messaging

Eine Service Oriented Architecture wird meistens nicht von Grund auf neu errichtet. Man darf nicht vergessen das SOA eine neue Architektur zur Lösung von derzeitigen Problemen in IT-Infrastrukturen ist. Man findet also immer eine schlecht wartbare Softwarelandschaft vor, die umstrukturiert werden muss. Neben der großen Aufgabe Altsysteme dienstfähig zu machen und neue Applikationen SOA-gerecht zu entwerfen müssen diese Systeme auch möglichst lose gekoppelt werden. Für diese Aufgabe ist u.a. der oben schon kurz erwähnte Service Bus zuständig und sie fällt allgemein in das Thema der Integration von Softwaresystemen.

Die unterschiedlichen Unternehmensaufgaben können heute schon lange nicht mehr durch ein einziges großes Softwaresystem erledigt werden. Hersteller haben sich auf unterschiedliche Kernbereiche von Funktionalitäten spezialisiert. Diese unterschiedlichen Produkte müssen nun in einem Unternehmen integriert werden. Über die Jahre sind nun verschiedene Integrationsmethoden wie z.B. Dateitransfer, gemeinsame Datenbank, entfernter Prozeduraufruf (RPC) und Nachrichtenübermittlung entwickelt worden. Der nächste Unterabschnitt widmet sich dem letzten Ansatz.

2.1 Kopplung von Softwaresystemen

Der Enterprise Service Bus ist eine optionale Infrastrukturkomponente einer Service Oriented Architecture. Durch ihn soll es möglich sein Applikationen möglichst lose zu koppeln. Das wirft die Frage auf, was überhaupt Kopplung ist und wie sie verringert werden kann. Kopplung ist im Wesentlichen eine Menge von Annahmen, die zwei Parteien über ihre Zusammenarbeit treffen müssen. Hier wurde mit Absicht der Begriff „Parteien“ verwendet, da sich Kopplung nicht nur zwischen eigenständigen Softwaresystemen ereignet.

Als Beispiel für eine sehr starke Kopplung sollen hier zwei Applikationen dienen, die Daten über ein proprietäres Anwendungsprotokoll über TCP austauschen. Es handelt sich dabei um den in Abbildung 1 dargestellten Sachverhalt.

Eine Webapplikation als Frontend nimmt Kundendaten und einen Betrag für die Überweisung von Geldern auf ein Konto entgegen und muss diese an das Backendsystem übermitteln. Zusammen sollen diese beiden Systeme zu einer Banking-Applikation integriert werden. Nachdem der Kunde die entsprechenden Daten eingegeben hat, öffnet die Webapplikation eine TCP-Verbindung zum Backend-System und schreibt die erhaltenen Daten in einen Bytestrom. Das Backendsystem ließt diesen Datenstrom aus und führt die erforderlichen Aktionen aus. Oben wurde schon gesagt, dass Kopplung eine Menge von Annahmen zwischen

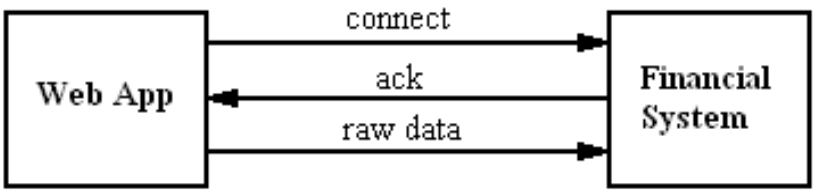

Abbildung 1: Starke Kopplung zwischen zwei Applikationen

zwei Parteien ist. Die Webapplikation und das Backend-System treffen bei dieser Integration folgende Annahmen:

- Beide Applikationen verlassen sich darauf, dass der Bytestrom einheitlich interpretiert wird. In Systemen die z.B. Big- und Little Endian Datenformate benutzen kann es hier zu Problemen kommen.
- Das Öffnen der TCP-Verbindung von der Webapplikation zum Backend-System wird über eine feste Adresse realisiert. Diese muss also dem System bekannt sein.
- Beide Systeme müssen für den Aufbau einer TCP-Verbindung verfügbar sein.
- Der Bytestrom muss genau die richtigen Informationen in der richtigen Reihenfolge enthalten.

Im Folgenden wird gezeigt, wie diese Annahmen Stück für Stück beseitigt werden können. Als erstes sollten sich beide auf ein selbstbeschreibendes, plattformunabhängiges Datenformat wie z.B. XML einigen. Als nächstes kommt eine dritte Entität, ein sog. Kanal ins Spiel. An diesen Kanal können Daten von der Webapplikation gesendet werden und von dem Backend-System wieder ausgelesen werden. Dadurch ist die Lokation der beiden Systeme nicht mehr wichtig, solange die beiden den Ort des Kanals kennen. Damit beide Systeme nicht mehr zur selben Zeit verfügbar sein müssen veranlassen wir den Kanal dazu Daten in einer Queue zu verwalten und damit zu puffern. Damit der Kanal das Wissen darüber hat, wie viele Daten er puffern und bei einem Lesevorgang wieder ausgeben muss, müssen diese Daten in einheitlichen Nachrichten verpackt werden. Zuletzt geben wir dem Kanal noch die Möglichkeit Daten von einem Format in ein anderes zu transformieren. Dadurch können die beiden Systeme unabhängig voneinander das Datenformat ändern ohne das das andere davon betroffen ist. Nur der Kanal muss davon in Kenntnis gesetzt werden.

Mit diesen Maßnahmen wurden die obigen Annahmen eines nach dem anderen außer Kraft gesetzt. Der Nachteil an der Sache ist nun der, dass diese Lösung sehr viel komplexer ist als die ursprüngliche. Dies ist der Preis für die jetzige Resistenz der Applikationen gegen Änderungen der anderen. Allerdings können diese zusätzlichen Funktionalitäten für das Versenden und Puffern von Nachrichten in einer Middleware gekapselt werden. Dadurch wird wieder ein gewisses Maß an Transparenz für den Applikationsentwickler bereitgestellt. Solch eine Message Oriented Middleware (MOM) übernimmt dann die komplette Kommunikation und sichert damit die lose Kopplung zwischen allen Teilnehmern. Ein Beispiel hierfür ist das Produkt **Mule** aus dem Open-Source Repository Codehaus, welches in Abschnitt 3 kurz vorgestellt wird. Mit Mule ist es möglich ein Bussystem für Dienste in einer Service Oriented Architecture aufzubauen.

2.2 Enterprise Service Bus

Eine Möglichkeit Applikationen in einem Unternehmen bei der Integration möglichst lose zu koppeln ist, sie alle an einen sog. Service Bus anzuschließen. Hier kann eine Analogie zu einem Hardware Bus in Computerarchitekturen gezogen werden. Der Hardware Bus verbindet u.a. den Prozessor mit dem Hauptspeicher und den Peripheriegeräten. Alle Geräte kommunizieren miteinander über einen einheitlichen Befehlssatz. Dadurch wird das Austauschen verschiedener Geräte möglich, ohne das dies Einfluss auf andere Geräte hat. Genau dasselbe Ziel hat sich der Service Bus gesetzt. Dazu muss dieses Paradigma aus der Hardwarewelt auf eine höhere Abstraktionsebene gebracht werden. Es soll also möglich sein einzelne Dienste miteinander zu verbinden und diese auch ohne Auswirkungen auf die anderen auszutauschen. In Abbildung 2 wird eine Service Oriented Architecture mit einem Service Bus, drei Diensten und zwei Applikationen gezeigt.

Die Applikationen haben selber nur die Kenntnis über das Vorhandensein des Busses, wissen aber nichts über die darunterliegende Komplexität der einzelnen Dienste. Jeder an den Bus angeschlossene Dienst hat einen Empfangs- und Antwortkanal, durch die er erreichbar ist. Da Applikationen von unterschiedlichen Herstellern kommen haben diese auch unterschiedliche Datenformate als die Applikationen. Es kann sich auch nicht auf ein einheitliches Datenmodell geeinigt werden, sobald es mehr als eine zugekaufte Applikation gibt. Die Lösung ist ein einheitliches Datenformat, nur für den Service Bus. Die eigens entwickelten Applikationen können sich diesem fügen, die hinzugekauften allerdings nicht. Deswegen wird vor diesen je ein Adapter geschaltet, der dann von dem einheitlichen Datenformat des Busses auf das applikationsspezifische transformiert. Durch diesen Ansatz wird quasi die komplette Logik für die Kommunikation wie z.B. das Wissen über den Ort des Dienstes, das Datenformat des Dienstes und auch die Verfügbarkeit des Dienstes aus den Applikationen

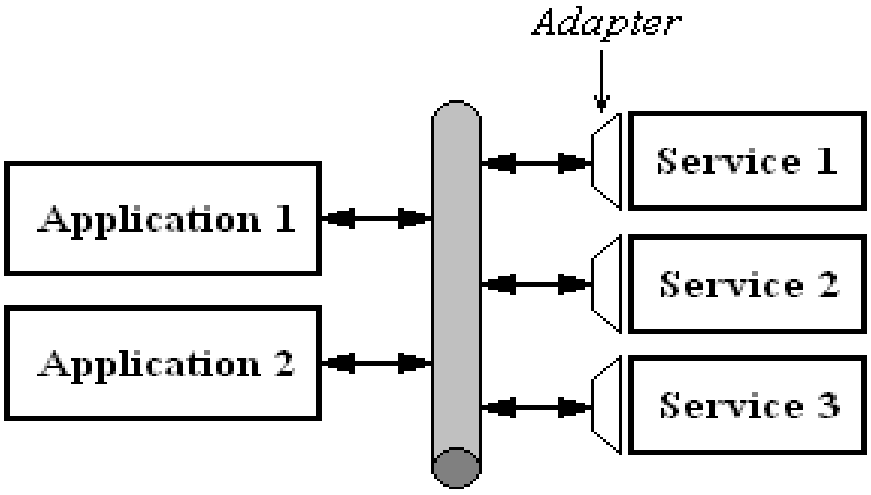

Abbildung 2: Beispiel für eine SOA mit Service Bus

herausfaktorisiert und in einen Service Bus transferiert. Dadurch werden die Applikationen maximal entkoppelt und es wird leicht diese ohne Auswirkungen auf andere auszutauschen.

Die Realisierung eines Services Busses wird oft, wie oben schon erwähnt, über eine Middleware abgewickelt. Mit dieser kann man z.B. verschiedene Nachrichten-Router aufsetzen, die dann die Nachrichten zu ihren Bestimmungsorten befördern. Hier muss dann zwischen Nachrichten, die eigentlich nur Befehle von den Applikationen zu den Diensten transportieren und den Nachrichten die Daten von den möglichen Antworten der Dienste enthalten unterschieden werden. Erstere gehören z.B. zu dem einheitlichen Befehlssatz des Busses. Dieser Befehlssatz ist allerdings sehr viel feingranularer als bei einem Hardware Bus, da er nur über simple Befehle wie z.B. „Lese von Adresse X“ verfügt. Der Befehlssatz von einem Service Bus ist mehr anwendungsspezifisch und wird wahrscheinlich mit jeder neuen Anwendung geringfügig erweitert werden müssen. Weitere Informationen zu diesem Thema und eine gute Diskussion des gesamten Integrationsthemas kann in (Gregor Hohpe, 2004) gefunden werden.

3 Message Oriented Middleware

Wie oben schon erwähnt wird die erforderliche Logik zur Nachrichtenkommunikation mit anderen Systemen oft in einer sog. Message Oriented Middleware (MOM) implementiert. Einer

dieser Ansätze ist das Produkt Mule von dem Open-Source Repository Codehaus. Im Folgenden wird die Architektur von Mule kurz beschrieben um deutlich zu machen, wie die oben beschriebenen Konzepte realisiert werden.

Mule ist im Wesentlichen eine auf der Programmiersprache Java basierende Middleware, die es Applikationen erlaubt in einer einheitlichen Weise miteinander zu kommunizieren. Die Architektur besteht aus mehreren Mule-Instanzen, die jeweils mehrere Objekte verwalten und miteinander kommunizieren können. Diese Objekte werden als Universal Message Objects (UMOs) bezeichnet und implementieren die Geschäftslogik. Die UMOs sind einfache Java-Bean Komponenten, d.h. Mule ist in der Lage die gesamte Kommunikation vor dem Applikationsprogrammierer zu verstecken. Allerdings wird, wie oben schon erwähnt, die Geschäftslogik zum größten Teil nicht neu entwickelt, sonders steht schon in Form von Altsystemen zur Verfügung. Auch diese können mit Mule eingebunden werden. In Abschnitt 2.2 wurde schon ein sog. Adapter für die Transformation von Datenformaten erwähnt. Diese Adapter werden in Mule über sog. Endpoints bereitgestellt. Diese Endpoints ermöglichen u.a. auch die Kommunikation mit Applikationen, die eigentlich nicht für die Kommunikation mit einer MOM ausgelegt sind. Wenn eine Applikation als Web-Service verfügbar gemacht worden ist, diese aber an einen Service Bus, der auf dem Java Message Service (JMS) basiert, angeschlossen ist bietet Mule hier einen expliziten Endpoint für die Kommunikation mit Web-Services an. Es ist sogar möglich selbst Endpoints für proprietäre Anwendungsprotokolle zu schreiben. Abbildung 3 zeigt die Grobarchitektur einer MULE-Instanz.

Zentral für jede Mule-Instanz sind der Mule-Manager und das Mule-Model. Letzteres verwaltet die unterschiedlichen UMOs, indem es den Nachrichtenfluss steuert und Operationen für den Lebeszyklus von Komponenten aufruft, falls diese vorhanden sind. Der Mule-Manager verwaltet die Komponenten, die zwischen die UMOs geschaltet sind um den Nachrichtenfluss zu manipulieren. Solche Komponenten sind z.B. Konnektoren, Router oder Transformatoren. In Abbildung 4 wird eine simple Ende-zu-Ende Topologie gezeigt, die die Verwendung der eben genannten Komponenten verdeutlichen soll.

Eine typische Kommunikation läuft über Mule wie folgt ab. Eine Applikation sendet Daten über einen Kanal, auf dem ein Message Receiver lauscht. Dieser Receiver übergibt die empfangenen Daten an einen Connector der den Receiver verwaltet. Der Connector kapselt das Wissen über verschiedene Kommunkationsarten. Z.B. ist der Connector für Http in der Lage Http-Anfragen zu versenden und auch zu verstehen. Er verbirgt die Heterogenität von externen Anwendungen. Die Daten gelangen danach an einen Transformator, der die empfangenen Daten z.B. auf ein internes Datenformat für einen Service Bus transformieren kann. Danach sucht Mule nach möglichen UMOs, die für diese Art von Daten in Betracht kommen. Dies geschieht über Eingangsrouter. Eine UMO, die Geschäftslogik implementiert erhält nun die einheitlich formatierten Daten und kann z.B. Prüfungen ausführen. Danach

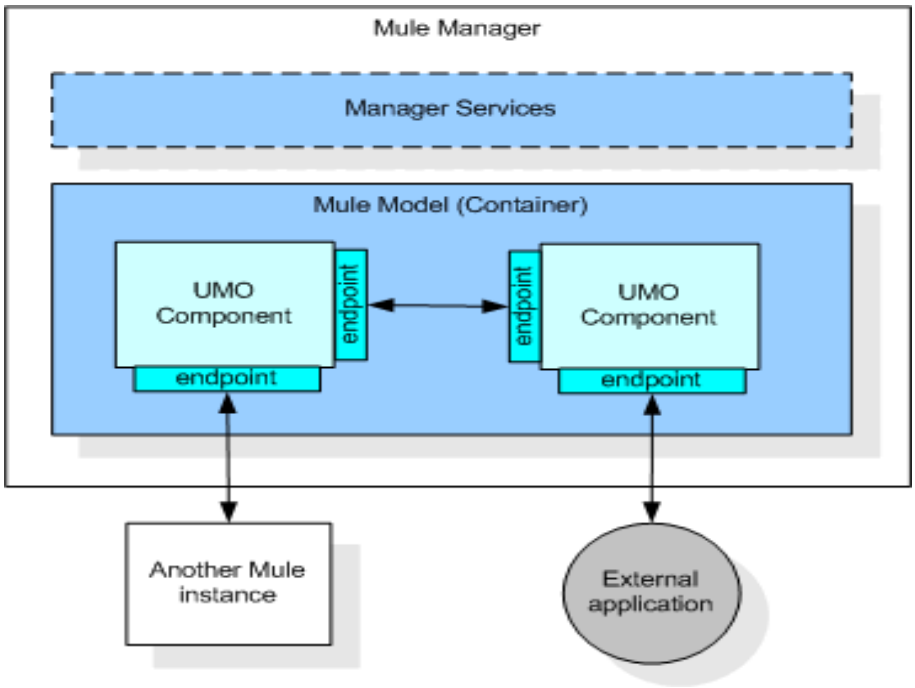

Abbildung 3: Architektur einer Mule-Instanz

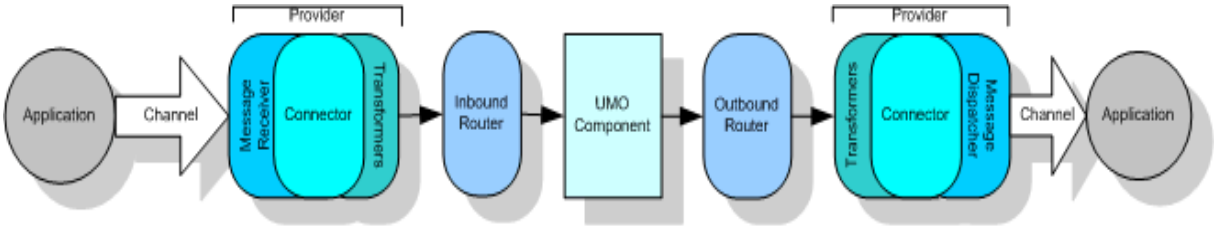

Abbildung 4: Simple Ende-zu-Ende Topologie

sendet er diese die Daten über einen Ausgangsrouter an dieselbe oder eine andere Applikation weiter. Dabei muss der Applikationsprogrammierer kein explizites Ziel beim Senden der Daten angeben. Er gibt einfach nur an welche Daten er versenden will und Mule kümmert sich um den Rest. Dazu braucht Mule gewisse Angaben über den Nachrichtenfluss und den verwendeten Transportmechanismen vom Benutzer in einer Konfigurationsdatei, die für die Instanz gültig ist. Weitere Informationen über Mule finden sich in (mule, 2005).

4 Eine Service Oriented Architecture für einen Ferienklub

Die oben beschriebenen Konzepte und Technologien wurden im Rahmen eines Projektes im Masterstudiengang INF-M3 an der Hochschule für Angewandte Wissenschaften Hamburg angewendet um eine kleine Applikationslandschaft aufzubauen. Das Anwendungsgebiet für dieses Projekt war ein fiktiver Ferienklub. Dieser diente für das Projekt im Wesentlichen als Metapher oder „Spielwiese“ für die Erprobung der oben beschriebenen Konzepte und Technologien.

Der Ferienklub beherbergt eine Reihe von Softwaresystemen zur Unterstützung der internen Geschäftsprozesse.

- Er bietet eine Web-Services Schnittstelle für Reisebürounternehmen an. Diese sind dadurch in der Lage die Anzahl von noch freien Zimmern abzufragen und ggf. Zimmer für Kunden zu buchen.
- Der Ferienklub hat intern eine Verwaltungsapplikation, welche die internen Datenbestände des Ferienklubs verwaltet.
- Ein System zur Verwaltung von Kundendaten.
- Ein System zur Verwaltung von freien und verbuchten Zimmern.
- Ein System zum Einholen und Buchen von Veranstaltungen.
- Ein System zum Bestellen von Getränken an der Strandbar des Ferienklubs.

Die Anwendungsfelder der Softwarelandschaft können daher in die vier Felder Buchen von Urlauben, Verwaltung von Gästen, Buchen von Freizeitangeboten und Bestellen von Getränken an der Strandbar eingeteilt werden. Um diese Aufgaben zu erledigen müssen die oben aufgezählten Systeme integriert werden. Dies wurde mit der oben beschriebenen Middlewarelösung Mule erledigt. Durch die Integration ist ein Service Bus und ein sog. Process Manager entstanden. Ein Process Manager ist eine zentrale Komponente, die den Nachrichtenfluss überwacht. Er wird gewöhnlich dann eingesetzt, wenn große Datenmengen im Inhalt einer Nachricht viele Dienste durchläuft, die mit diesen Daten nichts anfangen können. Der Process Manager speichert deswegen diese Daten lokal und fügt sie erst dann in die Nachricht ein, wenn der entsprechende Dienst aufgerufen wird.

Der Process Manager und der Service Bus sind beides Musterlösungen für Integrationsprobleme. Das Projekt hätte auch vollständig mit einem Service Bus realisiert werden können.

Der Grund für die Nutzung des Process Managers ist, dass er aus technischer Sicht einfacher zu realisieren war als der Service Bus und deswegen als Startansatz, um mit der Middleware vertraut zu werden, gewählt wurde. Abbildung 5 zeigt die Architektur des Ferienklubs.

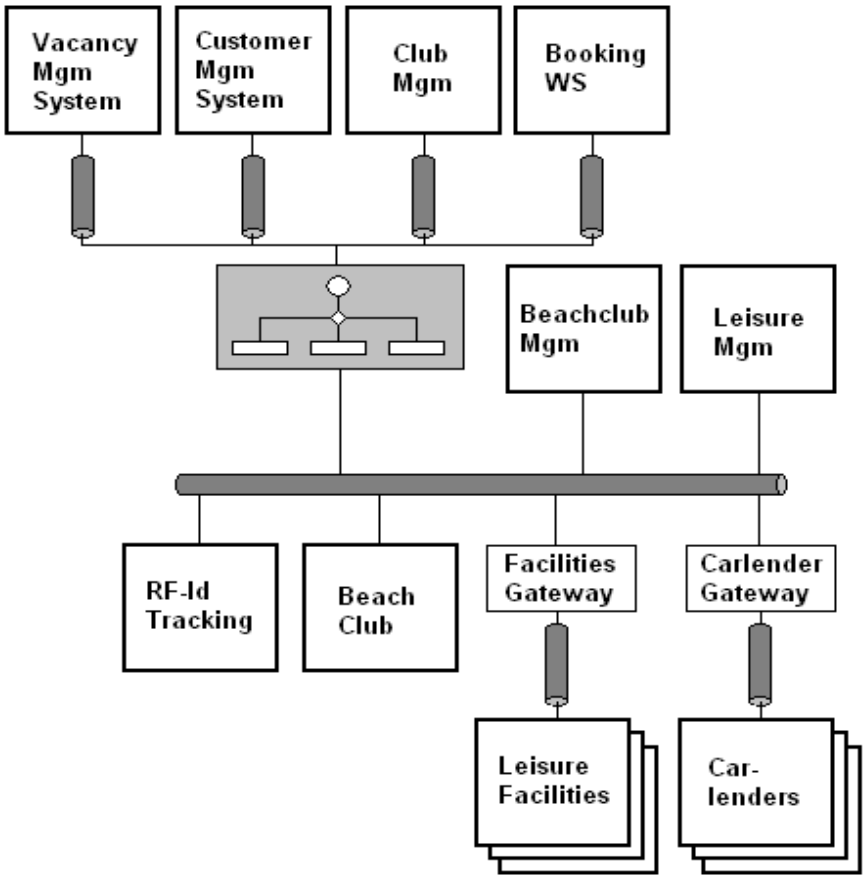

Abbildung 5: Service Orientierte Architektur des Ferienklubs

Die Applikationen für die Verwaltung von Kunden und Zimmern sind zusammen mit ihren zugehörigen Backend-Systemen an den Process Manager angeschlossen. Dieser steht wiederum mit dem Service Bus in Verbindung. Des Weiteren sind die beiden Applikationen zum Buchen von Freizeitangeboten und zur Bestellung von Getränken an der Strandbar zusammen mit ihren zugehörigen Backend-Systemen an den Service Bus angeschlossen. Der Ferienklub muss mit externen Unternehmen, in diesem Fall Unternehmen für Freizeitangebote und zur Vermietung von Autos, kommunizieren. Diese Systeme stehen nicht unter der Kontrolle des Ferienklubs und müssen daher über das Internet angesprochen werden. In unserem Fall bieten die Unternehmen jeweils einen Webservice an. Da die Anzahl an

Unternehmen, die wir ansprechen wollen variabel ist und der Service Bus und die Applikationen davon kein Wissen haben sollen, wurde dieses in Gatewaykomponenten gekapselt. Diese wissen wie viele externe Unternehmen z.B. für das Einholen von Freizeitangeboten angesprochen werden müssen.

Im Folgenden werden die komplexeren Prozesse, d.h. Prozesse mit mehr als einem beteiligten Service, aufgezählt und beschrieben.

- Hat ein Kunde des Ferienklubs sich ein Zimmer und den Zeitraum seines Urlaubs über ein Reisebüro ausgesucht, so kann er nun eine Buchung machen. Das Reisebüro ruft nun die Komponente „Booking WS“, welche einen Public Enterprise Service darstellt, auf. Im Ferienklub müssen nun folgende Dinge geschehen:
  - Es muss ein neuer Kunde bzw. neue Kunden (die Familie des Kunden) angelegt werden. Allerdings kann diese schon einmal einen Urlaub hier verbracht haben. Daher müssen nur für die neuen Kunden Einträge in der Datenbank gemacht werden. Dazu kommuniziert der Process Manager mit dem Customer Mgm System.
  - Die gebuchten Zimmer müssen für die Dauer des Urlaubs gesperrt werden, damit sie nicht überbucht werden können. Dazu wird das Vacancy Mgm System vom Process Manager angesprochen.
- Hat ein Kunde seinen Urlaub angetreten will er u.a. Freizeitangebote einholen. Zuvor teilt er dem Ferienklub seine Interessen mit, welche intern über ein Interessenprofil verwaltet werden. Will er also Angebote für seine Freizeit einholen, müssen im Ferienklub folgende Dinge geschehen:
  - Die Applikation Leisure Mgm sendet eine Kommandonachricht an den Service Bus.
  - Der Service Bus ruft zuerst den Processmanager auf um das Interessenprofil des Kunden auszulesen und es in die Nachricht einzufügen.
  - Die Nachricht wird danach an das Facilities Gateway gesendet, welches sie an alle bekannten externen Unternehmen sendet.
  - Die Unternehmen bekommen die Kommandonachricht mit dem Interessenprofil und fügen koherente Freizeitangebote in die Nachricht ein. Anschließend wird die Nachricht mit den Angeboten wieder an das Gateway zurückgesendet, welches alle Angebote zusammenfast und in die ursprüngliche Nachricht einfügt.

  - Der Service Bus sendet die Nachricht als nächstes an den Process Manager, welcher Kontakt mit dem Vacancy Mgm System aufnimmt. Es überprüft wie viele Personen mit dem Kunden in den Urlaub gefahren sind und filtert Freizeitangebote für weniger Personen aus der Nachricht heraus.
  - Als letztes sendet der Service Bus die Nachricht zurück an die Leisure Mgm Applikation, welche dem Kunden die benötigten Freizeitangebote präsentieren kann.

- Hat der Kunde sich ein Freizeitangebot ausgesucht, so möchte er dieses Buchen. Um die Buchung zu realisieren müssen im Ferienklub folgende Dinge geschehen:
  - Die Applikation Leisure Mgm erstellt eine Kommandonachricht und sendet diese an den Service Bus.
  - Der Service Bus sendet diese Nachricht an das Facilities Gateway, welche das erforderliche Unternehmen von der Buchung in Kenntnis setzt.
  - Hat der Kunde ein Auto angefordert um zu der Veranstaltung zu gelangen, so sendet der Service Bus die Nachricht zum Carlender Gateway, welches mit einer Autovermietung Kontakt aufnimmt und ein Auto für den entsprechenden Zeitraum mietet.
  - Für jeden Kunden wird im Ferienklub eine Historie seiner unternommenen Aktionen geführt. Deswegen sendet der Service Bus als letztes die Nachricht an den Process Manager. Dieser kommuniziert mit dem Customer Mgm System und fügt einen neuen Historyeintrag hinzu.

- Befindet sich ein Kunde am Strand und möchte mittels eines PDAs ein Getränk von der Strandbar bestellen, müssen im Ferienklub folgende Aktionen geschehen:
  - Die Beachclub Mgm Applikation erstellt eine Kommandonachricht und sendet diese an den Service Bus.
  - Der Service Bus sendet die Nachricht an das RF-Id Tracking System um die momentane Position des Gastes zu erfahren. Diese wird dann in die Nachricht eingefügt.
  - Als nächstes gelangt die Nachricht über den Bus bei der Strandbar, die daraufhin das Getränk zum Kunden bringen kann.

5 Zusammenfassung

In dieser Ausarbeitung wurde die Realisierung von Integrationslösungen mittels eines Enterprise Service Busses erläutert. Die Integration von Softwaresystemen ist in der heutigen Welt der Unternehmen eine absolute Notwendigkeit. Systeme die alle Unternehmensfunktionen realisieren können gibt es nicht, was zu einer Aufteilung der benötigten Funktionalität auf verschiedene Systeme führte. Folglich müssen diese Systeme zusammenarbeiten um eine Gesamtaufgabe zu verrichten. Bei der Zusammenarbeit kommt es aus Wartbarkeitsgründen auf eine möglichst lose Kopplung der Systeme an. Kopplung ist im Wesentlichen die Menge von Annahmen, die zwei Parteien über ihre Zusammenarbeit treffen. Sie kann durch eine nachrichtenorientierte Kommunikation minimiert werden. Eine gute Architektur für die lose Kopplung von Softwaresystemen ist die Service Oriented Architecture zusammen mit einem Enterprise Service Bus. An letzteren können Dienste sehr einfach angeschlossen oder ausgetauscht werden ohne das dies Auswirkungen auf andere angeschlossene Dienste hat. Damit ist der Service Bus analog zu einem Hardwarebus in einem Computer zu sehen.

Die Konzepte von SOA und dem Enterprise Service Bus wurden in einem Projekt Ferienklub angewendet. Es wurde eine kleine Softwarelandschaft für die Verwaltung von Kunden, das Buchen von Urlauben und Freizeitangeboten und dem Bestellen von Getränken geschaffen und diese über den Service Bus integriert. Sämtliche Kommunikation wurde vor der Geschäftslogik durch ein Middlewareprodukt namens Mule versteckt.

Literatur

[mule 2005] : *http://mule.codehaus.org*. 2005

[Dirk Krafzig 2005] DIRK KRAFZIG, Dirk S.: *Enterprise SOA Service-Oriented Architecture Best Practices*. 2005

[Gregor Hohpe 2004] GREGOR HOHPE, Bobby W.: *Enterprise Integration Patterns*. 2004