On Demand Business Lösungen
White Paper

WebSphere. software

# Gekoppelte (synchrone) oder nicht gekoppelte (asynchrone) Kommunikation und einfache oder komplexe Messaging-Funktionen in Zusammenhang mit einem Enterprise Service Bus.

*Von Andy Stanford-Clark, leitender Entwickler und langjähriger technischer Mitarbeiter, IBM Hursley Park Laboratory.*

Januar 2005

**Inhalt**



## Einführung

Im vorliegenden White Paper werden Überlegungen zur gekoppelten und entkoppelten Kommunikation sowie zu einfachen und komplexen Messaging-Funktionen in einer ESB-Umgebung (Enterprise Service Bus) angestellt. Nachdem das Konzept und die praktische Umsetzung eines ESB beschrieben wurde, wird erläutert, welche Aspekte bei der Entscheidung für eng oder lose gekoppelte Kommunikation zu beachten sind. An einen Vergleich zwischen einfachem und komplexem Messaging schließen sich detaillierte Betrachtungen zu einfacherem und einfachem Messaging an.

## Was ist ein ESB?

Ein ESB ist ein architekturelles Muster, mithilfe dessen die Verteilung von Informationen zwischen verschiedenen Arten von Anwendungen über mehrere Standorte hinweg optimiert werden kann. Das ESB-Muster basiert auf nachrichtenorientierten, ereignisgesteuerten und serviceorientierten Integrationskonzepten und vereinheitlicht sie. Die Kennmerkmale eines ESB (die alle auf eine serviceorientierte Infrastruktur ausgerichtet sein sollten) bieten folgendes:

- *auf Standards basierende Anwendungsintegration*
- *Unterstützung für Webservices, nachrichtenorientierten Transport sowie (ereignisgesteuerte) „Publish-and-subscribe"-Integration*
- *Konvertierung*
- *Intelligente Weiterleitung*

Ein ESB (siehe Abb. 1) stellt einen logischen Mechanismus bereit, der eine flexible und anpassbare Interaktion zwischen separaten Informationsautomationskomponenten ermöglicht, die bisher nicht ohne größeren finanziellen Aufwand möglich war. Diese Komponenten wurden bislang oft in einer Form bereitgestellt, die sich als unflexibel erwies, wenn weitere Änderungen, Verbesserungen oder Verbindungen erforderlich wurden (um die ursprünglichen Kosten niedrig zu halten).

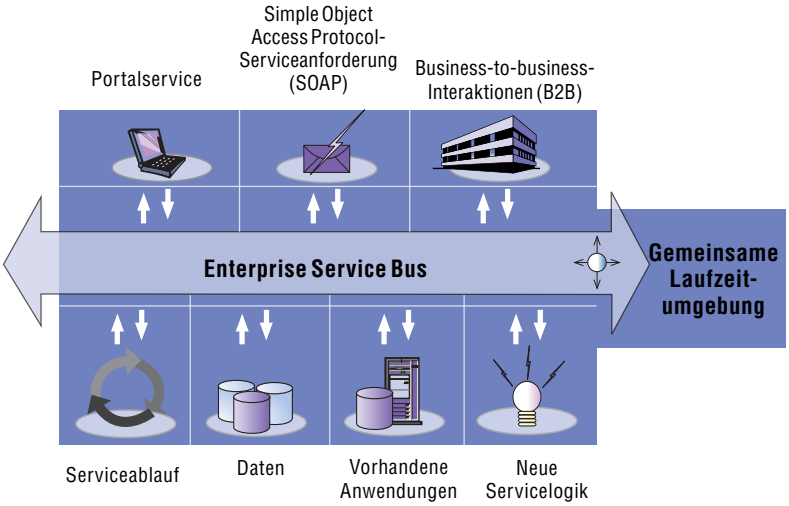


*Abbildung 1. Konzeptionelle Sicht eines ESB.*

Wenn diese Erklärung etwas abstrakt erscheint – und das ESB-Konzept ist nicht ganz einfach zu verstehen –, hilft uns vielleicht eine Analogie weiter. Stellen Sie sich einen Doppeldeckerbus vor. Sein Zweck besteht darin, Menschen von einem Ort zum anderen zu transportieren (informationstechnisch ausgedrückt hieße das, Informationen von einem Ort zum anderen zu übertragen). Aber während ein Doppeldeckerbus nur einen Eingang besitzt, kann ein ESB verschiedene Türen – mit verschiedenen Ein- und Ausgängen – aufweisen. Die Türen könnten unterschiedliche Formen und sogar unterschiedlich große Aufgänge haben – um Menschen (Informationsarten) unterschiedlicher Größe und unterschiedlichen Arten des Ein- und Ausstiegs gerecht zu werden.

Diese Aufgänge sind ebenso wichtig wie die Türen selbst. An diesen Aufgängen würden Sie gefragt, welche Sprache Sie vorziehen (beispielsweise „Welches Protokoll verwenden Sie?“). Wenn das geklärt wurde, wandelt jeder Aufgang alles, was in den Bus einsteigt, in eine gemeinsame Form um. Dasselbe gilt für den Ausstieg. Das heißt, etwas, das über einen Aufgang und durch eine Tür ankommt, kann den Bus durch jede andere Tür und über jeden anderen Aufgang in einer anderen Form wieder verlassen. Was immer in den Bus einsteigt, kann also auch wieder aussteigen – auch wenn das Ausgangsformat und -protokoll ein völlig anderes ist als das Einstiegsformat und -protokoll. Ein ESB gibt einer Anwendung, die eine bestimmte Form der Ausgabe produziert, die Möglichkeit, diese Ausgabe zu übergeben – und dabei zu wissen, dass andere Anwendungen diese Ausgabe empfangen (und nutzen) können, ohne dass die ursprüngliche Anwendung die Zielanwendung verstehen muss.

Aber damit sind die Möglichkeiten eines ESB noch nicht erschöpft. Stellen wir uns wieder den Doppeldeckerbus vor. Er hat zwei Etagen. Auf der unteren Etage befinden sich alle Türen und Aufgänge. Hier werden der Einstieg, die Umsetzung eintreffender Informationen in ein gemeinsames Format und der nachfolgende Ausstieg verwaltet. Aber möglicherweise ist mehr erforderlich. Durch einen geringen Aufpreis und den Aufstieg zum Oberdeck sind zusätzliche Verarbeitungsschritte erhältlich – von Permanenz, Zuverlässigkeit, Konvertierung und Weiterleitung bis hin zu Verschlüsselung, Regeln, inhaltsbezogener Verarbeitung und weiteren zusätzlichen Dienstleistungsangeboten. Vielleicht wollen Sie nicht all diese Services ständig in Anspruch nehmen. Aber bestimmte Passagiere (oder ihre Kunden) möchten sie vielleicht bei Bedarf nutzen.

*Ein ESB in der Praxis*

Idealerweise ist ein ESB eine Softwareanwendung mittlerer Größe, die in mehreren Exemplaren innerhalb eines Unternehmens oder sogar unternehmensübergreifend implementiert werden kann. Die einzelnen ESB-Exemplare sind gewissermaßen Leichtgewichte, besitzen aber die Fähigkeit, sich selbst zu organisieren, sodass zwei oder mehr ESBs zusammen einen logischen Verbindungsbus bilden können, durch den nach Bedarf Informationen fließen können.

Im Gegensatz zu einem Doppeldeckerbus, der sich (sofern es das Verkehrsaufkommen erlaubt) zu festen Zeiten auf festgelegten Routen bewegt, steigt der Wert eines ESB mit wachsender Flexibilität. Da (idealerweise) an jedem für eine Anwendung (die mit einer anderen kommunizieren muss) geeigneten Punkt ein ESB-Exemplar installiert werden kann, bietet die Kombination sich selbst organisierender ESB-Exemplare die Möglichkeit, alles, was an einem ESB-Exemplar zusteigt, durch ein anderes ESB-Exemplar zuzustellen. Die zugrunde liegenden Kommunikationsverbindungen zwischen allen ESB-Exemplaren werden durch ein lokales Netz (LAN), ein Weitverkehrsnetz (WAN) oder Kombinationen dieser beiden Netzarten bereitgestellt. Sie können auch die Größe und Position verschiedener ESB-Exemplare auf spezifische Anforderungen – bezogen auf berufliche Aufgaben, Abteilungen, das gesamte Unternehmen oder Kommunikation zwischen verschiedenen Unternehmen – ausrichten.

Die Hauptfunktion eines ESB ist daher das Verbinden mehrerer ESB-Exemplare. Ein ESB erbringt seine Leistung, indem er innerhalb einer Netzwerkwolke arbeitet, die die Verbindungen zwischen Anwendungen und Systemen herstellt. Früher funktionierte diese Art der Kommunikation oft nur mit teueren und starren Punkt-zu-Punkt-Verbindungen, oder die Anwendungen blieben voneinander isoliert – und Menschen fungierten als „Middleware“ (Verbindungsmechanismus) zwischen Anwendungen. Der besondere Vorteil von ESBs besteht darin, dass sie innerhalb eines logischen Netzwerks automatisierte Ein- und Ausgänge bereitstellen können.

Die Vorteile des ESB-Konzepts lassen sich auch noch auf andere Weise verdeutlichen, nämlich wenn es darum geht, die zusätzlichen Dienstleistungen des Oberdecks in Anspruch zu nehmen. Es ist nicht erforderlich, jede Konvertierungs- oder Weiterleitungsanweisung auf jedem ESB-Exemplar zu positionieren. Stattdessen können umfangreiche Zusatzleistungen an vielen Stellen repliziert werden, um Leistung oder Durchsatz zu optimieren, während andere Zusatzleistungen möglicherweise nur an ein oder zwei Stellen vorhanden sind. Der Zugang zu diesen Stellen erfolgt über die sich selbst organisierenden, miteinander kommunizierenden ESB-Exemplare.

Daher besteht der Vorteil eines ESB darin, dass die Positionen aller ESB-Exemplare innerhalb einer auf spezifische Anforderungen zugeschnittenen Topologie angeordnet werden können. Dies kann nach Bedarf auf der Ebene beruflicher Aufgaben, bestimmter Abteilungen, des gesamten Unternehmens oder der Kommunikation zwischen Unternehmen (bzw. einer Kombination dieser Ebenen) erfolgen. Auf diese Weise lassen sich auch Lastausgleich, Skalierbarkeit und Ausfallsicherheit realisieren. Durch Implementierung einer ausreichenden Zahl ESB-Exemplare kann nach Bedarf betriebliche und logische Flexibilität erzielt werden. Dabei kann die Starrheit herkömmlicher Verbindungen (beispielsweise dedizierter Punkt-zu-Punkt-Verbindungen) in das umfassendere ESB-Konzept integriert werden, ohne auf die Leistungsmerkmale zu verzichten, die herkömmliche Punkt-zu-Punkt-Lösungen bieten können.

### ESBs und die physische Dimension

Von einem ESB zu sprechen, ist zwar korrekt, kann aber (unbeabsichtigt) in die Irre führen. Wie bereits beschrieben, ist ein ESB in den meisten Fällen eine Ansammlung physischer ESB-Exemplare, die sich selbst organisieren, um zusammenzuarbeiten und voneinander zu wissen. Was oft verwirrt, ist der Unterschied zwischen einem logischen ESB und zahlreichen physischen ESB-Exemplaren. Man kauft keinen *ESB*. Was man kauft, ist die Software für die einzelnen ESB-*Exemplare*. Das ESB-Konzept wird durch die Zusammenarbeit zwischen vielen Softwareexemplaren realisiert.

Theoretisch kann ein ESB-Exemplar einen logischen ESB bereitstellen. Aber in der Praxis wird ein erfolgreicher ESB oft unerlässlich für das Erreichen geschäftlicher Ziele. Es ist möglich, ein großes einzelnes ESB-Exemplar zu implementieren, – das beispielsweise auf einem großen, zentralisierten System ausgeführt wird – und folglich einen logischen ESB zu besitzen. Aber diese Vorgehensweise wäre technisch nicht sinnvoll. Wenn dieses Exemplar ausfällt, fallen auch alle Verbindungen aus, die durch dieses Exemplar hergestellt werden. Wenn der einzige Doppeldeckerbus auf einer Route ausfiele, könnten auf dieser Route erst wieder Fahrgäste transportiert werden, wenn der betreffende Bus repariert oder ersetzt worden wäre.

In der Praxis umfassen technische Umsetzungen des ESB-Konzepts mehrere oder zahlreiche einzelne ESB-Exemplare innerhalb eines Netzwerks. Jedes Exemplar besitzt andere Eingangs- und Ausgangskombinationen von Türen und Aufgängen, Oberdecks, Zusatzleistungen und die Fähigkeit zur Kommunikation (mit den übrigen ESB-Exemplaren). Der Vorteil für Ihr Unternehmen besteht darin, dass viele ESB-Exemplare zusammenarbeiten können, um einen flexiblen Vermittler zu bilden. In manchen Fällen bietet es sich an, dies durch Einfügen von ESB-Exemplaren in der Nähe zentralisierter Funktionen zu erreichen. Eine andere Möglichkeit besteht darin, Exemplare und Funktionen zu verteilen. Oder eine bestimmte Kombination dieser gegensätzlichen Varianten zu implementieren, die Ihre geschäftlichen Anforderungen am besten erfüllt.

Das macht das ESB-Konzept so interessant. Physisch kann es so implementiert werden, dass es genau Ihre Anforderungen erfüllt. Logisch stellt es eine sich selbst organisierende Gruppe physischer Exemplare dar, die das ESB-Konzept umsetzen. Im Messaging-Kontext kann ein ESB beispielsweise folgende Modelle unterstützen:

- *Für den Einstieg – das PUSH-Modell (Ich will einsteigen) und das PULL-Modell (der Fahrer hält an und fordert mich zum Einsteigen auf).*
- *Für den Ausstieg – das PULL-Modell (Ich will aussteigen) und das PUSH-Modell (der Fahrer sagt mir, wann, wo und wie ich auszusteigen habe).*
- *Variationen dazwischen – von „publish-and-subscribe"-Funktionen über Punkt-zu-Punkt-Verbindungen bis hin zu Webserviceunterstützung, synchroner Anbindung und Beendung der Verbindung.*

#### Ein ESB auf der Basis von IBM Produkten

IBM WebSphere MQ kann die Basis für einen ESB mit Punkt-zu-Punkt-Messaging, „publish and subscribe"-Funktionen und Clustering bilden. IBM WebSphere MQ Everyplace ergänzt WebSphere MQ durch Nachrichtenbehandlung auf niedrigerem Niveau mit geringerem Speicherbedarf. Auf einer höheren Stufe der ESB-Funktionalität angesiedelt sind IBM WebSphere Business Integration Event Broker und IBM WebSphere Business Integration Message Broker. Sie fügen bereits erwähnte „Oberdeckservices" wie Routing, Konvertierungen und auf Regeln basierende Verarbeitung sowie Ein- und Ausgabefächerung hinzu. Wenn enge Verbindungen zu den Java™ Entwicklungs- und Laufzeitumgebungen erforderlich sind, kann IBM WebSphere Application Server diese Verbindungen sowie synchrone Möglichkeiten bereitstellen.

#### Der *Service* innerhalb eines Enterprise *Service* Bus

Der Begriff *Unternehmen (Enterprise)* bedarf keiner Erklärung. Auch das Konzept eines Netzwerk-*Bus* wurde bereits beschrieben. Wenn man zu verstehen versucht, was ein ESB ist, stellt sich oft die Frage, was genau mit *Service* gemeint ist. Die einfachste Möglichkeit, den Begriff „Service" zu erklären, besteht wahrscheinlich darin, deutlich zu machen, dass ein ESB drei einander ergänzende Servicedimensionen beinhaltet. Die erste Dimension bezieht sich auf die Vorstellung,

dass ein Service eine genau definierte, stabile, lose gekoppelte DFV-Schnittstelle ist, die von einer Anwendung oder Softwarekomponente für andere Anwendungen innerhalb eines Geschäftsprozesses bereitgestellt wird. Dieser Serviceaspekt steht in Einklang mit dem Konzept einer *serviceorientierten Architektur* oder SOA als Mittel zur Einrichtung flexibler Geschäftssysteme. Weitere Informationen zu diesem Konzept finden Sie unter folgender Adresse:
**ibm.com**/software/solutions/webservices

Die zweite Dimension betrifft den Konvertierungsservice zwischen verschiedenen Protokollen. In gewissem Sinne ist dieser Konvertierungsservice ein grundlegender (aber anspruchsvoller) Teil des *Ein- und Ausstiegsverfahrens* eines ESB. Nachdem etwas (über HTTP, WebSphere MQ, Java Message Service [JMS] etc.) beim ESB angekommen ist, muss es in die ausgewählte Sprache übersetzt werden, die die Arbeitssprache dieses ESB ist (vielleicht XML, extended Structured Query Language (eSQL) oder was immer sich sonst eignet).

Wichtig ist in diesem Zusammenhang, dass ein ESB nicht eine bestimmte Art der Umsetzung von Informationen erzwingen sollte. Der Benutzer sollte (innerhalb angemessener Grenzen) das tun können, was er will. Es sollten ihm keine Optionen aufgezwungen werden. Ein Servicefaktor besteht darin, dass ein ESB die Form, in der eine Nachricht eintreffen sollte, zur Auswahl stellt. Anschließend setzt der Konvertierungsservice des ESB die Informationen in Ihr gewähltes generisches Format sowie nach Bedarf in das für die Weiterleitung zum gewählten Ziel erforderliche Format um.

Die dritte Servicedimension steht mit den zusätzlichen Serviceleistungen des „Oberdecks“ in Zusammenhang. Diese Services sind dann relevant, wenn die Informationen, die durch einen ESB fließen, zusätzliche Arbeit erfordern, um die gewünschten Ergebnisse zu erzielen, wie beispielsweise:

- *Permanenz*
- *Zuverlässigkeit*
- *Wiedergabe (im Zusammenhang mit Archivierung)*
- *Umsetzung (im programmatischen Sinn, wie beispielsweise Fahrenheit in Celsius oder Kilogramm in Pfund)*
- *Erweiterung von Nachrichteninhalten – Datenbankzugriff (wie beispielsweise in WebSphere Business Integration Message Broker) für die Suche nach Namen und Adressen anhand einer Kundennummer, bevor diese Daten der ursprünglichen Nachricht hinzugefügt werden*

Das Wesentliche dieser dritten Servicedimension liegt darin, dass Aktivitäten für Sie automatisiert werden können. Die Aufgaben zum Verteilen einer Bestellung können durch einen ESB automatisiert werden, so dass es nicht länger nötig ist, dass eine Person eine E-Mail-Bestellung empfangen und mehrmals drucken oder mehrere E-Mail-Kopien erstellen und an viele Zieladressen senden muss. Durch Zusatzleistungen kann Permanenz oder sogar Speicherung automatisiert werden. Ein ESB-Service kann jeden dieser Services bereitstellen – ob innerhalb einer Datenbank oder beispielsweise in Zusammenarbeit mit einem E-Mail-Server.

Ein „Oberdeckservice“ in einem ESB kann Aktivitäten ausführen, bei denen menschliche Intervention nicht sinnvoll wäre. Allerdings sollte zwischen der Verarbeitungsleistung eines ESB-Brokers, bei dem die eingesetzte Logik absichtlich einfach und repetitiv gehalten wird, und der Verarbeitungsleistung eines Anwendungsservers (wie WebSphere Application Server), dessen Logik komplexer ist, unterschieden werden. Aus diesem Grund ist ein ESB kein Ersatz bzw. keine Alternative zu der anspruchsvolleren Anwendungslogik, für die ein Anwendungsserver ausgelegt ist. Ein ESB sollte nicht versuchen, die Arbeit eines Anwendungsservers zu erledigen. Und ein Anwendungsserver wäre ein zu schweres Geschütz, wenn es um ESB-Aktivitäten geht.

Ein ESB sollte die Bereitstellung von Daten für die auf einem Anwendungsserver ausgeführten Anwendungen und von diesen Anwendungen erleichtern. Ein weiteres Entscheidungskriterium hinsichtlich der von einem ESB zu erbringenden Leistungen sollte die Frage sein, ob die Verarbeitung „nebenbei“, während des Betriebs, stattfindet. Falls ja, ist der Einsatz eines ESB wahrscheinlich die richtige Vorgehensweise. Falls mehr Arbeit, beispielsweise die Interaktion mit einem Kunden, erforderlich ist, sollte dieses Element einer Anwendung übergeben werden.

### Eng oder lose gekoppelte Kommunikation in einer ESB-Umgebung

Die Hauptunterscheidungsmerkmale in Bezug auf eng oder lose gekoppelte Kommunikation in einer ESB-Umgebung lassen sich (beispielsweise) durch Auflisten der Unterschiede zwischen IBM WebSphere MQ Telemetry Transport und WebSphere MQ oder WebSphere MQ Everyplace zusammenfassen. Bei einem eng gekoppelten oder *synchronen* Programmmodell muss die Anwendung erkennen, ob am anderen Ende eine Verbindung besteht. Dieser Umstand kann als gut oder schlecht erachtet werden. Kritiker dieses Modells befürworten im Allgemeinen das lose gekoppelte oder *asynchrone* Modell, während diejenigen, die das Wissen der Anwendung um die bestehende Verbindung für wünschenswert halten, Anhänger des synchronen Modells sind.

Die Erfahrung von IBM hat jedoch gezeigt, dass der Hauptunterschied im Gegensatz zwischen beaufsichtigtem und nicht beaufsichtigtem Betrieb liegt. Bei einem nicht beaufsichtigten Gerät, wie beispielsweise einem Telemetriegerät, ist es vorteilhaft (und oft wesentlich) zu wissen, ob eine Verbindung besteht. Die Anwendung muss mehr tun als einfach nur Daten zu senden und Daten, die eines Tages ankommen könnten, zu empfangen. Übersteigt beispielsweise in einer Fabrik eine Temperatur einen bestimmten Wert, ist es wichtig, dass zuständige Mitarbeiter davon Kenntnis erhalten, damit die erforderlichen Maßnahmen eingeleitet werden können. Es nützt wenig, wenn die Warnung sofort gesendet, aber erst deutlich später oder auf eine Weise, die das Einleiten geeigneter Maßnahmen nicht erleichtert, empfangen wird. Bei nicht beaufsichtigtem Betrieb besteht das Ziel darin, die vorhandene Netzverbindung zu nutzen. Funktioniert das nicht, müssen Alternativen getestet werden, bis erfolgreich eine Verbindung hergestellt wird. Scheitern alle Versuche, muss das Gerät möglicherweise die autonome Entscheidung treffen, dass die Umschaltung zum sicheren Betrieb erfolgen sollte.

Der Preis, der für synchrone oder eng gekoppelte Kommunikation zu zahlen ist, besteht teilweise darin, dass die Arbeitslast des Anwendungsentwicklers erhöht wird. Der Entwickler muss alle möglichen Szenarien kennen und dann Lösungen erstellen, die einige davon abdecken. Dadurch werden Anwendungen umfangreicher und komplexer, weil gemäß der programmierten Sequenz verschiedene Kombinationen von Wiederholungen durchgeführt werden. Beispielsweise könnte die erste Verbindung über das normale interne LAN, die zweite über eine Wählverbindung, die dritte über Satellit und die letzte über Global Packet Radio Service (GPRS) erfolgen.

Das Problem ist, dass die Einbeziehung all dieser Alternativen zusätzliche Arbeit für den Anwendungsprogrammierer bedeutet. Aber wenn es um eine Produktionsanlage geht, könnte die Nichtberücksichtigung dieser Alternativen zu einer Katastrophe führen – zu einem Brand in einer Chemiefabrik, zu einem Leck in einer Raffinerie oder zur Herstellung verunreinigter Lebensmittel in einer Konservenfabrik. In diesen Kontexten ist es nicht wünschenswert, Nachrichten an eine undurchsichtige Middlewareschicht zu senden, bei der man nicht weiß, wann die Informationen ankommen. Stattdessen sollten Informationen durch eine transparente Middlewareschicht gesendet werden, die ausreichend Kontrolle über die Zustellung wichtiger Informationen bietet.

In diesem Kontext muss ein ESB eine weitere Dimension bieten, nämlich die Möglichkeit, dass Menschen direkte Verantwortung für ihre Daten übernehmen. Jeder ESB muss in der Lage sein, eng gekoppelte oder synchrone Funktionen bereitzustellen, die solche Anforderungen erfüllen, selbst wenn es bedeutet, dass eine (in Bezug auf den ESB) externe Anwendung (die von einem Menschen überwacht wird oder menschliche Intervention anfordert) die Verantwortung für die Aktion übernimmt. Und es ist zu berücksichtigen, ob der Status der Verbindung *jenseits* des ESB bekannt sein muss.

Beispielsweise könnte eine Quellenanwendung einem ESB wie WebSphere Business Integration Message Broker eine Nachricht synchron oder eng gekoppelt übermitteln. Sobald jedoch WebSphere Business Integration Message Broker die Eingabe akzeptiert, wird die enge Verbindung gelöst (auch wenn sie von einem synchronen Protokoll wie WebSphere MQ Telemetry Transport stammt). Wenn eine kontinuierliche Verbindung über den ESB (in diesem Fall WebSphere Business Integration Message Broker) hinaus, von der Quelle bis hin zu den Zieladressen, benötigt wird, sind andere Aspekte zu berücksichtigen – ob beispielsweise die Zielanwendung eine Bestätigungsnachricht in die Gegenrichtung sendet, aus der hervorgeht, dass die ursprüngliche Nachricht empfangen wurde.

#### Lose gekoppelte oder asynchrone Kommunikation in einer ESB-Umgebung

Der Vorteil lose gekoppelter oder asynchroner Lösungen besteht darin, dass sie den Arbeitsaufwand von Anwendungsentwicklern im Hinblick auf das erwünschte Verhalten dieser Lösungen verringern. Statt die Berücksichtigung aller möglichen Szenarien und Programmierung im Hinblick auf die relevantesten Szenarien zu erfordern, eröffnet die Entkopplung die Möglichkeit, dass verschiedene Teile eines Gesamtprozesses durch unterschiedliche Aktivitäten erstellt werden. Die Quellenanwendung muss nichts über die Zielanwendung wissen (und umgekehrt). Dies traf schon immer auf WebSphere MQ zu, weshalb WebSphere MQ eine beliebte Lösung für Großunternehmen ist.

Durch die Hinzunahme eines ESB ergibt sich noch mehr Flexibilität. Eine Quelle muss nur den ESB (wie WebSphere Business Integration Message Broker) kennen – nicht alle Zielanwendungen des ESB. Ebenso wenig muss eine Zielanwendung die Details aller Quellenanwendungen kennen. Der ESB ist das Ziel, soweit es die Quelle betrifft (oder die Quelle, soweit es eine der Zielanwendungen betrifft).

Nach Empfang einer Nachricht kümmert sich der ESB um die Weiterleitung der Quelleninformationen zum angegebenen Ziel bzw. zu den angegebenen Zielen. Um in der Doppeldeckerbus-Analogie zu bleiben: Die Verarbeitungsfunktionen des ESB können „Oberdeckservices" nutzen, um Nachrichten zu optimieren, auf Regeln basierende Weiterleitung anzuwenden oder Nachrichten auf Platte zu speichern. Anschließend empfangen die Zielanwendungen die Informationen (über die entsprechenden Türen und Aufgänge).

Ein weiterer entkoppelter Service, den ein ESB bieten kann, ist die Empfangsbestätigung. Die Quelle kann eine Antwort oder Nachricht erhalten, aus der hervorgeht, dass die Zustellung am Endpunkt erfolgt ist. Das ist wichtig, da es Anwendungen gibt, die eine ausdrückliche Bestätigung der Weiterleitung einer Nachricht von der Quelle zum Endziel benötigen. Die Bestätigung zeigt, dass die Middleware wie vorgesehen funktioniert hat. Entkoppelte Umgebungen sind ideal für Unternehmen, die keine große Zahl von Bestätigungen bearbeiten wollen (oder müssen). Innerhalb dieser Umgebung stellt sich die Frage, ob gesendete Nachrichten empfangen wurden, nicht mehr, weil man sich darauf verlassen kann, dass der ESB die Zustellung ausführt.

Ein ESB kann diese Funktionen bereitstellen – insbesondere im Kontext beaufsichtigter Anwendungen. Statt auf den Feinheiten der Netzkonnektivität (wie bei unbeaufsichtigten Systemen) kann der Schwerpunkt auf der Interaktion mit dem Benutzer und der Vorbereitung der zu sendenden Nachricht liegen. Dies gilt für Anwendungen, die miteinander kommunizieren, wie beispielsweise einen Personal Digital Assistant (PDA), der die Details eines Versicherungsleistungsanspruchs übermittelt, oder einen Laptop-Computer, der eine Bestellung aufgibt.

In solchen Fällen besteht das Problem normalerweise nicht darin, ob der Leistungsanspruch oder die Bestellung bereits eingereicht wurde, sondern darin, ob man sich darauf verlassen kann, dass dies geschehen wird. Was den Benutzer angeht, so ist für ihn mit dem Betätigen der Übergabeschaltfläche die Angelegenheit erledigt. Der ESB hat dafür zu sorgen, dass der Rest ebenfalls eintritt.

Die Anwendung gibt die Nachricht an die Middleware weiter, die sie zum nächsten Teil des ESB weiterleitet. Der ESB ermittelt, was als Nächstes geschehen soll – indem er beispielsweise die Content-Routing-Logik in WebSphere Business Integration Message Broker nutzt –, und unterstützt dann die Zustellung zu den vorgesehenen Zielen. Die Quellenanwendung stellt lediglich das entsprechende Verb für die Übergabe der Nachricht an die Middleware bereit, von wo aus dann alles weitere seinen Lauf nimmt, während die Nachricht zum vorgesehenen Ziel weitergeleitet wird.

Im Gegensatz zur eng gekoppelten Kommunikation übernehmen die Benutzer in einem Kontext mit lose gekoppelter Kommunikation weniger Verantwortung für die Übertragung von Nachrichten. Bei eng gekoppelter Kommunikation unterstützt die Infrastruktur die Anweisungen des Programmierers. Bei entkoppelter Kommunikation geht der Programmierer davon aus, dass die Verantwortung für die Übertragung und Bereitstellung ganz auf die Infrastruktur übergeht, die (unter Verwendung einer Komponente wie eines ESB) in der Lage ist, diese Aufgaben selbständig abzuwickeln.

**Einfache und komplexe Messaging-Funktionen in einer ESB-Umgebung**

Bei der Betrachtung einfacher („leichter“) und komplexer („schwerer“) Messaging-Funktionen müssen wir uns zunächst einmal klar machen, was unter dem „Gewicht“ einer Nachricht zu verstehen ist. Unterschiedliche Übertragungsumgebungen erfordern unterschiedliche Lösungen. In einem modernen Unternehmens-LAN ist mit großer Wahrscheinlichkeit ein Gigabit-Ethernet vorhanden. Wenn die Aufgabe darin besteht, eine Nachricht über einen Transfer in Höhe von 500 Millionen US-Dollar zu übertragen, muss gewährleistet sein, dass die Nachricht ankommt und dass sie nur ein einziges Mal empfangen wird. Dazu ist komplexes Messaging erforderlich, da es viele Funktionen (von der Permanenz über einmalige Bereitstellung bis hin zu Transaktionsfunktionen und Protokollierung) beinhaltet. Obwohl diese Nachricht möglicherweise nur einen winzigen Teil des Gigabit-Ethernets in Anspruch nimmt und obwohl der zugehörige Protokollaustausch möglicherweise mit einem hohen Systemaufwand verbunden ist, ist der Wert der Transaktion so groß, dass die gesicherte Zustellung von A nach B Vorrang vor allem anderen hat. Eine solche Nachricht zu verlieren oder doppelt zuzustellen, könnte sehr teuer werden. Dennoch sind die Netzwerkkosten belanglos, die Zustellung erfolgt mit hoher Geschwindigkeit und die Latenz ist niedrig.

Daher ist in einem solchen Fall komplexes Messaging praktikabel – und vorzuziehen. Es ist möglich, komplexe Messaging-Funktionen einzusetzen, auch wenn die konkreten Anforderungen niedrig sind – weil die Netzwerkkosten im Kontext des Unternehmens-WAN oder -LAN vernachlässigbar sind. Außerdem tragen Messaging-Funktionen, die das „Gewicht“ einer Nachricht – beispielsweise durch längere Felder, aussagekräftigere Zeitmarken oder Raum für zukünftige Verbesserungen – erhöhen, nicht zur Erhöhung der Kosten pro Nachricht bei. Die Kosten lassen sich vollständig über die betrieblichen Einsparungen durch eine einheitliche Messagingstrategie rechtfertigen.

Nachrichtenübertragung war in der Vergangenheit meist komplex, weil sie in einer Unternehmensumgebung stattfand, in der die Netzwerkkosten unerheblich waren. Auf Einzelnachrichtenbasis bedeutete das, dass die Merkmale und Anforderungen komplexer Messaging-Funktionen als die Norm betrachtet wurden. Aber solche Gewohnheiten können sich als teuer und unangemessen erweisen,

wenn in Wirklichkeit ein Bedarf an einfachem Messaging besteht. Das führt zu einer parallelen Schlussfolgerung. Der Bedarf an einfachem Messaging könnte in der Praxis höher sein als bisher angenommen wurde. Weshalb? Weil einfaches Messaging in den verschiedensten Anwendungsszenarien einsetzbar ist – auch wenn es (in erster Linie aufgrund fehlender Kenntnisse) schwieriger und teuerer einzurichten ist.

#### Einfachere Messaging-Funktionen

Durch den Vormarsch des Internets sowie die Verbreitung von Messaging-Systemen, die für kompakte (mobile) Hightech-Endgeräte optimiert sind, gewann einfacheres und einfaches Messaging an Bedeutung. Bei der Nachrichtenübertragung über das Internet ist die Bandbreite an den Endpunkten meist eingeschränkt. Und die besondere Natur des Internets bedingt Multihop-Verbindungen von der Quelle zum Ziel. Man muss nur eine Routenverfolgung durchführen, um zu verstehen, wie viele Hops beteiligt sind. Das bedeutet, dass mit dem Senden kleiner Nachrichten ein hoher Systemaufwand pro Nachricht von der Übergabe bis zur Zustellung verbunden ist. Dies kann sowohl teuer als auch unerwünscht sein, vor allem weil die Übertragung der Nachricht sehr zeitaufwändig ist. Ein *ausführliches Protokoll* bedeutet eine viel höhere Latenz, da die Ausführlichkeit bei jedem einzelnen der zahlreichen im Internet stattfindenden Hops Verarbeitungsschritte erfordert.

Eingeschränkte Bandbreite und möglichst geringe Latenz bedeuten, dass ein komplexes Messaging-Protokoll möglicherweise nicht geeignet ist. Das heißt nicht, dass komplexes Messaging über das Internet nicht möglich ist, aber es stellt oft nicht die optimale Lösung dar. Mit anderen Worten: Die Annahmen, die dem Einsatz von WebSphere MQ in einer Unternehmens-WAN- oder LAN-Umgebung zugrunde liegen (wie beispielsweise unter allen Umständen genau einmalige Zustellung), sind auf viele Internetverbindungen nicht notwendigerweise anwendbar. Wenn es also Umstände gibt, unter denen der volle Umfang komplexer Messaging-Funktionen nicht erforderlich ist, sind „leichtere“ Alternativen mit geringerem Funktionsumfang (und Systemaufwand) eine geeignete Option. Wenn Sie beispielsweise über das Internet ein Buch von einem Einzelhändler kaufen, ist es kein Weltuntergang, wenn eine Verbindung unterbrochen wird. Sie stellen einfach die Verbindung wieder her und beginnen von vorn. Auch in einer Wertpapierhandelsumgebung wäre es unerheblich, eine Preisschwankung zu verlieren, wenn kurz danach die nächste angezeigt wird. Gerade ausreichend ist gut genug.

Das Internet schafft neue Möglichkeiten der Kommunikation mit Zehntausenden oder gar Hunderttausenden von Applets, die in Webbrowsern ausgeführt werden. Solche Anwendungen benötigen einfachere Protokolle, die skalierbar sind und weniger Ressourcen beanspruchen. (Komplexes Messaging ist wegen der Nachrichtengröße, der Protokollverarbeitung, Datenbankspeicherung, automatisierten Wiederherstellung usw.) weniger skalierbar. Im Internet aber zählt Größe. Nehmen wir an, 450.000 Menschen wollen die Punktestände eines Tennisturniers in Wimbledon live verfolgen. Sie wollen den aktuellen Punktestand, sobald er verfügbar ist. Aber wenn sie einen Punkt verpassen, ist es nicht tragisch, da der nächste Stand den vorherigen außer Kraft setzt, wie beispielsweise beim Übergang von 40–15 zu 40–30.

Um schnelle Bereitstellung zu ermöglichen – so nahe an Echtzeit wie es die inhärente Latenz des Internets zulässt – muss mit jeder übertragenen Information ein möglichst geringer Systemaufwand verbunden sein. Je „leichter" die Nachricht (möglichst in einem einzigen TCP/IP-Netzpaket enthalten), desto schneller kann sie über das Internet übertragen werden. Und Bandbreiteneinschränkungen am Empfangsende (wie beispielsweise eine langsame Modemverbindung) spielen eine geringere Rolle. Es besteht die Möglichkeit, dass ein Punktestand oder eine Kursschwankung gar nicht übertragen wird, aber die nächste Aktualisierung setzt die Nachricht, die nicht ankam, ohnehin außer Kraft.

Das IBM WebSphere MQ Real Time Protokoll wurde aus diesem Grund erstellt. Es optimiert die Funktionalität eines ESB, indem es Erstellung und Empfang „leichterer" Nachrichten ermöglicht – ohne die Vorgaben des anspruchsvolleren WebSphere MQ. Eine einzelne Nachricht kann im Internet veröffentlicht und so von 10.000 oder 450.000 oder gar 10.000.000 Menschen gesehen werden. Das ist nicht nur für den Urheber der Nachricht effizienter (er muss keine individuell adressierten Nachrichten erstellen), sondern es ist auch besser für das Internet als Ganzes, da sich solche Großereignisse weniger stark auf das Netzwerk auswirken.

### Einfache Messaging-Funktionen

Einfaches Messaging stützt sich fast ausschließlich auf das Vorhandensein kompakter (mobiler) Hightech-Endgeräte, die überall – von Maschinen, über Pkws und Lkws, Sensoren an Übertragungsleitungen oder Pumpen bis hin zu Haushaltsgeräten wie Kühlschränken, Klimaanlagen und Öltanks (siehe Abb. 2) – installiert sein können. Der Hauptunterschied liegt in der Servicequalität und Skalierung – wobei ein zusätzliches Problem darin besteht, dass die verfügbare Bandbreite sowohl gering als auch sehr teuer sein kann. Modems mit 9600 Baud sind branchenüblich. Wenn (beispielsweise an abgelegenen oder unzugänglichen Orten ohne feste Telefonverbindungen) Satelliten eingesetzt werden, könnten die Kosten bis zu fünf Cent pro übertragenem oder empfangenem Byte betragen.

Die gute Nachricht besteht darin, dass die zu sendenden oder zu empfangenden Daten meist viel kleiner sind – es handelt sich nicht immer um das (mehrere Tausend Byte umfassende) Bankprofil einer Person, sondern oft um einen einfachen Befehl, etwas ein- oder auszuschalten, einen Bericht über einen Temperaturwert oder einen Alarmhinweis auf einen bekannten Zustand, der sich geändert hat. Das bedeutet, dass die zu sendenden Daten klein sein sollten – und

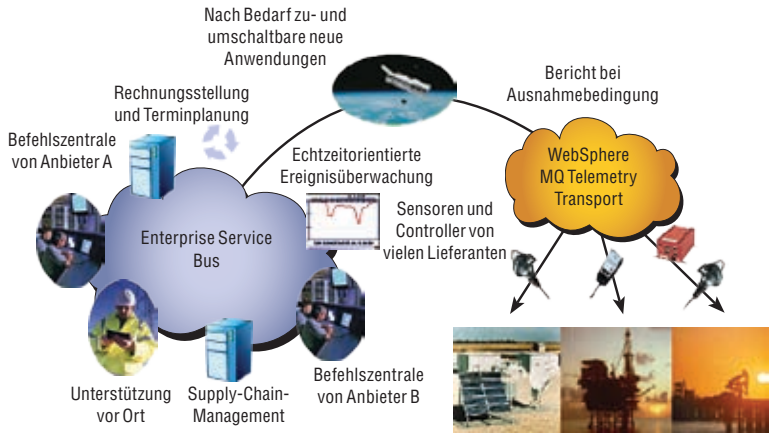


*Abbildung 2. Einfaches Messaging mit WebSphere MQ Telemetry Transport.*

der mit dem Senden verbundene Systemaufwand so gering wie möglich sein sollte. Es ist nicht sinnvoll, zwei Byte Daten zu senden, wenn mit der Übermittlung dieser Daten ein Aufwand von 10 KB für Weiterleitung und anderen Systemaufwand verbunden ist.

Wie bereits besprochen, besteht ein Vorteil eines ESB darin, dass er durch seine verschiedenen Türen und Aufgänge das Empfangen und Senden vieler verschiedener Formen von Nachrichten unterstützen kann. Im Kontext leichter bis schwerer Nachrichten stellen folgende Werte die Mindestgröße eines Nachrichtenheaders dar.

- *Für WebSphere MQ Telemetry Transport: 2 Byte*
- *Für WebSphere MQ Everyplace: 18 Byte*
- *Für WebSphere MQ Real Time: 24 Byte*
- *Für WebSphere MQ: 480 Byte*

Wenn eine Satellitenverbindung oder ähnliche Verbindung auf Byte-Basis bezahlt wird, sind die Vorteile einer WebSphere MQ Telemetry Transport- oder WebSphere MQ Everyplace-Nachricht offenkundig (wenn man davon ausgeht, dass andere damit verbundene Einschränkungen akzeptabel sind). Ein ESB gibt einem Unternehmen die Möglichkeit, zu wählen, was wo verwendet wird, und die beste aller relevanten Optionen zu erhalten. Ob die Wahl letztlich auf einfaches, einfacheres oder komplexes Messaging fällt, hängt von der erforderlichen Funktionalität sowie der zugrunde liegenden Ausfallsicherheit und Bandbreite der verschiedenen Kommunikationsoptionen ab. Aber bei einem ESB sind Kombinationen möglich – und damit bietet dieser erhöhte Flexibilität.

#### Schlussfolgerung

Einfaches Messaging ist nicht überall die beste Option. Beispielsweise können nicht Transaktionen auf mehrere Nachrichten bezogen werden, obwohl es möglich ist, eine solche Anforderung auch beim Einsatz einfacher Messaging-Funktionen durch entsprechende Programmierung zu erfüllen. Gleichzeitig ist der Einsatz komplexer Messaging-Funktionen für die Bereitstellung von Tennis- oder Golfpunktetänden oder die Weiterleitung von Signalen von Fernsensoren über einen Satelliten zu teuer (wenn auch möglich). Der Vorteil eines ESB besteht darin, dass die Vorzüge der einzelnen Messaging-Formen ausgewählt und zu Lösungen kombiniert werden können, die keine teuere, maßgeschneiderte oder starre Infrastruktur erfordern.

Mit einem ESB könnten Sie sogar den größten Teil Ihrer Nachrichten als „leicht“ einstufen, aber wenn gelegentlich doch eine wichtige oder wertvolle Nachricht zu übertragen ist, kann der ESB entscheiden, sie – wegen ihres Inhalts – mithilfe eines komplexen Mechanismus wie WebSphere MQ statt mit WebSphere MQ Telemetry Transport zu übertragen. Ebenso können lose und eng gekoppelte Komponenten nach Bedarf kombiniert werden. Ein ESB ermöglicht verschiedene Vorgehensweisen, sodass Sie die Möglichkeit haben, die unmittelbaren Anforderungen Ihres Unternehmens sofort zu erfüllen und dann die Komponenten zu verändern oder zu optimieren, wenn sich die geschäftlichen Bedingungen weiterentwickeln oder ändern.

Ebenso vorteilhaft ist es, dass ein ESB ein Gesamtkonzept darstellt, das ein oder mehrere physische ESB-Exemplare umfasst, die zusammenarbeiten, um Vorgänge zu automatisieren, die bislang menschliche Intervention erforderten. Ein ESB versucht nicht, alles zu erledigen, und das sollte er auch nicht tun (es gibt Ebenen der geschäftlichen Verarbeitung oberhalb eines ESB, wie beispielsweise Workflow- und Prozesskoordination, wie man sie bei IBM WebSphere Business Integration Server findet). Aber ein ESB kann die Routineintegration von Anwendungen, Systemen und Sensoren auf eine Weise automatisieren, wie es bislang nicht möglich war.

Man hört oft, in einem ESB komme das Nutzenpotenzial der Middleware voll zum Ausdruck – weil er das Bindeglied ist, das Flexibilität hinsichtlich der Interaktion zwischen Anwendungen ermöglicht. Ein ESB unterstützt die Integration von Anwendungen, was viele Unternehmen anstreben, weil sie es nicht mehr tolerieren können, dass ihre vorhandenen und neuen IT-Ressourcen nur aufgrund menschlicher Intervention zusammenarbeiten können. Wenn Systeme und Anwendungen zusammenarbeiten können, um Routineaufgaben auszuführen, können sich Menschen auf anspruchsvollere Aufgaben mit höheren Erträgen konzentrieren.

Technischer ausgedrückt: Bei Verwendung eines ESB brauchen sich Mitarbeiter keine Gedanken mehr über die Details der Verbindungen zwischen Anwendungen zu machen. TCP/IP und seine Protokolle stellen das Rückgrat des Internets dar, weil sie alle Anforderungen (ob von Webbrowsern, aus E-Mail- oder Instant Messaging-Programmen kommend) zur ausgewählten Zieladresse weiterleiten. Ein ESB erfüllt – auf einer höheren, anspruchsvolleren Ebene – eine ähnliche Funktion. Das allgemeine Konzept ist dasselbe, aber es können sehr viel wertvollere Ergebnisse erzielt werden.

Mit dem Einsatz eines ESB beginnen Unternehmen, das geläufige Szenario statischer Verbindungen, die (von der Definition über die Konfiguration bis hin zur Pflege) von Menschen physisch verwaltet werden müssen, zu eliminieren. Solche Szenarien sind zu teuer und zu unflexibel, um heutige geschäftliche Anforderungen zu erfüllen. Ein ESB hingegen ist die Umsetzung eines Nutzenpotenzials, bei dem Funktionalität den Vorrang vor physischen Verbindungen (TCP/IP) hat, um erhöhte und automatisierte Zuverlässigkeit und Flexibilität der Nachrichtenübertragung zwischen Anwendungen zu gewährleisten.

### Weitere Informationen

Weitere Informationen zum Enterprise Service Bus finden Sie unter folgender Adresse:

**ibm.com**/software/integration/esb

IBM Deutschland GmbH
70548 Stuttgart
**ibm.com**/de

IBM Österreich
Obere Donaustraße 95
1020 Wien
**ibm.com**/at

IBM Schweiz
Vulkanstrasse 106,
Postfach
8010 Zürich
**ibm.com**/ch

Die IBM Homepage finden Sie unter:
**ibm.com**



Hergestellt in den USA
01-05



G224-9149-00-GE