

## Composite Business Services

### Ein *Top-Down*-Ansatz für Web Services mit Ivory

---

“Ein Top-Down-Ansatz gibt ihnen die Flexibilität, um neue Mainframe-Web-Services aufzubauen, die die heutige Geschäftswelt braucht. Es bietet die Leistung einer programmatischen Lösung ohne Programmierkenntnisse.“

Joe Ganem President & CEO, GT Software



Composite Business Services sind neue Services, die durch die nahtlose Kombination bestehender Anwendungen, ohne vorgenommene Programmänderungen erstellt werden. Diese Composite Business Services können Geschäftsprozesse ausführen, für die einzelne Anwendungen nie erstellt wurden. Composite Mainframe Services können viele verschiedene Mainframe-Transaktionen und Datenquellen in einer einzigen Definition bündeln. Diese Composite Mainframe Services können wiederum in höherliegenden Composite Business Services integriert werden. Mit der Verfügbarkeit von Web Services auf der Mainframe ist es nun möglich, diese Composite Business Services als WSDL für die Nutzung mit jedem Web Service Client bereitzustellen.

Es gibt zwei Ansätze zum Design dieser Composite Business Services: Top-Down und Bottom-Up.

Der Bottom-Up-Ansatz sieht zuerst auf die verfügbaren Transaktionen und Daten. Dann werden die gesamten Transaktionen oder Datenquellen mit einer Schnittstelle versehen, durch die sie den Clients zur Verfügung gestellt werden. Dies ist eine eins-zu-eins Abbildung der Transaktion zu einer Web Service-Schnittstelle ohne inhärente Kombinationen von Funktionen. Dieser Ansatz kann auf dem Level von COBOL und/oder COPYBOOKs starten und diese als Grundlage des Designprozesses verwenden. Hierdurch würden alle Daten den Clients zur Verfügung stehen (möglicherweise als WSDL) aber die Fähigkeit die Komponenten in einem Composite Service zu kombinieren würde dem Web-Service-Verbraucher überlassen (z.B. einem Applikationsserver der auf einer Mid-Tier Plattform läuft). Bei diesem Ansatz ist der Applikationsserver verantwortlich für die Koordination aller Daten und der Ablaufsteuerung zwischen den einzelnen Web Services und kann somit zum Engpass werden. Alle Daten zwischen den verschiedenen Komponenten werden durch den Applikationsserver übertragen und nicht unmittelbar vom Punkt der Generierung bis zum Punkt des Abnehmers.

Dieser Ansatz kann als ein erster Ausgangspunkt für das Design verwendet werden, da es sich um eine eins-zu-eins Abbildung der zugrunde liegenden Mainframeressourcen handelt.

Der Top-Down-Ansatz sieht zunächst auf die Business-Services, die spezifiziert oder angefragt wurden oder auf ein zuvor erstelltes WSDL. Dann wird geprüft, welche In- und Outputs benötigt werden um den Anforderungen dieses Services gerecht zu werden. In der Regel liefert der Business Analyst die Voraussetzungen, welche in diesem Bereich benötigt werden oder es besteht ein WSDL, das von einem Geschäftsprozess-Modelingtool erstellt wurde. Sobald diese Anforderungen festgelegt wurden, prüft der Designer ob bestehende Businesskomponenten vorhanden sind, die zur Erfüllung der Anforderungen notwendig sind. Möglicherweise sind mehrere Komponenten (Transaktionen oder Datenquellen) notwendig, um die Anfrage zu beantworten. Nachdem diese Businesskomponenten identifiziert und spezifiziert wurden, werden sie in einer Definition einschliesslich der dazugehörigen Geschäftsbewegungen zusammengefasst. Dann werden die zugrunde liegenden Komponenten in der erforderlichen Logik des Geschäftsflusses abgebildet um die Anfrage zu vollenden. Diese zusammengesetzte Business Service Definition wird als eine einzige WSDL publiziert, welche von WSDL-Clients aufgerufen werden kann. Der anfordernde Client stellt eine Anfrage an den Business Service, dieser stellt die Antworten mehrerer Businesskomponenten zusammen und liefert ein Ergebnis. Die Koordinierung dieser Businesskomponenten werden auf dem Host-System durchgeführt – es ist weder Endnutzer-Wissen nötig noch wird Kenntnis über die Mid-Tier-Plattform benötigt. Es gibt keinen zentralen Koordinator, an dem es zu potentiellen Engpässen kommen könnte. Das Verteilen der Daten reduziert die Netzwerklast und verbessert die Transferzeiten. Wenn der Mainframe der Provider des Composite Business Services ist, gibt

es nur einen einzigen Aufruf – das bedeutet eine Verringerung der Netzwerklatenz, der TCP/IP Stack Zeit, kürzere Zeit für die Erstellung der Session und weitere Zeitersparnis, da die Daten nur einmal pro Anfrage zusammengestellt werden müssen und nicht für jede Teilanfrage einzeln.

## Ein Beispiel

Joe's Telephone Company (JTC) möchte seinen Kunden die Möglichkeit geben, auf ihre Kundenkontodaten zuzugreifen – einschließlich Ihrer Rechnungen und einer Liste der vor Ort installierten Telefonhardware.

JTC hat ein zuverlässiges, jahrzehnte altes CICS System, das diese Informationen zur Verfügung stellt, wenn JTC's Servicemitarbeiter vom Kunden angerufen werden. JTC möchte eine Self-Service-Web Anwendung erstellen, welche den Kunden auffordert, seine Kundennummer (Telefonnummer) anzugeben um die angeforderten Informationen zu erhalten.

Das aktuelle System besteht aus einer Vielzahl von Transaktionen - aber für die neuen Web-Service Anwendungen werden nur drei benötigt.

- **JTCBILL** – liefert Rechnungsinformationen des Kunden (input: Telefonnummer, output: mehrer Seiten Rechnungsdetails einschließlich der vorherigen Monatsrechnung, falls erwünscht)
- **JTCRI** – liefert den kompletten Kundeneintrag (input: Telefonnummer, output: kompletter Datensatz des Kunden)
- **JTCEQUIP** - liefert eine Liste mit Details der beim Kunden vor Ort eingesetzten Telefonhardware (input: Kundenadresse, output: komplette Ausrüstung, Details zur Installation)

Bisher muss der Kunde während der Geschäftszeiten einen Servicemitarbeiter anrufen um Kontoinformationen zu erhalten. Der Kunde gibt seine Telefonnummer (Kontonummer) an. Der Servicemitarbeiter greift auf das Kundeninformationssystem zu, gibt die Kundennummer ein, sucht die Informationen und leitet diese an den Kunden weiter. Wenn der Kunde Rechnungsinformationen benötigt, kann der Service-Vertreter mit einem Tastendruck die Abrechnung ausführen. Wenn der Kunde Informationen über die installierten Geräte braucht, kann der Service-Vertreter mit einem weiteren Tastendruck die Ausrüstungs-Anfrage-Transaktion ausführen.

JTC Kunden wollen jederzeit über das Web Zugang zu diesen Informationen haben. JTC möchte seine bestehenden, verlässlichen Systeme nicht verschrotten und kein Geld dafür verschwenden, bestehende Anwendungen neu zu programmieren. Stattdessen hat JTC beschlossen, die vorhandenen Funktionen weiter zu verwenden.

Wir werden uns Business-Service-Designs für JTC sowohl im Hinblick auf *Top-Down* als auch in Hinblick auf *Bottom-Up* Designansätze anschauen.

## Bottom-Up Design

Vorausgehende Untersuchungen haben ergeben, dass die bestehenden Transaktionen die erforderlichen Funktionen erfüllen, daher wird eine Schnittstelle für *jede* Transaktion definiert:

- getBill (input: Telefonnummer, output: Rechnungsverzeichnis)
- getCustInfo (input: Telefonnummer, output: Kundenverzeichnis)
- getEquipment (input: Telefonnummer, output: Geräteverzeichnis)

Diese Schnittstellen werden in drei unabhängigen WSDL-Dokumenten für den Gebrauch veröffentlicht.

Eine weitere Anwendung, die auf einem Mid-Tier-Server läuft, wird zur Schnittstelle zwischen Mainframe und dem Kunden, der über eine Website seine Informationen abrufen möchte. Diese Anwendung ruft alle drei Webservices auf und kombiniert aus deren Antworten die Informationen, die dem Kunden präsentiert werden. Wenn der Kunde ältere Monatsrechnungen einsehen möchte, fordert die Anwendung die jeweilige Rechnung an.

Dieser Ansatz nutzt eine Mid-Tier-Ebene, mit drei Web Service Aufrufen über das Netzwerk (erhöhte Netzwerklatenz), als Antwort werden die kompletten Datensätze geschickt. Die Zusammenstellung der Ergebnisse in einem Composite-Service findet auf Ebene des Applikationsservers statt.

## Top-Down Design

Der Web Service Designer legt fest, dass die Telefonnummer der erforderliche Input für den Service ist, der Output besteht aus Teilen des Kunden- und aus Teilen des Geräteverzeichnisses.

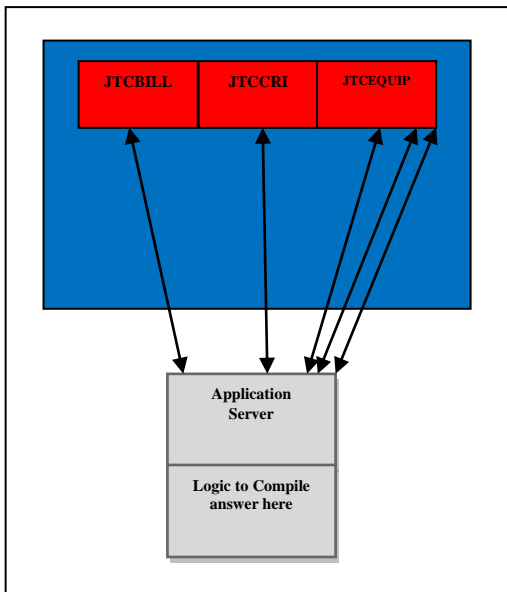
Die Schnittstelle ist wie folgend konzipiert:

GetCustomer (input: Telefonnummer, output: Kundenverzeichnis, Geräteverzeichnis)

Der Composite Business Service liefert dem Kunden nur die in der Schnittstelle festgelegten Felder. Die Schnittstelle ist über WSDL spezifiziert und ist für den Gebrauch veröffentlicht.

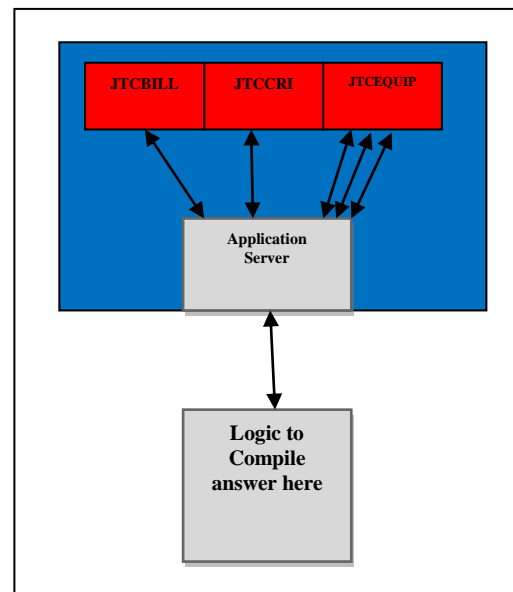
Der Kunde ruft den Composite Business Service auf und gibt die Telefonnummer als Input ein. Der Server, welcher auf dem Host läuft, erhält die Composite Business Service Anfrage, ruft alle drei Vorgänge auf und sendet die zusammengestellten Ergebnisse zurück an den anfragenden Kunden.

Dieser Ansatz untersucht zunächst die In- und Outputs die vom Business Service benötigt werden und legt dann fest, welche Ressourcen diese Anforderungen erfüllen können. Dies erfordert keinen Mid-Tier- oder Applikationsserver. Dieses Vorgehen entspricht weitgehendst dem Weg, den ein Business Analyst zur Lösung dieses Problems gehen würde.



#### Mid-Tier Konsolidierung auf Anfrage (Bottom-Up)

- Mehrere Anfragen an den Mainframe
- Logik notwendig um Antworten off-Mainframe zu erstellen
- Application Server wird Engpass für Antworten
- JTCEQUIP ruft Screens mehrmals ab
- Mehrere Schnittstellen für den Applikationsserver um den Mainframe zu kontaktieren



#### Composite Reaktion der Anfrage (Top-Down)

- Einziger Antrag auf Mainframe
- Logik notwendig um Antworten on-Mainframe zu erstellen
- Applikationsserver wird einziger Anforderer für Services
- Schnittstelle zu JTCEQUIP handelt on-host
- Eine Schnittstelle für Anwendungen um Composite Applications aufzurufen

### Top-Down Design verwendet Web Services Ivory

Ivory Web Services nutzen den Top-Down Ansatz um Web Services zu entwerfen, die Composite Business Services auf dem Mainframe erstellen. Der Ivory Server läuft direkt auf dem Großrechner und verringert damit die Netzwerkbelastung, die notwendig ist um die erforderlichen Antworten zu erhalten. Der Ivory Server kann mehrere Transaktionen und Datenquellen verarbeiten um die Geschäftsanforderungen, die vom Business Analysten in Ivory definiert wurden, zu erfüllen. Ivory Studio ermöglicht dem Entwickler, neue Businesslogik aus vorhandenen Applikationen zusammensetzen, ohne dass neuer Code geschrieben werden muss. Alle Interaktionen der Anwendung können mit Verknüpfungssteuerung (Entscheidungspunkte, Schleifen, etc.) definiert werden ohne dass dabei die Anwendungen verändert werden müssen oder Kenntnisse über die genaue Funktion der einzelnen Anwendungen vorliegen müssen.

Ivory Studio ermöglicht die Erstellung von Prozessabläufen, welche auf neu zusammengesetzter Funktionalität beruhen, die von bestehenden Ressourcen angetrieben werden. Mit Ivory Studio kann ohne zusätzlichen Programmieraufwand eine Lösung erstellt werden, für die es im Normalfall nötig gewesen wäre, zusätzliche Logiken zu programmieren. Jegliche Logik im Ivory Studio wird per Drag-and-Drop Technologie definiert und dann auf dem Ivory Server veröffentlicht. Der gesamte Composite Business Service wird durch das generierte WSDL vertreten, welches im Ivory Server bereitgestellt wird. Er kann von jedem Web Service, der diesen Composite Business Service benötigt, aufgerufen werden.

Ivory Studio kann auch verwendet werden, um komplexe Composite Web Services, Outbound Web Services (Web-Service-Konsumenten) und Web Services, die Mainframe-Datenquellen miteinbeziehen, zu erstellen und SOAP-Anfragen zu bedienen.

## Zusammenfassung

Die Verwendung von Composite Mainframe Services, die durch WSDL spezifiziert werden erlaubt es, Clients verschiedenste Mainframeressourcen zugänglich zu machen, ohne sich Gedanken darüber machen zu müssen, wie die einzelnen Teilanwendungen funktionieren. Durch eine einzelne zusammengefasste Anfrage wird die Netzwerklatenz verringert und die Nutzung von Mainframeressourcen optimiert. Ein Top-Down Ansatz gibt Ihnen die Möglichkeit, In- und Outputs von verschiedensten Anwendungen, die auf der Mainframe laufen, zu definieren, sowie neue Businesslogik oder neue Prozessabläufe hinzuzufügen ohne sich von den Begrenzungen der bestehenden Anwendung eingeschränkt zu lassen. Ein Bottom-Up Ansatz ermöglicht die Nutzung der vorhandenen Programme (COBOL COPYBOOKS oder Code) als Startpunkt für die Gestaltung von Web Services. Die Verwendung des leistungsstarken Design-Wizard ermöglicht die schnelle Erstellung eines Web Services mit einer Schritt-für-Schritt Herangehensweise, durch die der Benutzer geführt wird.

Zusammenfassend liefern Ihnen ein Top-Down-Ansatz eine leistungsstarke, programmatische Lösung, ohne den sonst anfallenden Programmieraufwand. Ein Bottom-Up Ansatz beschränkt Sie auf die Geschäftsprozesse, Daten und Namenskonventionen der bereits bestehenden Anwendungen. Beim Erstellen der meisten neuen Web Service Routinen benötigen Sie aber mehr Informationen, als eine einzige Anwendung anbietet.

## Über GT-Software

GT Software ist ein führender Anbieter von Unternehmensanwendungs- und Datenlösungen die Mainframedaten und -anwendungen mittels ODBC, JDBC, J2EE und Web Services in das Client/Server-Umfeld integrieren. GT Softwares Lösungen bieten eine umfassende, flexible und nicht-programmatische Grundlage um Unternehmenssysteme und -anwendungen wirksamer und mit höherem Nutzen einzusetzen und um die heutigen Internettechnologien voll zum Unternehmensvorteil auszunutzen.

Anwendungen von GT Software sind über 2.000 mal weltweit installiert und bieten qualitativ hochwertige, innovative und kostengünstige Lösungen für viele Fortune 1000 Unternehmen. Gegründet im Jahr 1982 und mit Hauptsitz in Atlanta, ist GT Software ein inhabergeführtes Unternehmen, das derzeit in den Vereinigten Staaten, Kanada, Asien, Europa, Afrika, Australien und Süd Amerika vertreten ist.

### **GT Software**

1314 Spring Street, NW  
 Atlanta, Georgia 30309  
 Telephone: +1 (404) 253-1300  
 Fax: (404) 253-1314  
 Web: [www.gtsoftware.com](http://www.gtsoftware.com)

### **EDV-Beratung Machold GmbH**

**Systemhaus 21**  
 Nordbahnhofstr. 17  
 70191 Stuttgart  
 Telefon: +49 (711) 25772-0  
 Fax: +49 (711) 25772-22  
 Web: [www.machold.de](http://www.machold.de)