Enterprise Architecture

Vol. 11, No. 9

CUTTER CONSORTIUM

Building an SOA with Infrastructure, Application, and Orchestration Services from the Ground Up

by Max Dolgicer, Senior Consultant, Cutter Consortium; and Gerhard Bayer

A large amount of information is available describing the potential benefits of service-oriented architecture (SOA) as well as the best practices, architecture guidelines, and design patterns to achieve them. However, putting SOA to the test in a real-world project always provides the most valuable insights. This *Executive Report* discusses a comprehensive case study based on a project that was conducted for one of the world's leading chauffeured services companies.

Executive Report

ABOUT CUTTER CONSORTIUM

Access to the Experts

Cutter Consortium is a unique IT advisory firm, comprising a group of more than 100 internationally recognized experts who have come together to offer content, consulting, and training to our clients. These experts are committed to delivering top-level, critical, and objective advice. They have done, and are doing, groundbreaking work in organizations worldwide, helping companies deal with issues in the core areas of software development and agile project management, enterprise architecture, business technology trends and strategies, innovation, enterprise risk management, metrics, and sourcing.

Cutter offers a different value proposition than other IT research firms: We give you *Access to the Experts*. You get practitioners' points of view, derived from hands-on experience with the same critical issues you are facing, not the perspective of a desk-bound analyst who can only make predictions and observations on what's happening in the marketplace. With Cutter Consortium, you get the best practices and lessons learned from the world's leading experts — experts who are implementing these techniques at companies like yours right now.

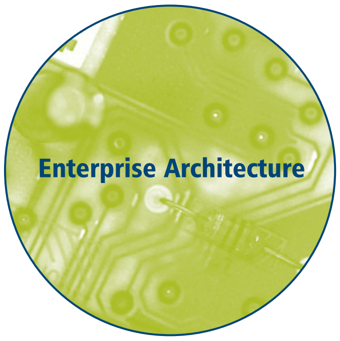
Cutter's clients are able to tap into its expertise in a variety of formats, including print and online advisory services and journals, mentoring, workshops, training, and consulting. And by customizing our information products, training, and consulting services, you get the solutions you need while staying within your budget.

Cutter Consortium's philosophy is that there is no single right solution for all enterprises, or all departments within one enterprise, or even all projects within a department. Cutter believes that the complexity of the business technology issues confronting corporations today demands multiple detailed perspectives from which a company can view its opportunities and risks in order to make the right strategic and tactical decisions. The simplistic pronouncements other analyst firms make do not take into account the unique situation of each organization. This is another reason to present the several sides to each issue: to enable clients to determine the course of action that best fits their unique situation.

Expert Consultants

Cutter Consortium products and services are provided by the top thinkers in IT today — a distinguished group of internationally recognized experts committed to providing top-level, critical, objective advice. They create all the written deliverables and perform all the consulting. That's why we say Cutter Consortium gives you *Access to the Experts*.

For more information, contact Cutter Consortium at +1 781 648 8700 or sales@cutter.com.



The Enterprise Architecture Advisory Service Executive Report *is published by the Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA. Client Services: Tel: +1 781 641 9876; Fax: +1 781 648 8707; E-mail: service@cutter.com; Web site: www.cutter.com. Group Publisher: Kara Letourneau, E-mail: kletourneau@cutter.com. Managing Editor: Cindy Swain, E-mail: cswain@cutter.com. ISSN: 1530-3462.*

Cutter Business Technology Council

Building an SOA with Infrastructure, Application, and Orchestration Services from the Ground Up

THIS MONTH'S AUTHORS

Max Dolgicer
Senior Consultant, Cutter Consortium

Gerhard Bayer

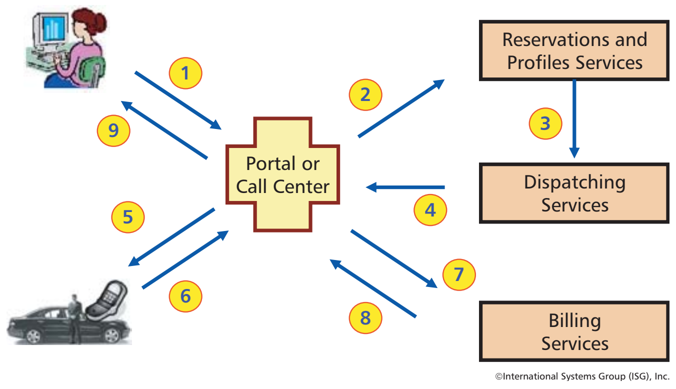
IN THIS ISSUE

THE BUSINESS CASE

This case study is based on a project that was conducted by our company, International Systems Group, Inc. (ISG), for one of the world's leading chauffeured services companies (the company wished to remain anonymous). This company operates globally, and there is no major city in the US that does not have its service coverage. The clientele of the company includes individuals and corporate clients as well as travel agencies. Its value proposition is based on delivering a highly personalized level of service and mission-critical reliability, with a high level of consistency and safety.

Figure 1 illustrates a business process that covers the typical phases of a car service booking (excluding exception processing). This business process consists of the following nine steps:

1. A customer or travel agent requests a new reservation.
2. The reservation system creates a new reservation.
3. The dispatching system picks up the new reservation.
4. A dispatcher uses the dispatching system to send a job allocation to a driver.
5. The driver receives the job allocation.



Reservations and Profiles Services
Portal or Call Center
Dispatching Services
Billing Services
1
2
3
4
5
6
7
8
9
©International Systems Group (ISG), Inc.

Figure 1 — Typical business process to book cars.

6. The driver reports end-of-job billing information back to the corporate systems.
7. The billing system receives the end-of-job information and calculates the bill.
8. The billing system sends an invoice to the customer.
9. The customer receives the invoice.

Many of these steps were not automated but were conducted through manual, error-prone activities. For example, reservation requests were received in call centers through the phone and from business partners through e-mail. The requests then had to be keyed into the back-office applications. The dispatchers needed to interact with the drivers via phone calls and had to perform manual data entry. Finally, there was no way for corporate clients to directly retrieve billing information between the regular billing cycles. A business mandate to expand into new B2B revenue channels was the major driver for this project, and the lack of automation and the cost of one-off integration solutions was an impediment for IT to meet the business requirements.

Business Objectives of the B2B Gateway

The company had been defining and implementing a new component and service-oriented architecture (SOA) for its core applications, which handles job dispatching, billing, and trip reservations. The new applications and services that are based on this architecture improve the IT capabilities "behind the firewall." At the same time, there was an urgent need to move away from mostly manual or semimanual interaction with business partners (e.g., some of the company's subsidiaries) and most importantly to enable growth of the business by entering into automated B2B relationships with new business partners. The multitude and complexity of these automated B2B interactions warranted the implementation of a B2B gateway, which shields the core applications from external partner systems and, on the other hand, provides a uniform and standardized interface that the external partner systems can access. Figure 2 shows an overview of the involved systems.

Figure 2 also shows the core applications and services (depicted to the right) that handle reservations, dispatching, and billing. The B2B gateway provides a uniform interface to the different user constituencies (depicted to the left). Those include some of the large customers who want to automate the way the company is billing them, supply chain partners (i.e., chauffeured service providers that subcontract with the company), as well as global distribution systems (GDSs) such as Sabre and Apollo, which can be accessed through a system provided by the Ground Travel Technology Team (GT3). GT3 provides technologies to automate the ground transportation industry. It is a vendor-neutral infrastructure company that delivers advanced reservations and confirmations. It connects chauffeured service providers, travel agencies, and corporations.

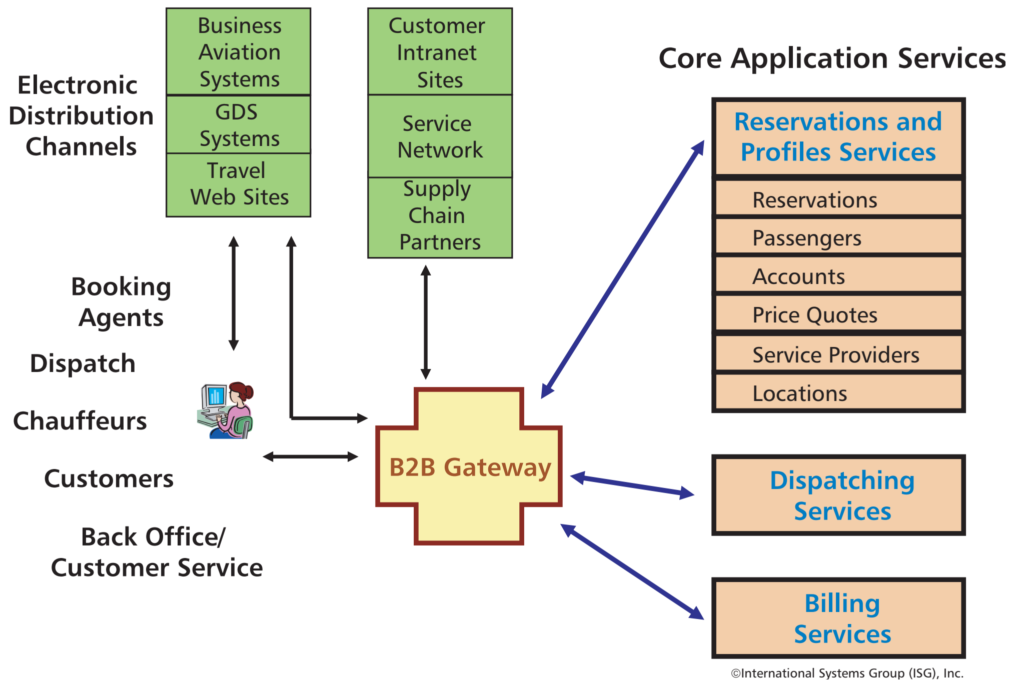


Figure 2 — Systems overview.

Furthermore, the company wanted to receive reservation requests directly from travel Web sites that offer search engines (e.g., Orbitz) and from a number of private aviation companies (e.g., NetJets).

Another requirement for the B2B gateway was to support wireless communication with chauffeurs through a third-party system ("Vettro"). This system interacts with handheld devices that the drivers use to receive, display, and respond to job allocations. On the other hand, the wireless system utilizes two-way interaction with the B2B gateway (i.e., it acts as a client of the gateway in order to relay data to the back-office applications, and it acts as a service that the gateway calls to push information to the wireless system to forward it to a driver).

In the first phase of the project, the B2B gateway enabled the company to receive reservation requests from a multitude of partners electronically and accept them into the back-office reservations application automatically. The system was required to support the following functionality:

- **Create reservation.** A request for a new reservation is received from a business partner.
- **Modify reservation.** A request for the modification of an existing reservation is received from a business partner.
- **Retrieve reservation.** A request to retrieve information about an existing reservation is received from a business partner.
- **Cancel reservation.** A request to cancel an existing reservation is received from a business partner.
- **Quote request.** A request for a rate quote is received from a business partner.
- **Provider modifies reservation.** A business partner needs to be informed about reservations that have been made originally through one of the B2B channels but subsequently the customer cancelled the reservation directly with the company.

How Business Objectives Are Fulfilled by IT

The business sponsors had agreed with the CIO on a list of high-priority objectives that they wanted to achieve through their SOA effort. Table 1 provides an overview of these business objectives and how they were addressed by IT, in particular through the migration to SOA.

Some examples of the business objectives include new service offerings (e.g., sharing a limousine service with another customer and cross-selling by enabling a service rep to offer a customer who is going to the airport car service at his or her destination city). In terms of improving customer service, the company wanted to offer direct access to billing information for its corporate customers as well as reduce pricing inconsistencies that were a result of duplicated business logic in different applications that handle price quotes versus invoicing.

Total cost of ownership (TCO) and risk management were objectives that originated on the IT side. Streamlining the application portfolio and reducing the

Table 1 — Business Objectives Addressed by IT

Business Objectives	Addressed by IT
Revenue enhancement: • New service offerings • Cross-selling • Acquisitions	• Better leverage of the IT infrastructure • Increased extensibility and adaptability of the core applications • Avoid one-off integration solutions
Customer service enrichment: • Single view of the customer • Better customer self-service • Improved business process consistency	• Creation of a centralized reservation system with customer profiles • SOA enables efficient integration of a portal with back-office systems • Centralized SOA leads to improved business process consistency
Allow travel agents to book with The Company using major GDS systems	Automated B2B integration with major GDSs and other types of business partners
TCO	Service reuses, decreased complexity
Risk management	Core services (foundation and some application) are implemented by highly skilled designers/developers

integration points should reduce the TCO, while the separation of services into clearly distinct categories allowed utilizing highly skilled software engineers for mission-critical service implementations (e.g., services that deal with low-level infrastructure coding versus those that are mainly concerned with business logic).

SOA takes a process-centric approach to development, and the focus should be on the composition of process flows that orchestrate services into a business process.

Project Scope

The overall goal of the B2B gateway project was to implement a central gateway in such a way that business partner integration becomes a repeatable, low-risk effort that allows the company to capitalize on partnering opportunities in a timely fashion. The scope of the project included:

- **Encapsulation of the legacy reservation system.** Since the legacy reservation system was being migrated to a new system in several phases, it was essential that business partners were shielded from any incompatibilities that may be caused during the migration phases.
- **Connectivity to GT3.** The GT3 GDS presented a substantial business opportunity for receiving automated reservation requests. The B2B gateway needed to support the proprietary B2B protocol employed by GT3.
- **Definition and implementation of a B2B protocol standard for the chauffeured services industry.** Many business partners want to utilize a standard, Web services–based protocol that the company defines and maintains. To that end, the company has defined the Chauffeured Service Interface (CSI), which is a collection of XML schema–based services for the B2B interactions that pertain to reservations (e.g., create and retrieve).
- **Definition of a common XML-based data architecture.** The diversity of systems that are connected to the B2B gateway (i.e., partner systems, legacy systems, and newly developed applications) necessitated the definition of a comprehensive data architecture with clearly defined data-format boundaries and mappings between proprietary or legacy formats and standard formats.
- **Integration with a third-party wireless system.** Chauffeurs use a third-party wireless system so their smart phones can connect to the company's corporate systems. The B2B gateway needed to mediate between the third-party system and the back-office applications.
- **Implementation of an SOA foundation.** The items listed above constituted the primary goals of the project. However, the company wanted to capitalize on the opportunity and build an SOA foundation consisting of common services that could be reused across several business partner integration solutions.

ADAPTING A TRADITIONAL DEVELOPMENT METHODOLOGY FOR A SERVICE-ORIENTED SDLC

Building a successful SOA requires an appropriate development methodology. Most development organizations have been utilizing a "traditional" methodology for their object or component-based projects. The major software development project tasks in an SOA project are the same as in traditional (component-oriented) projects. They can, for example, be governed by a RUP approach. However, the traditional RUP needs to be extended in order to address the SOA-specific issues. Figure 3 shows the typical breakdown of RUP activities and phases.

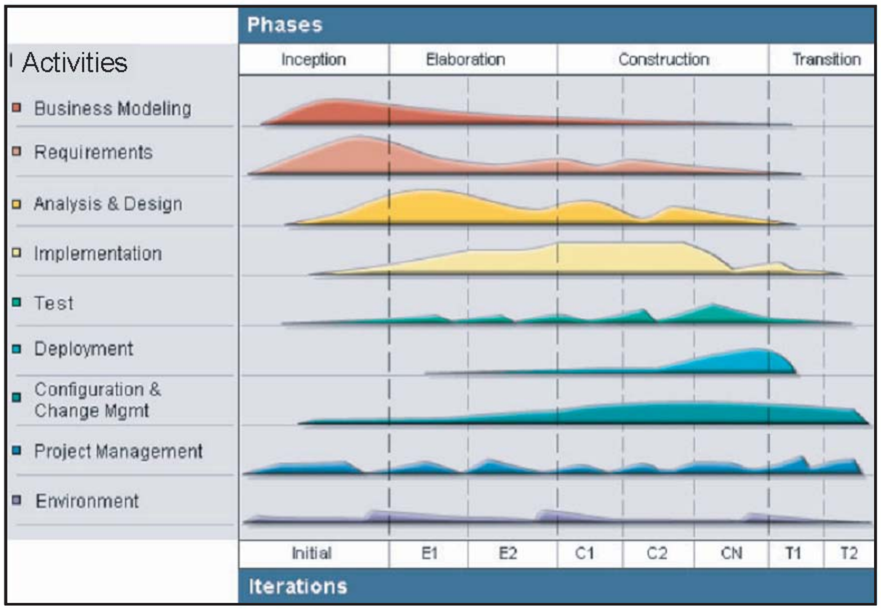
At the center of traditional development methodologies are the concepts of OO analysis and design (OOAD), which usually focus on a level of granularity that is too small compared to what is needed for service-oriented analysis and design (SOAD). In other words, class-level modeling doesn't fit well with business-service modeling, and SOAD must be predominantly process-driven. A service-oriented development methodology must combine OOAD and SOAD since services and components need to be combined for a complete solution implementation.

Adaptation of the Activities

This following sections address the RUP activities one by one and provide an introduction to the adaptations that are required to make them feasible for developing service-oriented applications.

Business Modeling Activities

SOA takes a process-centric approach to development, and the focus should be on the composition of process flows that orchestrate services into a business process (in contrast, OO development is focused on component design). An SOA that follows this approach should lend



©International Systems Group (ISG), Inc.

Figure 3 — RUP activities and phases.

itself for better alignment with the requirements of the business. However, aligning business strategy to IT has often failed due to lack of traceability from business models to IT architecture and implementation. A successful SOA must start with an architectural representation that the business sponsors can understand. Next, one has to create an audit trail from business models through IT architecture to project implementations.

In many cases, the problem with this approach is how to obtain the business models. Ideally, they would have been created before a particular IT project is even initiated. They should identify coarse-grained business functions and processes first, which might span several application areas or lines of business, before they drill down into detailed business processes that will be the focus of a specific IT project. An example of a coarse-grained business function is "inventory management," which should be defined so that it includes an identification of the owner of the business function; his or her role and responsibilities; the priorities in fulfilling this function (e.g., minimize inventory); the actors that relate to this function; the tasks that the business function performs (e.g., shipping and restocking); and pre- and post-conditions for each task.

Although the company had already mapped out its major business processes, it was mostly concerned with the operation of the back-office systems. There were only limited, manual B2B interactions in place, and the majority of B2B business processes were defined during the B2B gateway project. This was a workable scenario since it allowed IT to cooperate with the business side on the creation of the process models.

Requirements Activities

Since a major focus of SOA is reusability, the requirements activities need to account for a wider scope than in traditional development methodologies. They should at least take into account cross-project requirements, and, in some cases, address the enterprise as a whole. This necessitates bringing a variety of different stakeholders to the table, including business owners, project managers, and IT managers.

The company had already negotiated with a number of potential business partners regarding the integration of their respective reservation systems. Therefore, high-level requirements were known, and for some of the partners, even detailed requirements such as the specifics of the B2B interfaces and communication protocols had been laid out.

In general, setting the project scope becomes a more critical activity. This is more important in SOA since addressing the immediate business requirement (e.g., of one particular line of business) has to be balanced against building services that are of enterprise value for later reuse. Business process–agnostic requirements need to be determined, particularly for the services that will be designed as application or infrastructure services.

Analysis and Design Activities

The business models should guide the service-modeling process so that high-level service modeling can be performed with a focus on business processes. This is an area where SOA provides an opportunity to improve the alignment of business and IT, and it should be emphasized during the analysis and design activities.

Next, candidate services should be defined, classified, and separated into the three layers of the SOA, namely orchestration (business) services, application services, and infrastructure services. Functionality that is logically related should be grouped into one application or infrastructure service, and the service interfaces should be defined, including the service contracts and any required service-level agreements (SLAs). At the same time, an inventory of existing services that can be reused for the project has to be created (this could potentially include services outside the enterprise), and the potential for encapsulating legacy systems as services needs to be analyzed.

An issue to consider when wrapping legacy systems is the impedance mismatch that often arises between the design principals of the SOA and how the legacy application was designed.

The B2B gateway project reused several existing services that had been developed as part of the core application project (e.g., the security service and the transformation service). The project also capitalized on the opportunity to reduce some of the redundancies in the existing applications by introducing the City of Service Calculation service, which could then be used by several applications both within the B2B gateway as well as the core applications.

During the analysis and design of services, it is important to *repeatedly* verify that the core principles and best practices of SOA are being adhered to. The major SOA principles are self-containment, reusability, statelessness, and loose coupling.

Self-containment is achieved by well-defined boundaries of application logic (business logic) that is only provided by one particular service, not by another service or application. In the case of a wrapper service that encapsulates a legacy application, there is only partial self-containment since the legacy application is used (without the wrapper) by its legacy clients.

In order to achieve *reusability,* it is important to distinguish between orchestration, application, and infrastructure services and to keep an eye on future reuse (i.e., by designing functionality that goes beyond the requirements of one project such that broader reuse opportunities are enabled).

Services should be *stateless*. This refers to client state and not service state, meaning that a service should not remember at what point a client is in regards to a sequence of service invocations. Whatever information is required to identify a particular step in a conversation should be passed to the service by the client.

Loosely coupled services minimize the dependencies between service provider and service requestor. This allows new business processes to be composed easier and more quickly from existing services and existing business processes to be adapted to new requirements faster since there are fewer dependencies between services.

Another service-oriented design activity is the encapsulation of legacy applications, which is achieved by building wrappers that provide a service-based interface to access existing application logic. However, in many cases, this will require restructuring the legacy applications (i.e., breaking legacy applications apart in order to achieve a modularization that is aligned with the service model). This may require a substantial effort and as such necessitates an ROI estimate. Also, it is usually not desirable to maintain two code bases for the legacy application (i.e., one based on the modularization for integration with the SOA and another code base to support the existing clients of the legacy application).

Another issue to consider when wrapping legacy systems is the impedance mismatch that often arises between the design principals of the SOA and how the legacy application was designed. Examples include synchronous service invocation versus asynchronous application processing and the SOA mandate of stateless services versus the design of stateful legacy applications.

Furthermore, reusing existing business logic and data stores to create services often requires a strategy of how to deal with business logic and data that reside in multiple places. This must be addressed through an enterprise-wide analysis to determine what system would be best suited to provision the service such that duplicate systems could be retired in the long term and which system is considered the “master” (i.e., the authoritative source of business rules or up to date data). One of the solutions that can be applied is to introduce a form of rationalization via the service wrapper (e.g., to transform legacy data into a new enterprise standard representation).

One of the major challenges of the B2B gateway project was the different data formats employed by the various legacy systems, which was amplified by the different formats that one of the first business partners required. The challenge was not just the data formats; a bigger difficulty was to determine the semantics — the exact meaning and requirement for information of some of the legacy systems was not documented and had to be obtained from the original application developers in a rather time-consuming process. A more detailed explanation of the construction of a "wrapper" for the legacy reservation system is covered in a later section.

Since a major focus of SOA is adherence to standards, the decisions made in the analysis and design phase need to follow the standards that have been established by the overall enterprise architecture (e.g., based on XML schema, WSDL, and SOAP).

Services typically communicate via XML-based messages, which essentially represent the interface of the service. The messages/interfaces need to be defined as part of the analysis and design activities. In a sense, they can be compared to class interfaces, but they have to be designed in a context that goes beyond one service. They should be decomposed into a hierarchy of reusable message component layers so that new services can be defined more rapidly through these message components (this will be discussed in more detail in a later section).

Implementation Activities

During the implementation activities, the platform-specific support for services in general, and Web services in particular, need to be taken into account. Examples include the automatic mapping from SOAP-based Web service invocation onto method execution of components that is provided by all major application servers, the capabilities of the Windows Communication Foundation (WCF) that is part of .NET, and the host of functionality that is available today in the form of open source software (e.g., Enterprise Service Bus).

Some of these platform-specific characteristics obviously need to be considered during the analysis and design activities as well. Another example is the question as to whether the orchestration services should be implemented in a regular programming language or if high-level tools should be employed (i.e., BPEL/BPMN-based tools).

The initial orchestration services of the B2B gateway project only dealt with reservation requests from business partners. They implemented fairly simple business processes, and as such, didn't warrant the deployment of a BPEL tool. However, since they contain a relatively small amount of Java code, it would be easy to migrate them into a tool, should the company make a strategy decision to implement business processes using BPEL.

Test and Deployment Activities

Since a major focus of SOA is reusability, the testing phase needs to account for usage scenarios that go beyond the requirements of the current project. This includes different types of clients, a much larger variety of exception scenarios, interoperability requirements, and so on. Not all service requestors and usage scenarios will be known to the service development team. Therefore, particular focus must be given to test service versioning, and a greater emphasis on performance and scalability tests is required. The deployment issues are similar to those that have to be considered during testing.

During the B2B gateway project — as with any B2B project — the company had to manage two release cycles: that of its own software and the cycle imposed by the business partners. Managing software release cycles is also a good example of the benefits that can be achieved with an SOA that follows a proper separation of concerns and a clean service layering.

TOP-DOWN VS. BOTTOM-UP APPROACH

The question as to whether a project should be approached with a top-down or bottom-up strategy is not part of any particular RUP activity. Rather, it leads to a decision that determines which activities might be cut short and which activities might deserve greater emphasis. Service-oriented development is significantly different from component-based development in this regard since it focuses (or should focus) on business processes (i.e., orchestration) and not just the codification of business functions.

Top-Down Strategy

A top-down strategy typically starts with a business process analysis, focusing on alignment of the SOA with the business models. This analysis includes business processes, business functions, and business owners and their roles and responsibilities. It depends on the availability of business models, or at least well-defined and well-documented business processes. This analysis feeds into the modeling of orchestration services, which will form the implementation of the business models.

The definition and design of new services (or reuse of existing ones) that will provision the required business functionality that are orchestrated into business processes follows later. This means that the service design is an outcome of the business process modeling, and the services should therefore not determine the overall structure of the SOA.

In addition, the service interfaces (i.e., data) should be aligned with an enterprise data model, meaning that the service interface schemas should follow existing data models — again, the enterprise models come first; the top-down strategy does not encourage a service design to come up with its own data model. This ensures that service semantics are aligned with enterprise standards. In the long run, it will simplify data transformation requirements.

Bottom-Up Strategy

A bottom-up strategy typically focuses on application-centric requirements of one particular business process or one project. This strategy is also often employed in an application integration context, which includes building wrapper services for legacy systems and using auto-generation of service wrappers for recently developed component-based applications.

This approach focuses on fulfilling particular requirements of one (narrow) project, usually within aggressive time and cost constraints. It does not consider the bigger picture of an SOA; rather, it starts with the service design and often propagates legacy structures into a flawed SOA. Table 2 summarizes and compares the pros and cons of the two strategies.

THE B2B GATEWAY ARCHITECTURE

The SOA for the B2B gateway has been developed following best practice guidelines for service analysis, modeling, and design as outlined in the previous section. Key architectural principles that were applied are the separation of concerns and design for reusability. This is achieved by a separation of functionality into layers.

Service Layering

The breakdown into distinct layers facilitates decoupling of the services. A typical service layer model for SOA is comprised of an orchestration services layer, an application services layer, and an infrastructure services layer, as illustrated in Figure 4.

The top layer of the SOA is comprised of orchestration services. An orchestration service acts as a controller, composing application services and infrastructure services to implement a business process. For the most part, an orchestration service is made up of workflow

Table 2 — Top-Down vs. Bottom-Up Approach

	Top Down	Bottom Up
Pro	• Achieves business and IT alignment • More likely to create an SOA that is not constrained by legacy architectures • Enables standardized, repeatable, enterprise-scale integration solutions • Can align services with the enterprise data architecture • Ensures that key goals are met in the long term (development and operational efficiency, business efficiency)	• Cheaper and faster, since there is less up-front analysis and design effort • IT can demonstrate quick, albeit perceived, success
Con	• High up-front cost • Lengthy timelines could conflict with business demands • Often difficult to obtain required information (e.g., business models).	• Achieves only limited reuse; creates service silos • No adherence to enterprise data models • No strategic alignment between business and IT • Recreates existing architectures that are not truly SOA • Proliferates more one-off integration points

logic and calls to lower-level services. This allows orchestration services to be changed and adapted to new business process requirements without affecting the underlying application and infrastructure services.

The middle layer is comprised of application services that implement business logic. They should represent a business entity such as a reservation or an invoice, including the operations that can be performed on that entity (e.g., create new reservation, cancel existing reservation, or retrieve invoice). If designed properly, they will be aligned with the corporate business model and they can be reused in any business process that requires access to that business entity.

The bottom layer of the SOA contains infrastructure services. They implement technology-specific functionality (e.g., security, transformation, communication, and persistence service) and do not include business logic.

Conceptually, one can think of components and legacy systems forming an additional layer below the three services layers. They represent popular choices for implementing services.

The interactions between services consist of calls from a higher to a lower layer in the stack, as illustrated by the thick arrows in Figure 4. For example, an orchestration service can call an application or infrastructure services but not the other way. It should be noted that orchestration services should not invoke components or legacy systems; those should be encapsulated by application services. Furthermore, the services within one layer can call each other, which allows for aggregate services to be developed.

High-Level B2B Gateway SOA

These architectural guidelines were applied to the definition of the company's service-oriented B2B gateway architecture. Figure 5 provides an overview of the service layers of the B2B gateway, including some of the services that are part of each layer, and the interaction between the layers. The following sections offer a brief description of each layer.

Business Partners Layer

Business partners use their own proprietary B2B client software to connect to the company over the Internet using different means of communication. The logical entity consisting of a B2B client and the software within the B2B gateway that supports a particular B2B client is also referred to as a "channel." Figure 5 shows three different partners: Vettro, the wireless service used to communicate with drivers in their cars; GT3, the GDS that sends reservation requests to the company; and "Future Partners," which denotes a number of B2B relationships that the company has been negotiating.

Orchestration Services Layer

The interaction with each business partner is managed by a particular orchestration service (e.g., GT3, CSI).

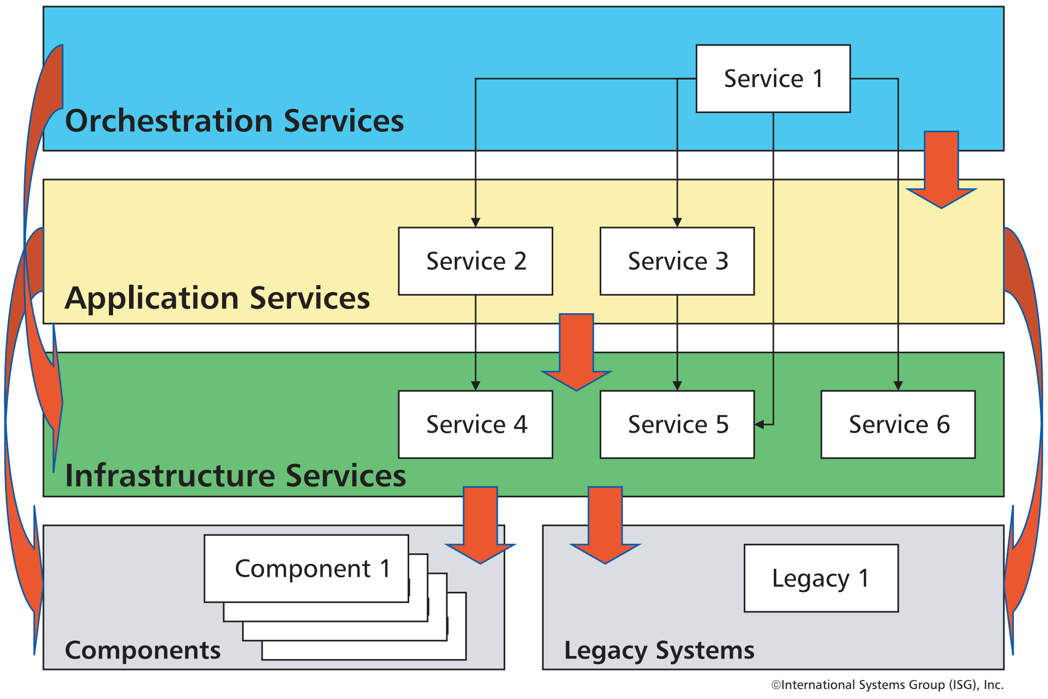


Figure 4 — Service layering.

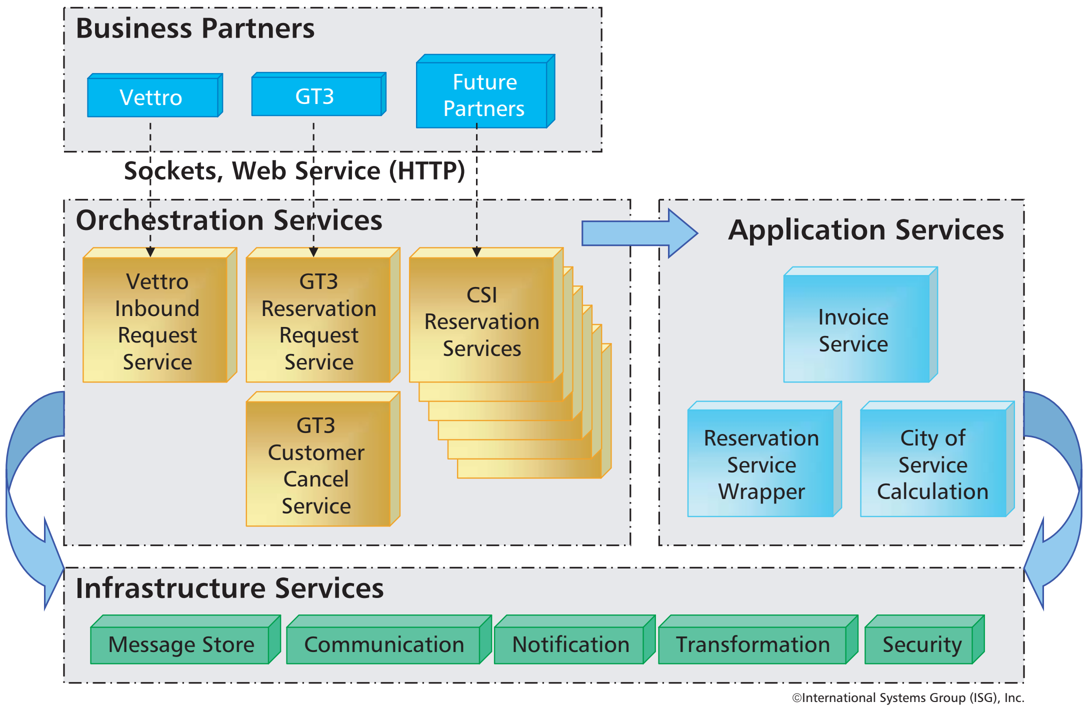

Business Partners
Vettro
GT3
Future Partners
Sockets, Web Service (HTTP)
Orchestration Services
Vettro Inbound Request Service
GT3 Reservation Request Service
CSI Reservation Services
GT3 Customer Cancel Service
Application Services
Invoice Service
Reservation Service Wrapper
City of Service Calculation
Infrastructure Services
Message Store
Communication
Notification
Transformation
Security
©International Systems Group (ISG), Inc.

Figure 5 — High-level B2B SOA.

Although each channel implements somewhat different functionality, all channels follow the same high-level pattern: they consist of an orchestration service that manages the flow of business transactions between the company and a partner for the typical reservation requests (e.g., create, modify, and cancel reservation). The orchestration service also maps the reservation requests to activities that are internal to the company, that is, they coordinate which application services (e.g., the Reservation service or the City of Service Calculation service) and which infrastructure services (e.g., the Security service) have to be invoked and in what sequence.

The Vettro Inbound Request service implements a business process that receives messages from chauffeurs via the Vettro wireless network provider. The flow logic in this orchestration service and the business logic that is subsequently executed is completely different from all the reservation processes, yet the structure of the SOA ensures that the Vettro service fits seamlessly into the architecture.

Application Services Layer

The application services are agnostic of the specific partner and can be called within the business process context of any of the B2B channels. This enables reusability and allows adding new partners with rapid time to market. The Reservation service encapsulates some of the functionality of the legacy reservation application and makes it available through a high-level, service-oriented API, thus creating an abstraction to the legacy system. It allows different B2B channels to use a common interface for making reservations. This abstraction of the legacy reservation system delivers the following benefits:

- The ability to provide an easier-to-use interface for making or changing reservations
- The ability to enhance or change underlying business rules without changing the business logic in the B2B channels
- The ability to minimize the exposure of the company's internal business rules to the B2B channels

The company uses the concept of City of Service (i.e., for each reservation, the closest service center is determined based on information that is provided by the channel partner as part of the request to create a new reservation or to change an existing reservation). This process is hidden from the B2B channel partners; as a result, each channel has the requirement to calculate the correct City of Service, and this calculation is performed in the City of Service Calculation service, which is part of the application services layer and thus business process–agnostic and reusable across all channels. Finally, the Invoice service encapsulates the billing system in order to provide external partners access to their invoices.

Infrastructure Services Layer

The infrastructure services are also partner-agnostic; they can be called within the business process context of any of the B2B channels (i.e., from any orchestration service). This also reduces time to market and cost to implement new partner integrations.

Message Store Service

All requests received from business partners are written to persistent storage (message store) in their raw, unaltered format in order to maintain an audit trail of all communication with business partners. All response messages sent from the company to business partners are written to the message store as well. The Message Store service is used to maintain the message store. The supported functionality includes store messages in the message store, and the messages can be of different types — for example, request message, response message, as well as pertain to different channels (i.e., have various formats); retrieve messages from the store; and perform a search for messages based on a set of criteria.

Communication Service

The Communication service provides an abstraction to different communication protocols. When two services need to communicate with each other, they use the API of the Communication service to exchange data in the form of the common XML format. As a result, the services are not concerned with the intricacies of the underlying communication protocols. The Communication service supports data exchange over HTTP, SOAP, RMI, and JMS.

Notification Service

The Notification service can be used to inform an internal or external user of a business event (e.g., a problem with the data that has been received in a reservation request). The notification can be sent via different communication media (e.g., through e-mail or fax). The Notification service is an example of a service that has been developed as part of the core applications project and was reused in the B2B gateway project.

Transformation Service

The Transformation service translates data from the format that a particular business partner uses to the company's internal common XML format. It also transforms between the common XML format and the proprietary format of the legacy reservation system. The Transformation service can handle different data formats, including XML-based data and binary data. Furthermore, since the data that is supplied by a partner in a reservation request in many cases needs to be enriched with data that is stored in the company's databases, the Transformation service performs database lookups to retrieve the appropriate data for the enrichment.

Security Service

The functionality provided by the Security service includes the following: validating the user ID and password provided by a business partner; validating the certificate provided by the requestor to make sure that it is the business partner represented in the request (since this was not required for the first release of the B2B gateway, the functionality has been designed but not implemented); and encryption of the entire response message that is to be sent to a business partner. The Security service is another example of a service that had been developed as part of the core applications project and then reused in the B2B gateway project.

Service Layer Example for GT3

Figure 6 shows the service model of the GT3 channel as an example of the service layering in the B2B gateway. This service model implements a business process where the GT3 system sends a request to the company in order to create a new reservation. The process returns a reservation number or a negative response code accompanied by a textual explanation to GT3.

The GT3 Reservation Request orchestration service is the only service in this model that is specific to the GT3 B2B channel. It contains business logic that implements the particular requirements of the GT3 interface and uses common services (Reservation, City of Service, Message Store, Security, Transformation, Notification, and Communication) to conduct tasks that are not specific to GT3. In addition to invoking the appropriate application and infrastructure services, the orchestration service performs the following functionality:

- **Checks for duplicate message.** The service checks if a particular message has already been received from GT3 before and performs actions according to business rules.
- **Checks if data is valid.** The system checks if the data in the reservation request XML document received from GT3 contains all required information and if the data is valid.
- **Checks the response from the reservation service.** The system checks if the reservation service performed the requested action or declined it and formulates a response to GT3 accordingly.

Approach to Business Partner Integration

The B2B gateway supports business partners that want to utilize automated reservation requests, as well as other automated B2B interactions such as billing or obtaining car status information. From a technical perspective, integrating with a business partner requires addressing three issues:

1. **Data.** Some business partners use different data formats, which require data transformation/translation from the partner's format to the company's internal standard format.
2. **Connectivity.** This involves establishing a connection with a business partner over the Internet using different mechanisms (e.g., HTTP, SOAP, or Sockets).
3. **Business process.** This involves implementing the business rules that a partner follows in order to conduct a specific transaction (e.g., to modify a reservation or to cancel a reservation) and mapping this process to internal activities.

These business partner integration issues can be considered a pattern, and the layered SOA of the B2B gateway is structured to match this pattern. As a result, any new B2B integration becomes a repeatable and thus predictable effort, and each new project benefits greatly, both in terms of cost and time to market, by following the architecture pattern. In addition, the company has strived to standardize the B2B communication protocols and data formats.

Some of the partners demand that the company adapt to their particular B2B protocol. One example is GT3. Since GT3 connects many chauffeured service providers, travel agencies, and corporations, it did not want to adhere to a particular B2B protocol that the company would want to utilize. On the other hand, there were many business partners who wanted to utilize a standard, Web services–based protocol that the company maintains. To that end, the company has defined the CSI. The CSI allows the company and its partners to integrate their respective reservations systems in a consistent, reliable, and cost-effective manner. The benefits of this standardized automation include:

- Reduced cost, effort, and risk related to building customized interfaces with channel partners
- Improved customer service resulting from best practice business processes reflected in the CSI consistently applied across an ever-increasing number of reservation requests from different partners

The CSI server, which is part of the B2B gateway, exposes an API that the CSI client (i.e., a business partner) invokes in order to make a new reservation, cancel an existing reservation, and so on. Associated with each type of request is a well-defined, XML-based message structure that contains all the information that is required to conduct one particular business transaction between a business partner and the company. The CSI provides specifications for the following components:

- Internet protocol–level connectivity
- Security (authentication and authorization of business partners, message encryption)
- Definition of business transactions (e.g., create reservation and cancel), as well as request and response message structures for each transaction

Legacy System Integration

Since the legacy reservation system was being migrated to a new system in several phases, it was essential that business partners were shielded from any

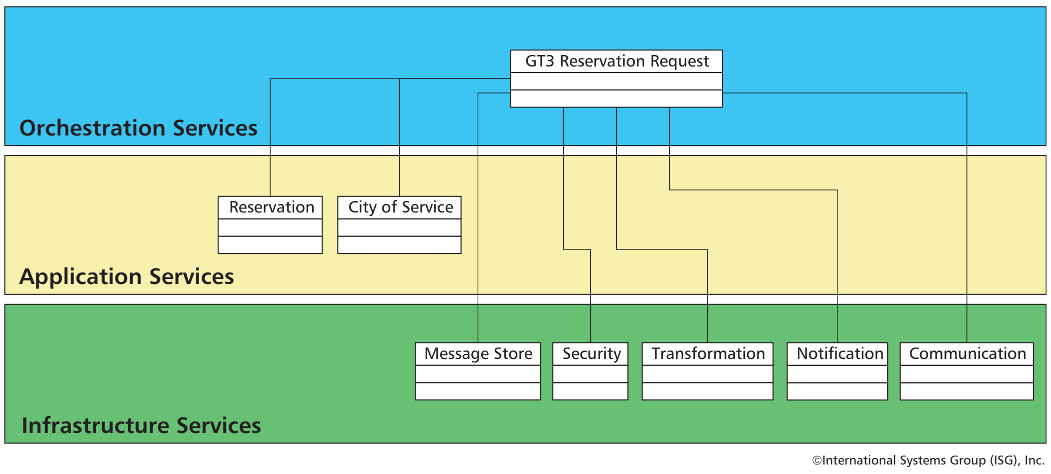


Figure 6 — Service layer example for GT3.

incompatibilities that might have been caused during the migration phases.

The Reservation service encapsulates some of the functionality of the legacy reservation application and makes it available through a high-level, service-oriented API, thus creating an abstraction to the legacy system. It allows different B2B channels to use a common interface for making reservations. In addition to the B2B channels, the new interface to the legacy reservation system is also used by different client applications, specifically a Swing-based client and a Web client. Figure 7 shows how the different client implementations and B2B channels utilize the new reservation service API (referred to as the unified code base) and compares it to legacy Web front-end code.

Exposing specific business functions of the legacy reservation system as services was not a simple matter of implementing wrapper code around a number of object-based APIs. The reservation system was tightly coupled to a Web application, such that the code consisted of intermingled presentation logic and business logic. This logic had to be broken apart, restructured, and partially reimplemented in order to create a service wrapper, which could then be used not only by the B2B gateway, but also by the Swing client and the Web front end.

Service Reusability

Reusability is achieved by designing application services and infrastructure services that are autonomous and agnostic of the business process context within which they are executed. This increases the potential for reusing them when a new business process is composed. Orchestration services typically have a limited potential for reuse since they implement particular business processes. However, when business processes are aggregated as subprocesses into larger business processes, there is also an opportunity for reuse of orchestration services.

Application and Infrastructure Services Reuse

The reuse of application services is based on identifying functionality that is common to a particular business domain. The B2B gateway is not specific to any particular business domain, and it can be considered more of a middleware system than software that is related to business applications. The number of application services that have been developed as part of the B2B gateway project is therefore limited.

There are two application services that have been developed specifically for this project but are targeted for enterprise-wide reuse (in fact, they are deployed outside of the physical domain of the B2B gateway): the Reservation Service wrapper and the City of Service Calculation service. An additional application service, the Invoice service, was defined during the project but scheduled to be implemented in another project.

In order to illustrate one of the design decisions that effect reusability, consider the Reservation Service wrapper. Besides other logic, it also contains business logic that handles requests to change an existing reservation. Under certain circumstances, a change reservation request will require a cancellation of the existing reservation followed by a rebooking (i.e., creation of a new reservation). The initial design consideration was based on the requirements of only one business partner (GT3), and one option was to put this logic into the GT3 orchestration service, which would coordinate cancelling the existing reservation and creating a new one.

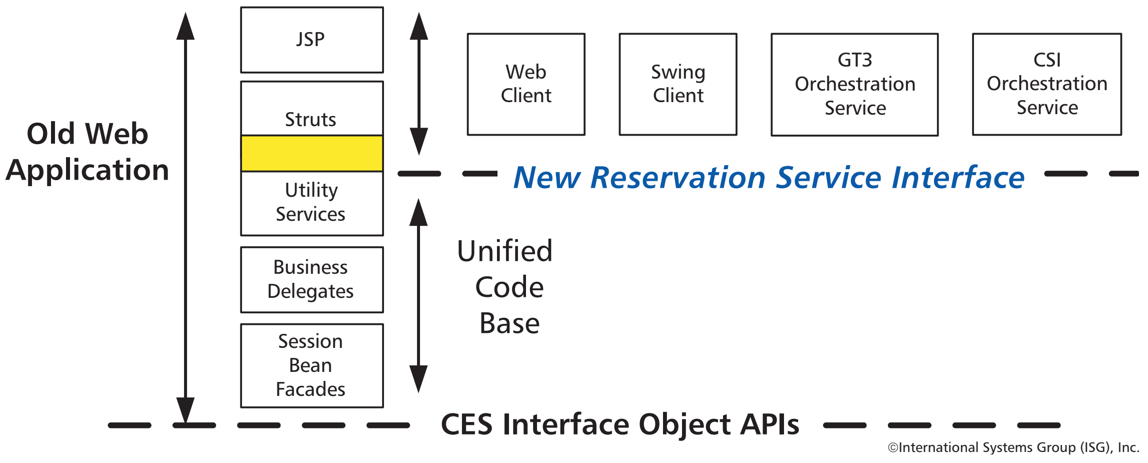

Old Web Application
JSP
Struts
Utility Services
Business Delegates
Session Bean Facades
Web Client
Swing Client
GT3 Orchestration Service
CSI Orchestration Service
New Reservation Service Interface
Unified Code Base
CES Interface Object APIs
©International Systems Group (ISG), Inc.

Figure 7 — Legacy system integration.

However, this orchestration service is specific to the GT3 business process, which makes it nonreusable for other business processes (e.g., the CSI channel). It was therefore decided to implement this business logic in the Reservation Service wrapper (i.e., an application service), which is business process–agnostic and can be reused across all channels that are part of the current or future projects.

Infrastructure services generally have the highest reuse potential since they provide functionality that is needed in most applications.

The City of Service Calculation service is another example of an application service that is highly used by several business processes. Although it was designed and implemented as part of the B2B gateway project, the requirements of other applications that would be using it in the future had to be considered.

Infrastructure services generally have the highest reuse potential since they provide functionality that is needed in most applications. The degree to which the infrastructure services have been reused depends on the customization that was required for the different business processes. Some infrastructure services are very generic in nature; the Message Store service and the Notification service, for example, are always utilized in the same way. The Transformation service, on the other hand, does require a fair amount of customization since it provides data transformation functionality that goes beyond self-describing formats like XML. It has to deal with a variety of proprietary legacy data formats.

An important consideration for enhancing the reuse potential is to anticipate and model additional functionality that goes beyond the requirements of the current project. One example is the Security service. The first phase of the B2B gateway project only required authentication based on user ID and password. The capability to authenticate a business partner through certificates was anticipated and considered during service modeling. Another example is the Notification service, which only had to support user notifications based on e-mail but was modeled so that it could also send out notifications to a pager.

Reusing the B2B Gateway for the Vettro Project

As described earlier, the Vettro system is utilized to facilitate interactions between the drivers in the field and the back-office systems, and the role of the B2B gateway is to function as a conduit. At first glance, there doesn’t seem to be much similarity between the functionality of the Vettro system (dispatching drivers, exchange of status and billing information) and the primary goal of the B2B gateway (integrating with the reservation systems of B2B partners). However, the Vettro project benefited from the B2B gateway SOA in three ways: (1) by reusing the architecture as a blueprint; (2) by reusing individual services; and (3) through reuse of schema components (i.e., data definition).

Although it is always difficult to quantify the benefit of reusing an architecture as a blueprint for a new project, it did become very obvious how the well-defined SOA of the B2B gateway accelerated the design of the Vettro integration. There were a significant number of architecture patterns in place that determined most of the design. These included exposing a service to a partner system as a Web service based on XML over HTTP communication, the separation of the required services into the three layers of the SOA, the boundaries of the data architecture, and the interaction with the back-office systems, to name a few.

The Vettro project could also reuse several of the existing services, like the Security service and the Transformation service. Finally, there was a high level of reuse in terms of the data architecture (i.e., the XML schemas). This may sound surprising since dispatching drivers is a very different business function compared to managing reservations. However, many of the essential data entities are the same in both cases. Examples include basic customer information (e.g., name, address, and contact information), pick-up and drop-off addresses, pricing, flight information, and so on. The key design element that made this level of reuse possible was to break down the XML schemas in small entities (e.g., an address), which then allowed composing many different types of schemas based on these core entities.

Reusing the SOA foundation that has been laid out by the B2B gateway for the Vettro project is one example of how service reuse can improve the responsiveness of IT to the requirements of the business. The next section illustrates in more general terms how service reusability relates to business agility.

Service Reuse and Business Agility

There are different approaches in how SOA can be employed to facilitate the alignment between business and IT. Of particular interest to the company is the increase of business agility, which an SOA can facilitate through reduced application portfolio complexity and through service reuse.

Most companies face an increasing complexity of their enterprise application portfolios. They fulfill business demands by adding new applications and packages and by building more connections between systems in order to achieve integration. Therefore, the application portfolio complexity increases, which in turn slows down the responsiveness of IT to business requests. The result is a negative effect on business agility since many IT organizations spend most of their budget on maintenance instead of innovation.

The company has experienced a similar evolution and is in the process of reducing the total number of applications by about 50% while at the same time restructuring business functions into reusable services. SOA allows streamlining the application portfolio by reducing redundancies among different applications and by simplifying the connectivity across internal and external enterprise boundaries through standardization. A key contributing factor to achieve this is the reusability gained by a thorough design and implementation of service layers. This is exemplified in Figure 8.

As illustrated in Figure 8, the company is implementing more application services and infrastructure services across a number of projects, including both the core application projects and the B2B gateway project. Essentially, a service repository is built over time, which eventually will comprise most of the lower-layer services (i.e., application and infrastructure services) that are required by the majority of business processes. This allows rapid implementation of new business processes as well as easy modification of existing processes.

New business processes are implemented as new orchestration services. As a result of the expansion of the service repository, a high degree of reusability is achieved and none or just a few additional application and infrastructure services need to be developed, thus speeding up the delivery of new business processes (or applications, in traditional terminology). Furthermore, services implemented for use by internal processes can be reused for B2B processes. (It should be noted that external partners do not utilize specific application or infrastructure services directly but connect to orchestration services that have been exposed for external use.)

"REST-like" but Not "REST-ful"

Web services–based systems that follow the concepts of representational state transfer (REST) have enjoyed a lot of publicity and attention over the last couple of years. A growing number of high-profile companies are providing their services in a REST form including Yahoo!, eBay, Amazon.com, Flickr, and YouTube.

REST is an architectural style for distributed hypermedia systems (i.e., the Web). The term originated in a

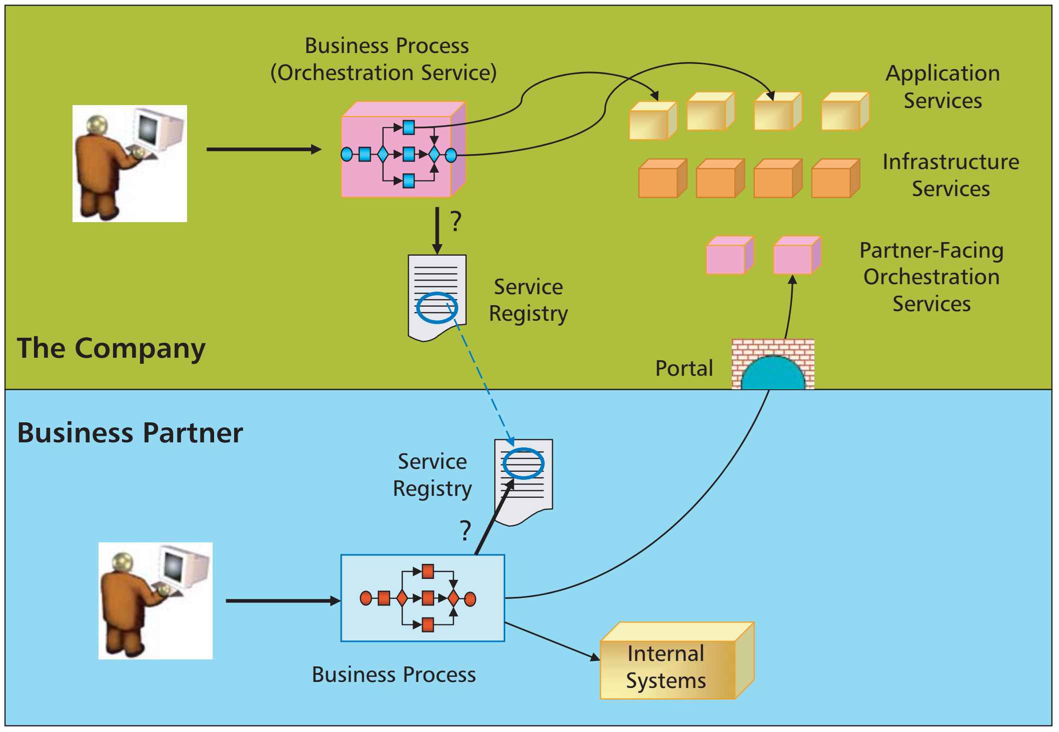


Figure 8 — Service reuse and business agility.

2000 doctoral dissertation about the Web written by Roy Fielding, one of the principal authors of the HTTP protocol specification.[1] The basic concepts are *addressable resources* (instead of functions in the traditional Web services model) and *uniform interfaces* (instead of custom interfaces). Resources are entities (e.g., a bank account) that can be accessed through a standard linkage, specifically through URLs. A REST-based system therefore benefits from the proven addressing scheme of the Web. Another characteristic of resources is that they should not maintain client state, which exploits the capability of the Web to be highly scalable.

The second concept of REST is uniform interfaces. Client-server systems usually have custom interfaces. For example, a "bank account" Web service interface is different from the "rental car reservation" Web service interface, and in distributed object systems, the objects (i.e., classes) are called on their methods. Both Web services and objects typically expose several operations to clients. In contrast, there is only one REST interface for all services that follow this architectural style. It consists of the standard HTTP commands: get, put, post, and delete (this can conceptually be mapped to the often-used CRUD operations).

The growing popularity of REST is therefore motivated by the fact that it exploits the proven concepts of the Web; its general simplicity makes it easy to get started with SOA projects without limiting their extensibility (REST-based systems can be migrated to full Web services implementations); and the fact that the complications of having to use SOAP and WSDL are avoided.

Building the SOA for the B2B gateway as a "REST-ful" architecture has been a consideration for the project. However, the company wanted to follow an industry standard, namely the standard promoted by the OpenTravel Alliance (OTA), which can be described as "REST-like," but not strictly REST-ful (more information on the standards promoted by the OTA are presented later).

For example, there are six business functions that are provided by the Reservation Request service. The concrete service interface could have been designed as one service (i.e., one interface) with six operations and the associated input and output data items. It was decided to have six different services (and interfaces), whereby each interface has no explicit operation — invoking the service constitutes an implicit function call. This achieves overall simplicity since the different services can evolve independently.

Another example that illustrates the advantages of this design is the Rate Quote service. It has different security requirements than creating or changing a reservation, and it is not as important and therefore may run on a server with less performance (i.e., separation of SLA requirements).

Many of the services that make up the B2B gateway have been designed in a REST-like fashion, but they are not completely REST-ful since they don't follow the rules for utilizing the HTTP commands. They are all based on the HTTP post command (e.g., the Cancel Reservation service should be based on HTTP delete command, and the Retrieve Reservation service should be based on HTTP get command). Furthermore, they do not follow a strict addressing of resources based on URLs. To be REST-ful, all reservations that the system maintains should be made addressable as individual resources.

THE DATA ARCHITECTURE

The diversity of systems that are connected to the B2B gateway (i.e., partner systems, legacy systems, and newly developed applications) necessitates the definition of a comprehensive data architecture. This data architecture deals with the interfaces of the services and the messages that are being exchanged between service providers and consumers. It should not be confused with the permanent storage of the data entities that are typically maintained in a system of records.

For example, a passenger name data entity is defined in the data architecture of a system of records and it physically exists in a relational database, but the semantically equivalent data entity is also defined in the data architecture that describes service interfaces and it physically exists in messages that a service receives or returns. The latter is the data architecture that is the topic of the following section.

Loose Coupling Through Canonical Data Format

The canonical data format is typically introduced in integration solutions as a message format for the information that flows through the integration infrastructure. It is a standardized superset of information that the integration endpoints (i.e., applications or services) employ. The canonical format is independent of all application-specific data formats, and it reduces the number of point-to-point transformations that need to be built and maintained. Instead of building data transformation logic for each pair of applications/services that communicate, only one transformation is required between the proprietary format of each application/service and the common format. Furthermore, more of the applications/services should adopt the common format over time, such that the need for transformation becomes even less (the CSI and all new business partners who will adhere to the CSI format are an example of this standardization).

In addition, since the canonical format is based on XML, it can be extended without affecting the applications or services that have already been integrated. Changes in the data format of one application will not ripple through to the transformations that other applications use. This concept is referred to as logical decoupling. Figure 9 illustrates three applications, whereby two applications employ a proprietary data format internally and one application has been designed for the canonical format.

Figure 10 shows the realm and the boundaries of standard (i.e., canonical) and proprietary data formats across the company's business partners, the B2B gateway, and the core applications.

Some of the company's "inbound" business partners (GT3 and Vettro, depicted at the top of Figure 10) use proprietary data formats, which are transformed to the common XML format (i.e., the internal company standard) when the data enters the B2B gateway. The business partners that conform to the CSI, however, are adhering to this standard, and the data that is exchanged with them does not have to be transformed. The "outbound" partners (Hudson and Vettro) use proprietary formats that do require transformation.

The orchestration services within the B2B gateway exchange data encoded in the common XML format with the application services. Finally, the core applications that handle billing and dispatching also utilize the common XML format when they communicate with the B2B gateway (and for some of the internal data exchanges as well), so there is no need for transformation. The legacy reservation application uses its own proprietary format, which is transformed into the common format on the boundary of the B2B gateway.

Company Standard Format and OTA Formats

The company's common XML format is XML-based and follows the guidelines provided by the OTA.[2] Founded in May 1999, the OTA is a consortium of suppliers in all sectors of the travel industry, including air, car rental, hotel, travel agencies, and tour operators, as well as related companies that provide distribution and technology support to the industry. The OTA now has more than 125 members representing influential names in all sectors of the travel industry.

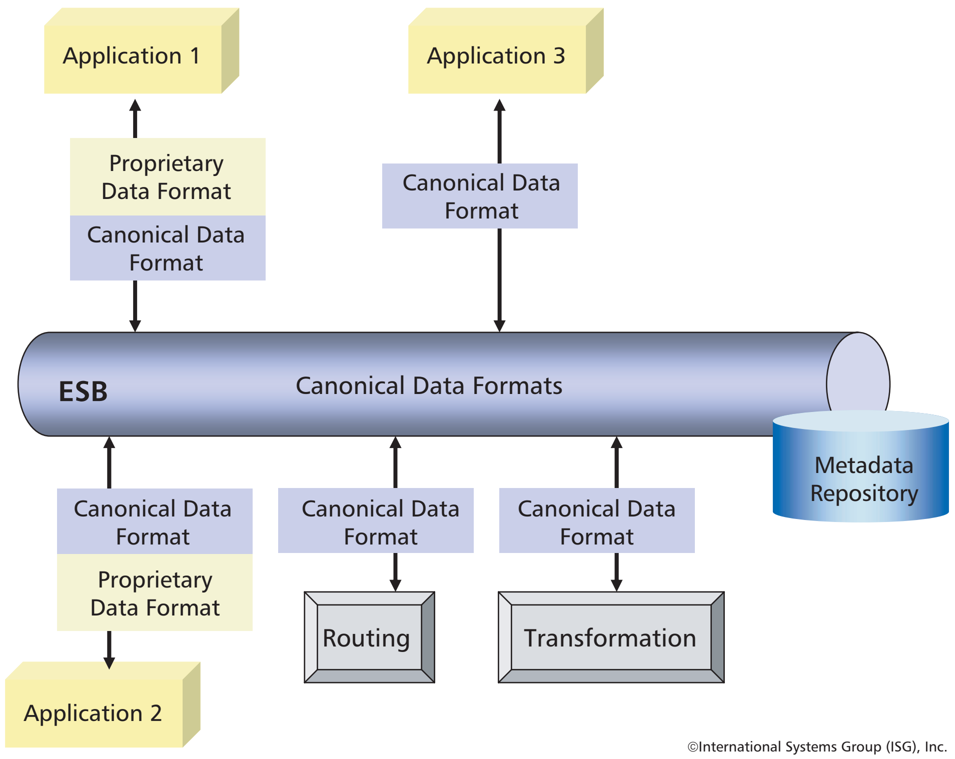


Figure 9 — Canonical data format.

The OTA defines open messages in XML that make it possible to exchange business data seamlessly among different systems, companies, and industries over the Internet. For each of the business transactions that a travel company would typically conduct with a B2B business partner (e.g., making a reservation), one request and one response message is defined through XML schemas. One company would send an XML request message (adhering to the appropriate XML schema), and the partner company would reply with the corresponding response message.

OTA schemas are arranged in several layers, as shown in Figure 11. The OTA has defined a set of common data types that can be (and should be) reused across the different travel sectors. On top of that, each sector typically defines a set of common types that can be reused across the different business transactions within the particular sector. Finally, there are schemas for each business transaction in a particular travel sector.

As illustrated in Figure 11, the company's standard common XML format consists of schema definitions that are part of the OTA common types, the Chauffeured Services common types, and the Chauffeur Transaction schemas.

Reuse of Schema Definitions

Figure 12 provides an abbreviated example how schema components that are defined as common types in the OTA common types and the Chauffeured Services common types are being reused to define the schemas for the Chauffeured Services transactions.

The schema for the response message of the reservation transaction ("create reservation" operation) contains a PassengerOfRecord complex element. This element does not have to be defined from scratch for the response message schema; it can be assembled out of the AddressType element, the CustLoyaltyType element, and the PersonNameType element, all of which are part of the OTA common types.

Similarly, the ReserveConfirmation element in the Reservation Response schema reuses the ReserveConfirmType element from the Chauffeured Services common types, which in turn reuses the RateQualifierCoreType from the OTA common types.

CONCLUSIONS

The B2B gateway provides two key benefits to the company: insulation and automation. It insulates business

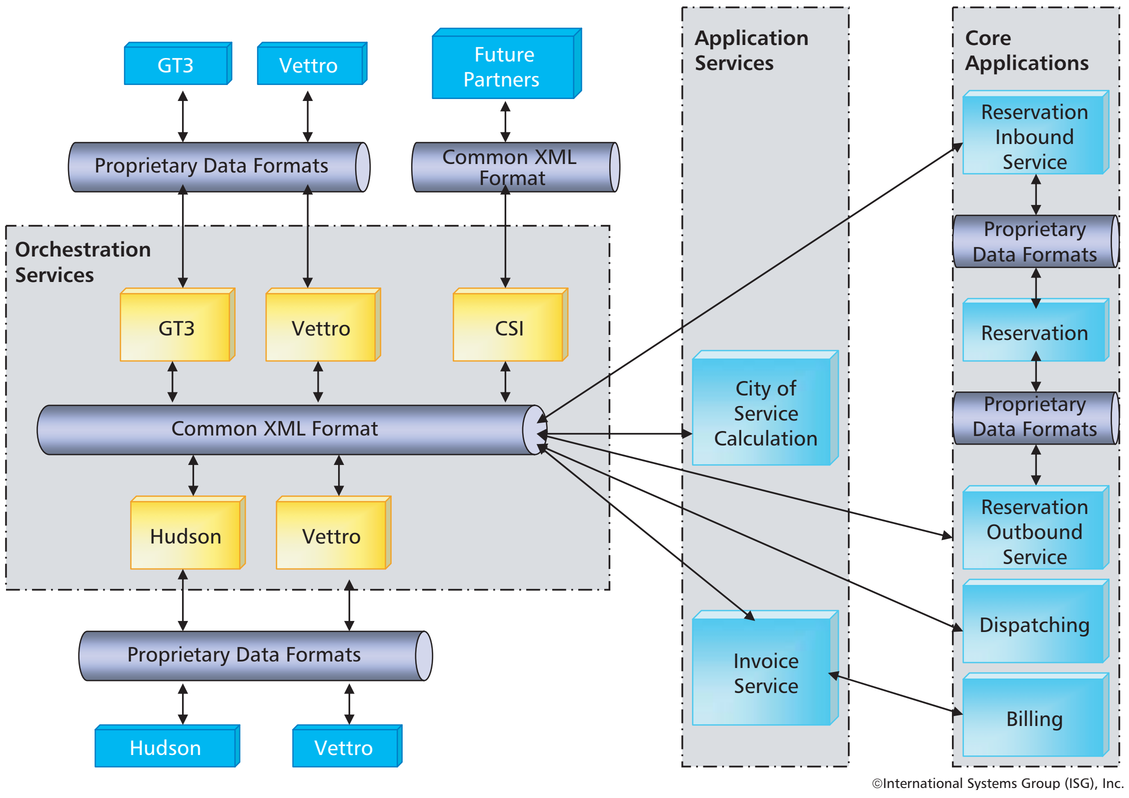


Figure 10 — B2B gateway data format boundaries.

partners from the intricacies of the company's internal applications and business processes. A business partner does not have to deal with proprietary application data formats. A partner can exchange information with the B2B gateway either in its own (potentially proprietary) format or can utilize the standard chauffeured services XML notation that the company has defined.

At the same time, the B2B gateway insulates the company's internal (core) applications from any dependencies on business partners. They can evolve

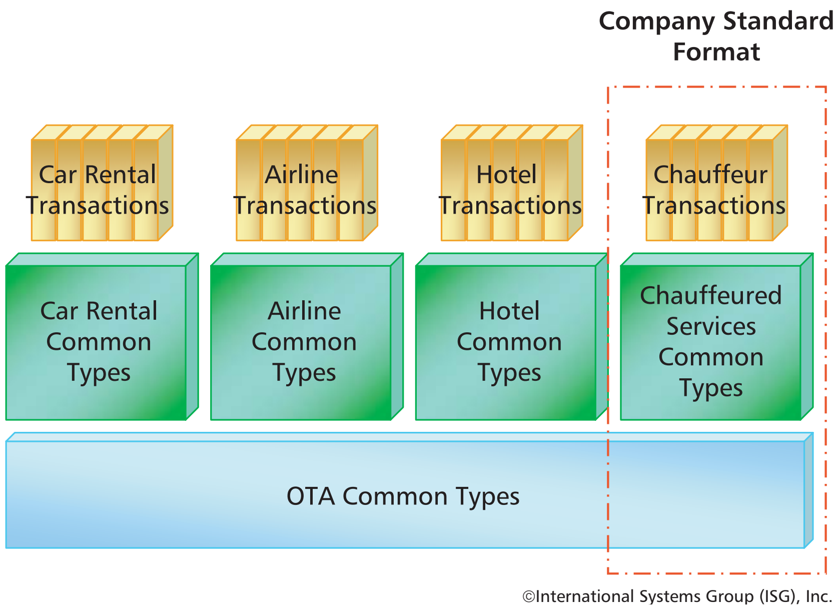


Figure 11 — OTA schema layers.

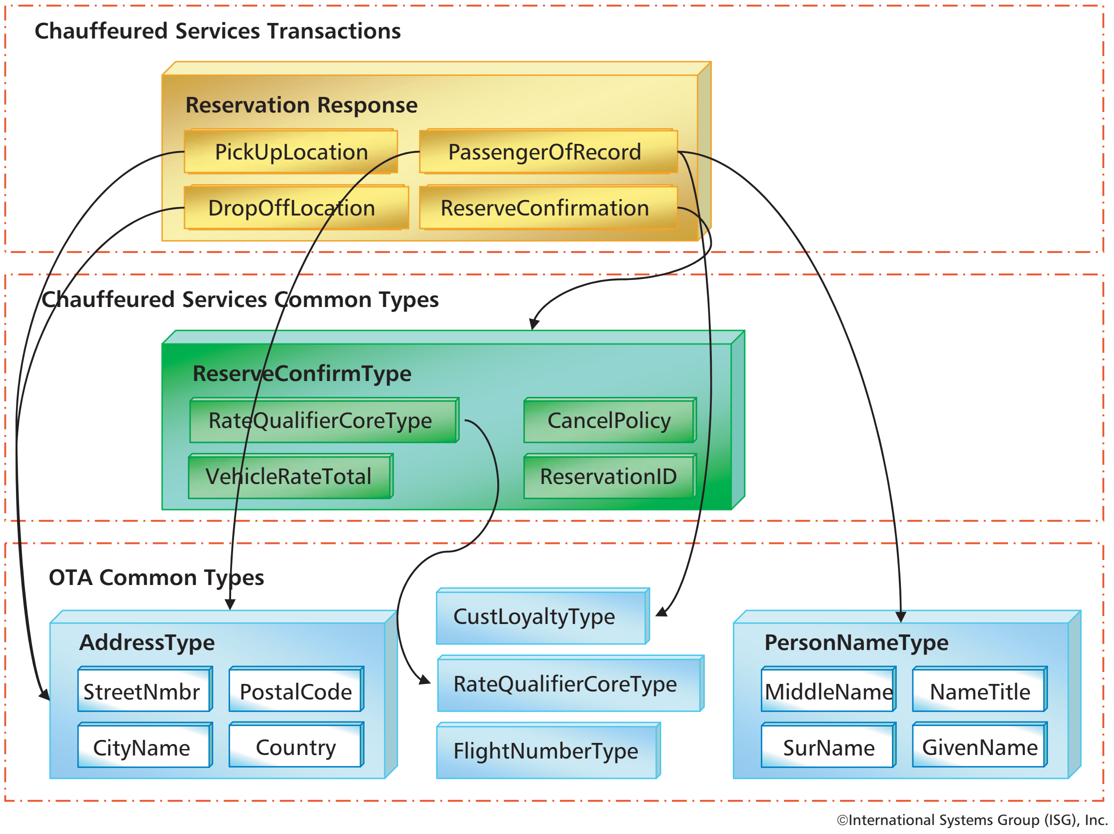


Figure 12 — Schema component reuse example.

independently, relying on the B2B gateway to "make up" for any API changes. They are also protected from business-partner client applications from a security perspective.

The second key benefit that the B2B gateway provides is the automation of a diversity of business processes. They range from dealing with reservation requests over exchange of billing information to wireless communication with drivers. This illustrates how a well-defined SOA can be utilized to support a variety of business requirements with an efficient low-cost approach.

This approach depends to a large degree on service reusability. The potential for reusing a service is significantly improved when the SOA follows a proper layering of the services into orchestration, application, and infrastructure services. Application and infrastructure services provide business process–agnostic functionality that can be leveraged across different business processes.

While SOA is often discussed from a code perspective (i.e., the relevant aspect of a service is its code), the importance of the data architecture can not be over emphasized. (As described earlier, the data architecture defined in this project addresses the schemas for the service interfaces.) The data architecture is a key contributor to reusability in an SOA. Once a hierarchical structure of reusable data entities has been defined, the process of developing schemas for new service interfaces as well as the code that implements a service becomes a predictable task that follows repeatable patterns.

All newly developed services adhere to this XML-based standard for the data entities that are input and output of a service. Legacy systems continue to use proprietary data formats, but they are integrated into the SOA through wrapper services that transform the data to the standard formats. In addition, the mapping to the data formats exchanged with business partners can be achieved with the same efficiency.

Reusability and interoperability have been key to this project, and the company is clearly benefiting from standards like XML — in particular in this B2B gateway project — since it includes a variety of external system. However, too many standards can lead to complexity and inhibit achieving the benefits of SOA. Adding SOAP, WSDL, BPEL, UDDI-based repositories, and some of the extended Web services standards (WS-*) will continue to be carefully considered but only implemented if there is a clear value proposition.

We've witnessed a growing plethora of standards, different versions of standards, and different vendor implementations. It speaks for itself that the "standard of standards" has been created in form of the WS-I, and that OASIS has formed a committee to simplify SOA, while at the same time they complicate matters by adopting the Service Component Architecture (SCA). For now, the B2B gateway project has tried to keep it simple and focus on delivering tangible ROI.

Achieving ROI is one thing, but the success of SOA and the efforts it takes to get there also need to be conveyed to the business sponsors. We have found that a bubble chart, as shown in Figure 13, provides an efficient means to illustrate this.

The vertical axis in the chart indicates to what degree one particular project contributes to the SOA, either

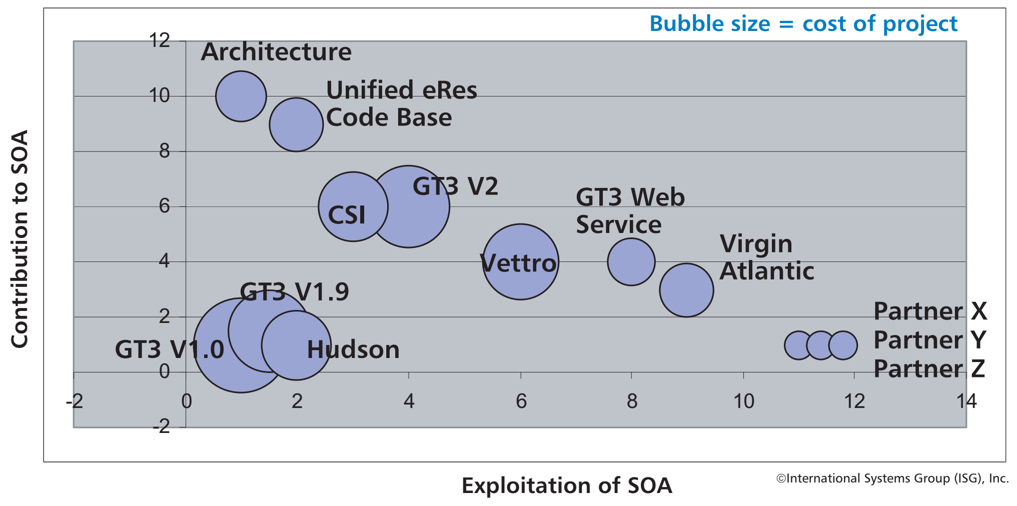


Figure 13 — Bubble chart for B2B gateway project.

by providing reusable assets (e.g., services, schema definitions) or by enhancing the architecture and the development processes toward increased SOA maturity (e.g., policies, best practices). This contribution can not be quantified in a completely objective way; it involves a significant amount of estimation. The horizontal axis shows the level of reuse of existing SOA assets (e.g., services) in a particular project. This exploitation can be measured more objectively than the contribution, based on cost-saving calculations, leaving a relatively small degree of uncertainty. Finally, the size of each bubble is a measure of the cost of that project, which is a well-known number.

The first three projects (bubbles labeled GT3 V1.0, V1.9, and Hudson) did not follow an SOA and consisted of one-off implementations that reused code only occasionally when a developer found an asset that could be shared. These projects also did not create any significant amount of reusable services. This puts the projects into the left bottom corner of the chart (i.e., the "worst spot"). Ideally, initial projects should be on the left top part, contributing heavily to the SOA, followed by projects that are found on the bottom right, exploiting the existing SOA assets and show (relatively) smaller bubbles (i.e., reduced cost). The project called Architecture was a pure architecture definition project. As such, it constitutes an initial investment into the definition of the SOA, which represents a high value in terms of contributing to the SOA.

The next six projects (moving from the top left of the chart down to the right bottom) continued to contribute additional reusable services to the overall architecture; at the same time, it became possible to exploit more of the existing assets for reuse, thus reducing the cost of these projects. Finally, there was a series of planned projects labeled Partner X, Y, and Z, which revolved around integration of new business partners. Since the core requirements for these projects were known, their implementation effort could be estimated, and it could be shown that they could be highly efficient based on a plug-and-play approach.

Using this kind of bubble chart to illustrate the progression of the IT organization along increasing SOA maturity was well received by the business side. It allowed them to visually comprehend how their money had been spent and what benefits had been derived.

ENDNOTES

[1]Fielding, Roy Thomas. "Architectural Styles and the Design of Network-Based Software Architectures." PhD dissertation, University of California, Irvine, 2000 (www.ics.uci.edu/~fielding/pubs/dissertation/top.htm).

[2]The OpenTravel Alliance (www.opentravel.org).

ABOUT THE AUTHORS

Max Dolgicer is a Senior Consultant with Cutter Consortium's Enterprise Architecture practice and a Technical Director with International Systems Group, Inc. (ISG). With more than 24 years' management and technical experience, he is an internationally recognized expert who specializes in IT strategy, business and IT alignment, enterprise architectures, and development/integration of large-scale applications using SOAs.

As a Technical Director with ISG, Mr. Dolgicer has been involved in leading management and technical roles for many of ISG's clients, including Carey International, US Patent Office, New York Stock Exchange, Credit Suisse, Federal Reserve, Allstate, Financial Times Interactive, MetLife, Principal Financial Group, Cigna, CitiGroup, Morgan Stanley, Delta Airlines, Goldman Sachs, and McKenzie Financial Corporation.

Mr. Dolgicer is a recognized speaker, instructor, and lecturer. He presents extensively at most industry conferences. His SOA tutorials, including "SOA — an IT Managers Guide"; "SOA: Organization, Architecture, Technologies, and Design"; "Service-Oriented Analysis, Modeling, and Design"; and "Service-Oriented Integration — Technologies and Best Practices," have educated many senior managers, architects, and developers worldwide. He is also a contributing editor for several major trade publications. Mr. Dolgicer holds a master's degree in computer science from Technion, Israel Institute of Technology. He can be reached at mdolgicer@isg-inc.com or mdolgicer@cutter.com.

Gerhard Bayer is a Senior Consultant and Principal Architect with International Systems Group, Inc. (ISG). His activities include performing leading architectural role on large-scale SOA and enterprise application integration projects for numerous ISG's clients. He is also responsible for the development and delivery of ISG's broad education curriculum as well as product competitive analysis and evaluation to assist clients in the major software products selection process.

Prior to joining ISG, Mr. Bayer was Director of Technology Planning with Software AG Americas, responsible for the evaluation of new technologies and the planning of product directions. In this function, he also coordinated the exchange of technology with business partners. Mr. Bayer holds an MS in physics and a BS in computer science. He can be reached at gbayer@isg-inc.com.

Enterprise Architecture Practice

Today the demands on corporate IT have never been greater. Cutting costs and accelerating time to market for individual line-of-business projects are still priorities, but even that's not nearly enough anymore. Companies are now looking for strategies to better leverage their entire IT infrastructure. They want IT to deliver sophisticated enterprise applications that can provide value across many lines of business and provide marked differentiation from their competitors. The Enterprise Architecture Practice provides the information, analysis, and strategic advice to help organizations commit to and develop an overarching plan that ensures their whole system fits together and performs seamlessly.

The Enterprise Architecture Practice offer continuous research into the latest developments in this area, including Web services, enterprise application integration, XML, security, emerging and established methodologies, Model Driven Architecture, how to build an enterprise architecture, plus unbiased reports on the vendors and products in this market. Consulting and training offerings, which are customized, can range from mapping an infrastructure architecture to transitioning to a distributed computing environment.

Products and Services Available from the Enterprise Architecture Practice

- The Enterprise Architecture Advisory Service
- Consulting
- Inhouse Workshops
- Mentoring
- Research Reports

Other Cutter Consortium Practices
Cutter Consortium aligns its products and services into the nine practice areas below. Each of these practices includes a subscription-based periodical service, plus consulting and training services.

- Agile Product & Project Management
- Business Intelligence
- Business-IT Strategies
- Business Technology Trends & Impacts
- Enterprise Architecture
- Innovation & Enterprise Agility
- IT Management
- Measurement & Benchmarking Strategies
- Enterprise Risk Management & Governance
- Social Networking
- Sourcing & Vendor Relationships

Senior Consultant Team

Our team of internationally recognized specialists offers expertise in security issues, e-business implementation, XML, e-business methodologies, agents, Web services, J2EE, .NET, high-level architecture and systems integration planning, managing distributed systems, performing architecture assessments, providing mentoring and training, overseeing or executing pilot projects, and more. The team includes:

- Michael Rosen, Practice Director
- Paul Allen
- Douglas Barry
- Dan Berglove
- Max Dolgicer
- Don Estes
- Pierfranco Ferronato
- Clive Finkelstein
- Michael Guttman
- David Hay
- Tushar K. Hazra
- J. Bradford Kain
- Bartosz Kiepuszewski
- Sebastian Konkol
- Jean Pierre LeJacq
- Arun K. Majumdar
- Thomas R. Marzolf
- Jason Matthews
- Terry Merriman
- James Odell
- Ken Orr
- Wojciech Ozimek
- Jorge V.A. Ronchese
- Oliver Sims
- Borys Stokalski
- John Tibbetts
- Sandy Tyndale-Bisco
- William Ulrich
- Mitchell Ummel
- Jeroen van Tyn
- Jim Watson
- Tom Welsh
- Bryan Wood