CBDI Report

Service Oriented Architecture and OptimalJ

Web Services has been the subject of much discussion, industry hype and promotion by the software industry and analysts. CBDI has promoted not only Web Services but also Service Oriented Architecture (SOA). On their own, Web Services solve some technical issues to do with platform incompatibility and distributed computing across the Internet, but also build a web of interdependencies and a potential management nightmare. We strongly recommend adoption of SOA as part of the migration to Web Services. The combination of the two provides a platform for business collaboration inside the enterprise and B2B.

SOA is important for enterprise IT because it provides the framework that unites the business model with the applications that provide the functionality required for efficient business.

This report, sponsored by Compuware, looks at how OptimalJ supports SOA and examines how its model-driven approach complements Web Services to provide SOA.

Inside…

Introduction

Web Services has been the subject of much discussion, industry hype and promotion by the software industry and analysts. CBDI has promoted not only Web Services but also Service Oriented Architecture (SOA). On their own, Web Services solve some technical issues to do with platform incompatibility and distributed computing across the Internet, but also build a web of interdependencies and a potential management nightmare. We strongly recommend adoption of SOA as part of the migration to Web Services. The combination of the two provides a platform for business collaboration inside the enterprise and B2B.

SOA is important for enterprise IT because it provides the framework that unites the business model with the applications that provide the functionality required for efficient business. Without SOA IT systems become a disjointed collection of packages, functions and screens that consume ever-increasing resources to maintain and evolve. SOA imposes a direct correlation between business operations and software services, making it a simple task to maintain and re-factor new systems from existing services.

In this report we will also discuss the symbiosis of SOA and J2EE and show how J2EE lacks some important features of SOA unless used in combination with Web Services. J2EE servers are complex platforms for corporate IT but they do make an ideal foundation on which to build SOA provided you have the right tools and techniques. OptimalJ from Compuware is one of the first development environments to fully implement a model-driven approach to application design and development. Model-driven tools start with a logical business model and automatically transform this into an application. The model-driven approach is described in our report entitled ('Governance and MDA'); in this report we examine OptimalJ's ability to generate automatically an application that conforms to SOA and supports Web Services.

Service Oriented Architecture

The SOA concept includes some structural features that go deeper into the corporate IT system than Web Services. Where Web Services address the communications mechanisms to allow functions to be made public, SOA addresses the layering and structure of software. Enlightened organizations have been moving towards SOA by exposing core services in a loosely coupled way that reduces complexity, improves re-use and enables agility.

Some key features of SOA are:

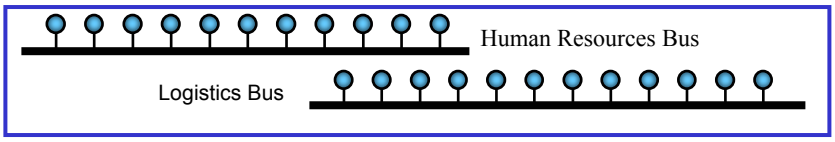
- Platform-independent interfaces
- Loosely coupled
- Business level granularity
- Discoverable

OptimalJ generates a Java application from a high-level business model. In the context of this discussion on SOA, the important point is that OptimalJ creates code that is separated into

logical layers, and builds a set of components that provide the services required by the application. It is the capability to publish very easily these services as Web Services that make them fully platform independent, and the generation of the WSDL (Web Services Description Language) that provides the interface definition required for discoverability.

So to re-cap, OptimalJ generates applications that are logically layered and service oriented. It supports Web Service standards and can selectively publish its services as Web Services. In addition to these essential features, the model-driven approach taken by OptimalJ brings with it some further benefits which differentiate it from a simple framework. To explain these we now look at SOA from a higher level; at the level of the business model, and introduce a concept first coined by CBDI, 'The Business Services Bus'.

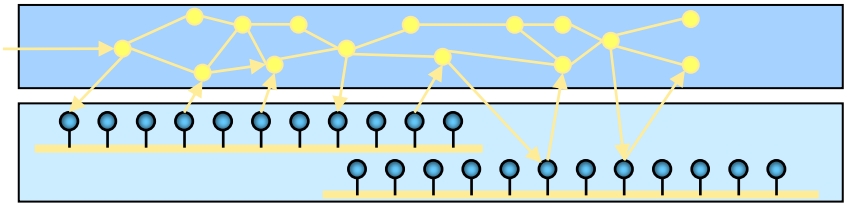
The Business Services Bus



The Business Services Bus concept, first coined by CBDI Forum, is a way to tie together services into logical sets that reflect the structure of the business and are designed for wide-spread use across the enterprise. The Human Resources Bus would for example contain the set of services that underpin HR applications for the enterprise. The logical grouping and design of each bus ensures that there is minimal duplication plus uniformity in naming, ordering, and types of parameters.

OptimalJ provides a domain service modeling capability that positively encourages analysts to architect the service bus in the domain services model. The Business Services Bus concept extends the Domain Model of the organization by identifying the functionality required by the various departments in an organization and expressing that functionality in a collection of IT services. This approach brings the full benefit of SOA. Simply exposing some methods in your code as a Web Service creates only limited benefits to the organization and can create maintenance problems if done badly.

Isolating service interfaces from implementation



The well-designed service bus acts as a façade for the more detailed implementation software that lies behind. Some early attempts at using Web Services simply exposed software functions that already existed in old applications. This often resulted in services that were not consistent nor of the right level of granularity. A service façade can be used to aggregate and normalize existing functionality into an SOA.

Modeling SOA

It is our experience that in order to realize the benefits of SOA, a business model should include a service model which accurately reflects the implementation of services in the code. The documentation and graphical representation of the service domain encourages re-use and ensures a logical consistency between services. OptimalJ's Domain Model includes a service model where your business operations are modeled. OptimalJ transforms these business operations into services. Consistency between the business model and the generated application is ensured and over the life of the applications; the ability to re-use and re-factor new applications from existing services is massively improved through the existence of an accurate and up to date model.

OptimalJ's domain modeling capability ensures that your service bus is designed specifically to your company's business requirements and is *not* just a random collection of session beans that have been given a Web Service interface.

Model Driven Architecture

OptimalJ is a full implementation of MDA [2] as outlined by the OMG. OMG's approach describes separate models; the Platform Independent Model (PIM) and the Platform Specific Model (PSM), which maps to code to create a viable application. OptimalJ implements these two models as the Domain Model and the Application Model, and maps the code to a Code Model. Transformation from one model to the next is carried out by OptimalJ using a set of rules known as transformation patterns.

The **Domain Model** is created in OptimalJ or a separate modeling tool and must be based on UML. Services are defined as a domain service and these typically define a business operation, its name, inputs, outputs and description.
The **Application Model** generated from the Domain Model defines the DBMS model (data entities), the Web model (web components definitions) and the EJB model. Services in the Domain Model are represented as session components.
The **Code Model** generated from the Application Model consists of Java beans (EJB Session and Entities), JSP web pages, deployment descriptors for the selected target platform and WSDL files for Web Services.

Figure 1 shows how a service in the Domain Model is transformed into a session component in the application model.

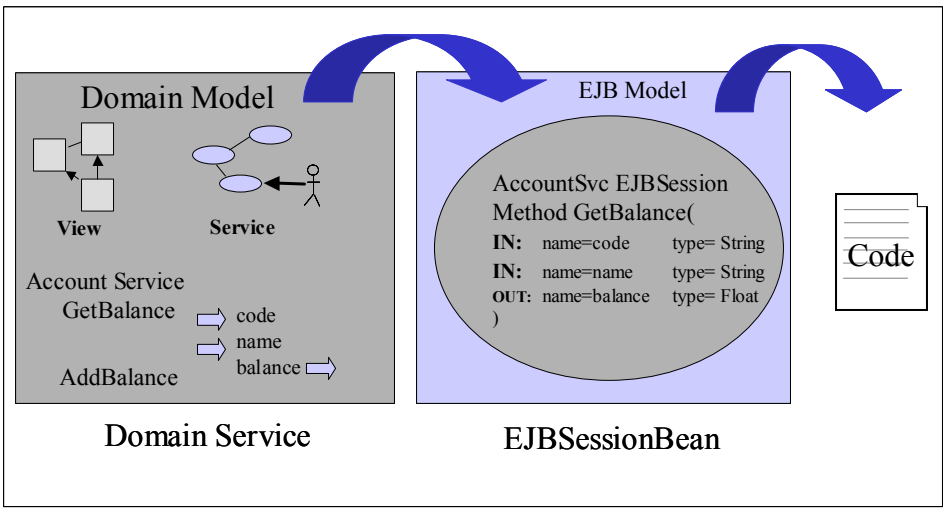


Figure 1: Service Model to EJB Model Transformation

The second transformation step takes the Application Model and generates Java code and deployment descriptors for the application. Code that is fixed by the model is 'guarded' and can't be changed by developers. Services become methods in session beans and developers use the 'free' blocks in the code to customize the business logic for the method. Code added into free blocks is preserved when the application is re-generated from the model.

An important feature in OptimalJ is the facility it gives its users to customize the code generation pattern so that company standards can be accommodated. This also provides hooks to add company policies for the deployment descriptors and other generated XML files.

There are increasingly in the market, proprietary J2EE frameworks that tie companies into run-time components and black box classes. We believe that OptimalJ's open, standards-based approach will be much more acceptable to corporate IT strategists and is in keeping with the 'open platform' paradigm.

MDA ensures that the implemented software is in step with the business model and increases productivity by automatically generating code that conforms to SOA.

Web Services and OptimalJ

OptimalJ supports the provision and consumption of Web Services and generates all the Java code to support them on all the targeted J2EE servers. The two most important standards at the heart of Web Services are:

- WSDL (Web Services Description Language)
- SOAP (Simple Object Access Protocol)

WSDL is the XML standard for describing the message structure and port address of a Web Service. With the WSDL file, a consumer of Web Services can identify exactly the parameters and address of the service. In OptimalJ, the WSDL is imported and used to automatically generate a component that calls the service. When providing a service, the Java bean which implements the interface to the service has to be expressed as a WSDL document. In OptimalJ this is transparent to the developer; when generating the code OptimalJ makes use of the industry standard Apache AXIS implementation to carry out the translation from Java class to WSDL. The AXIS implementation fully complies with WS-I interoperability standards so that your services can be consumed by, for example, .NET clients.

SOAP is the XML protocol, which ensures that the parameters and response messages can be formatted as XML text in order to complete the end-to-end communication. The SOAP protocol can be implemented using different transport mechanisms, but HTTP and SMTP are the only two in use.

At the time of writing, targeted J2EE servers differ in their support for Web Services. OptimalJ's Web Services implementation generates the deployment descriptors required for the J2EE application server selected as the target platform.

Web Service Provision with OptimalJ

A SOA in OptimalJ starts with the Domain Model. The Domain Model includes the Domain Service Model where you model business operations. Domain services may need to make use of business objects to store and retrieve data and these are included in the domain service as domain views. A domain service, once transformed to the Web and EJB model, creates a session bean with methods for the service operations. The session bean is the façade for the entity beans.

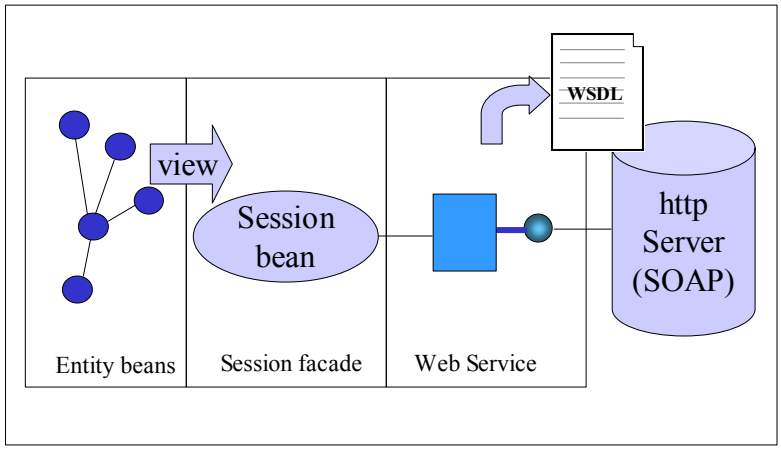


Figure 2: Session bean and service module architecture

Important steps in designing and implementing a Web Service using OptimalJ are as follows:

1. Define the domain service in the business model defining business operations.

2. Generate the EJB model from the domain service; this step creates an EJBSessionComponent in the Application Model.

3. Generate a Web Service from the EJBSessionComponent; this creates a Web Service component and a WSDL file.

4. Generate code from the application models for the EJBSessionComponent and Web Service connector; this builds the session beans to implement the service logic as pre-defined in the domain service, the WSDL, deployment descriptors and interface session bean.

5. Customize the Java code for the session bean to implement the business methods.

6. Copy the WSDL file to a virtual directory on the web server in order to publish the service interface.

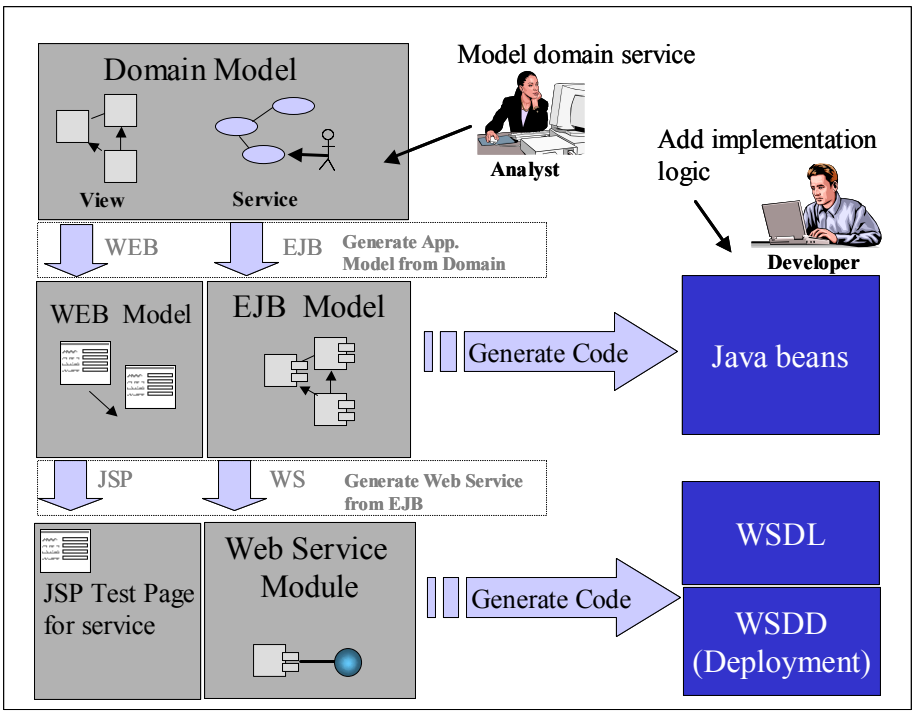


Figure 3: Web Service Implementation In OptimalJ

Consuming Web Services in OptimalJ

A Web Service can be called from your OptimalJ application without writing any code by hand. The process includes importing the WSDL definition of the target service in the connector model and then the generation of a domain service derived from the connector model. From this, a session bean can be generated (and its deployment descriptor) to act as a wrapper for the new capability. Once you have created the EJB model, a web page to test the service can be created, all without having to write a line of Java. The step-by-step process is as follows:

1. Locate the service you require, this may be on your local network or an external web server such as WWW.XMETHODS.COM.

2. Once you have the web address of the WSDL file, use the menu option **Import Web Service WSDL from file…** to create a connector model for the service.

   This creates a connector model from which the Java proxy will be generated using JAX-RPC classes to invoke the remote service. You could stop there if you wanted to write code to make use of this proxy, or continue by building a Domain Model for the service.

3. EITHER: Select the new connector model in the repository and use the **Generate domain from connector** menu option to create a domain service and class model for the service. Custom data structures used in the service will result in new classes in your Domain Model and the process is modeled as a domain service. See Figure 4. Additionally a JSP will be generated for testing purposes.

   OR: Select the new connector model in the repository and use the **Generate EJB from connector** menu option to create an EJB that acts as a client to the Web Service. See Figure 5. In this case no JSP will be generated

4. From the new Domain Model you can generate a session bean to act as a wrapper for the service and a Web model to test the service. The JSP pages generated from the Web model allow you to start testing the service by entering parameters in the web form and checking the results. The JSP page uses the session bean to invoke the Web Service proxy code generated in step 2.

OptimalJ - Extreme Productivity

What stands out from Compuware's approach to Web Services is the complete absence of any need to get out the Java JAX-RPC documentation, or the application server manual. The annoying 'glue' code and XML config. files are built for you using the supplied patterns. The absence of any runtime layer also means that once the code is built and deployed to the target server your application runs within the application server container; there are no black boxes to add uncertainty when tracing or maintaining your code. If you change the application server your Java classes do not need any tweaks; just re-target the new server and generate the required deployment files.

OptimalJ hides the complexity of the deployment platform by generating all the necessary deployment archives and descriptors leaving developers and designers more time to build a coherent architecture that fully supports the business.

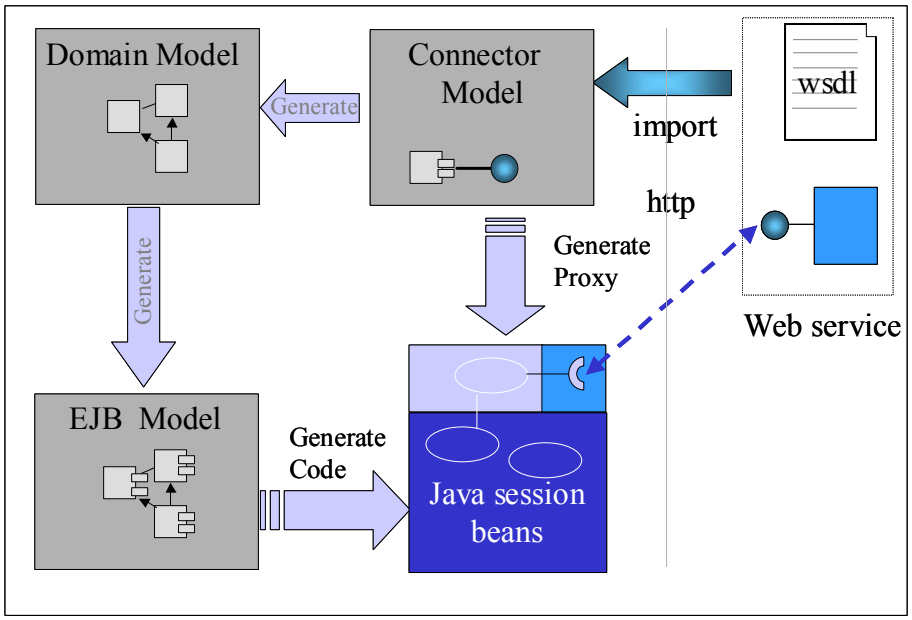


Figure 4: Consuming a Web Service in OptimalJ

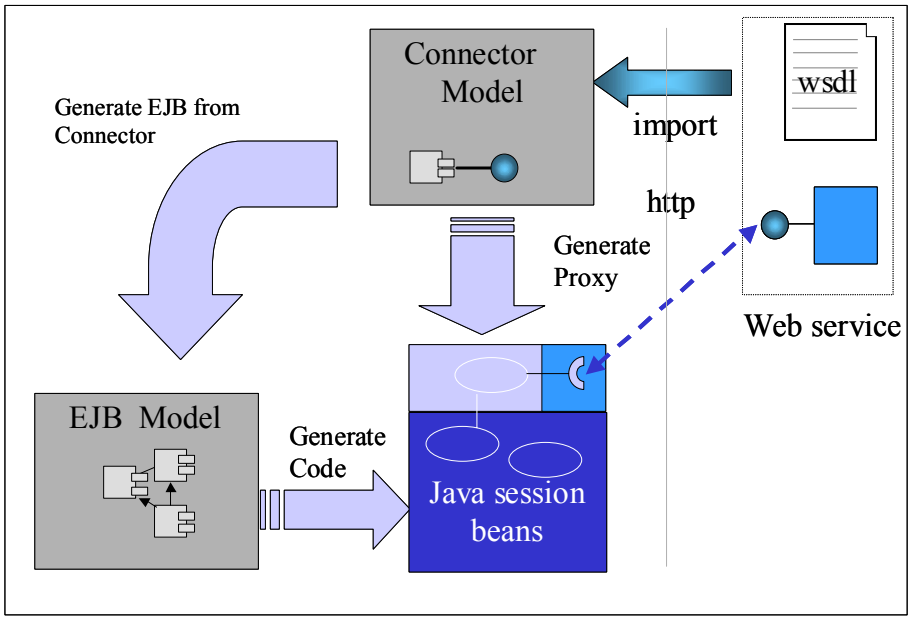


Figure 5: Consuming a Web Service in OptimalJ

OptimalJ Roadmap

We understand future releases of OptimalJ will address four priorities for development:

- Developer and design tool integration
- Support for testing
- J2EE platform support
- Workflow support

The first point is important because OptimalJ's capabilities cross boundaries of analysis, design and code development. It is a Compuware priority to support the leading Java IDEs. Currently OptimalJ 2.2 supports Borland's JBuilder and NetBeans. IBM's WSAD and SunONE are scheduled for a future release. The ability to generate code for testing is another feature that has been identified for the forthcoming releases.

Further integration with the J2EE stacks of the leading J2EE players, such as Sun, IBM, BEA and Oracle will ensure that OptimalJ exploits the capabilities of the deployment platform and seamless integration with developer tools will improve productivity.

Support for evolving standards, not only for MDA but also business process execution (BPEL/WSCI), will continue to be a focus of the OptimalJ development programme. Compuware currently has a fully featured workflow modeling and execution product, OptimalFlow; this capability will be ported to the J2EE environment and integrated with OptimalJ.

Conclusions

There is clearly much more to SOA than the ability to provide and consume web services. Many advantages accrue from the MDA approach and the combination of MDA and Web Services should enable enterprises to move rapidly towards the SOA goal. The following table summarizes the main points raised in this report.

Web Services support	OptimalJ offers full and declarative support for Web Services including deployment descriptors for your target server and WSDL.
Service Modeling	A business-level service model for operations that can be transformed by OptimalJ into services. Web Services consumed by the application are represented in the model.
Model/Code Synchronization	The Domain Service Model and implemented code are always in step, through use of guarded and free blocks
Code Generation for Service Layer	Automatic generation of a EJB session bean to implement each service
Pattern Customization	Code generation is driven by a pattern that can be customized to comply with company standards and styles.
Test Pages	Generation of JSP test pages

Links and Footnotes

Footnote 1	CBDI white paper ‘MDA and Governance’
Footnote 2, MDA	http://cgi.omg.org/docs/ormsc/01-07-01.pdf

Compuware	http://www.compuware.com/
Java Central	http://javacentral.compuware.com/
CBDI	http://www.cbdiforum.com/

The Author

The author welcomes any questions or comments on this report. He can be contacted at:

jonathan.stephenson@cbdiforum.com

Insight for Web Service & Software Component Practice

CBDI Raison d'etre

We aim to provide unique insight on component and web service technologies and processes for the software industry and its customers. To provide high quality analysis and other information resources on best practice in business software creation, reuse and management. To maintain the highest level of independence.

Modus Operandi

CBDI has three channels:

- Subscription services - provision of continuous commentary and information.
- Workshops and seminars - providing in-depth briefing and guidance on advanced architectures, processes and practices.
- Consulting - including related product management and marketing advice, application audit and guidance, technical and business evaluation for investment

How we compare with others

We are not a mass market, media oriented organization. All material published by the forum is unique. The majority of material is published by our own analysts, or commissioned from others, and under strict editorial control to ensure accuracy. We rigorously exclude spurious marketing.

We aim to provide depth insight that covers a narrow topic footprint in a deeper way than the other analysts, and in particular cover not just the technology, but also the architectures, usage, practices and processes.

Also we are unusual as analysts we do not simply echo what the vendors say, we are a think tank, identifying new ideas, opportunities and providing stimulus for thinking.

We try to be thought leaders providing ideas generation and a rich source of conceptual thinking based on practical, real world feedback.

Who reads CBDI Journal?

Technical and Application Architects, Business analysts, Consultants, CTOs,

Designers, Product strategists, Senior Developers, Project Managers etc .

Subscription is split 40% USA, 50% Europe.

Contact us:

For further information on any of our services contact us at: info@cbdiforum.com or +353 2838073 (Ireland)

Subscribe to the

CBDI Journal

The CBDI Journal is published monthly. In addition to the CBDI Journal, subscription includes electronic access to all back numbers that now represent a significant resource library. There are individual and corporate subscriptions schemes.

Corporate subscription includes access to our workshop materials.

For more details and to subscribe see: ***www.cbdiforum.com***