Model Driven Solutions

Where Business Meets Technology

Enterprise Service Oriented Architecture Using the OMG SoaML Standard

A Model Driven Solutions, Inc. White Paper

http://www.modeldriven.com

Cory Casanave

December, 2009

Enterprise SOA

Agility, collaboration and efficiency are the cornerstones of modern business, government and defense. These key goals are motivating enterprise transformations; transformations in the way we architect our enterprise and enterprise information systems. Service Oriented Architecture, or SOA, has emerged as a key enabler to these strategic enterprise goals.

There has been some confusion in the industry as to the positioning of SOA as a technology or business architecture. SOA, as presented here, is *both* a business and a technical approach. The SOA approach to organizing an enterprise focuses on agility by treating customers, suppliers as business entities collaborating in a system of services. The SOA approach to technology helps realize these enterprise services with an agile, interoperable and modular technology base. The *combination* of service oriented business and technical architecture provides a path for transforming the enterprise and enterprise systems.

To achieve pervasive business value, the architecture of services in this system focuses on how business and technical entities, or "participants" work together to achieve business, government or military goals. The focus of architecture is not on "a service" but the system of services that serve a community or organization. SOA with SoaML™ is a path to architecting this system of services, incrementally transforming both the enterprise and enterprise systems. SoaML service architectures are also an important part of governing a substantial service offering.

SOA is, essentially, about how people, organizations and systems work together to achieve some mutual business goal. The SOA abstraction does this by understanding how we can be independent and "loosely coupled" yet work together collaboratively. In the SOA paradigm we work together by providing and using each other's services.

SOA can be applied at a very high level, to understand an enterprise or community or at a very detailed level to understand and specify a particular service operation offered between systems. This wide scope and the ability to "connect the dots" between business and technology concerns is one of the advantages of SOA and what we call ***Enterprise SOA***.

While Enterprise-SOA enhances the business architecture it is also part of the development process. The services defined in the SoaML model are *part of the source code of applications* supporting the enterprise. SoaML services can be used to produce technology artifacts such as XSD, WSDL, Java, Ruby, C#, BPEL, HTML and Deployment Descriptors. These technology artifacts may be sufficiently complete to execute the services directly or may provide a template for developers to add implementation code – perhaps by wrapping existing systems. Using "Model Driven Architecture" to connect the architecture to the systems implementation in this way ensures that implementation and architecture match and also provides a way to support multiple technologies on different systems, or over time as technologies change. Combining the other capabilities of UML® with SoaML provides a full life-cycle capability for architecting and implementing solutions.

This paper provides a business and technical framework for achieving agility, collaboration and efficiency with Service Oriented Architecture using the SoaML standard of the Object Management Group™ (OMG™) to architect Enterprise-SOA solutions. We will introduce the SoaML standard and then show how it provides value to various stakeholders in the enterprise – from a CEO to a developer.

The SoaML Standard

Where the SoaML Standard Comes From

The Object Management Group (http://www.omg.org) is an industry consensus standards organization focused on interoperability through architecture and middleware. The OMG is the home of standards such as the Unified Modeling Language™ (UML®), Business Process Modeling Notation (BPMN™), the Common Object Request Broker Architecture (CORBA®), Records Management Service and many others. The OMG is a vibrant 20-year old organization with several hundred members and a proven process for defining standards.

Adoption of the SoaML Standard

SoaML was adopted in 2009 by the OMG after a three-year process involving multiple participants from large and small companies as well as academia. SoaML is based on the prior experience, methodologies and products of these SOA experts and is designed to support SOA best practices while normalizing divergent terms and notations.

The web page for the SoaML standard can be found on: http://www.SoaML.org. This web page also lists some of the tool vendors that have already released tools or services in support of SoaML. The ”take up” of SoaML by users, tool and services vendors has been quite rapid, which is a good sign for the future of the standard. As of December 2009 SoaML has completed the “Finalization” stage of the OMG process, which means it is implemented and ready for use.

SoaML and UML

The Unified Modeling Language (UML) is the core modeling standard of OMG. UML is widely supported by tools, service providers and education. UML can be used for a wide variety of purposes and methodologies and is designed to be extended for particular purposes using “profiles”. SoaML is such a UML profile: the standard UML profile for modeling services and service oriented architectures. SoaML can “plug into” most UML tools with little or no extra support and can also be used with other features and profiles of UML. SoaML users typically have some background in modeling and UML. Some UML tools have also been extended to provide specific services modeling capabilities based on SoaML. Such tools require less of a UML background.

Scope of SoaML

One of the strengths of SoaML is its scope. There are a lot of technology-specific tools that allow you to create a web service. But, these technology tools do not address high

level concerns: they have trouble showing an entire service oriented architecture – how the services and service participants work together to provide business value. The technology tools are also ill equipped to support SOA for modeling businesses, government or defense and are typically are locked to a particular technology and/or vendor. However, if your only concern is creating a couple of services using a specific technology, these dedicated tools can work well.

As organizations' use and understanding of SOA matures they begin to realize that they can have hundreds or thousands of services and that these services are connected to each other and to the structure and processes of the enterprise. They also begin to realize that there is not one "SOA technology", but a range of technologies that support services.

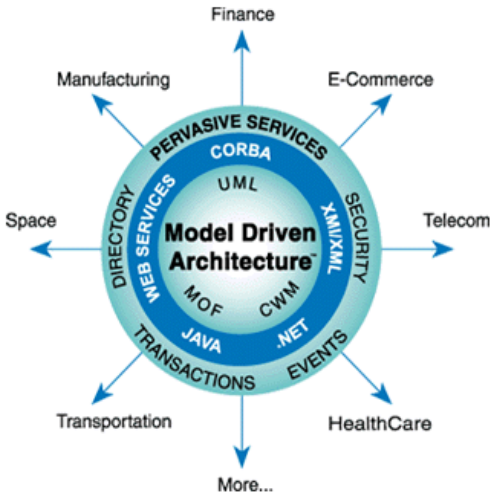
Organizations are most effective when they understand their technology services as they relate to their business services, information and processes. SoaML provides the capability to create and leverage an architecture that helps people, organizations and systems collaborate via services and shows how those services connect to other parts of the architecture, such as processes, information and business rules.

When you "scale up" from "a service" to multiple services that may have interrelated concerns you need an architecture. Many organizations have hundreds of services – defining and governing such an inventory of services without an architecture is chaotic and unlikely to meet business goals.

As an architectural language, SoaML provides a technology independent and standard way to create, communicate and leverage a service oriented architecture.

Implementing SoaML Solutions with MDA®

By design, SoaML is not a complete SOA solution. SoaML provides the capability to model a service oriented architecture at the enterprise, system and systems of systems levels. Most users then want technology services that support and implement their architecture. Using the Model Driven Architecture® (MDA®) techniques of the OMG, SoaML models can be used to implement SOA solutions on top of popular SOA technology standards such as Web Services, Enterprise Service Buses, Application Servers and Business Process Execution suites.



When combined with MDA tools and supporting infrastructures, services and service components defined in SoaML can be part of the "source code" for service implementations – providing a high level development environment where enterprise level solutions can be developed quickly and efficiently from the architecture. By integrating the architecture, design and implementation processes using MDA, solutions can be created more quickly, are easier to maintain and last longer. Because SoaML architectures are technology independent, the "next great technology" can be embraced more quickly, based on the same architecture.

The full toolkit for implementing your SOA solutions will probably contain:

- A UML tool with SoaML support and MDA automation. There are various levels of MDA automation from full execution to generation of only the application or middleware structure.
- Web services or other middleware, including security
- A runtime platform such as an ESB or Application Server with a DBMS
- Optionally, a business process engine and/or legacy system integration
- A User Interface Framework and Tooling

Of course for use only at the design level or when using SoaML without MDA support, only a UML tool is required.

How Enterprise SOA and SoaML provide Business Value

The ability to create and maintain standards-based and technology-independent service oriented architectures and then implement those architectures with automated tools "changes the game" with respect to agility, collaboration and efficiency. Below are a few of the "features and benefits" of using a SoaML based approach:

- Service modeling at the business level – understanding the enterprise as it provides and uses services in the supply chain as well as between departments, units and divisions
- Service modeling at the systems level – understanding how the "system of systems" within the enterprise as well as outside of the firewall work together as a coherent solution
- Loose Coupling – the SOA paradigm encourages and supports loose coupling between systems and business units. This loose coupling supports enterprise integration without entanglement.
- Integration with business process, business information and data models in UML
- Modeling Service Contracts and Service Interfaces to define the roles, interfaces, message data of services
- Modeling service "choreography", the ordering of messages within a service
- Modeling services of various complexity and scope: from a simple service operation to rich bi-directional and asynchronous enterprise scale services
- Modeling participants and components that provide and use services
- Modeling "Services Architectures" which show how systems of participants and components provide and use services to achieve business value
- Modeling capabilities and how they provide and consume services
- Modeling the progress of services with mileposts

- Modeling components and composite components to build service components from other services and service components
- Automation of the development, testing and maintenance processes with Model Driven Architecture to reduce development and maintenance costs while improving agility
- Supporting enterprise transformation, open government, joint operations and supply chains
- Supporting SOA governance

The following sections illustrate scenarios where SoaML can achieve the results summarized above.

SoaML for the Business Architect & Stakeholders

In this scenario "**Acme Manufacturing**" wants to improve service to its dealers and partners as well as organize internally to better support its supply chain. Acme also sells some products manufactured by others and wants orders for these items to be passed to these business partners. Acme has asked Donna, an enterprise business architect, to come up with a blueprint for a solution.

Goals for the Enterprise SOA Architecture

In taking an architectural approach, Donna explains the goals for Acme's enterprise level SOA architecture:

- To understand how Acme would like to evolve to service customers better and at lower cost
- To understand the supply chain in and around Acme. Donna will show who Acme's major business partners are, what services Acme provides customers and what services it requires of providers.
- To understand the organization of major departments within Acme and how they contribute to the supply chain
- To understand in some detail what information is exchanged in the services Acme provides and uses

Donna will validate the this enterprise-level supply chain model with management and subject matter experts and then "drill down" into more detail to make the plan actionable.

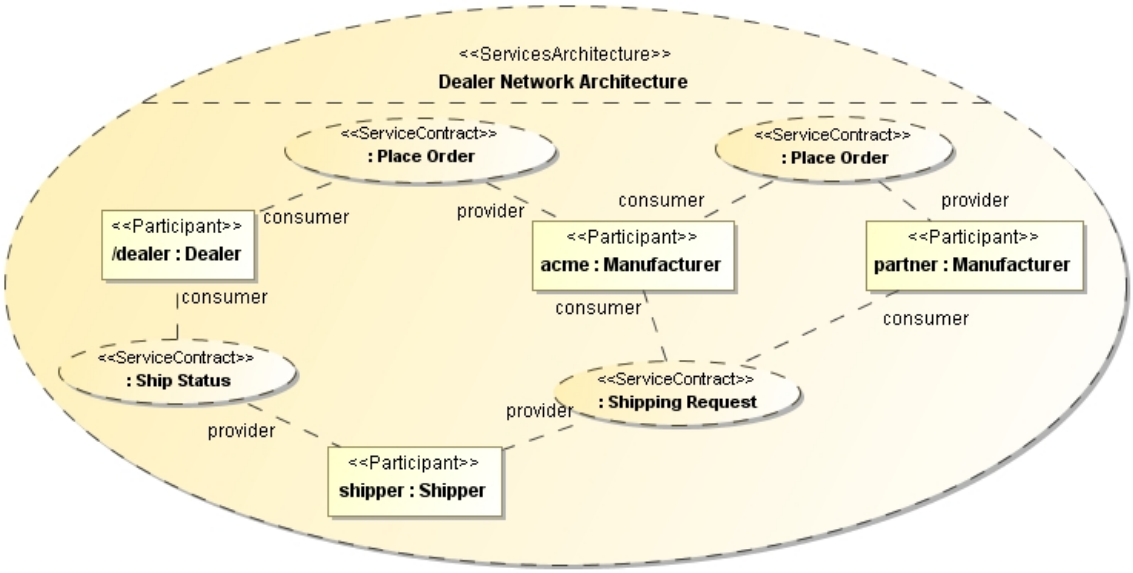
Creating and validating the high-level services architecture

Donna first looks *outside* of Acme to see whom they do business with and how service to customers can be improved. Donna finds that Acme needs to do a better job of fulfilling dealers' orders and quote requests and also needs to be able to delegate orders to other manufacturers, as partners. Acme can also improve its shipping process to speed delivery. Dealers have also noted that through the manual processes of order processing

mistakes are often made that impact their customers. Everyone is on-board to improve business for the entire community.

Donna uses SoaML with her favorite tool, Cameo™ SOA+, to create a high-level architecture that describes the "community" in which Acme does business.

This community uses a SoaML "Services Architecture" she called the "Dealer Network Architecture". The dealer network architecture focuses on how all the participants in the community work together by providing and using each other's services.



It took Donna about 10 minutes to explain to business stakeholders how to understand the services architecture diagram. Acme is represented by the rectangle marked as <<Participant>> acme: Manufacturer. What this means is that in the context of this community Acme is one of the participants. Other participants are the dealers, shippers and partner manufacturers. At the business level, like this, participants are placeholders for organizations, people or kinds of organizations that can work together in this way.

Connecting the participants "service contracts" are being used, represented by ovals. The line connecting the service contracts show what role each participant will play in that service. Donna was able to use the same service twice – the "Place Order" service is used when a dealer places an order with Acme and when Acme places an order with one of its partner manufacturers.

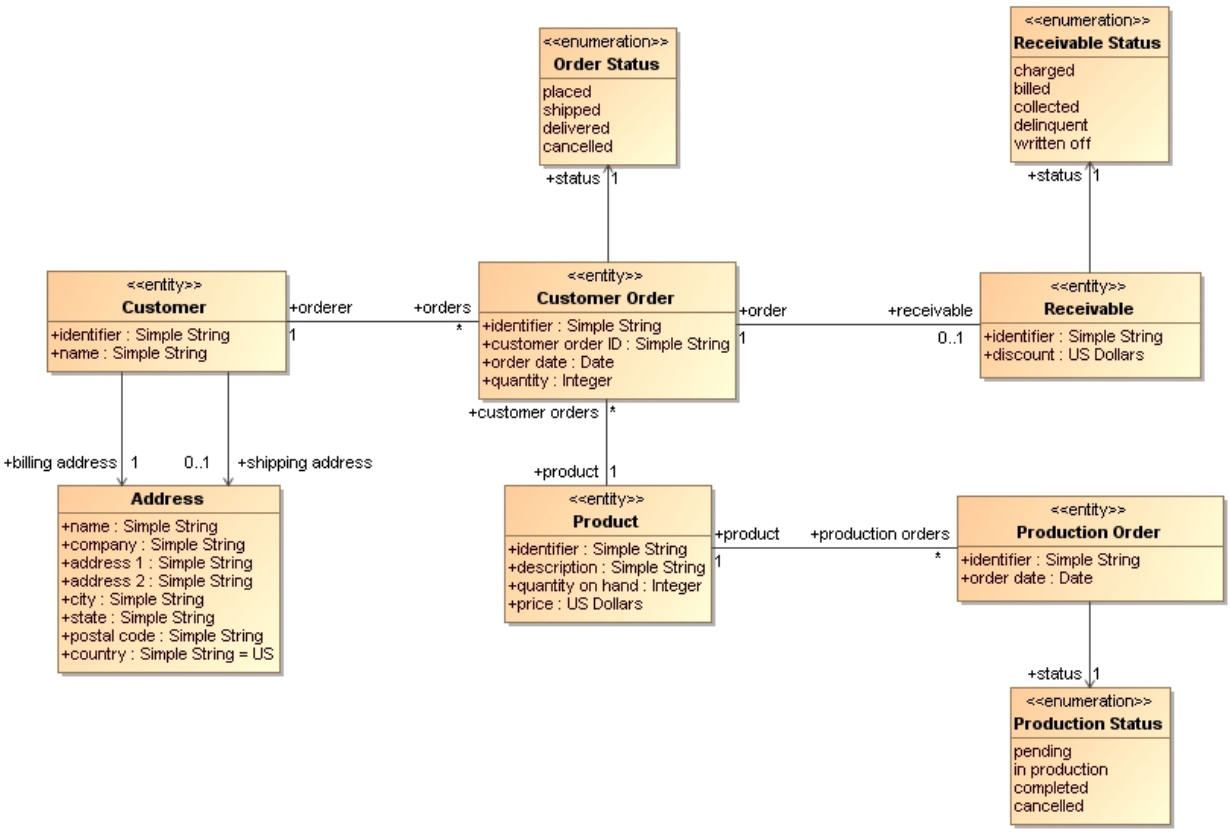
In meetings with management Donna is able to confirm that if Acme could do a better job interacting with the supply chain in this way it would have a direct bottom-line impact. What Acme will do is develop a full service oriented architecture and SOA based web services with web pages that will allow dealers and other business partners to directly interact with Acme. Acme will create some new services as well as wrap capabilities in their existing systems to provide the services. The end result will make it easier for dealers to order from Acme; in doing so, they will be more likely to order more often. The final customer will also get quicker and more accurate service. Since ordering will be more automated with fewer mistakes, Acme will enjoy lower costs.

Completing the business architecture

Based on the positive response from management, Donna adds more detail to the business architecture. This includes meeting with departmental experts, customers and vendors so she understands the business requirements at a more detailed level. Based on this information, Donna defines the business requirements of the services – what information, commitments and products will be exchanged in the services and any business rules about those services.

The Business Information Model

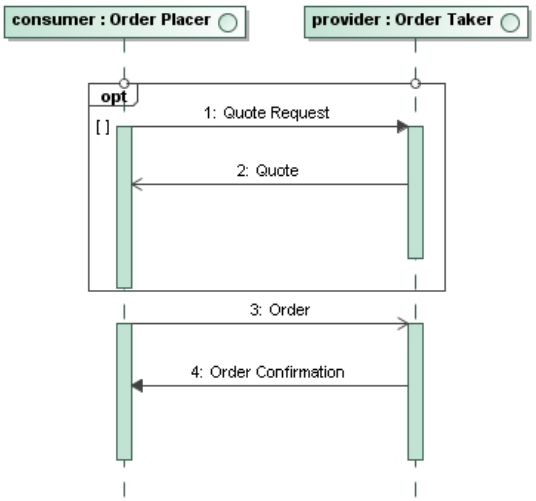
Donna provides a high level UML information model for Acme; one that captures the terms and concepts Acme uses to do business. These same terms and concepts will be used in defining the data that is exchanged as part of the services and will also be used to enhance their internal DBMS.



The above diagram shows some essential elements of the business information model for Acme. The entities (shown as <<Entity>>) are the primary identifiable concepts in the enterprise, which have associations with other data elements, such as "Address". Each class (box) defines a "noun concept" within Acme. Each association (line) shows how those noun concepts are related. The lines within the boxes (such as city: Simple String) are properties of these classes. This class diagram captures some of Acme's terminology in a precise way such that it can later be used to define messages and DBMS tables. As an example of how to read this diagram, starting with "Customer": Each Customer may

have a “Shipping address”. Each shipping address has the following properties: name, company, etc. Associated with the diagram is additional information in the model, such as a description of each concept. Nte that not everything in the model is shown on every diagram.

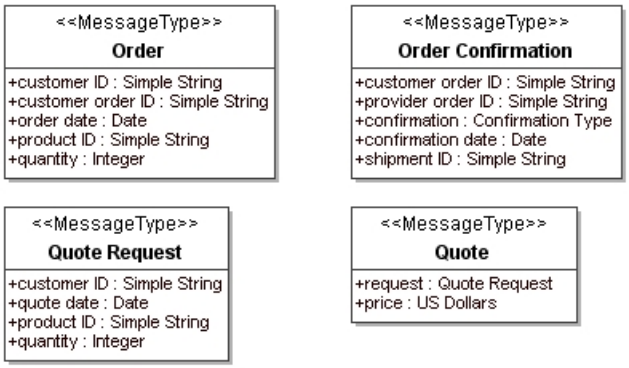
Modeling Services



For each service in the services architecture there will be additional detail that defines the specifics of that service. Sometimes the detail is added in the business model and at other times it is added by the systems architect. The “Place Order” service is crucial for Acme so Donna has specified it in the business architecture. The place order service specification has two primary parts: the “choreography” (diagram to the right) which shows what is exchanged between the service participants, and when. It also specifies the message type – the data that is exchanged. While the SoaML model can express anything exchanged between the parties, Donna is focusing on the information exchanged and what it means.

The “Opt” box in the choreography shows that the quote is optional but the order is not. Note that the exchange of information between the order placer and order taker is “asynchronous”, that is it could happen over an extended period of time. As in business, services in SoaML are frequently long lived and involve information, products and even obligations going in both directions between the parties.

The labels on the arrows in the choreography are the names of what goes between the parties, the “what” is detailed in another diagram – the message diagram (shown to the right). Each message type has a class that models the information to be exchanged. This detail can be simple properties (like those we see here) or more complex message types containing more other class – resulting in a rich data structure.

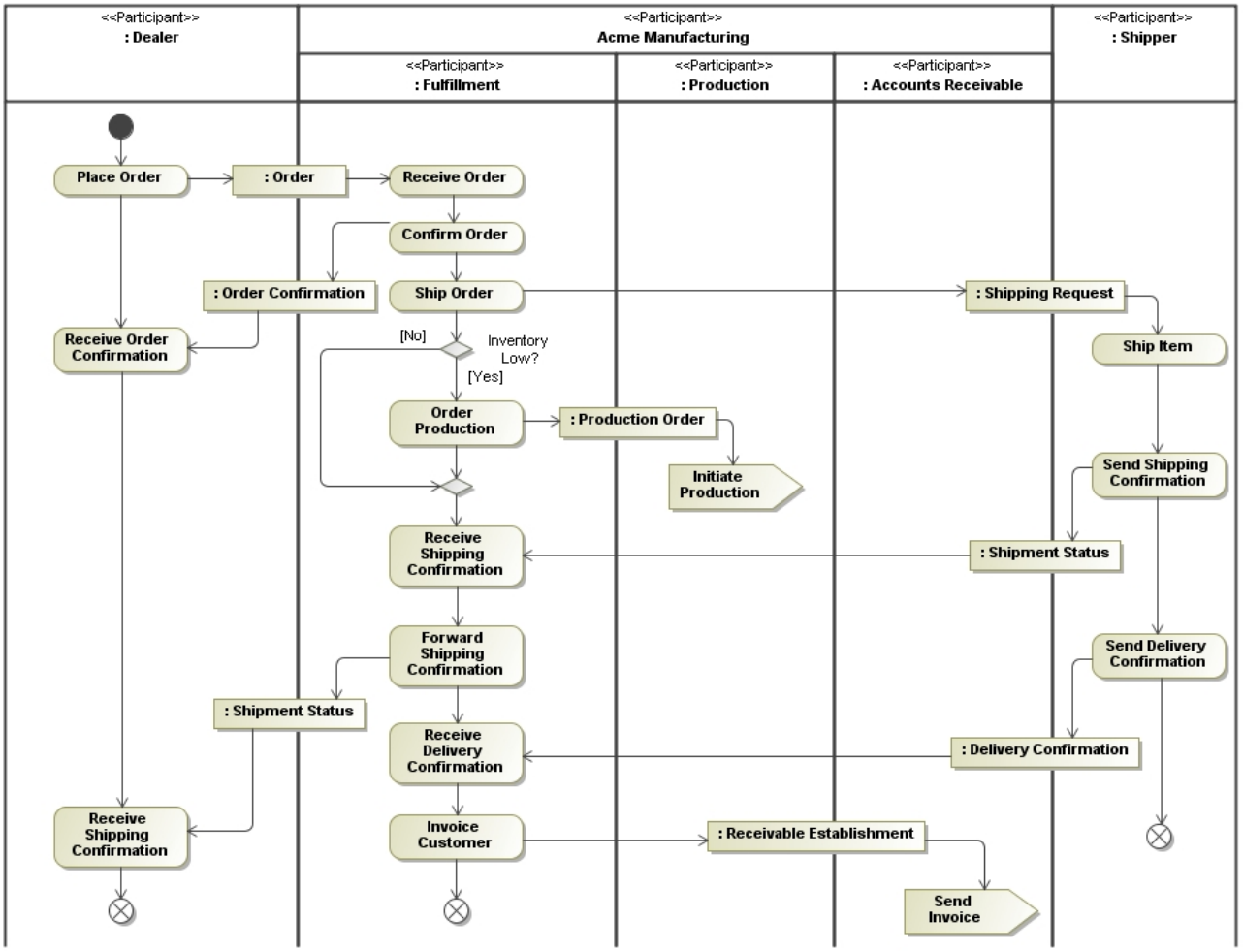


The message types represent the information that is exchanged – the connection to the meaning of that exchange, such as business commitments and change in ownership is possible in UML, but it outside of the scope of this paper. Note that while we are only modeling the information of business interest, these message types will contribute significantly to the technical messaging model that could be implemented in XML and web services.

The Business Process Model

While the services model is important, other aspects of the business are crucial as well.

The order processing and fulfillment process is part of the transformation Acme needs. To make sure it is effective and meeting business needs, Donna creates a target business process as a UML activity diagram for the department – one that matches the external business services she has defined.



The above diagram shows a part of the business process model associated with order processing. The vertical columns are "swim lanes" that represent a participant in the process. The rounded rectangles are the activities that each participant performs. The squire boxes show where SOA messages pass between the swim planes to the activities of each participant. In UML the process, services and information models are related with the same architecture.

Summary of the Business Model

The purpose of Acme's business model is to formalize their plan for transitioning to a more customer centric organization. The business model is about the business, not the technology. But, as we will see later, the business model helps in creating the technology model and systems that make the transition possible.

With a new business model for this crucial aspect of Acme modeled in SoaML and UML, Acme will have a way to understand the advantages of the proposed transformation and a

roadmap for getting there. The information Donna has captured in UML with SoaML is the blueprint for "Acme 2.0".

Governing the transition

With the SoaML based blueprint in place, Acme management has a basis to govern the transition. This involves changes in personnel, facilities and the I.T. systems. The ability to look at Acme in terms of how services connect people, organizations and systems provides the basis to govern the transition. Management also defines a set of metrics that will measure success and makes sure that those metrics will be captured by the I.T. systems as they are transformed – this will be the basis for the business dashboard that will allow more effective governance. Due to the key role of technology, one of the first tasks is to start the process of making the I.T. systems able to support the transition – this means a system specification to support the business.

SoaML for the Systems Architect

Acme has started its enterprise transformation and wants to improve I.T. support as well as provide web services for customers to place and manage their orders. This will allow customer's I.T. systems to directly interact with Acme's systems. Acme has a combination of purchased applications, home grown solutions and ad-hoc spread sheets that are helping the enterprise work, but also holding it back. The problem is that with all of the applications, databases and technologies in play there is a lot of redundancy, poor data quality and inefficient processes. No one really knows how the whole thing works, so it a risky proposition to change anything. As a result, I.T. has gotten the reputation of holding back Acme instead of helping it evolve.

The Systems Architect, Greg, has been assigned to come up with a plan to help make the enterprise transition Donna has architected a reality with I.T. support. Acme is highly dependent on its information systems, and "Acme 2.0" without better I.T. support would just not work. I.T. needs to be more agile, more business focused and less costly. While Acme is doing better than some of its competitors, it has to reduce overhead costs so Acme can provide better value to customers – which is, of course, what "Acme 2.0" is all about.

Greg realizes that the business architecture that Donna did has a lot in common with how Acme needs to transform its systems. Greg proposes a "Service Oriented Enterprise" approach at the I.T. level that builds on the business architecture. While there is a lot to work with in the business architecture, Greg has other things to think about as well. There are some technology services needed to support I.T. and while Greg doesn't want to get locked in to a particular technology, he wants a consistent platform that will be the bases for Acme 2.0 systems.

There is also a need to be open and standards based, particularly in the technologies that interface with suppliers and customers. For efficiency, Greg proposes a "factory like" approach to developing and deploying I.T. services where components defined in the architecture are developed using MDA and then plugged into an Enterprise Service Bus

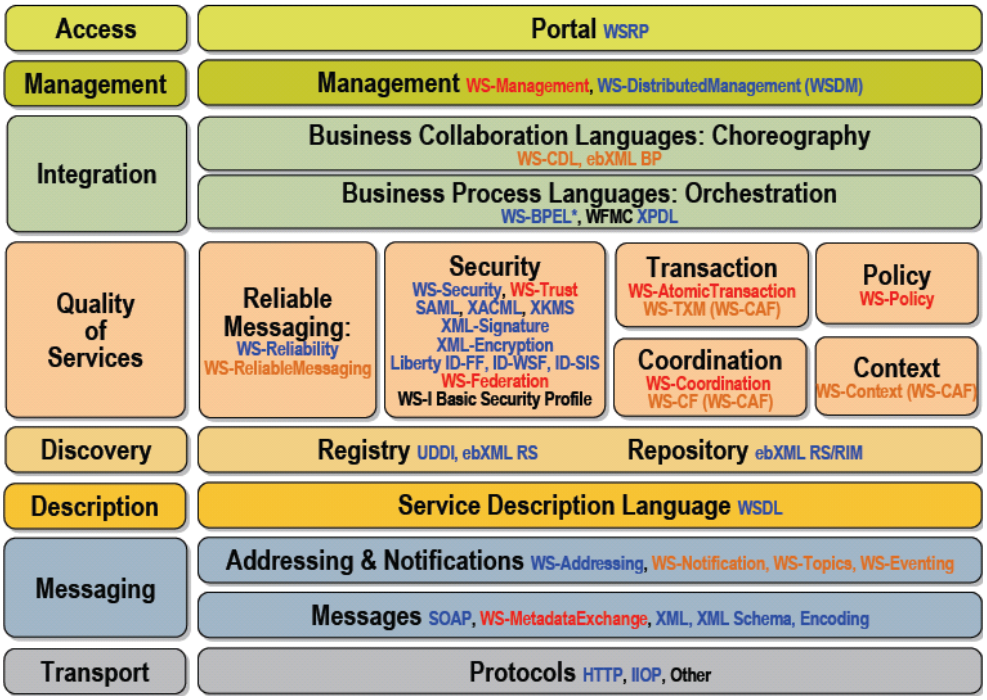
(ESB), providing new application capabilities as well as integrating legacy systems. The core of Greg's plan is:

- Develop a "system of systems" Service Oriented Architecture expressed in SoaML as an extension of the business architecture to make sure all of the systems are loosely coupled but integrated in support of business goals. This will define the services the enterprise needs as well as how they work together to solve business problems.
- Work with technology platform experts to build the ESB platform and technology services based on the business services Donna created. Greg makes sure that the enterprise level system services have all the data the systems needs as well as the "technical bits" that support the ESB Greg will be using.
- Use the order processing function as the first service area to automate – this will help start the process – going through the process will help the whole team learn how to implement architected services effectively. Greg and the governance team chose this function because it is important to customers without being overly complex. Greg will go though the entire lifecycle of the related application components and services so that his team and the technologies they use are proven and up to speed.
- Implement services in this ESB. Greg has narrowed down the choices for a "SOA Platform" built on an "Enterprise Service Bus" as well as a popular DBMS and business process engine. To support this ESB environment Greg defines some new services in SoaML – these are more technology focused than Donna's services and will help keep his "system of systems" working smoothly. Greg decides to use "Web services" for communication with external organizations but to use the JMS messaging capabilities of the ESB for internal systems integration – both the web services specifications and JMS messages will be produced by the SoaML MDA tool based on the architecture.
- Automate as much of the process as possible using Model Driven Architecture (MDA) based tools to keep costs down and quality up. The modeling suite Donna used, SOA+, comes with "ModelPro ™" MDA technology that can generate a lot of the application scaffolding that Acme will need. What it doesn't generate Greg will implement in-house using Java. There is one large piece for integrating a legacy system that Greg will outsource to the application vendor – but Greg will give them the specific WSDL interfaces he wants to use so that the legacy system fits into his architecture.
- Work iteratively and develop the SOA architecture in pieces. Greg is working with the developers so that as the architecture develops they can test it out and refine it as it goes – he sees architecture is a collaborative and iterative process. Initial system testing revels that the existing systems have some inconsistent data that has to be taken care of and that the business rules in these existing systems are not exactly as they thought. Greg makes some changes to the SoaML architecture to account for what he has learned and also has someone start to fix up the bad data.

- Look for existing services. Greg looks around and finds that a couple of the services he needs already exist, one "on the web" for finding service locations near the customer and another for data cleanup that he can purchase. One of the inventory applications they purchased also has a service interface built in. Greg imports the WSDL definitions of these services so that he can call then from the composite applications he is designing in SoaML. Impressed by the capability of calling these existing services, Greg makes sure and look for existing capabilities whenever he is asked to do something new.
- Define enterprise services for systems missing from the business architecture. Greg is also responsible for two systems that are involved in order processing, but didn't show up in Donna's business architecture. Greg includes these systems as part of his systems architecture by defining enterprise services for each of these systems and using these services as part of the order processing application. Greg realizes that just this system integration capability has huge value and that he can start using a SOA approach immediately, even without a business architecture.

Greg's Technology Architecture

The technologies and standards for the ESB based technology architecture is complex.



The above diagram shows the set of standards that will be supported be the ESB Acme will be using. What Greg has decided to do is select a primary ESB that covers most of this landscape and then interface everything else through that ESB. The ESB will, essentially, be the "container" into which business focused services and application components will be deployed. The business service implementation can focus on business needs and the ESB will take care of some of the technical requirements, such as providing the web service interfaces.

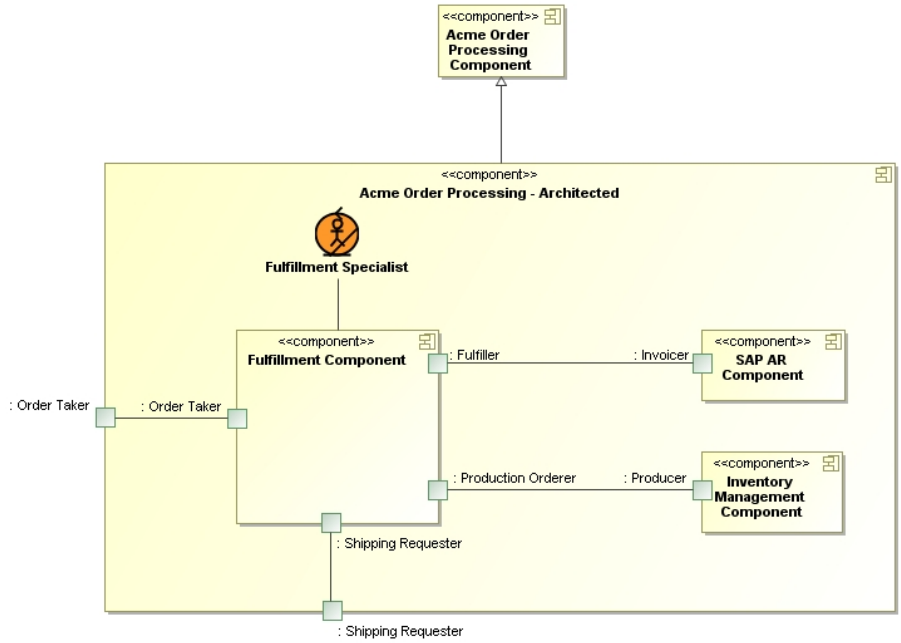
Many of the artifacts required to support the ESB can be produced from the architecture using the ModelPro MDA tool.

The Platform Independent SOA Model

Greg will use Donna's business model as a base and produce a "Platform Independent Model" (PIM) – a model of the I.T. system that will support the business. While the PIM models the I.T. system it does so at a level above the technology details described in Greg's technology architecture. The PIM describes the system components, their service interfaces and the way they are composed to build the system. Additional implementation details may also be included but go beyond the subject of this white example.

Component Specifications

The Acme ESB is component oriented and components can be used to assemble other components. Each component provides and uses services.



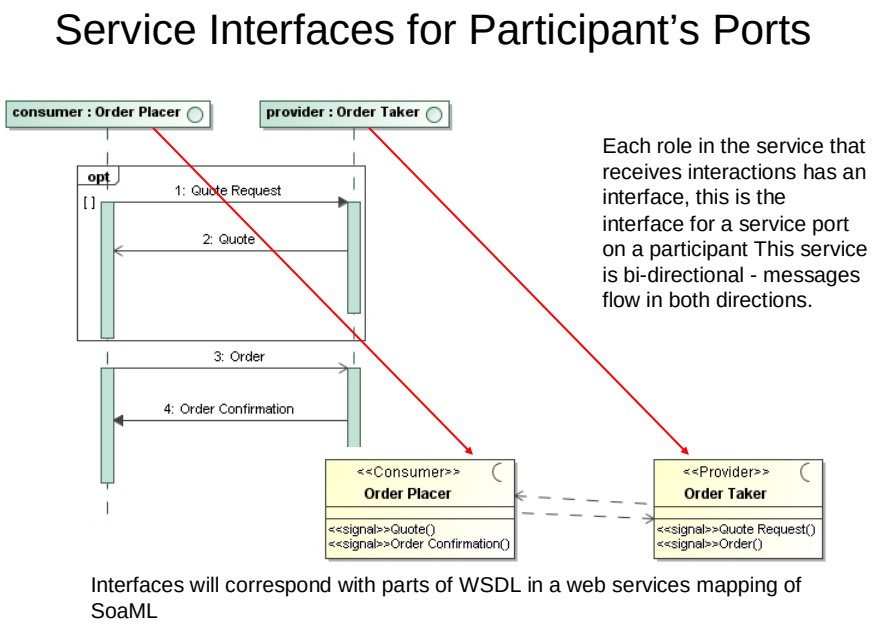
The diagram, above, shows a component specification in SoaML from Greg's PIM. This shows the Acme order-processing component. Notice the small ":Order Taker" box on the left – this is the "port" where the order processing component will provide the place order service. In this case the service is delegated to the "Fulfillment Component" to actually process the order. This pattern is quite common – a component to implement an enterprise service delegates responsibility to one or more internal components.

Notice the "Fulfillment Specialist". The fulfillment specialist represents a person – a user of the system component. By making people and systems a part of the architecture we can understand how the system fulfills the needs of business users and how users play a role in the business process. The fulfillment component will provide a user interface for use by the fulfillment specialist.

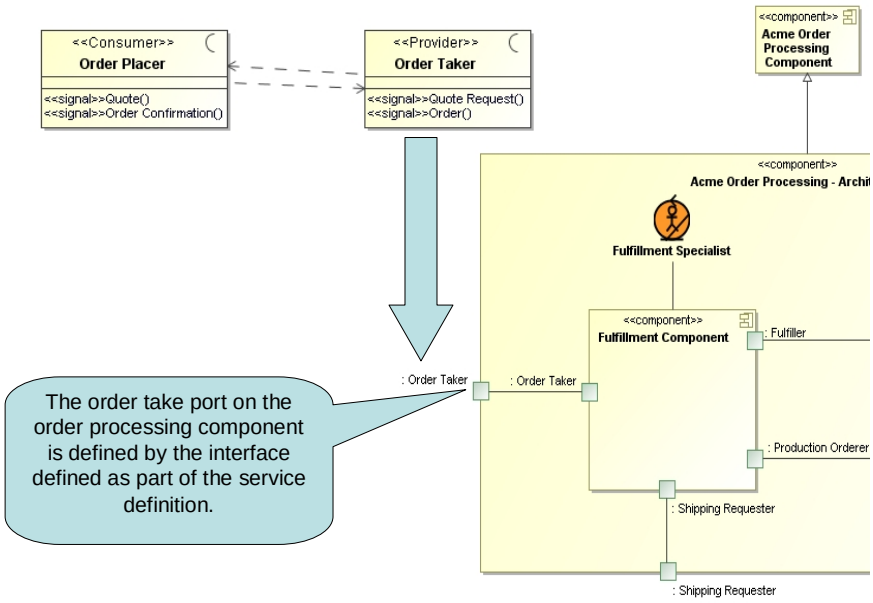
The order-processing component also uses a “SAP AR Component” that wraps an existing SAP system as well as an “Inventory Management Component”. The ports on each of the components used within the order-processing component show the interfaces of the services that each provides and uses. The lines between the ports show how the order-processing component assembles these components into a working application. Of course service oriented components can use other components to any level of granularity required.

PIM Service Specifications

The service ports on the components, above, are defined based on the business services required.



The diagram, above, shows the relationship between the business service and the interfaces used in the component specifications. The interfaces must meet the requirements of the business model and the UML tool fills these in automatically. Each port will have an interface that defines what that port will do – the service it will provide or use.



This diagram shows how the parts fit together – the service interface (a part of the service specification) is the type of the port of the order-processing component. This "connects the dots" all the way from the business model to the PIM, and ultimately to the technology artifacts and source code.

Other aspects of the systems architecture

The following aspects of the systems architecture can, optionally, also be specified in UML, as part of the architecture but are not detailed in this paper because they go beyond SoaML:

- The behavior of methods – the "implementation" of methods can be made part of the model, including sequence and activity diagrams for each as well as source code. This can be used to represent more detail in the application's source code
- The logical data model can be designed, one that could ultimately define the DBMS schema
- The user interface can be modeled and related to the users workflow
- State machines and activity diagrams can show the behavior of a participant as it interacts with services

The SOA Pilot

Greg worked with a developer to implement his PIM and demonstrate a working system that executes a portion of the business architecture. At the end of Greg's pilot implementation of order processing he is able to automatically produce about 60% of the application, his developers do the rest – particularly the web based UI that is so critical to Acme's customer facing systems. Greg was able to integrate two systems, eliminate an

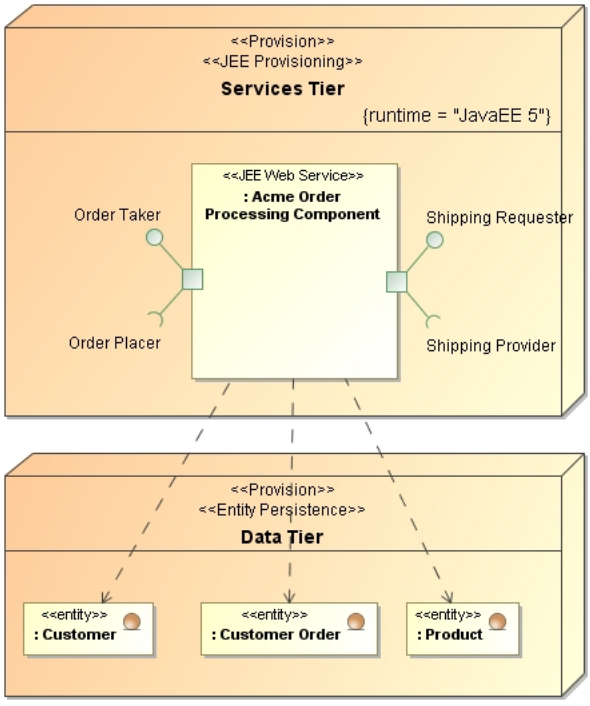
ad-hoc spreadsheet and provide a set of services that are used by some of the larger customers as well as their web based applications. After a successful pilot, Greg is now ready to start repeating the "factory" process to create their architected services supporting Acme 2.0.

The core advantage Greg has demonstrated for Acme is that the I.T. group is now able to take business-focused specifications and quickly create enterprise solutions to automate the business transformation. Because of the SoaML based architecture, the business and I.T. functions are more connected and the systems can be integrated without becoming hopelessly entangled. Since this is all standards-based, Greg is confident they will not get "trapped" into a vendor, as happened in the past. Acme is well on the way to achieving their strategic business transformation.

SoaML for the Services Developer

Jim heads a small team in Acme that will create new services as well as integrate existing systems. Based on the ESB that Greg decided on, Jim decides to use the "Eclipse" Integrated Development Environment (IDE) augmented with the "ModelPro" engine for MDA. Most of Jim's developers already know eclipse and the JEE based application server that is part of the SOA platform. The services Jim will implement are defined by Greg's PIM.

After learning the basics of the MDA environment, Jim decides to give it a try. He takes a part of Greg's architecture and defines a "provisioning specification" (diagram to the right) that defines how the architecture should be transformed into the technologies Jim is using – JEE, an ESB with web services, a DBMS and messaging.



Jim is also able to specify the application tiers as part of the provisioning specification, assigning each component to the appropriate tier with the dependencies between them.

Jim starts the MDA tooling process to see what will happen. Jim is amazed with what comes out – the model has produced all of the XML files – XSD and WSDL, java wrappers, converters, and even components with "put your code here" comments. It would have taken Jim weeks to produce the scaffolding that ModelPro produced in minutes – and if it changes, the artifacts can be reproduced just as fast.

Portions of the Generated WSDL and XSD

This is a part of the "WSDL" for the web service.

```
<portType name="OrderTaker">
    <documentation>UML model element at Dealer Network::Serv
    <operation name="order">
      <documentation>UML model element at Dealer Network::Se
      <input name="OrderSignal"
            wsaws:Action="http://Dealer Network/Services Architect
            message="tns:OrderTaker_OrderSignal"/>
    </operation>
...
```

This is a part of the "XSD" for the web service.

Note that the vocabulary in the WSDL and XSD come directly from the architecture.

```
<xsd:complexType name="Order__Part">
    <xsd:annotation>
      <xsd:appinfo>Dealer Network::Message Types::Order</xsd:
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="customerID" type="xsd:string" minOcc
        <xsd:annotation>
          <xsd:appinfo>Dealer Network::Message Types::Order::c
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="customerOrderID" type="xsd:string" mi
```

The WSDL and XSD fragments, above, are produced directly from the architecture using MDA automation. This becomes the specifications for the components produced by developers and contractors. Additional artifacts, such as Java, Ruby, HTML, etc can be produced as well.

Jim notes that some things don't look right and checks Greg's model –some parts were not complete and there were a couple of errors. Jim works with Greg to make sure the model is correct – after all, this model has become "source code" for the new services and has to be correct. Iterating on the model with Jim, Greg comes out with a result that looks right, and he is able to start connecting it with his existing systems as well as writing some code where it is required.

Because the scaffolding for the application is generated, Jim's programmers are coding into a provided template rather than making up new way of doing things for every module. Jim starts to realize how his game has changed – he is now getting architecture models he can use from Greg and has much better control of the development process, where each developer is assigned to a service component with very well defined interfaces.

Not all of the technologies are supported by ModelPro, but the central ones are. There are a few things that Jim and his JEE team want to change about the way ModelPro is producing the application and there is more that could be automated. Jim wants to extend the MDA automation capabilities for Acme's technical architecture to use JMS messaging. Fortunately, everything that controls the way the application is produced is contained in "Templates" that Jim and his team can modify. Jim assigns one of his developers, Joe, to be in charge of the SOA platform and MDA automation. Joe already knows the ESB technology, but takes a class in extending the MDA tooling and then digs in. Joe also finds the ModelPro forum a great place to find ideas and answers. What Joe

will do is create new options in the MDA tool that the other developers can use, but they will not have to know the details of the MDA tool or even some of the bits and bytes of the ESB. Joe has become the "platform and MDA support" resource for Acme.

After the initial pilot, Jim outlined the development plan for Acme and related it to their business needs:

- The architecture will drive the I.T. solutions – The systems architecture from Greg (which is, in turn, augmenting Donna's business architecture) is to be the cornerstone of the system implementation. It will be used to define middleware interfaces, generate scaffolding, integrate existing applications and produce tests for the services. Some of the business processes will feed directly into a BPM engine.
- The ESB platform will be used for all of the new components, but some components will be built on top of legacy systems. Joe will produce the templates to allow these legacy systems to be able to communicate as part of the architecture. As a next step a business process engine will be used to help automate processes as well as services in the same environment.
- Modeling tools will replace some of the vendors' tools. Jim compared the SoaML approach using Cameo SOA+ to the tools they were previously using, which came from the application server vendor. While the application server tools worked well, they were only good for that particular product and there was no way to connect them to the business services. What Jim realized is that their prior SOA efforts were overly focused on "a WSDL service" and not on how services helped the business and business systems work together.
- The development team will be divided into groups: The business-logic-focused team that will provide new services, the legacy team that will adapt existing systems and data, the Web-UI team that will design cool user interfaces on top of the services and the "Platform" team, headed by Joe, which will support the other teams with the ESB, BPM and MDA technologies. With these teams in place and a flow of architectures from Jim, Joe estimates that they will be producing at a rate that is 2-3 times the old way of doing things, with fewer errors!
- The development team will create solutions that will be built directly from the architecture making management much more confident that what will be coming out of I.T. will meet their business needs. The development "factory" (as implemented by the teams) will be able to respond quickly as new needs are identified.
- SoaML will be used as part of the developer's toolkit. Like Greg, Jim had some services he needed just to help with implementation and systems management. Jim also got into the game and defined these services in SoaML to provide a more unified systems management environment. Using SoaML for services at all levels makes a lot of sense.
- The entire team will design for a changing environment. Since everyone knows technologies change, the model based approach gives Acme confidence that they

will be able to adapt quickly and outrun the competition. The approach to development will be designed to expect and profit from change.

While there was a bit of a learning curve, Jim recognized that his group was now able to provide business-focused solutions faster and cheaper and that maintenance costs would be lower due to both the MDA automation and the consistency of their SOA platform and application components. With the roadblocks out of the way, Jim is far less worried that that "Acme 2.0" would break everything and is excited about the possibilities ahead! The business architecture and service implementations are becoming a competitive advantage for Acme!

Summary

Acme has been able to use SoaML as the basis for an enterprise transformation that spans the full lifecycle of services from business requirements to implemented solutions. SoaML can be used at any level, with top-down or bottom-up approaches to SOA. Please see http://portal.modeldriven.org/content/soaml-presentations-white-papers for follow-on articles and more examples.

References and Notices

Resources

- SoaML Web Site (Supported by OMG)
  - http://www.SoaML.org
- Cameo SOA+ Web Site (SoaML Product from NoMagic)
  - http://soaplus.cameosuite.com
- More in this series of white papers (Including example models from Donna & Greg)
  - http://portal.modeldriven.org/content/soaml-presentations-white-papers
- Model Driven Solutions (Business, SOA, & BPM Architectures and Solutions)
  - http://www.modeldriven.com
- ModelDriven.org (Community for ModelPro everything model driven)
  - http://www.ModelDriven.org
- Object Management Group (SoaML, UML, Middleware and MDA standards)
  - http://www.omg.org
- Cory Casanave (Author of this what paper and contributing author of SoaML)
  - Email: Cory-c (at) modeldriven (dot com)

Notices