Dynamic Modeling: Aligning Business and IT

ARC

1105 Part No. 098-104679

Microsoft

Microsoft

Dynamic Modeling: Aligning Business and IT

January 2006

Microsoft

Architect Resource Library

www.microsoft.com/architecture

Dynamic Modeling:
Aligning Business and IT

January 2006

Visit the Microsoft Architecture Center at
www.microsoft.com/architecture

Dynamic Modeling: Aligning Business and IT Series

Foreword to the Dynamic Modeling: Aligning Business and IT Series

Businesses are always looking for a competitive advantage. In creating a dynamic and effective enterprise, the journey begins with architecture and design. In connecting the architecture of a winning business plan with the architecture of a well-designed IT platform there is an imperative for aligning the enterprise.

This volume of the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* looks at how you can use different architectures, from Open Business Standards to IT industry architectures, in a connected fashion to support a dynamic enterprise. This Series is about aligning business architecture and IT architecture through dynamic modeling.

The way we chose to present this Series was with a number of articles that can be read individually or together for a total view of the enterprise, from business architecture to IT architecture.

We start the Series with the customer and an article called the *Northern Electronics Scenario*, which looks at the real business process of shipping products from a warehouse to a customer. It's a business example you shouldn't need an MBA to understand, and in the article we explore how dynamic business requirements often develop. We go on to show how gathering business requirements can be made easier with Microsoft's InfoPath, taking us closer to an executable digital business plan. We introduce the concept of a Business Operations Health Model, which can be architecturally aligned to a Systems Health Model in order to help business communicate requirements more effectively to the IT Department.

The next stop in this Series is *Communicating Business Operations Requirements to IT*, where we begin with a deeper exploration into an Open Business Architectural Standard to appreciate better that type of an architecture, which we later map to IT architecture. One of the architectural design talks I give these days uses an analogy of "If Business is from Mars, then IT is from Venus"; but whether it's Mars and Venus, Ying and Yang, or Matter and Anti-Matter, the same business principle applies: the enterprise is a total relationship. The better everyone understands each other, the stronger and more agile a relationship they share as they grow and the enterprise grows with them.

The Series then moves on to *Architectural Issues in Managing Web Services in Connected Systems*, where we explore the basic architecture of Connected Systems and new concepts on Model-Based (IT) Management, which Microsoft's Dynamic Systems Initiative (DSI) more richly develops.

Returning to our customer shipping scenario with *Web Service Solution Design*, the next article in the Series explores how to design a shipping business Web service, from a Web service solution design to a Web service Health Model and to an actual solution design for our Northern Electronics scenario.

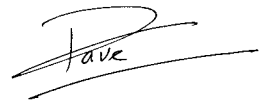
The Series then ventures to *Web Service Health Modeling, Instrumentation and Monitoring*, covering the basic architecture behind a Health Model, the workflow to diagnosing a Health Model, and how to apply these designs to the Northern Electronics Scenario. This article is a must-read for anyone who wants better business management of Web services and Microsoft Management products.

Our journey through *Dynamic Modeling: Aligning Business and IT* ends with a discussion on *Web Service Deployment*, which is also a must-read for anyone using Web services and Microsoft products.

In addition to all the articles, there is also a walkthrough demo that illustrates how to use this architectural thinking with several Microsoft products. The real code behind the demo and the InfoPath business worksheets are available on demand.

I hope you enjoy this Series and find practical use for your architectures. On behalf of the authors, we look forward to hearing from you and sharing your architectural vision.

Sincerely,



Dave Welsh
Architect
Architectural Strategy Team
Developer & Platform Evangelism
Microsoft Corporation

Articles in This Series

- Northern Electronics Scenario: Supporting Business Operations Requirements in the IT Organization
- Communicating Business Operations Requirements to IT: Using Business Operations Modeling in the Northern Electronics Scenario
- Architectural Issues in Managing Web Services in Connected Systems
- Web Service Solution Design: Developing a Solution Design for Web Services in the Northern Electronics Scenario
- Web Service Health Modeling, Instrumentation, and Monitoring: Developing and Using a Web Services Health Model for the Northern Electronics Scenario
- Web Service Deployment: Deploying Web Services in the Northern Electronics Scenario

Go to http://www.microsoft.com/downloads/details.aspx?FamilyID=121755f1-e339-4b83-ba10-143f5671c292&displaylang=en to download the Dynamic Modeling: Aligning Business and IT Demo.

Download the sample:

- InstallerVersion

About the Authors

Dave Welsh
Architect
Dave is with the Architecture Strategy Team in Redmond, involved with helping develop Microsoft's Business Architecture strategies and helping direct Microsoft's industry standards interests worldwide. Dave comes from industry and was also the United Nations Rapporteur for Standards, Chair of the UN's Business Process standards working groups, one of the original ebXML authors, on the International Chamber of Shipping standards team, a representative to the ISO, and member of many European and American industry standards bodies.

- http://blogs.msdn.com/dave%5Fwelsh/
- dmwelsh@microsoft.com

Frederick Chong
Solution Architect
Fred is with the Architecture Strategy Team in Redmond and currently working on architecture best practices that address Web service management challenges. At Microsoft, he has designed and implemented several security systems, including Web single sign-on services for Web sites and Web services, Web service security gateway, SSL VPN solution and secure licensing protocol for Microsoft Terminal Services.

- fredch@microsoft.com

Jim Clark
Technical Evangelist
Jim is with the Architecture Strategy Team in Redmond, having spent more than 30 years in the telecommunications industry. Jim was also one of the principals of the original RosettaNet architecture and a principal contributor to UN/CEFACT and ISO eCommerce standards.

- http://blogs.msdn.com/jimclark/
- jimclark@microsoft.com

Max Morris
Senior Program Manager
Windows Media Platform Group, Microsoft Corporation
Max Morris is a Senior Program Manager in Microsoft's Windows Media Platform Group where he develops architecture strategies and guidance for the Media, Entertainment, and Telecommunications industry sectors. Previously, Max worked in both the Windows and Mobility Divisions on communications technologies, including NetMeeting, TAPI, Mobile Information Server, and Exchange Server.

- maxm@microsoft.com

Northern Electronics Scenario: Supporting Business Operations Requirements in the IT Organization

Architecture Chronicles
Dynamic Modeling: Aligning Business and IT

Max Morris with Frederick Chong, Jim Clark, and Dave Welsh.
September 2005

Applies to:

- Enterprise Architecture
- Solution Architecture
- Service Oriented Architecture (SOA)
- Service Oriented Management (SOM)
- Application Integration
- Business Process
- Business Operations Modeling

Summary

The *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* seek to present to business, solution, and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. Using the story of how Northern Electronics works with its business partners to improve its product shipping process, the authors discuss a systematic approach to formalizing information from many domains into reusable, structured, and connected models, demonstrating business operations and IT operations can increase their communication with each other, resulting in improved business performance and stronger partner relationships.

Contents

This Document

This document is part of the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT*. This volume seeks to present to business, solution, and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. The information map to the series provides an up-to-date description and cross-index of the information available and can be found at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Abstract

Making sure business operations are working well across organizational boundaries is a hard challenge. But it is an increasingly important one to address in today's connected world. A key to success in meeting this challenge is making sure the business and IT operations teams within an organization are well connected. A thorough discussion of a real-world business problem helps highlight the issues involved in making these connections and meeting this challenge. Particularly, the discussion shows how supporting business operations requirements in the IT organization can play an important role in the solution.

Acknowledgments

Many thanks to Nelly Delgado for her help with technical writing, Claudette Siroky for her graphics skills, and to Tina Burden McGrayne for her copy editing.

The authors would also like to thank Gajanan Phadke, Vishal Kapoor, Amol Wankhede, Aziz Matheranwala, Amit Kumar Srivastav, Bhushan Pawade, Bhasha Johari, Avadh Jain, Mughda Gadre, Maneesha Nalawade, Sonika Arora, Anil Sharma, and Anurag Katre (all of Tata Consulting Services) for their contributions to the implementation of the Northern Electronics solution.

The authors would also like to thank Klaus-Dieter Naujok and Allyson Winter (of Global e-Business Advisory Council) for their contributions to the development and documentation of the proof-of-concept business operations modeling worksheet tool used to illustrate this document.

Introduction

This document follows the story of how Northern Electronics works with its business partners to improve its product shipping process. The scenario and discussion are designed and organized to illustrate common characteristics of how businesses deal with each other in a value chain. The scenario is described in a narrative fashion, focusing on a single example throughout to highlight key challenges and possible solutions people and technology encounter when trying to meet business operations requirements with IT solutions. The discussion makes the argument that by following a systematic approach of formalizing information from many domains into reusable, structured, and connected models, business operations and IT operations can increase their communication with each other and thereby achieve better performance internally and with their business partners. An important use of this increased communication is in the effective and timely management of business activities and business exceptions that occur within the IT domain.

More detailed information on achieving operational agility by aligning business and IT can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Also, throughout this document, other documents in the series that rely on the Northern Electronics scenario are noted. These other documents use the scenario to examine in more detail from a variety of perspectives how to align business and IT through dynamic modeling.

Scenario Background

Northern Electronics is an electronics maker based in Everett, Washington, with a partly-owned manufacturing subsidiary based in Nanjing, China.

The company operates in an increasingly difficult market, facing nimble, global competitors and a demand from customers for flexibility to deliver a wider variety of products more efficiently. The executives have concluded that possessing a distinct competitive advantage relative to their competition in how agile they are able to manage their value chain is critical to the company's increased profitability. In practical terms, this translates into a need to identify and capitalize on opportunities to consolidate and automate business activities within their value chain.

In stepping up to this challenge, the executives decided not to go about restructuring their business activities in a top-down fashion. They determined that such an approach would be too disruptive to existing processes and potentially the business as a whole. The approach also seemed too impractical to coordinate. The bottom-up approach was also

rejected, primarily because they determined that setting a goal and hoping it would simply happen was too wishful; they wanted to see results.

Instead, the executives decided to take a middle-out approach in implementing their strategy of change. At the executive level they set clear goals and communicated widely the need, support, and rewards for projects that improve the value chain efficiency. In turn, they became opportunistic about proactively initiating projects that they believed in, and in supporting ones that emerged independently from the various business units.

Product Pickup and Delivery at Northern Electronics

The order fulfillment department at Northern Electronics coordinates the pickup and delivery activities around getting an ordered product to a customer. These activities include verifying and validating purchase orders (POs) with the sales department, verifying credit with the accounting department, coordinating inventory management with the production department across the various warehouses, managing the transport of products to customers, and updating delivery status for the accounts receivable department.

Within the transport function, product shipping is an area that the company has had problems with. Product shipping involves significant coordination, including ordering transport from the right consolidator, moving the right product in the right amount from the right warehouse to the right loading dock, and getting the waiting product onto the right truck. That truck must be driven by the right trucker and arrive at the right time as an agent from the right consolidator.

Product shipping is a core business process for Northern Electronics. Businesses expect problems to arise and have procedures in place to handle them. In some cases, the consequences can amount to just a marginal impact, because the exception can be managed easily or simply accommodated. In other cases, the consequences can be severe and can lead to financial penalties or even loss of business. From the perspective of the warehouse, problems such as the truck not arriving on time, a truck arriving but being the wrong truck, or the truck being too full to accommodate the order can occur. From the perspective of the customer, problems such as non-delivery of product, incorrect delivery of product (the wrong product or the wrong amount of product), or delayed delivery of product can occur.

With good up-front planning and experience to draw on, the difficulty in handling exceptions at the business level isn't in knowing what to do about them. Most issues can be anticipated, and a good plan will include some way of dealing with the completely unexpected. Instead, the difficulty in exception handling is mostly about getting useful information about an exception that has occurred to the right person in time, or even ahead of time, so that he or she can do something about the problem as it happens; or before it happens. When the right information arrives to the right person in time, there's a better chance that whatever intervention the person can implement can have a more significant mitigating effect. When this doesn't happen, the single, unresolved problem can

cause a cascade of other problems, soon bringing the coordination of the activities to a halt. Figuring out what has gone wrong and where, and getting things back on track (if that is even possible), can take a lot of time, energy, and expense.

Making a Change in Product Shipping

The company has had ongoing problems with the product shipping process. Trucks don't always arrive on time. And even when they do, the requirements for the truck aren't always met. Also, the wrong cargo shows up at the loading dock more often than it should. All of these logistical problems result in much higher overhead costs, and especially, in delay for the customers expecting on-time arrival.

Product shipping is an area that Northern Electronics has decided to improve. The company has decided to tackle two aspects of the product shipping process:

- Achieve more efficient coordination in product shipping with business partners.
- Achieve more efficient handling of business exceptions in product shipping when they occur.

Let's take a look at the corporate structure of Northern Electronics and review the individuals who will be involved in improving the product shipping process.

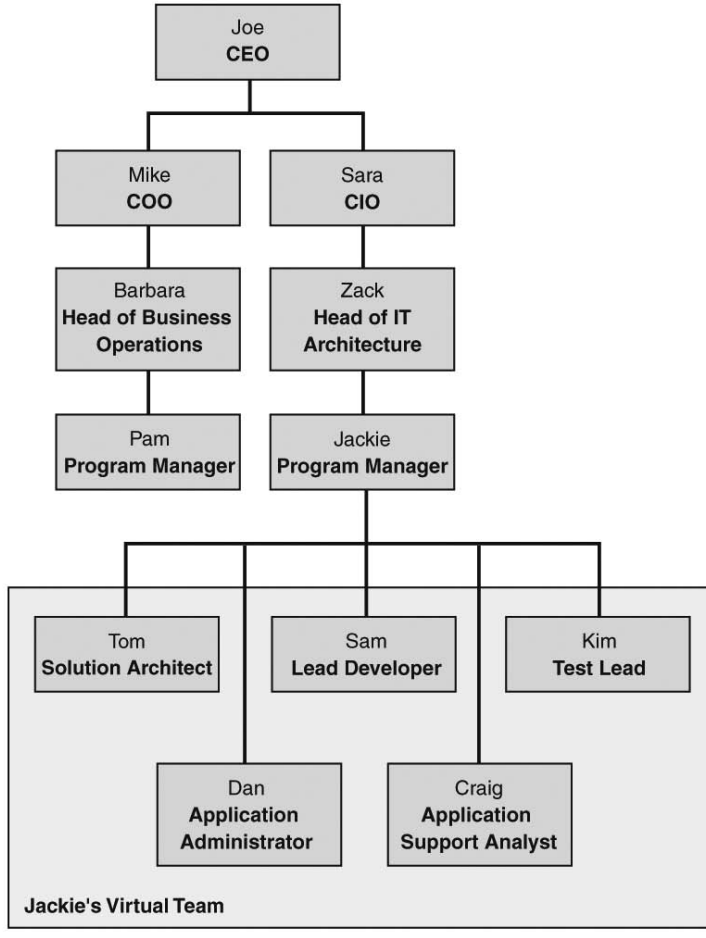


Figure 1. Some of the Northern Electronics employees involved in product shipping

Barbara is the head of Business Operations and reports to the COO. It was her initial investigation of the product shipping process that led to a proposal to improve it. The COO has approved Barbara's proposal, so she is moving forward with her team to carry out the project. Barbara assigns Pam, a program manager on her team, to carry out the day-to-day work. Zack is the head of IT Architecture who reports to Sara, the CIO. Zack is assigned overall responsibility of working with Barbara and Pam to implement the IT components of the improved solution. Zack is responsible for the overall design of the software solution, working with Pam to meet the business operations requirements and Jackie to coordinate implementation. Jackie is a program manager on the IT side who works for Zack. Jackie manages the implementation of the project and works primarily with Tom, the solution architect; Sam, the lead developer; Kim, the test lead; and Dan, the application support analyst.

Modeling the Product Shipping Activities

As Pam gets to work, her first decision is that she must get a handle on understanding the product shipping process. She decides the best way to do this is to pull together a formal description of the existing processes in product pickup and delivery, so she can better determine what the problems with product shipping are and where changes should be made. In a sense, she begins an effort to formally model how product shipping is done.

Understanding Product Shipping

To understand the domain, Pam interviews a number of experts within Northern Electronics and related third parties including the individuals in the following roles:

- The *business analyst* provides business knowledge about product shipping, including what data needs to be collected and how it should be manipulated to reflect the physical reality. This includes requirements for handling and packaging the goods, requirements for third-party transport services, and shipping schedules.
- The *financial analyst* provides the financial plan regarding the shipping budget, customs charges, any other taxes, freight rates (tariffs), and insurance rates.
- The *shipping manager* supervises logistics and enforces any regulatory requirements.
- The *consolidator* is a third party who provides input regarding requirements for packaging, delivery options, scheduling, and relevant forms.
- The *attorney* provides guidance on industry standards and regulations that are best practices and legal requirements for product shipping.

It's especially important to note that not all of these individuals work at Northern Electronics. For example, the consolidator is a third-party company that Northern Electronics does a lot of business with. Getting improvements in the product shipping process is not something Northern Electronics can do alone; it involves the entire value chain, as will be shown more clearly later in this document.

Through Pam's interactions with these individuals and from market and regulatory information, Pam begins to develop a good model for how the product shipping process works.

An example of the broader product pickup and delivery process, namely how a product moves from the manufacturing floor to a customer's facility, that the product shipping process is a part of helps illustrate Pam's findings. Figure 2 (on the following page) describes the high-level business entities involved in the larger product pickup and delivery process. (The entities in this example will be used in the discussion through the rest of this document.)

Some of Northern Electronics's products are manufactured at their plant in China and then shipped to their warehouse in Everett. One such product line is the electronic controller for remote-controlled airplanes. One of Northern Electronics's customers is a wholesaler named Wingtip Toys that assembles remote-controlled airplanes in Dallas, Texas, and then sells the airplanes to retail companies. Northern Electronics hires Acme Consolidation Company, a freight consolidator based in Portland, Oregon, to arrange the transport of the electronic controllers from the warehouse in Everett to the wholesaler's warehouse in Dallas.

Acme Consolidation Company performs many tasks, including:

- Collecting the appropriate paperwork from Northern Electronics.
- Providing the freight cost.
- Reserving space for freight with freight companies and coordinating with the freight companies to deliver the cargo to the destination warehouse.
- Offering Northern Electronics insurance for the cargo while in transit.

In this example, Acme Consolidation Company hires Blue Yonder Truckers, a local trucking company to pick up and transport the goods.

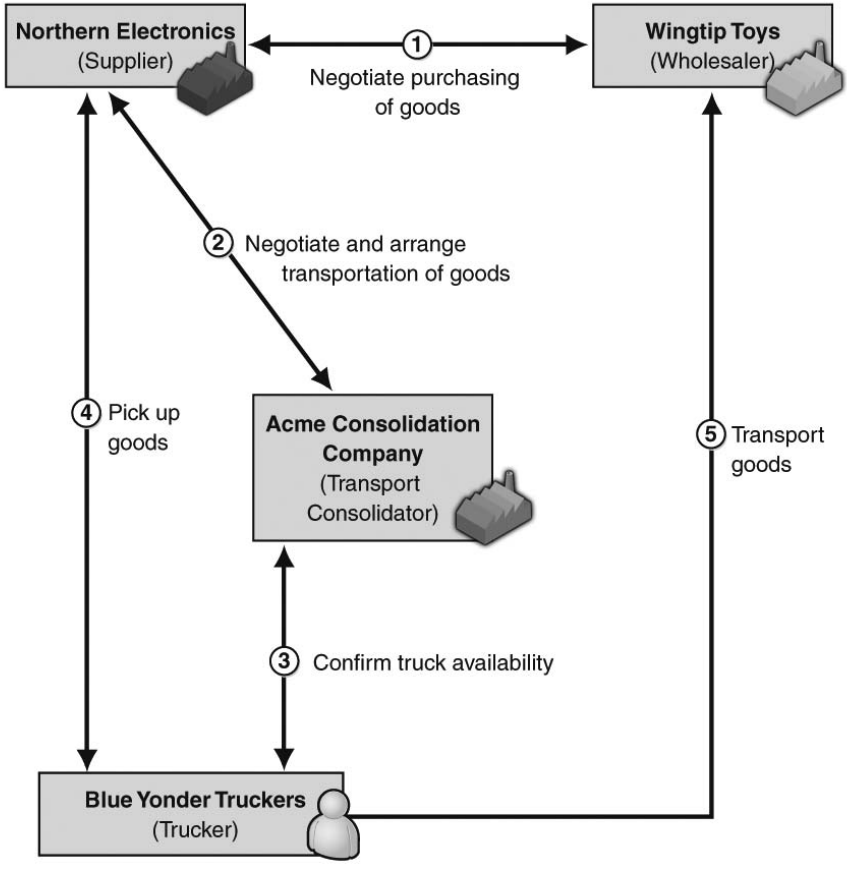


Figure 2. Entities involved in the product pickup and delivery example

Pam learns that whenever Wingtip Toys wants to place an order with Northern Electronics, the purchasing agent at Wingtip Toys calls Northern Electronics's sales agent. Pam listens in on the initial phone call and notes the following:

- The purchasing agent at Wingtip Toys, Frank, tells the sales agent at Northern Electronics, Koji, that he needs 1,800 controllers.
- Koji tells Frank his cost will be $17,316 ($9.62 per piece) plus shipping and handling charges.
- Frank doesn't immediately accept the offer.
- Frank talks to his boss and his boss asks Frank to try to get a lower price.
- Frank calls Koji back and says that the most he can pay is $5.25 per piece. "Perhaps if you increase your order number," Koji replies.
- Frank agrees to increase the order number to 4,000 controllers.
- Koji talks to his boss and they decide they can give Wingtip Toys each piece for $6.74.
- When Koji lets Frank know of the new price, Frank says, "Okay, but only if you give me free shipping and handling."
- Koji and Frank agree on the total price of $29,960 for 4,000 controllers plus free shipping and handling.
- Koji then collects the information and lets Frank know his order will arrive in 6–8 weeks.
- Koji then faxes the order information to Frank.
- Koji writes down the purchase order (PO) details, which include the PO number, wholesaler name, wholesaler address, product code, number of items ordered, expected day (and time not before), expected day (and time not after), and order status.

From this exchange and others like it, Pam is able to determine that the product shipping business process, as a narrower part of the broader product pickup and delivery process, begins when a PO is created. A **business process** is composed of a business activity or set of business activities conducted to achieve a business goal or objective. Here, the objective is to deliver an ordered product to the customer.

In talking to the business analyst, Pam learns that Northern Electronics collaborates with other parties, namely the freight consolidation company and the transport company, to fulfill the activities in the product shipping business process. Therefore, Pam determines that she is really documenting a specific type of business process, namely a business collaboration. A **business collaboration** is a set of activities involving two or more parties to achieve a business goal or objective. In fact, she realizes that the higher-level product pickup and delivery.

In the case of the example, Pam determines that the product shipping business collaboration works like this today:

- Richard, the shipping clerk at Northern Electronics gets new POs on his desk every morning. One of his daily tasks is to verify that the warehouse has the items in stock. If the items are not in stock, Richard notifies the customer that the PO is backordered. The PO is placed in a "Waiting for Goods to Arrive" backorder pile.
- When Richard knows that he has the goods in stock, he performs a stock allocation to cover the order. He calls Julie, the transport consolidator clerk at Acme Consolidation Company, to arrange the delivery of the goods to Wingtip Toys. Julie was on her lunch break when he called, so he left her a voice mail and put the PO into the "Pending" pile. When she gets back from lunch, she calls him back to get the details. Because Richard has a lot of piles on his desk, it takes him a few minutes to locate the PO he needs. He finally finds the PO and lets her know what the requirements are. Julie considers these, and then she agrees to transport the goods. Richard then writes up a new transport order request and puts the PO into the "In Progress" pile. He updates and gives the pickup list to David, the loading dock clerk, to get the required number of items of the product from the warehouse and arrange for the packaging of the goods, including putting the items on pallets, shrink-wrapping them, and labeling the package.
- Julie has a pool of trucking companies that she calls. She goes down the list to order a truck that can drive the shipment from Everett, Washington, to Dallas, Texas. She finally finds one that can do the job. Blue Yonder Truckers can arrive on the following Tuesday at 10:00 A.M. to load the truck. The clerk at Blue Yonder Truckers gives Julie the license number of the truck and she sends the transport order details to the trucker. Julie also faxes Richard a transport notification form with the specific details of the truck.
- Richard keeps track of the scheduling of when packages are supposed to be picked up and makes sure the goods are ready to be loaded onto the dock when the truck arrives. He has a schedule of the pickups on the white board in his office and writes down the new request.
- On the morning of the scheduled pickup, Bob, the trucker, checks in with Richard at 10:05 A.M. Richard pulls up the order and checks Bob's paperwork. He notes the time of arrival and tells Bob which loading dock he should pull his truck to.
- Bob pulls his truck up to the correct loading dock. David asks Bob for the paperwork and loads the packaged and labeled order on the truck. Bob counts the items to verify it's the same number as the number on the paperwork. David enters the final load weight into the transportation documentation and Bob signs the paperwork, assuming liability during transportation of the goods. When Bob drives away, David notes and enters the time and gives a copy of the paperwork to Richard.

- Richard gets the documents and faxes the information to both Wingtip Toys and Acme Consolidation Company to let them know that the trucker has picked up the shipment. He then updates the PO and places both the PO and transport order documents into the "In Progress" pile.
- When the truck arrives at the customer's warehouse, the customer faxes a confirmation of the arrival to Richard. He then places the PO and transport order into the "Completed" pile.
- While all of the shipping is happening, Northern Electronics invoices Wingtips Toys, and Acme Consolidation Company invoices Northern Electronics.

Pam models the information flow around the product shipping business collaboration by using a flow chart, as shown in Figure 3.

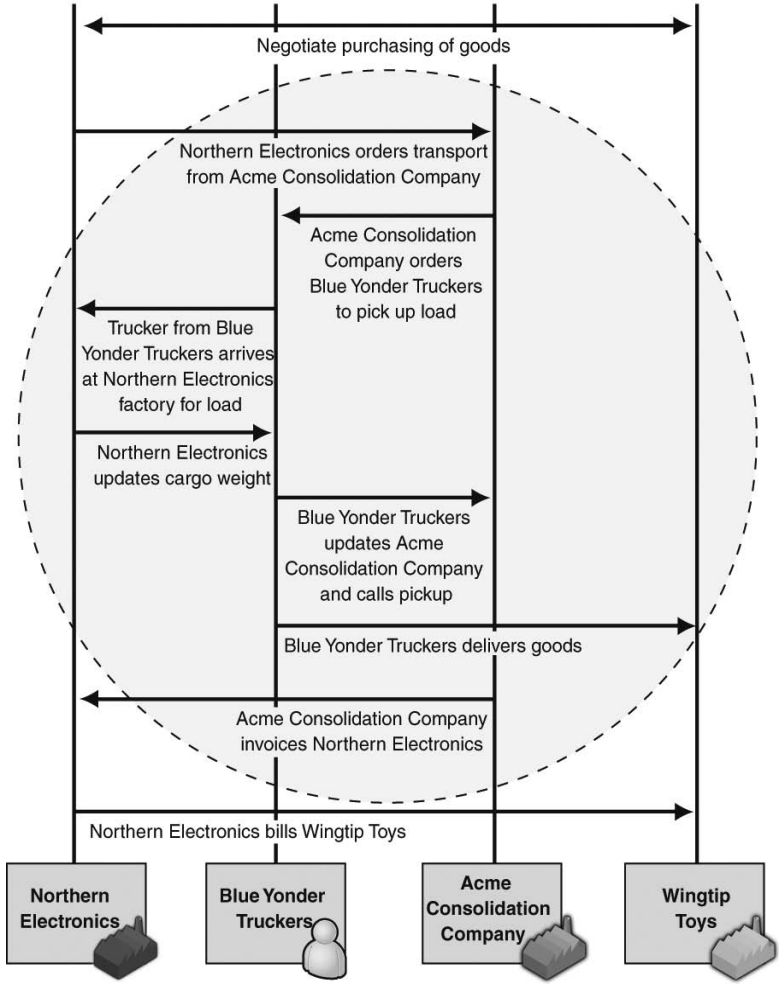


Figure 3. Product shipping business collaboration information flow

As Barbara observed in her original proposal to improve product shipping, Pam can see now more specifically that many of the processes surrounding the product shipping business collaboration are dependent on human interaction and manual processes. There is information that is transmitted over the phone, paperwork that is manually filled out, faxes that are sent and received, and paperwork that is manually handed from one person to another. Pam wonders how often a message is not received or a piece of paperwork gets lost. This must certainly waste the employees' time, waste the company's money, and cause a lot of frustration for everyone involved. Moreover, how well can this process scale when the company starts to grow, or as it tries to be more competitive? How many more employees are going to need to be hired and trained to process paperwork and answer phone calls? Ultimately, the concern is: What is the impact to the customer and customer service?

- Pam recognizes that some aspects of product shipping will still need to be manual but other parts of the process can certainly be automated. Pam works with Zack to identify the areas that can be automated.
- Pam and Zack decide that they need to identify discrete business services that make up the product shipping business collaboration. A **business service** is a set of offered business actions for the purposes of achieving a business objective.

Automating business services in the product shipping business collaboration, where possible, should provide the means for better communication, coordination, and collaboration between Northern Electronics and its business partners. Automating business services should help Northern Electronics to:

- Manage for business performance.
- Stipulate authorized business rules.
- Obtain real-time business intelligence through key performance indicators.
- Ensure the services are designed for reusability and scalability.

One significant source of improvement from automated business services should be the ability of Northern Electronics to be in better synchronization with its business partners involved in the product shipping business collaboration. As noted earlier, one major problem area in product shipping is dealing with exceptions in a timely and effective way. The challenge in exception handling is mostly about getting useful information about an exception that has occurred to the right person in time, or even ahead of time, so that he or she can do something about the problem as it happens; or before it happens. When the right information arrives to the right person in time, there's a better chance that whatever intervention the person can implement can have a more significant mitigating effect.

Pam interviews Richard and asks him, "What are the key things in product shipping that you worry about?" He lets her know there are key events that must happen for the product shipping process to flow smoothly. The most important are:

- The truck must arrive on the correct day within the specified period of time. The pickup time is Monday through Friday from 10:00 A.M. to 10:30 A.M. If the trucker does not arrive within the designated time, the trucker will need to pick up the goods on the next business day.
- The truck must be the *right* truck. This means that the truck must have the correct license plate number as agreed in the paperwork, must be the right size, must be insured, must have the right trucking permits, and must be in good working condition.
- The truck driver must be the *right* trucker. The truck driver must have the correct paperwork containing the shipping reference number allocated by Northern Electronics.
- The loading dock must be ready with the goods when the truck arrives.
- The truck must depart with the shipment loaded within the specified period of time.
- In the event that one (or more) of these conditions is not met, the process will be delayed. In the worst case, one delay can have a snowball effect and delay other shipments as well. This will severely and negatively affect the business.
- Pam talks to Zack to find out whether some of these key events can be managed with IT infrastructure. Pam reasons that if Richard can get an advanced warning before one or more of these conditions are not met, it can help them plan for the situation before it becomes a problem.

Formalizing the Product Shipping Business Collaboration

Pam's investigations into how product shipping works at Northern Electronics have made her the expert. She has put a lot of organization into arranging the information she has collected, much of which came from many different sources. But still, only she really has the full picture of what is happening. The challenge is that many other people within Northern Electronics (and Acme Consolidation Company) need to know some or all of the details about how the process works and where she wants to improve it. It is important for Pam to formally capture the information in such a way that she can put context onto how the information is interrelated. To do this, Pam uses a business process modeling approach.

Pam recalls that a business process is composed of a business activity or set of business activities conducted to achieve a business goal or objective. To model the business process, she must classify the participants and describe what they are doing. She starts with Northern Electronics, which she identifies as a participant in the product shipping business process. The **product shipping business process** begins when a PO is created and it ends when the package is put on the delivery truck. There are many business activities that happen in between. Pam visualizes the model of this information as shown in Figure 4.

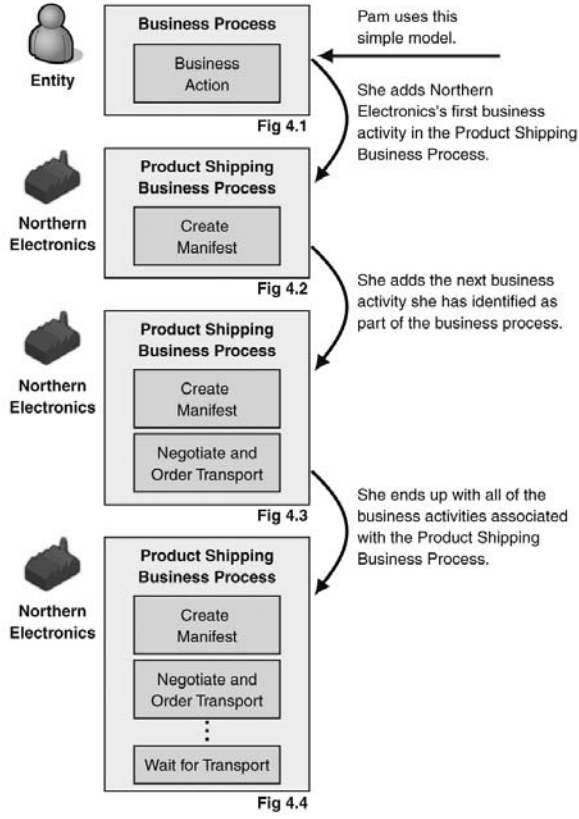


Figure 4. Product shipping business process

Pam realizes that the model visualization in Figure 4 doesn't capture the detail she needs to express. As she determined earlier, what Pam is trying to describe is the activities that Northern Electronics and its business partners jointly care about in making sure the product shipping process happens. It's when these activities don't go as planned that Northern Electronics, or one of its business partners, begins to experience the kind of problems that have led to high costs. Instead of modeling product shipping as a business process that's just about Northern Electronics, Pam needs to model product shipping as a business collaboration that includes the involved business partners.

Pam writes down the business activities conducted within the product shipping business collaboration as shown in Figure 5.

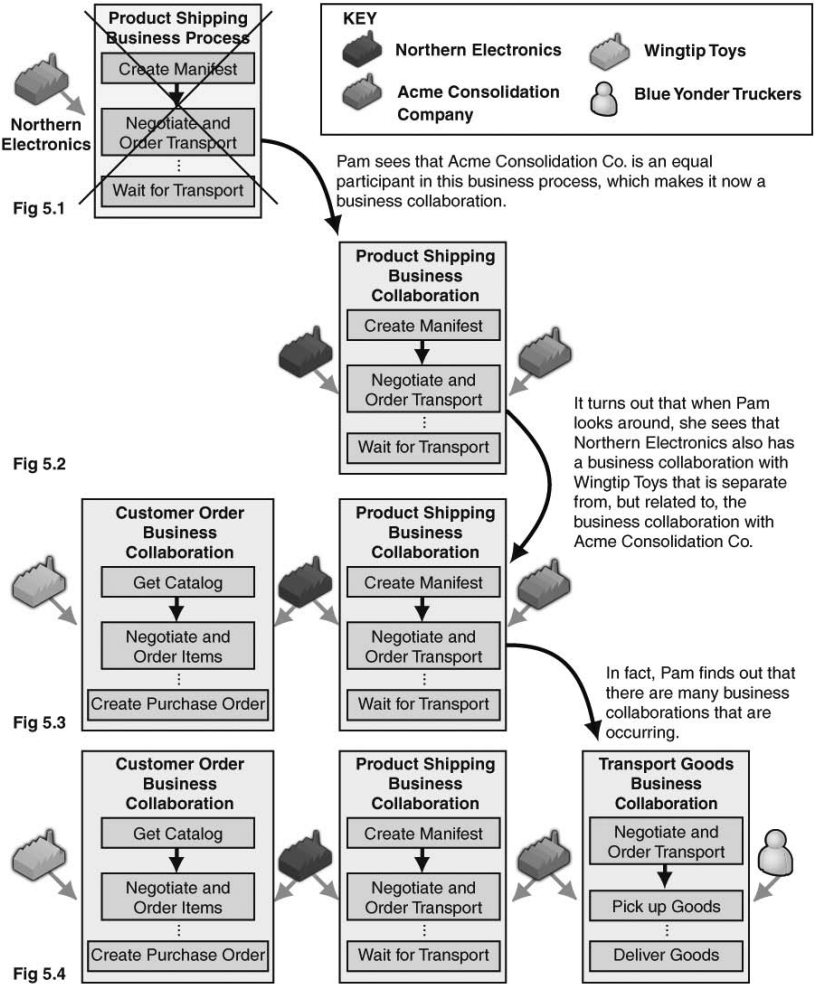


Figure 5. Product shipping business collaboration and related collaborations

The product shipping business collaboration with Acme Consolidation Company is shown in Figure 5.2. Pam knows that as part of the end-to-end example she's been looking at, Northern Electronics also has a business collaboration with Wingtip Toys, namely in ordering the product in the first place, as shown in Figure 5. Pam also knows through her work with her business partners that Acme Consolidation Company has a business collaboration with its freight company, Blue Yonder Truckers, as shown in Figure 5.4. Some of the business activities in that business collaboration operate in parallel with the product shipping business collaboration, but Pam doesn't have to worry about those because she is treating Blue Yonder Truckers as Acme Consolidation Company's agent.

For the purposes of product shipping, Pam is only concerned with the product shipping business collaboration, and in the example, that only involves Northern Electronics and Acme Consolidation Company.

Now that she's looked at the entities as a whole, Pam wants to better understand the business collaboration by identifying the following actors in the scenario and their responsibilities:

- The *supplier shipping clerk* receives new POs and verifies that the warehouse has the items in stock. The clerk is responsible for ordering the necessary transportation for delivering goods to the wholesaler. The clerk is responsible for verifying that the details in the transport notification satisfy the conditions in the requests or take corrective actions if necessary. The clerk arranges for the packaging of the product and is responsible for preparing the loading dock with the goods to be shipped. The clerk uses a pickup scheduling application to determine the PO that has an associated transport notification. The goods are then queued in first-in, first-out order at the loading dock.
- The *transport consolidator* clerk is responsible for processing the transport requests coming from the supplier. The clerk will research with the contracted trucking companies to find available transport for their supplier client. After the transport provider is identified, the clerk fills up a transport notification form and sends it to the supplier.
- The *trucker* is responsible for picking up the goods from the supplier. On the day of pickup, the trucker arrives at the designated time and checks in. When the loading is completed, the trucker signs a pickup completion form to transfer and assume liability of the goods in transit.
- The *supplier loading dock clerk* prepares the final transportation document that contains the pickup details. Upon completion of loading, the clerk sends the document to the transport consolidator to confirm the goods pickup.

Pam uses an activity diagram to document the business activities in the product shipping business collaboration as shown in Figure 6.

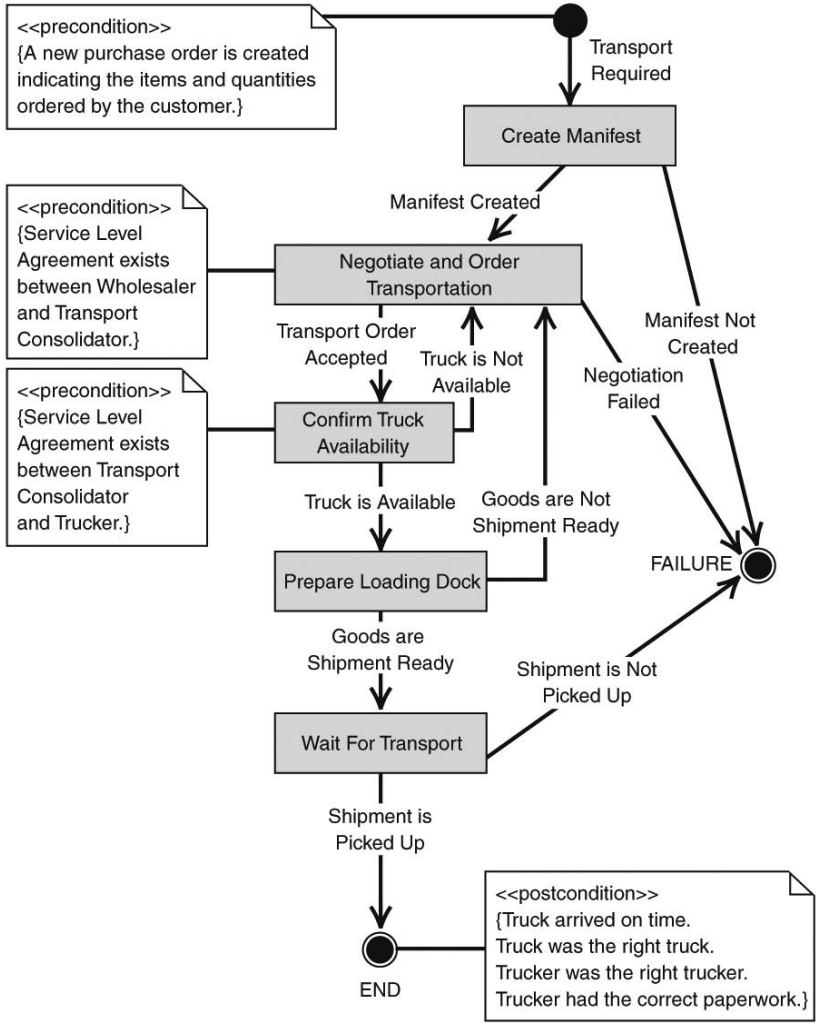


Figure 6. Product shipping business collaboration activity diagram

Some of the business activities are internal activities only relevant to Northern Electronics. However, other business activities rely on a successful collaboration between Northern Electronics and Acme Consolidation Company.

Business Operations Requirements of Product Shipping

Now that Pam has a firm understanding of how product shipping works in the business environment, she needs to translate the information into something that the IT organization can use to design, implement, and operate a solution. Particularly, Pam wants to make sure that the IT solution is set up and operated in such a way that it can help detect and correct any business exceptions that will occur when the product shipping business collaboration operates.

Pam uses the information she has collected and works with Mark, a business analyst, to enter the information into a business operations modeling tool. The specific kind tool they use employs a set of worksheets to capture information about a business collaboration. Similar to how complex tax law lies behind tax forms, behind these worksheets is a pre-defined, interrelated, formal structure of information that corresponds to the things that happen in business collaborations. Because Pam's business collaboration is quite typical, using a worksheets modeling tool is a convenient and sufficient way of ordering the information she has collected.

The value of the business operations modeling tool they use lies in its ability to transform the information after it has been entered into a form useful for IT. Because the tool is designed around the idea of business collaborations, it is able to select and relate connections in the data that aren't obvious to the business analyst. The tool is also able to transform the structure of those relationships into other structures that are much easier for IT architects, programmers, and operators to work with.

Specifically, this kind of tool enables Pam to:

- Capture the formal definition of the business collaboration into an electronic form.
- Derive a business operations specification that describes the overall business collaboration, including the business documents exchanged between the parties in the business collaboration and the semantics of those exchanges. The specification can be used by the IT staff to help guide both design and implementation.
- Derive a business operations specification that describes the critical business exceptions to manage for in the business collaboration, as well as the potential corrections that can be made. The specification can be used by the IT staff to help guide design, implementation, and operations.

An example of a user interface of the kind of tool Pam uses is shown in Figure 7.

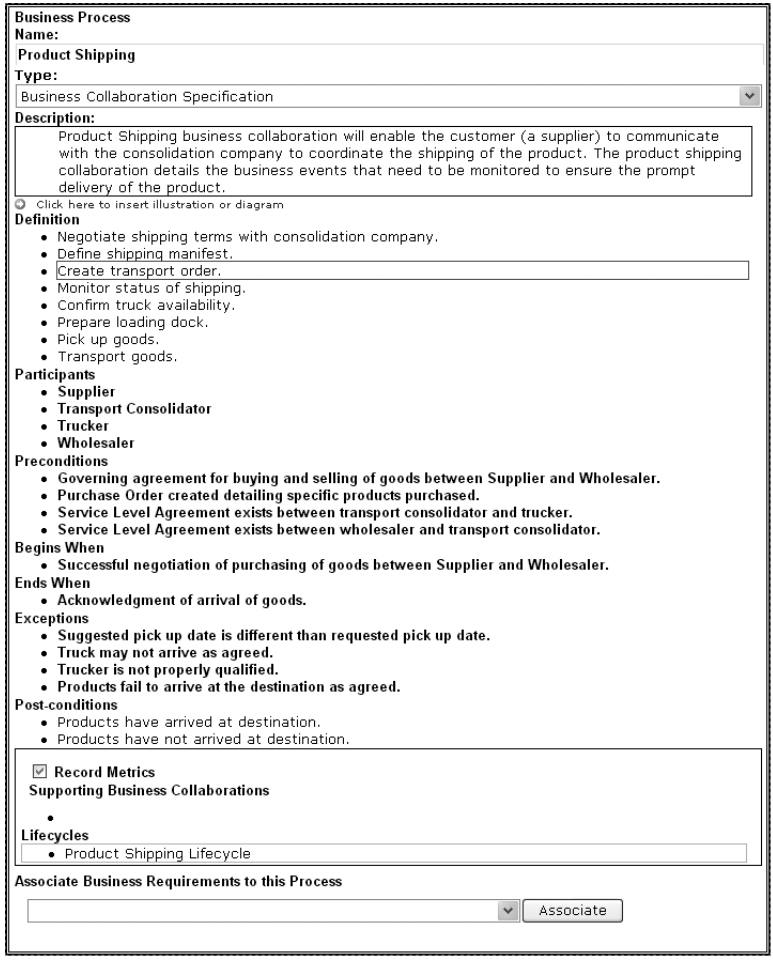


Figure 7. A business collaboration modeling tool capturing information about product shipping

Pam is able to use the specifications to communicate her requirements to Zack and the rest of his team in the IT organization in a way that relates to the kind of work they do.

Particularly, the business operations specification that specifies management requirements helps Zack know which management functions need to be set up to support the business operations requirements as he and his team design the system. The system infrastructure will need to be set up and configured so that appropriate business activities can be managed or logged. The business operations specifications present requirements to IT in a way that can be used to create a business operations health model. A **business operations health model** is a specification of the detection, verification, diagnostic, resolution, and re-verification actions for any manageable condition in a business collaboration. Of course, not all business exceptions are predictable, and not all manageable conditions are business exceptions. But business exceptions already known to impact the business collaboration negatively are contained in the information Pam has collected. The information also contains requirements for other manageable conditions, such as logging certain activities that occur to produce an evidentiary record. The business operations specifications pull these manageable conditions out of the formal business collaboration model and present them in a way that Zack and the rest of the IT organization can work with.

An example of such a manageable condition from the product shipping business collaboration that Northern Electronics wants to pay close attention to has to do with the on-time arrival of the trucks. The business operations specifications call out this requirement formally:

- **Business Operations Requirement (BOR):** The truck shall arrive to pick up the goods at the supplier's warehouse on or before the specified **SuggestedPickupTime.**

Two tables describe the details around this business operations requirement.

Table 1. Business Operations Health Model (Detection Information)

Problem	Health State and Operational Condition	Indicators	Operational Alerts	Criticality
Violation of BOR	Degraded; Integrity Problem	Event (ID = EVENT_ TRUCK_ARRIVAL_ DELAYED) This event is logged to the event log.	An e-mail notification will be sent to the Business Operations Managers notification group.	An event processing rule generates an alert with the severity set to Severe.

Table 2. Business Operations Health Model (Verification, Diagnostic, Resolution, and Re-verification Information)

Problem	Verification Procedure	Diagnostic Information	Resolution and Final Desired State	Re-verification
Violation of BOR	Manual check that truck has not arrived (should return TRUE). Operational condition confirmed as integrity problem.	Manual check. Call to transport consolidator to find out where truck is.	Manual resolution. Supplier renegotiates truck arrival time with consolidation company and updates the time in the system. Supplier updates. Suggested PickupTime. The logs and databases are both updated with new information.	Manual check that truck has not arrived (should return FALSE).

Table 1 shows how to detect the condition (in this case, a business exception) and Table 2 shows how to proceed after the detection has been made. The final resolution in this case from the IT domain is just to provide feedback to the business operations managers. No control action by the IT management system is needed to change the state of the system.

More detailed information on communicating business operations requirements to IT using business operations modeling can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Building the Product Shipping Solution

With the business operations requirements well documented in specifications, and with a good line of communication between him and the business operations team, Zack is able to move forward with his team on the design, implementation, and roll-out of the IT solution.

Zack's team works differently from Pam's. Just as there is a program manager in the business realm, there is also a program manager on the IT side working under the guidance of Zack. Her name is Jackie.

The work now is to design, implement, and build out the infrastructure in the IT domain that supports the product shipping business collaboration. Jackie takes the requirements

contained in the business operations specifications as a starting point and begins to build a plan for meeting them. Throughout the course of the project, she monitors the project status and works with the following individuals who have the noted skills and tasks:

- Tom. He is the *Solution Architect,* provides the technical knowledge and drives the actual solution design. Tom understands how different business systems work together. He understands technology capabilities and its limitations. He communicates with business system analysts to understand the business functions that need to be supported and built and the desired business user experience. He creates and communicates the application design to the software developers and assigns tasks to the project. He is also responsible for determining software development methodologies and tools.
- Sam. He is the *Lead Developer,* develops the software to meet the functional and non-functional requirements. Sam writes and runs unit tests, runs code analysis, writes load tests, and checks in work. When any bugs are reported, Sam diagnoses and fixes the bugs, and checks in the work.
- Kim. She is the *Test Lead,* generates test cases for the application. She checks the build status, loads and runs tests, and reports bugs.
- Dan. He is the *Application Administrator,* understands the capabilities and constraints of the server and network infrastructure. Dan works with solutions architects to understand application design and communicate necessary operational requirements into the application design and development. Dan uses operational tools to codify operational requirements into management scripts for monitoring and automated tasks. He deploys the application on the appropriate computers. He also troubleshoots and fixes advance operation faults.
- Craig. He is the *Application Support Analyst,* monitors the performance and health of the services. He troubleshoots and fixes basic operation faults.

Zack's team uses the service life cycle approach shown in Figure 8 to work together to meet the business operations requirements.

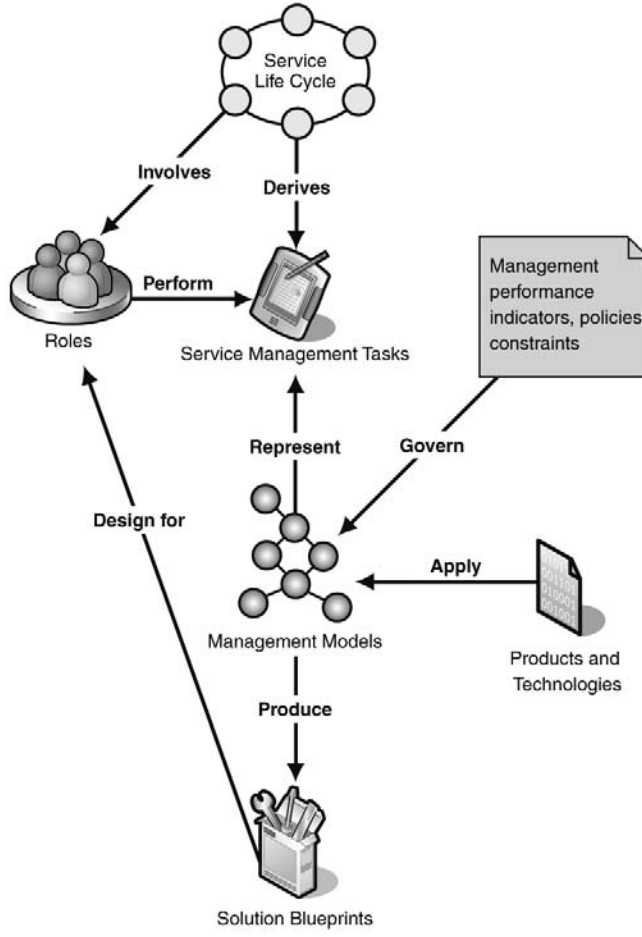


Figure 8. Using the service life cycle

The service life cycle approach helps the team:

- Consider the non-functional attributes that allow services to be operated, maintained, and upgraded over the course of time.
- Identify the roles involved in the service life cycle and derive management tasks that each role performs.
- Describe the conceptual elements and constraints of the management tasks using a collection of management models.
- Build solutions for the management models using products and technologies. The resulting tools and applications are tailored to meet the needs of the individual roles in the organization.

More detailed information on issues in aligning business and IT through dynamic modeling, the service life cycle approach, and IT systems management can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Design Phase

Tom's primary responsibility as the architect is to design the solution. To do that, he considers:

- The business operations requirements as provided in specifications by Pam and Zack.
- The non-functional attributes that enable services to be operated, maintained, and upgraded over the course of time.

Tom analyzes the business operations specifications that Pam and Zack have given him as well as IT organizational policies to create a design for the solution. He focuses on two areas of design, the Web service solution design and the Web service health model.

Tom considers many issues as he develops his design:

- Service metadata:
  - Service configuration (system and network parameters).
  - Message security policies and business rules.
- Identities and entitlements.
- Service composition.

- Roles of directory services:
  - Managing security data (credentials and entitlements).
  - Managing service configuration and policies.
- Consistency with infrastructure.
- Operational processes.

Tom follows a model-driven development approach that uses highly structured information models to capture the intentions and desired results of stakeholders like Pam and Zack. In terms of the business operations requirements in particular, this approach ensures that Tom proactively thinks about the instrumentation and tasks required to achieve the desired consequences; he also thinks about actions for the mitigation strategies in the event of business exceptions.

Web Service Solution Design: Business Services to Web Services

Tom's design approach is to rely on Web services to implement automatable business services within the product shipping business collaboration. Tom expects that some of the Web services will be internal to Northern Electronics, while others will be used directly by business partners.

Tom works with Pam and Zack to identify candidate Web services for the product shipping business collaboration. They review the way shipping is currently done while thinking about what can be improved through automation. Pam and Zack already considered this question from the business perspective when they looked at the business services involved in product shipping. While the business operations specifications capture the detail of that analysis in a formal way, Tom needs the consultation with Pam and Zack to help him get a thorough understanding of what is happening and why the specifications have the requirements they do.

One example of something they review together has to do with when the product shipping business collaboration begins. The negotiation between customers and Northern Electronics is done verbally, usually over the phone. This is outside the scope of the product shipping business collaboration. Moreover, the negotiation is not a good candidate for something that can be automated because the human interaction is probably a necessary aspect of negotiating the purchase. The starting point of the product shipping business collaboration is when the PO is created. The PO is an artifact that can be stored electronically instead of on paper.

Another example of something they review together is the ordering of transport. After the PO is created, the supplier shipping clerk checks to make sure the items are in stock. If the items are in stock, the supplier shipping clerk works with the transport consolidation clerk to coordinate the transport that will pick up the shipment and deliver it to the customer.

Yet another example of something they review together is the security requirements. Identities of the participants in the business collaboration must be represented as digital identities in the Web services domain. The processes of authentication and authorization are necessary to control access, usage, and administration of the services, especially across administrative boundaries—in this case, between the supplier and the transport consolidator. Tom has to make sure he understands issues such as who has the authority to create or change transport order information and who can confirm acceptance of transport. These issues are especially important to meet the business operations requirements, which outline strict legal requirements. For example, they specify the role of who in Northern Electronics is actually allowed to order transport from a transport consolidator (and therefore commit the company to payment for the ordered transport). (Other requirements related to security and identity, such as logging specific activities as they are performed by certain roles, are expressed in the business operations specifications.)

Tom agrees with the conclusion from Pam and Zack that the interactions between the supplier and transport consolidator are good candidates for being automated and implemented using Web services. He reasons that:

- If the transport consolidator has a Web service the supplier shipping clerk can use to send his request to, instead of relying on a phone call, it is unlikely the request will get lost. The supplier shipping clerk can also get proof that the request was received. The transport consolidation clerk can then process the request and generate an acceptance.

- If the supplier has a Web service the transport consolidator can use to send his acceptance of the transport order to, instead of relying on a fax machine, it is similarly unlikely the request will get lost. The acceptance can contain the details and terms of the product pickup. The transport consolidation clerk can get proof that the acceptance was received and thereby that the terms were agreed to.

After Tom examines the business collaboration further, he reasons that the supplier shipping clerk and loading dock clerk can work more efficiently by using an internal Web service to let them know about the shipments that need to be loaded onto the docks in the correct order based on the trucker's expected time of arrival. They can use this internal Web service to record the trucker's credentials and can also use it to confirm that the shipment was picked up. In turn, the internal Web service can then automatically send a pickup confirmation to Acme Consolidation Company's Web service.

Based on this analysis, to meet the requirements laid out in the business operations specifications and to create the internal efficiencies he identified, Tom proposes a design that includes three Web services to facilitate the product shipping business collaboration:

- ShippingService Web Service. This is the supplier's Web service used to send and receive the details of the shipment pickup.
- PickupService Web Service. This is the supplier's Web service used internally to be notified of product pickup and to confirm the shipment was picked up.
- TransportService Web Service. This is the transport consolidator's Web service used by the supplier to initially order the transport and finally to confirm that the shipment was picked up.

Obviously, the TransportService Web service is not something that Northern Electronics can implement, because it belongs to the transport consolidator. But it is a needed part of Tom's design.

Tom works with Zack and Pam to coordinate the development of the TransportService Web service with the chosen transport consolidator business partner, Acme Consolidation Company. If the approach proves successful, Pam and her team can organize similar coordination with other transport consolidation business partners.

Note: The coordination with Acme Consolidation Company is outside the scope of this discussion, which simply relies on the coordination happening successfully.

Tom uses the business operations specifications to define the actual interfaces of the Web services.

More detailed information on Web service solution design can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Web Service Health Model: Designing for Operations

The proposed three Web services allow messages to be sent and received in a systematic and verifiable manner. In addition to satisfying the business operations interface requirements, Tom also needs to think about how service instrumentation can be used to provide visibility and control over running instances of Web services. Web service instrumentation may take the following forms:

- Debugging traces can help with diagnosing code execution behaviors.
- Events are used to raise alerts that signify meeting certain management conditions.
- Performance counters are used to generate statistics that reflect the health of the services.
- Probes are internal service states that can be queried and controlled by management applications.

Tom needs to make sure the two Web services that Northern Electronics will implement can be designed for operations. This means that the Web services need to be designed to be managed to meet their performance objectives at multiple levels.

The IT organization uses a health modeling approach to develop a thorough understanding of how the Web services should perform, how to learn about how they are performing, and what to do when they are not performing according to plan.

Health modeling is a process within the design phase of the service life cycle to formally document the understanding of the system designers about what a healthy system is and what an unhealthy system is. It considers in advance all the manageable conditions the designers can anticipate. The health model that results contains detailed consideration of each anticipated manageable condition by laying out how to identify it and the actions to take to manage it. The health model details how to:

- Detect a manageable condition.
- Verify that it really is the condition.
- Diagnose what might be causing the condition as needed.
- Resolve the condition through corrective actions as needed.
- Re-verify the condition to see whether the system is in good health.

Tom works with others in the IT organization and relies on existing policies and procedures to formulate an initial health model for the two Web services he has proposed. This initial plan takes into account the company's standard system-level and application-level health models for Web services (for example, the requirement to perform "heartbeat" checking as a way to ensure a Web service is still running). Tom also needs to take into account the requirements in the business operations specifications.

More detailed information on Web service health modeling can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Development and Test Phases

When Tom completes his design, he reviews it with Pam and Zack to make sure they agree it meets their requirements. He then works with Jackie to coordinate the hand-off of the design to the lead developer Sam and his team.

One of Sam's tasks is to use the solution design as the basis for developing the detailed specifications of the system. The specifications will take many other factors into account, such as development policies for the IT organization, existing infrastructure, and existing components. Sam must take into account many issues in developing the specifications, including:

- How to make the application testable.
- How to make the application deployable and configurable.
- How to deal with versioning.
- How to deal with security.
- How to handle deprecation.

Another of Sam's tasks is to use the health model as the blueprint for instrumenting the solution so it can be managed. Instrumentation is used to provide visibility and control over running instances of Web services. Instrumentation may take the following forms:

- Debugging traces can help with diagnosing code execution behaviors.
- Events are used to raise alerts that signify meeting certain management conditions.
- Performance counters are used to generate statistics that reflect the health of the services.
- Probes are internal service states that can be queried and controlled by management applications.

Based on these specifications, Sam's team programs the software for the solution. Sam works with Jackie to come up with the schedule for code reviews and hand-off dates to Kim in test.

Kim collaborates with her team to design the tests that will verify application compliance with governance. She designs the test harness to subject the application to different

kinds of tests, for example, functional, performance, stress, and security penetration testing. The key issues Kim must decide are:

- Unit testing.
- Datacenter deployability.
- Health model conformance.
- Manageability.

The pre-production test environment simulates real-life conditions to validate the Web services design assumptions. Kim works with Jackie and Sam to schedule the test passes and bug triages.

More detailed information on Web service solution design can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Deployment Phase

After the new Web services have satisfied all testing requirements, they are ready for deployment into the production environment. Dan starts by preparing the network infrastructure, hardware, and operating systems. He then deploys the application to the production environment. Key issues that Dan must decide are:

- Provisioning.
- Versioning.
- Discoverability.
- Windowing of update rollout.
- Cluster management.
- Integration with infrastructure change management.

More detailed information on Web service deployment can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Operations Phase

After the application is deployed, Jackie's work is mostly done. Craig and his team take over to monitor the application to determine whether there are any operational problems. The goal of this phase is to maintain system health according to the health model. Key issues that Craig must focus on are:

- Health, performance, and service-compliance monitoring.
- Root-cause analysis.
- Metering.
- Service control and configuration.

More detailed information on Web service health modeling can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Change Management Phase

During the change management phase, Craig, Sam, and Tom look at the operations data for the service including real-time and post-processing. This phase involves a decision-making process to decide on design or implementation changes. Key issues the operations staff, developers, and architects must focus on are:

- Health model metrics.
- Deprecation.
- Infrastructure adjustments for scaling.

Over time, new business operations and technical requirements will necessitate re-evaluation and possible changes to existing implementations of the product shipping business collaboration and the IT system that underpins it. As these changes occur, Northern Electronics will be faced with the challenges of minimizing or insulating the change impact to existing participants within the business collaboration, as well as to consumers of the Web services.

For example, if Northern Electronics decide to extend the reach of the product shipping business collaboration to transport consolidator partners who ship overseas (instead of those who only use national trucking), adjustments will need to be made to the formal model of the business collaboration—for example, to take into account international customs regulations. To support these new requirements, additional input or output data may need to be passed between the supplier and the transport consolidator. These changes in business operations requirements can be communicated clearly and effectively to the IT organization by means of revisions to the business operations specifications.

Because Web services are used to enable parts of the business collaboration, a couple of technical options may be considered to facilitate such new data exchanges. Northern Electronics may introduce a new Web service interface to support the new data parameters. Or it might use a Web service proxy façade to support the existing interface, with the proxy performing data transformation and internal service routing to support messaging requests from current and new service consumers. Depending on the Web services solution design, changes to the Web service interface or message schema may affect the Web service consumer and the Web service proxy. In either case, the consuming party would need to discover the new Web service definition to interact with the new Web service endpoint.

The Web services that underpin the product shipping business collaboration could require changes not necessitated by changes in business operations. Such changes could include a slight change in the Web service access addresses, or substantial changes to an existing message schema that proves unuseful or incorrect. The IT organization should be able to make these changes on its own, but it will need to coordinate the changes with the business operations staff for their consideration of the impact and feedback on time frame within which to accomplish the changes.

Conclusion

This document has followed the story of how Northern Electronics worked with its business partners to improve its product shipping process. The discussion showed how business operations requirements can be supported in the IT organization. Other documents in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* rely on the Northern Electronics scenario to examine in more detail from a variety of perspectives how to manage connected systems.

Notes

Communicating Business Operations Requirements to IT: Using Business Operations Modeling in the Northern Electronics Scenario

Architecture Chronicles

Dynamic Modeling: Aligning Business and IT

Jim Clark, Max Morris, and Dave Welsh with Frederick Chong
September 2005

Applies to:

- Enterprise Architecture
- Solution Architecture
- Service Oriented Architecture (SOA)
- Service Oriented Management (SOM)
- Application Integration
- Business Process
- Business Operations Modeling

Summary

The *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* seek to present to business, solution, and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. This document focuses on communicating business operations requirements to the IT organization by applying business operations modeling to the product shipping business collaboration in the Northern Electronics scenario.
(38 printed pages)

Click here (http://www.microsoft.com/downloads/details.aspx?FamilyID=121755f1-e339-4b83-ba10-143f5671c292&displaylang=en) to download the Dynamic Modeling: Aligning Business and IT Demo.

- Download the sample
- Installer Version
- MSDNSamples\Arch

Contents

This Document

This document is part of the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT*. This volume seeks to present to business, solution and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. The information map to the series provides an up-to-date description and cross-index of the information available and can be found at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Abstract

This document examines how to communicate business operations requirements to the IT organization by applying business operations modeling. It examines business operations concepts, business operations modeling methodologies, and the practical aspects of modeling using worksheets. It connects these concepts together through a set of proof-of-concept forms running within Microsoft Office InfoPath® 2003 to first model the product shipping business collaboration from the Northern Electronics scenario, and then generating a set of specifications for the IT organization to use.

Acknowledgments

Many thanks to Nelly Delgado for her help with technical writing, Claudette Siroky for her graphics skills, and Tina Burden McGrayne for her copy editing.

The authors would also like to thank Klaus-Dieter Naujok and Allyson Winter (of Global e-Business Advisory Council) for their contributions to the development and documentation of the proof-of-concept business operations modeling worksheet tool used to illustrate this document.

Introduction

This document focuses on communicating business operations requirements to the IT organization by applying business operations modeling to the product shipping business collaboration in the Northern Electronics's scenario. Background information on the Northern Electronics scenario can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

The document examines how the business program manager and a business analyst can apply a business operations modeling methodology to derive a set of requirements for use by the IT department in building a solution. Specifically, this document discusses:

- Business operations concepts.
- Business operations modeling methodologies, including a background on such modeling approaches as well as a brief discussion of a modeling approach used to illustrate this document.
- Modeling through worksheets, including a discussion of worksheets concepts and basic procedures such as the following:
  - Production of a solution set that contains business operations requirements information for use by the IT organization, with a specific focus on the inputs to the service management and health modeling functions.
  - Demonstration of the approach using the Northern Electronics scenario.

Business Operations

When two businesses collaborate with each other in a value chain, whether as supplier-customer or partner-partner, the dealings they have with one another generally follow procedures the two agree upon in advance. These procedures are usually defined with enough clarity through a contract that each company can clearly know its role, can perform the activities it has committed to, and when its commitments are fulfilled, can collect whatever benefit or payment is due from the other.

Note: The approach presented here readily applies as well to two organizations within a company. The difference is a matter of business trust. There is less business trust between two companies than between two organizations within a company. When there is more business trust, there is less potential for business relationship drift; when it does occur, it is easier to drive change to achieve business state alignment.

It is the job of the business operations organization to put into place the infrastructure necessary to support at an operational level a company's own side of the business collaborations it has with others. In performing its work, the business operations organization maps contractual requirements onto business capabilities as it defines, implements, and operates a collection of business processes.

The business operations organization does not work in isolation. It works carefully with corresponding business operations organizations at other companies with whom its company collaborates throughout all phases of the business collaboration in question. Considerable attention is paid to these interactions, or shared activities, to make sure the right information is exchanged in a timely and secure way without any loss of information integrity. These exchanges are done using a variety of technologies, including face-to-face meetings, faxes, phone calls, postal mail, express mail, e-mail, and other forms of specialized, structure information exchange using computers. Which technology is used and how it is used depends on many different circumstances such as historical precedent, industry standards, regulation, and business capability. Therefore, the business operations organization also works closely with the IT organization to support the IT organization's work in defining, developing, and operating technology solutions that realize the collection of business processes.

There are two ongoing challenges business operations organizations have as they oversee the running of the business, as illustrated in Figure 1:

- Business relationship drift. This situation occurs as the two businesses in a business collaboration experience different levels of information about the ongoing business activities they share.
- Business operations drift. This situation occurs as the business operations organization and the IT organization within a company experience different levels of information about requirements and ongoing operational performance of business processes and the technology solutions that support them.

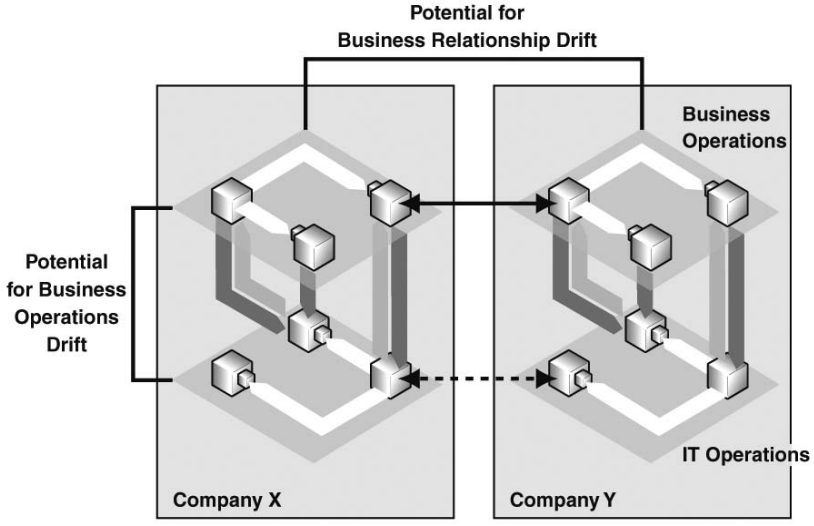


Figure 1. Illustration of business relationship drift and business operations drift

Business relationship drift in a business collaboration is inevitable. It's not possible for one business to have all the same information at the same time as the other. But business relationship drift does not become a problem, and remains hidden away, when a business

collaboration proceeds as expected. The different, shared activities unfold in a timely way, continuously bringing the business relationship back into alignment, and each business continues to experience the outcomes it expects.

It's when business exceptions occur—when an expected event is delayed, or an expected event occurs out of sequence—that business relationship drift manifests itself as a problem. Typically the manifestation is a cascade: one important activity completes late, or not at all, leading to other activities not starting, and so on. It's very hard for the businesses to get back on the same page, or to get back into business relationship alignment. Figuring out what has gone wrong and where, and getting things back on track (if that is even possible) can take a lot of time, energy, and expense.

With good up-front planning and experience to draw on, the difficulty in handling business exceptions isn't in knowing what to do about them. Most issues can be anticipated, and a good plan will include some way of dealing with the completely unexpected. Instead, the difficulty in handling business exceptions is mostly about getting useful information about an exception that has occurred to the right person in time, or even ahead of time, so that he or she can do something about the problem as it happens—or before it happens. When the right information arrives to the right person in time, there's a better chance that whatever intervention the person can implement can have a more significant mitigating effect.

Getting useful information about what exceptions are occurring to the right person in time is no easy task, and it is the hallmark symptom of business operations drift. Expected performance is not achieved, but the current performance level actually remains unknown. The information to achieve alignment of business operations usually exists, but it is almost always distributed in a piecemeal fashion throughout the technology solution that implements the business processes that work together to enable the business collaboration. In such situations, non-performance of a contractual commitment isn't something that is known at the business operations level until some time later, when an after-the-fact audit is performed. Such an audit, usually directed manually, uncovers that certain requirements from the contracted service level agreement were or were not met. In failure cases, penalties are often assessed. Meanwhile, in real-time, the non-performance of the commitment often leads to cascading business exceptions in the business operations.

The key to ensuring business relationship alignment between two businesses would seem to be business operations alignment between the internal business operations and IT operations organizations of each business. If there were some way for the IT organization to know in advance what business operations requirements needed to be met so that IT operations could then let business operations know as those requirements were not met, the business operations team would be better able to detect business exceptions that will likely worsen business relationship drift as they occur—or even before they do.

This document argues that a modeling approach to formalizing business collaborations can facilitate the identification of conditions that will worsen business relationship drift. When these conditions are identified and transformed into a model that the IT organization can use, the IT solution can be designed, implemented, and operated in way that combats business operations drift. With tighter coupling between business operations and IT operations at both businesses in a business collaboration, following this approach, much more timely knowledge about the state of shared business activities can be known to both parties. This condition is referred to as **business state alignment,** and it is illustrated in Figure 2. With business state alignment achieved, business operations can run more smoothly and unnecessary expense and downtime can be avoided.

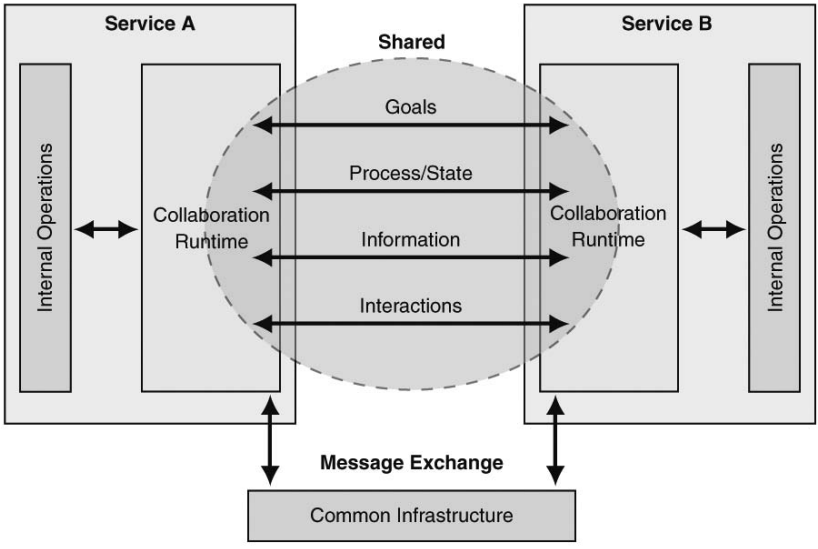


Figure 2. Illustration of business state alignment

Business Operations Modeling Methodology

The process of modeling business operations is a team effort and provides several benefits to the organization:

- Means of communication. Modeling is a means of communication across and between organizations and within an organization.
- Logical framework. Modeling provides a logical framework within which to relate an agreed collaborative view and partner responsibilities.
- Clarify business objectives. Modeling helps clarify business objectives by understanding and relating dependencies between business management, business operations, application systems, and IT operations.

Background

Several approaches to formally modeling the business collaborations that occur when running a business have been developed over the last decade. The Open-EDI Reference Model is commonly understood to be the basis for most of these approaches (ISO/IEC 14662), (http://www.iso.ch/) or (www.disa.org/international/is14662.pdf). More recently and of particular interest to the discussion in this document, UN/CEFACT developed a well-known derivative named the UMM. The UMM was developed by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT). UN/CEFACT "supports activities dedicated to improving the ability of business, trade and administrative organizations, from developed, developing and transitional economies, to exchange products and relevant services effectively." UMM stands for UN/CEFACT Modeling Methodology. More information about UN/CEFACT and the UMM can be found at http://www.unece.org/cefact/.

The UMM was developed as a technology-neutral methodology to formally describe a business collaboration from multiple perspectives. At the heart of the UMM is a metamodel derived from economic and accounting theories, as well as from commercial best practices and laws from around the world. The use of the UMM and its metamodel is not restricted to only business collaborations in bilateral value chains. Both have been used widely (in part or in full) by industry standards organizations to describe the standard business collaborations that occur within their industry-specific value chains. Such industry standards organizations include AIAG (http://www.aiag.org), SWIFT (http://www.swift.com/), GS1 (http://www.ean-int.org/), and ISO TC 204 (http://www.iso.ch/).

Several incremental changes to the UMM and its metamodel have been made by two of the co-authors. (Jim Clark and Dave Welsh were both collaborators on the development of the UMM and also collaborators on the development of the variant used here. Notable changes include structural corrections to the UMM Business Relationship View (and a renaming of the variant view to Business Collaboration View) as well as the introduction of a sub-model of the BCV specifically for SLA modeling.) Though similar in many

ways, because the variant is distinct from the UMM, it has a different name, the Business Operations Modeling Methodology (BMM). The BMM is used to illustrate the discussion about business operations modeling in this document. Salient points about the BMM are provided in this document. However, full elaboration of the BMM is not provided and the interested reader is referred to the UMM for further detail.

Supporting Multiple Perspectives: Key to Problem Solving

Considering the complexity of business and process automation problems, fully describing a business solution requires input from many people with different backgrounds and disciplines. Unfortunately, without using care, the varying consistency and precision of their input can directly impact how well-formed the solution is. An architecture designed up-front to support this variety and coalesce it all into a harmonized view helps ensure the solution that emerges is well-formed.

By separating business and technical perspectives, it is possible to provide a means for business people to model what they know without requiring specialized knowledge or skills. These models can then be mapped into targeted technologies and architectures.

BMM Structure

The BMM comprises a set of architectures, patterns, and business semantics defined in accordance with certain business reference models and ontologies (domain-specific schemas). The BMM provides for the transformation of process and information as defined in one view (or perspective) to the next view without the loss of semantic or computational integrity.

The BMM includes five different perspectives, each with its own concepts and defining its own set of canonical semantics. As the BMM is put to use, each perspective adds more detail to the solution definition. Concepts defined in one perspective are transformable into the other perspectives. The five perspectives are:

- Business Domain. This perspective describes the context for the business operations modeling.
- Business Collaboration View. This perspective describes the business goals and the business collaborations involved in achieving them. It enumerates the scenarios, inputs, outputs, constraints, and system boundaries.
- Business Transaction View. This perspective describes the logical transactions of information between business entities that are involved in accomplishing the business collaborations.
- Business Service View. This perspective describes the exact information exchanges in accomplishing the business collaborations.
- Implementation Framework View. This perspective describes the technology-specific bindings for the information exchanges.

Certain patterns that are useful for reuse are also defined in the BMM, including business collaboration patterns, business transaction patterns, and service interaction patterns.

Finally, a well-formed metamodel defines the syntax and semantics for each view. Uniformity of notation and precision of semantics provide concise and unambiguous business process definitions.

Business Domains: Giving Context to the Problem

The **business domain view** (BDV) is the framework for understanding business area process interrelationships. A particular BDV is typically a domain-specific business reference model that is used to categorize business processes to aid in business process definition, business process integration, and auto discovery.

Business areas comprise process areas. A process area is a set of business processes that implements a particular business operations model.

Business operations areas such as "Deliver stocked product" and "Deliver make-to-order products" are two different business operations models that use many of the same business processes.

The **business domain model** (BDM) provides a facility for categorizing and classifying processes and information according to a domain taxonomy. A taxonomy that is defined according to a particular domain ontology will closely represent the structure or organization of the problem space of the domain. This same taxonomy can be used to classify business processes to determine where within the organization of the problem domain these processes or solutions fit. Table 1 presents many of the BDM elements.

Table 1. Business Domain Modeling Elements

Element	Description
Business Process	A business process is a set of activities that are conducted to achieve a certain goal or objective. Inputs to the business process are specified in the preconditions and outputs from the business process must be specified in the post-conditions. In the BDV, business processes are used to identify and categorize business processed at a top/root level.
Business Domain Map	A business domain map is a framework for understanding business area sub-process interrelationships. Typically a business domain map is a domain-specific business reference model that is used to categorize business process to aid in business process definition, business process integration and auto discovery. Registries and repositories would support the ontology and taxonomy specified by a business domain map.
Business Area	A business area is a category of decomposable business process areas. A business area collates process areas.
Process Area	A process area is a category of business processes and business transactions. A process area collates business processes and business transactions.
Business Entity	A business entity is a real-world, first-class definitive phenomenon that exists; it has structure and a life cycle (state). In the context that the world understands what an object is, an object is usually system-oriented and a life cycle is optional, structural only. A business entity has structure and a life cycle (state). The concept of entity defines the real world instance of an object, not just a representation.
Location	A location identifies or defines the spatial aspect with respect to time that a business entity or business process can be realized.
Stakeholder	A stakeholder represents a role played by a business entity in relation to the business domain view, business areas or process areas.
Business Event	A business event is an observance of and identification of the time, location, and type of an occurrence or activity.

Element	Description
Requirements	A requirement, identified by a requirement category, that is either partially or completely satisfied by this use case. Requirements categories include 1) static and structural; 2) dynamics; 3) exception conditions; 4) non-functional; and 5) system administration.
Justification	A justification defines the set of rules and constraints that rationalizes the adoption of the solution identified in the associated business area or process area.

Business Collaborations: Process Elaboration

The **business collaboration view** (BCV) is the view of a business process model that captures and defines the business scenarios, inputs, outputs, constraints, and boundaries for business transactions and their interrelationships.

The goal of the BCV is to fully specify units-of-work that achieve defined business objectives. Objectives are elaborated in the form of business requirements and constraints.

To arrive at the BCV, business processes from the BDV are mapped into high-level business processes. These are then further specialized into sub-level business processes. This approach continues until the specialization reaches a level of specificity that defines a scope particular to a business collaboration. A **business collaboration** is a set of activities that are conducted by two or more parties to achieve a certain goal or objective. The BCV defines a business collaboration according to economic and jurisdictional ontologies.

The business collaboration is the primary concept within the BMM, providing modularity to sets of business transactions. The key difference between a business process and a business collaboration is the requirement of two or more parties having a shared perspective. This multiparty nature of the business collaboration brings many required aspects into the problem and solution spaces. Such aspects include who does what (role played), relationships between the parties (trust), commitments, expectations, ownership, and the consumption and production of resources. With a business process, these aspects are reflected as part of the definition, but they are optional. With a business collaboration, such aspects are integral to its definition. They must be specified to ensure the completeness of the solution. (This difference also plays an important role in defining a set of immutable properties that make up a business collaboration pattern.)

Business collaborations specify the relationships between business parties. They provide the business events that move to and from business parties. They define business entities or objects and their life cycles and states within the context of the business collaboration. Finally, they define business constraints and rules according to the context of the business collaboration.

The **business collaboration model** (BCM) provides for the structure and semantics necessary to elaborate operational, functional, and informational aspects of a business collaboration. A set of extensions elaborate the logical relationships, business relationships, commitments, expectations, resource consumption, and resources production of a business collaboration. Table 2 presents many of the BCM elements.

Table 2. Business Collaboration Modeling Elements

Element	Description
Business Entity	A business entity is a real-world, first-class definitive phenomenon that exists; it has structure and a life cycle (state).
Business Entity Life Cycle	A business entity life cycle defines the observable conditions that may occur to a business entity within a given context
Business Event	A business event is an observance of and identification of the time, location, and type of an occurrence or activity.
Business Process	A business process at the BCV level is a fully detailed definition of a business process, opposed to the business processes defined at the BDV level.
Business Rules	A business rule is a set of computational constraints that are evaluated against the validity of the structure or state of business entities and business processes within the defined context.
Partner	A partner is a participant in a business collaboration.
Partner Type	A partner type is a category of participant in a business collaboration. Partner types include manufacturer, distributor, retailer, end user, carrier, and financier.
Process Life Cycle	A process life cycle defines the observable conditions that may occur to a business process within a given context.
Business Collaboration	A business collaboration is a set of activities that are conducted by two or more business parties to achieve a certain business goal or objective.
Economic Elements	An economic element is used to express the structure, interrelationships, and behavior of economic entities in the BCV of an economic model depicting resources, events, and agents.

Element	Description
Agreement	An agreement is an arrangement between two partner types that specifies in advance the conditions under which they will trade (for example terms of shipment, terms of payment, and collaboration protocols). An agreement does not imply specific economic commitments.
Commitment	A commitment is an obligation to perform an economic event (such as a transfer ownership of a specified quantity of a specified economic resource type) at some future point in time. Order line items are examples of commitments.
Claim	A claim is a request for fulfilment of a reciprocating economic commitment based on an economic event. A claim automatically arises when there is an unrequited economic event, for example a partner needs to be compensated for an economic event that is fulfilling an economic commitment.
Location	A location identifies or defines the spatial aspect with respect to time that a business entity or business process can be realized.
Contract	A contract is the specification of a set of reciprocal business commitments between two or more parties. Each commitment is fulfilled by one or more business events. The contract also defines terms and conditions for the execution of a business collaboration

The BMM has identified five business collaboration patterns that are commonly used in business:

- Negotiation
- Auction
- Escalating commitments
- Order fulfilment
- Settlement

Business Transactions: Adding Legal Enforceability

The **business transaction view** (BTV) is the view of a business collaboration that captures the semantics of business entities and the exchange of business information between business partners as they perform business activities. The BTV facilitates the definition and specification of a legally binding business transaction.

A **business transaction** is a set of business information and an exchange of messages (of the business information) between two business partners that must occur in an agreed upon message format, in an agreed upon sequence, and under certain, previously agreed upon time period conditions. There are two types of messages that get exchanged in a business transaction: business action messages that contain business information mapped to business events, and business signal messages that are used by the business services during processing and are mapped to a business action message technical processing event instead of a business event.

There is a clear intent by a business transaction, agreed to ahead of time by both business partners. A **business transaction pattern** provides a basic mechanism for each business partner to do business transaction state management and business-to-business process alignment. When one business partner knows a business event has or should occur, the other business partner can also align or share in the knowledge of that business event.

Every business transaction follows one of the six business transaction patterns listed in Table 3. These business transaction patterns comprehensively cover all the known legally binding business collaborations at the lowest level of request/response interaction between two business applications. The specific business transaction pattern to be used in a business collaboration is determined by application of the modelling methodology in the BCV. A short description of each business transaction pattern is provided in Table 3.

Table 3. Business Transaction Patterns

Business Transaction Pattern	Description
Offer & Acceptance	Used to model an "offer and acceptance" business transaction that results in a residual obligation between both parties to fulfill the terms of the contract. In other words, a commitment between the two parties is left over after the transaction is completed. The pattern specifies that an originating business activity sending a business action message to a responding business activity, may return a business signal or business action message as a last responding message. The pattern mandates the acknowledgment of the requesting business action message when it passes a "business acceptance" test, which is the content validation step. This acknowledgment can be substantive or non-substantive, it may contain the terms of acceptance of a contract or a general auditable business signal.

Business Transaction Pattern	Description
Request & Confirm	Specifies the exchange of a requesting and responding business action message and usually has no residual obligation between both parties to fulfill the terms of a contract. Different from request & response because it requires confirmation. For example, a request for authorization to sell certain products expects a confirmation response to the request that confirms if the requestor is authorized or not authorized to sell the products. An example is obtaining order status.
Request & Response	Specifies the exchange of a requesting and responding business action message and usually has no residual obligation between both parties to fulfill the terms of a contract. For example, a request for price and availability does not result in the responding party allocating product for future purchase and it does not result in the requesting party being obligated to purchase the products. This information may be dynamic, for example, obtaining a price for an airline ticket given certain inputs (location, dates of travel) and availability.
Query & Response	Specifies one business action message as output and one business action message as input. The query & response pattern does not permit the return of auditable business signals, or receipt acknowledgment, or business acceptance acknowledgment. The responding activity is most likely to be serviced by an organizational role, typically not by an employee role, and there is no non-repudiation requirement for these activities. An example may be the sending of static information, such as a store catalog, to a customer.
Notification	Specifies the exchange of a notifying business action message and the return of an acknowledgment of receipt business signal. This pattern is used to model a formal information exchange business transaction that therefore has non-repudiation requirements.
Information Distribution	Specifies the exchange of a requesting business action message and the return of an acknowledgment of receipt business signal. This pattern is used to model an informal information exchange business transaction that therefore has no non-repudiation requirements.

Each business transaction pattern stipulates parameters that need to be specified by the service provider and service consumer.

Depending on the pattern, the service consumer may specify these parameters (valid values are shown in Table 4):

- Time to acknowledge receipt *(time interval)*. Both parties agree to verify receipt of a requesting business document within a specific duration.
- Time to acknowledge acceptance *(time interval)*. Both parties agree for a business acceptance document to be returned by the responding party after the requesting business document passes a set of business rules.
- Time to perform *(time interval)*. Overall time both parties agree to perform a total business transaction.
- Authorization required *(Boolean)*. True if the identity of the sending role is verified, false otherwise.
- Non-repudiation of Origin and Content (Boolean). True if the business activity stores the business document in its original form, false otherwise.
- Non-repudiation of receipt (Boolean). Both parties agree to mutually verify receipt of a requesting business document.
- Recurrence of transmission (Int). Agreement to the number of times to retry a transaction when a time-out exception condition is signaled.

Table 4. Service Provider Parameters

Business Transaction Pattern	Time to Ack Receipt	Time to Ack Accept	Time to Perform	Auth Req	Non-repudiation of Origin and Content	Non-repudiation of Receipt	Recurrence of Transmission
Offer & Acceptance	Time Interval	Time Interval	Time Interval	True	True	True	Integer
Request & Confirm	NULL	NULL	Time Interval	False	False	True	Integer
Request & Response	NULL	NULL	Time Interval	False	False	NULL	Integer
Query & Response	NULL	NULL	Time Interval	False	False	NULL	Integer
Notification	Time Interval	NULL	Time Interval	False	True	True	Integer
Information Distribution	Time Interval	NULL	Time Interval	False	False	False	Integer

Depending on the pattern, the service consumer may specify these parameters (valid values are shown in Table 5):

- Time to acknowledge receipt *(time interval)*. Both parties agree to verify receipt of a requesting business document within a specific duration.
- Time to acknowledge acceptance *(time interval)*. Both parties agree for a business acceptance document to be returned by the responding party after the requesting business document passes a set of business rules.
- Time to perform *(time interval)*. Overall time both parties agree to perform a total business transaction.
- Authorization required *(Boolean)*. True if the identity of the sending role is verified, false otherwise.
- Non-repudiation of Origin and Content (*Boolean*). True if the business activity stores the business document in its original form, false otherwise.

Table 5. Service Consumer Parameters

Business Transaction Pattern	Time to Acknowledge Receipt	Time to Acknowledge Acceptance	Time to Perform	Authorization Required	Non-repudiation of Origin and Content
Offer & Acceptance	Time Interval	Time Interval	Time Interval	True	True
Request & Confirm	Time Interval	NULL	Time Interval	True	False
Request & Response	NULL	NULL	Time Interval	False	False
Query & Response	NULL	NULL	Time Interval	False	False
Notification	Time Interval	NULL	Time Interval	False	False
Information	Time Interval	NULL	Time Interval	False	False
Distribution	Time Interval	NULL	Time Interval	False	False

Legal Enforceability

A business transaction specifies a contract formation process between two business partners. Typically a business contract is used to legally bind parties to a clearly stated intention (promise, obligation) and responsibilities of each party.

A business contract usually outlines what each party can do if the intended actions are not carried out (for example, promised services not rendered or services rendered but payment not issued). Prudent business parties contract with one another prior to carrying out their intended business actions to limit their liability and to protect their

business interests. In the BMM, all business transactions are treated as contract formation processes in that there is always an obligation (perhaps not residual; that is, left over after the event) between each of the parties participating in the transaction.

The power of business transaction patterns is that business partners can know whether they are in business state alignment with each other. Business state alignment means that both parties, the sender and receiver, are in sync with each other at a business operations-level perspective. When there is a transfer of information at the technical level, there must also be an acknowledgment and indication whether the information was accepted or not accepted at the business operations level. Thus, the transaction patterns help drive business partners to the same level of understanding to establish business trust.

As it is commonly understood, digital identity management works because it binds a person's identity to cryptographic keys using procedures that require a qualified (trusted) entity to the person to be authenticated to a computing network to which the person should be authenticated. The qualified entity is the basis of the legal chain of trust.

In building business transaction patterns, it was necessary to find a way to connect the intent (or meaning) of a legally binding business transaction to generic software transaction principles. Software transactions can enable business transactions, but business and software transaction types have a very different intent. Software transactions support actions such as rollback that are not allowed in business transactions. Sometimes this restriction is because of the nature of business. Sometimes this restriction is because of regulatory rules.

The goal with defining business transactions in the BMM was to build a general-purpose business transaction model structure. The structure was intended to clearly show how only certain business people, and the business entities who gave those people permission to act, could make promises and legally binding statements for the business entity to their business partners. Furthermore, the structure was specifically designed for all types of business transaction pattern use cases.

The "legally binding" mechanism used to accomplish a binding of a business transaction to a software transaction is referred to as "construed intent." That is, how easy it is to identify an intentional arrangement of business actions that rationally fit with an intent to do legally binding business.

The business transaction patterns (along with their associated service interaction patterns described later in this paper) help legally document whether business intent existed at the exact time a business transaction was concluded. The proper, well-formed existence of a business transaction, together with the service interaction pattern, models in software two important pieces of information to be assessed by a decision maker in trying to determine the intent of a promisor in the possible context of any legal dispute.

At the application level, other information about how the parameters in the business transaction pattern were triggered for inclusion will probably be needed, including the user interface the user experienced. The goal is to enable drawing a conclusion that what the user activated was what the user intended, and what was intended was correctly recorded and transmitted by the application.

For this reason, the business transaction patterns have been designed to provide the basic evidentiary needs in case of a legal dispute. Only a binding to a well-composed technology infrastructure can effectively and efficiently provide the full audit trail necessary.

Business Action Messages

Business action messages, sometimes referred to as business documents, are what is used to convey the actual business information content between the business parties.

A business action message is a legal instrument that represents a cognitive definition of a real-world entity. The **business action message** has identity and it exists. It also has a beginning, an ending, and behavior in a life cycle. The business action message can come into existence in the business transaction or be supplied to the business transaction already composed.

As a business action message moves through the business transaction processing, the state of the message changes as defined by the business action message's life cycle.

A business action message (also referred to as business information in the metamodel), can be a structured document containing defined information entities or an unstructured document of information entities of an undefined structure (for example, pictures and binary information).

Information Entity

Within the business transaction pattern model, the information entity is the container structure used for holding the structured and unstructured (digital image) business action messages.

In a proper business transaction, it is necessary to structure and have security on the business information so that only qualified business roles may operate on it. Because of this, an information entity structure is used to contain the business message with business role-based security instructions. Because some business information is built from other business information, in a compound fashion, information entities may also include or reference other information entities through associations.

Security controls on an information entity need to be specified when information must be secured within an enterprise organization until it is accessed by an authorized partner business role. These parameters on the information entity element must be specified in a manner that ensures document integrity by maintaining a "chain-of-custody" from the sender to the intended recipient of the business information.

An information entity has three security controls:

- Confidentiality. The information entity is encrypted so that unauthorized parties cannot view the information.
- Tamper Proof. The information entity has an encrypted message digest that can be used to check whether the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.
- Authentication. There is a digital certificate associated with the document entity. This provides proof of the signer's identity.

These security controls are also applied to the business action message.

Business Partner Roles

A business partner participates in the context of one or more authorized business roles that in turn define a façade for their business service capabilities.

The business partner participates as an *AuthorizedRole* of type *EmployeeRole, OrganizationalRole,* or both *(FunctionalRole)*. The distinction is in the need to identify who is authorized to the individual in a company to perform some business action, such as only a certain manager being allowed to approve vendor payments, while other times anyone in the organizational group may be allowed to perform the action.

An *OrganizationalRole* is a role that only an organization performs in a business collaboration. An *EmployeeRole* is a role that, for business and legal reasons, only an employee can perform. Usually the details of the employee must be captured, stored, and transmitted to another partner for auditing and liability purposes when the two partner roles are not in the same organization. Both partners must agree to exchange business information using a secure transport channel.

The following security controls ensure that business action message content is protected against unauthorized disclosure or modification and that business services are protected against unauthorized access. This is a point-to-point security requirement. Note that this requirement does not protect business information after it is off the network and inside an enterprise (document security provides for outside transport security).

The following are requirements for secure transport channels:

- Authenticate sending role identity. Verify the identity of the sending role (employee or organization) that is initiating the role interaction (authenticate). For example, a driver's license or passport document with a picture is used to verify an individual's identity by comparing the individual against the picture.
- Authenticate receiving role identity. Verify the identity of the receiving role (employee or organization) that is receiving the role interaction.

- Verify content integrity. Verify the integrity of the content exchanged during the role interaction; that is, check that the content has not been altered by a third party.
- Maintain content confidentiality. Confidentiality ensures that only the intended, receiving role can read the content of the role interaction. Information exchanged during role interaction must be encrypted when it is sent and it must be decrypted when it is received. For example, seal envelopes so that only the recipient can read the content.

Business Services: Constructing a Business Dialogue

The **business service view** (BSV) captures the structure and semantics of business action messages and their exchange between enterprise components that provide business services. The BSV specifies the elements of an execution process that comprises the business transaction exchange between enterprise business services as a result of the execution of business actions. The BSV is a transformation of the BTV in context of the types of business transactions being performed and the class of partner role performing the transactions.

Predefined design patterns are described for the service interactions appropriate for each business transaction pattern. These **service interaction patterns** specify interaction sequences (protocols) between two services, including the message and information exchanges. The specific service interaction pattern used depends on the type of business transaction, type of role, security parameters, and timing parameters. Unless parameter default values are required to be overridden, the appropriate service interaction pattern is derived according to the metamodel and is instantiated in the solution specification.

A **business service** is a façade that exposes a business transaction. (The façade may also be presented by a **business service agent,** which is an entity that acts on behalf of the business controlling the business service.) Business services, and their possible agents acting as their business surrogates, need to always be in alignment with any specific business transaction needs when sending and receiving business messages. For example, if the business transaction needs a quality of service to be performed within certain time constraints and with a degree of security, the service interaction pattern must in turn be faithful to meeting those requirements.

There is a standard set of service interaction patterns that support the six business transaction patterns:

- Service-Service. Messages flow between two business services.
- Agent-Service-Service. Messages flow between a business service agent and its business service, and then between two business services.
- Service-Service-Agent. Messages flow between two business services, and then a business service and its agent.
- Service-Agent-Service. Messages flow between two business services through an agent.

- Agent-Service-Agent. Messages flow between two business service agents through their business service.

Each of the service interaction patterns may have as many as five variants, as follows:

- Interaction Variant Pattern A. Applies to the business transaction pattern where time to perform equals time to acknowledge acceptance and there is no responding business document.
- Interaction Variant Pattern B. Applies to the business transaction pattern where time to perform equals time to acknowledge acceptance and a responding business document.
- Interaction Variant Pattern C. Applies to the business transaction pattern where time to perform is greater than time to acknowledge acceptance.
- Interaction Variant Pattern D. Applies to the Query/Response, Request/Response, and Request/Confirm business transaction patterns.
- Interaction Variant Pattern E. Applies to the Information Distribution and Notification business transaction patterns.

Processing Business Action Messages

Service interaction patterns are stateless message exchange patterns, as messages are interchanged between two business services. There are two types of messages:

- Business action messages. These have business information.
- Service messages. These report on the processing of the business action message.

As they are processed by a business service, all business action messages must undergo a technical evaluation process to determine whether the message is well-formed. The business action message processing rationale behind the service interaction patterns is based on a document-processing framework of the following steps:

- Grammar validation. The task of verifying the syntax grammar of a business action message is valid (usually only the header of the message is checked at this step).
- Sequence validation. The task of verifying that the collaboration control information is valid with respect to the business transaction specification.
- Schema validation. The task of verifying that the message schema is valid with respect to a message guideline agreed to in advance by both partners. It is recommended that message receipt be acknowledged after this validation step to ensure that documents are "readable" as well as "accessible."

- Content validation. The task of verifying that the content of a message is valid with respect to any business rules that govern the formation of a contract. It is recommended that business acceptance be acknowledged after this validation step.
- Activity processing. The task of processing the business transaction request in the initiating business action message.

Figure 3 illustrates the processing of a message when the contract-closing (contract acceptance document) message is an acknowledgment of receipt. The acknowledgment of receipt is a business signal.

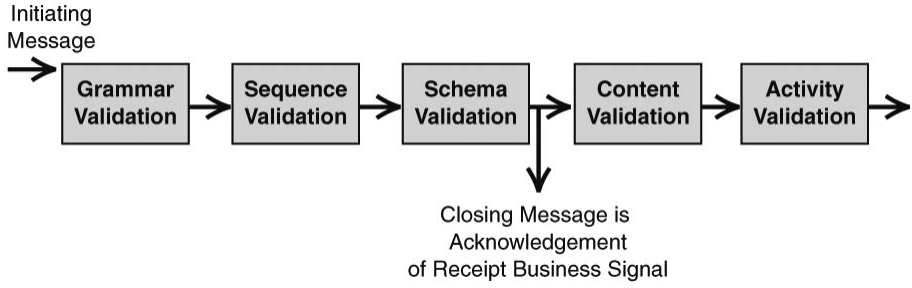


Figure 3. Acknowledgment of receipt closing message

In processing the business action messages, the rationale around timeouts is a particularly important concept to understand. This is because performance of business responsibilities within agreed time boundaries is an obligation placed on all trading partners.

The service interaction patterns need to deal with timeout exceptions whenever a requesting partner expects one or more responses to a business action message request. The idea is that a requesting partner shall not remain in an infinite wait state for the responding party.

There are four possible responses and hence four potential time-out specifications to service interaction patterns:

- Acknowledge Receipt. The time a responding role has to acknowledge receipt of a business action message.
- Non-Substantive Acknowledge Business Acceptance. The time a responding role has to non-substantively acknowledge business acceptance of a business action message.
- Substantive Acknowledge Business Acceptance. The time a responding role has to substantively acknowledge business acceptance of a business action message.
- Perform Transaction. The time a business transaction has to complete.

By convention, the value for each time-out parameter is absolute and not relative to each other. Also by convention, all message processing timers start when a requesting business action message is sent. The rules around whether service interaction patterns are well formed include:

- If the retry count is not zero and a time-out condition is signaled for any of the expected responses, the original business action message shall be resent from the initiating partner role.
- The original business action message shall be sent even if responding acknowledgments have already been received.
- If an initiating partner receives a response after a time-out condition is signaled and the original business action message has been resent, ignore this response.
- A responding partner that receives a business action message from a retry shall terminate its responding transaction for the previous business action message and the retry request shall be serviced.
- Upon sending a business action message retry, it shall be guaranteed that the sending party resends an identical business action message, except for a timestamp. Otherwise, a receiving partner shall be capable of rolling back an incoming business action message at any point in time through the acknowledgment interval, acceptance interval, and back-end processing interval.
- When the time to perform an activity equals the time to acknowledge receipt or the time to acknowledge business acceptance, the highest priority time-out exception shall be used when the originator provides a reason for revoking the original business action message offer.
- The time to perform exception is of a lower priority than both the time to acknowledge receipt and the time to acknowledge business acceptance.

Often, for technical reasons, it is not possible for one of the business partners to complete processing a business action message. With service interaction patterns, a business protocol exception also terminates the business transaction. The following are standard business protocol exceptions:

- Negative acknowledgment of receipt where the structure or schema of a message is invalid.
- Negative acknowledgment of acceptance where the business rules are violated.
- Performance exceptions where the requested business action cannot be performed.
- Sequence exceptions where the order or type of a business action message or signal message is incorrect.
- Syntax exceptions where there is invalid punctuation, vocabulary, or grammar in the business action message or signal message.
- Authorization exceptions where roles are not authorized to participate in the business transaction.
- Business process control exceptions where business action messages are not signed for non-repudiation.

A responding role that throws a business protocol exception signals the exception back to the requesting role and then terminates the business transaction. A requesting role that throws a business protocol exception terminates the transaction and then sends a notification revoking the offending business action message request. The requesting role cannot send a business signal to the responding role.

Implementation Framework: Connecting to IT

The **implementation framework view** (IFV) defines the nominal set of properties a targeted technology must be capable of in order to enable the upper views and specifications of the BMM. The BMM helps define business collaboration and information models in non-technology-specific terms. However, actually using the resulting business collaborations may be facilitated by, and may even require, automation of certain aspects. The more standardized the automation to apply can be, the more cost effective the implementation can be. The IFV does not define or specify any particular technology to bind to; instead, it simply provides for a logical loose-coupling between the technology-neutral domain of the BMM and an underlying technology infrastructure.

Modeling Using Worksheets

A business environment may be large and complex. Any basic understanding of this environment begins with information and documentation.

Worksheets use a question and answer process to facilitate the application of the BMM. The worksheets process takes a business analyst (working with business experts from the enterprise) through an incremental business process and information model construction process that provides a conceptual framework for common concepts across the enterprise.

Worksheets can be used top-down, bottom-up, or both for business analysis. The end result of completing all forms is a complete business operations model of the business requirements, defining a legally binding set of business service relationships.

As illustrated in Figure 4, when bound through the IFV to a technology stack, the worksheet process can produce a **solutions set.** A solution set is a set of specifications that can be used to configure a messaging, choreography, and health model IT system. Where the technology stack is Web services, the solution set can be in the form of XML files including Web Services Definition Language (WSDL), Business Process Execution Language (BPEL), and a health model.

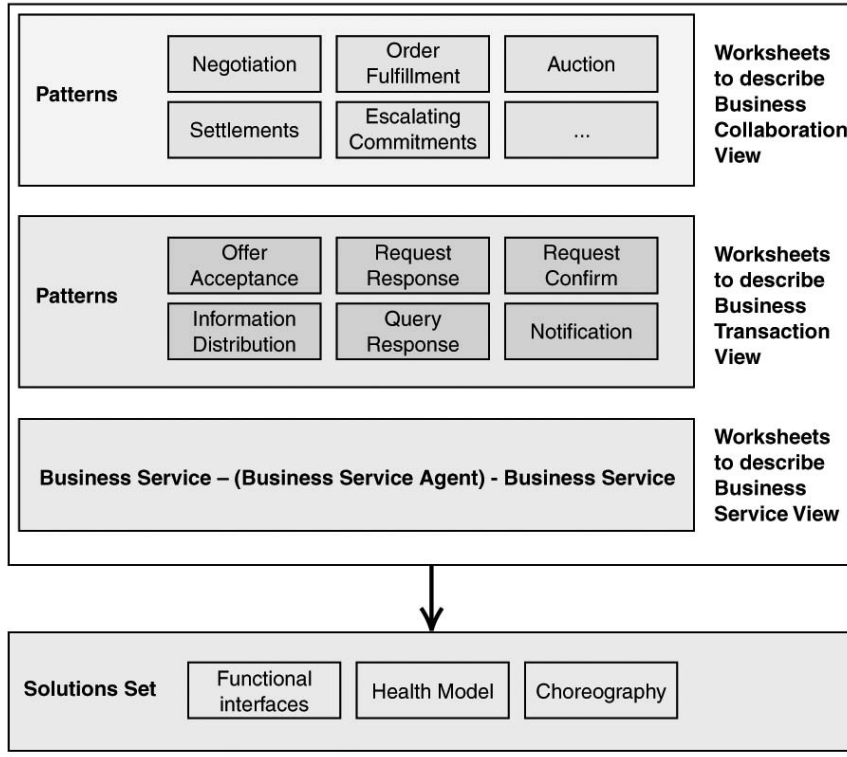


Figure 4. Modeling through worksheets to produce a solutions set

Basic Worksheet Concepts

Worksheets are a question and answer process using forms. Worksheets:

- Are a comprehensive business process and information process analysis methodology.
- Retain business acumen that is reusable over generations of new technology.
- Provide a methodology to capture business process knowledge, independent of the underlying technology.

- Help discover and define a set of reusable process and information descriptions. Patterns are used to help enforce consistent, reproducible results.
- Implement processes that help insure predictable results from a software project.
- Build a model of the business requirements into a technology implementation solutions set.

The following are basic worksheet concepts. They include descriptions of the people who participate in completing worksheets as well as key modeling concepts behind worksheets. They are:

- Industry Expert. Able to categorize and decompose a business environment into business areas, process areas, and business processes.
- Business Expert. Knowledgeable of the business area being modelled.
- Business Domain. A set of business concepts, a taxonomy, for understanding business area and process interrelationships.
- Business Collaboration. A set of activities conducted by two or more parties to achieve a commonly defined goal or outcome.
- Business Stakeholder. Someone, or an organization, who has some stake in the success of the business.
- Business Process Activity. A set of activities conducted by one or more parties for the purpose of achieving a business objective and can be conducted with time as a constraint or business rule.
- Business Collaboration Activities. Multi-partner business process activities that scope business requirements gathering and specification. A business collaboration activity is a predefined set of activities or processes initiated by one partner to accomplish an explicitly shared business goal and a business collaboration activity is finished on recognition of one of the agreed conclusions by all the involved partners.
- Business Interaction Activities. Define requirements in the business environment placed by one partner's activities on multi-partner activities.
- Business Information. The business rules on business collaboration activities (goals, requirements, and constraints). Business information is gathered from commercial trading agreements, business process agreements, and system interface agreements.

Business Collaboration Model is made up of:

- Business Process Models. Defined by business analysts which specify the choreography of business objects through business collaboration activities.
- Information Models. Formal representations of business information that can be understood and confirmed by the business environment experts.
- Business Collaboration Patterns. Selected to fit the requirements of a business collaboration activity and to facilitate business process model and information model reusability. However, in the absence of a suitable business collaboration pattern, the selection of pre-specified business transaction patterns simplifies the reuse components in a business collaboration activity.
- Business Object. A reusable class of attributes from which business information structures can be assembled for information exchange.
- Business Transactions. Atomic business processes involving two trading partners. There are six canonical business transaction patterns.
- Business Entity. A real-world thing, concept, process or event having business significance that is shared among two or more trading partners, and which exists in two or more states of at least one life cycle.
- State. Describes the status of a business entity and a business collaboration before or after a transition (for example, delivery of goods triggers the transition of order line status from "pending" to "fulfilled").
- State Transition. Describes the progress of a business entity from one state to another as expressed by pre-conditions (for example, order-line pending) and post-conditions (for example, order-line fulfilled) of a business entity triggered by an event.
- Event. Represents a business collaboration activity (for example, order and goods transfer) directed towards meeting a requirement of a business entity.
- Life Cycle. Describes all of the permitted states and transitions of the business entity during its lifetime.

Participants in the Worksheets Process

The overall worksheets process can be logically broken down by audience and their contribution towards filling in the worksheets. Table 6 presents a summary view of this breakdown.

Table 6. Participants in the worksheet modeling process

Role	Stakeholder	Business Focus	Model Activity
Between organizations	Management	Goal alignment	Business agreements and metrics
	Operations	Process alignment	Business operation and interaction processes
	Analysts, developers	Information alignment	Business documents
Within the organization	Systems integrator, network administrator	Systems Infrastructure	Partner authentication and authorization
	Application integrator	Messaging infrastructure	Component interaction sequence diagrams

There are participants for each of the three worksheet views. Each view contains a collection of worksheet forms. Depending on the view, the roles could be played by different participants. The following list provides further detail:

- BCV modeling captures the business scenarios, inputs, outputs, constraints and boundaries for business processes and their interrelationships within business process collaborations. This view is how the business domain expert sees and describes the process to be modeled. The BCV is expressed in the language and concepts of the business domain expert. A special set of forms in BCV modeling are the Service Level Agreement (SLA) worksheets, which describe the metrics and activities in the SLA. The BCV participants are:
  - *Business stakeholders:* Executive management, business owners, information modelers, process modelers.
  - *Worksheet modelers:* Business analysts, business modelers.
- BTV modeling captures the semantics of business information entities and their flow of exchange between roles as they perform business activities. This view is an elaboration on the BCV by the business analyst and is how the business analyst sees the process to be modeled. This view uses the language and concepts of the business analyst to convey requirements to the software designer and to the business domain expert. The BTV participants are:
  - *Business stakeholders:* Business analysts, systems architects, implementers.
  - *Worksheet modelers:* Information modelers, process modelers.

- BSV modeling specifies the component services and agents, and their message (information) exchange as interactions necessary to execute and validate a business process. The BSV is expressed in the language and technical concepts of the software developer. The BSV participants are:

  - *Worksheet modelers:* Derived from BTV worksheets.

Practical Approaches for Modeling Using Worksheets

Several practical considerations apply to modeling using worksheets.

Top-Down, Bottom-Up

Building a worksheets-compliant business operations model can start top-down or bottom-up. In the end analysis, when all the forms are complete, the final controlled check is a top-down modeling control.

Any use of worksheets first starts off with a clear understanding of the specific domain of business activities. Working with worksheets de-emphasizes the use of business documents and transactions to model business service interactions because that approach may have captured only one part of the required legally-binding business process model. An emphasis with worksheets is placed on the definition of business entities, the identification of their state management, and the identification of their state life cycle to produce a model that encompasses all instances and can evolve as new business requirements emerge.

Bottom-up modeling can be used as a starting point to fill in parts of the worksheets through use of existing business documents and transactions. It can help identify some model elements. However, the top-down approach must ultimately be applied to produce evolvable and maintainable models that support reuse and manage loosely coupled business processes between trading partners on the Internet.

Business Information Dependencies, Not Document Exchange

The goal of worksheets is to understand and formalize the dependencies between partner processes for a problem domain. Historically business partner communication methodologies (such as Electronic Data Interchange, or EDI) have focused on modeling the business documents being exchanged, while worksheets focus on modeling the business actions and objects that create and consume business information. Of course documents are an important part of business operations; worksheets allow for declaring business documents.

Measurability and Traceability

Worksheets top-down approach drives out the identification of measurable business objectives and requirements, which can be verified by stakeholders. Worksheets and their production rules ensure the true representation of these objectives down to their technical implementation. Traceability of these objectives is the basis for ultimate "success" or "failure" of the business operations model when in operation.

The other benefit from the top-down modeling activity is that it expresses the common semantics that will be used to describe a public business collaborative process.

Model Production Approach

Worksheets are a simple tool to collect and organize the information needed to produce a complete business requirements model. The process of gathering information for the various worksheet forms is very iterative in real world practice. As one works through the various worksheets, new information will be discovered and previous worksheet forms may need to be updated to reflect any changes.

In gathering information from business stakeholders for the forms, the modeling facilitator may learn information that is required for worksheets that would be covered at a later time. Vital information should be captured at the time it is discovered to avoid losing track of it. Worksheet facilitators should be prepared to keep track of such information on a notepad, for later transfer to the appropriate worksheets.

Simplified Procedure for Modeling Using Worksheets

This section illustrates a very simplified, step-by-step procedural overview to using the worksheets. The approach described sequentially in Tables 7, 8, and 9 proceeds in a top-down fashion with each of the three views (sets of forms) in the worksheets. Procedures within each of these groups of forms describe how to populate the worksheets. The procedures here are meant to be a typical, not authoritative or normative work procedure. Sometimes in the real business world it is best to start with what little information you have available and continue through a progressive refinement process of verifying facts, refining business needs, and coming to agreement with all parties involved in the modeling exercise.

Table 7. Business Collaboration View Forms

Steps	Worksheet Name
1. Describe each business process.	Business Process Requirements
2. Identify and describe business collaborations starting with a large collaboration and breaking it down to smaller business collaborations use cases. Each needs to be further described until business transactions are identified and described.	Business Collaboration / Business Process / Collaboration Life Cycle / Business Process Metric
3. Identify and describe any partners and their roles, specifically in relation to business collaborations, agreements, and commitments.	Business Partners and Roles
4. Identify and describe business entities	Business Entities
5. Identify and describe any business events that are used to signal the change in state of business entities and thus a business collaboration.	Business Events
6. Identify and describe the contracts as well as the agreements and commitments that are part of it.	Contracts / Commitments

Table 8. Creation of the Service Level Agreement

Steps	Worksheet Name
1. Define the Service Level Agreement (SLA) that is agreed to, the terms and conditions of the agreement, the parties involved, and the measurements to be used.	Service Level Agreements
2. Provide a description of the activities (services) that comprise this SLA.	Service Definitions
3. Identify and describe the measurements that result from this agreement.	Key Performance Indicators

Table 9. Business Transaction View Forms

Steps	Section / Worksheet Name
1. Define a business collaboration life cycle for each business collaboration to capture the dynamic requirements; that is, business transaction activities.	Business Collaboration Life Cycle
2. Provide a description of the business partners and their authorized roles in this business domain.	Business Partners and Roles
3. For each business transaction activity define a business transaction worksheet. Identify requesting information and optional responding information	Business Transaction

Steps	Worksheet Name
4. Document the key informational elements that are important to a transaction especially in achieving document element level interoperability.	Business Information
5. Define the structural aspects, constraints, relationships of the business entities as well as the various states of its life cycle.	Business Entities / Business Entity Life Cycle
6. Define the structural aspects, constraints, relationships of the information entity.	Information Entity

Product Shipping at Northern Electronics

The order fulfillment department at Northern Electronics coordinates the pickup and delivery activities around getting an ordered product to a customer. These activities include verifying and validating purchase orders (POs) with the sales department, verifying credit with the accounting department, coordinating inventory management with the production department across the various warehouses, managing the transport of products to customers, and updating delivery status for the accounts receivable department.

Within the transport function, product shipping is an area that the company has had problems with. Product shipping involves significant coordination, including ordering transport from the right consolidator, moving the right product in the right amount from the right warehouse to the right loading dock, and getting the waiting product onto the right truck. That truck must be driven by the right trucker and arrive at the right time as an agent from the right consolidator.

Product shipping is a core business process for Northern Electronics. Businesses expect problems to arise and have procedures in place to handle them. In some cases, the consequences can amount to just a marginal impact, because the exception can be managed easily or simply accommodated. In other cases, the consequences can be severe and can lead to financial penalties or even loss of business. From the perspective of the warehouse, problems such as the truck not arriving on time, a truck arriving but being the wrong truck, or the truck being too full to accommodate the order can occur. From the perspective of the customer, problems such as non-delivery of product, incorrect delivery of product (the wrong product or the wrong amount of product), or delayed delivery of product can occur.

The company has had ongoing problems with the product shipping process. Trucks don't always arrive on time. And even when they do, the requirements for the truck aren't always met. Also, the wrong cargo shows up at the loading dock more often than it should.

All of these logistical problems result in much higher overhead costs, and especially, in delay for the customers expecting on-time arrival.

Product shipping is an area that Northern Electronics has decided to improve. The company has decided to tackle two aspects of the product shipping process:

- Achieve more efficient coordination in product shipping with business partners.
- Achieve more efficient handling of business exceptions in product shipping when they occur.

An important step in the effort to improve product shipping is to model how the process works.

Who's Who in Northern Electronics

The following individuals from Northern Electronics are involved with the modeling of the product shipping business collaboration:

- Pam: She is the business program manager whose responsibility in this context is to understand and model the product shipping process.
- Zack: He is the head of IT architecture whose responsibility in this context is to work with Pam to help her understand the IT organization and its requirements.
- Mark: He is the business analyst whose responsibility in this context is to collaborate with Pam and Zack to facilitate the worksheets process.

One of the goals of the modeling exercise for Pam will be to translate the formalized information in the model into something that the IT organization can use to design, implement, and operate a solution. Particularly, Pam wants to make sure that the IT solution is set up and operated in such a way that it can help detect and correct any business exceptions that will occur when the product shipping business collaboration operates.

Modeling the Product Shipping Activities

As Pam gets to work, her first decision is that she must get a handle on understanding the product shipping process. She decides the best way to do this is to pull together a formal description of the existing processes in product pickup and delivery, so she can better determine what the problems with product shipping are and where changes should be made. In a sense, she begins an effort to formally model how product shipping is done.

Understanding Product Shipping

To understand the domain, Pam interviews a number of experts within Northern Electronics and related third parties including the individuals in the following roles:

- The *business analyst* provides business knowledge about product shipping, including what data needs to be collected and how it should be manipulated to reflect the physical reality. This includes requirements for handling and packaging the goods, requirements for third-party transport services, and shipping schedules.
- The *financial analyst* provides the financial plan regarding the shipping budget, customs charges, any other taxes, freight rates (tariffs), and insurance rates.
- The *shipping manager* supervises logistics and enforces any regulatory requirements.
- The *consolidator* is a third party who provides input regarding requirements for packaging, delivery options, scheduling, and relevant forms.
- The *attorney* provides guidance on industry standards and regulations that are best practices and legal requirements for product shipping.

It's especially important to note that not all of these individuals work at Northern Electronics. For example, the consolidator is a third-party company that Northern Electronics does a lot of business with. Getting improvements in the product shipping process is not something Northern Electronics can do alone—it involves the entire value chain.

Through Pam's interactions with these individuals and from market and regulatory information, Pam begins to develop a good model for how the product shipping process works.

An example of the broader product pickup and delivery process, namely how a product moves from the manufacturing floor to a customer's facility, that the product shipping process is a part of helps illustrate Pam's findings. Figure 5 describes the high-level business entities involved in the larger product pickup and delivery process.

Some of Northern Electronics's products are manufactured at their plant in China and then shipped to their warehouse in Everett. One such product line is the electronic controller for remote-controlled airplanes. One of Northern Electronics's customers is a wholesaler named Wingtip Toys that assembles remote-controlled airplanes in Dallas, Texas, and then sells the airplanes to retail companies. Northern Electronics hires Acme Consolidation Company, a freight consolidator based in Portland, Oregon, to arrange the transport of the electronic controllers from the warehouse in Everett to the wholesaler's warehouse in Dallas.

Acme Consolidation Company performs many tasks, including:

- Collecting the appropriate paperwork from Northern Electronics.
- Providing the freight cost.
- Reserving space for freight with freight companies and coordinating with the freight companies to deliver the cargo to the destination warehouse.
- Offering Northern Electronics insurance for the cargo while in transit.

In this example, Acme Consolidation Company hires Blue Yonder Truckers, a local trucking company to pick up and transport the goods.

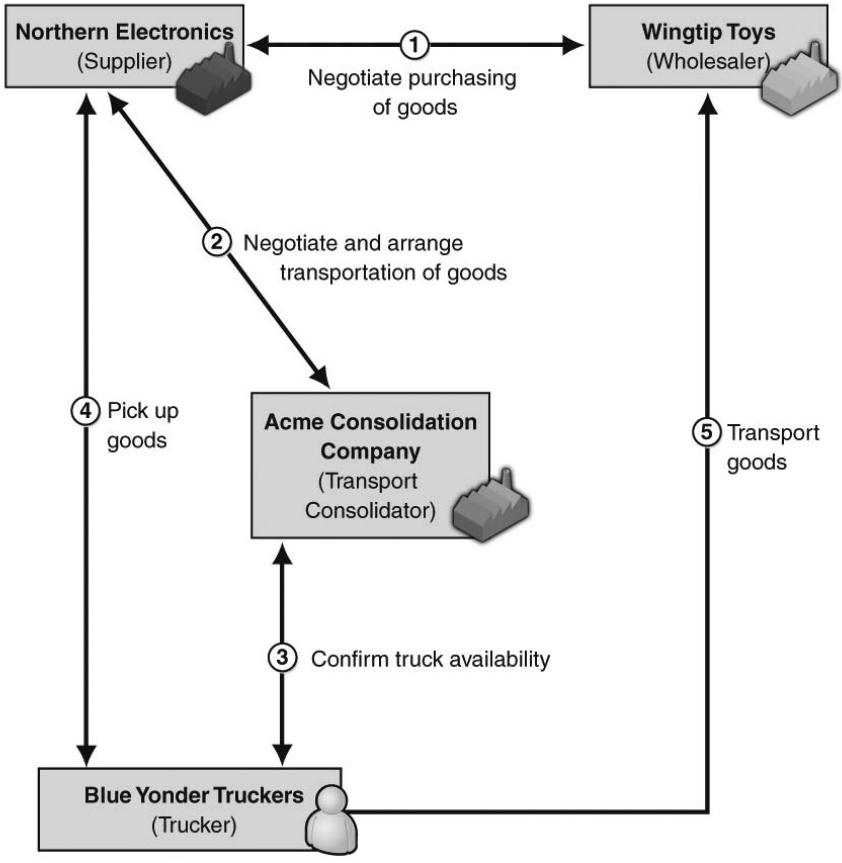


Figure 5. Entities involved in the product pickup and delivery example

Pam learns that whenever Wingtip Toys wants to place an order with Northern Electronics, the purchasing agent at Wingtip Toys calls Northern Electronics's sales agent. Pam determines that the product shipping business collaboration begins when a PO is created.

In the case of the example, Pam determines that the product shipping business collaboration works like this today:

- Richard, the shipping clerk at Northern Electronics gets new POs on his desk every morning. One of his daily tasks is to verify that the warehouse has the items in stock. If the items are not in stock, Richard notifies the customer that the PO is backordered. The PO is placed in a "Waiting for Goods to Arrive" backorder pile.
- When Richard knows that he has the goods in stock, he performs a stock allocation to cover the order. He calls Julie, the transport consolidator clerk at Acme Consolidation Company, to arrange the delivery of the goods to Wingtip Toys. Julie was on her lunch break when he called, so he left her a voice mail and put the PO into the "Pending" pile. When she gets back from lunch, she calls him back to get the details. Because Richard has a lot of piles on his desk, it takes him a few minutes to locate the PO he needs. He finally finds the PO and lets her know what the requirements are. Julie considers these, and then she agrees to transport the goods. Richard then writes up a new transport order request and puts the PO into the "In Progress" pile. He updates and gives the pickup list to David, the loading dock clerk, to get the required number of items of the product from the warehouse and arrange for the packaging of the goods, including putting the items on pallets, shrink-wrapping them, and labeling the package.
- Julie has a pool of trucking companies that she calls. She goes down the list to order a truck that can drive the shipment from Everett, Washington, to Dallas, Texas. She finally finds one that can do the job. Blue Yonder Truckers can arrive on the following Tuesday at 10:00 A.M. to load the truck. The clerk at Blue Yonder Truckers gives Julie the license number of the truck and she sends the transport order details to the trucker. Julie also faxes Richard a transport notification form with the specific details of the truck.
- Richard keeps track of the scheduling of when packages are supposed to be picked up and makes sure the goods are ready to be loaded onto the dock when the truck arrives. He has a schedule of the pickups on the white board in his office and writes down the new request.
- On the morning of the scheduled pickup, Bob, the trucker, checks in with Richard at 10:05 A.M. Richard pulls up the order and checks Bob's paperwork. He notes the time of arrival and tells Bob which loading dock he should pull his truck to.

- Bob pulls his truck up to the correct loading dock. David asks Bob for the paperwork and loads the packaged and labeled order on the truck. Bob counts the items to verify it's the same number as the number on the paperwork. David enters the final load weight into the transportation documentation and Bob signs the paperwork, assuming liability during transportation of the goods. When Bob drives away, David notes and enters the time and gives a copy of the paperwork to Richard.
- Richard gets the documents and faxes the information to both Wingtip Toys and Acme Consolidation Company to let them know that the trucker has picked up the shipment. He then updates the PO and places both the PO and transport order documents into the "In Progress" pile.
- When the truck arrives at the customer's warehouse, the customer faxes a confirmation of the arrival to Richard. He then places the PO and transport order into the "Completed" pile.

While all of the shipping is happening, Northern Electronics invoices Wingtips Toys, and Acme Consolidation Company invoices Northern Electronics.

Pam models the information flow around the product shipping business collaboration by using a flow chart, as shown in Figure 6.

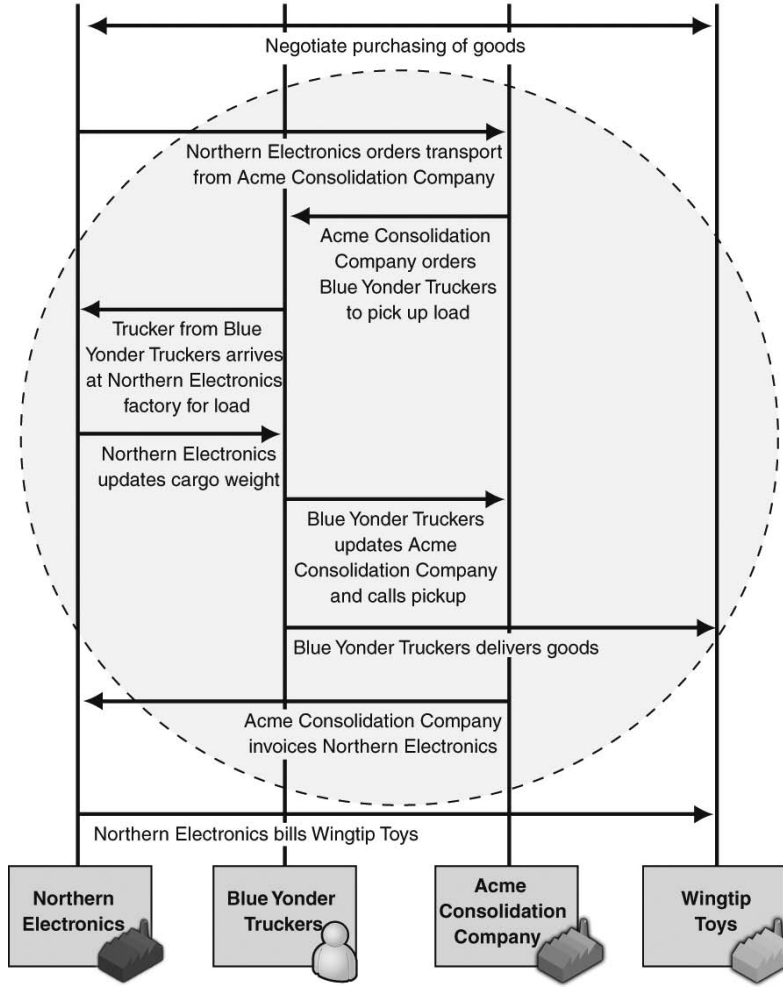


Figure 6. Product shipping business collaboration information flow

Pam can see now more specifically that many of the processes surrounding the product shipping business collaboration are dependent on human interaction and manual processes. There is information that is transmitted over the phone, paperwork that is manually filled out, faxes that are sent and received, and paperwork that is manually handed from one person to another. Pam wonders how often a message is not received or a piece of paperwork gets lost. This must certainly waste the employees' time, waste the company's money, and cause a lot of frustration for everyone involved. Moreover, how well can this process scale when the company starts to grow, or as it tries to be more competitive? How many more employees are going to need to be hired and trained to process paperwork and answer phone calls? Ultimately, the concern is: What is the impact to the customer and customer service?

Pam recognizes that some aspects of product shipping will still need to be manual but other parts of the process can certainly be automated. Pam works with Zack to identify the areas that can be automated.

Pam and Zack decide that they need to identify discrete business services that make up the product shipping business collaboration.

Automating business services in the product shipping business collaboration, where possible, should provide the means for better communication, coordination, and collaboration between Northern Electronics and its business partners. Automating business services should help Northern Electronics to:

- Manage for business performance.
- Stipulate authorized business rules.
- Obtain real-time business intelligence through key performance indicators.
- Ensure the services are designed for reusability and scalability.

One significant source of improvement from automated business services should be the ability of Northern Electronics to be in better synchronization with its business partners involved in the product shipping business collaboration. As noted earlier, one major problem area in product shipping is dealing with exceptions in a timely and effective way. The challenge in exception handling is mostly about getting useful information about an exception that has occurred to the right person in time, or even ahead of time, so that he or she can do something about the problem as it happens—or before it happens. When the right information arrives to the right person in time, there's a better chance that whatever intervention the person can implement can have a more significant mitigating effect.

Pam interviews Richard and asks him, "What are the key things in product shipping that you worry about?" He lets her know there are key events that must happen for the product shipping process to flow smoothly. The most important are:

- The truck must arrive on the correct day within the specified period of time. The pickup time is Monday through Friday from 10:00 A.M. to 10:30 A.M. If the trucker does not arrive within the designated time, the trucker will need to pick up the goods on the next business day.
- The truck must be the *right* truck. This means that the truck must have the correct license plate number as agreed in the paperwork, must be the right size, must be insured, must have the right trucking permits, and must be in good working condition.
- The truck driver must be the *right* trucker. The truck driver must have the correct paperwork containing the shipping reference number allocated by Northern Electronics.
- The loading dock must be ready with the goods when the truck arrives.
- The truck must depart with the shipment loaded within the specified period of time.

In the event that one (or more) of these conditions is not met, the process will be delayed. In the worst case, one delay can have a snowball effect and delay other shipments as well. This will severely and negatively affect the business.

Pam talks to Zack to find out whether some of these key events can be managed with IT infrastructure. Pam reasons that if Richard can get an advanced warning before one or more of these conditions are not met, it can help them plan for the situation before it becomes a problem.

Formalizing the Product Shipping Business Collaboration

Pam's investigations into how product shipping works at Northern Electronics have made her the expert. She has put a lot of organization into arranging the information she has collected, much of which came from many different sources. But still, only she really has the full picture of what is happening. The challenge is that many other people within Northern Electronics (and Acme Consolidation Company) need to know some or all of the details about how the process works and where she wants to improve it. It is important for Pam to formally capture the information in such a way that she can put context onto how the information is interrelated. To do this, Pam uses a business process modeling approach.

After Pam has looked at the entities as a whole, Pam wants to better understand the business collaboration by identifying the following actors in the scenario and their responsibilities:

- The *supplier shipping clerk* receives new POs and verifies that the warehouse has the items in stock. The clerk is responsible for ordering the necessary transportation for delivering goods to the wholesaler. The clerk is responsible for verifying that the details in the transport notification satisfy the conditions in the requests or take corrective actions if necessary. The clerk arranges for the packaging of the product and is responsible for preparing the loading dock with the goods to be shipped. The clerk uses a pickup scheduling application to determine the PO that has an associated transport notification. The goods are then queued in first-in, first-out order at the loading dock.
- The *transport consolidator clerk* is responsible for processing the transport requests coming from the supplier. The clerk will research with the contracted trucking companies to find available transport for their supplier client. After the transport provider is identified, the clerk fills up a transport notification form and sends it to the supplier.
- The *trucker* is responsible for picking up the goods from the supplier. On the day of pickup, the trucker arrives at the designated time and checks in. When the loading is completed, the trucker signs a pickup completion form to transfer and assume liability of the goods in transit.
- The *supplier loading dock clerk* prepares the final transportation document that contains the pickup details. Upon completion of loading, the clerk sends the document to the transport consolidator to confirm the goods pickup.

Pam uses an activity diagram to document the business activities in the product shipping business collaboration, as shown in Figure 7.

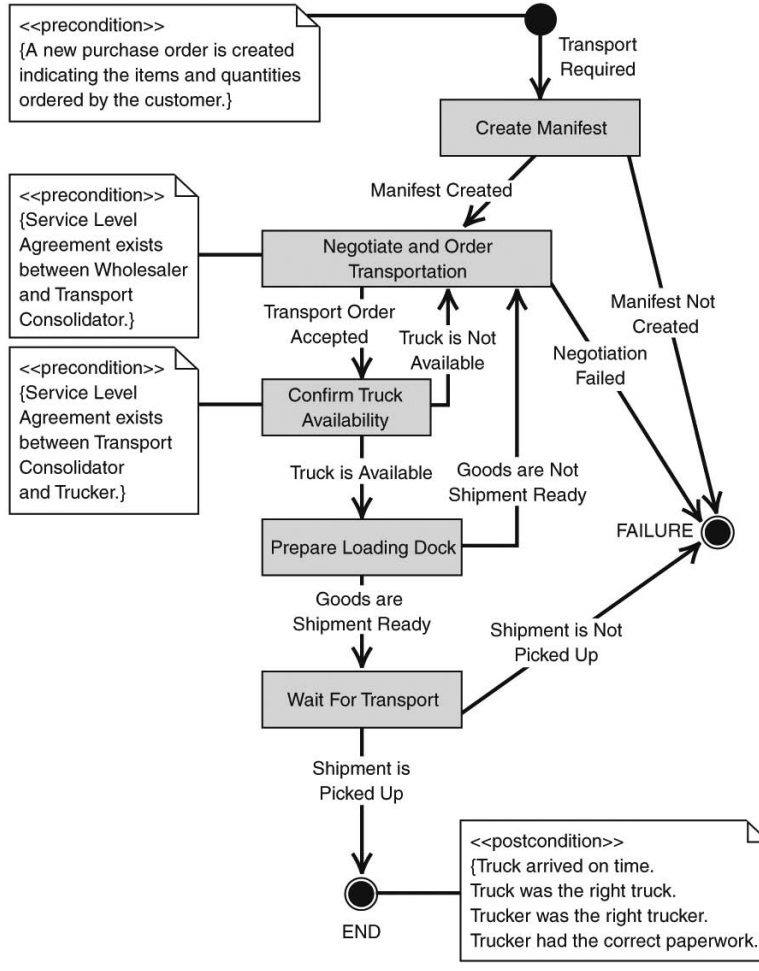


Figure 7. Product shipping business collaboration activity diagram

Some of the business activities are internal activities only relevant to Northern Electronics. However, other business activities rely on a successful collaboration between Northern Electronics and Acme Consolidation Company.

Business Operations Requirements of Product Shipping

Now that Pam has a firm understanding of how product shipping works in the business environment, Pam needs to translate the information into something that the IT organization can use to design, implement, and operate a solution. Particularly, Pam wants to make sure that the IT solution is set up and operated in such a way that it can help detect and correct any business exceptions that will occur when the product shipping business collaboration operates.

Pam uses the information she has collected and works with Mark, the business analyst, to go through the worksheets process.

Mark uses Microsoft Office InfoPath 2003 to conduct the worksheets process. He has a set the worksheets from the BMM implemented as a collection of forms. Figure 8 shows what the worksheets tool looks like.

```
<a href="Chapter03_F08-large.gif" target="_top"><img src="Chapter03_F08_
thumb.gif" alt="Click here for larger image." border="0"/></a>
```

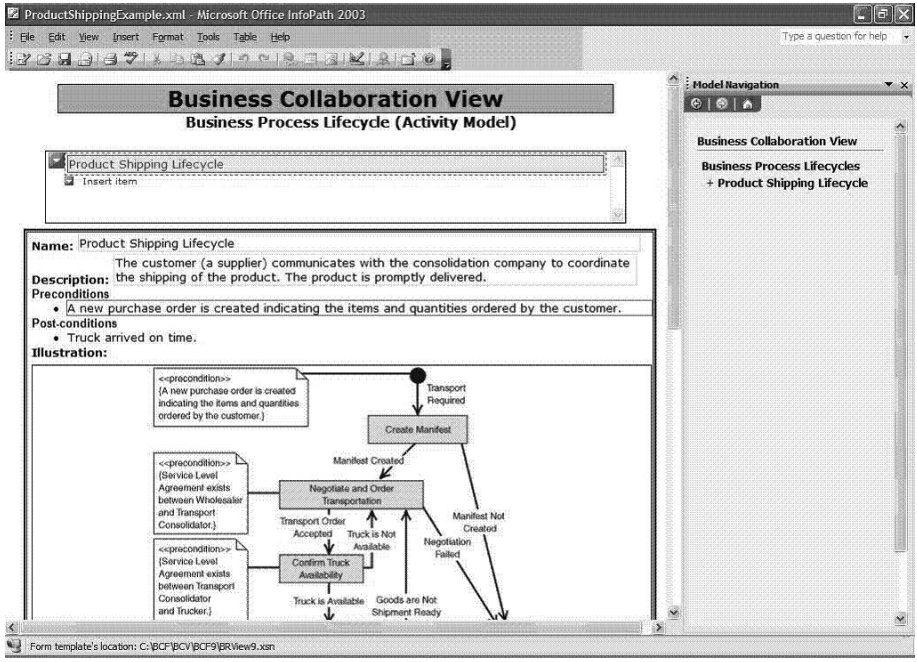


Figure 8. Worksheets process based on the BMM implemented in Microsoft Office InfoPath 2003

Modeling Product Shipping Using Worksheets

With Mark's help, Pam starts using the worksheets to describe the product shipping business collaboration. Table 10 describes what is going on at the virtual and physical levels in the flow identified in Figure 5. It also states the event that identifies when the legal responsibility is transferred from one entity to another.

Table 10. Product Shipping Service Events

Order	Virtual (Technology) Level	Physical Level	Event
1.	Supplier calls the service with the transport order details. Consolidator calls the service to verify details and enters additional information.	Consolidator dispatches the trucker to the supplier.	Transport arrangement between the consolidator and the supplier.
2.	Consolidator and supplier agree on protocol. Service transaction is executed.	The trucker arrives at the supplier's warehouse at the designated time. A licensed driver picks up the cargo.	Supplier accepts the truck (verifies the correct truck, condition, and size). Supplier accepts the trucker (verifies the correct trucker).
3.		Supplier loads the container with the packaged goods, does a final check of the cargo, and produces the final paperwork.	The truck is loaded. At this point, the supplier is still responsible for the goods.
4.	Responding business transaction service details.	Trucker signs paperwork and consolidator accepts responsibility.	The truck leaves the supplier's premises.

The worksheets help Pam determine which business transaction pattern applies to the business activities. In the product shipping scenario, there is a response required, the responder does not already have the information, pre-editor contact validation is required before processing, and there is a residual obligation between roles to fulfill the terms of the contract. This means that the product shipping business collaboration uses the offer & acceptance pattern. The documents that formalize the contract are a pickup instruction document and a pickup confirmation document response.

There are other important details about the product shipping business collaboration. Northern Electronics and Acme Consolidation Company agree to the terms of a service contract. The service contract encompasses the service level agreement (SLA). It includes both business and technical conditions and defines parameters including:

- Response time, time-out, and availability
- Development, test, and production environments
- Priorities and keywords
- Legal enforceability requirements

The worksheets ask Pam to specify the parameters in Tables 11 and 12. Table 11 refers to the parameters that Acme Consolidation Company needs to provide to Northern Electronics, while Table 12 refers to the parameters that Northern Electronics needs to provide to Acme Consolidation Company. To determine these parameters, Pam relies on the SLAs and her understanding of the business collaboration.

Table 11. Service Consumer Parameters

Business Transaction Pattern	Time to Ack Receipt	Time to Ack Accept	Time to Perform	Auth Req	Non-repudiation of Origin and Content	Non-repudiation of Receipt	Recurrence of Transmission
Offer & Acceptance	30 minutes	2 hours	48 hours	True	True	True	3

The transport agreement is that Acme Consolidation Company will call the product shipping service no less than 24 hours in advance of the truck arriving. They collectively agree that the loading of the truck should take no more than 24 hours. Acme Consolidation Company will receive an event notifying that the truck departed (with the goods) no later than 48 hours after the original call to the product shipping service.

Table 12. Service Provider Parameters

Business Transaction Pattern	Time to Ack Receipt	Time to Ack Accept	Time to Perform	Auth Req	Non-repudiation of Origin and Content
Offer & Acceptance	30 minutes	2 hours	48 hours	True	True

After Pam enters all the information about the product shipping business collaboration into the worksheets, the tool is able to provide output of a set of specifications that formally specify a solution set for the product shipping solution.

Pam is able to use the specifications to communicate her requirements to Zack and the rest of his team in the IT organization in a way that relates to the kind of work they do.

Particularly, the business operations specification that specifies management requirements helps Zack know which management functions need to be set up to support the business operations requirements as he and his team design the system. The system infrastructure will need to be set up and configured so that appropriate business activities can be managed or logged. The business operations specifications present requirements to IT in a way that can be used to create a business operations health model. A **business operations health model** is a specification of the detection, verification, diagnostic, resolution, and re-verification actions for any manageable condition in a business collaboration. Of course, not all business exceptions are predictable, and not all manageable conditions are business exceptions. But business exceptions already known to impact the business

collaboration negatively are contained in the information Pam has collected. The information also contains requirements for other manageable conditions, such as logging certain activities that occur to produce an evidentiary record. The business operations specifications pull these manageable conditions out of the formal business collaboration model and present them in a way that Zack and the rest of the IT organization can work with.

An example of such a manageable condition from the product shipping business collaboration that Northern Electronics wants to pay close attention to has to do with the on-time arrival of the trucks. The business operations specifications call out this requirement formally:

- **Business Operations Requirement (BOR):** The truck shall arrive to pick up the goods at the supplier's warehouse on or before the specified **SuggestedPickupTime.**

Tables 13 and 14 describe the details around this business operations requirement.

Table 13. Business Operations Health Model (Detection Information)

Problem	Health State and Operational Condition	Indicators	Operational Alerts	Criticality
Violation of BOR	Degraded; Integrity Problem	Event (ID = EVENT_ TRUCK_ARRIVAL_ DELAYED) This event is logged to the event log.	An e-mail notification will be sent to the Business Operations Managers notification group.	An event processing rule generates an alert with the severity set to Severe.

Table 14. Business Operations Health Model (Verification, Diagnostic, Resolution, and Re-verification Information)

Problem	Verification Procedure	Diagnostic Information	Resolution and Final Desired State	Re-verification
Violation of BOR	Manual check that truck has not arrived (should return TRUE). Operational condition confirmed as integrity problem.	Manual check. Call to transport consolidator to find out where truck is.	Manual resolution. Supplier renegotiates truck arrival time with consolidation company and updates the time in the system. Supplier updates. Suggested PickupTime. The logs and databases are both updated with new information.	Manual check that truck has not arrived (should return FALSE).

Table 13 shows how to detect the condition (in this case, a business exception) and Table 14 shows how to proceed after the detection has been made. The final resolution in this case from the IT domain is just to provide feedback to the business operations managers. No control action by the IT management system is needed to change the state of the system.

Conclusion

This document examined how business operations requirements can be communicated to the IT organization by applying business operations modeling.

The document described how the business program manager and a business analyst can apply a business operations modeling methodology to derive a set of requirements the IT department can use to build a solution. Specifically, this document discussed:

- Business operations concepts.
- Business operations modeling methodologies, including a background on such modeling approaches as well as a brief discussion of a modeling approach used to illustrate this document.
- Modeling through worksheets, including a discussion of worksheets concepts and basic procedures such as the following:
- Production of a solution set that contains business operations requirements information for use by the IT organization, with a specific focus on the inputs to the service management and health modeling functions.
- Demonstration of the approach using the Northern Electronics scenario.

Architectural Issues in Managing Web Services in Connected Systems

Architecture Chronicles
Dynamic Modeling: Aligning Business and IT

Max Morris and Frederick Chong with Jim Clark and Dave Welsh
September 2005

Applies to:

- Enterprise Architecture
- Solution Architecture
- Service Oriented Architecture (SOA)
- Service Oriented Management (SOM)
- Application Integration
- Business Process
- Business Operations Modeling

Summary

The *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* seek to present to business, solution, and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. The authors propose and develop a framework that seeks to address how to coordinate people, process, and technology issues throughout the service life cycle, presenting a simplified view into Web service goals and examining issues of Web service life cycle management, Web service health management, Web service deployment, and Web service identity and access management.

Contents

This Document

This document is part of the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT*. This volume seeks to present to business, solution, and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. The information map to the series provides an up-to-date description and cross-index of the information available and can be found at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Abstract

This document examines architectural issues in managing Web services in connected systems. The broader contexts of the Dynamic Systems Initiative and connected systems are considered. The business goals of a Web service and how it should be managed to achieve them are considered from resource and business collaboration perspectives. A Web service management framework is proposed that serves to address topics such as the Web service life cycle, Web service health management, Web service deployment management, and Web service identity and access management.

Acknowledgments

Many thanks to Nelly Delgado for her help with technical writing, Claudette Siroky for her graphics skills, and to Tina Burden McGrayne for her copy editing.

The authors would also like to thank Michel Burger, Bala Balabaskaran, and their colleagues on the Connected Services Framework for their contribution to the discussion on resource modeling.

Introduction

Like most important IT initiatives, managing Web services requires the balanced coordination of people, process, and technology.

We begin this document with the broad context of discussing what distinguishes a connected system and identifying what role Web services play in connected systems. We then offer a brief introduction to the Dynamic Systems Initiative to give further context to the complex tasks in managing distributed applications such as connected systems.

Next, consider how to determine what the goals of a Web service actually are to help pin down more formally why and how a Web service should be managed. We consider several perspectives as illustrations, including the resource perspective and the business collaboration perspective.

The model-based development and management approaches suggest that a framework for Web service management addressing management states, actions, and rules can be overlaid onto existing IT service management systems.

We propose and develop such a framework that seeks to address how to coordinate people, process, and technology issues throughout the service life cycle. After taking a simplified view of one of the perspectives of a Web service's goals, we examine issues of Web service life cycle management, Web service health management, Web service deployment, and Web service identity and access management.

What's a Connected System?

In the most general sense, a distributed system is a set of related software and hardware resources working together to accomplish a common function.

There are many different types of distributed systems. Computer networks, clustered databases, massively parallel computers, and the World Wide Web are all examples of distributed systems. One particular type of distributed system is a distributed application, where the components collaborate on a set of related computational tasks in a transparent and coherent way.

A connected system is an emergent type of distributed application. This type of application is increasingly common nowadays as business owners and developers seek to connect people, processes, and information into effective value chains. Connected systems are characterized by how they combine several recurring logical capabilities in a similar and coherent way. These capabilities include rich user-computer interaction, service-oriented messaging using Web services, process and workflow choreography, identity and access control, shared data, and federation.

Each component in a connected system does its part to collaborate on the set of tasks in the distributed application. One or more components might provide rich user interfaces. Others might help coordinate workflows. One component might act as an identity service provider. There is no required functionality for a connected system and no rule about how its components must come together. Rather, a connected system is simply a distributed application that combines a common set of logical capabilities to achieve whatever functionality it is organized to deliver.

This dynamic and organic nature of a connected system is powerful. A connected system can evolve as business needs change; for example, a system may easily expand its role from serving only a small department to serving many organizations at the same time. Yet a connected system that connects just a few machines together has the same architectural hallmarks as one that is much larger in scope.

One particularly noteworthy hallmark of a connected system is that regardless of its specific role, each component is designed using a service-oriented approach and communicates with the other components using Web services.

Don Box provides an insightful perspective on why this service-oriented approach makes such a difference (Don Box, A Guide to Developing and Running Connected Systems with Indigo (http://msdn.microsoft.com/msdnmag/issues/04/01/indigo/default.aspx).

Service-oriented development focuses on systems that are built from a set of autonomous services. This difference has a profound impact on the assumptions one makes about the development experience. A set of deployed services is a system. Individual services are built to last—the availability and stability of a given service is critical. The aggregate system of services is built to allow for change—the system must adapt to the presence of new services that appear a long time after the original services and clients have been deployed, and these must not break functionality.

Mike Burner has also commented on the service orientation aspect of connected systems and how it relates to their emergence as a particularly useful type of distributed application for enterprises. (Mike Burner, Service Orientation and Its Role in Your Connected Systems Strategy (“srorientwp”).

Management Challenges with Connected Systems

The dynamic and organic nature of a connected system presents a conceptual challenge when envisioning how to manage such a system because it makes it difficult to pin down answers to a whole range of questions. Some simple questions that are hard to answer include how to identify what the connected system in question actually is, how it should

perform, and who it should be used by. Moreover, because these are questions of management, many more-complicated questions about policy, governance, and life cycle come into play. These include who in an organization is allowed to decide what a connected system is, who in an organization is allowed to decide how a connected system should behave and perform, and who in an organization is responsible for understanding and managing the complexity of a connected system throughout its life cycle.

The insight into the difference between organic and dynamic connected systems and the more static and precise Web services that give rise to them not only has a profound impact on the development experience, but also on how to address these management questions.

The answer lies in employing separate perspectives on how to manage Web services and the connected systems that use them. All of the management questions asked about a connected system can also be asked about the Web services that give rise to it. The difference is that while these questions are hard to answer for a connected system, they are more readily addressed for a Web service, whose nature is more static and precise.

While this approach of employing separate perspectives may help by factoring the problem space, it brings up other challenges. Especially, having separate perspectives should not mean having separate and expensive management systems to handle each. What's called for is a holistic architecture that can support these separate management perspectives while harnessing one common base of people, process, and technology.

Today's IT Service Management Systems

In today's IT service management systems, the complexity of technology is not the only problem that organizations have to overcome. The quality of service management processes also affects their ability to perform. In many cases, the problems can be distilled down to visibility and control issues in the technology and process and the competency of the human users. Effective IT service management is the blending of people, process, and technology.

Maturity in the organization's service process and technology are keys to realizing this goal. The current industry landscape is decorated with a variety of service management process best practices and specifications.

The IT Infrastructure Library (ITIL) is a widely accepted approach to IT service management. (IT Infrastructure Library and ITIL are registered trademark of OGC, the Office of Government Commerce of the United Kingdom. For more information about ITIL, see http://www.itil.co.uk or http://www.itil-itsm-world.com for more information about ITIL.) The British Standards Institution has codified the best-practice processes promoted in ITIL in the BS15000 standard for IT service management. (British Standards Institution, http://www.bsi-global.com) The ITIL process abstraction consists of a series of tasks that are performed by a set of functional roles. A collection of rules govern how those tasks are to

be accomplished with the given input and expected output denoted as real world objects (RWO). Subsequently, the processes in ITIL's service management functions are derived from this process definition.

The Microsoft Operation Framework (MOF) is a specification that derives from ITIL and several other approaches to service management. It includes adaptations and inline references to Microsoft products and solutions as appropriate for automating a process. MOF defines a high-level cyclical process for managing change, operation, support, and optimization. Each process consists of a set of service management functions that are driven by well defined sub-processes. (For more information about the Microsoft Operations Framework, see the TechNet Web site, http://www.microsoft.com/technet/itsolutions/cits/mo/mof/default.mspx.)

The Microsoft Solution Framework (MSF) contains two models to help guide businesses realize their market visions through delivering IT solutions. One model captures the team view while the other presents a process view. The team model describes a collection of role clusters. Each role cluster defines a group of related job functions to be performed. The process model is meant to prescribe the logical sequence of steps from capturing market visions and requirements to deploying the IT solution. The intersecting relationship between the two models highlights the tasks performed by each role to the process phases. (For more information about the Microsoft Solution Framework, see the Visual Studio 2005 Team System Developer Center (http://lab.msdn.microsoft.com/teamsystem/workshop/msfagile/default.aspx.)

Model-Based Management

In today's IT management systems, much of the knowledge for configuring, diagnosing, and tuning IT systems is captured within the minds of a few technical IT personnel.

There are many drawbacks to this human-centric model:

- The knowledge is not highly transferable. When a staff member is absent or leaves the organization, the knowledge is lost and it may take a long time to replace the expertise.
- The approach is prone to human errors. Many of the management activities are driven through arcane command line input and scripts. A single typographical error can lead to undesired results.
- The system can be slow to adapt to change. There can be a lack of timely intervention or the need for manual intervention and analysis. Also, slowness in patching and updating software leads to a significant number of security and virus incidents.

An alternative approach, referred to as model-based management, attempts to address these issues. Model-based management advocates using highly-structured information models to capture the intentions and desired results of management. The approach encourages proactive, up-front thinking of the instrumentation and tasks required to achieve desired management outcomes. Moreover, the approach attempts to identify mitigation strategies to employ in case deviations occur.

Model-based management is at the heart of the Dynamic Systems Initiative (DSI). DSI is an effort to develop such a holistic architecture, and it is especially geared for management. DSI is a commitment from Microsoft and its partners to help IT teams capture and use knowledge to design more manageable systems and automate ongoing operations. DSI is about building software that enables knowledge of an IT system to be created, modified, transferred, and operated on throughout the life cycle of that system. These core principles—knowledge, models, and life cycle—are the keys to addressing the complexity and manageability challenges that IT organizations face today.

We next provide a brief introduction to DSI, but we then refer the reader to other DSI information for further guidance on management questions from the connected systems perspective. In the remainder of the document, we put DSI into context as we turn our focus to management questions from the Web services perspective.

Dynamic Systems Initiative

Note: The information presented here about DSI is excerpted from the Dynamic Systems Initiative Overview White Paper (http://www.microsoft.com/windowsserversystem/dsi/dsiwp.mspx), where more information is available.

During the mid-1990s, a team from Microsoft Research initiated a focused effort to examine the operational challenges that medium-sized and enterprise business customers face in their IT environments. Their goal was to understand the core drivers of the overwhelming complexity they were facing and architect a software solution that would help dramatically reduce the associated operating costs.

In these environments, it was clear that customers were running applications that had evolved to become increasingly distributed and service-oriented in nature. At the same time, low-cost, high-volume, industry-standard hardware, such as load balancers, switches, servers, and centralized storage, had become common building blocks for—and an integral part of—these applications. The definition of these distributed applications included much more than just the software.

The nature of these distributed systems had resulted in dramatically increased complexity, and that complexity spanned the entire application life cycle. Design of new systems required considerable time and cross-team coordination. Deployment of these new systems required new hardware acquisition and involved multiple iterations with development to optimize the system. The manual nature of operations was requiring some businesses to spend as much as 70 to 80 percent of their IT budget on maintaining their existing systems. (Accenture IT Spending survey.)

While these problems were felt more acutely by larger, enterprise customers, many of the small and medium-sized businesses were beginning to face similar challenges.

DSI Technology Principles

Note: This has been excerpted from http://www.microsoft.com/windowsserversystem/dsi/dsicore.mspx. For more detailed information about DSI, see http://www.microsoft.com/windowsserversystem/dsi/dsiwp.mspx.

From a core technology perspective, DSI is about building software that enables knowledge of an IT system to be created, modified, transferred, and operated on throughout the life cycle of that system.

Knowledge is essential to system management: knowledge of the deployed systems, knowledge of the environment in which they operate, knowledge of a designer's intent for those systems, and knowledge of IT policies.

Specifically, knowledge could include:

- Developer constraints on settings of a component, including constraints on related systems that the component is hosted on or communicates with.
- IT policy that further constrains settings or deployments.
- Installation directives that describe how a system is to be installed.
- Health models that describe system states and the events or behavioral symptoms that indicate state transitions.
- Monitoring rules, ranging from polling frequency to event filtering and forwarding to diagnostic or corrective action in response to problems.
- Schemas for instrumentation, settings, events, and actions.
- Service-level agreements that define performance and availability.
- Transaction flows and costs of processing steps for performance analysis.
- Reports.

As IT organizations have become more geographically dispersed and individual roles more specialized, they tend to operate in silos, making it increasingly difficult to communicate relevant system knowledge across the IT life cycle. Consequently, organizations find it very difficult to collaborate across roles, promote continuous improvement of a system's design and operation, and conduct typical management tasks such as deployment, updating, and patching.

The silos that form across IT organizations interact with an application or system at some point during its life cycle. However, each silo possesses its own pocket of system-relevant knowledge that does not get communicated effectively to the rest of the organization.

Capturing Knowledge in Software Models

Software models can be used to capture this system-relevant knowledge and facilitate the communication and collaboration of this knowledge, which is required to improve the efficiency of the entire IT life cycle.

Live Software Models

A software model provides a level of abstraction for administrators, similar to what a blueprint provides to an architect or a prototype provides to an automotive designer. But for a dynamic and distributed software environment, a static model or blueprint is insufficient. The model must be living—it must evolve throughout the life of a system.

When a system is developed, basic rules and configurations are defined. As the system is deployed, the details of its configuration, environmental constraints, and requirements are added. As operational best practices are developed or enhanced, they can be incorporated into the model as well, providing a feedback loop between the operations staff and the model. In the end, the model becomes a live, dynamic blueprint that captures knowledge about a complete distributed system in terms of its structure, behavior, and characteristics.

Consider the experience provided when pressing the gas pedal in an automobile. The user, in this case a driver, has a high-level, abstract view, a model, that maps to a desired behavior, such as making the automobile go faster. Behind the simple operation of pressing the gas pedal, the system—in this case the automobile control system—takes care of adjusting the fuel intake timing, exhaust valve timing, gear selection, and so on, taking into account constraints such as engine speed and air pressure. Essentially, it carries out changes on the model as instructed by the driver.

What are the benefits of capturing knowledge in these dynamic models?

- The system model captures the entire system's composition in terms of all interrelated software and hardware components.
- The system model captures knowledge as prescriptive configurations and best practices, allowing the effects of changes to the system to be tested before the changes are implemented.
- Tools that take advantage of the system model can capture and track the configuration state so that administrators do not need to maintain it in their heads. The software maintains the desired state so that humans do not need to.
- Administrators do not need to operate directly on real-world systems; instead, they can model changes before committing to them. In this way, "what if" scenarios can be tried without impact to a business.
- The system model becomes the point of coordination and consistency across administrators who have separate but interdependent responsibilities.
- The modeling system becomes the integrated platform for design and development tools that enable the authoring of system models. It also becomes the platform for operational management and policy-driven tools used for capacity planning, deployment, configuration update, inventory control, and so on.

Capabilities Enabled by Software Models

Several key capabilities of IT organizations and IT systems become possible when software models are used to capture all relevant system knowledge.

Design for Operations

When creating mission-critical software, application architects often find themselves communicating with their counterparts who specify data center architecture. In the process of delivering a solution, an application's logical design is often found to be at odds with the actual capabilities of the deployment environment. Typically, this communication breakdown results in lost productivity as developers and operations managers reconcile an application's capabilities with a data center's realities.

With model-based development tools, Microsoft will mitigate these differences by offering a logical infrastructure designer that will enable operations managers to specify their deployment environment and architects to verify that their application will work within the specified deployment constraints. These tools use software models to capture the knowledge of a designer's intent, knowledge of an operational environment, and knowledge of IT governing policies to ensure IT systems are designed with operations and manageability in mind from the start.

System-Level Management

Models can capture the entire structure of an application, including all the underlying and interrelated software and hardware resources. Management tools can use those models to provide a system-level view of the health and performance of that application, enabling administrators to understand the impact of changes or errors in the system and to manage the application more effectively.

This system-wide view will enable management tools to perform robust health monitoring and problem solving, as well as end-to-end performance and service-level management.

Policy-Driven Operations

Models can also capture policies tied to IT and corporate governance, such as Sarbanes-Oxley compliance or basic security standards and operating system versioning Management tools can use these models for desired-state management.

By comparing the model of the real-world state with the model of the compliance definition, management tools can make systems compliant before allowing them access to corporate resources.

Abstraction of Hardware Resources

Software models can capture an entire system's composition in terms of all interrelated software and hardware components. As a result, a system will contain a specific description of the hardware requirements of the environment into which it will be deployed.

This knowledge will enable resource management technologies to interpret these hardware requirements and to be used by management tools to ease the initial provisioning, ongoing change, or removal of hardware from an application based on changing business needs.

Perspectives on the Service Abstraction

This paper now turns its focus to management questions from the Web services perspective. In doing this, we must first establish a foundation for the fundamental abstraction of a service.

We start by discussing Don Box's four tenets of service orientation, as nicely summarized by Mike Burner (Don Box, op. cit.; Mike Burner, op. cit.):

- Boundaries Are Explicit. Services interact through explicit message-passing behind the boundaries. We make no assumptions on the space behind the service boundaries. Crossing service boundaries can be costly (for example, you may need to span geography, trust boundaries, or execution environments). We explicitly opt in to service invocation, by formally passing defined messages between services. The explicit boundaries allow us to formally express implementation independent interaction; we can be agnostic to choices of platform, middleware, or coding language used to implement other services.
- Services Are Autonomous. Services behave reasonably as independent entities. We make no assumptions on the space between the service boundaries. There is no presiding authority in a service-oriented environment. Services are independently deployed, versioned, and managed. The topology in which a service executes can and will evolve. The service should expect that peer services can and will fail, and that it can and will receive malformed or malicious messages. Services should be built not to fail, using techniques such as redundancy and failover.
- Services Share Schema and Contract, Not Class. Services interact solely on their expression of structures using schema and of behaviors using contract. The service's contract describes the structure of messages and ordering constraints over messages. The formality of the expression allows a computer to verify incoming messages, protecting the service's integrity. Contracts and schema must remain stable over time, so building them flexibly (for example, through use of **xsd:any** in schema) is important.

- Service Compatibility Is Based on Policy. Both service-providers and service-consumers will have policies—operational requirements—for interactions across boundaries. A simple example of provider-side policy is that a service may require that the invoker have a valid account with the service provider. From the consumer-side, an organization may require that service invocations across the Internet be encrypted, so it will only use services that offer the capability of bi-directional security-enhanced data exchanges. Services express their capabilities and requirements in terms of a machine-readable policy expression. Policy assertions are identified by a stable, globally unique name. Individual policy assertions are opaque to the system at large; services must simply be able to satisfy each other's policy requirements.

These tenets are very helpful in understanding how a Web service should be designed and function to handle technical interactions within a service-oriented environment, especially with other software components within a connected system. But we are left still to consider how a Web service should be designed for business and IT operations and how it should be managed. For example, we recognize that there is an equivalent set of business tenets, grounded in law and economics, that constrain and influence the design of well-formed Web service business functions.

One simple perspective to take on the business design is to ask what the goals of the Web service are. After the goals are identified in the form of a model, those goals can be used throughout the Web service life cycle to ensure the Web service is well-designed, implemented, tested, and deployed for operations. With enough foresight in design, the model can be enhanced with the mitigation activities that should be taken when the goals are not being met. Then, the model can be used to help set up the IT management system to measure performance of the Web service and to take action as necessary.

The challenge is then to identify what the goals are. Several different approaches can be used to meet this challenge. A Web service might describe for its consumers (such as other components within a connected system) how it has modeled its goals by expressing them coherently by way of its policies. This would enable the invoking application to know what it should do to comply with the invoked Web service's management expectations. Further elaboration on this topic is beyond the scope of the current paper. We discuss a couple of these approaches here and have explored one in detail elsewhere in this series. (For more detailed information about communicating business operations requirements to IT using business operations modeling, see the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center, http://microsoft.com/architecture. Regardless of how the business goals of a Web service are determined, we believe effective Web service management begins with knowing up-front what these goals are, collecting them into a model, and using that model throughout the Web service life cycle to align IT with the business plan.

Resource Perspective

One approach to take in considering the overall goals of a Web service is one that employs the resource perspective. This approach was developed within the Service Network model by Michel Burger, Bala Balabaskaran, and their colleagues in the Microsoft Communications Sector. For more detailed information about the Service Network model, service enablement, and the Connected Services Framework, see the Microsoft Web site (http://microsoft.com/csf). This section was summarized from an unpublished work by Michel Burger, Bala Balabaskaran, et al. entitled "Service Enablement Cookbook."

The resource approach looks at a Web service from the perspective of its resources. A resource is the unit of consumption between a Web service and its consumer. The resource approach considers a comprehensive set of management models in that its scope includes provisioning and usage models alongside the traditional health model.

With the resource approach, a Web service must provision a resource for a consumer. Then, the health of the resource (inclusive of service level expectations) must be managed as operations that use the resource are performed against the Web service. Finally, the usage of the resource must be measured (or metered). The resource approach assumes that a Web service brokers one or more resources at its service boundary. The manageability requirements of a Web service that does not broker an underlying set of resources, such as a simple random number generator Web service, do not fit the scope of the resource approach.

With the resource approach, three semantic models and their operations can be used to effectively manage the operational functionality of a Web service. These are:

- Provisioning model. This model includes provisioning states, events, and related task-based operations.
- Health model. This model includes health states, events, performance counters and events, and related task-based operations
- Usage model. This model includes usage events and related task-based operations.

Identifying the core set of resources associated with a Web service helps with understanding the details of the three models that need to exist at the level of the service boundary. The underlying models that exist for each resource owned by the Web service will ultimately contribute most of the components to the service models.

A Web service in the Service Network brokers resources. The identification of the resources is an essential task.

Two easy criteria can be used to identify what a resource is:

- The resource tends to be the unit of transaction between the Web service and the Web service consumers.
- A resource has identifiable provisioning, usage, and health models.
- A few examples illustrate how resources can be identified:
- The Web service may offer clearly identifiable resources if it is set up as a commercial service. Examples include a mailbox on an e-mail service, a calendar on a calendaring service, a video channel package on an IPTV service, bandwidth on a network service, and an active switch port on a DSL service.
- The Web service may offer clearly identifiable resources if it represents tangible resources. A tangible resource includes physical resources such as a DSL router, a set-top device, or a mobile handset. A tangible resource could also be a book, a CD, a ring tone, or a user interface in use by a person.
- The service may offer less clearly defined resources if it brokers a back-end process. An example is an order management application delivering a service. The active instance of the process can be treated as a resource brokered by the service. In the example of order management, an "order" is the resource. When aspects of the life cycle of an instance of an entity (for example, an order) are affected every time a service is invoked, that entity can be treated as a resource.

Figure 1 illustrates the mailbox example from a resource perspective. The resource that is consumed externally is the mailbox, and it must be provisioned to be usable, managed to stay healthy as it is being used, and billed after it is used (if it is offered commercially). Internally to the Web service, many resources are brokered to deliver this functionality.

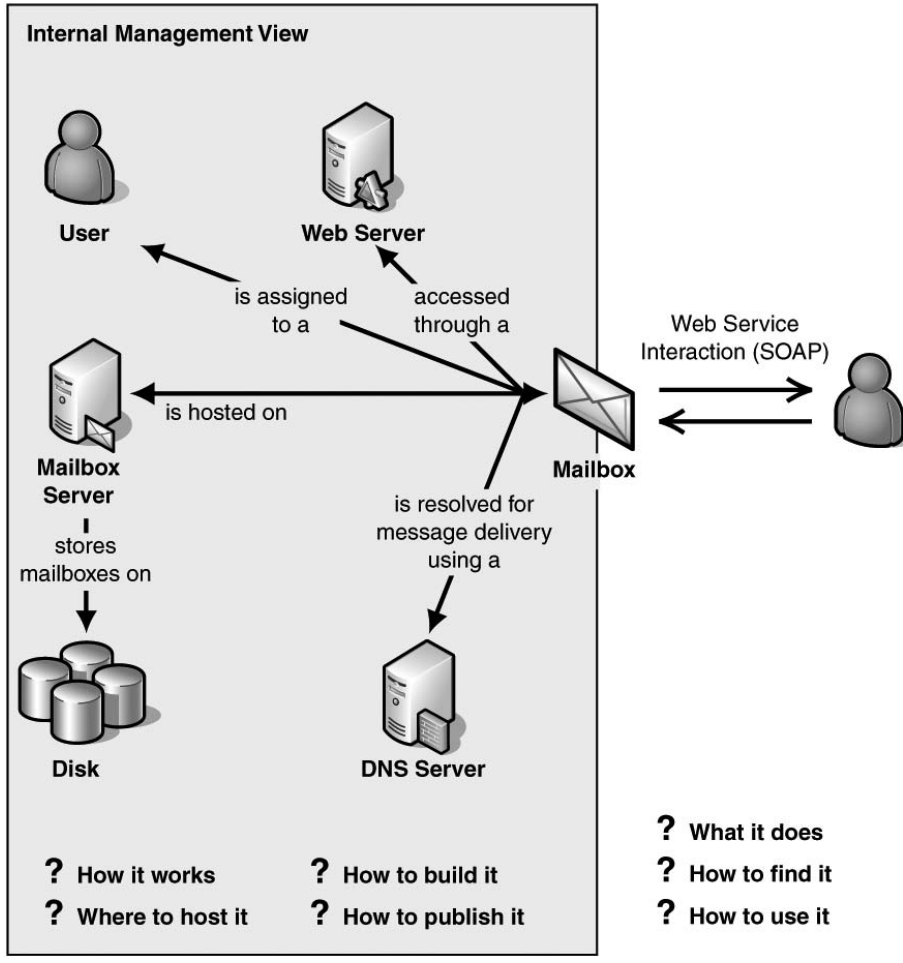


Figure 1. Illustration of external and internal resources for a mailbox Web service

From the resource perspective, the goals of a Web service are largely defined from the Web service designer's viewpoint. This is a practical and useful approach for defining the goals of a Web service. The design effort is one of bringing many internal resources together to deliver on a higher-level offering made to a Web service consumer. The definition of the offering is certainly relevant to the design of the Web service and therefore also the resource perspective, but why the offering exists and how it is defined are outside the scope of the resource perspective.

Business Collaboration Perspective

The business collaboration perspective is a more comprehensive approach than the resource perspective. This approach seeks to understand the goals of a Web service by considering business resources, business events, and business agents (people) at the business operations level. It uses formal business operations modeling to find these requirements by focusing on the business collaboration that the Web service is intended to participate in; that is, how the Web service will be used collaboratively in a business context. The approach then translates these business operations requirements into a high-fidelity format that can directly drive, as much as possible, the Web service solution design and Web service health model. For more detailed information on communicating business operations requirements to IT using business operations modeling, see the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

When two businesses collaborate with each other in a value chain, whether as supplier-customer or partner-partner, the dealings they have with one another generally follow procedures the two agree upon in advance. (For more detailed information on communicating business operations requirements to IT using business operations modeling, see the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center, http://microsoft.com/architecture.) These procedures are usually defined with enough clarity through a contract that each company can clearly know its role, can perform the activities it has committed to, and when its commitments are fulfilled, can collect whatever benefit or payment is due from the other.

It is the job of the business operations organization to put into place the infrastructure necessary to support at an operational level an organization's own side of the business collaborations it has with others. In performing its work, the business operations organization maps contractual requirements onto business capabilities as it defines, implements, and operates a collection of business processes. The business operations organization also works closely with the IT organization to support the IT organization's work in defining, developing, and operating technology solutions that realize the collection of business processes.

There are two ongoing challenges business operations organizations have as they oversee the running of the business, as illustrated in Figure 2:

- Business relationship drift. This situation occurs as the two businesses in a business collaboration experience different levels of information about the ongoing business activities they share.
- Business operations drift. This situation occurs as the business operations organization and the IT organization within a company experience different levels of information about requirements and ongoing operational performance of business processes and the technology solutions that support them.

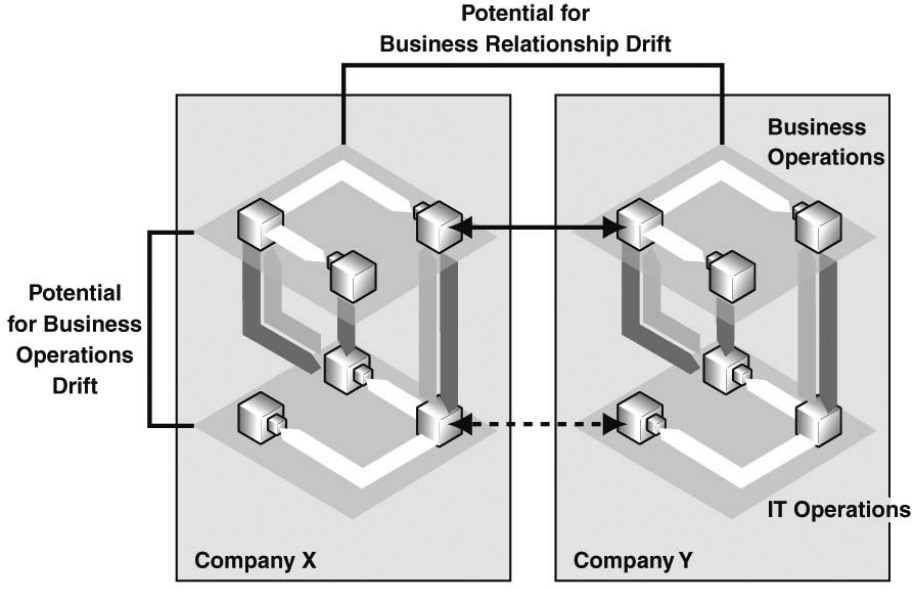


Figure 2. Illustration of business relationship drift and business operations drift

Business relationship drift in a business collaboration is inevitable. It's not possible for one business to have all the same information at the same time as the other. But business relationship drift does not become a problem, and remains hidden away, when a business collaboration proceeds as expected. The different, shared activities unfold in a timely way, continuously bringing the business relationship back into alignment, and each business continues to experience the outcomes it expects.

It's when business exceptions occur—when an expected event is delayed, or an expected event occurs out of sequence—that business relationship drift manifests itself as a problem. Typically the manifestation is a cascade: one important activity completes late, or not at all, leading to other activities not starting, and so on. It's very hard for the businesses to get back on the same page, or to get back into business relationship alignment. Figuring out what has gone wrong and where, and getting things back on track (if that is even possible) can take a lot of time, energy, and expense.

With good up-front planning and experience to draw on, the difficulty in handling business exceptions isn't in knowing what to do about them. Most issues can be anticipated, and a good plan will include some way of dealing with the completely unexpected. Instead, the difficulty in handling business exceptions is mostly about getting useful information about an exception that has occurred to the right person in time, or even ahead of time, so that he or she can do something about the problem as it happens—or before it happens. When the right information arrives to the right person in time, there's a better chance that whatever intervention the person can implement can have a more significant mitigating effect.

Getting useful information about what exceptions are occurring to the right person in time is no easy task, and it is the hallmark symptom of business operations drift. Expected performance is not achieved, but the current performance level actually remains unknown. The information to achieve alignment of business operations usually exists, but it is almost always distributed in a piecemeal fashion throughout the technology solution that implements the business processes that work together to enable the business collaboration. In such situations, non-performance of a contractual commitment isn't something that is known at the business operations level until some time later, when an after-the-fact audit is performed. Such an audit, usually directed manually, uncovers that certain requirements from the contracted service level agreement were or were not met. In failure cases, penalties are often assessed. Meanwhile, in real-time, the non-performance of the commitment often leads to cascading business exceptions in the business operations.

The key to ensuring business relationship alignment between two businesses would seem to be business operations alignment between the internal business operations and IT operations organizations of each business. If there were some way for the IT organization to know in advance what business operations requirements needed to be met so that IT operations could then let business operations know as those requirements were not met,

the business operations team would be better able to detect business exceptions that will likely worsen business relationship drift as they occur; or even before they do.

A modeling approach to formalizing business collaborations can facilitate the identification of conditions that will worsen business relationship drift. When these conditions are identified and transformed into a model that the IT organization can use, the IT solution can be designed, implemented, and operated in way that combats business operations drift. With tighter coupling between business operations and IT operations at both businesses in a business collaboration, following this approach, much more timely knowledge about the state of shared business activities can be known to both parties. This condition is referred to as business state alignment, and it is illustrated in Figure 3. With **business state alignment** achieved, business operations can run more smoothly and unnecessary expense and downtime can be avoided.

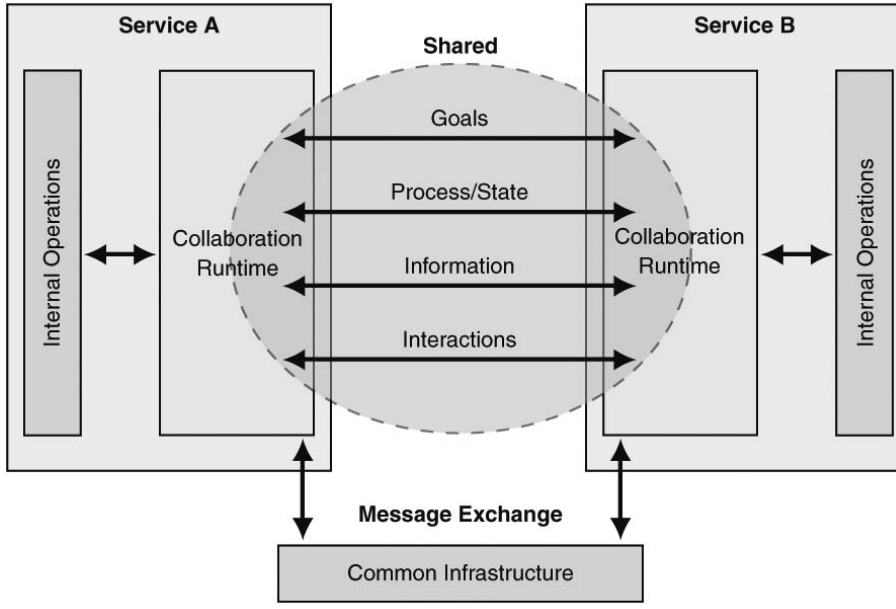


Figure 3. Illustration of business state alignment

The conceptual framework in Figure 4 links business operations modeling to IT service management and helps explain the business collaboration perspective. The conceptual framework provides a coherent way of relating business and IT management tasks. It connects business operations requirements to existing IT management processes and capabilities. The conceptual framework contains a services life cycle approach that includes the business operations-level scope.

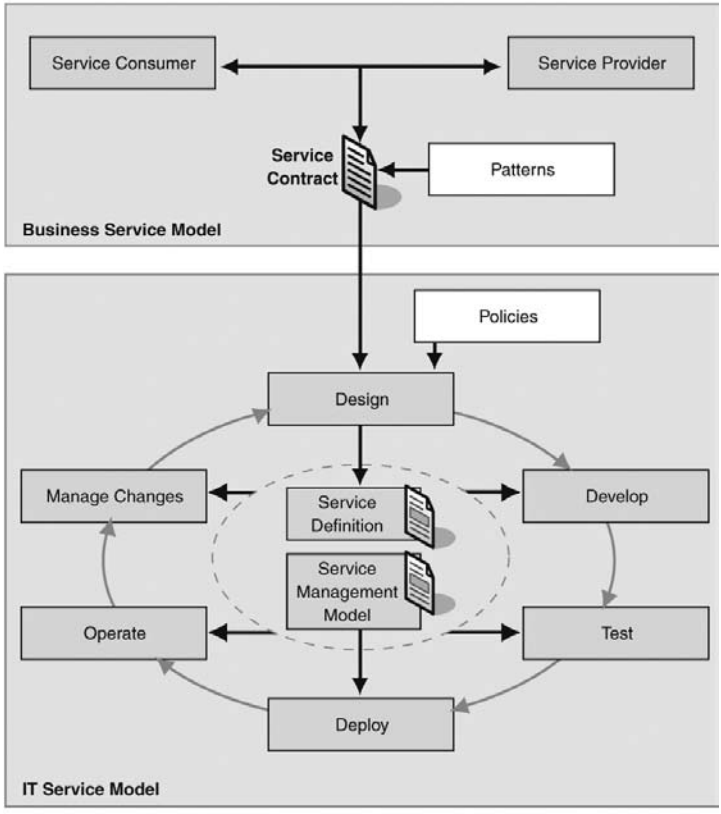


Figure 4. Conceptual framework illustrating the business collaboration perspective

Business operations modeling can be used to help identify the goals of a Web service from the business collaboration perspective. The approach can help determine why a Web service should exist, what it should do, and how it should do it. This approach is more difficult and complex than the resource approach. This approach involves working with many people in the business operations domain to develop a correct model. Then, to be useful by the IT staff that must design, implement, and operate the Web service, the requirements must be translated into a form that can drive both the Web service solution design and the Web service health modeling processes. Nonetheless, despite the effort involved in determining them, Web service goals from the business collaboration perspective are very useful for helping guide how to manage a Web service: these goals are directly relevant to the company's operations.

Demystifying Web Service Management

"Web service management" is a term often used to describe the application of new technology solutions to traditional IT service management problems such as instrumentation and monitoring applied to Web services. The term is also used to describe how new, special-purpose infrastructure can connect Web services to business functions such as usage control, usage metering and billing, and customer care systems. The scope of the term is even cast to include the application of new tools and their run times designed to change the behavior of an operating Web service, sometimes transparently to it, and therefore "manage" it.

Clearly, Web service management crosses many boundaries, and there are new investments an organization can and should make to improve its ability to manage Web services. We especially believe that's true when considering the complexities of managing the connected systems that Web services give rise to, as we argued in the "Model-based Management" section earlier in this document.

But a new term such as Web service management with its attendant buzz and urgency can create confusion over what are the right investments for an organization, and when to make them. The term can also create confusion over the role existing IT service management solutions can already play in managing Web services. For example, it may not be clear that guidance that describes how to use existing health and performance monitoring infrastructure can almost directly apply to detecting problems in operational Web services.

Many businesses embarking on Web services may be new to the service paradigm. The paradigm shift requires not only adjustments to development methodologies (for instance, a focus on the four tenets), but also to the IT governance mentality. The task of managing Web services is not likely found in the panacea of a new technology solution; instead, it is likely to be found today in the coherent dedication to a set of coordinated activities throughout the service life cycle. Over time, we argue that a model-based management infrastructure will align with these activities, helping to automate them as the knowledge in the organization is formalized and more widely communicated and applied.

The capacity to achieve the shift to the service paradigm lies in a business's ability to see past the buzz and urgency around short-term approaches to Web service management into the business's motivations for managing Web services. When the motivation is clear, the extent to which a business can execute the service paradigm shift hinges on its ability to address the challenge today of coordinating and rationalizing the people, process, and technology issues throughout the service life cycle.

Web Service Management Framework

In the sections that follow, we describe a framework to help approach the challenge today of managing Web services using existing IT service management capabilities by coordinating and rationalizing people, process, and technology issues throughout the service life cycle.

We first consider the business collaboration perspective to help define Web service goals. We cast business operations requirements into a much-simplified service model, and then describe in a more granular fashion the logical entities to map onto management requirements in a Web service implementation.

Next, we describe the Web service management issues through the decomposed entities.

Finally, we address these requirements and issues by presenting a technology framework grounded in the service life cycle to help harness existing people, process, and technology to manage Web services.

Web Service Goals and Service Model Logical Entities

Figure 5 illustrates the logical entities of a much-simplified service model derived from the business collaboration perspective presented earlier and illustrated in Figure 4.

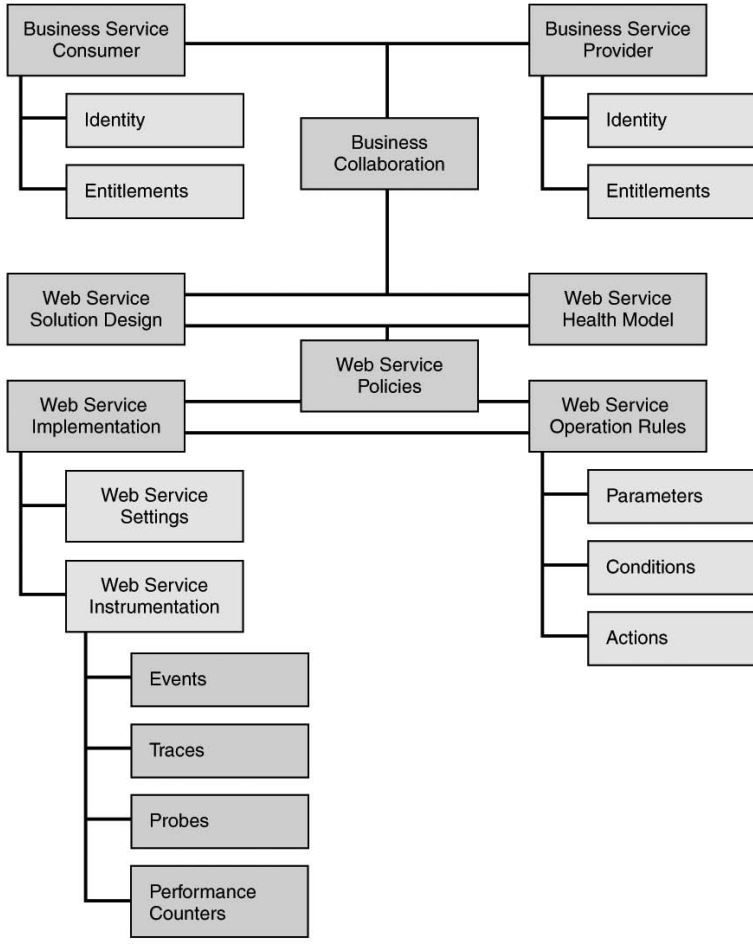


Figure 5. Service model logical entities

The logical entities are related in various ways:

- The top of the figure shows entities representing the simplified service model. These entities are the business service consumer, the business service provider, and the business collaboration engaged in by both parties.
- The business operations requirements in the business collaboration can be categorized in two purposeful ways. Service definitions describe the service that the parties agree to provide and consume within the business collaboration. Service commitments describe the manner in which the services are to be delivered and consumed within the business collaboration.
- Service definitions contain information such as the service access points, service policies, and the message schema definition. Because a particular defined service may be offered to one or more business service consumers, this part of the business collaboration is considered to be business service consumer independent.
- Service commitments are templates of the promised terms and conditions binding the business service provider to business service consumers for services within the business collaboration. Specific business service consumers will have run-time bindings for the templates that must be honored by the business service provider. For example, a service commitment template for a certain service from a business service provider might specify availability. For business service consumer A the run-time binding might be "available Monday through Friday, from 9 A.M. to 5 P.M." For business service consumer B, the run-time binding might be "available everyday, 24 hours a day."
- The service definitions are realized through deployed instances of service implementations. Each of the service instances has a set of service policies that govern the type of information that must be present in the service requests and service responses. The service policies may also specify if the information is mandatory or optional. For example, the service policy can state that every service request message header must include a signed time stamp. Request messages that do not meet the policy requirements are to be rejected by the service.
- Every deployed service instance also has a set of configuration settings. The configuration settings can enable services to be more flexible in adjusting to the deployment environment. For instance, the service uses the configuration data to help resolve the identity authority that it trusts for authenticating users.
- Management instrumentation is used to provide visibility and control over running instances of Web services. Service instrumentation may take the following forms: debugging traces are used to help with diagnosing code execution behaviors; events are used to raise alerts that signify meeting certain management conditions; performance counters are used to generate statistics that reflect the health of the services; and probes are used to query and control internal service states by management applications.

- The instrumented data is consumed by rules that enforce and govern Web service compliance with respect to the business operations requirements. Each rule can contain metrics or performance indicators that are checked against a set of conditions. The result of the check is then used to trigger and set off the pre-determined management actions.

We make a distinction between rules that are applied in a service operation environment and rules that are relevant to the fulfillment of the business operations. In a service environment, the rules are typically concerned with conditions and actions that technology operators and administrators deal with. An example of such rule is "if the number of SOAP packets processed in the last 60 seconds exceeds 600, auto-provision a new instance of Web service." Service level fulfillment rules are concerned with tracking data and events abstracted at the business operations level. For instance, a service-level rule may state the following: "promote the customer to the next service level if the business service consumer has used the service more than 6 times in the last month."

Perhaps not so obvious from Figure 5 is the fact that the service entities are not immune to updates and changes introduced by new business and technical decisions. For instance, when an existing business service is introduced in a new geography, the existing service definition may need to change to take into consideration a new service parameter that reflects the regional consumer protection scheme. In another example, the business service provider may decide to monitor a new business event as an indicator for adjusting inventory forecast. The Web services management environment must deal with these service life cycle management issues and provide the appropriate mechanism to help with version control, change notification, service deployment, and update in manners that are less intrusive to the business service consumers.

Finally, we must not forget that the identities of business service consumers, business service providers, and Web service operators are represented as digital identities in the online world. The processes of authentication and authorization are necessary to control access, usage, and administration of the Web service. Entitlements are used to represent the rights and privileges given to these parties.

Web Service Management Issues

It is logical to expect that Web services will be able to offer the same level of technical command and control capabilities as other IT solutions. However, this expectation is yet to materialize today without some generalization and implementation of Web services management technology.

Specifically for Web services, the management information model will have to document the different internal service states, desired views of the services (for example, the number of times a service has been accessed in the last hour), and the rules that govern how those states and views can be attained. Subsequently, various management activities

can be automated and driven through the service metadata that is embodied in those management documents.

The logical entities represented in the service model shown in Figure 5 can be further analyzed to derive management-related properties, entry points, and task-oriented rules. Such information is to be expressed as management metadata that is used by IT service management tools to monitor and adjust the management environment to satisfy the operational requirements and fulfill the business operations requirements.

For example, the management metadata can describe the mapping between a Web service instance and the corresponding software components that can be instrumented, configured, and deployed. Properties and attributes for the components describe the events, probes, and performance counters the managed Web service has been defined to expose. A collection of operations rules determines how the instrumented data should be handled and how the service requests should be routed in order to satisfy service level requirements.

Web Service Management Technology Framework

In the previous sections, we described the logical entities that are derived from a simplified service model. We then examined the relationships between these entities as we discussed the mechanisms that are needed to manage Web services.

In this section, we describe the technology framework that can be overlaid onto existing people, process, and technology capabilities to support managing Web services.

The Web services management issues we have presented can be summarized in the following way:

- Control the identities that can use, operate, and administer a Web service.
- Gain visibility and control into the performance, usage, and health of a Web service at the business operations, application, and system levels.
- Manage versioning and deployment of a Web service's updates and changes.

Our Web services management technology framework to support these requirements is structured around the following four core technology pillars that correspond to the categories of management issues that need addressing:

- Web service life cycle management
- Web service health management
- Web service deployment management
- Web service identity and access management

Web Service Life Cycle Management

Some common observations of the ITIL, MOF, and MSF approaches to IT service management discussed earlier include the following:

- Each framework takes a process approach that when captured as a high level model, describes the complete life cycle phases of an IT solution.
- The life cycle phases can be analyzed to derive functions that need to be performed at each phase. Each function can contain a set of tasks that must be executed to complete each function.
- Functions and tasks can be assigned to role responsibilities.
- In the same way, we assert we should use the same life cycle approach in managing Web services.

Commercial software packages are hard to deploy, maintain, and update in the production environment. Web services are no exceptions. This observation shouldn't be surprising when considering the linkages of the phases in a service's life cycle, as illustrated in Figure 6.

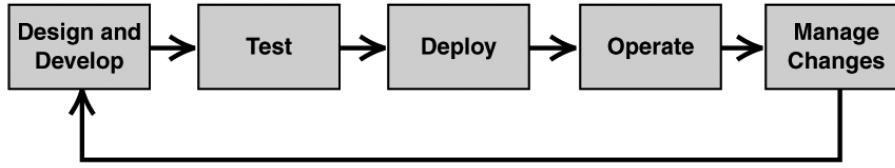


Figure 6. Service life cycle

The approach toward using the service life cycle shown in Figure 7 helps an IT team work together to manage a Web service to meet its goals.

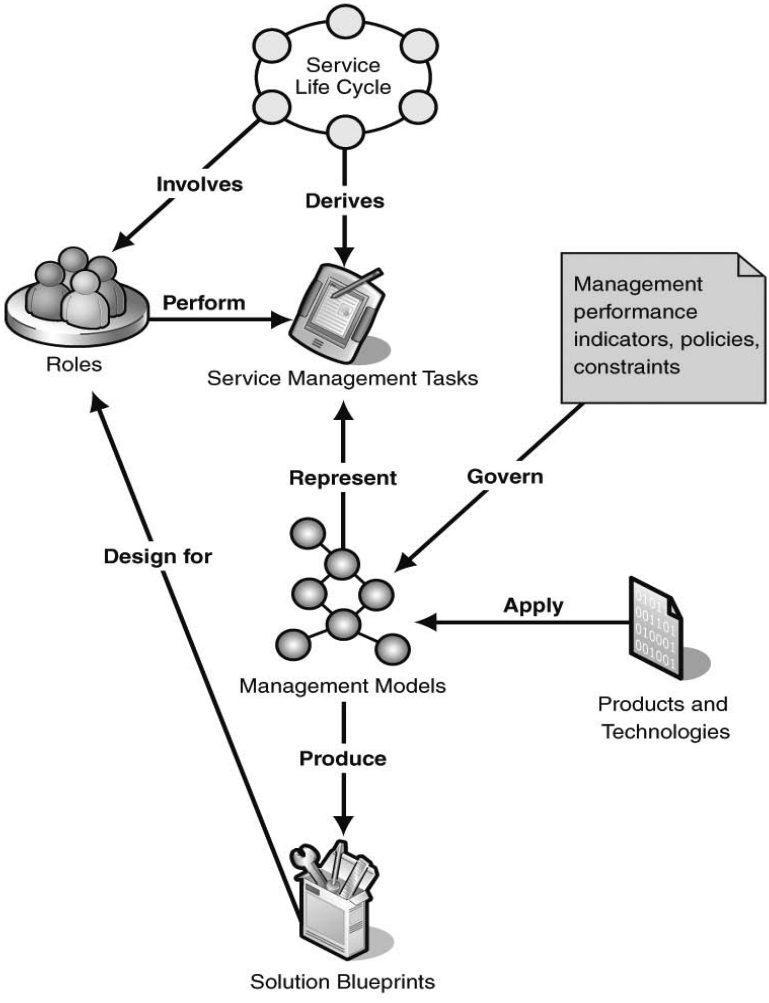


Figure 7. Using the service life cycle

The service life cycle approach helps the team:

- Consider the non-functional attributes that allow services to be operated, maintained, and upgraded over the course of time.
- Identify the roles involved in the service life cycle and derive management tasks that each role performs.
- Describe the conceptual elements and constraints of the management tasks using a collection of management models.
- Build solutions for the management models using products and technologies. The resulting tools and applications are tailored to meet the needs of the individual roles in the organization.

The people responsible for each phase of the life cycle are often separated organizationally and sometimes also geographically. There has been little success to date with tools to help bridge communication across these phases. So today, there is often a gap in assumptions between the programmers who have implemented the software and the operators who are responsible for the day-to-day availability, reliability, and security of the software. This observation is at the core of model-based development and management, namely that the design and implementation of software services must take into consideration the deployment and operational requirements to reduce the ongoing service management cost.

Note Visual Studio 2005 Team System (VSTS) is a new model-based development tool from Microsoft especially designed with the DSI model-based management approach in mind. For more information about VSTS, see the Microsoft Web site (http://msdn.microsoft.com/teamsystem).

Web Service Health Management

Because there is no such thing as a perfect environment, a Web service will experience problems at some time during its execution. If a problem is not detected in a timely fashion, not diagnosed correctly, or not fixed properly, the Web service can degrade to the point where it is no longer available to its customers. This is expensive, time consuming, and ultimately creates dissatisfaction among users. It is up to the administrators to detect and fix issues with as little downtime for their users as possible.

A Web service health model defines what it means for a Web service to be healthy (operating within normal conditions) or unhealthy (failed or degraded) and the transitions in and out of such states. Good information on a Web service's health is necessary for the maintenance and diagnosis of running systems. The contents of the health model become the basis for system events and instrumentation on which monitoring and automated recovery is built.

To keep a Web service up and running, the operations team needs to watch the Web service's health metrics, detect symptoms of a problem early, correctly diagnose the cause of that problem, and fix the problem before the application performs unacceptably. This activity is referred to as *health monitoring and troubleshooting*.

To create a Web service health model, the modeler needs to do the following:

- Collect the right information about the Web service at the right time—when running normally or when something fails.
- Document all management instrumentation exposed by a Web service.
- Document all service health states and transitions that the Web service can experience when running.
- Identify the steps and determine the instrumentation (events, traces, performance counters, and Windows Management Instrumentation [WMI] objects/probes) necessary to detect, verify, diagnose, and recover from bad or degraded health states.
- Document all dependencies, diagnostics steps, and possible recovery actions.
- Identify which conditions will require intervention from an administrator.
- Improve the model over time by incorporating feedback from customers, product support, and testing resources.

This section provides the background information for describing and understanding the health model concepts. Figure 8 provides a partial service life cycle view of when health models can be effective in specifying the manageability requirements. This high-level health model provides the fundamental health modeling framework that can be further extended and analyzed to identify the potential problems, the indicators of the problems, the diagnostics steps, and the recovery actions that need to be taken to ensure that the Web service's health and performance meets expectations.

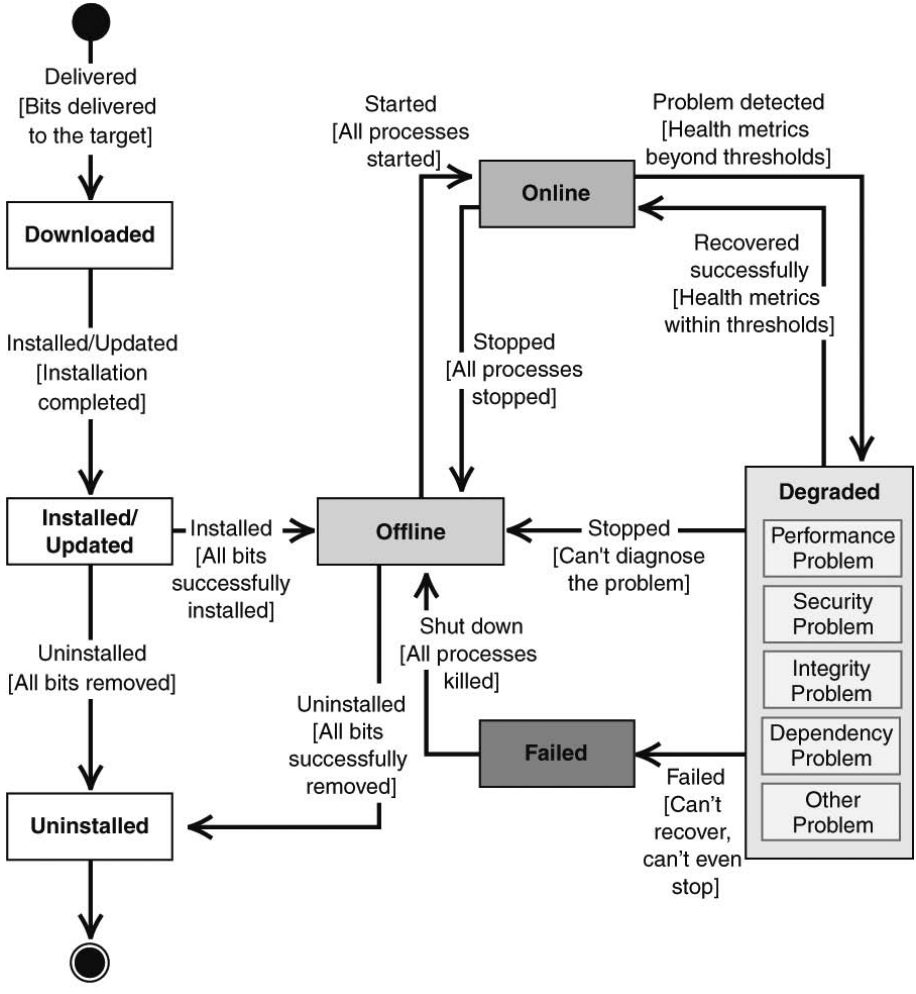


Figure 8. Service life cycle from a health model perspective

Health states are high-level indicators of the Web service's ability to function correctly. A health state provides non-quantitative data, but it tells the administrator if the Web service is in trouble and gives an idea of the severity of the situation. It is important to remember that a health state is a judgment call about the severity and is almost always relative. For example, one customer may decide that a total network saturation level of 90 percent is a degraded state, while another customer would consider a level of 80 percent to be in a failed state.

The health and diagnosability workflow as shown in Figure 9 defines logical stages of the monitoring and problem recovery process. The stages of this process are:

- Detection
- Verification
- Diagnostics
- Resolution
- Re-verification

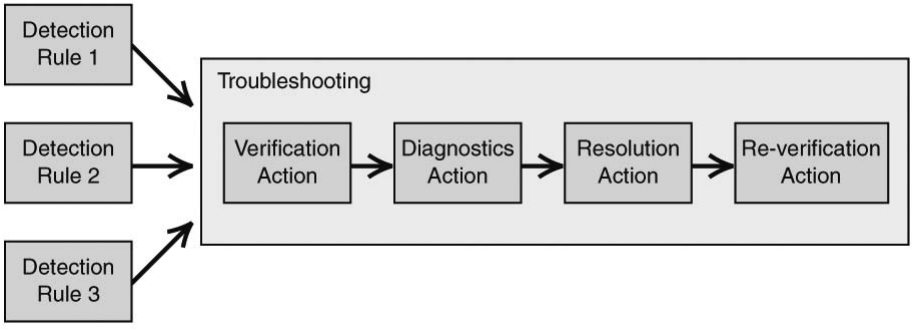


Figure 9. Health and diagnosability workflow

To be effectively monitored, a Web service needs to provide the right types and levels of instrumentation—it needs to expose its internal state, report changes in its state and behavior, and supply methods for administrative control. This provides the data essential for problem detection, verifications, diagnosis, and recovery. Additionally, the instrumentation needs to be lightweight and have no noticeable effect on an application's performance. There are three primary types of lightweight instrumentation that can be used for different aspects to support the health and diagnosability workflow:

- Events. An application sends out an event when the service encounters a particular operational condition or a failure state.
- Traces. Applications log trace information when a certain checkpoint in the code is passed.
- Performance counters. Performance counters are numeric values that reflect the state of the application at a precise moment or averaged over a period of time.

More detailed information on Web service health modeling can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Web Service Deployment Management

After a Web service satisfies all testing requirements, it is ready for deployment. Traditionally, deployment of software services requires preparing the network infrastructure, hardware, and operating systems. The same rigor in preparation also applies to Web service deployment. An effective Web service deployment mechanism must allow the Web service images to be centrally prepared and configured and efficiently distributed throughout the network and Web service farm environment. Configuration management must deal with different types of management data, including service policies that service clients need to adhere to; software settings that affect how the Web service uses application, system, and network resources; and service catalog information that enables the running Web service instance to be discovered and operated. Technical solutions for configuration management must also deal with change propagation latencies that affect when configuration changes can take effect at the affected Web services.

A logical architecture for Web service deployment is shown in Figure 10.

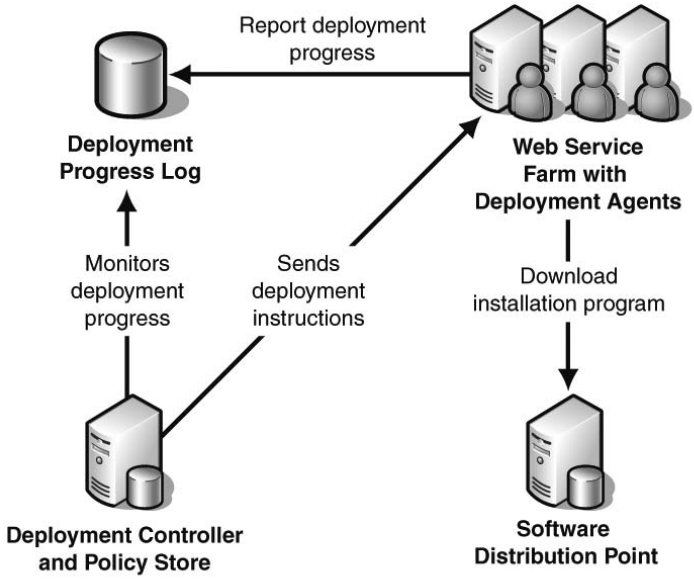


Figure 10. Logical architecture for Web service deployment

The logical architecture consists of the following entities:

- Deployment controller. The application administrator uses the deployment controller to:
  - Configure and manage deployment policy.
  - Schedule and trigger deployment actions at the Web service farm computer.
  - Monitor deployment progress and take corrective actions if necessary.
- Deployment agent. This interacts with the deployment controller and performs software deployment actions according to the instructions and policy received from the deployment controller.
- Deployment policy store. This provides the central location where the deployment policies are managed and stored.

- Software distribution point. This provides the network location where the installation program and Web service software component can be downloaded at deployment time.
- Deployment progress log. This provides the logical location where deployment agents write deployment status and the deployment controller monitors progress.

Over time, new technical and business requirements will necessitate technical re-evaluation and possible changes to existing implementations of Web services. The technical changes can be anything from the Web service access address, to message schema updates, to service interface or name modifications. When implementation changes occur, service providers are confronted with the challenges of minimizing or insulating the change impact to service consumers.

For example, if a local U.S. business offering financial service decides to expand its service offering to a different state in the country, the service will need to take into consideration the new state's consumer regulations. It may require additional input or output data be passed between the business and the consumer. When a Web service is used to enable this service, a couple of technical options may be considered to facilitate the new data exchanges. The business may choose to introduce a new Web service interface to support the new data parameters. Or it may choose to use a Web service proxy façade to support the existing interface, relying on the proxy to perform data transformation and internal service routing to support messaging requests from current and new service consumers. (With this latter option, consistent with the four tenets for service design, the service boundary conceptually shifts outward from the original Web service implementation to the new Web service proxy façade.)

The software industry has been applying software version control techniques and process to traditional software development for a long time. Some of the existing versioning techniques need to be adapted and applied in the Web services world to differentiate changes in service implementations and message schemas.

Depending on the Web services architecture, changes to the Web service interface or message schema may affect the Web service consumer and the Web service proxy. In either case, the consuming party needs to discover the new service definition in order to interact with the new Web service endpoint. The Web service architecture needs to specify whether the change notification and discovery is handled automatically or manually.

More detailed information on Web service deployment can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Web Service Identity and Access Management

As one of the co-authors has observed in a related work, identity and access management is the set of processes, technologies, and policies for managing digital identities and for controlling how digital identities can be used to access resources. (Frederick Chong, Identity and Access Management, *Microsoft Architect Journal 3*, July 2004 (http://www.microsoft.com/architecture/library.aspx?pid=journal.3&abver=E9A00024-3DC1-4B6A-BC20-22716E4D2FEA&id=http://msdn.microsoft.com/architecture/journal/default.aspx?pull=/library/en-us/dnmaj/html/aj3identity.asp.)

In the Web services management context, there are three primary parties whose identities we are interested in managing:

- Web service consumers. Web service consumers are the users of the services. However, the definition of Web service users can vary depending on the consumption environment. For example, in a development and testing environment, the users are the software developers and testers; whereas in a production environment, the user population expands to include the end users of the services. Even within the end user environment, the consumers may be further classified according to the rights and privileges conferred through their service agreement with the service provider.
- Web service providers. This group includes people and system identities associated with the organization providing the services. In many cases, users in service providers are interested in the usage and the performance patterns of the Web services to help them determine future technical and business directions. They may also need the ability to configure and update service level settings of the service instances.
- Web service operators. This group usually refers to the administrators and operators concerned with managing the Web service execution and hosting environment. Web service operators monitor the health and performance of the Web services and ensure that the service operation rules are followed. These personnel also often deal with deploying and updating the Web service instances and will require visibility and control over the network and operating system infrastructure that are hosting the Web services.

Figure 11 shows the conceptual architecture of the Web service identity and access management framework.

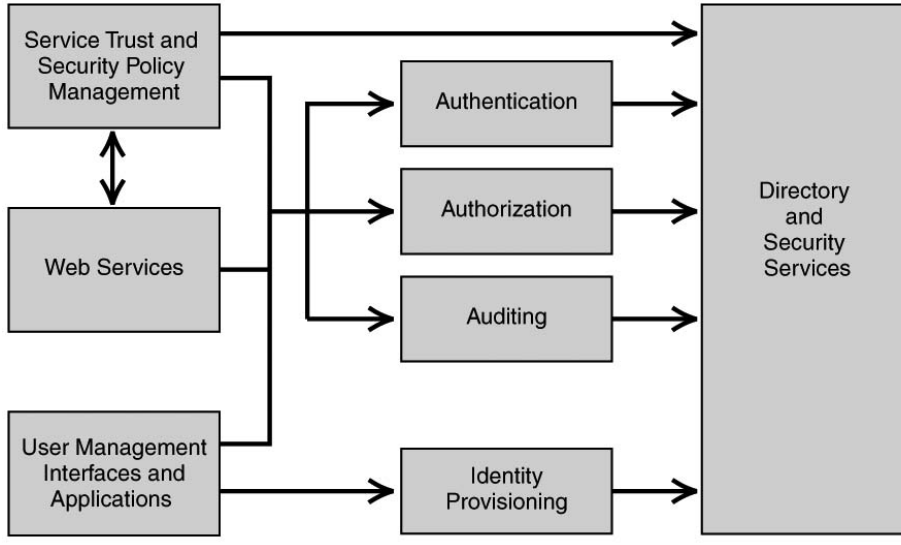


Figure 11. Web service identity and access management conceptual framework

The core function of access control is provided through three underpinning security mechanisms: authentication, authorization, and auditing (commonly known as AAA). These three mechanisms depend on the directory and security services layer for security data-related operations. For example, storing and retrieving user credentials, entitlements, and profiles; and discovering security policies.

A provisioning component provides the capability to initialize, update, and synchronize the security-related data. User management applications and interfaces can then rely on the provisioning component to create, retrieve, and change the identity information without having to interact directly with the data layer.

Trust relationships and security policies are critical elements in an identity system because they determine the type of security proofs, statements, and actions that the Web services will accept from trusted parties. A trust and security policies management component is responsible for updating such security configuration information in the directory and security services, and for configuring the Web service instances to use the security settings.

The next few sections highlight a few important concepts in identity and access management and how they relate to Web services.

Authentication and Single Sign-On

Authentication is the process of validating the credential of an identity. The subject of authentication may concern members of the Web service consumers group, members of the Web service providers group, or members of the Web service operators group.

Most enterprises have more than one identity system to support existing services and applications. As a result, enterprise users are commonly faced with the challenge of managing multiple authentication credentials for login. Single sign-on mechanisms help deal with the complexity of using and managing multiple application credentials. Identity and access management system in the service-centric world often have to deal with the challenges of integrating with existing identity systems as well as delivering a single sign-on experience to the service users.

Entitlements and Authorization

Entitlements refer to the rights and privileges conferred to an identity while authorization is the process of verifying that a given identity has the permissions to access a protected resource—in this case, a Web service.

Like authentication, the process of authorization also applies to different kinds of tasks performed by the various categories of identities. For a Web service consumer, other than verifying that it has the ability to invoke a Web service, the Web service instance may also check more granular data access permissions to verify that the consumer is allowed to receive a particular data field in the Web service response. For Web service providers and Web service operators, the authorization process verifies that the identity has the rights to view and update internal business and operation-related information respectively.

In addition to traditional access control lists, role-based and rule-based authorization mechanisms are commonly used to represent and enforce authorization policies.

Security Auditing

Many industries involved in services are governed by governmental regulations. For example, in the United States, recent enactment of the Sarbanes-Oxley Act (Sarbox) and its compliance requirements put pressure on enterprises governed by the Sarbox law to re-evaluate their current systems for any process and technology gaps that could cause them to fail the Sarbox audit.

Given the present corporate focus on governance and transparency, information that answers the questions of "who did what, when?" needs to be readily available. Security audits provide the mechanism for tracking security actions and information access. In many cases, audit trails may also need to prove legal intent and non-repudiation so that the information can be used as evidence in the court of law.

When the reach of business information and actions is extended through Web services, it becomes even more critical for auditing and evidentiary capabilities to be integrated into the Web services implementation. The business operations modeling approach described earlier can be used to help identify the activities in a business collaboration that show legal intent. The transformation of this model into business operations requirements can help the IT team integrate appropriate capabilities into the Web service implementation.

Privacy

Privacy is concerned with the business use and disclosure of individual data. In many cases, the privacy requirements have to do with how legitimate business entities that are using personal information agree to protect and limit the use of the data that is harvested from the Web service consumers.

Privacy and security are actually paradoxical concepts—as we become more secure through the processes of authentication, authorization, and auditing, it becomes easier to track, gather, correlate, and reveal user information. Therefore, it is very important that for every security condition and event that generates identity data, there is a set of peer conditions that limit the use of that data. Some of these conditions may need to be translated into machine-enforceable business rules. A Web service implementation can then depend on these business rules to restrict the visibility and access of the data.

Identity Life Cycle Management

Identity life cycle management refers to the process of identity provisioning, identity de-provisioning, and updating accounts, profiles, and entitlements associated with identities.

As mentioned earlier, most enterprises have existing identity systems. In many cases, Web service implementations must integrate with these current identity schemes. Web services depend on these systems to authenticate and authorize identities and to gather individual attributes such as phone numbers.

Identity provisioning is the process of creating and initializing accounts, credentials, and attributes in the identity systems. Identity de-provisioning is the process at the other end of the identity life cycle that de-activates the account. While the account is active, account information and credentials may be changed and updated.

Often, identity data in an organization may spread or be duplicated across different systems so that some level of data synchronization is required to maintain a consistent view of the identities. Identity integration and synchronization technology is used to address this class of identity management problems.

Identity Federation

To fulfill the Web service technology promise of facilitating cross-organization communication, it must be possible for organizations to verify and authorize the use of a partner organization's identities. Identity federation refers to the process and technology for managing and configuring trust relationships and to using identities across security administration domains.

In the Web service world, identity federation enables an organization to delegate specific security responsibilities to trusted third parties. For example, a service provider can configure its Web services to trust a well known identity provider to authenticate the identities of its Web service consumers, thus simplifying the identity management infrastructure it needs to employ while extending business reach through Web services. In other cases, a Web service provider may choose to further delegate the user authorization process to a trusted identity system of the business partner who is consuming the service.

Web Service Management Using Web Services

The discussion about Web service management has focused on the management of Web services. Another topic worth mentioning is the management of Web services (or other applications or systems) using Web services.

Whether it is used for managing Web services or other computing software or devices, a management infrastructure involves communication protocols to facilitate management functions such as discovery of managed Web service instances and raising management events to an eventing infrastructure. Existing examples of industry-standard management protocols include SNMP (Simple Network Management Protocol) and DMTF CIM (Distributed Management Task Force Common Information Model).

With the adoption of Web services technology for a variety of application and infrastructure services, such as federated identity management, the management protocol community is also realizing that the tenets of service orientation and Web services technology are highly supportive of desirable distributed management goals. For example, discoverability speaks of the ability for managed resources to be dynamically discovered, and federation enables management applications to gain visibility into trusted infrastructure without requiring complete control.

WS-Management is a Web services–based protocol that combines the practicality of a small number of fixed operations with the scalable, composable Web services architecture as its technical foundation. (For more information about WS-Management, see the Microsoft Web site, http://www.microsoft.com/whdc/system/pnppwr/wsm/default.mspx.) WS-Management is largely a composite definition over other existing Web Services standards and specifications that describe the management operations. Because the protocol is defined as a composable standard, it can be implemented in the resource constrained environment, eliminating the need for a low-level wire protocol.

Conclusion

Like most important IT initiatives, managing Web services requires the balanced coordination of people, process, and technology.

We grounded the discussion of Web service management in the broader contexts of connected systems and the Dynamic Systems Initiative. We provided some perspectives to consider in determining what the business goals of a Web service actually are to help pin down more formally why and how a Web service should be managed.

The model-based development and management approaches suggested that a framework for Web service management addressing management states, actions, and rules could be overlaid onto existing IT service management systems.

We proposed such a framework that addressed how to coordinate people, process, and technology issues throughout the service life cycle. After taking a simplified view of one of the perspectives of a Web service's goals, we examined issues of Web service life cycle management, Web service health management, Web service deployment, and Web service identity and access management.

Web Service Solution Design: Developing a Solution Design for Web Services in the Northern Electronics Scenario

Architecture Chronicles

Dynamic Modeling: Aligning Business and IT

Frederick Chong with Jim Clark, Max Morris, and Dave Welsh
September 2005

Applies to:

- Enterprise Architecture
- Solution Architecture
- Service Oriented Architecture (SOA)
- Service Oriented Management (SOM)
- Application Integration
- Business Process
- Business Operations Modeling

Summary

The *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* seek to present to business, solution, and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. This document focuses on the high-level issues of the Web service design for a product shipping solution in the Northern Electronics scenario.

Contents

This Document

This document is part of the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT*. This volume seeks to present to business, solution, and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. The information map to the series provides an up-to-date description and cross-index of the information available and can be found at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Abstract

This document describes a high-level solution design for the Web services that enable the automation of certain aspects of the product shipping process at Northern Electronics. It describes how to use different software components in a solution to realize the normal operation and exception flows in a business process. It illustrates how a solution architect can use model-driven design and service orientation approaches to develop a design for a flexible, interoperable solution. The document also illustrates how a solution can provide real-time notification of business operations conditions that need attention without attempting to automate rectification actions that can be unduly complicated and costly to implement.

Acknowledgments

Many thanks to Nelly Delgado for her help with technical writing, Claudette Siroky for her graphics skills, and Tina Burden McGrayne for her copy editing.

The authors would also like to thank Gajanan Phadke, Vishal Kapoor, Amol Wankhede, Aziz Matheranwala, Amit Kumar Srivastav, Bhushan Pawade, Bhasha Johari, Avadh Jain, Mughda Gadre, Maneesha Nalawade,Sonika Arora, Anil Sharma, and Anurag Katre (all of Tata Consulting Services) for their contributions to the implementation of the Northern Electronics solution.

Introduction

This document focuses on the high-level issues of the Web services solution design for Northern Electronics's product shipping solution. Background information about the Northern Electronics scenario can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Northern Electronics's products are manufactured in China, shipped to a port in Seattle, Washington, and then transported and stored in a warehouse in Everett, Washington. The products are then picked up and transported to customers across the United States. The company is working to improve the product shipping process, which is part of the broader product pickup and delivery process. Product shipping involves a great deal of coordination. Product shipping includes ordering the transport, moving the right product in the right amount from the right warehouse to the right loading dock, with the right accompanying paperwork, loaded onto the right truck at the right time. Getting everything to work out in product shipping so that the right products are shipped to the customer in a timely and cost-efficient manner is a challenge.

The hand-off of goods from the warehouse involves the coordination and cooperation of partner companies who are transport consolidators. Northern Electronics needs a way to effectively and efficiently communicate with the transport consolidators to coordinate the business activities and to handle problems as soon as they occur. There are many problems that arise quite normally in product shipping. Businesses expect these problems to arise and handle them as they occur. From the perspective of the warehouse, the truck might not arrive on time or the truck might arrive but be the wrong truck. From the perspective of the customer, failure ranges from non-delivery of product, incorrect delivery of the product (the wrong product and/or the wrong amount), delayed delivery of the product—or some combination of these failures. Done poorly, the activities are sequenced wrong and significant delays (and costs) are involved.

Understanding the Product Shipping Process: An Example

Part of Northern Electronics's product line is the manufacturing of electronic parts for remote-controlled airplanes. One of Northern Electronics's customers, Wingtip Toys, assembles remote-controlled airplanes in Dallas, Texas, and then sells the airplanes to retail companies. Northern Electronics hires Acme Consolidation Company, a freight consolidator based in Portland, Oregon, to arrange the transporting of the goods from the warehouse in Everett to the wholesaler's warehouse in Dallas.

In this case, Acme Consolidation Company hires Blue Yonder Truckers, a local trucking company to pick up and transport the goods. Figure 1 describes the high-level business entities involved in the product pickup and delivery scenario.

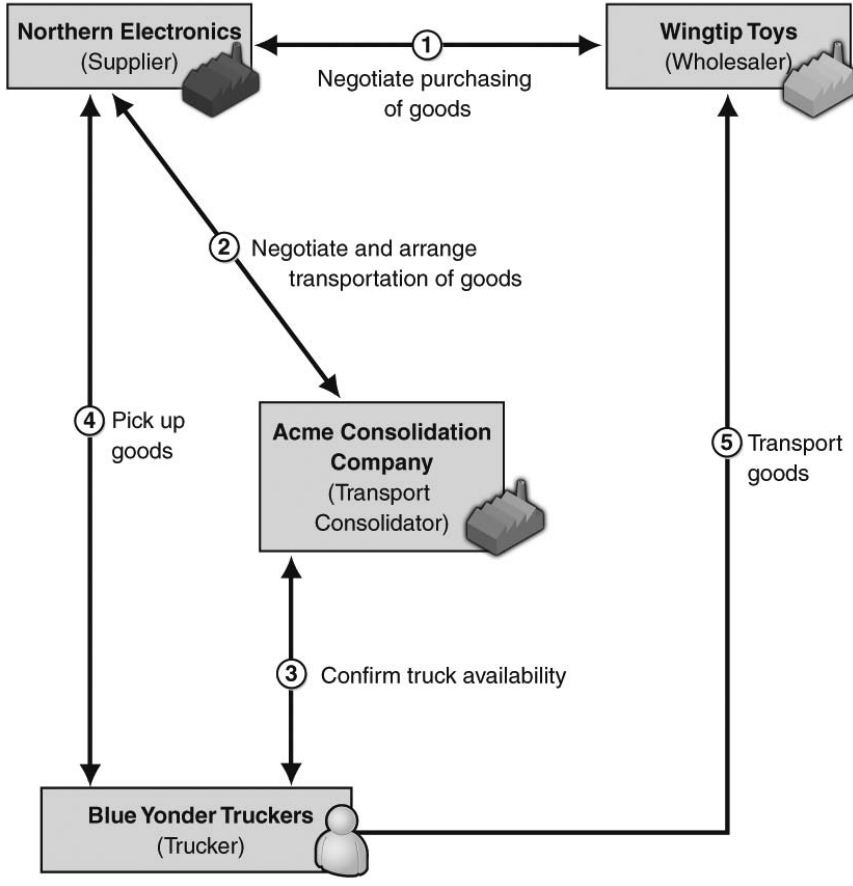


Figure 1. Entities involved in the product pickup and delivery example

Some of the business activities are internal activities that are relevant only to Northern Electronics. However, other business activities rely on a successful collaboration between Northern Electronics and Acme Consolidation Company.

Who's Who in Northern Electronics

The following individuals from Northern Electronics are involved with designing and implementing the product shipping solution:

- Pam: She is the business program manager whose responsibility in this context is to understand and model the product shipping process.
- Zack: He is the head of IT architecture whose responsibility in this context is to work with Pam to help her understand the IT organization and its requirements.
- Jackie: She is the program manager whose responsibility in this context is to work with Pam, the business program manager, to identify the functional needs of the business users and ensures that the solution meets those requirements.
- Tom: He is the solution architect whose responsibility in this context is to work with Jackie, Pam, and Zack to understand the business and technical requirements. Tom also provides the technical expertise to design the solution.
- Sam: He is the lead developer whose responsibility in this context is to collaborate with Tom to design and implement the solution.

Developing a Web Services Solution Design

Tom's primary responsibility as the architect is to design the solution. To do that, he considers:

- The business operations requirements as provided in specifications by Pam and Zack.
- The non-functional attributes that enable services to be operated, maintained, and upgraded over the course of time.

Tom follows a model-driven development approach that uses highly structured information models to capture the intentions and desired results of stakeholders such as Pam and Zack. In terms of the business operations requirements in particular, this approach ensures that Tom proactively thinks about the instrumentation and tasks required to achieve the desired consequences; he also thinks about actions for the mitigation strategies in the event of business exceptions.

In addition, an overarching mandate that Tom received from the company's executive team is to improve Northern Electronics's ability to remain competitive and grow through the use of information technology. In the near future, the business will need to coordinate with more companies, and specifically, more transport consolidators. Each additional transport consolidator represents another organizational boundary and potentially another platform that Northern Electronics will need to send and receive information in a secure way in real time.

Before designing the solution, Tom reads and thinks a lot about the benefits of a service-oriented architecture. In particular, he decided that it is especially crucial for the solution to observe the four tenets of service-orientation as described in Service Orientation and Its Role in Your Connected System Strategy ("srorientwp"). Tom also surveys the technology options for implementing the solution. He performs a feasibility study and concludes that Web services are the right building blocks for implementing a service-oriented solution.

Web Service Solution Design: Business Services to Web Services

Tom works with Pam and Zack to identify candidate Web services for the product shipping business collaboration. They review the way shipping is currently done while thinking about what can be improved through automation. Pam and Zack already considered this question from the business perspective when they looked at the business services involved in product shipping. While the business operations specifications capture the detail of that analysis in a formal way, Tom needs the consultation with Pam and Zack to help him get a thorough understanding of what is happening and why the specifications have the requirements they do.

One example of something they review together has to do with when the product shipping business collaboration begins. The negotiation between customers and Northern Electronics is done verbally, usually over the phone. This is outside the scope of the product shipping business collaboration. Moreover, the negotiation is not a good candidate for something that can be automated because the human interaction is probably a necessary aspect of negotiating the purchase. The starting point of the product shipping business collaboration is when the purchase order (PO) is created. The PO is an artifact that can be stored electronically instead of on paper.

Another example of something they review together is the ordering of transport. After the PO is created, the supplier shipping clerk checks to make sure the items are in stock. If the items are in stock, the supplier shipping clerk works with the transport consolidation clerk to coordinate the transport that will pick up the shipment and deliver it to the customer.

Yet another example of something they review together is the security requirements. Identities of the participants in the business collaboration must be represented as digital identities in the Web services domain. The processes of authentication and authorization are necessary to control access, usage, and administration of the services, especially across administrative boundaries—in this case, between the supplier and the transport consolidator. Tom has to make sure he understands issues such as who has the authority to create or change transport order information and who can confirm acceptance of transport. These issues are especially important to meet the business operations requirements, which outline strict legal requirements. For example, they specify the role of who in Northern Electronics is actually allowed to order transport from a transport consolidator (and therefore commit the company to payment for the ordered transport). (Other requirements related to security and identity, such as logging specific activities as they are performed by certain roles, are expressed in the business operations specifications.)

Tom agrees with the conclusion from Pam and Zack that the interactions between the supplier and transport consolidator are good candidates for being automated and implemented using Web services. He reasons that:

- If the transport consolidator has a Web service the supplier shipping clerk can use to send his request to, instead of relying on a phone call, it is unlikely the request will get lost. The supplier shipping clerk can also get proof that the request was received. The transport consolidation clerk can then process the request and generate an acceptance.
- If the supplier has a Web service the transport consolidator can use to send his acceptance of the transport order to, instead of relying on a fax machine, it is similarly unlikely the request will get lost. The acceptance can contain the details and terms of the product pickup. The transport consolidation clerk can get proof that the acceptance was received and thereby that the terms were agreed to.

After Tom examines the business collaboration further, he reasons that the supplier shipping clerk and loading dock clerk can work more efficiently by using an internal Web service to let them know about the shipments that need to be loaded onto the docks in the correct order based on the trucker's expected time of arrival. They can use this internal Web service to record the trucker's credentials and can also use it to confirm that the shipment was picked up. In turn, the internal Web service can then automatically send a pickup confirmation to Acme Consolidation Company's Web service.

Based on this analysis, to meet the requirements laid out in the business operations specifications and to create the internal efficiencies he identified, Tom proposes a design that includes three Web services to facilitate the product shipping business collaboration:

- ShippingService Web service. This is the supplier's Web service that is used to send and receive the details of the shipment pickup.
- PickupService Web service. This is the supplier's Web service that is used internally to be notified of product pickup and to confirm the shipment was picked up.
- TransportService Web service. This is the transport consolidator's Web service that is used by the supplier to initially order the transport and finally to confirm that the shipment was picked up.

Obviously, the TransportService Web service is not something that Northern Electronics can implement, because it belongs to the transport consolidator. But it is a needed part of Tom's design.

Tom works with Zack and Pam to coordinate the development of the TransportService Web service with the chosen transport consolidator business partner, Acme Consolidation Company. If the approach proves successful, Pam and her team can organize similar coordination with other transport consolidation business partners.

Note: The coordination with Acme Consolidation Company is outside the scope of this discussion, which simply relies on the coordination happening successfully.

Tom uses the business operations specifications to define the actual interfaces of the Web services. A more detailed description of the design process he uses is provided later in this document.

Web Service Health Model: Designing for Operations

The proposed three Web services allow messages to be sent and received in a systematic and verifiable manner. In addition to satisfying the business operations interface requirements, Tom also needs to think about how service instrumentation can be used to provide visibility and control over running instances of Web services. Web service instrumentation may take the following forms:

- Debugging traces can help with diagnosing code execution behaviors.
- Events are used to raise alerts that signify meeting certain management conditions.
- Performance counters are used to generate statistics that reflect the health of the services.
- Probes are internal service states that can be queried and controlled by management applications.
- Tom needs to make sure the two Web services that Northern Electronics will implement can be designed for operations. This means that the Web services need to be designed to be managed to meet their performance objectives at multiple levels.

The IT organization uses a health modeling approach to develop a thorough understanding of how the Web services should perform, how to learn about how they are performing, and what to do when they are not performing according to plan.

Health modeling is a process within the design phase of the service life cycle to formally document the understanding of the system designers about what a healthy system is and what an unhealthy system is. It considers in advance all the manageable conditions the designers can anticipate. The health model that results contains detailed consideration of each anticipated manageable condition by laying out how to identify it and the actions to take to manage it. The health model details how to:

- Detect a manageable condition.
- Verify that it really is the condition.
- Diagnose what might be causing the condition as needed.
- Resolve the condition through corrective actions as needed.
- Re-verify the condition to see whether the system is in good health.

Tom works with others in the IT organization and relies on existing policies and procedures to formulate an initial health model for the two Web services he has proposed. This initial plan takes into account the company's standard system-level and application-level health models for Web services (for example, the requirement to perform "heartbeat" checking as a way to ensure a Web service is still running). Tom also needs to take into account the requirements in the business operations specifications.

More detailed information on Web service health modeling can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Product Shipping Web Services Solution Design

At a high level, Tom designs the flow of interactions between the individual workstations, Web services, and databases at Northern Electronics as shown in Figure 2. Although this document is primarily focused on the design concerns at Northern Electronics, it also describes the solution flow into the systems at Acme Consolidation Company to provide an end-to-end view of the automated process.

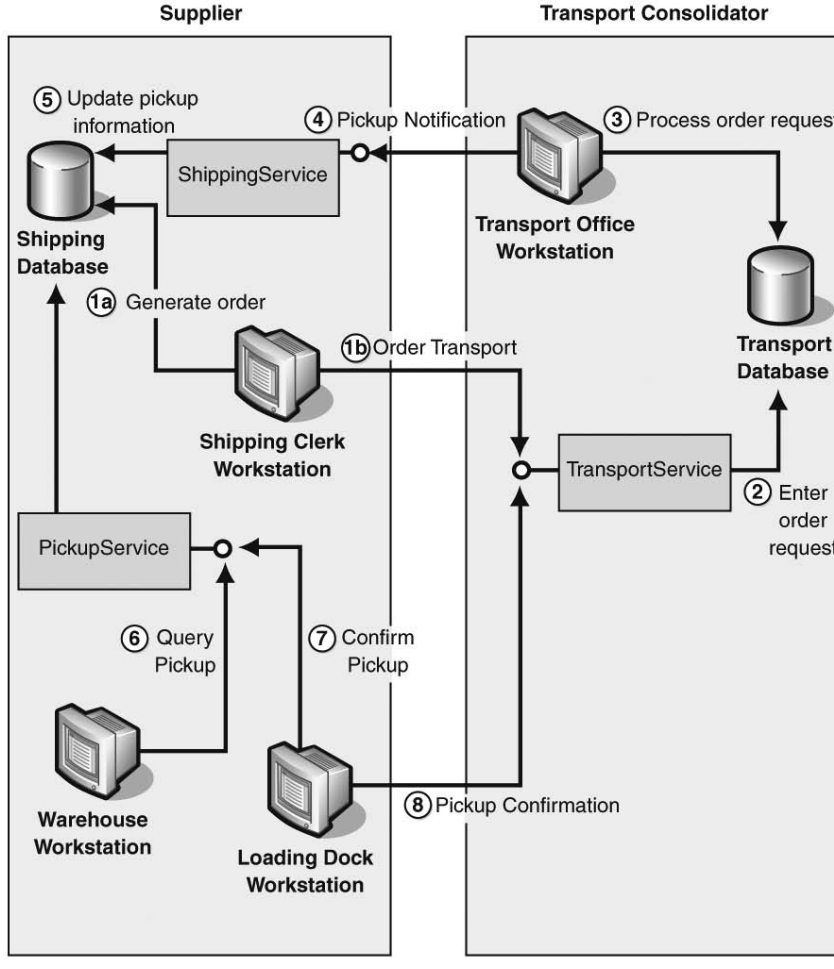


Figure 2. Product shipping service interaction flow

The following walkthrough begins with the assumption that the goods are already ordered by Wingtip Toys. The interaction flow outlines the automated process of obtaining transportation that will transport the goods to Wingtip Toys:

- Generate order. Richard, the shipping clerk at Northern Electronics uses a transport ordering application to query the Shipping database to find POs that do not already have arranged transportation. Every PO has a status field that is used to indicate the state of processing. The possible states are:
  - Received. When a PO is received by the supplier.
  - Processed. When a PO has an associated transport order that is in ordered status.
  - Shipped. When the goods are loaded and transferred to the contracted trucker.
  - Completed. When the goods have been delivered and payment received.

When a new PO is created and saved in the Shipping database, its status is set to *Received* and a new PO number is automatically generated. Therefore the *Received* status is used to search for POs that do not already have transportation ordered.

For each resulting PO of the previous query, Richard generates a new transport order request that is saved in the Shipping database. Only the shipping clerk or shipping manager can create a transport order. Every transport order has one of the following statuses:

- Initiated. When the supplier submits a transportation order.
- Ordered. When the supplier has processed the transport order notification.
- Arrived. When the truck has arrived at the supplier's warehouse.
- Loaded. When the goods are loaded and transferred to the contracted trucker.
- Completed. When the goods have been delivered to the wholesaler.

When a new transport order request is saved in the Shipping database, its status is set to *Initiated* and a new transport order number is generated for the request.

- Order transport. After the transport order is saved in the Shipping database, the transport ordering application creates a copy of the *TransportOrder* document with the order details and makes a Web service call to the transport consolidator's TransportService Web service.
- Enter order request. The consolidator's TransportService Web service receives the transport order request and saves it in the Transport database.
- Process order request. The transport consolidator clerk, Julie, uses a transport order processing application on the transport office workstation to query for new transport orders from the customers. For each new transport order, Julie finds a suitable trucking company to help fulfill the order. For example, after making a few phone calls to

confirm with the trucking company, Julie enters the details in the transport order processing application to reflect that Blue Yonder Truckers will arrive on the following Tuesday at 10:00 A.M.

- Pickup notification. When the transportation details are finalized, Julie uses the transport order processing application to create a *PickupNotification* document and sends it to the ShippingService Web service hosted by Northern Electronics.
- Update pickup information. The ShippingService Web service receives the *PickupNotification* and verifies that the information complies with the specified delivery options (such as pickup time and date). If the conditions are met, the ShippingService Web service sends an acceptance of the notification to Acme Consolidation Company and updates the Shipping database with the arranged transport information. The transport order status is set to *Ordered* and the PO status is set to *Processed.* If Northern Electronics does not accept the *PickupNotification,* the acknowledgment reflects this and also explains why the notification is not okay. One reason for rejecting the notification might be because the pickup time specified is different from the requested time.
- Query pickup. David, the loading dock clerk, uses a warehouse management application on the warehouse workstation to identify the goods that are going outbound to prepare the loading dock for load pickup. The application uses the internal PickupService Web service to query the Shipping database for pickup information. The PickupService Web service gets the list of transport orders that have the status set to *Ordered,* their respective POs, and prepares and returns a list of *ItemsToPickup.* The list is sorted according to the earliest time of pickup so that the goods can be prepared and queued in the right order on the loading dock. Based on the information in the transport order and POs, David arranges for the packaging of the items to be shipped and begins to fill out the appropriate documentation. This includes declaring the correct product identification and piece count. Northern Electronics must also declare that the parts they produce are not perishable, hazardous, or radioactive, and that the electronic parts do not include any batteries.
- Confirm pickup. On the day of the pickup, Bob, the trucker, checks in with Richard. Richard checks Bob's identification and verifies his paperwork. Richard enters Bob's time of arrival into the system and looks up which dock Bob should go to. At this point, the transport order status will be updated to *Arrived.* Upon completion of loading, David uses the warehouse management application to enter the weight of the load in the transportation documentation. Bob enters his signature in a signatory device connected to the loading dock workstation to assume accidental liabilities during transportation of the goods. David uses the warehouse management application to confirm that the goods were picked up. The PickupService Web service updates the transport order status to *Loaded* and the corresponding PO status to *Shipped* in the Shipping database. This formally transfers liability of the goods from Northern Electronics to Acme Consolidation Company.

- Pickup confirmation. The warehouse management application sends a pickup confirmation to the TransportService Web service to inform Acme Consolidation Company that the goods have been delivered to the trucker.

Designing the Web Services Message Contracts

One of the key implementation considerations is to ensure that the Web services on both sides will be able to communicate without having to agree on the vendor technology that is used to implement the services. For instance, Northern Electronics has chosen to implement their Web service clients and services using Microsoft .NET technology. However, Acme Consolidation Company should not be required to implement their Web services using the same Microsoft technology for the two companies to communicate.

To realize this goal, Northern Electronics and Acme Consolidation Company must first agree on the format and type of the messages that the Web services will send and receive so that they can communicate. To do this, the messages are described using industry standards such as XML Schemas (also known as XSD). Furthermore, it is also expected that the message schema description can be understood and consumed by the current generation of development tools to generate the corresponding software data structures for the respective development environment. For instance, XML Schema is one of the industry standard mechanisms for describing data formats and the Xsd.exe tool in Microsoft Visual Studio .NET can import a file containing the XML Schema representation of a business document to generate a Visual C .NET class that the developers then use for implementing the Web services.

Table 1 and Table 2 provide a high-level description of the business documents that Acme Consolidation Company and Northern Electronics have agreed to exchange. The *TransportOrder* document is the business document that is sent to the transport service at Acme Consolidation Company. The high-level schema of this document is described in Table 1. The *PickupNotification* document is the business document sent by the Acme Consolidation Company to the ShippingService Web service. The document is described by the high-level schema in Table 2. The *TimeWindow* data type is used by the *TransportOrder* and *PickupNotification* documents and is described by the high-level schema in Table 3.

Table 1. TransportOrder Document High-Level Schema

Field	Type	Description
VersionNumber	String	Version number of the business document.
TONumber	String	Transport order number assigned by Northern Electronics as an identifier of an order.
CargoType	String	Type of cargo. For example, garment, produce, fishery, etc.
ProductCode	String	International code for identifying the product.
Origin	String	Location for loading the goods.
Destination	String	Location for delivering the goods.
ExpectedWeight	Float	Expected weight of the goods.
Hazardous	Boolean	Whether the goods is classified as hazardous materials.
RequestedTime	TimeWindow	A time window when the transportation company can begin loading the goods from the loading dock.

Table 2. PickupNotification Document High-Level Schema

Field	Type	Description
VersionNumber	String	Version of this business document.
TONumber	String	Refers to the transport order number previously assigned by Northern Electronics.
CargoType	String	Type of cargo.
ProductCode	String	Product identification code assigned by an international standard body.
Origin	String	Location for loading the goods.
Destination	String	Location for delivering the goods.
ExpectedWeight	Float	Expected weight of the goods.
Hazardous	Boolean	Whether the goods is hazardous materials.

Field	Type	Description
ShipmentNumber	String	A shipment number assigned by the transport consolidator.
TruckLicenseNumber	String	This is the license number of the truck that will arrive to load the goods.
SuggestedPickupTime	TimeWindow	This is the time window that the trucker will arrive to load the goods.

Table 3. TimeWindow Data Type High-Level Schema

Field	Type	Description
NotBefore	DateTime	The beginning of a time window.
NotAfter	DateTime	The end of a time window.

For Acme Consolidation Company to invoke the ShippingService Web service, Northern Electronics must provide a service description for the Web service through a Web Services Description Language (WSDL) file. Sam, the lead developer at Northern Electronics,
completes the following actions to generate the WSDL file that defines the Shipping Web service:

- First, Sam uses the XML Schema editor in Visual Studio to create the .xsd files that describe the format of the *TransportOrder* and *PickupNotification documents*. A developer at Acme Consolidation Company reviews and accepts the formats.
- Next, Sam uses the Xsd.exe tool in Visual Studio to consume the .xsd files produced in step 1 and then generate the corresponding Visual C .NET classes.
- After that, Sam uses the ASP.NET Web service project template to create the Shipping Web service.
- Sam can then add a Web method named NotifyPickup() to the Shipping Web service. He specifies the Visual C class for PickupNotification as the method parameter. This is the Web method that Acme Consolidation Company calls in order to send the PickupNotification document.
- Finally, Sam builds the Visual Studio project to render the WSDL for the Shipping Web service.

The WSDL file for the Web service is then made available to Acme Consolidation Company. Acme Consolidation Company uses the WSDL file to generate the Web service client proxy that will invoke the Shipping Web service.

Handling Business Exceptions

The previous scenario walkthrough of the product shipping solution shows how the system works in the best case situation—when everything goes as planned. However, the automated solution must also handle other conditions when real-world events interfere with the desired process flow. For example, one of the significant business exceptions to handle is when the truck fails to arrive on time to pick up the goods. Figure 3 shows the components in the solution that detects the truck non-arrival exception. In the normal flow of events, after the trucker checks in with the shipping clerk at Northern Electronics, the transport order status is updated to Arrived. To detect for the truck non-arrival event, Sam implements a business exception monitoring service that periodically queries the Shipping database for the list of transport orders with the current status set to Ordered and where the stated truck arrival time windows have expired.

The business exception monitoring service raises a truck non-arrival event in the Windows event log for each of these exceptions. A Microsoft Operations Manager 2005 (MOM) management agent monitors for the event and reports the event to the MOM server. A set of MOM rules is configured to raise a management alert that will trigger an e-mail notification to be sent to the relevant business personnel to handle the exception.

More information about modeling and implementing this business exception can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

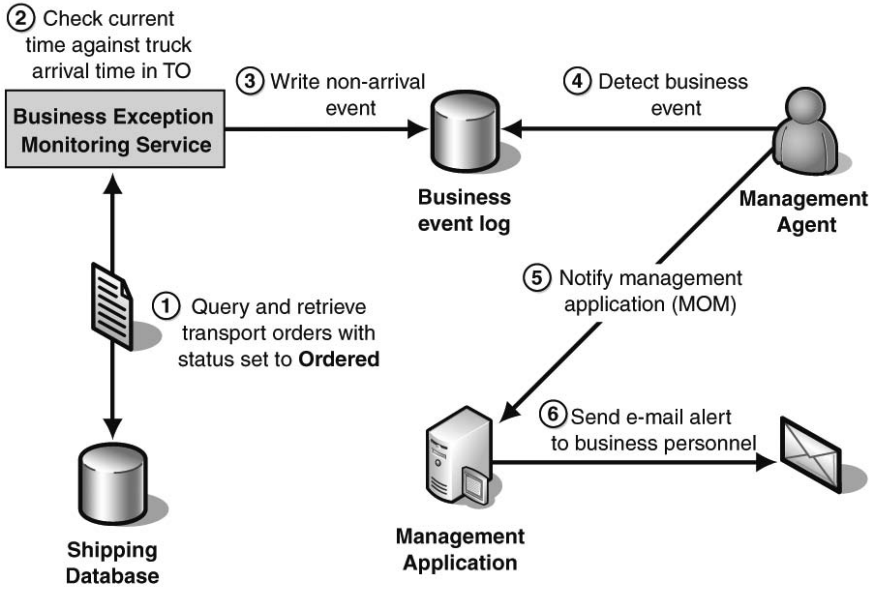


Figure 3. Detecting truck non-arrival event

Because of the unpredictability of the real world, it is often complicated and ineffective to try to completely model and automate the entire exception handling process. This statement definitely rings true for the preceding business exception. Therefore, after Tom, Jackie, and Pam evaluate the cost and benefits of automating the truck non-arrival exception handling process, they decide that after the appropriate individuals Figure 4 shows what happens after the e-mail notification is sent.

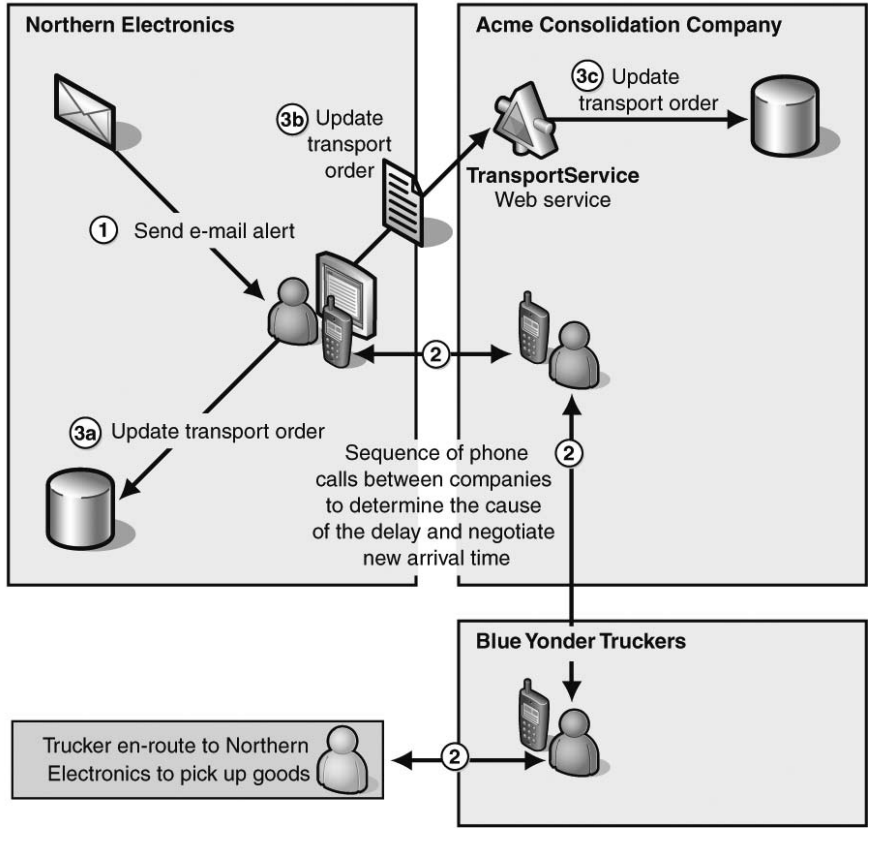


Figure 4. Handling truck non-arrival exception

The manual exception handling step begins after the shipping clerk receives the e-mail notification that states the identifier of the transport order violating the service agreement. The clerk uses the transport ordering application to retrieve the transport order and calls the transport ordering service department at Acme Consolidation Company. At this point, a series of phone conversations take place between business personnel in Northern Electronics, Acme Consolidation Company, and Blue Yonder Truckers, as well as with the trucker en-route to Northern Electronics to determine the cause of the delay (for instance, severe weather or poor road conditions) and to negotiate a new arrival time for the truck. Eventually, the shipping clerk at Northern Electronics records the outcome of the phone conversations and updates the TransportOrder document with the new arrival information. The updated transport order record is stored in the Shipping database and a copy of the document is also sent to the Acme Consolidation Company's TransportService Web service so that both parties have the new agreement in their systems' records.

Conclusion

This document described the application solution for enabling the automated transportation ordering process. It showed how the individuals involved use the different software components in the solution to realize the normal operation and exception flows in the business process. It showed how the solution architect used model-driven design and a service orientation approach to develop a flexible solution that was interoperable with business partners in a platform-neutral way, making use of Web services described using WSDL to communicate via messages specified and described using XML Schema. For the truck non-arrival event, the solution architect implemented a monitoring service that watches for situations where trucks fail to arrive at the loading dock at the designated time. An important point to note is that the solution seeks to provide real-time notifications of business operations conditions that need attention without attempting to automate the rectification actions which can be unduly complicated and costly to implement.

Additional Resources

Service Orientation and Its Role in Your Connected System Strategy ("srorientwp") on the MSDN Web site.

Notes

Web Service Health Modeling, Instrumentation, and Monitoring: Developing and Using a Web Services Health Model for the Northern Electronics Scenario

Architecture Chronicles
Dynamic Modeling: Aligning Business and IT

Frederick Chong with Jim Clark, Max Morris, and Dave Welsh
September 2005

Applies to:

- Enterprise Architecture
- Solution Architecture
- Service Oriented Architecture (SOA)
- Service Oriented Management (SOM)
- Application Integration
- Business Process
- Business Operations Modeling

Summary

The *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* seek to present to business, solution, and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. Focusing on health modeling, instrumentation, and monitoring in the Northern Electronics scenario, this document examines how the solution architect applies a health modeling approach to Web services in the product shipping solution to derive a set of requirements for instrumentation and monitoring.

This Document

This document focuses on health modeling, instrumentation, and monitoring Web services in the Northern Electronics product shipping solution. Background information on the Northern Electronics scenario can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

This document examines how the solution architect applies a health modeling approach to the Web services in the product shipping solution to derive a set of requirements for instrumentation and monitoring. It also shows how the rest of the IT organization uses the current generation of management tools to put the health model into action.

Abstract

This document describes how Northern Electronics uses a health modeling approach to design, implement, and operate manageable Web services in the product shipping solution. The health modeling approach encompasses the process steps of detection,

verification, diagnosis, recovery, and re-verification. The document shows how a solution architect translates a set of service requirements into management policies to enforce the required service levels. The document then shows how the development team uses the health model to instrument the software using Microsoft Visual Studio .NET, and how the application analyst and operations staff monitor the application using Microsoft Operations Manager 2005.

Acknowledgments

Many thanks to Nelly Delgado for her help with technical writing, Claudette Siroky for her graphics skills, and Tina Burden McGrayne for her copy editing.

The authors would also like to thank Gajanan Phadke, Vishal Kapoor, Amol Wankhede, Aziz Matheranwala, Amit Kumar Srivastav, Bhushan Pawade, Bhasha Johari, Avadh Jain, Mughda Gadre, Maneesha Nalawade, Sonika Arora, Anil Sharma, and Anurag Katre (all of Tata Consulting Services) for their contributions to the implementation of the Northern Electronics solution.

Introduction

This document focuses on health modeling, instrumentation, and monitoring the Web services in Northern Electronics's product shipping solution. Background information on the Northern Electronics scenario can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

This document examines how the solution architect applies a health modeling approach to the Web services in the product shipping solution to derive a set of requirements for instrumentation and monitoring. It also shows how to the rest of the IT organization uses the current generation of management tools to put the health model into action.

Specifically, this document discusses:

- Health modeling concepts.
- Business, application, and system operations requirements for the product shipping solution.
- Capturing the requirements in the form of health models.

Implementing the health models by instrumenting performance counters, events, and setting up the monitoring infrastructure. This includes discussion about:

- Using the **system.management.instrumentation** and **system.management.diagnostics** .NET namespaces:
- Creating a Microsoft Operations Manager (MOM) Management Pack for the ShippingService Web service using the Web Sites and Services MOM Management Pack (WSS MP).

Health Modeling

Because there is no such thing as a perfect environment, an application will experience problems at some time during its execution. If a problem is not detected in a timely fashion, not diagnosed correctly, or not fixed properly, the application can degrade to the point where it is no longer available to its customers. This is expensive, time consuming, and ultimately creates dissatisfaction among users. It is up to the administrators to detect and fix issues with as little downtime for their users as possible.

A **health model** defines what it means for a system to be healthy (operating within normal conditions) or unhealthy (failed or degraded) and the transitions in and out of such states. Good information on a system's health is necessary for the maintenance and diagnosis of running systems. The contents of the health model become the basis for system events and instrumentation on which monitoring and automated recovery is built.

To keep an application up and running, the operations team need to watch the application's health metrics, detect symptoms of a problem early, correctly diagnose the cause of that problem, and fix the problem before the application performs unacceptably. This activity is referred to as health monitoring and troubleshooting.

To create a health model, the modeler needs to do the following:

- Collect the right information about the application at the right time—when running normally or when something fails.
- Document all management instrumentation exposed by an application or service.
- Document all service health states and transitions that the application can experience when running.
- Identify the steps and determine the instrumentation (events, traces, performance counters, and Windows Management Instrumentation [WMI] objects/probes) necessary to detect, verify, diagnose, and recover from bad or degraded health states.
- Document all dependencies, diagnostics steps, and possible recovery actions.
- Identify which conditions will require intervention from an administrator.
- Improve the model over time by incorporating feedback from customers, product support, and testing resources.

Basic Concepts

This section provides the background information for describing and understanding the health model concepts. Figure 1 provides a partial service life cycle view of when health models can be effective in specifying the manageability requirements. This high-level health model provides the fundamental health modeling framework that can be further extended and analyzed to identify the potential problems, the indicators of the problems, the diagnostics steps, and the recovery actions that need to be taken to ensure that the application's health and performance meets expectations.

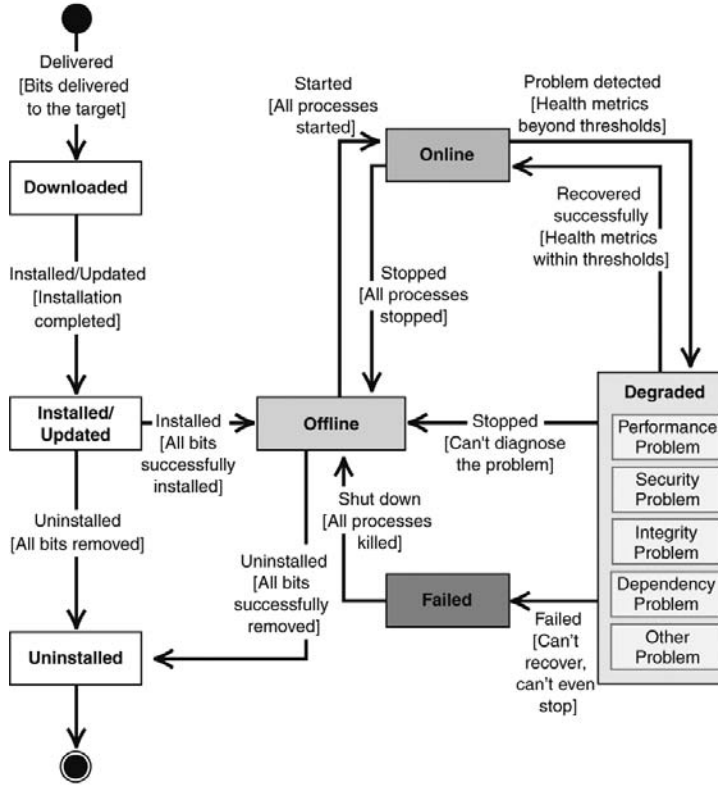


Figure 1. Service life cycle from a health model perspective

Health states are high-level indicators of the application's ability to function correctly. A health state provides non-quantitative data, but it tells the administrator if the application is in trouble and gives an idea of the severity of the situation. It is important to remember that a health state is a judgment call about the severity and is almost always relative. For example, one customer may decide that a total network saturation level of 90 percent is a degraded state, while another customer would consider a level of 80 percent to be in a failed state.

The service life cycle view of the health model in Figure 1 defines four generic health states of an application:

- Offline (depicted by a black-colored state). The application is in a healthy stopped state ready to be started normally. In other words, the application is not available and is in a dormant state but not because of any unexpected problems. The "Initializing" and "Stopping" phases of the application life cycle can also be regarded as offline because the application cannot provide service. This is expected behavior.
- Online (depicted by a green-colored state). The application is up, running, and performing as expected. The various operational conditions in this state are:
  - Processing. Service is busy processing a request.
  - Idle. Service idle and waiting for a new request.
  - Stopped. Service is online but has been currently put under maintenance mode by the operators. It is not serving any more requests but will start doing so once it is out of maintenance mode.
- Degraded (depicted by a yellow-colored state). The application is unable to provide a subset of its functionality, or is providing service at an unacceptable level. Problems have been detected and chances are it is possible to diagnose and recover from the situation. Although not fully functional, the application can still perform at some level. The various operational conditions in this state are:
  - Overloaded. The service is working but it is responding very slowly. Its performance has degraded and it may make a transition to either the processing, then idle, operational condition or to stopped, then failed. If further calls to it were diverted for sometime, it could come back to a healthy state (processing or idle operational condition) very soon
  - Not responding. The service is alive but not responding. If further calls to it were diverted for sometime, it could come back to a healthy state (processing or idle operational condition) in sometime but it will not need a reboot to become healthy again.

Note: The preceding two types of performance problems can occur due to numerous reasons such as bugs in the code that cause long or unnecessary loops; insufficient memory or other system resources; competition with other applications for resources; memory or resource leaks; denial of service attacks; or simply failing to load-balance properly across the pieces of the application.

  - Dependency problems. Such types of problems occur when a component on which the service is dependent for its healthy operation fails. For example, such problems would arise (a) if one of the backend databases with which the service communicates is down; or (b) a Windows service responsible for processing the input data of the service failures.
  - Security problems. Such problems can occur when the security of the service or the security of the overall system gets compromised. Detecting these types of intrusions early and correcting them quickly is critical. Otherwise, they can ultimately cause performance and integrity problems in the service.
  - Integrity problems. Such problems arise when the runtime stability of the application's state is violated. This may occur when inconsistencies arise in internal data structures, in corrupted stored data, or in missing data files/code. Usually, the application cannot proceed in such conditions and requires being stopped or being taken off-line for recovery.
  - Other problems. Other problems that do not fit in any of the preceding categories may also occur.
- Failed (depicted by a red-colored state). The performance of an application as a whole has degraded to the level where it no longer can be called functional or it is providing none of the services it is supposed to. This does not necessarily mean the application or a dependent application is stopped. The application itself and the external monitoring agent may have tried to diagnose and recover from the problem. But they have failed, and administrative action is now required.

The following terms are useful to understand in the health model discussion:

- Health. The health of the service at an instant refers to the performance of that system under different, defined conditions.
- Transition. A transition is the passage of an application or system from one state to another resulting in changed status.
- Indicator. An indicator provides information that helps detect the state of the application. An event that occurs and a threshold defined for a performance counter are two examples of indicators.

- Monitor. A monitor is the entity that observes the indicators, applies a set of monitoring rules to the indicators, and based on the results, invokes the control action. For example, if the database goes down, an application event is sent to the monitor. The monitor recognizes the type and severity of the event that it has received, automatically restarts the database server, and raises an alert to the application administrator.
- Event. An event is a defined occurrence in the system or in an application. Events are the criteria defined for checking various possible conditions for controlling the services. Events could be an occurrence of any condition of interest, such as an application heartbeat (this indicates whether the Web service is operational), a business operations condition (such as a shipping date in the shipping scenario), a database not available condition, and a .NET exception.
- Performance counter. A performance counter is used for statistical monitoring of certain conditions. A performance counter can be a system counter generated by the system or a custom counter defined by the application. A counter may be used to determine Web service requests per second, memory usage, and processor time.

Health and Diagnosability Workflow

The health and diagnosability workflow as shown in Figure 2 defines logical stages of the monitoring and problem recovery process. The stages of this process are:

- Detection
- Verification
- Diagnostics
- Resolution
- Re-verification

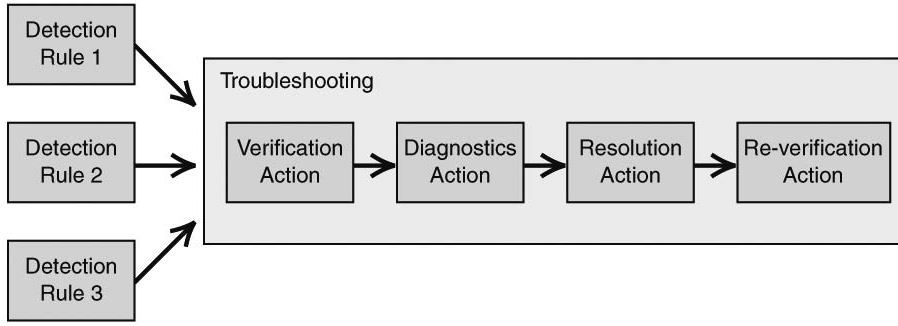


Figure 2. Health and diagnosability workflow

The following describes each stage:

- Detection. Monitoring is a continuous process that attempts to ensure that any problem with the application is detected as early as possible. Typically there are multiple ways to detect a problem. To detect a problem, the monitoring agent can:
  - Listen for events related to the health of the application.
  - Poll and compare performance counters against the specified thresholds as the basis to detect a problem.
  - Scan trace logs for information used to detect a problem.

  When the defined problem signatures are detected, a problem associated with the operational condition and health state is indicated. Until the condition is verified, the associated health state is not updated and diagnosis and recovery steps should not be attempted.
- Verification. After a problem is detected, it is necessary to verify that it actually still exists. This step is critical because detection logic is usually very lightweight and simply reports the first symptoms. Verification is basic confirmation that the application is in a particular operational condition without trying to diagnose why or to recover from it. The purpose of the verification step is to make sure the problem was not simply a surge in resource consumption, spike in workload, or simply a transient issue that has gone away. Verification is a procedure that takes an operational condition as an input and returns true or false.

- Diagnostics. After the operational condition has been confirmed, it is necessary to find out what is actually causing the problem to a point where a matching resolution can be applied. The context of the problem is usually the best place to start when investigating the cause. The root cause must be found out before the path to recovery can be mapped out. For example, while the condition may be that network connectivity is down, the path toward resolving the problem is not clear until it's known that the cause is a loss of the IP address lease from Dynamic Host Configuration Protocol (DHCP).

  The diagnostics step uses all forms of available instrumentation such as events, performance counters, WMI providers, and activity traces to correlate information and determine the root cause. While the verification step simply confirms the operational condition and is relatively lightweight, the diagnostics step can take a long time and further disrupt service while it is happening. It can be necessary to inspect a much broader set of internal state parameters and correlate between applications to perform the diagnostics step.
- Resolution. After the root cause is determined, the next step is to try resolving the problem. This process can involve reconfiguration, restarting the application, manipulating internal state through WMI, or performing administrative tasks.
- Re-verification. The same verification procedure that was used to verify the existence of the operational condition is used to re-verify that the operational condition has indeed been corrected. When the issue is successfully resolved, this returns FALSE.

Instrumentation

To be effectively monitored, an application needs to provide the right types and levels of instrumentation—it needs to expose its internal state, report changes in its state and behavior, and supply methods for administrative control. This provides the data essential for problem detection, verifications, diagnosis, and recovery. Additionally, the instrumentation needs to be lightweight and have no noticeable effect on an application's performance. There are three primary types of lightweight instrumentation that can be used for different aspects to support the health and diagnosability workflow:

- Events. An application sends out an event when the service encounters a particular operational condition or a failure state.
- Traces. Applications log trace information when a certain checkpoint in the code is passed. Tracing is akin to a running commentary on what is going on in the code. Tracing can be configured for different levels and switched on and off at run time to provide both diagnostic and troubleshooting information. Developers of the application and product support use traces mostly for offline debugging and troubleshooting as the amount of information is often too large to be constantly analyzed at run time.

- Performance counters. Performance counters are numeric values that reflect the state of the application at a precise moment or averaged over a period of time. As the name suggests, performance counters usually represent some performance-related metric such as the number of resources of a particular type used or available (of handles), rate of requests (Lightweight Directory Access Protocol [LDAP] or HTTP requests/sec), micro-events (page faults/sec), and other tracking numbers (disk queue length, percent CPU).

For more information about topics surrounding health modeling, see Health Modeling: A Key Step to DSI-Enabled Applications (http://www.microsoft.com/windowsserversystem/dsi/designwp.mspx).

Who's Who in Northern Electronics

The following individuals from Northern Electronics are involved in developing, implementing, and operating the health model:

- Tom. He is the solution architect whose responsibility in this context is to define the health model and suggest solutions for implementing the health model.
- Sam. He is the lead developer whose responsibility in this context is to implement the performance counters and events.
- Dan. He is the application administrator whose responsibility in this context is to implement the monitoring rules.

Supporting Multiple Requirements

The requirements at the business, application, and system levels drive the creation of the health model. Tom lists the requirements and any background information about the requirements.

Note: The purpose of this document is to demonstrate a health model that addresses a few key issues. The requirements specified in this section and the corresponding health model are not exhaustive and are only intended for illustrative purposes. For example, this document does not address activities such as using traces for diagnostics, resetting the complete system, handling unknown states, and handling multiple concurrent unhealthy states. However, these are important activities in any operations plan and should be taken into consideration.

Business Operations Requirements

Business operations requirements are important to monitor. Namely, problems with the operation of the application that may not obviously degrade application health but can affect the health of business operations can occur directly within the application. Two such business operations requirements Northern Electronics monitors are shown in Table 1.

Table 1. Business Operations Requirements

Name	Requirement Description	Background
Business Operations Requirement 1 (BOR1)	The SuggestedPickupDate entered by the transport consolidator in the TO document shall be on or before the ExpectedShippingDate entered by the supplier.	Northern Electronics has agreements with customers such as Wingtip Toys to ship the goods on a particular date. Northern Electronics prepares a Transport Order (TO) document specifying the date the goods will be shipped. In reality, partners such as Acme Consolidation Company are responsible for shipping the goods. If the date that Acme Consolidation Company agrees to ship the goods is later than the date that Northern Electronics promises Wingtip Toys that their product will ship, there is a violation of the Service Level Agreement (SLA). The instrumented code will compare dates and if there is a violation, raise an event that alerts the business analyst at Northern Electronics of the SLA violation.
Business Operations Requirement 2 (BOR2)	The truck shall arrive to pick up the goods at the supplier's warehouse within the specified SuggestedPickupDate time window.	Partners such as Acme Consolidation Company send trucks to pick up the goods that need to be shipped. Acme Consolidation Company agrees in its SLA for Northern Electronics that its trucks will arrive by the SuggestedPickupDate. A process running in the background will check that trucks arrive within the timeframe of the suggested time. An event is raised if the trucker fails to arrive within this time. When the event is raised, the business analyst is made aware that the particular TO is delayed due to the delayed arrival of a particular trucker and that there is a violation of the SLA.

Application Requirements

Application requirements specify issues related to application performance, availability, and response to application-generated exceptions. Table 2 lists some of the application requirements that Northern Electronics monitors.

Table 2. Application Requirements

Name	Requirement Description	Background
Application Requirement 1 (AR1)	The business methods of the ShippingService Web service shall maintain high performance levels to incoming requests. To achieve this, the total processing time taken by the Web service to respond to a request is calculated on a per request basis and is sampled periodically. If any of the sampled values are greater than 50 ms, the Web service performance has deteriorated. As a result, an alert is generated and the appropriate individuals are notified.	To stay in business in this competitive market, Northern Electronics must ensure that it ship its products to their customers in a timely manner. A performance counter can be used to monitor the performance levels of the business methods of the ShippingService Web service. If the processing time of the ShippingService increases above a specified threshold, it indicates that the performance has deteriorated below an acceptable level and the business analysts are notified. Additionally, the operations staff may want to monitor the counter to make sure that the ShippingService is performing up to expectations. They may take some corrective action if it is in a degraded state (such as deploying multiple instances of the service to load balance, upgrading the hardware, or changing the deployment configuration for increasing performance).
Application Requirement 2 (AR2)	The ShippingService Web service shall be available 99.9 percent of the time. To monitor the heartbeat of the Web service on a continuous basis, heartbeat requests are periodically sent to the Web service. These requests are sent to some pre-specified dummy methods of the Web service. If the ShippingService Web service does not respond, an alert is raised and an e-mail notification is sent to the appropriate individuals.	The ShippingService Web service must be available to the transport consolidators so that transport arrangements can be made at any time of the day.

Name	Requirement Description	Background
Application Requirement 3 (AR3)	The Shipping database and the Transport database shall be available at all times. If the database is not available, an event is raised and this event is logged in the Windows event log.	The ShippingService Web service depends on these databases to initiate and complete the transport order.
Application Requirement 4 (AR4)	There shall be fewer than twenty messages in the message queue over an average of five samples taken at the rate of one sample every twenty seconds.	A message queue receives transport orders from the ShippingService Web service and a Windows service picks up the transport orders from the message queue, processes the orders and saves the orders to the database. Northern Electronics needs to make sure that both the Windows service and the message queue are healthy. The goal is to process 20 messages in 20 seconds (one sample per second).
Application Requirement 5 (AR5)	If the Web service raises a SOAP exception, the event is logged, an alert is raised, and the appropriate individuals receive notification of the exception.	In case there is an error processing the request on the transport consolidator side, Northern Electronics needs to know that the request did not complete successfully.
Application Requirement 6 (AR6)	If a null reference exception occurs, the event is logged, an alert is raised, and the appropriate individuals receive notification of the exception.	A null reference exception may occur if one of the consumers of the ShippingService has code that incorrectly sets the PickupNotification document to NULL and calls the NotifyPickup Web method by passing the NULL document as its input; or if a malicious attacker tampers with the ShippingService Web service by sending NULL inputs.
Application Requirement 7 (AR7)	If any general .NET exception occurs in the ShippingService Web service, the event is logged, an alert is raised, and the appropriate individuals receive notification of the exception.	Any .NET exception that occurs in the ShippingService Web service may indicate a code implementation issue that needs to be taken care of by the developer.

System Requirements

System requirements specify issues related to the performance of the Web server at the hardware and infrastructure level. If the Web server is not performing as expected, the Web service health will also be affected. The system requirements that Northern Electronics monitors are shown in Table 3.

Table 3. System Requirements

Name	Requirement Description	Background
System Requirement 1 (SR1)	CPU utilization at the sampled interval of time shall not be greater than 90 percent at the rate of one sample per minute.	If the Web server is not performing as expected at the system level, the Web service health will also be directly affected due to the CPU availability. Windows exposes performance counters that can be used to monitor the health of the Web server (instead of that of the Web service) by identifying the CPU load.
System Requirement 2 (SR2)	Total available system memory at the sampled interval of time shall not be less than 15 percent at the rate of one sample per minute.	If the Web server is not performing as expected, the Web service health will also be affected due to the availability of memory. A counter can monitor the health of the Web server (rather than that of the Web service) by identifying the memory load on the Web server.

Capturing Requirements in a Health Model

Tom builds the ShippingService health model keeping in mind the manageability requirements of the various users and the abstract health model. Tom breaks down the overall service health model into three sub-models. The three models are:

- Business operations health model.
- Application health model.
- System health model.

Each of these is logically distinct from the others. This section shows how the high-level health models for three specific types requirements in the business operations, application, and system categories can be represented graphically. Following each graphical representation, we also show how information for the health and diagnosability workflow can be filled in to derive the detailed health models used for managing the ShippingService Web service.

Business Operations Health Model

The business operations health model is concerned with capturing the concerns of the SLAs the solution has to adhere to and is of interest to the various business operations users of the system. The business operations health model is derived from the high-level business operations specifications developed by the business operations team at Northern Electronics through a business operations modeling exercise.

More detailed information about business operations modeling and communicating business operations requirements to IT can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Tom needs to take into account the requirements in the business operations specifications. After considering and discussing the requirements expressed in the specification with the business operations analysts, Tom derives the state representation of the business operations health model as shown in Figure 3.

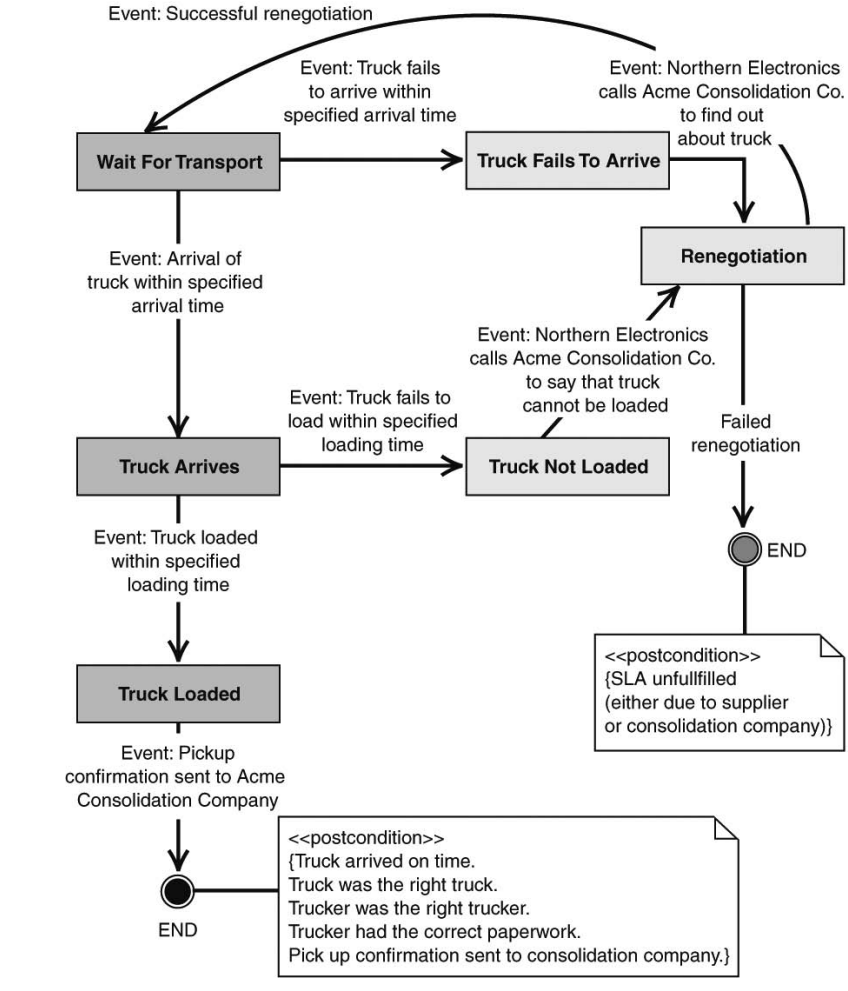


Figure 3. Business operations health model

Business Operations Health and Diagnosability Workflow

Drawing the states in the business operations health model helps Tom think about the events that need to be instrumented and the resolutions to degraded states. Tom is then able to dive into a more detailed specification of the detection, verification, diagnostics, resolution, and re-verification actions for each problem identified at the business operations level. Some of the business operations requirements are:

- Business Operations Requirement 1 (BOR1). The SuggestedPickupDate entered by the transport consolidator in the TO document shall be on or before the ExpectedShippingDate entered by the supplier.
- Business Operations Requirement 2 (BOR2). The truck shall arrive to pick up the goods at the supplier's warehouse on or before the specified SuggestedPickupDate.

Tables 4 and 5 capture the health and diagnosability workflow for the business operations health model.

Table 4. Business Operations Health Model–Detection Information for Step 1 of Health and Diagnosability Workflow

Problem	Health State and Operational Condition	Indicators	Operational Alerts	Criticality
Violation of BOR1	Degraded; Integrity Problem	Windows Event (ID = EVENT_TO_SUGGESTED_DATE_MISMATCH) This event is logged in WMI	An e-mail notification will be sent to the Business Operations Managers notification group.	An event processing rule generates an alert with the severity set to Severe.
Violation of BOR2	Degraded; Integrity Problem	Windows Event (ID = EVENT_TRUCK_ARRIVAL_DELAYED) This event is logged to the event log.	An e-mail notification will be sent to the Business Operations Managers notification group.	An event processing rule generates an alert with the severity set to Severe.

The resolution in all of the cases is to provide feedback to the business operations managers.

Table 5. Business Operations Health Model–Verification, Diagnostic, Resolution, and Re-verification Information for Steps 2-5 of Health and Diagnosability Workflow

Problem	Verification Procedure	Diagnostic Information	Resolution and Final Desired State	Re-verification
Violation of BOR1	Poll the TO and PickupNotification table in database. Check that the SuggestedPickup Date (in TO) is greater than the-Expected Shipping Date in the Pickup Notification Table (should return TRUE). Operational condition confirmed as integrity problem.	Manual check. Call to Transport Consolidator to verify issue and reconfirm truck arrival date.	Manual resolution. Supplier must updateSuggested-PickupDate in the PickupNotification table.	Manual check.
Violation of BOR2	Manual check that truck has not arrived (should return TRUE). Operational condition confirmed as integrity problem.	Manual check. Call to Transport Consolidator to find out where truck is.	Manual resolution. Supplier renegotiates truck arrival time with consolidation company and updates the time in the system. Supplier updates SuggestedPickup Time. The logs and databases are both updated with new information.	Manual check that truck has not arrived (should return FALSE).

Application Health Model

Application health addresses management of the runtime health of the service. This involves monitoring the application-related parameters of the service such as the current requests that the Web service is processing, average request processing time, number of currently logged on users, number of exceptions that have occurred in the service, a crash of any sub-component of the service.

The example requirement (Application Requirement 3) that the Shipping database and the Transport database should be available is modeled in the application health model as shown in Figure 4.

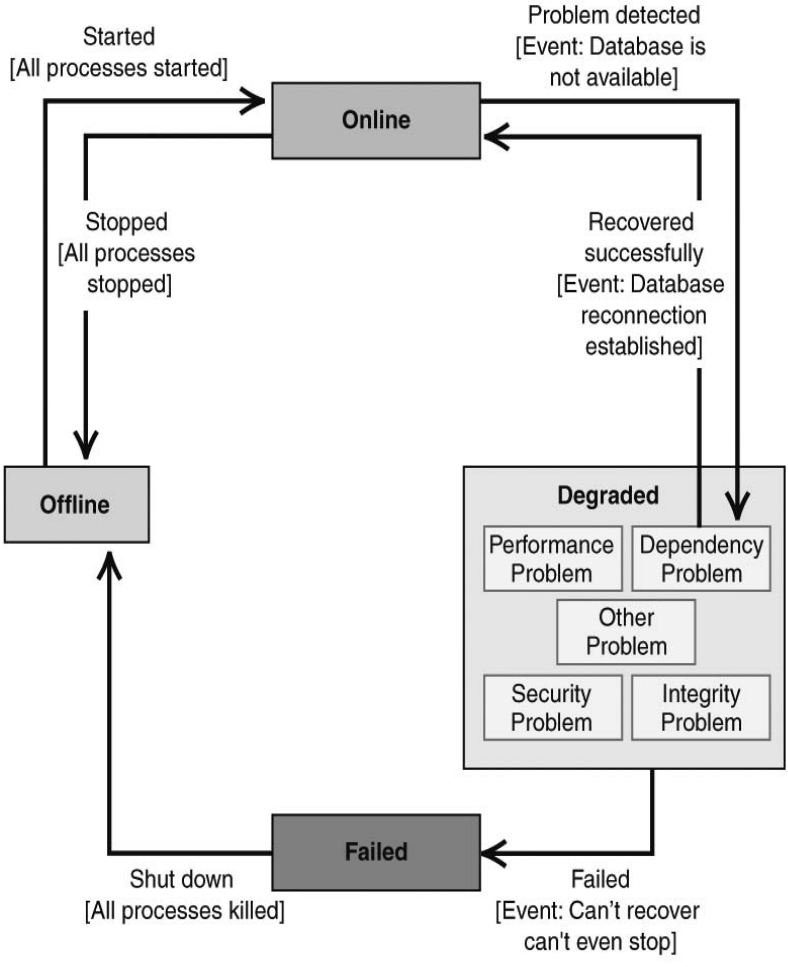


Figure 4. Application health model

Application Health and Diagnosability Workflow

The requirements at the application level are:

- Application Requirement 1 (AR1). The business methods of the ShippingService Web service shall maintain high performance levels to incoming requests. To achieve this, the total processing time taken by the Web service to respond to a request is calculated on a per request basis and is sampled periodically. If any of the sampled values are greater than 50 ms, the Web service performance has deteriorated.
- Application Requirement 2 (AR2). The ShippingService Web service shall be available 99.9 percent of the time. To monitor the heartbeat of the Web service on a continuous basis, heartbeat requests are periodically sent to the Web service. These requests are sent to some pre-specified dummy methods of the Web service.
- Application Requirement 3 (AR3). The Shipping database and the Transport database shall be available at all times.
- Application Requirement 4 (AR4). There shall be fewer than twenty messages present in the message queue over an average of five samples taken at the rate of one sample every twenty seconds.
- Application Requirement 5 (AR5). If the Web service raises a SOAP Exception, the event is logged and an alert is raised.
- Application Requirement 6 (AR6). If a null reference exception occurs, the event is logged and an alert is raised.
- Application Requirement 7 (AR7). If any general .NET exception occurs in the ShippingService Web service, the event is logged, an alert is raised, and the appropriate individuals receive notification of the exception.

Tables 6 and 7 capture the health and diagnosability workflow for the application requirements monitored by MOM.

Table 6. Application Health Model–Detection Information for Step 1 of Health and Diagnosability Workflow

Problem	Health State and Operational Condition	Indicators	Operational Alerts	Criticality
Violation of AR1	Degraded; Overloaded or Not Responding	Application level Performance Counter Name = Document Processing Time; Threshold: > 50ms (Indicates the processing time taken by the NotifyPickup Web method of ShippingService to process a Pickup Notification) Sampling at regular intervals; Sampled value greater than 50ms.	An e-mail notification will be sent to the MOM operators added to the Operations notification group.	A threshold rule generates an alert with the severity set to Critical Error if the specified threshold value is reached.
Violation of AR2	Degraded; Overloaded or Not Responding	Event ID = EVENT_HEART_BEAT_MISSING	An e-mail notification will be sent to the Operations notification group	An event processing rule generates an alert with the severity set to Error.
Violation of AR3	Degraded; Dependency Problem	Event ID = EVENT_DATABASE_NOT_AVAILABLE	An e-mail notification will be sent to the MOM operators added to the Operations notification group	An event processing rule generates an alert with the severity set to Error.
Violation of AR4	Degraded; Dependency Problem	Counter Name = MSMQ Queue Length; Threshold: > 20	An e-mail notification will be sent to the MOM operators added to the Operations notification group informing him that the queue is overloaded	A threshold rule generates an alert with the severity set to Critical Error if the specified threshold value is reached.

Problem	Health State and Operational Condition	Indicators	Operational Alerts	Criticality
Violation of AR5	Degraded; Dependency Problem	Event ID = EVENT_SOAP_EXCEPTION	An e-mail notification will be sent to the MOM operators added to the Operations notification group	An event processing rule generates an alert with the severity set to Error.
Violation of AR6	Degraded; Integrity Problem	Event ID = EVENT_NULL_REFERENCE	An e-mail notification will be sent to the MOM operators added to the Operations notification group	An event processing rule generates an alert with the severity set to Error.
Violation of AR7	Degraded; Other problem (general)	Event ID = EVENT_DOTNET_EXCEPTION	An e-mail notification will be sent to the MOM operators added to the Operations notification group	An event processing rule generates an alert with the severity set to Error.

Table 7. Application Health Model–Verification, Diagnostic, Resolution, and Re-verification Information for Steps 2-5 of Health and Diagnosability Workflow

Problem	Verification Procedure	Diagnostic Information	Resolution	Re-verification
Violation of AR1	Check event log; verify value of performance counter (should return TRUE). Condition confirmed as overloaded or not responding.	1. Look at specific transaction that caused the problem; looking at stack traces; code dumps; logs. 2. Check CPU and memory.	If the performance counter goes down, the problem is resolved. If performance counter is still at a high level, restart the application.	Re-check performance counter value (should return FALSE).
Violation of AR2	1. Check that the URL of ShippingService is alive by making an HTTP request to it. 2. Check if the ShippingService itself is alive at the SOAP level by making a Web method call to a predefined Web method called SelfTest (should return FALSE). Condition confirmed as overloaded or not responding.	1. Look in event log for any other events (CPU may be overloaded). 2. Check system level indicators; CPU and memory. 3. Check status of Web server.	If no response to heartbeat, restart Web service.	1. Check that the URL of ShippingService is alive by making an HTTP request to it. 2. Check if the ShippingService itself is alive at the SOAP level by making a Web method call to a predefined Web method called SelfTest (should return TRUE).
Violation of AR3	Go to database and check database connection to server status (should return FALSE). Condition confirmed as dependency problem.	1. Look in event log for database server problems. 2. Check database configuration settings in configuration files.	1. Restart database. 2. Fix configuration files.	Go to database and check database connection to server status (should return TRUE).

Problem	Verification Procedure	Diagnostic Information	Resolution	Re-verification
Violation of AR4	Look at the number of outstanding requests in the message queue. Verify that there are more than twenty message (should return FALSE). Condition confirmed as dependency problem.	1. Look at traces of Message Queuing (also known as MSMQ). 2. Look at log of Message Queuing. 3. Check the state of service. 4. Check event logs. 5. Check stack traces.	1. If Message Queuing is not responding, restart it manually. 2. If Windows service is not responding, restart the unhealthy service.	Look at number of outstanding requests in message queue. Verify that there are more than twenty message (should return TRUE).
Violation of AR5	Resend document to see if another SOAP exception occurs (should return TRUE). Condition confirmed as dependency problem.	1. Check event log. 2. Check your document (your own Web service). Figure out why Web service is causing problem. Look at application event logs for application errors (for example, see AR 5). 3. Call Transport Consolidator to see if there are problems with their Web service. This event is logged in the event log.	1. If our Web service caused problems, shut down Web service, rebuild, and redeploy Web service (see AR5). 2. Resolution depends on Transport Consolidator to fix Web service.	Resend document to transport consolidator and see if you get a SOAP exception (should return FALSE).

Problem	Verification Procedure	Diagnostic Information	Resolution	Re-verification
Violation of AR6	Look in application log for null reference exception (should return TRUE). Condition confirmed as dependency problem.	Look in stack trace. Possible root causes: 1. Database string not set and causes a null exception. 2. Receive null document. 3. Security violation.	1. If security violation, deal with security issue. 2. Update configuration files. 3. Figure out who called the Web service with the Null document from an Internet Information Services (IIS) log. Potentially treat as a security incident.	Look in application log for null reference exception (should return FALSE).
Violation of AR7	Look in application log for .NET exception (should return TRUE). Condition confirmed as other problem (general).	Check for stack trace information in the log.	Use a stack trace to determine the cause of the .NET exception and decide if the application code or the operation environment needs to be debugged and fixed accordingly.	Look in application log for .NET exception (should return FALSE).

System Health Model

The system health model addresses the operational infrastructure on which the service is deployed. It takes into account runtime system parameters such as network load, CPU usage, memory usage, availability of system services and others. For example, the requirement to monitor the CPU load of the computer on which the Web service is deployed (**System Requirement 1**), is modeled as shown in Figure 5.

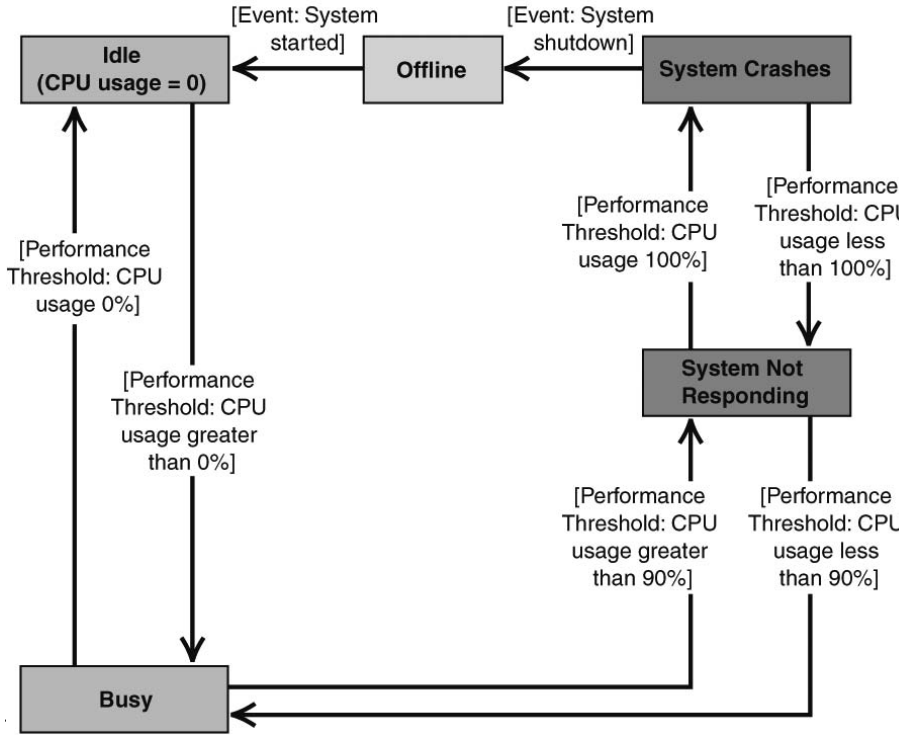


Figure 5. System health model

System Health and Diagnosability Workflow

Tables 8 and 9 capture the health and diagnosability workflow for the system requirements. The system requirements are:

- System Requirement 1 (SR1). CPU utilization at the sampled interval of time shall not be greater than 90 percent at the rate of one sample per minute.
- System Requirement 2 (SR2). Total available system memory at the sampled interval of time shall not be less than 15 percent at the rate of one sample per minute.

Table 8. System Health Model–Detection Information for Step 1 of Health and Diagnosability Workflow

Problem	Health State and Operational Condition	Indicators	Operational Alerts	Criticality
Violation of SR1	Degraded; not responding	Performance Counter = percent Processor Time; Threshold: > 90 percent	An e-mail notification will be sent to the MOM operators added to the Operations notification group.	A threshold rule generates an alert with severity set to Critical Error if the specified threshold value is reached.
Violation of SR2	Degraded; overloaded	Counter = Available System Memory ; Threshold < 15 percent	An e-mail notification will be sent to the MOM operators added to the Operations notification group	A threshold rule generates an alert with severity set to Critical Error if the specified threshold value is reached.

Table 9. System Health Model–Verification, Diagnostic, Resolution, and Re-verification Information for Steps 2-5 of Health and Diagnosability Workflow

Problem	Verification Procedure	Diagnostic Information	Resolution and Final Desired State	Re-verification
Violation of SR1	Open Performance Monitor to check that CPU utilization is greater than 90 percent (should return TRUE). Condition confirmed as not responding.	1. Open Task Manager to see which application is causing the problem. 2. If the problem is our own Web service, need to take Web service offline and perform debugging.	1. One potential resolution may be to redeploy the fixed Web service. Refer to guidance on CPU resolution. 2. If other applications, follow resolution for CPU usage problems for specific applications.	Open Performance Monitor to check that CPU utilization is greater than 90 percent (should return FALSE).
Violation of SR2	Open Performance Monitor to check that total available system memory is less than 15 percent (should return TRUE). Condition confirmed as overloaded.	1. Open Task Manager to see which applications are using the most memory. 2. If the problem persists, take the application offline and debug.	1. One potential resolution may be to redeploy the fixed Web service. Refer to guidance on memory resolution. 2. If other applications, follow resolution for memory usage problems for specific applications.	Open Performance Monitor to check that total available system memory is less than 15 percent (should return FALSE).

Implementing the Health Models

After reviewing the list of requirements in the tables representing the health model, Tom is able to identify where instrumentation should be implemented by the application code.

The following summarizes the application-specific instrumentation:

- Performance counters:
  - Document processing time
  - Message Queuing queue length

Application events:

- EVENT_TO_SUGGESTED_DATE_MISMATCH
- EVENT_TRUCK_ARRIVAL_DELAYED
- EVENT_HEART_BEAT_MISSING
- EVENT_DATABASE_NOT_AVAILABLE
- EVENT_SOAP_EXCEPTION
- EVENT_NULL_REFERENCE_EXCEPTION
- EVENT_DOTNET_EXCEPTION

This section illustrates how Sam uses Visual Studio .NET and the management features in the Microsoft .NET Framework to help integrate application-specific performance counters and events into the ShippingService Web service implementation. Specifically, the steps for implementing the **Document Processing Time** performance counter and the **EVENT_DATABASE_NOT_AVAILABLE** event are described.

Implementing Application Performance Counters

One of the performance goals is to ensure that the Web service maintains high performance levels to incoming requests. In this case, a good way to measure the throughput of the Web service is to measure how long it takes to finish processing each incoming pickup notification document that is received from Acme Consolidation Company.

To accomplish this task, Sam implements code in the ShippingService Web service to record the time taken to process each pick-up notification document. Instead of introducing and embedding complex performance measurement algorithms into the ShippingService Web service source code, Sam decides that the Web service should record only the latest document processing time. The monitor then collects the value of the performance counters and decides if it meets the conditions of the performance thresholds.

The performance counter feature in the .NET Framework provides an easy way for Sam to implement the code and takes care of making the recorded value available to MOM, which is the performance monitoring application that Northern Electronics is using to monitor and manage the Web services in the data center. The performance counter programming interfaces are defined in the **system.management.diagnostics** namespace in the .NET Framework. Performance counters are categorized according to the type of systems or application resource that the counter is providing performance data for. For example, existing performance counter categories such as "Process" and "Network Interfaces" are built into Windows and provide performance statistics such as "Total Handles" and "Bytes sent/sec" respectively.

To differentiate the custom performance counter from other existing performance counters, Sam creates a new category named "Northern Electronics Web Services" and adds a new counter named "Pickup Notification Processing Time" to this category. Sam creates the performance counter category and uses Visual Studio .NET to add the custom performance counter through the following code snippets:

```
PerformanceCounterCategory.Create( "Northern Electronics Web Services",
"Custom performance counters used by Northern Electronics Web Services",
"Document Processing Time", "Time taken to process the pickup
notification business document" );
```

Subsequently, the ShippingService Web service implements the following code to update the performance counter with the time taken to process the last pick-up notification document:

```
PerformanceCounter counter = new PerformanceCounter();
counter.CategoryName = "mySingleCategory";
counter.CounterName = "myCounter";
counter.ReadOnly = false;
counter.RawValue = CurrentProcessingTime;
counter.Close();
```

Implementing the Business Operations and Application Events

An event is a very useful mechanism to indicate that an activity or condition has occurred. The nature of the activities and conditions may be related to the functionality of the application or the business environment. For example, in the case of the ShippingService Web service, the **EVENT_DATABASE_NOT_AVAILABLE** application event is used to indicate when the database connection is lost and the ShippingService Web service is in a degraded state. On the other hand, the **EVENT_SUGGESTED_DATE_MISMATCH** business event is raised by the ShippingService Web service to indicate that a transport notification document that was received did not meet a business operations service level agreement between Northern Electronics and Acme Consolidation Company.

Events are usually recorded in the form of log entries so that the event details can be tracked, analyzed, and handled appropriately if the condition requires further actions. For instance, an event can help in detecting (and possibly troubleshooting) the cause of the lost database connection in order to remedy the ShippingService Web service.

Currently, enterprise-class applications may log application and business events to a variety of places, such as the Windows event log, SQL databases, and disk files. In the case of the ShippingService Web service, the following strategy is used to determine where the events should be logged to:

- Business operations events are logged to a SQL database and WMI repository. The SQL database provides a permanent backing store for Northern Electronics to track and analyze business operations service level agreements with its business partners using SQL tools. In addition, channeling the business operations events to the WMI repository allows MOM rules to be configured for monitoring the business operations events and sending alerts to the business operations managers who will handle the issue at the business operations level.
- Application events are logged to the Windows event log and MOM rules are configured to monitor the events and raise corresponding alerts.

However, the current set of APIs for writing to the Windows event log, to WMI, and to Microsoft SQL Server are all different, complicating both the developer's learning curve and implementation efforts to push events out to these different channels. This situation motivates Sam to search for a higher-level programming abstraction that can help simplify the developer's task of writing log entries.

With the Enterprise Library Logging and Instrumentation Application Block, developers can incorporate common logging and instrumentation functionality into their applications. Figure 6 shows the design of the Logging and Instrumentation Application Block.

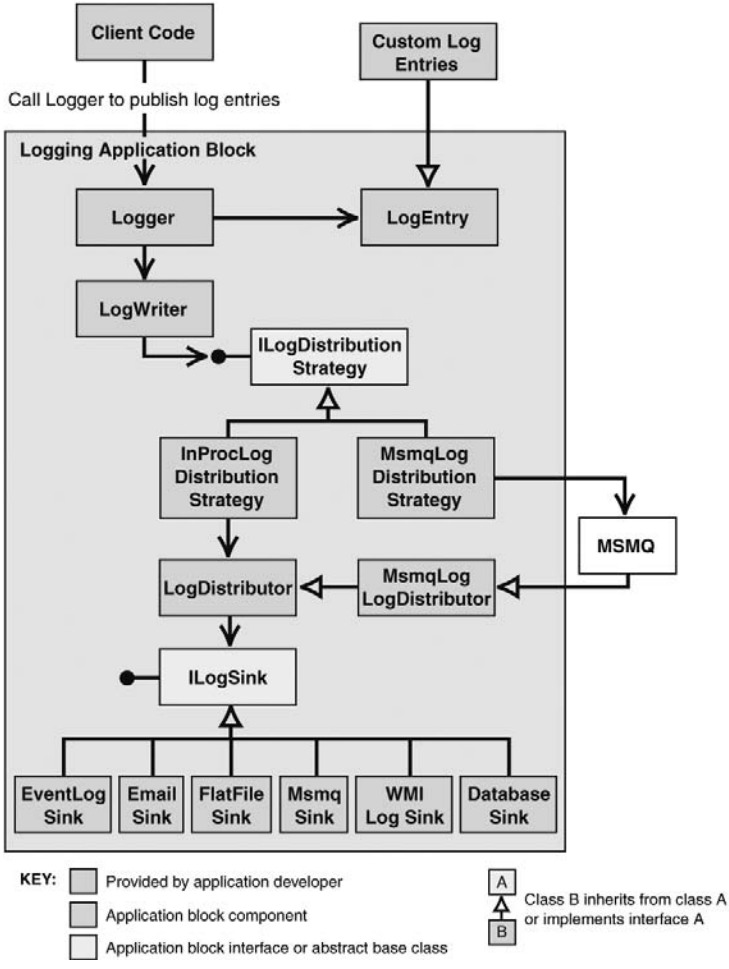


Figure 6. Design of the Logging and Instrumentation Application Block

Sam learns that he can use the Logging and Instrumentation Application Block to log events to a variety of locations including:

- The event log.
- E-mail notifications.
- A database.
- A message queue.
- A file.
- WMI.

The Logging and Instrumentation Application Block also allows:

- Filtering the events according to the category and priority of the events.
- Formatting the events into custom formats.
- Logging the instrumented events in a synchronous and asynchronous fashion.
- Extending and customizing the event sinks, formatters, and distribution strategies.

To satisfy the logging strategy previously outlined, Sam uses the Logging and Instrumentation Application Block to write events to the Event Log Sink, the Database Sink, and the WMI Log Sink.

At a high level, the Logging and Instrumentation Application Block provides a few advantages:

- It provides a simple programming-level abstraction over the disparate APIs that Sam would otherwise have to learn and master to deliver the log information to the various log depositories.
- It decouples and defers the types and locations of the log depository where the events are to be logged until the deployment of the Web service. The types and locations of the event destination are specified through configuration files that are set during deployment, instead of at development time.
- There is no need to rewrite or recompile the event instrumentation code in the Web service to change the type of destination of the event log.

The **Logger** is the main interface to the Logging and Instrumentation Application Block. It is a façade for writing a log entry to one or more log sinks. The Logger class accepts logging messages, determines the appropriate strategy for logging messages based on the configured distribution strategy, filters the messages based on category and priority, and hands them off to the distributor. A log entry is sent to the strategy only if the log entry passes both the priority and category filters. The components shown in Figure 7 are described in more detail as follows:

- LogEntry class. Represents a log message and contains the common properties that are required for all log messages.
- Logger class. A façade for writing a log entry to one or more log sinks.
- LogWriter class. An instance-based class to write log messages to a specific configuration context. Messages are routed based on category.
- ILogDistributionStrategy interface. Represents the interface for distribution strategies.
- LogDistributor class. Distributes a log message to the corresponding sinks as defined in the configuration file.
- InProcLogDistributionStrategy class. Processes the distribution of log messages synchronously by sending the log entry directly to the corresponding log sinks.
- ILogSink interface. Represents the interface for all logging sinks.
- Event Log sink. Writes the message to the application log of the local computer.
- WMI Log sink. Represents a WMI event that is fired by the WMILogSink class.

As previously mentioned, the task of writing log entries can be accomplished through the APIs provided by the **Logger** and **LogEntry** classes in the **Microsoft.Practices.EnterpriseLibrary.Logging** namespace. Sam uses Visual Studio .NET to add l**ogging to his application. For example, he writes the** EVENT_DATABASE_NOT_AVAILABLE application event to the Windows event log using a few lines of code shown in the following code snippets:

```
LogEntry objLog = new LogEntry();

objLog.Message = "Database connection not available";
objLog.Priority = 2;
objLog.EventId = 5001;
objLog.Severity = Severity.Warning;
objLog.Category = "Application";
objLog.Title = "Database Error";

Logger.Write(objLog);
```

Based on the category of the log entry, the database connection error is channeled to the Windows event log according to the following configuration section in the Logging and Instrumentation Application Block configuration file. In this case, it is the Windows event log:

```
<categories>
...
     <category name="Application">
          <destinations>
               <destination name="Event Log Destination" sink="Event Log
               Sink" format="Text Formatter" />
          </destinations>
     </category>
</categories>
```

For more information or to download the Logging and Instrumentation Application Block, see the Microsoft patterns & practices Enterprise Library ("entlib").

Monitoring

This section covers many aspects of monitoring including the conceptual architecture Tom designs that will address the monitoring requirements as well as the following monitoring-related topics:

- Heartbeat monitoring.
- Using Microsoft Operations Manager (MOM) for heartbeat monitoring.
- Using MOM for event and performance monitoring.
- Event monitoring.
- Performance counters monitoring.
- Handling management alerts.

Heartbeat Monitoring

In certain cases, such as in a low memory condition, Web services may not be able to process new requests or raise operational events when they become unstable. Therefore, one of the aspects of health monitoring is to ensure that the Web services are still active and are responding to requests in a timely fashion. The main idea involves using a monitoring application that periodically sends messages to the Web services and watches for the presence and the rate at which the Web service responds to the monitoring application.

In addition to exposing the Web method that performs business operations, each deployed instance of Northern Electronics's Web services also implements a **Ping()** method that does not perform any real business function. The method is invoked by a monitoring application that checks for SOAP responses from the **Ping()** method. The monitoring application also measures the round trip response time to make sure that the Web service is responding within a reasonable timeframe.

To really determine the availability of a particular Web service, the monitoring application would ideally invoke the actual business method, such as **SubmitPurchaseOrder().** However, many business Web service are non-idempotent and have real business effects that are difficult to undo. For a business service that provides a read-only data operation, for example, retrieving a sales catalog, it may be feasible to invoke the actual Web method for health monitoring purposes.

Another reason that Tom may want to use a non-business related method for health monitoring is that the Web service's health may be dependent on other back-end components and he also wants to be able to detect the health of those dependencies. For example, the ShippingService Web service is a component in an n-tier distributed application that consists of Message Queuing, a Message Queuing monitoring process, and a SQL database. Failure of any of these components may affect the performance and integrity of the ShippingService Web service. As a result, Tom designs the **Ping()** management method to test for the availability of those dependencies.

Conceptual Architecture for Heartbeat Monitoring

To monitor the availability of the ShippingService Web service, Tom and Dan investigate the details of a possible monitoring architecture that will satisfy the requirements. The resulting conceptual architecture is illustrated in Figure 7.

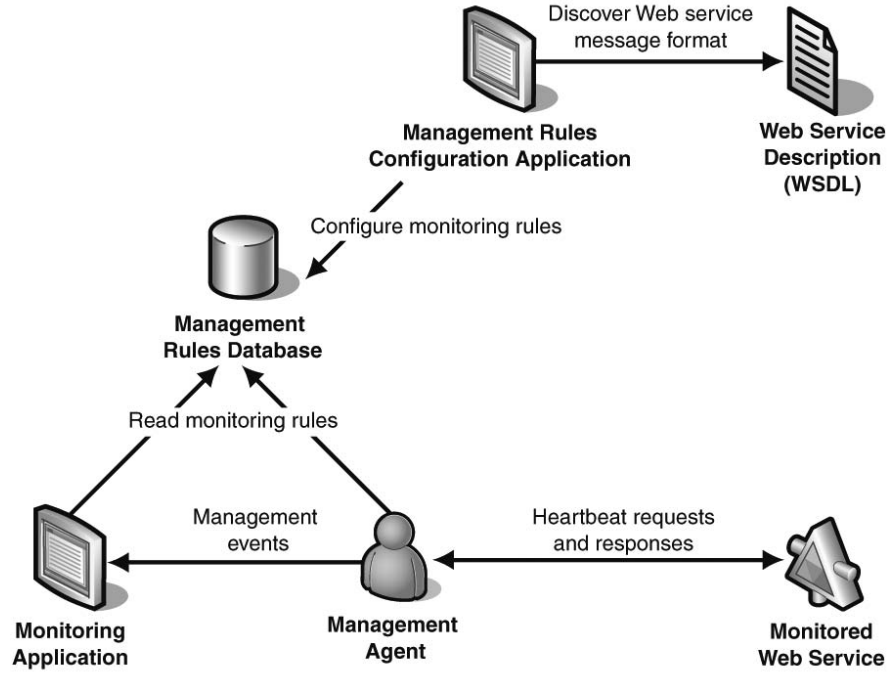


Figure 7. Conceptual heartbeat monitoring architecture

There are a few key components of the architecture:

- Management agent.
- Monitoring application.
- Management rules configuration application.

The next sections describe each of these.

Management Agent

This is a software component that sends SOAP messages to the monitored Web services. It uses management policies in the management rules database to generate the SOAP requests that correspond to the Web method that is exposed for heartbeat monitoring. It also triggers management events in response to the pinging requests. For example, the software agent can raise an error event if it did not hear back from the monitored Web service within 30 seconds.

Monitoring Application

The monitoring application receives and processes management events from the management agent. This application handles the received events according to a set of rules configured in the management rules database. For instance, it may send an e-mail notification to the application support analyst when a missed heartbeat error event is received.

Management Rules Configuration Application

As mentioned earlier, the software agent and monitoring application use a set of rules in the management database to interact with and react to the responses of the Web service. The management rules configuration application is the component that enables these management policies to be configured. Some examples of input for configuring the management rules are:

- The format of the SOAP messages that the agent sends and receives to the Web service. The format can be discovered from the WSDL file that describes the Web service message schema. If the Web service method contains any additional input, such as custom headers or method parameters, it is also specified at this time.
- The Web service endpoints to be monitored.
- The time intervals that heartbeat requests will be sent to the monitored Web services. For example, the rule can specify that a request is sent every 30 seconds. This parameter is usually tuned to correspond to the availability commitment in the service level agreement.
- The warning or error event and the details in the event to send to the monitoring application in response to the heartbeat request.

Using Microsoft Operations Manager (MOM) for Heartbeat Monitoring

During the course of designing the monitoring infrastructure, Tom learns from Dan that the Microsoft Operations Manager 2005 solution is already being used at Northern Electronics for monitoring existing Windows server infrastructure such as Microsoft Exchange servers and the Active Directory servers within their corporate datacenter. Therefore, Tom and Dan are interested in extending the existing MOM capabilities to monitor the ShippingService Web service as well.

MOM is an extensible monitoring and management application that can be customized to monitor different kinds of server applications. The Web Sites and Services MOM Management Pack (WSS MP) is an add-on to the MOM 2005 product for monitoring Web sites and Web services. WSS MP consists of a few product components that map well to the desired functionalities outlined in the conceptual architecture.

In addition to the MOM 2005 Administrator Microsoft Management Console (MMC), the WSS MP ships with the WSS Server Configuration Wizard that provides the user interface to help the administrator specify the Web service endpoint to monitor, the SOAP messages to send to those monitored endpoints, and additional monitoring parameters such as the monitoring interval that heartbeat messages are sent. The wizard allows the administrator to generate the SOAP messages in three ways:

- Derive SOAP message from a WSDL file.
- Capture SOAP messages sent and received by the Microsoft Internet Explorer browser.
- Manually create and edit SOAP messages.

The message generation screenshot is shown in Figure 8.

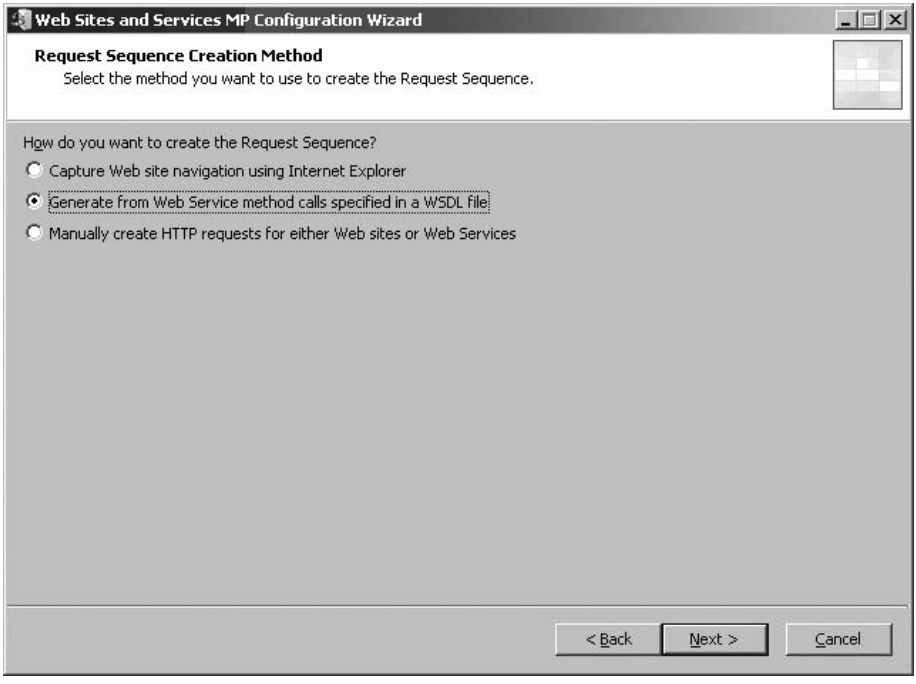


Figure 8. Creating a WSS MP request sequence

After the message format is captured by the wizard, other properties such as the message parameters, the monitored Web service endpoint, the heartbeat interval, and the error event that is generated when a response is not received can be associated with the message through a collection of property pages as shown in Figure 9. The SOAP message with its management properties is known as a request sequence in WSS MP.

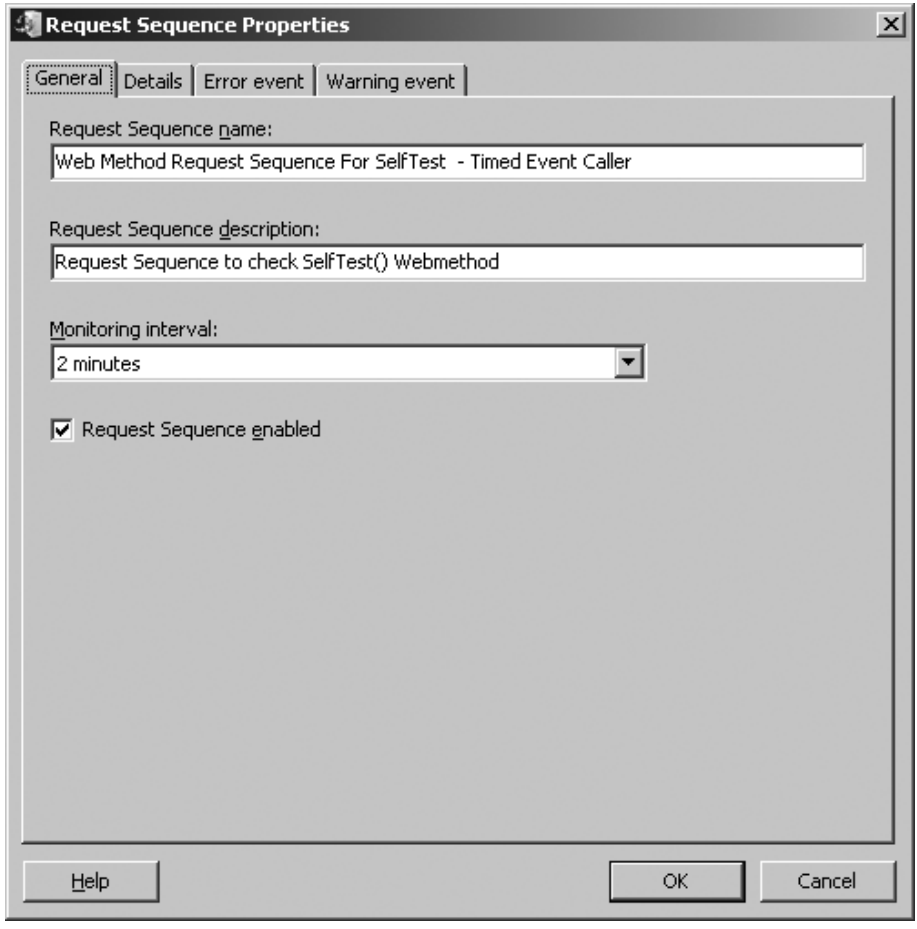


Figure 9. Property page for a request sequence

When created, each request sequence belongs to a request sequences group. The request sequence group is a mechanism for specifying and grouping various request

sequences and associating them with a collection of MOM agents that will send out the heartbeat messages defined by the request sequences. To complete the configuration task, each request sequence group is associated with one or more MOM computer groups that define computers installed with the MOM agent and the WSS Agent Configuration Component. Figure 10 illustrates these concepts.

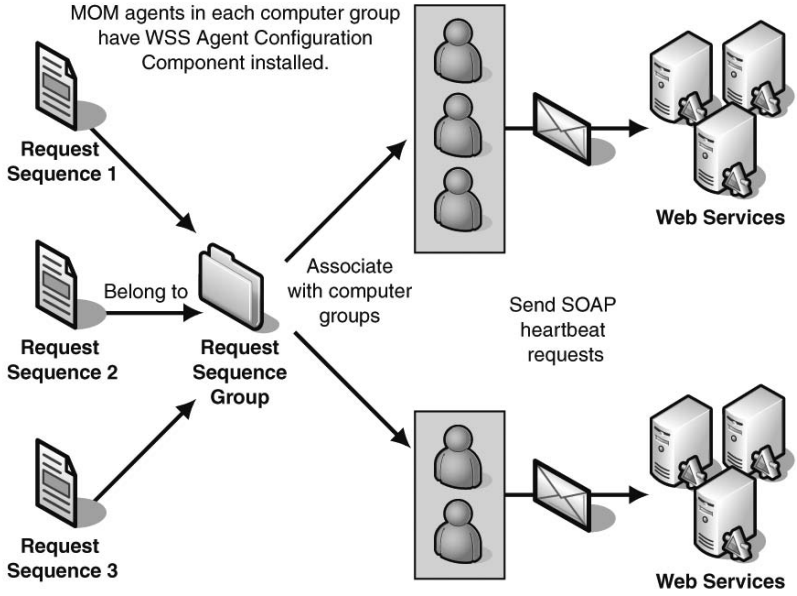


Figure 10. Configuring request sequences and relating them to management agents that send heartbeat requests

Each request sequence also specifies a set of criteria that the responses must meet to satisfy the heartbeat requests. For instance, the response message must contain a particular string in a stated regular expression. Figure 11 shows the property page for capturing the response requirements. When the requirements are not met, the MOM agent will generate either an error or a warning event. The request sequence group can also contain a set of alert rules that monitor for the error and warning events to send out an alert notification, such as sending an e-mail notification to the application support analyst indicating the heartbeat failure event.

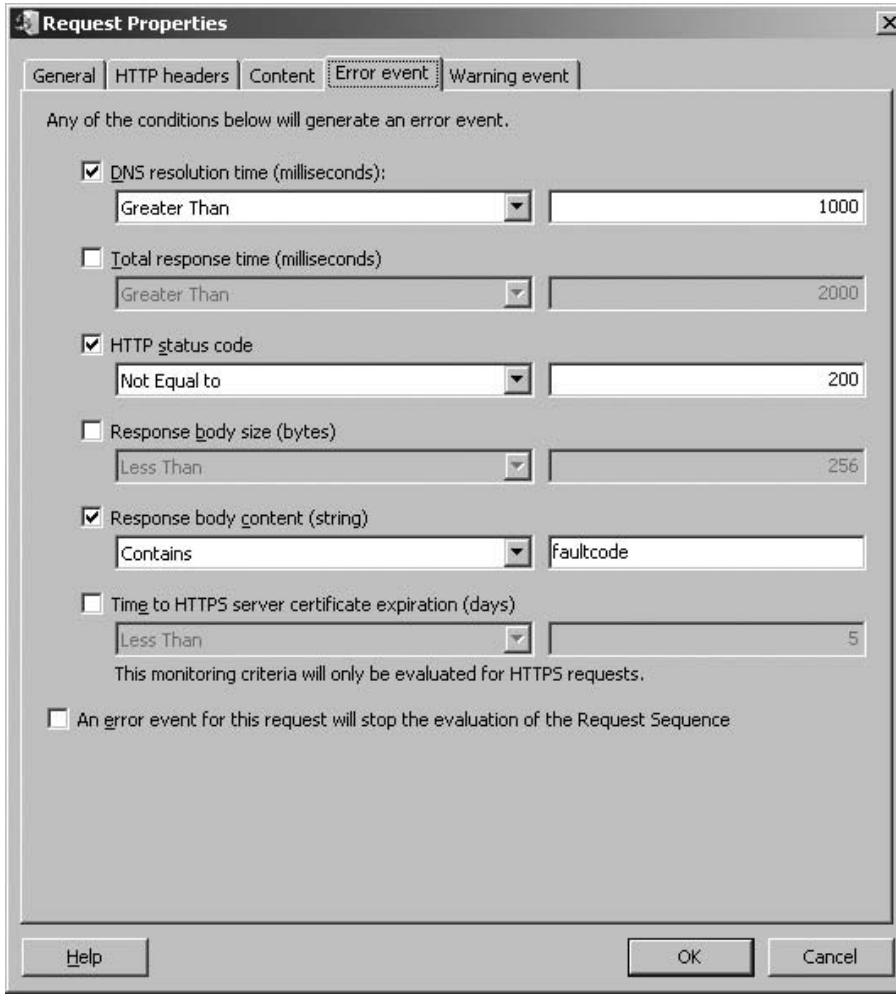


Figure 11. Error generation criteria

Using MOM for Events and Performance Monitoring

In addition to monitoring for the availability of the Web service, it is also important to ensure that the Web services are functioning optimally. Factors such as the following may lead to sub-optimal performance:

- Dependency issues. Today's distributed applications do not work in isolation. For instance, the ShippingService Web service relies on the database for storing and retrieving business documents. In the event that the database connection is lost, the Web service will still be able to receive and process client requests, but it will not be able to complete the requests.
- Security issues. Application activities such as unexpected changes in process privileges may indicate potential security elevation of privileges attacks. Attackers may then use the application process and privilege that cause the application to perform unintended tasks.
- Integrity issues. Application errors such as unhandled memory exceptions can lead to processing errors even though the Web service may appear to be accepting requests and functioning.

For these reasons, it is necessary to monitor management events and performance counters to keep track of and maintain the services' health. To do so, Dan uses the health model information described in Tables 4, 5, 6, 7, 8, and 9 to develop a MOM pack for monitoring the ShippingService Web services. Specifically, the ShippingService Web service MOM pack consists of the following categories of rules that specify:

- The business operations, application, and system events to process and how to handle them. For instance, an event rule can capture a specification such as "For each Windows event with ID 5002 that is written by the **ShippingWebServiceEvents** provider, raise an alert with the severity set to **Error.**"
- The set of performance counters to monitor, the sampling intervals of those performance counters, and the performance threshold that is used to generate management alerts. For example, the rule may specify "Sample the percentage processor time performance counter every 20 seconds and raise an alert if the average value over the last 5 samples is greater than 80 percent."
- The manner to handle the alerts that are generated by the events and performance monitoring rules. For instance, "For each alert with severity set to **Critical,** send an e-mail notification to the users in the **Operations** user group."

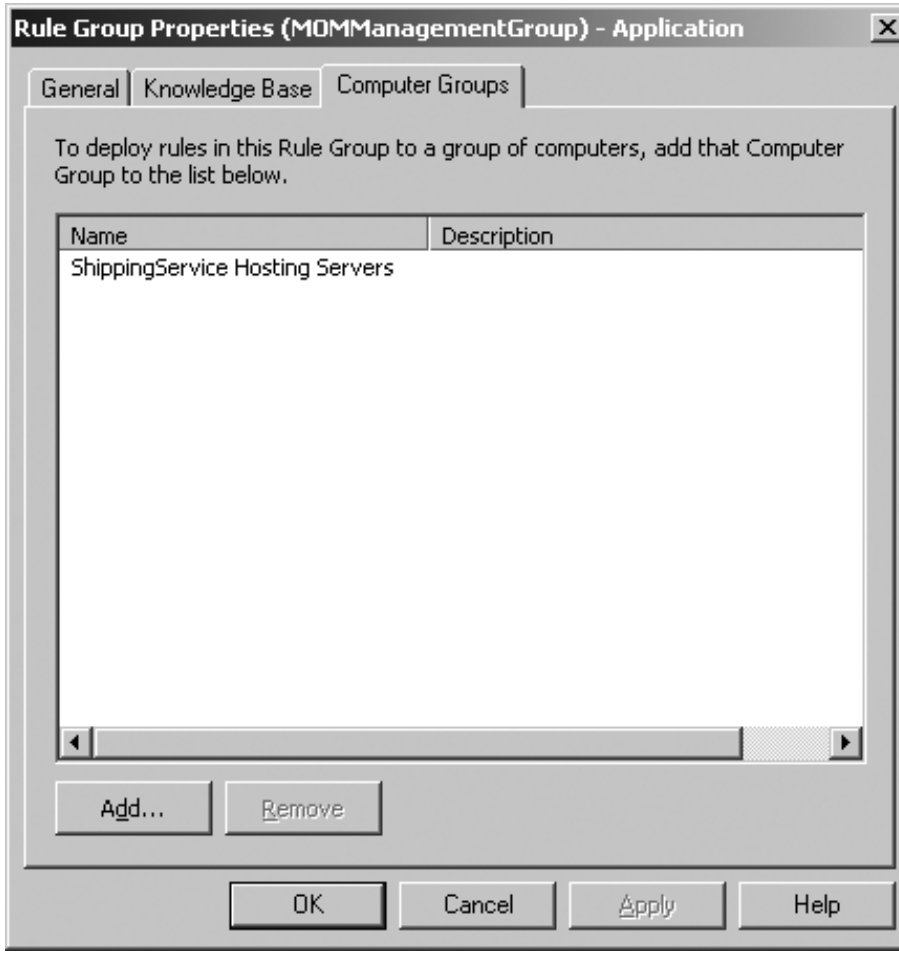


Figure 12. Associate MOM pack with machine group

After the MOM pack is created, it is associated with a MOM server group that contains the computers in the Web service farm. This step can be accomplished by right-clicking on the MOM pack and then clicking **Associate with Computer Groups** option to activate the dialog box shown in Figure 12.

Figure 13 shows the structure of the rules that have been created in the MOM pack to monitor the ShippingService Web service. At the top level, the rules are categorized according to instrumentations that are implemented for monitoring events and metrics at the business, application, and system levels. This is a reasonable organization for the management policies, because it considers the fact that there are multiple levels of service metrics that are meaningful to different personnel in Northern Electronics, who can then take actions on handling exceptions at the respective levels.

For each category, the event handling rules define the interesting types of events to process at that level. The event rules capture information details such as the type, identifier, source, and severity to define the kinds of events MOM should process. The event rules also define the details of the management alert to generate in response to the events. Performance counters handling rules define the types and source of the performance counters to monitor and the threshold on those counters that will trigger management alerts. The alert rules define the management actions that should occur in response to monitored events and performance counters. Generally speaking, management actions can be grouped into two broad categories: sending notification or starting a computer program to automate diagnostic and recovery actions.

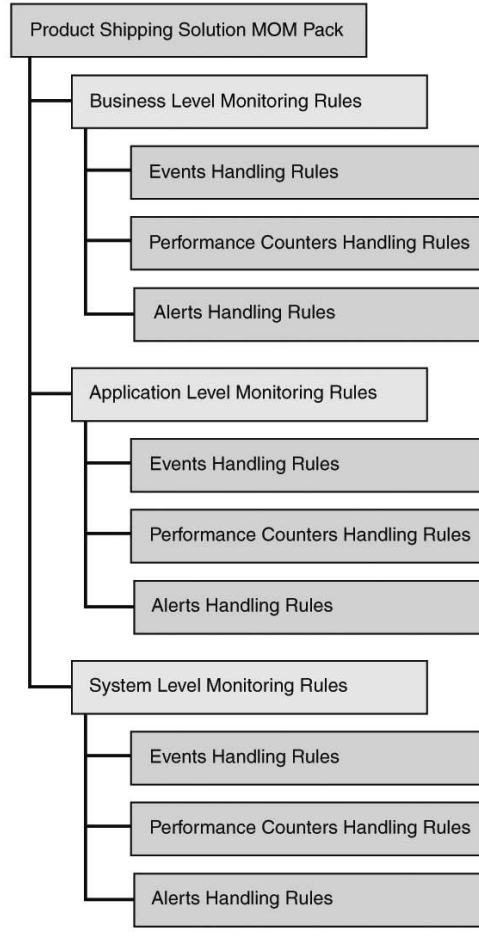


Figure 13. Rules structure of Product Shipping Solution MOM Pack

Event Monitoring

The previous section discussed how the ShippingService Web service raises events (such as when the database connection is lost) and writes the events to the Windows event log, WMI, or a SQL database. To monitor and react to such events, Dan has created a set of rules for monitoring the events generated by the ShippingService Web service.

The next few screenshots show some of the properties associated with the rules that have been configured to monitor the lost database connection event.

Figure 14 shows some general information about the rule and that the event rule has been enabled.

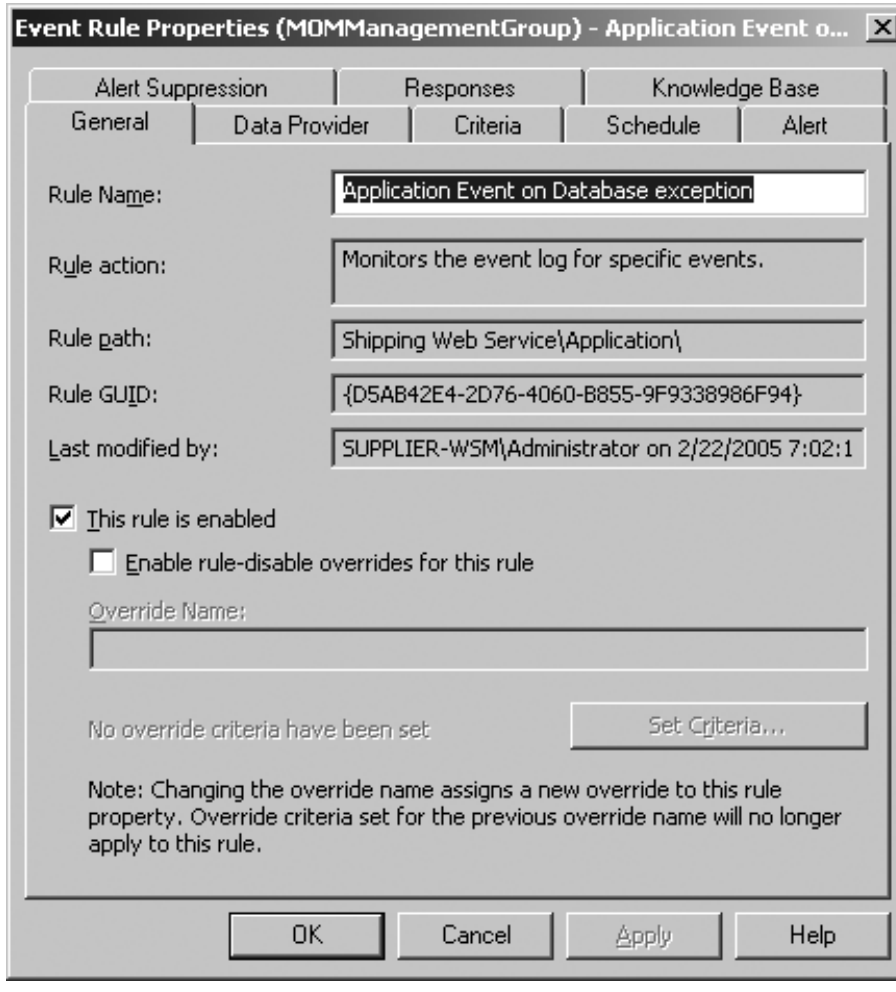


Figure 14. Lost database connection event rule

Figure 15 shows that the event provider named "WebServiceManagement" is the generator of the event and the event is logged to the Windows event log.

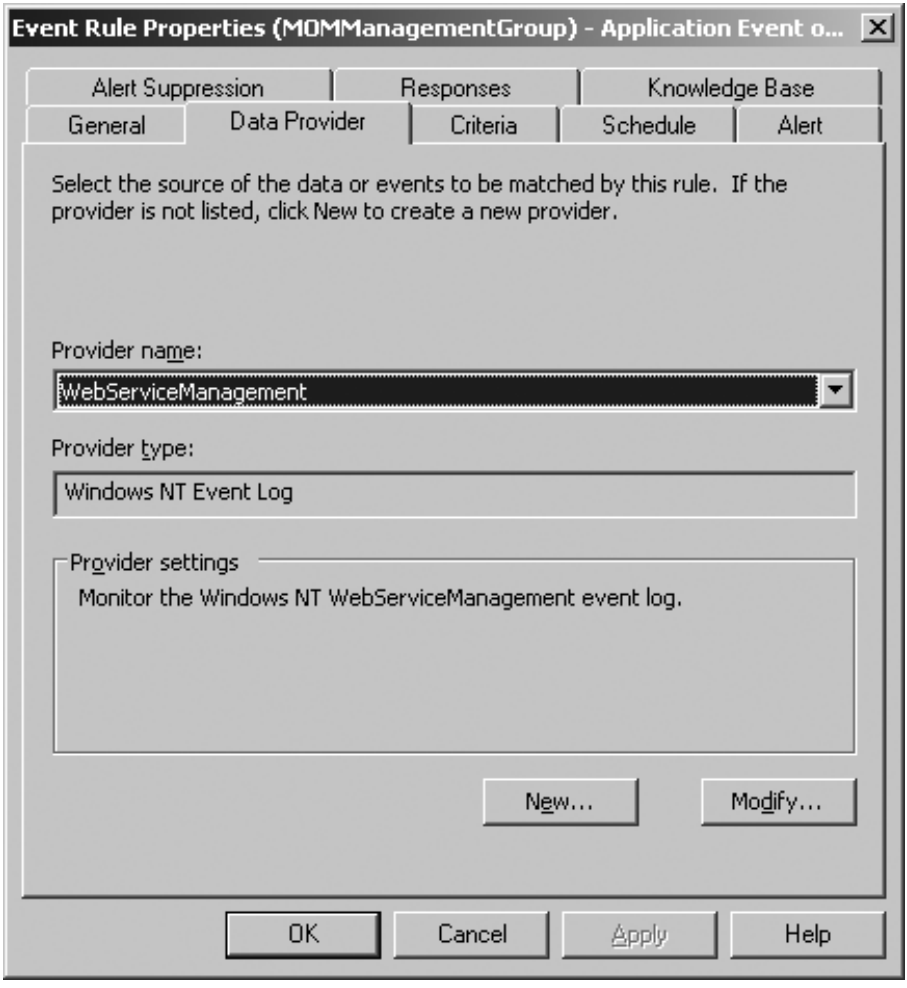


Figure 15. Event provider for lost database connection event

Figure 16 shows the property page that specifies that an alert with the severity "Error" should be raised when this event is encountered.

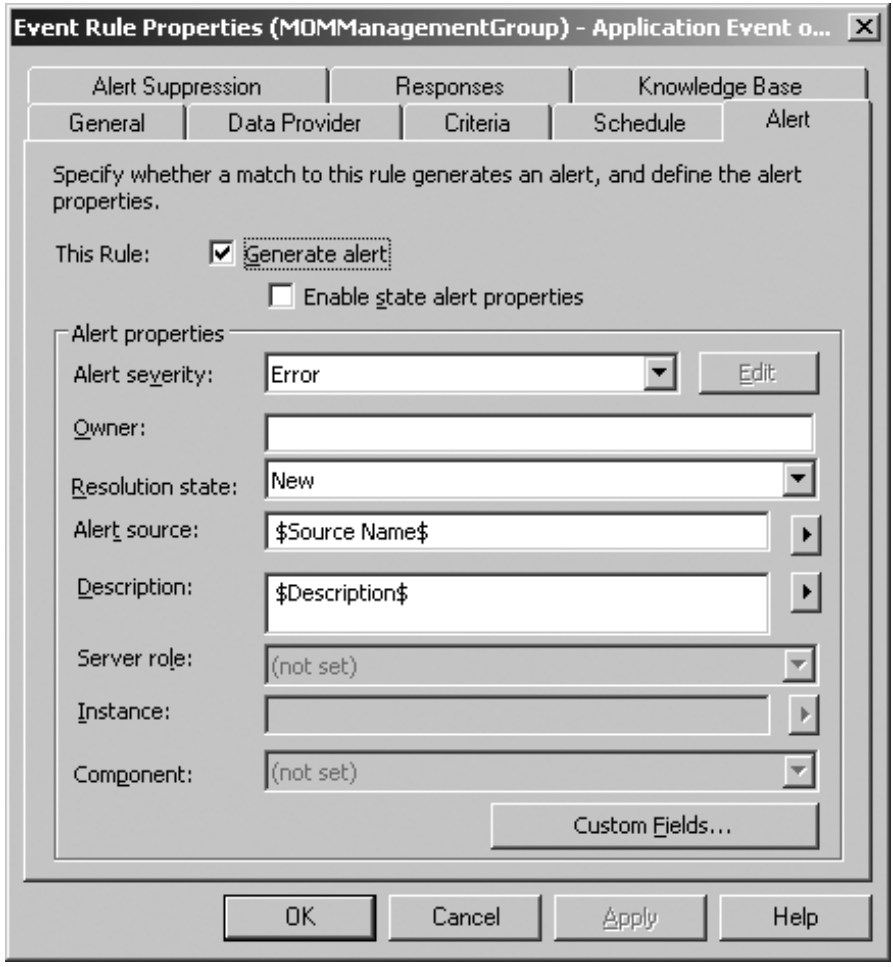


Figure 16. Alert raised by lost database connection event rule

Dan configures event rules in the Shipping Product Shipping Solution MOM pack to handle the other business and application events generated by the Web service in a similar manner. These other events are:

- Mismatch of dates in pick-up notification document (violation of BOR1).
- Delay of truck arrival (violation of BOR2).
- SOAP exception (violation of AR5).
- Null reference exception (violation of AR6).
- NET application exception (violation of AR7).

In all cases, the data that has been gathered during the health modeling exercise and summarized in Tables 4 through 9 representing the detection, diagnosis, recovery, and verification process steps are used to configure the MOM rules.

Performance Counters Monitoring

The primary goal of performance monitoring is to ensure that the Web service is functioning optimally in various aspects according to a set of pre-determined quantitative measurements. Therefore, the basic concept behind performance monitoring involves the following mechanisms and approaches:

- Performance counters are implemented to capture the quantitative measure that the operations team is interested in monitoring. For instance, a processor time performance counter is defined and used to capture the percentage of time that the CPU remains active over a given time period.
- Under normal operating conditions, the value of the performance counter is expected to remain within a particular threshold. Specifically for the ShippingService Web service, in order to maintain a reasonable user experience, Northern Electronics has decided that the computers running the Web service should maintain less than 90 percent CPU utilization.

When the threshold is reached or exceeded, the monitoring application triggers a management alert.

Figure 17 shows the dialog box for configuring a performance measure rule for monitoring the processor utilization on computers hosting the product shipping solution Web services. It shows the Windows NT Performance Counter provider as the source of information for this particular counter.

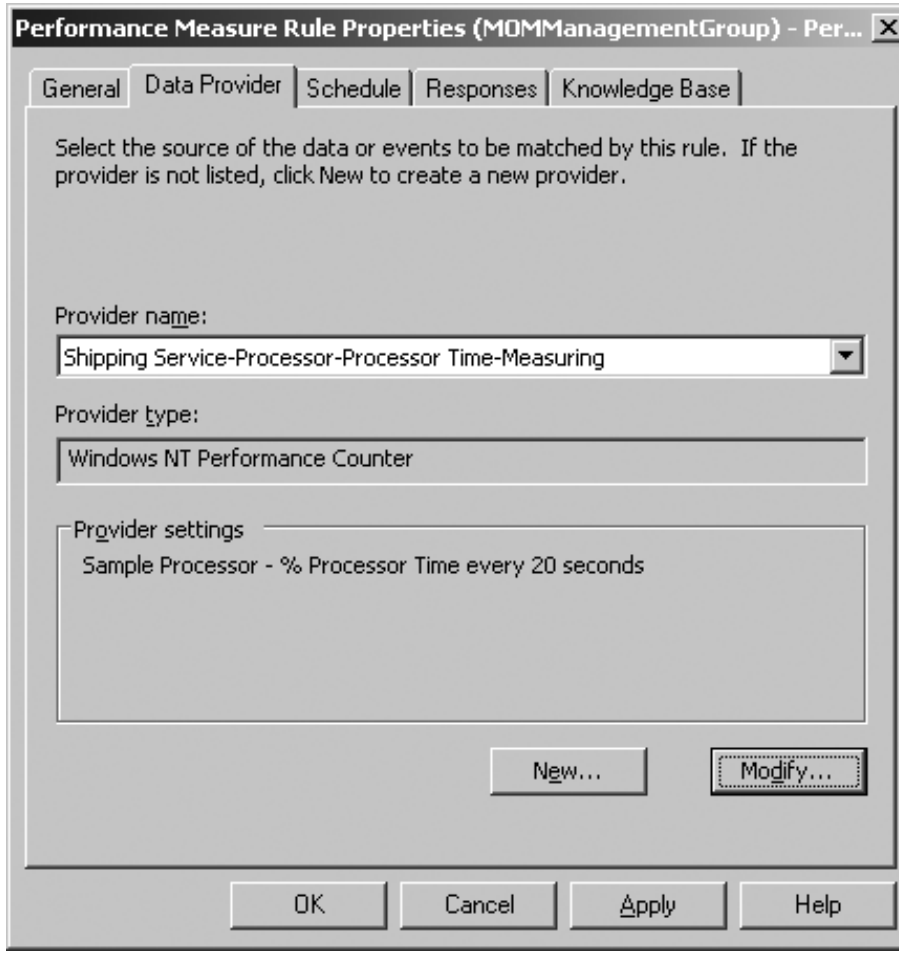


Figure 17. Processor time performance counter

Figure 18 shows that this counter is being sampled every 20 seconds.

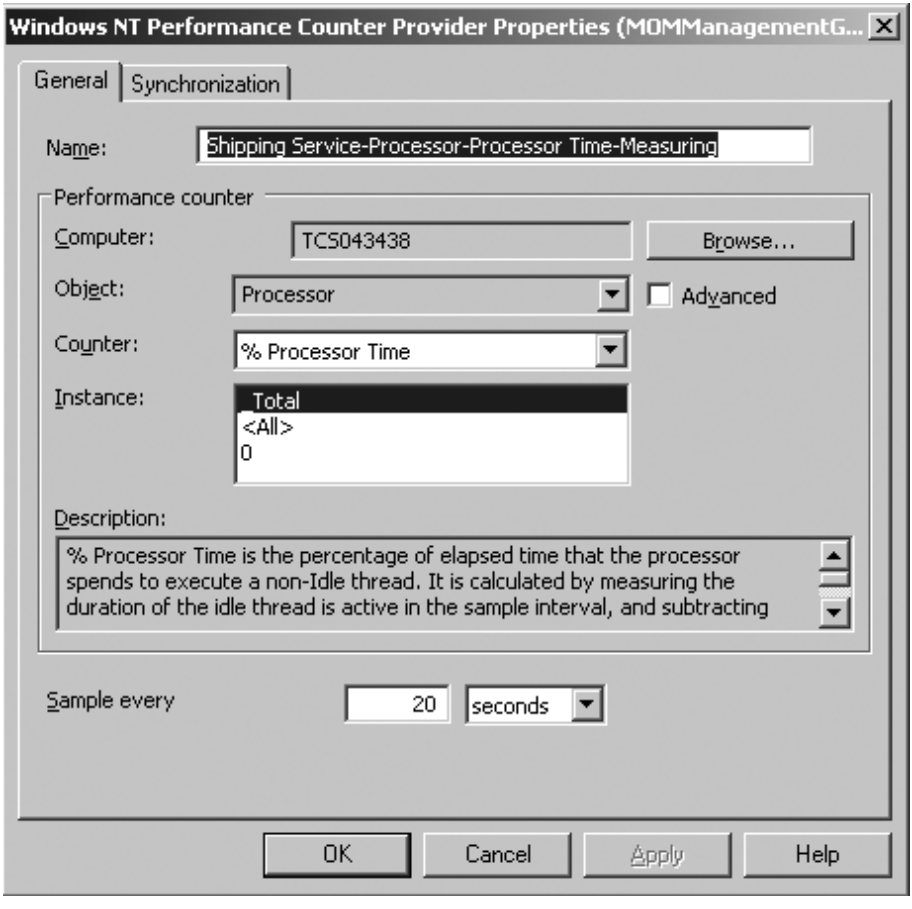


Figure 18. Sampling interval for processor time performance counter

Figure 19 shows the performance threshold rule for this counter which is set to generate an alert if the average value calculated over five sampling intervals exceeded 90 percent CPU utilization.

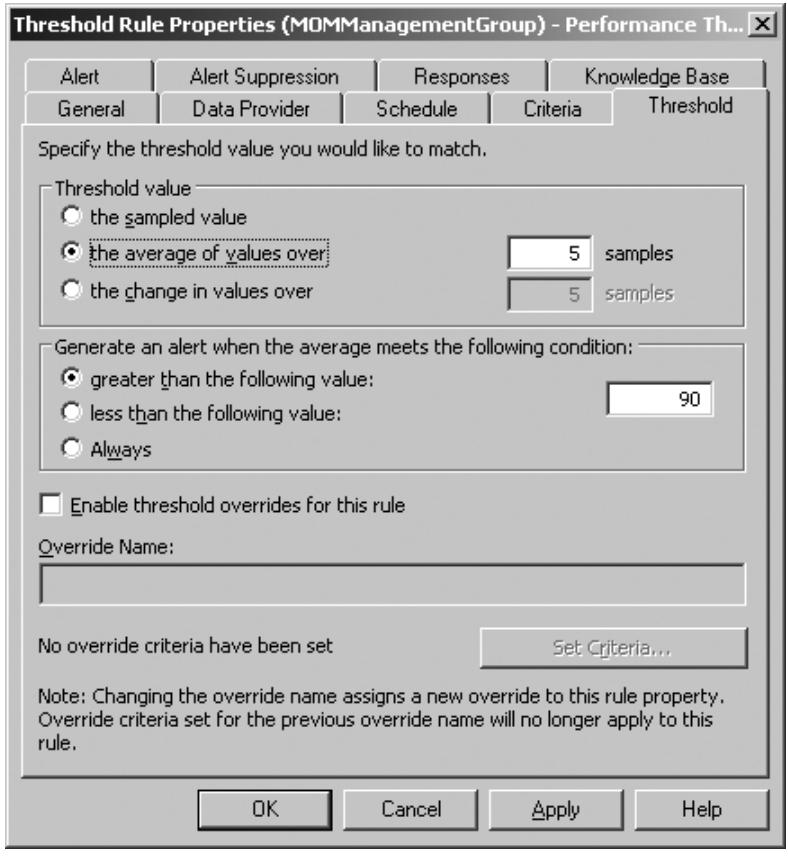


Figure 19. Performance threshold for processor time performance counter

Other performance rules are similarly configured to handle application and system performance counters that are essential for monitoring the ShippingService Web service. Specifically, performance rules are defined for the following counters with the requirements listed in parentheses:

- Available system memory (SR2)
- NotifyPickup document processing time (AR1)
- MSMQ queue length (AR4)

Handling Management Alerts

As demonstrated earlier, management alerts are raised when events or performance thresholds that are triggered need to be followed up with management actions. The actions fall into two main categories: sending a notification and/or starting a program command. For example, when the message queue length is exceeded, a message queue critical error alert is sent. Dan has configured a MOM alert rule to handle this particular alert.

Figure 20 shows the dialog box that set the two management responses to automatically restart the message queuing service and to send an e-mail notification to the **Operations** user group.

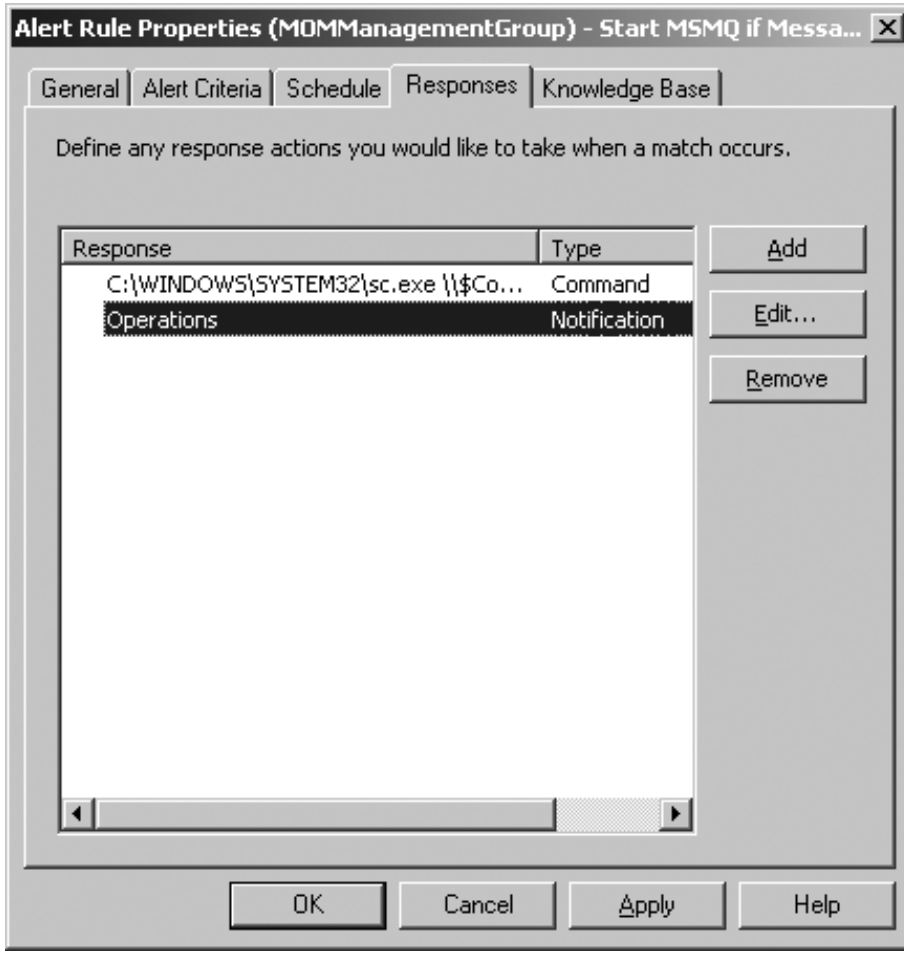


Figure 20. Alert rules to handle Message Queuing management alert

At Northern Electronics, different organization roles are responsible for handling the different categories of management alerts. The MOM alert rules provide a convenient way to enable this requirement by allowing Dan to create different alert rules to specify exactly how each alert should be channeled. For example, business operations managers are expected to remedy transport ordering issues with Acme Consolidation Company. Therefore, specific alert rules are created to handle alerts generated by business event rules. These business operations alert rules are configured to send out an e-mail notification to the **Business Operations Managers** user group.

Conclusion

Health modeling is a key stepping stone to delivering an instrumented system that can be monitored and managed at the expected service levels. Northern Electronics chose to adopt the health modeling approach that encompasses the process steps of detection, verification, diagnosis, recovery, and re-verification. Through the exercise of identifying the instrumentations, actions, and conditions that were required to complete each step, Northern Electronics was able to translate a set of service requirements into management policies that enforce the required service monitoring activities.

Northern Electronics also chose to deploy and use MOM in the following capacities:

- Actively monitor for the availability of the product shipping solution Web services by making sure that Web services respond to requests in a timely fashion.
- Subscribe to management events to learn about potential problems indicated by Web services.
- Check up on performance counters to detect out-of-bounds conditions.
- Notify management personnel of issues that require attention.
- Automatically invoke programs to facilitate auto-recovery or diagnostics tasks.

Additional Resources

Health Modeling: A Key Step to DSI-Enabled Applications (http://www.microsoft.com/windowsserversystem/dsi/designwp.mspx)
Logging and Instrumentation Application Block" on MSDN at (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag2/html/logging.asp?frame=true)

Notes

Web Service Deployment: Deploying Web Services in the Northern Electronics Scenario

Architecture Chronicles
Dynamic Modeling: Aligning Business and IT

Frederick Chong with Jim Clark, Max Morris, and Dave Welsh
September 2005

Applies to:

- Enterprise Architecture
- Solution Architecture
- Service Oriented Architecture (SOA)
- Service Oriented Management (SOM)
- Application Integration
- Business Process
- Business Operations Modeling

Summary

The *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* seek to present to business, solution, and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. This document focuses on the deployment of Web services for a product shipping solution in the Northern Electronics scenario.

Contents

This Document

This document is part of the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT*. This volume seeks to present to business, solution and infrastructure architects a holistic and integrated approach to aligning business and IT through dynamic modeling to achieve better performance, accountability, and business results. The information map to the series provides an up-to-date description and cross-index of the information available and can be found at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

Abstract

This document describes how Northern Electronics uses a deployment modeling approach to deploy Web services that make up part of the product shipping solution. It shows how Northern Electronics specifies the deployment policy for one of its Web service. Then, it shows how Northern Electronics uses this deployment policy and Microsoft Systems Management Server 2003 to package software components and configuration settings of different types and versions of Web services into deployment packages; associate different Web services deployment packages with computers in Web service farms; and both schedule and manually activate and control when the Web service deployment begins and completes.

Acknowledgments

Many thanks to Nelly Delgado for her help with technical writing, Claudette Siroky for her graphics skills, and Tina Burden McGrayne for her copy editing.

The authors would also like to thank Gajanan Phadke, Vishal Kapoor, Amol Wankhede, Aziz Matheranwala, Amit Kumar Srivastav, Bhushan Pawade, Bhasha Johari, Avadh Jain, Mughda Gadre, Maneesha Nalawade, Sonika Arora, Anil Sharma, and Anurag Katre (all of Tata Consulting Services) for their contributions to the implementation of the Northern Electronics solution.

Introduction

This document focuses on the deployment of the Web services in Northern Electronics's product shipping solution. Background information about the Northern Electronics scenario can be found in the *Architecture Chronicles* on *Dynamic Modeling: Aligning Business and IT* at the Microsoft Architecture Resource Center (http://microsoft.com/architecture).

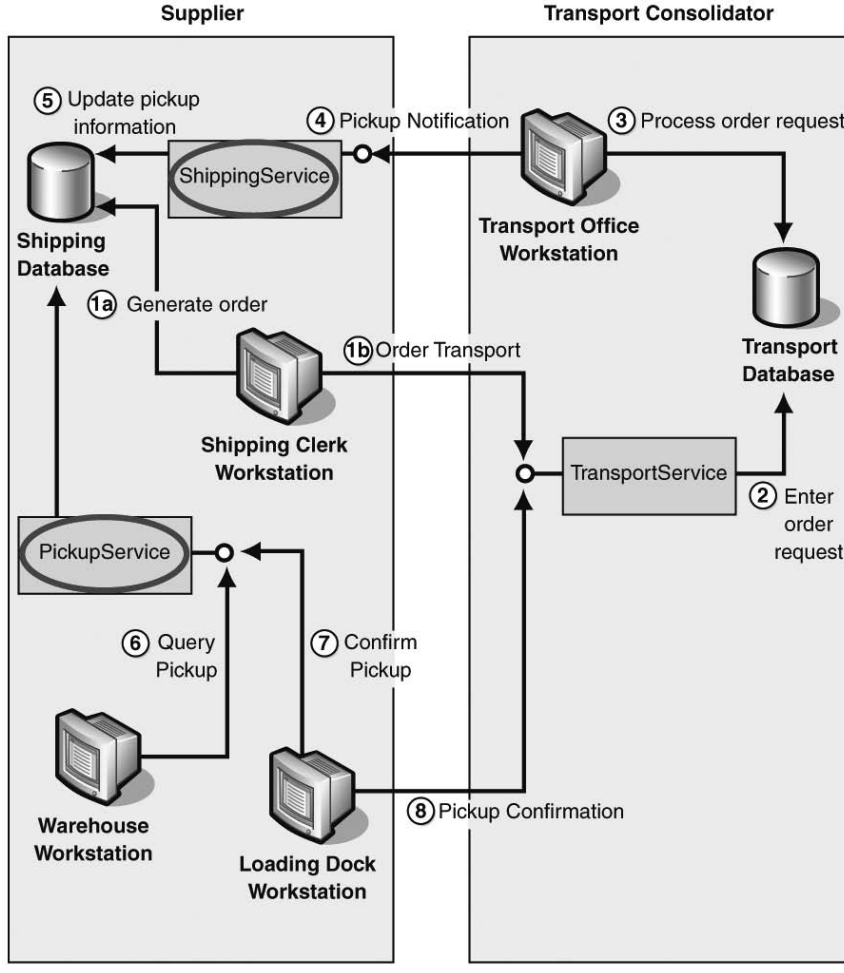


Figure 1. Northern Electronics Web service

Northern Electronics designed and implemented two Web services as part of its product shipping solution: ShippingService and PickupService. The ShippingService Web service is an external-facing Web service that enables Northern Electronics to receive a transport order notification from its partner, Acme Consolidation Company. The transport order notification indicates whether Acme Consolidation Company can meet the requirements stated in a corresponding transport order. The PickupService Web service is an internal Web service that supports the Northern Electronics warehouse management application in organizing the queue of goods that are to be picked up and loaded in future working days. These two Web services are shown in Figure 1.

After the software components that implement the Web service functionality are tested, the software programs are ready to be deployed into the corporate data center where the IT servers are hosted. This document examines Northern Electronics's requirements, architecture, and tools for deploying the ShippingService Web service. This reflects the general techniques that Northern Electronics uses to deploy all of their Web services.

Who's Who in Northern Electronics

The following individuals from Northern Electronics are involved in deploying the Web services:

- Jackie: She is the program manager whose responsibility in this context is to identify the deployment requirements and organize the effort to complete this phase of the project.
- Tom: He is the solution architect whose responsibility in this context is to work with the lead developer and application administrator to design the deployment policy and infrastructure.
- Sam: He is the lead developer whose responsibility in this context is to create the installation program files.
- Dan: He is the application administrator whose responsibility in this context is to implement the deployment policy and oversee the deployment of the Web services.
- Craig: He is the application support analyst whose responsibility in this context is to deploy the Web services and monitor the deployment status.

Deployment Requirements

Jackie, the program manager, assesses the deployment requirements. One of the main goals is to have multiple run-time instances of the Web service deployed in Web farms within the data center so that there is minimal impact to the service availability in case a subset of the servers goes down. The Web services instances that are deployed must be images of one another, meaning that the software components and configuration are all identical within the Web farm. Instead of manually deploying the Web service instances one-by-one on each computer, Jackie wants to be able to take advantage of the similarity of the deployed instances to automate the installation and configuration of Web services in the Web farm.

From the deployment management perspective, Jackie realizes that the administrator needs to be able to decouple the activity of specifying the deployment policy for a particular Web service from the activity of executing the deployment. These two functions are typically performed by individuals in different roles. Jackie works with Tom to decide that the deployment policy should specify:

- The collection of software components and programs that implement the Web service.
- The application and computer configuration settings that affect the runtime execution of the Web service.
- The server farm or collection of computers where the service should be installed.

When deploying a new Web service, Northern Electronics's policy is to schedule the deployment to occur at a time when it is least disruptive to peak time operation, for example, just after midnight. On the other hand, the company also wants to use the same deployment infrastructure to push out critical security patches and bug fixes that may require immediate deployment action. Therefore, Dan must be able to trigger the actual installation of the service manually or automatically through a schedule. After the installation starts, the operations staff must be able to monitor the progress of the installation within a given time interval. The status of the installation may indicate that the installation is in progress, has succeeded, or has failed. If the installation does not complete successfully, the operation staff must be able to diagnose the reasons for the failure.

Deployment Modeling

After considering the deployment requirements, Dan, the application administrator, notes that that there are two levels of design details that must emerge to support the requirements.

The first set of design details is focused on the deployment infrastructure components that can help to:

- Activate, download, install, and configure software in a distributed environment.
- Control, monitor, track, and troubleshoot the set of deployment activities from a central console.

The second set of design details focuses on the policy mechanisms that specify:

- The software components and configuration settings for different types of Web services.
- The computer groupings targeted for deployment.
- The Web service types to be deployed and where they will be deployed.
- The timing of the deployment.
- The deployment infrastructure settings, such as where the installation files reside and where progress should be logged.

Deployment Infrastructure Design

The first task involves designing a deployment infrastructure that will facilitate the starting, ending, and troubleshooting of deployment activities. To enable such activities, Dan collaborates with Tom, the solution architect, to come up with a logical architecture of the deployment infrastructure as shown in Figure 2.

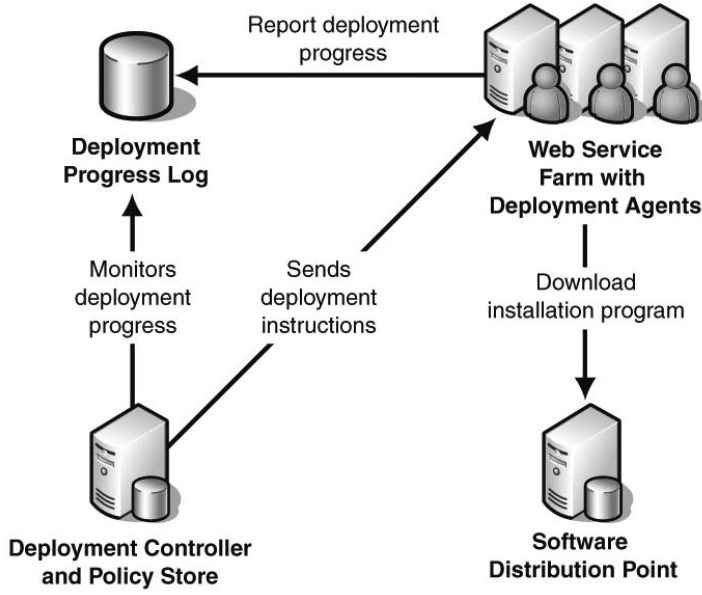


Figure 2. Logical architecture of deployment infrastructure

The logical architecture consists of the following entities:

- Deployment controller. The application administrator uses the deployment controller to:
  - Configure and manage deployment policy.
  - Schedule and trigger deployment actions at the Web service farm computer.
  - Monitor deployment progress and take corrective actions if necessary.
- Deployment agent. This interacts with the deployment controller and performs software deployment actions according to the instructions and policy received from the deployment controller.

- Deployment policy store. This provides the central location where the deployment policies are managed and stored.
- Software distribution point. This provides the network location where the installation program and Web service software component can be downloaded at deployment time.
- Deployment progress log. This provides the logical location where deployment agents write deployment status and the deployment controller monitors progress.

Deployment Policies

The second task involves designing a policy structure to express information related to what, where, and when to deploy. The following design concepts are used for this purpose:

- Computer collections
- Web service deployment versions
- Software to computer bindings
- Deployment scheduling
- Deployment infrastructure settings

Computer Collections

Computer collections are used to express logical groupings of physical computers in a Web farm so that all computers within a grouping can be deployed and configured in the same manner.

Web Service Deployment Versions

The specification of the operation interface of a Web service will evolve over time. The software components and the deployment configuration will likely differ among different deployments of the Web service. Each version of the Web service may support a different service interface or document contract.

Note Architects and developers need to know at design-time and run-time which version and which deployment of a Web service they intend to create or use. Versioning is a solution design consideration that is not addressed within this document. Related topics such as discovery and directory are also not addressed within this document.

Over time, the implementation of Northern Electronics's ShippingService Web service may be enhanced for the following reasons:

- A new version of the Web service needs to understand a new element in the transport order notification document. For example, a new field, such as a transport company identification code, may be introduced due to new transportation regulation requirements. When the new version of the Web service is introduced, the Web service should recognize the identification code in the updated format of the transport order notification document.
- A new implementation of the Web service is built using a new messaging runtime.
- An existing implementation of the Web service needs to be reconfigured to use a different instance and implementation of message queuing.

When Northern Electronics decides that a new set of software components or application configuration settings (or both) need to be deployed, it must be possible to package the software components and settings into distinguishable deployment packages even though the resulting Web service provides the same logical business service. This idea is shown in Figure 3.

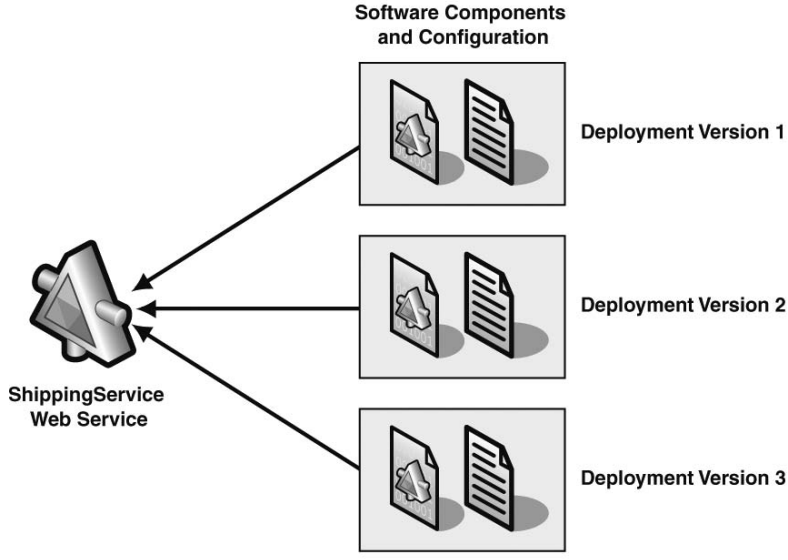


Figure 3. Deployment versions of ShippingService Web service

Software to Computer Bindings

Now that Dan has computer collections and Web service deployment versions, he needs to specify the Web service version that will be deployed to the specific collection of servers. By associating the two entities, Dan creates a deployment unit that can be made available on the network. Note that it must also be possible to deploy different versions of the same Web service on the same collection of servers. This requirement can be illustrated by a simple example where different versions of the ShippingService Web service are to be deployed in multiple Web server virtual directories created on a particular computer in a server collection. Figure 4 illustrates the ShippingService Web service Version 2 associated with Server Collection 1.

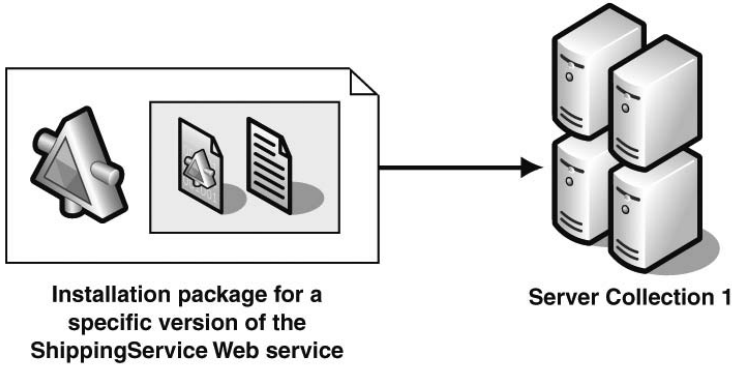


Figure 4. Binding Web services components and settings to server collections

Deployment Scheduling

Not only must the policy allow Dan to specify what needs to be deployed and where it should be deployed, it must also allow him to specify when a deployment is to take place. A deployment can begin immediately, when the administrator selects the **Start Software Distribution** option in the deployment management console. In other cases, Dan may want to schedule a deployment activity, for example, in the middle of the night when the installation network traffic is least disruptive to the operational environment. Figure 5 illustrates this policy concept of associating a time element with the unit of deployment.

Deployment Infrastructure Settings

For each deployment activity, the deployment agents require additional information to

deploy the software, such as pointers on how and where to download the installation package. In many cases, these are distributed file shares that already store the installation package files. As another example, the deployment agents will receive information about the location of the log file and the log file name that will label the progress log. The log file is used to write installation progress. Figure 5 illustrates that it must be possible to specify these deployment infrastructure settings with the Web services that are to be deployed.

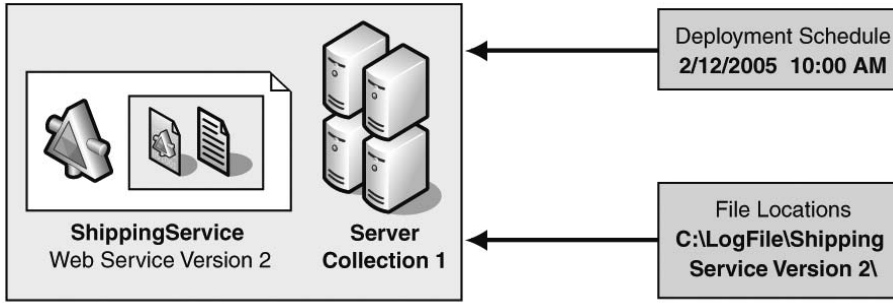


Figure 5. Deployment policies

Systems Management Server 2003

After Dan has a good understanding of the deployment requirements and the approaches to deploying the Web services, he investigates existing products that can help him distribute the software in a straightforward and accountable manner.

The Northern Electronics IT organization uses Microsoft Systems Management Server (SMS) 2003 to deploy desktop software applications and manage security patches on client workstations. Dan believes he can leverage the SMS knowledge and capabilities of the IT organization to help him deploy Web services. He also believes this will simplify the number of different management products that the company must acquire and receive training for. Dan has heard that SMS supports a multi-purpose software distribution capability. For these reasons, Dan decides to take a closer look at the possibility of using SMS.

In his research, Dan finds the following advantages to using SMS:

- All Windows-based platforms are supported by SMS.
- SMS can monitor its bandwidth usage to ensure that software is not installed over the wide area network (WAN) at inappropriate times.
- Software can be distributed based on user names, group names, computer names, domain names, network addresses, or inventory collection values so it provides a wide range of parameters that can be used to specify computer collections.
- Software can be distributed at a specific time.
- SMS shows the status of application deployments.
- SMS reports if a deployment succeeded or failed.
- SMS provides a record of the existing hardware and software deployed.
- SMS allows administrators to monitor activity to ensure that the correct software is deployed to the correct locations.
- SMS provides a scalable change and configuration solution for centrally managing client computers and servers, which increases productivity and improves service quality within the organization.

Preparing Installation Packages

Sam, the lead developer, is responsible for creating the installation packages. He works with Craig, the application support analyst, to understand any operations constraints, such as making sure that the components that are deployed can run under the data center security policies. Before Sam can let Craig know that the developed and tested Web services are ready to be deployed, he must complete a few packaging and management tasks. One task is to make sure that the desired combination of software components and settings can be bundled together into an installation program that can be executed on the deployed application server. The Microsoft Windows Installer is the recommended technology for packaging run-time executables and their configuration information. Using an .msi file allows Sam to distribute one file that contains all of the files needed for installation as well as the logic needed to complete the installation. The .msi file contains all the file types to be deployed including the following files:

- Executables
- Dynamic-link libraries (DLLs)
- .NET configuration files
- Databases
- Web pages

- Web forms
- Web service files
- Discovery files for Web services
- XML Schema definition (XSD) files

In addition, the .msi file can be created using the Visual Installer component of Microsoft Visual Studio .NET.

To help distinguish the deployment instance that the .msi files are targeted for, the files are named to reflect the name of the Web service that the software component implements, as well as the deployment version of the Web service. For example, a file named ShippingService1001.msi means that the .msi file will install the ShippingService Web service with a unique combination of deployed binaries and settings version labeled 1001.

After the .msi files are created, the files are copied over to a distributed file share directory. Because Northern Electronics relies on the same deployment infrastructure to deploy multiple Web services, it implements the following file share structure to help organize and structure the installation files:

```
\\BinaryDFS\Binaries\ShippingBinaries
\\BinaryDFS\Binaries\TransportBinaries
```

Using the preceding directory structure, all deployment versions of the ShippingService Web service are stored in the **\\BinaryDFS\Binaries\ShippingBinaries** directory.

SMS Software Distribution Concepts

After preparing the installation files, the next step is to configure the Web service deployment policies. Deployment policy can be configured using the SMS 2003 Administrator console. The following SMS concepts closely relate to the elements in Sam's logical designs:

- SMS Client. An SMS client corresponds to the installation agent in Dan's design. The SMS agents can be invoked and controlled through a set of interfaces based on Windows Management Instrumentation (WMI).
- Collection. A collection is a set of computers grouped together because they satisfy one or more rules. For example, a rule can specify that computers that are assigned to be in a certain system role be added to a collection. Another example of a collection membership rule is "all computers within a resource domain." The notion of a collection enables Northern Electronics to logically group its Web servers so that Web services can be deployed on a specified set of computers.
- Package. A package is the basic unit of software distribution and is used to specify the names of programs and files to be installed as well as the server location where those

files can be downloaded. This works well for deploying the Web services because Sam can create a package named ShippingService and use it to organize the different versions of the ShippingService Web service.

- Program. A program is used to indicate the activities that should occur at the client agent when the package is received, such as running a program or copying files to a particular directory. A package can have more than one program associated with it. As previously indicated, this is useful for specifying the program files that are used for installing different versions of a Web service. Each program definition contains a command line instruction that is used to kickoff the software installation process at the client agent. For example, the command line can specify the name of an .msi file that will install and configure the Web service.
- Advertisement. An advertisement is the policy mechanism for associating programs that are defined in packages with the collection of computers where those programs are to be deployed. Deployment actions are then initiated by running an advertisement. After the advertisement runs, the SMS agents on the collection of computers can either be triggered to detect for those advertisements or will poll periodically to find the applicable advertisements. An advertisement can also be associated with a schedule so that it is only available during a certain period of time. This is useful for scheduling multiple planned deployments so that the number of parallel deployment activities is not overwhelming to track and administer.

Finally, all the information related to packages, programs, collections, and advertisements are stored in a SQL database that is set up to store the deployment policies of the SMS site. This means the deployment policies can be accessed and managed in a distributed environment.

Using SMS to Manage Deployment Policies

The SMS 2003 Administrator console is used for managing the deployment policies and for initiating deployment tasks such as running an advertisement and triggering the SMS agents in a server collection to start polling for the latest deployment policies. For Northern Electronics, Dan, the application administrator, creates a package named ShippingWebService to represent the software distribution unit for installing all versions of the ShippingService Web service. Sam views the **ShippingWebService Package Properties** dialog box by right-clicking **ShippingWebService** in the SMS Administrator console and then clicking **Properties.**

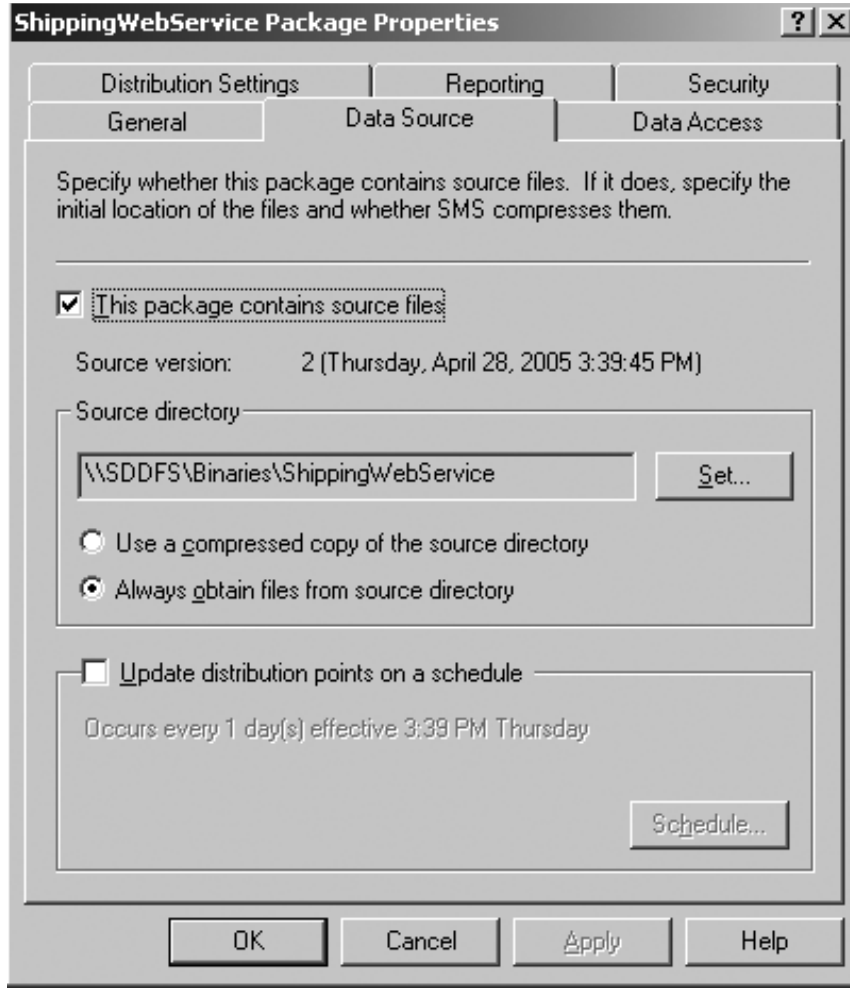


Figure 6. ShippingWebService package properties

Figure 6 illustrates the configuration of the location where installation files for the ShippingWebService package can be downloaded. In this case, the files can be downloaded from the following network location:

```
\\SDDFS\Binaries\ShippingWebService.
```

After the package is created, the .msi files for installing different versions of the ShippingService Web service can be associated with the package. Dan defines the program definition for version 1 of the ShippingService Web service as shown in Figure 6.

To view the Program Properties in the SMS Administrator console (as in Figure 7):

- In the tree control in the left pane, expand the Packages node.
- Expand the ShippingWebService node.
- Right-click Programs, point to New, and then click Programs.

The corresponding .msi file is named Shipping100.msi. Using this scheme, when deployment version 2 of the ShippingService Web service is available, a new program can be created to appropriately reference another .msi file named Shipping200.msi. Each .msi file contains the binary and configuration (for example, registry information) files that will install the ShippingService Web service.

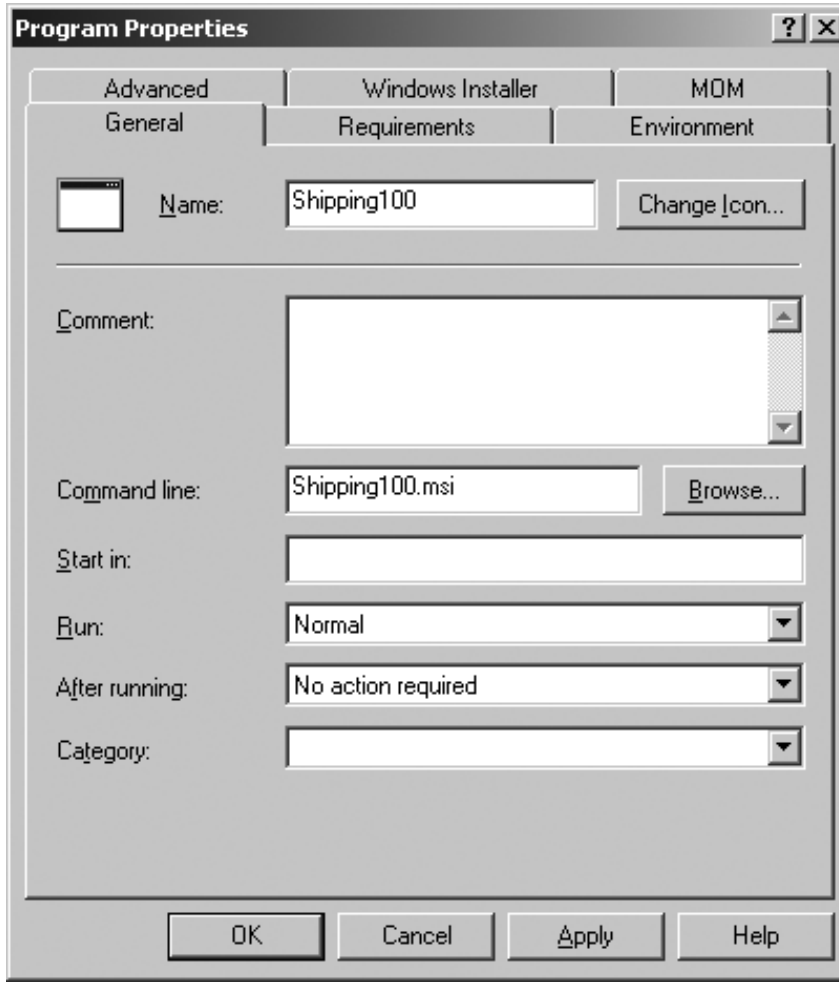


Figure 7. Program properties for version 1 of ShippingService Web service

At the same time, the file directory at the data source is structured in the following way so that the installation programs for all versions of the ShippingService Web service are located within a single directory. The hierarchy is shown below:

```
\\SDDFS\Binaries\ShippingWebService
    \Shipping100.msi
    \Shipping200.msi
    \Shipping300.msi
```

A computer collection named **ShippingWebServiceFarm** contains the group of Web farm computers where the ShippingService Web service will be deployed.

Prior to deploying the ShippingWebService package to the Web service farm, Dan uses the SMS Administrator console to create an advertisement for associating the ShippingService Web service with the computers in the Web service farm. Dan right-clicks **Advertisements,** points to **New,** and then clicks **Advertisement.** The dialog box in Figure 8 is displayed. Dan selects the **Shipping100** program so that version 1 of the ShippingService Web service will be installed for this particular deployment instance.

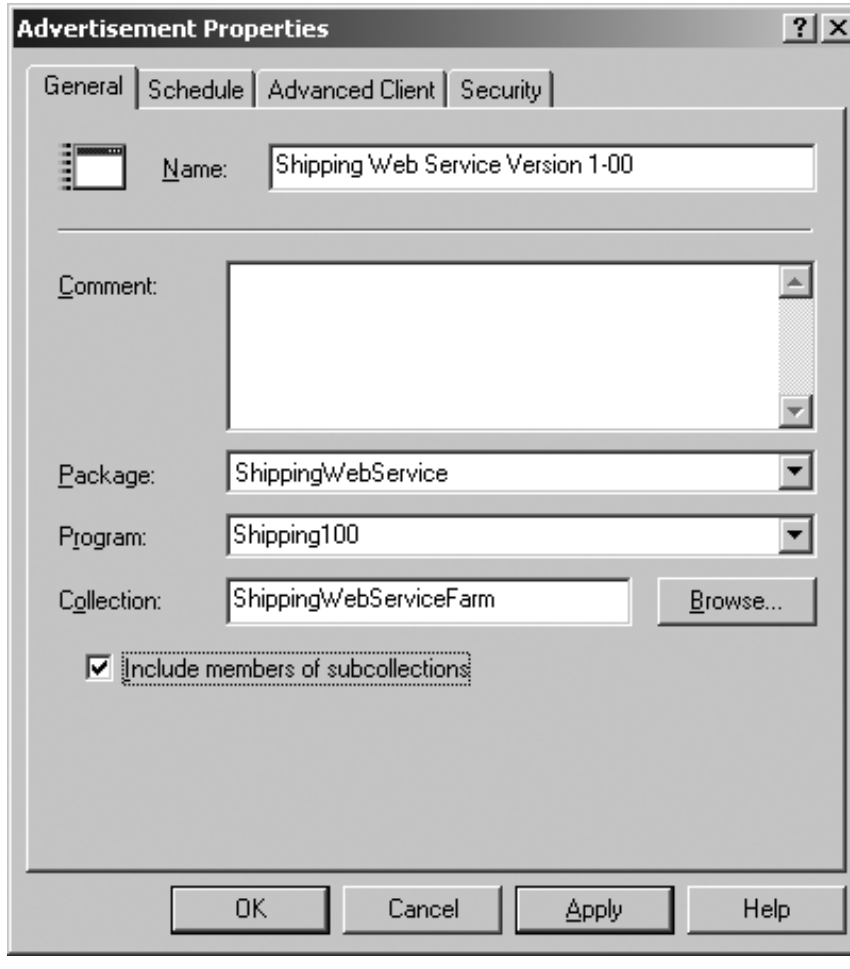


Figure 8. Advertisement associating ShippingService Web service deployment with a collection

If multiple versions of the ShippingService Web service have to be deployed in the same Web service farms at different times, multiple advertisements can be created to capture and kickoff those activities. Table 1 summarizes how the policy configuration may look for this deployment scenario.

Table 1. Policy Configuration

Advertisement	Package	Program	Collection	ScheduleTime
Shipping Web Service Version 1-00	Shipping Web Service	Shipping100	WebFarm Collection001	10:00 A.M. (12 Feb 2005)
Shipping Web Service Version 2-00	Shipping Web Service	Shipping200	WebFarm Collection001	10:30 A.M. (12 Feb 2005)
Shipping Web Service Version 3-00	Shipping Web Service	Shipping300	WebFarm Collection001	11:00 A.M. (12 Feb 2005)

This same approach of defining different advertisements can also be used for configuring the policies of other deployment scenarios where different types of Web services need to be deployed.

Enabling Deterministic Service Deployment

When the Shipping Web Service Version 1-00 advertisement runs, the deployment policy is in effect for the SMS agents that reside on the specified computer collections. By default, SMS agents poll for new software advertisements according to a preset interval that is an SMS site-wide configuration policy. When SMS is used for server-side application deployment, a more instantaneous and deterministic policy-change detection behavior is desirable. This way, administrators can have more control in activating the deployment and tracking the deployment status.

To change the default SMS agent polling behavior, Dan makes the following extensions to the default SMS Microsoft Management Console (MMC) behavior:

A Visual Basic script is used to invoke the TriggerSchedule WMI method at the SMS agents of the computers in the collection. When this method is called, the SMS agent starts polling the SMS server for new policy changes, including software distribution advertisements. The following code sample shows how to invoke the SMS Advanced Client to download the policy:

```
'Connect to the SMS_Client WMI namespace and get the client instance
Set oCCMNamespace = GetObject("winmgmts://WebServer001/root/ccm")
Set oInstance = oCCMNamespace.Get("SMS_Client")
```

```
'After getting the instance, instantiate an instance of the input

parameters 'for its "TriggerSchedule" method
Set oParams = oInstance.Methods_("TriggerSchedule").inParameters.
SpawnInstance_()

'Then set the scheduleID as an input parameter to "TriggerSchedule"
oParams.sScheduleID = "{00000000-0000-0000-0000-000000000021}"

'Finally, execute the "TriggerSchedule" method over the "SMS_Client"
'instance by passing the input parameters to it
oCCMNamespace.ExecMethod "SMS_Client", "TriggerSchedule", oParams
```

To invoke the Visual Basic script from the SMS MMC, a new menu item extension named **Start Software Deployment** is added to the SMS Administrator console so that administrators can select this menu item to invoke the SMS agents at targeted computer collections. To add the new menu item to trigger the Visual Basic script, add the following registry keys:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MMC\NodeTypes\
{3AD39FF1-EFD6-11D0-BDCF-00A0C909FDD7}\Extensions\SMS_Tools]
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MMC\NodeTypes\
{3AD39FF1-EFD6-11D0-BDCF-00A0C909FDD7}\
Extensions\SMS_Tools\Push_Deploy]
"Name"="Start Software Deployment"
"CommandLine"="C:\\WINDOWS\\system32\\cscript.exe
C:\\SMS_Tools\\Push_Deploy\\Push_Deploy.vbs {00000000-0000-0000-0000-
000000000021}
SUB:collectionID SUB:__Server SUB:__Namespace //NoLogo"
```

The **CommandLine** registry key contains the path to the Visual Basic script that executes when the **Start Software Deployment** option is selected. Figure 9 shows that the **Start Software Deployment** menu item can be selected by right-clicking the target computer collection and clicking **SMS Tools.**

```
<a href="Chapter07_F09-large.gif" target="_top"><img src="Chapter07_F09_
thumb.gif" alt="Click here for larger image." border="0"/></a>
```

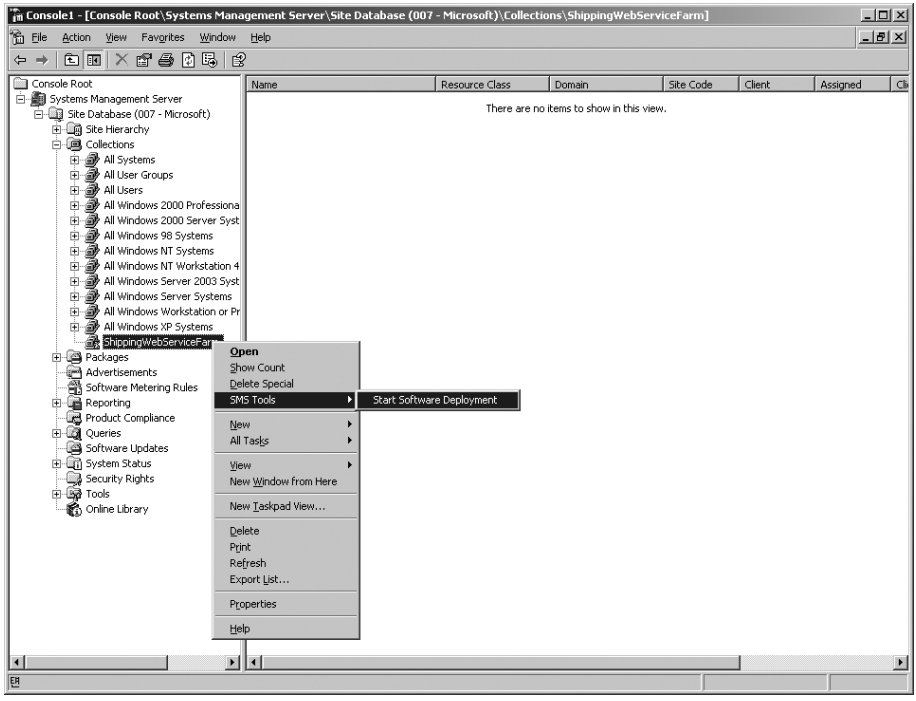


Figure 9. Adding and using new deployment menu item in SMS Administrator console

Tracking Deployment Status

The deployment status for each Web service installation can be tracked at three levels of detail:

- The first level of status tracking is provided through the SMS Administrator console by drilling down into the **Advertisement Status** container under the **System Status** container object. For each of the advertisement status, Craig can view the status of the deployment by looking at the various categories of messages that are reported by the SMS client agents. The message categories represent stages and results of the deployment, such as whether the program has been started and if the program has installed successfully.

- The next level of status tracking is obtained by querying and drilling down into each of the messages that are received from the SMS client agent. For example, Figure 10 shows that by right-clicking the advertisement status, clicking **Show Messages,** and then clicking the **Failures** option, Craig can query for all the received failure messages.

```
<a href="Chapter07_F10-large.gif" target="_top"><img src="Chapter07_F10_
thumb.gif" alt="Click here for larger image." border="0"/></a>
```

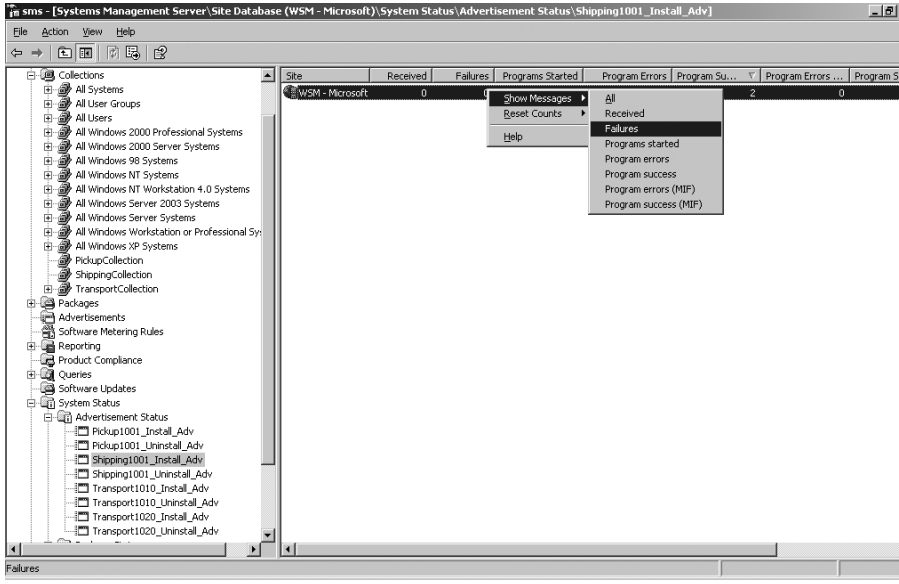


Figure 10. SMS advertisement status

Craig can then look into the details of the messages by double-clicking on the messages in the query results. An example of the **Status Message Details** dialog box appears in Figure 11. This shows the message details when the program indicated by an advertisement fails to deploy at the targeted computer named SMSCLIENT1.

```
<a href="Chapter07_F11-large.gif" target="_top"><img src="Chapter07_F11_
thumb.gif" alt="Click here for larger image." border="0"/></a>
```

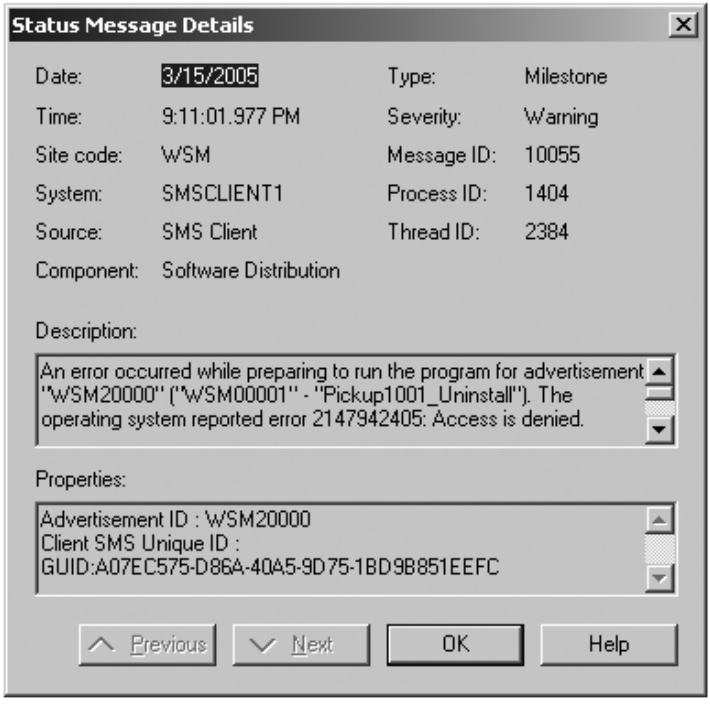


Figure 11. Failure message when installation is aborted

The third level of status tracking has the most detailed history of the installation progress and is provided by the .msi installation log file. Information in the .msi log file is used to diagnose and troubleshoot problems that might have caused the .msi to abort the installation. For the Windows Installer to generate a log file when installing an .msi file, the name and location of the .msi log file is specified through the command line text box in the program property page as shown in Figure 6. For instance, Craig might specify the following program command line for installing version 1-00 of the ShippingService Web service:

```
Msiexec /I \\SDDFS\Binaries\ShippingWebService\Shipping100.msi /qr /L*
\\LogDFS\InstallationLogs\%COMPUTERNAME%_Shipping100_Install_Log.txt
```

The computer named WebServer01 installs the Shipping100.msi and creates a log file named WebServer01_Shipping100_Install_Log.txt in the \\LogDFS\InstallationLogs file share. This scheme centralizes storage of the detail installation logs to a known file share so that the deployment administrators can quickly locate the relevant log file based on the computer name and the program that is being installed.

Conclusions

This document examined Northern Electronics's approach for deploying Web services. Northern Electronics used a deployment modeling approach to specify the software components and configuration settings of different types and versions of Web services into deployment packages; associate different Web services deployment packages with computers in Web service farms; and schedule or manually activate and control when the Web service deployment begins and completes.

Using the SMS 2003 software distribution capabilities, Northern Electronics was able to:

- Group computers in Web farms together into collections.
- Develop packages and programs that defined the various types and versions of Web services to be deployed.
- Use the Windows Installer's .msi file to package the software components and configuration settings for Web services.
- Use the SMS Administrator console and the SMS Advanced Client to download and install .msi files from software distribution points.
- Use the SMS Administrator console and .msi log files to track installation status and troubleshooting setup problems.

Note: Although SMS's software distribution feature is highlighted for deploying Web services, it can be used in a similar fashion to deploy other server-side applications and software components, such as .NET components in n-tier distributed applications.

Additional Information

For information about Windows Installer, see Windows Installer ("windows_installer_start_page") on MSDN.

For information about Microsoft Systems Management Server, see the Microsoft Systems Management Server Web page at http://www.microsoft.com/smserver/.

Notes

0105


CD Pocket