NETWORK AND SYSTEMS MANAGEMENT
Series Editor: Manu Malek

Web Services in the Enterprise

Concepts, Standards, Solutions, and Management

Akhil Sahai
and
Sven Graupner

Web Services in the Enterprise

Concepts, Standards, Solutions, and Management

Web Services in the Enterprise

Concepts, Standards, Solutions, and Management

Akhil Sahai and Sven Graupner
Hewlett-Packard Laboratories
Palo Alto, California

Springer

Library of Congress Cataloging-in-Publication Data

Sahai, Akhil.
Web services in the enterprise : concepts, standards, solutions, and management / Akhil Sahai, Sven Graupner.
p. cm. – (Network and Systems management)
Includes bibliographical references and index.
ISBN 0-387-23374-1 (alk. Paper)
1. Web services. 2. Electronic commerce. I. Graupner, Sven. II Title. III Kluwer Academic/Plenum Publishers network and systems management.

TK5105.88813.S24 2005
006.7'6—dc22

2004062645

Springer Science+Business Media, Inc.
New York, Boston, Dordretch, London, Moscow

ISBN 0-387-23374-1 Printed on the acid-free paper.

Printed in the United States of America

9 8 7 6 5 4 3 2 1 SPIN 11323570

springeronline.com

Dedicated to people who gave me life and who make my life worth living: my parents, Dada, my beloved wife Nina, Daddy and Mama

-Akhil

Dedicated to my parents, friends and colleagues who gave me inspiration to write this book

-Sven

PREFACE

Enterprise IT infrastructure is getting increasingly complex. With the increase in complexity has arisen the need to manage it. Management in general can be seen as the process of assuring that a managed entity meets its expectations in a controlled and predictable manner. Examples of managed entities are not only components, entire systems, processes, but also people such as employees, developers, or operators, and entire organizations. Traditional management has addressed some of these issues in varied manner. The emergence of Web services has added a new complexity to the management problem and poses a new set of problems. But it also adds to the mix a set of technologies that will make the task of management simpler.

Management of Web services will be critical as businesses come to rely on them as a substantial source of their revenue. The book tries to cover the broad area of web services, the concepts, implications for the enterprise, issues involved in their management and how they are being used for management themselves. The book is intended as a reference for current practice and future directions for web services and their management.

The book is directed at:

- Computing professionals, academicians and students to learn about the important concepts behind the web services paradigm and how it impacts the enterprise in general and how it affects traditional application, network and system management.
- Business Managers and Analysts to understand the implications of web services and their management on the enterprise

- System managers to understand the concepts, impact, techniques, and standards that are either available today or are emerging to manage web services
- Researchers, to complement their current knowledge and expertise.
- Network and System Management vendors in understanding where management capabilities are required in the domain of web services and how to go up the management stack.

Acknowledgements

We would like to thank the reviewers (who remained anonymous till the book went into production), Prof. Lundy Lewis, University of New Hampshire, Durham, New Hampshire, USA and Dr. Pradeep Ray, University of New South Wales, Sydney, Australia for the invaluable comments and suggestions that they provided. These comments and suggestions greatly improved the book. We would also like to thank researchers at HP Labs, Palo-Alto who we collaborated and researched some of the web services and management technologies with and who gave us the motivation to write the book. Thanks are due to management team, namely Vijay Machiraju, Mohammed Dekhil, Kumar Goswami and Rich Friedrich for being supportive and encouraging of the whole endeavor.

My deepest gratitude are towards my parents whose immense belief, love and self-less sacrifice has made me what I am today. I would like to thank my wife Nina for her understanding and patience and for putting up with me during writing of the book. I also thank her for the painstaking review she undertook for the book. She has been an immense source of strength and of unconditional love throughout. Acknowledgements are also due to my brother for reviewing and giving me feedback on the book and in general providing me guidance at every step of my life and for being there for me.

Akhil Sahai
June 2004

I would like to acknowledge my parents, friends and colleagues who gave me inspiration to write this book

Sven Graupner
June 2004

CONTENTS

PART III – THE PRACTICE OF WEB SERVICES MANAGEMENT

Chapter 7 – Instrumentation of Web Services

Appendix – Web Services Management Products And Solutions

PART I

WEB SERVICES AND MANAGEMENT

Chapter 1

INTRODUCTION

1. INTRODUCTION

The emergence of web services paradigm brings both complexity as well as a new set of technologies to the enterprise. Just as they complicate the task of traditional enterprise and application management they also simplify it by providing capabilities to virtualize network, system, application resources and thus uniformly manage them. In order to understand web services and the field of Web service management let us define web services and look at the recent market trends and their impact on traditional network and system management.

2. WEB SERVICES: DEFINITION

Web services may be defined as distributed services that are accessible via the Web through Uniform Resource Locators (URLs). With this broad definition, any object that can be invoked using Internet protocols qualifies as a Web service. W3C however has a much stricter definition of Web services. *Web services (as per w3c) are described as distributed services that are identified by URI, whose interfaces and binding can be defined,* *described and discovered by XML artifacts, and that support direct XML message-based interactions with other software applications via internet-based protocols*. For our purpose, we will refer to Web services in both the looser sense (in terms of web sites that are currently used[1]) and in the stricter sense. Web services that perform useful task would often exhibit the following properties:

1. Discoverable: One of the critical requirements for a web-service to provide service to other users. So it has to be discovered by consumers (human users or other Web services).
2. Communicable: This is often asynchronous messaging as opposed to synchronous messaging.
3. Conversational: A conversation involves sending and receiving documents in a context. Complex interactions between Web services involve multiple steps of communication that are related to each other.
4. Secure and Manageable: Security, manageability, availability, and fault tolerance are critical for a commercial web-service.

3. MARKET TRENDS

3.1 Transformations in Enterprise System Management

Enterprises continue turning their applications into services, for their internal operation as well as for external collaboration. Services are made out of applications, which in turn run on machines (systems) distributed over a network. The trend that is appearing fast is that of gradual transition to e-businesses. An e-business infrastructure would comprise a large number of business processes [Leymann 2000] Steps in a business process are handled by either humans (as is the case in work-flow management systems) automated systems, or some times will be outsourced to external providers. As the intention of an e-business is to undertake business on the web, they will also need mechanisms that enable customers to access their services through the Internet. *Web services* [Cerami 2002] are becoming a well-accepted way of putting e-businesses on the web as well as enabling users (either humans or other Web services) to use them. Web service refers to a business or computational function delivered using standard web

[1] We will refer to the current web service implementations that do not correspond to web service standards as web sites

technology, such as HTTP (Hypertext transfer protocol), XML (Extensible markup language) [Pitts 2000], and SOAP (Simple object access protocol) [Box] delivered over the Extra-, Intra- or Internet. According to many market research firms [Gartner 2004] it is likely that, by the year 2006, enterprises that want to remain competitive will need to provide commonly requested data to its partners through web services, and they will transform the IT industry and software professional services in five years. These Web services have to be interfaced with the internal business processes to receive, fulfill and deliver orders. A complex infrastructure is usually a reality in an e-business environment.

Traditionally, the problem of enterprise management has been limited to the areas of network and system management. For example, the IT staff was concerned primarily about the smooth operation of networks and systems. More recently, as the role of IT is transforming from "infrastructure support" to "service provisioning", more and more emphasis is given to the overall application and service management rather than management of bits and pieces of networks and systems. In order to manage the e-business infrastructure, it is necessary to have frameworks and tools that allow business managers/analysts/users to define, measure, and analyze business and IT metrics and to understand their correlation. They also need management systems that can manage business processes and Web services that constitute their own enterprise e-business infrastructure but also manage relationships with other enterprises.

3.2 Emergence of Web Services

Services appeared when the web became established as an infrastructure where people could freely and easily access information by clicking and viewing documents in browsers. Services in general are understood as useful functions that provide value to people. Providing information has been the main service in the web. Other services were adopted from the underlying Internet: email, news, and file transfer. Actual services in the web that have been recognized as services were related to searching and classifying information. The need for such services emerged from continuous growth in amounts of information, this growth has been happening in the web from its very beginning in an uncontrolled and unstructured manner. Generally, targeted search for information has been one of the fundamental services in the web.

Search services in the web quickly expanded into professional information providers where people could find well-classified and high-quality information. At the same time services quickly expanded into businesses where products were advertised and sold, and in the case of digital products, could also be delivered through the web. Payment services were needed to facilitate those businesses.

To summarize, this first initial wave of Web services has been dominated by people, human users interacting through web browsers with information or business providers offering services on the web. The interaction pattern was characterized by human to machine, or client to server, or customer to business or information provider.

Since human users have been drivers and the target audience of first-wave of services in the web, technology underlying those services was focused on access, delivery and presentation of information in web browsers. HTTP, the Hypertext Transfer Protocol has been used to deliver documents across the web. HTML, the Hypertext Markup Language has been used to present web content in browsers.

It soon became obvious that it was desirable using the web as pervasive communication and interaction infrastructure not only for connecting people to information, but also connecting applications to one another. Different requirements occurred. Connecting applications did not require browsers and presenting information suitable for human beings which has been the foundation of HTML. Another set of technologies was needed that could encompass the needs for connecting applications in the open web infrastructure. HTTP, as a transport protocol, continued to be used. XML, the eXtended Markup Language, became the substitute for HTML for connecting applications omitting the presentation capabilities of HTML and introducing a well-defined structure that was suitable and convenient for application interactions.

The second wave of services related to the web, or what is actually being referred to as Web services, is characterized by using HTTP as transport protocol and XML as format for representing and exchanging information between applications in the web infrastructure. Web services have well-defined interfaces, also described in XML. Web services use service registration and discovery facilities for contacting one other in the dynamic and constantly changing environment in the web. Once contact has been established, a set of XML languages exists by which business interactions can be described and conducted.

XML as a technology enabled the interaction between business applications and thus businesses over the web infrastructure. Business to business interaction dominates this second wave of Web services in contrast to customer to business interaction of the first wave. A variety of XML specializations appeared for conducting business functions among business partners. Online market places emerged. Corporations have been organizing their supply chains and financial chains around Web services. Terms used for these electronic business functions are e-Commerce and e-Business.

The important achievement of Web services technology has been the unification of former proprietary protocols and technologies that were used for connecting business applications, and their consolidation into a unified set of technologies characterized by HTTP as transport protocol and XML as representation and exchange format, as well as Java as the preferred implementation language for business functions.

After the first wave had linked people to information and the second wave had linked business applications and business functions together, a third wave is about to appear. This third wave can be seen as linking computational resources together providing access to information and processing capability anytime and anywhere. This third wave of services on the web refers to computational resources and services offered on a web-like connection fabric. This fabric is also referred to as the Grid. Computational grids are established in networks that provide certain minimum capacities in terms of throughput and bandwidth required to make sharing and using computational resources reasonable in a web-like manner. Those networks have been emerging recently providing sufficient capacity for linking computational resources together as well as information about resources offered on the grid and their terms of use. Information about available computational resources may be advertised in the same way as information about products or services offered on the web. Users can locate computational resources on the web exactly the same way they find any other information on the web today. The new quality of computational services on the web is the capability to link users to those resources in such a way that users can use those resources effectively. Using resource from the network requires a certain capacity in networks that has just recently evolved. For this reason, the computational grid (or web) appeared later than the information web and the business web that required less bandwidth for operation.

Several other manifestations and variations of web-services are finding their way into businesses. In general, software systems seem to become services. For instance, Sun offers StarOffice over the Internet, SAP R/3 provides functionality over a portal. Buying and using services over the Internet has become as convenient as buying any product. This trend is also fostered by the convergence of Web services systems with commercial Grid systems that is currently underway.

Commercial Grid systems are inherently service-centric. In the Open Grid Services Architecture (OGSA), all functions that traditionally belonged to an enterprise IT infrastructure are represented as Web services using the same set of technologies introduced above. The notion of virtual organizations emphasizes openness and cross-enterprise connectivity allowing tightly coupled collaboration.

Another consequence of Internet-based services is the huge growth in virtual enterprises and outsourced services. As service providers focus on creating and offering value-added services, they are more and more inclined to outsource non-strategic parts of their businesses to other service providers. Web services enable this trend by facilitating easy composition of external services. The business logic of any given service is no longer confined within the boundaries of a single enterprise. Rather, it is federated across several components that are spread across multiple enterprises.

Another trend is turning entire computational platforms and infrastructures into services offered through the Internet. Utility computing is the keyword here meaning that Web service providers do not need to own the resources where their services are being performed, they may rather rent them on a per-use basis with utility infrastructures in place that automatically adjust service capacities to demand fluctuations. Sustainable business models need to be developed for utility use models as well as management of systems needs to be expanded beyond the current scopes of rather closed enterprise management systems.

3.3 Need for Web Service Management

With the potential and growth of Web services, more and more developers are developing Web services and other internet-based applications. Increasingly, as more enterprises rely on Web services as a new channel for conducting business, management of web-services will become particularly important. The problem of **Web Service Management** is to maintain a service in a desired state by monitoring and controlling its operations and use of resources. Poor performance, lack of availability, crossing pre-determined thresholds, and contract violation are some of the ways in which Web services can deviate from their desired state of operation. Such deviations could occur for various reasons - errors in application or service logic, failures in networks and systems that host these services, improper configuration parameters, and unauthorized intrusion are some examples. A Web service management system both detects and corrects problems in real-time, or it observes the trends in order to predict and rectify the situation before problems occur. Web service management systems can also perform historical analysis of the services they manage and help in planning activities. Further, they can also be integrated into more powerful business-management systems to assess the impact of the services on the overall business.

While the problems of traditional application and service management remain largely unsolved, Web services and business processes add another layer of complexity in terms of management. These additional complexities occur for three reasons: First, they are caused by further automation in enterprises. Some of the tasks that have traditionally been performed by humans such as negotiation and brokering can now be automated. As the number of tasks that are automated through software components increases, so does the complexity in managing them. Second, Web services require more dynamic software composition. Traditionally, a set of software components was statically glued together to create a service. This reduced the flexibility of the overall service. With Web services and business processes, appropriate service components that match a given set of requirements can be dynamically located over the Internet and bound together on demand. This binding can be changed dynamically when better matches are found. Many of the existing management systems are not designed to handle such dynamism. Third, the business logic is federated across several enterprises as portions of the service could be outsourced. The problem of enterprise management has now changed to "cross-enterprise management" performed in infrastructures that follow a utility model of operation.

With all these demands and complexities, there is a strong need for Web service management solutions.

3.4 Growing Potential for Federated Management

Web services management is essentially a problem that pertains to multiple enterprises as they will need to interact across management domains. This will lead to management that can cope with federated environments.

4. MANAGEMENT OF WEB SERVICES

Service management involve functionality that range from both top-to-bottom and end-to-end (Figure 1).

Web services fit nicely into the existing enterprise stack. The existing enterprise stack has IT infrastructure at the bottom. The applications that execute on this IT infrastructure form the next layer. Web services are actually applications with standard interfaces and behavior defined on them.

They effectively form part of the next layer i.e. the service layer. Since Web services are inherently applications, Web service management deals a lot with application management. The business process layer is the next layer. The business process layer deals with how to orchestrate operations from multiple Web services so as to achieve a business objective. These business processes may or may not involve humans. The business processes enable the topmost layer of the organization stack i.e. the business layer to coordinate the services in a particular manner. Undertaking top-to-bottom management of an enterprise involves putting Web service management in perspective. This involves understanding what Web service management means in terms of managing the layers below it and how it relates to the management of layers above it.

Just as Web services fit nicely into the enterprise stack, they introduce new set of problems. They enable enterprises to interact amongst each other. These services may compose with other services from other enterprises. These services have to agree on standard interfaces of describing themselves, discovering others and conducting business with each other. Web service management thus also has to deal with end-to-end management issues.

There are another alternate three dimensions to Web service management. These three dimensions exist irrespective of whether we are managing top-down or end-to-end. These dimensions can be classified, in need of better name, as tasks, metrics, and time dimensions.

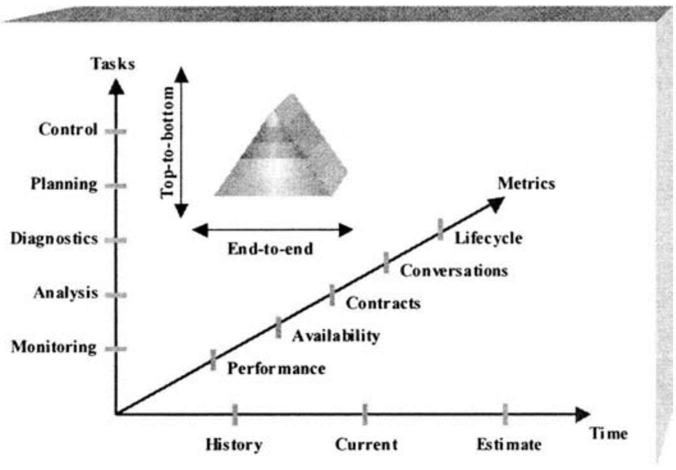


Figure 1: Web services space of tasks, metrics and time.

The time dimension is important because the management data that is collected has to be archived and studied by the management system. Historical archiving of data is important for post fault analysis, diagnostics and analytics and capacity planning. Not only are historical archives important, the current data that is being obtained is extremely relevant. The current data is important for real-time monitoring and control. It is important that the current data be relevant and sampled at the frequency that is relevant. Based on the measurement estimates of the measurements that are relevant for proactive management can be undertaken.

On the tasks dimension, the simplest task is that of monitoring. Monitoring involves invasive and non-invasive instrumentation. Once monitoring data is collected, analytics can be performed on top of the data so collected. Diagnostics involves identifying problems and undertaking root-cause analysis. Planning involves usually short-term and long-term capacity management issues. Control, the ultimate holy grail of management is to automatically detect problems and turn control knobs to correct the problems. Most of the current control mechanisms are human controlled and the management community is trying to automate the process.

The metrics dimension is probably the most important dimension of management as it describes what the quantities that need to be measured are. These metrics relate to performance, availability and reliability at the least. Contracts have to be signed and agreed to between Web services. Contract/SLA management (pg. 228) thus involves the next step of complexity in this dimension. Web services indulge in *conversations* (XML document exchanges in a context). These conversations have to be managed as well. Conversations cross enterprise boundaries and so add further complexity to the metric determination and measurement. These services have a life cycle of their own. They can be brought up, paused, resumed, and brought down in a planned or unplanned manner. Life cycle management is thus an important aspect of the metrics dimension. Undertaking life cycle management necessitates defining management interfaces on the Web services being managed that enable these life cycle operations.

5. SERVICE LIFE CYCLE MANAGEMENT

A typical life cycle associated with Web services involves the

- Service creation: In this phase, Web service interfaces, operations are described, service is designed, and its corresponding implementation undertaken.

- Service provisioning: Once the service has been created, the corresponding resources have to be assigned in terms of servers, software, network bandwidth so that the service can be commissioned. Service provisioning will involve marshalling the requisite resources for putting up the Web service and bringing the Web service into existence.
- Service composition: Sometimes services may have to compose with other services for fulfilling their tasks, for example most of the web sites that undertake travel booking send credit card information to Verisign web site for authorization. This is an example of service composition. Service composition thus may be pre-specified at the service creation time or may be done dynamically by discovering partner services through directory mechanisms, like UDDI (Universal Discovery and Declaration Interface) registries or through marketplaces.
- Service usage: This is the phase when the service actually gets used. Users would start sending requests to the service. The transactions that the users initiate, the resources that the service uses and the performance of the service has to be monitored and corresponding data collected through instrumentation.
- Service Management: Once the management data is collected the information has to be modeled so that service management can be undertaken. This will involve analytics on top of the service data, characterization of performance, Service-Level Agreement monitoring and violation detection, and finally control that may involve changing service design/or service re-creation with new set of requirements that appear because of changed user requirements or monitoring data.

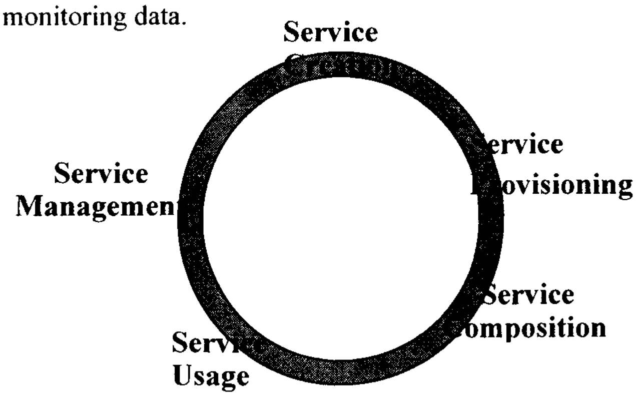


Figure 2: Service life cycle.

6. WHAT TO EXPECT IN THE BOOK

This book describes the concepts and implications on the enterprise because of emergence of web services. It also addresses various aspects related to management of web-services. The main focus of the book is to provide technical details involved in understanding and constructing the role of web services in the enterprise, understanding the role that the multiple standards play, and how to create well-managed web-services and design flexible management systems for managing them.

This book is organized in three parts:

- **Part 1** gives an overview of Web services, the standards and concepts involved. It tries to seamlessly connect them to the domains of traditional application and enterprise management. At the end of this part, the reader will be able to understand what the involved standards and technologies are and how they will interoperate for performing e-business, EAI and Web service to Web service communication. Considering that standards are working at cross-purposes, it is necessary to understand how the different standards on Web services fit together. It will also describe what management in this area means, what the existing standards are, and how it relates to traditional application and enterprise management. The reader will get an understanding of the management space and an understanding of the significant tasks in web-service management, e.g., monitoring the performance, availability, and other SLA/Contract-related metrics, analyzing the collected data, and diagnosing problems.
- **Part 2:** Web services fit in nicely into the enterprise stack. At the lowest layer of the stack is the IT infrastructure. This is followed by the application/Web services layer. The business layer and the business processes layer usually reside on top of the Web services layer. Web service management thus not only has to relate to management of IT infrastructure below but also to management of business processes and the business layer. This part thus deals with putting Web service management in perspective of the overall management and relating it to overall management task both of IT management and that of e-business management.
- **Part 3:** Building a good management solution requires three things – defining the metrics based on the business contracts, properly instrumenting the web-services to generate the required measurements and events based on the defined metrics, and

building the appropriate management system that can use the data for the evaluation of metrics. The focus in Part 3 of the book is to explain the techniques and standards for instrumentation. It describes the existing invasive standards (e.g. ARM, JMX, NSAPI/ISAPI filters) and non-invasive instrumentation technologies (SNMP, logs). Some of the web-services are designed with manageability in mind - others are not. Developers need tools and techniques for placing instrumentation in both cases. This part also deals with how the management systems can be used by business managers/users/IT administrators to track their e-businesses, and how the management system could be used for managing composite web services. It describes the standards that enable web service to web service interactions, especially inter-enterprise business relationship management based on SLA/Contract management with other Web service. This part also describes the recent work in the area of utility computing and grid computing that utilize the approach of using Web services for virtualization of resources that they manage.

Chapter 2

OVERVIEW OF WEB SERVICES

1. INTRODUCTION TO WEB SERVICES

Web services are at the cross point of the evolution paths of service centric computing and World Wide Web. The idea of Web service has been to provide service centric computing by using the Internet as the platform. While services are being delivered over the Internet (or Intranet), the World Wide Web has strived to become a distributed, decentralized all pervasive infrastructure where information is put out for other users to retrieve. It is this decentralized, distributed paradigm of information dissemination that on meeting the concept of service centric computing has led to the germination of the concept of Web services.

1.1 Tightly Coupled Distributed Software Architectures

Despite an early beginning, distributed computing remained mostly in discussions and theoretical dissemination, until the introduction of Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) and Microsoft's Distributed Component Object Model (DCOM). Both CORBA and DCOM vastly simplified object sharing among applications by creating an illusion of a single machine over a network of (heterogeneous) computers. They did so by building their abstractions on middleware layers that are more or less OS- and platform independent.

In these software architectures, objects define a number of interfaces in Interface Description Language (IDL) and advertise their services by registering the interfaces. These interfaces are compiled to create the stubs.

Objects are assigned identifiers on creation. The identifiers are used for discovering their interfaces and their implementations. Sun Microsystems' Java RMI (Remote Method Invocation) provides similar middleware functionality, where a network of platform-neutral Java virtual machines provides the illusion of a single machine. Java RMI is a language-dependent solution though Java Native Interface (JNI) provides language independence to some extent.

1.2 Loosely Coupled Distributed Software Architectures

The notion of service-centric computing has emerged to provide a capability of virtualizing a large number of devices and their functionality through concepts of service and method invocation. For instance, viewing document printing as a printing service provided by a printer can be viewed as a method invocation on a proxy object of a printer.

These services tend to be dispersed over wider administrative area, often straddling domain boundaries for better resource utilization through load balancing and exploitation of locality. Such physical distribution called for more loosely coupled software architectures where scalable advertising and discovery are a must and low-latency, high-bandwidth inter-processor communication is highly desired. Specifying and enforcing security policies as well as protecting data themselves at every corner of a system can no longer be put off.

A slew of service-centric middleware developments have come to light. We note three distinctive systems from computer industry's research laboratories, namely, HP's client utility (e-speak), Sun Microsystems' Jini, and IBM's T-spaces (here listed in the alphabetic order). These have been implemented in Java for platform independence.

1.3 Client Utility System (E-Speak)

HP's client utility system is a less-publicized effort that eventually led to HP's e-Speak platform. It was one of the earlier forms of the peer-to-peer system meant for Web service registration, discovery and invocation (Kim, 2002). The fundamental concept is to visualize every element in computing as a uniform representation called "service (or resource)". Using the abstraction as a basic building block, it provides facilities for advertising and discovery, mediation and management, fine-grain capability-based security and dynamic service composition. In client utility as opposed to other such efforts, advertisement and discovery were made visible to clients. Clients can describe their services using vocabularies and can specifically state what services they want to discover. Vocabularies are a set of attributes that can be parameterized by the advertising services. The services compose with each other after discovering partner services through such registries.

1.4 T-Spaces

IBM's T-Spaces (T-Spaces 1999) is a middleware that intends to enable communication amongst devices and applications in a network comprising of heterogeneous computers and operating systems. It is a network communication buffer with database capabilities that extends Linda's Tuple space communication model with asynchrony. T-Spaces support hierarchical access control at the Tuple space level. Advertisement and discovery are implicit in T-Spaces and provided indirectly through shared Tuple spaces.

1.5 Jini

Sun Microsystems Jini technology is a set of protocol specifications that allows services to announce their presence and discover other services in their proximity. It enables a network-centric view of computing. However, it relies on the availability of multicast capability that limits its applicability to services/devices connected with a local area network (such as home network). Jini exploits Java's code mobility and allows a service to export stub code implementing a communication protocol using Java RMI. Joining, advertisement, and discovery are done transparently from other services. It has been developed mainly for collaboration within a small, trusted workgroup and offers limited security and scalability supports.

2. THE STATE OF THE ART IN WEB SERVICES

Web services are becoming more and more prevalent. They are emerging as e-business related web sites and portal sites. Some of these portal sites have pre-negotiated understanding with a set of Web services that they provide a front-end for. For example the travel related portal sites, are statically composed Web services that have pre-negotiated understanding with certain airlines and hotels and broker their services. These are mostly Business-to-Consumer (B2C) kind of Web services with some amount of static B2B interactions happening between the Web services. A large number of platforms and technologies are emerging and are being standardized upon so as to enable the paradigm of Web services, for satisfying B2C and Business-to-Business (B2B) scenarios in a uniform manner. These standards and platforms enable creation and deployment, description, discovery and communication amongst them.

A standard that has emerged for Web services to describe their interfaces is Web Services Description Language (WSDL). These interfaces can then be published at registries together with information about services' access points (i.e., bindings), both of which are described in the WSDL which is an XML-based description language. These WSDL descriptions are advertised at registries. These registries may use the Universal Description, Discovery and Integration (UDDI) technology. UDDI has been standardized as a technology for publishing and discovering service related descriptions. After having discovered its partners, web-services use the asynchronous communication model to exchange documents, and Simple Object Access Protocol (SOAP) for service invocation (which is an incarnation of remote procedure call (RPC) in XML) over hypertext transfer protocol (HTTP). Most services are implemented using platform independent languages such as Java and C# on platforms like J2EE and .Net. The primary means for enforcing security are digital signature and strong encryption with public-private key pairs. A large number of payment mechanisms are being defined as well.

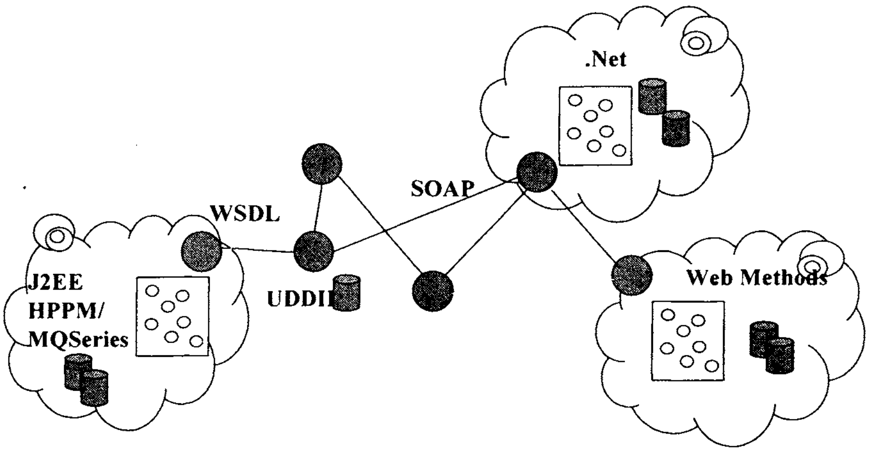


Figure 3: Interacting Web services.

2.1 Web Services Description

In the middleware platforms that we discussed in tightly coupled software architecture sub-section, software component interfaces are defined through Interface Definition Languages (IDL). The interfaces describe the methods/operations that the software component supports, their expected outputs, and the input parameters. The actual implementation just has to follow these interfaces. This enables the interfaces to be decoupled from the actual implementation. As Web services are envisaged as software available on the web that other Web services or users will use, they need to be described so that other components can easily use them without coupling the interfaces with the implementations. Web Services Description Language (WSDL) is an attempt to describe the Web service interfaces.

Emerging standards such as Web Services Definition Language (WSDL) are creating flexible and generic interaction models for Web services. WSDL enables description of Web services irrespective of the message formats and network protocols used. For example, in WSDL a service is implemented through a set of endpoints. An endpoint is in turn a set of operations.

An operation is defined in terms of messages that they receive and send:

- Message – an abstract definition of data being communicated consisting of message parts,
- Operation – an abstract definition of a method supported by the service. Operations are of the following type namely, one-way, request-response, solicit-response, and notification,

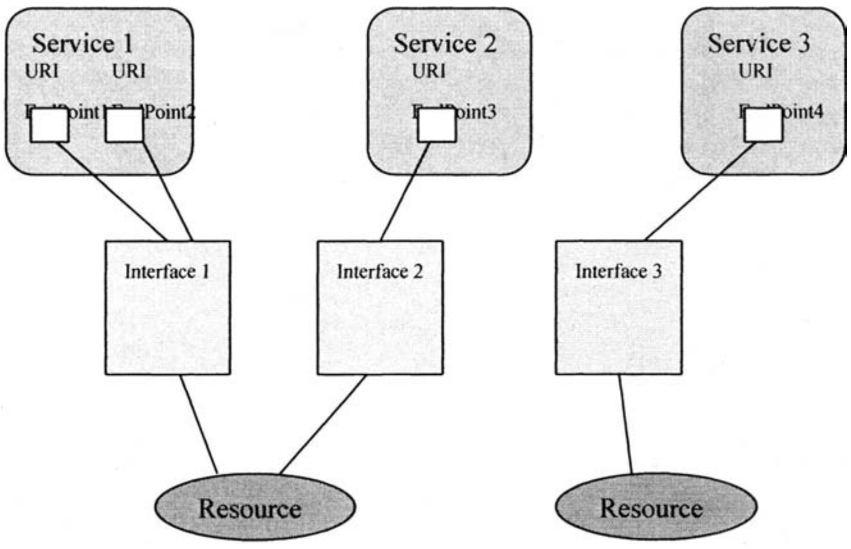


Figure 4: Service abstractions built upon resources.

- Port type – an abstract set of operations that are supported by one or more end points,
- Binding – a concrete protocol and data format specification for a particular port type,
- Port – a single end point defined as a combination of a binding and a network address,
- Service – a collection of related end-points bound to the same interface.

Here's an example of a WSDL document for a StockQuote service. The simple example demonstrates how a port type can be defined comprised of operations that in turn are made up of messages.

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://book.example.com/stockquote/definition
s"

xmlns:tns="http://book.example.com/stockquote/definitions"

xmlns:xsdl="http://book.example.com/stockquote/schemas"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import
namespace="http://book.example.com/stockquote/schemas"
location="http://book.example.com/stockquote/stockquote.xsd"/>

    <message name="GetLatestTradePriceInput">
        <part name="body" element="xsdl:TradePriceRequest"/>
    </message>

    <message name="GetLatestTradePriceOutput">
        <part name="body" element="xsdl:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLatestTradePriceInput"/>
           <output message="tns:GetLatestTradePriceOutput"/>
        </operation>
```

Figure 5: Example of a WSDL Web services definition for the StockQuote service.

2.2 Web Services Discovery

Before Web services can interact with each other, they need to discover other compatible Web services with which they can undertake business.

2.2.1 E-Marketplace

An E-MarketPlace is such a virtual meeting place managed by an intermediary supplying a series of benefits to participating Web services:

- Enabling registration and discovery of Web services thus
- Enabling inter Web service interaction with or without the direct participation of the e-marketplace in the actual interaction after the discovery.
- Enabling supply and demand mechanisms like traditional catalogue purchasing, RFP or through more dynamic auctions, exchanges.
- Providing value added services, such as rating, secured payment, financial handling, certification services, notification services etc; and,
- Supply-chain management through collaborative planning and inventory handling.

Vertical and horizontal E-MarketPlaces are coming up in this regard. VerticalNet, GlobalNetXchange, and Retailers Market Exchange are examples targeting a specific section of the industry with key players performing B2B transactions. Other examples like Chemdex, E-Steel and DirectAg.com have been successful in their respective industries. Horizontal exchanges in turn are directed at a broad range of players such as e-Bay targeting a range of clients and businesses.

2.2.2 UDDI

Universal Description Discovery Integration (UDDI) is an initiative supported by IBM, Microsoft, and HP. It is a group of web-based registries (operator sites) maintained by these organizations that expose information about a business and its technical interfaces and APIs by data structures:

- tModel,
- businessEntity,
- business service,
- binding template.

It also exposes a set of APIs for inquiry and publication of information related to Web services. The inquiry APIs enable browsing of the information in the repository site (e.g. find_business) and also to drill down (e.g. get_businessDetail). The publication APIs are for publishers to put their information on these repositories.

2.2.3 Web Service Inspection Language

The Web service Inspection Language (WSIL) is a complementary effort to UDDI, for enabling discovery and aggregation of Web services. UDDI implements service discovery using a centralized model of one or more repositories containing information on multiple business entities and the services they provide. WSIL provides service discovery in a decentralized fashion, where service description information can be distributed to any location using a simple extensible XML document format. WSIL assumes that the service provider is already known, and relies on other service description mechanisms such as the Web Services Description Language (WSDL) for describing information about the services that the service provider makes available. In that context it is similar to RDF Site Summary (RSS) for Web services.

```
<?xml version="1.0" encoding="UTF-8"?>
<inspection xmlns=
"http://schemas.xmlsoap.org/ws/2001/10/inspection">
<service>
 <name>Stock Quote Service
 </name>
<description
referencedNamespace="http://schemas.xmlsoap.org/wsdl/"

location="http://book.example.com/stockquote.wsdl">
</description>
</service>
```

Figure 6: Example XML document for Inspection.

The convention is that a Web services makes available a document called inspection.wsil at its address. The wsil file has description tagsets defined in them that contains all of the information necessary to discover a specific service description. The description tag uses a required attribute of referencedNamespace to indicate the namespace of the service description. The description also uses an optional location attribute to define a direct reference to the service's description. A service block can have multiple description tags to provide consumers different service description options. For instance a service could define references to a WSDL file, a UDDI repository entry, and an HTML document.

2.3 Web Services Choreography

Web services provide a front-end for the internal business processes of an enterprise. They help expose subsets of the internal business processes and activities so that external business processes may compose with them. This leads to what is termed an orchestration or choreography. The enterprise business processes have to be defined and some of their activities have to be linked to the WSDL operations. This requires modeling of Web service's back-end business processes.

2.3.1 Web Services Flow Language

WSFL introduces the notion of activities and flows – which are useful for describing both local business process flows and global flow of messages between multiple Web services. XLANG is another Microsoft technology that provides a mechanism for process definition and global flow coordination.

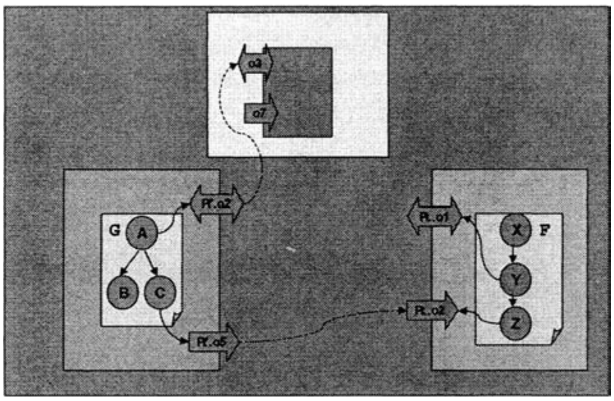


Figure 7: Workflow between Web services.

Web services Flow Language (WSFL, 2001) conceptualizes business processes as set of activities and links. An activity is a unit of useful work. The links could be control links where decisions are made to follow one activity or another, or data links where data is fed into an activity from another. These activities could be exposed through one or more operations that are grouped through end-points (as defined in WSDL). A service is comprised of a set of end-points. A service provider can provide multiple services. Just like internal flows, global flows can be defined. Global flow consists of plug links that link up operations of two service providers.

2.3.2 XLANG

Microsoft's XLANG defines services by extending WSDL. The extension elements describe the behavioral aspects. A behavior spans multiple operations. A behavior has a header and a body. An Action is an atomic component of a behavior. The action elements could be an *operation*, a *delay* element or a *raise* element. A delay element could be of types delayFor and delayUntil. The delayFor and delayUntil introduce delays in the execution of the process to either wait for something to happen (for example a timeout) or to wait till an absolute date-time has been reached respectively. Exceptions are flagged through raise constructs. Raise handle the exceptions by calling the handlers registered with the raise definition. Processes combine actions together in useful ways. A process form could be a sequence, switch, while, All, Pick, Context, Compensate, and Empty.

2.3.3 BPEL4WS

Business Process Execution Language for Web services combines the WSFL and XLANG capabilities. It is an attempt to standardize business process language. A single BPEL4WS process describes the global process that links multiple Web services. Entry-points are defined in the BPEL4WS specification of a global process. These entry points either consume WSDL operations' incoming messages from input-only or input-output operations.

BPEL4WS only utilizes input-only and input-output (request-response) operations of WSDL. BPEL4WS does not require or support output-only (notification) and output-input (solicit-response) operations.

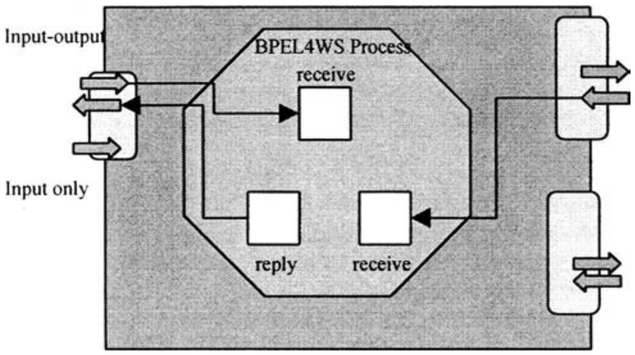


Figure 8: BPEL4WS business interaction.

The BPEL4WS process itself is comprised of activities. There are a collection of primitive activities: invoking an operation on a Web service (<invoke>), waiting to receive a message to operation of the service (<receive>), creating a response to an input/output operation (<reply>), waiting for some time without doing anything (<wait>), indicating an error (<throw>), copying data from one place to another (<assign>), closing the entire service instance down(<terminate>), or doing nothing through (<empty>). These basic primitives may be combines through (sequence), branching through (switch), define loops (while), execute one of the chosen paths (pick), executing activities in parallel (flow). Within activities executing in parallel, one can indicate execution order constraints by using the *links*. BPEL4WS provides the capability to combine activities into complex algorithms that may represent Web service to Web service interactions.

2.3.4 ebXML

In order to enable e-business on the web, Web service to Web service interactions need to follow certain business protocols. X-EDI, ebXML, BTP, TPA-ML, cXML, CBL are some of the B2B technologies that have been proposed to enable this paradigm with Web services.

In ebXML (ebXML 2001) the parties that undertake business have Collaboration Protocol Profiles (CPP) that they register at ebXML registries. A GUID is assigned to each CPP by the ebXML registry. Discovery mechanisms are used to find out a compatible party's CPP. Once a party is found they form a Collaboration Protocol Agreement (CPA). CPAs are formed after negotiation between the parties. The intent of the CPA is not to expose business process internals of parties but to expose the visible process that involves interactions between parties. The messages exchanged between the Web services may utilize ebXML Messaging Service (ebMS). The CPA and the process specification document it references define a *conversation* between parties. This conversation involves multiple *Business Transactions.* A Business Transaction may involve exchange of request reply messages The CPA may refer to multiple process specification documents. Any one conversation will involve only a single process specification document however. Conceptually, the B2B server at the parties is responsible for managing the CPAs and for keeping track of the conversations. It also interfaces the functions defined in the CPA with the internal business processes. The CPP contains the following:

- Process specification Layer: This details the business transactions that form the collaboration. It also specifies the order of business transactions.
- Delivery Channels: describes party's message receiving and sending characteristics. A specification can contain more than one delivery channels.
- Document Exchange Layer: deals with processing of the business documents like digital signatures, encryption, and reliable delivery.
- Transport layer: The transport layer identifies the transport protocols to be used the end point addresses, along with other properties of the transport layer. The transport protocols could be SMTP, HTTP, and FTP.

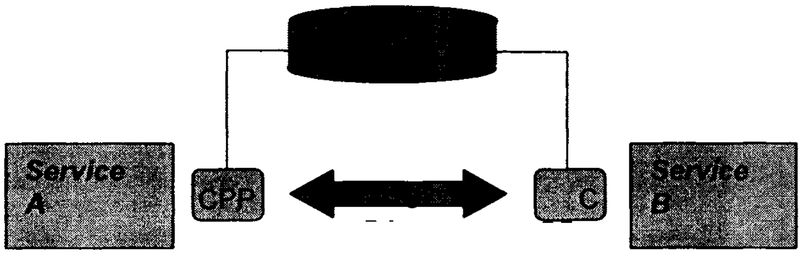


Figure 9: ebXML registry between two Web services.

3. WEB SERVICES MANAGEMENT

Besides the foundations of Web services that have been laid out by W3C in terms of XML and SOAP, further standardization efforts are needed. As a relatively new field, Web services management is subject to standardization by various organizations including the Organization for the Advancement of Structured Information Standards (OASIS), Distributed Management Task Force (DMTF), Object Management Group (OMG), and the Global Grid Forum (GGF).

Focus is on the manageability for the components of the Web Services Architecture (WSA) as defined by the W3C. It includes a model of a Web service as a manageable resource. The Global Grid Forum is another standards body proposing OGSA Common Resource Model.

OASIS has a proposal in a Web Services Distributed Management (WSDM) working group that is discussing WSDL described manageable resources and the XML schema to complete those descriptions. The specification will also define manageability for the components of the Web Services Architecture by the W3C.

A Grid Service is a Web service that conforms to a particular set of interfaces and behaviors for a client. A Grid service includes entities as hardware components, software components, complete applications or transient items.

The Open Grid Services Architecture (OGSA) Platform (Pg. 257) defines the core set of interfaces, behaviors, and bindings that define Grid Services. The OGSA Platform is currently under development by the Global Grid Forum (GGF) OGSA working group. The Web Services Resource Framework (WSRF) effort at GGF is looking at providing the lower level services that are required to enable OGSA.

Two standards are discussed in more detail in the following:

- WSRF, the Web Services Resource Framework and
- WSDM, the Web Services Distributed Management standardization effort.

3.1 WSRF – The Web Services Resource Framework

The Web Services Resource Framework (WSRF) [WSRF 2004] represents a comprehensive set of Web services technologies that are standardized at the time of the writing of this book (June 2004). It can be assumed that WSRF will continue to evolve and change.

3.1.1 History

Web services primarily emerged in the Web and XML communities (W3C). Another stream of technology emerged from the Grid computing community. Initially targeting interconnecting and sharing large compute facilities, the notion of sharing compute, storage and network resources had been broadened beyond supercomputing around the year 2001. Pioneers in Grid computing such as Ian Foster, Carl Kesselman, Jeffrey Nick, and Steven Tuecke formulated the vision of an Open Grid Services Architecture (OGSA) in their paper: *The Physiology of the Grid* which appeared in 2002 [OGSA 2002]. OGSA was a significant extension to prior views and definitions of Grid as presented in Foster and Kesselman's book: *The Grid: Blueprint for a New Computing Infrastructure*, published in 1999 [Foster 1999].

OGSA introduced the notion of a Grid Service as unit of providing computational, storage or transmission services. OGSA introduced a unified view on these services from elementary, base resources up to application services. The motivation for OGSA was defined around an increasing need for collaboration, service-orientation, and the formation of so-called virtual organization. Those notions were widely applicable and not bound to scientific computing.

The Global Grid Forum (GGF) has been the organizational body for defining Grid technology for years. With the formulation of OGSA, GGF also made a move towards adopting more recent Web services technology, away from prior, proprietary formats and protocols used in Grid implementations such as the Globus Toolkit up to version 1.2. The Globus Toolkit provided the free, open-source reference implementation of Grid definitions and was developed by the Globus Alliance.

However, the status of Web services technologies did not appear sufficient as infrastructure for OGSA in 2002. Major properties such as events, security support, or addressing were not finalized. For this reason, extensions have been introduced that were specific to "Grid Web services" over other Web services. Those specific properties had been defined in the Open Grid Services Infrastructure (OGSI) [OGSI 2002] and were meant to be used only for the time until Web services standards had matured.

Web services definitions had progressed after 2002 in standards bodies, primarily in W3C and OASIS, such that OGSI was declared obsolete at Globusworld in January 2003 and succeeded by a new set of Web services infrastructure definitions that were introduced as *Web Services Resource Framework* (WSRF). WSRF adopted and merged the different Web services standards and definitions into a single framework overcoming the differences between "Grid Web services" and "non-Grid Web services".

3.1.2 Web Services and Service Oriented Architecture

WSRF adopts the definition of Web services from the W3C Web Services Architecture working group [W3C-WSA 2003]:

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAPmessages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

This definition can be seen as currently the most comprehensive and most widely accepted definition for Web services.

In [Burbeck 2000], the term Service Oriented Architecture is introduced:

A Service-Oriented Architecture (SOA) is a specific type of distributed system in which the agents are "services." A service is a software agent that performs some well-defined operation (i.e., "provides a service") and can be invoked outside of the context of a larger application. That is, while a service might be implemented by exposing a feature of a larger application ... the users of that server need be concerned only with the interface description of the service. "Services" have a network-addressable interface and communicate via standard protocols and data formats.

3.1.3 Stateful Resources and Stateless Web Services

Based on these definitions, WSRF build another model where it distinguishes between stateful resources and stateless Web services, or between stateful resources behind stateless Web services interfaces [WSRF-Model 2004].

WSRF provides the following brief explanation of associations of state with an interface:

1.) A stateless service implements message exchanges with no access or use of information not contained in the input message. A simple example is a service that compresses and decompresses documents, where the documents are provided in the message exchanges with the service.

2.) A conversational service implements a series of operations such that the result of one operation depends on a prior operation and/or prepares for a subsequent operation. The service uses each message in a logical stream of messages to determine the processing behavior of the service. The behavior of a given operation is based on processing preceding messages in the logical sequence. Many interactive Web sites implement this pattern through use of HTTP sessions and cookies.

3.) A service that acts upon stateful resources provides access to, or manipulates a set of logical stateful resources (documents) based on messages it sends and receives.

3.1.3.1 Stateful Resources

The term state can encompass many different aspects of a computation or a computer system. WSRF describes a stateful resource [WS-Resource 2004] by:

- having a specific set of state data expressible as an XML document;
- having a well-defined lifecycle; and
- known to, and acted upon, by one or more Web services.

Examples of system components that may be modeled as stateful resources are files in a file system, rows in a relational database, and encapsulated objects such as Entity Enterprise Java beans. A stateful resource can also be a collection or a group of other stateful resources.

Multiple instances of a stateful resource type may exist. An instance of a stateful resource may be created via a Web service referred to as a stateful resource factory. A stateful resource is defined by a single XML Global Element Declaration in a given namespace. This Global Element Declaration defines the type of the root element of the resource's XML document and hence the type of the stateful resource. When a stateful resource instance is created, it may be assigned an identity by the entity that created it. Applications using the resource may assign the resource additional identities. A specific form of stateful resource identity may be used privately by one or more Web service implementations to identify the stateful resource used in the execution of a message exchange.

3.1.3.2 The Implied Resource Pattern

Another concept in WSRF is the Implied Resource Pattern defining how stateful resources are referred to by Web services clients. An Implied Resource Pattern describes a specific kind of relationship between a Web service and one or more stateful resources. The implied resource pattern refers to the mechanisms used to associate a stateful resource with the execution of message exchanges implemented by a Web service. The term implied is used because the stateful resource associated with a given message exchange is treated as an implicit input for the execution of the message request. The requestor does not provide the stateful resource identifier as an explicit parameter in the body of the request message. Instead, the stateful resource is implicitly associated with the execution of the message exchange. This can occur in either a static or a dynamic way. A stateful resource is associated with the Web service statically in the situation where the association is made when the Web service is deployed. A stateful resource is dynamically associated with the Web service when the association is made at the time of a message exchange. In the dynamic case, the stateful resource identifier also designates the implied stateful resource and may be encapsulated in the WS-Addressing endpoint reference (see section on WS-Addressing below) used to address the target Web service at its endpoint.

The term pattern indicates that the relationship between Web services and stateful resources is embodied by a set of conventions on existing Web services technologies, in particular XML, WSDL, and WS-Addressing.

3.1.4 WSRF Definitions

At the time of this writing, WSRF includes the following base definitions:

- WS-Resource and WS-Addressing,
- WS-Resource Properties,
- WS-Resource Lifecycle,
- WS-Notification,
- WS-Service Group, and
- WS-Base Faults.

Those definitions are expected to continue changing. The following selection provides some more detail on those definitions.

3.1.4.1 WS-Resource and WS-Addressing

When a stateful resource is associated with a Web service and participates in the implied resource pattern, WSRF refers to the component resulting from the composition of the Web service and the stateful resource as a WS-Resource. A WS-Addressing endpoint reference is an XML serialization of a network-wide pointer to a Web service. This pointer may be returned as a result of a Web service message request to a factory to create a new resource or it is returned from the evaluation of a search query in a registry, or as a result of some other application-specific request.

WS-Addressing standardizes the endpoint reference construct used to represent the address of a Web service deployed at a given network endpoint. An endpoint reference may contain, in addition to the endpoint address of the Web service, other metadata associated with the Web service such as service description information and reference properties, which help to define a contextual use of the endpoint reference. The reference properties of the endpoint reference play an important role in the implied resource pattern. The endpoint reference contains information that expresses the implied resource pattern relationship between the Web service and the newly created stateful resource.

The endpoint reference contains two important components:

- The wsa:Address component refers to the network transport-specific address of the Web service (often a URL in the case of HTTP-based transports). This is the same address that would appear within a port element in a WSDL description of the Web service.
- The wsa:ReferenceProperties component may contain an XML serialization of a stateful resource identifier, as understood by the

Web service addressed by the endpoint reference. The stateful resource identifier represents the stateful resource to be used in the execution of the request message.

The XML serialization of the stateful resource identifier uses a service-specific XML element to represent the stateful resource identifier information that is opaque to the service requestor. The service requestor's applications should not interpret the contents of the stateful resource identifier. The stateful resource identifier is meaningful only to the Web service, and is used by the Web service in an implementation-specific way to identify the WS-Resource related stateful resource needed for the execution of the request message.

From the point of view of the service requestor, the endpoint reference represents a pointer to the WS-Resource, composed of a Web service that may be further constrained to execute message exchanges against a specific stateful resource. It is assumed that the service requestor understands that the endpoint reference refers to a WS-Resource.

3.1.4.2 WS-Resource Properties

The WS-Resource Properties specification [WS-ResourceProperties 2004] defines the type and values of those components of a WS-Resource's state that can be viewed and modified by service requestors through a Web services interface. This implies that a WS-Resource has an XML resource property document defined using XML schema. It also implies that a service requestor may determine a WS-Resource's type by retrieving the WSDL portType definition by standard means, or a service requestor may use Web services message exchanges to read, modify, and query the XML document representing the WS-Resource's state. The underlying state may be in any or multiple formats, and in a single or multiple locations.

3.1.4.3 WS-Resource Lifecycle

The lifetime of a WS-Resource is defined as the period between its creation and its destruction. The actual mechanisms by which a specific WS-Resource is created and destroyed are implementation-specific. WSRF addresses the following three aspects of the WS-Resource lifecycle:

1. WS-Resource Creation – through the use of a WS-Resource factory,
2. WS-Resource Identity – the assignment and use of the stateful resource identifier, and
3. WS-Resource Destruction – the end of existence of a WS-Resource.

WS-Resource Creation. A WS-Resource can be created by some external mechanism, or through the use of a WS-Resource factory. A WS-Resource factory is any Web service capable of bringing a WS-Resource into existence by creating a new stateful resource, assigning the new stateful resource an identity, and creating the association between the new stateful resource and its associated Web service. The response message of a WS-Resource factory operation contains a WS-Resource endpoint reference containing a stateful resource identifier that refers to the new stateful resource. A factory may also place the WS-Resource endpoint reference into a registry for later retrieval. There are many types of Web services that may return WS-Resource endpoint references in their response messages.

WS-Resource Identity. WSRF describes the role and use of WS-Resource identity from two perspectives:

1. from the private perspective of the WS-Resource implementation, and
2. from the public perspective of a service requestor to whom an endpoint reference to a WS-Resource is provided.

Each stateful resource has at least one form of identity that uniquely identifies a stateful resource component within the WSResource composition. This identity may be used as a "stateful resource identifier" which has a specific role as a component in a reference to the WS-Resource. The stateful resource identifier is placed into the reference properties portion of a WSAddressing endpoint reference. A WS-Resource endpoint reference can then be made available to other entities in a distributed system, which can subsequently use that endpoint reference to direct requests to the WS-Resource.

WSRF defines an identity as a portable, namespace-scoped value. Portability is important as it allows one application to pass the identity to another. Namespace scoping is important since it allows disambiguation of multiple identities that may originate from different sources.

WS-Resource Destruction. A requestor that sends a message request to a WS-Resource factory that causes the creation of a new WS-Resource will typically only be interested in that new WS- Resource for some finite period. After that time, it should be possible to destroy the WS-Resource so that its associated system resources can be reused.

WSRF does not define specific interfaces supporting the destruction of WS-Resources. It describe only general requirements such as a service requestor may use the appropriate WS-Resource endpoint reference to send a destroy request message to the Web service when this Web service supports a destroy operation. The receipt of the response to the destroy request message can be used for synchronization between the service requestor and the Web service receiving the destroy message. Upon receipt of the response message, any further message exchanges with the service will result in fault messages.

The WSRF definition also mentions message exchanges for establishing and renewing scheduled destruction times on WS-Resources for time based destruction when a WS-Resource cannot be destroyed explicitly.

3.1.4.4 WS-Notification

Another family of specifications is called WS-Notification [WS-Notification 2004]. It defines a general, topic-based mechanism for publish and subscribe (pub/sub) interactions building on the WS-Resource framework. The basic approach is to define mechanisms and interfaces that allow clients to subscribe to topics of interest, such as resource property value changes for a WS-Resource. From the perspective of WS-Notification, the WS-Resource framework provides useful building blocks for representing and structuring notifications. From the perspective of the WS-Resource framework, the WS-Notification family of specifications extends the utility of WS-Resources by allowing requestors to ask for asynchronous notifications sent by to the subscriber when changes to resource property values occur.

WS-Notification includes three sub-definitions:

- WS-BaseNotification,
- WS-Topics, and
- WS-BrokeredNotification.

WS-BaseNotification describes the basic roles, concepts, and patterns required to allow a subscriber to register interest in receiving notification messages from a notification producer. A notification can concern a change in the value of a resource property, some other internal change in the state of the notification producer, or some other situation within the environment that can cause a notification to be sent. A subscriber registers interest in receiving notification messages on one or more *topics* by issuing a subscribe message. In response, the subscriber receives a WS-Resource endpoint reference to a "subscription" WS-Resource. The subscription WS-Resource models this relationship between the subscriber and the producer. It uses WS-ResourceProperties and WS-ResourceLifetime to manage this relationship.

WS-Topics presents an XML description of topics and associated meta data. Topics are a mechanism known from other publish-subscribe systems such as Java Messaging Service (JMS) for organizing notification messages so that subscribers can conveniently understand what types of notification are available for subscription. Topics can be organized hierarchically. One topic can be further decomposed with child topics. Topics are scoped by namespaces similarly like XML types and elements are scoped by XML namespaces.

WS-BrokeredNotification defines the interface to a NotificationBroker that implements an intermediary service to manage subscriptions for other entities in the system that produce notification messages.

3.1.4.5 WS-Service Group

The WS-ServiceGroup specification defines means for representing and managing heterogeneous by-reference collections of Web services. This specification can be used to organize collections of WS-Resources, for example to build registries, or to build services that can perform collective operations on a collection of WS-Resources. The WS-ServiceGroup specification can express service group membership rules, membership constraints, and classifications using the resource property model from WS-ResourceProperties. Groups can be defined as a collection of members that meet the constraints of the group and that are expressed through resource properties. The WS-ServiceGroup specification also defines interfaces for managing the membership in a service group. The interfaces defined by WS-ServiceGroup are composed with other Web services interfaces, which define more specialized interaction with the service group and with the services that are members of the service group. For example, specialized interfaces may provide other means of querying the contents of the service group, and for performing collective operations across members of the service group.

3.1.4.6 WS-Base Faults

The WS-BaseFaults specification defines a base fault type for returning faults occurring during a Web services message exchange. While there is nothing specific to WS-Resources in this specification, it is used by all other WS-Resource framework specifications in order to bring consistency to fault handling including consistent reporting of faults.

3.1.5 Summary of WSRF

The WS-Resource Framework defines a set of Web service specifications and conventions designed to standardize representation of stateful resources in a distributed environment. This framework identifies and standardizes the patterns by which state is represented and manipulated, such that a Web service can describe the stateful resources to which it provides access, and a service requestor can discover the type of this pairing of Web service and stateful resource and use standardized operations to read, update, and query values of this state, and to manage its lifecycle. The definition of the WS-Resource framework facilitates the construction and use of interoperable services, by enabling different service providers and service consumers to describe, access, and manage their stateful resources in standard ways. WSRF introduces support for stateful resources without compromising the ability to implement Web services as stateless message processors. WSRF also addresses issues of renewable references, grouping, notification, and fault reporting.

As mentioned in the introduction to WSRF, Web services standards have not been finalized at the time of the writing of this book in June 2004. Changes, even significant changes, may occur.

Web services standardization activities are not bound to WSRF alone. Web services standardization remains a dynamic area with many parties and activities involved. Basic XML technology and protocols (SOAP) continue being defined by W3C. OASIS is another important standard's body. At the time of the writing of the book, a discussion is going on to move standardization of WSRF from GGF to OASIS. OASIS is also the body for standardizing the Web Services Distributed Management (WSDM), specifically targeting management aspects of Web services. The Distributed Management Task Force (DMTF) has been focusing on models and developed the standard for the Common Information Model (CIM), which recently received more attention around the move towards model-driven operation. All these activities show that Web services standardization remains a dynamic area with many parties and many activities involved.

3.2 Web Services Distributed Management (WSDM)

The OASIS workgroup on web service distributed management is looking at a framework for management of Web services and other computing resources through Web services. Web Services Management Framework (WSMF) is a proposal that has been submitted to WSDM. The intent of the proposal is to provide a management model that is based on managed objects, their properties and operations, and their relationships. Managed objects provide management capabilities by implementing management interfaces which are described using WSDL. Hence management interfaces define management Web services. Management clients can provide management information through an external standard way without having to expose their internal implementations.

Additionally, they can provide the management capabilities by implementing what they can support and gradually augmenting their list as their management capabilities grow.

A manager role can use this framework for:

- discovering the management WSDL definitions
- discovering the list of the managed objects
- sending and receiving notifications
- monitoring, auditing, and controlling various aspects of managed objects by utilizing the supported management operations
- providing services to further aid the complex interactions in which the Web services engage

Currently there is no standard way for managing Web services. What management vendors can offer is instrumentation at the SOAP end points and intermediaries (e.g. SOAP servers, UDDI servers, etc.). This will provide information about the Web services as they use these applications. However, this management view is incomplete and lacks critical information on the state of the Web service as it is executing and messages are traveling between various end points and consumers. For such information, Web services need to become inherently manageable. A minimal set of interfaces are required that describe:

- Service: A managed representation of a Web service resource.
- Conversation: A service's view of the state associated with a set of related messages.
- Execution Environment: A managed representation of a Web services Execution Environment.
- Intermediary: A managed representation of a SOAP intermediary.

These through Web services managed objects also share a basic set of relationships:

- Containment: between execution environment and services, and between services and conversations.
- Dependency: between services, between intermediaries, and between services and intermediaries.
- Correspondence: between conversations.

As new capabilities and standard use models for Web services are developed and adopted, their users (preferably their creators) can use the model to define additional managed objects, relationships, and interfaces to add manageability support to their model.

4. PLATFORMS FOR WEB SERVICES

Web services platforms provide the libraries and support for executing Web services. Platforms have gone through change over the course of time. We can classify the Web service platforms into different generations to understand the progression of growth in them:

- First Generation – HTML/CGI: mostly static content in HTML pages, HTML FORMS for simple choices and dialogs, and the CGI (Common Gateway Interface) to connect web servers to application programs, mostly Perl or Shell scripts.
- Second Generation – Java: dynamic generation on server side of HTML pages, user session support, the Java servlet interface became popular to connect to application programs,
- Third Generation –further developed environments appeared: J2EE as foundation for application servers.
- Fourth Generation – XML Web services platforms: characterized by the introduction of XML and WSDL interfaces for Web services with SOAP-based messaging, and a global service infrastructures for service registration and discovery emerged – UDDI, WSIL
- Fifth Generation – dynamic Web services aggregation, characterized by flow systems, business negotiations, agent technology, etc.

Technically, Web services have been built according to a pattern of an n-tier architecture that consists of: a front-end tier: firewall (FW), load balancer (LB), a web-server tier (WS), an application server (AS) tier, and a backend tier for persistent data, or the database tier (DB).

4.1 First Generation: CGI and Perl

The World Wide Web has facilitated the emergence of HTML pages on user's web browsers. Initially the web pages had mostly static HTML content. Passive information services could be built that provided users with the only capability of navigating though static pages. However, FORMS was supported by HTML from the very beginning that allowed users to enter text or select from multiple choices menus. FORMS were treated specially by the web server. They were passed on to an interface, CGI – the Common Gateway Interface, behind which small applications, mostly Perl or Shell scripts, could read the user's input, perform corresponding actions, and create an HTML page that can be visualized in the user's browser. This primitive mechanism enabled a first generation of services in the web beyond pure navigation through static content. Some of the services still use these technologies today.

4.2 Second Generation: Java

With the proliferation of the web and the need for richer set of services such as online shopping or travel booking, the initial means to build Web services quickly proved to be too primitive. Java appeared as language of choice for Web services and brought certain interactive capabilities with it. Servlets provided a better interface between the web server and the application.

Technology was introduced to support the dynamic generation of HTML pages at the server side: ASP (Active Server Pages) by Microsoft, JSP (Java Server Pages) by Sun Microsystems, or PHP pages in the Linux world enabled the separation of presentation, the appearance of pages in browsers, from content data. Since user identification was critical for business services, user login and user sessions were introduced. Applications were becoming more complex, and it turned out that there was a significant overlap in common functions needed for many services such as session support, connectivity to persistent data bases, security functions etc.

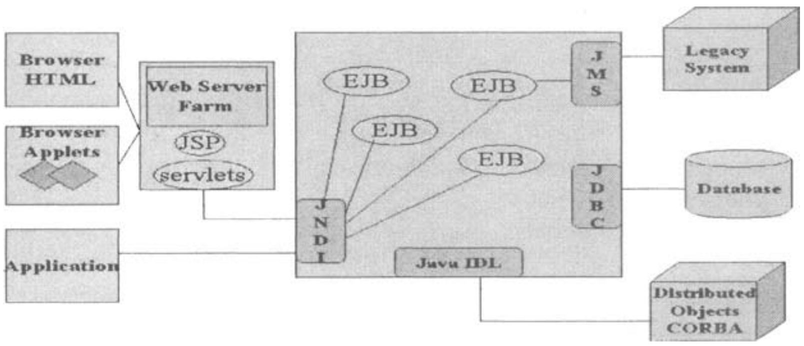


Figure 10: Example of Java-based Web service.

4.3 Third Generation: Richer Development and Run-time Environments

As people developed Web services they realized that there were a basic set of services that were common across applications. This led to specification of standard set of libraries in Java. A cornerstone of these environments became J2EE. J2EE (Java 2 Platform, Enterprise Edition) is a

Java platform designed for enterprise-scale computing. Sun Microsystems (together with industry partners such as IBM) designed J2EE to simplify application development for Web services by decreasing the need for programming through reusable modular components and by providing standard functions such as session support and database connectivity.

Application Server: J2EE is comprised of a set of libraries used by application programs performing the various functions. Developers of these Web services still had to assemble all the pieces, link them together, connect them to the web server, and manage the various configurations etc. This led to the creation of pre-assembled packages that could easier be deployed on a variety of machines. These pre-packaged set of libraries have been termed application server. They significantly reduced the amount of configuration work during service deployment so that service developers could spend more time on creation of the actual service and the business logic of the service. Most application server are based on J2EE technology. Examples are BEA's WebLogic environment, IBM's WebSphere suite, Sun's iPlanet Application Framework or Oracle's 9*i* application server.

4.4 Fourth Generation: Web Services Frameworks

Most of the previous generation Web services were B2C kind of services that were focused on the customer. It turned out that to enable Web service to Web service interactions, it was necessary to have a set of standards that will enable seamless interactions between them. HTML content was being parsed by the browsers in different ways and it was necessary to have a standardized way of expressing and parsing information. XML provided means of expressing content and semantics in schemas. The Web services community decided to use XML for not only specifying content but also to express standards. XML is being pervasively used for messaging (SOAP) and for Web service interface descriptions (WSDL). XML enhancements were also added to J2EE and application servers. XML turned out to be a major differentiator between Web services platforms of the 3rd and the 4th generation in this classification.

UDDI (Universal Description and Discovery Interface) appeared a standard for enabling service registration and discovery. It is a set of registries that are maintained by IBM, HP, and Microsoft. Its goal is to enable companies and individuals to find one another on the web in a much more structured and classified manner than it is possible through search engines. Microsoft, IBM, and Ariba spearheaded UDDI. However, UDDI has not taken off in a big way. People have also proposed the use of Web

Service Inspection Language (WSIL) as means of distributed discovery and introspection.

Two major platforms are currently offered that explicitly aim for further Web services interaction and integration: Sun Microsystems' SUN ONE (Open Net Environment) and Microsoft's .NET.

SUN ONE: SUN ONE' is a platform for Web services. Its architecture is build around a DART model: Data, Applications, Reports, and Transactions. It supports major standards like: XML, SOAP, J2EE, UDDI, LDAP, and ebXML. The architecture is comprised of three product lines: the iPlanet Application Framework (JATO), Sun's J2EE application framework for enterprise Web services development, the Solaris Operating Environment, and the Forte Development tools.

Microsoft .NET: Microsoft's .NET platform intends to provide Web service platforms and a set of technologies that will enable creation and interaction with Web services. With Microsoft .NET, Web services' application code is built in discrete units, XML Web services, that handle a specified set of tasks. Because standard interfaces based on XML simplify communication among software, XML Web services can be linked together into highly specific applications and experiences. The vision is that the best XML Web services from any provider from around the globe can be used to quickly and easily create a needed solution. Microsoft is using C# as its language of choice for development and provides a set of tools like Visual Studio .Net for enabling creation of Web services.

5. PUTTING IT ALL TOGETHER

A typical e-business infrastructure would comprise of large number of business processes. As the intention of an e-business is to undertake business on the web, they will also need mechanisms that enable customers to access their services through the Internet. *Web services* have become a well-accepted way of putting e-businesses on the web so as to enable users (either humans or other Web services) to use them. According to many market research firms, it is likely that, before the year 2005, many of the companies' offerings will be available as Web services, and that large corporations will deploy tens or hundreds of Web services. These Web services have to be interfaced with the internal business processes to receive, fulfill and deliver orders. A complex infrastructure is usually a reality in an e-business.

In order to explain, how these varying technologies may be put together, let us consider an example scenario. We assume that business partners provide Web services in order to enable dynamic interactions with each other.

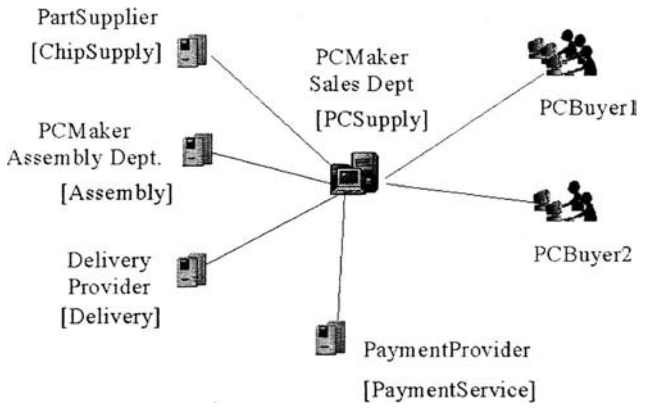


Figure 11: PC purchase scenario.

Individual Web services are implemented as business processes that are managed by business process management systems.

Figure 11 summarizes overall layout of our example scenario. The names of Web services provided by participating businesses are shown in brackets.

The scenario includes the following participants:

- *Clients:* PCBuyer1 and PCBuyer2 are two example clients that purchase PCs from a vendor (PCMaker). We assume that some clients, such as PCBuyer1, are capable of using the Web service provided by their vendors in order to submit their purchase orders; whereas, some other clients, such as PCBuyer2, prefer to submit their orders through fax or phone calls.
- *Providers:* PCMaker Sales Department provides the Web service "PCSupply" in order to allow its clients to interact with itself electronically. This department also acts as client and uses Web services of various service providers and product suppliers in order to satisfy its own clients' orders. In our scenario, PartSupplier, DeliveryProvider, and PaymentProvider are three external service providers, and PCMaker Assembly Department is an internal service provider.

There may be many more providers, customers, and internal departments involved in such a scenario, but we list only few for simplicity.

The PCMaker.com e-business is shown in Figure 12. This infrastructure is set up by PCMaker.com that receives orders from companies/humans interested in buying PCs. It has internal business processes like user authentication, PC manufacturing, preparation of invoices etc. For some of the PC order parts, it needs to contact its supplier and similarly uses a shipping company to ship the PCs it makes. In order to do business on the web it needs to have a Web service that enables other users to access its e-business. The PCMaker.com Web service has a Web service that has operations, namely Login, Price_quote, Order_request, Send_invoice, and Send_shipment. It also has other operations, namely Order_parts and Ship_order.

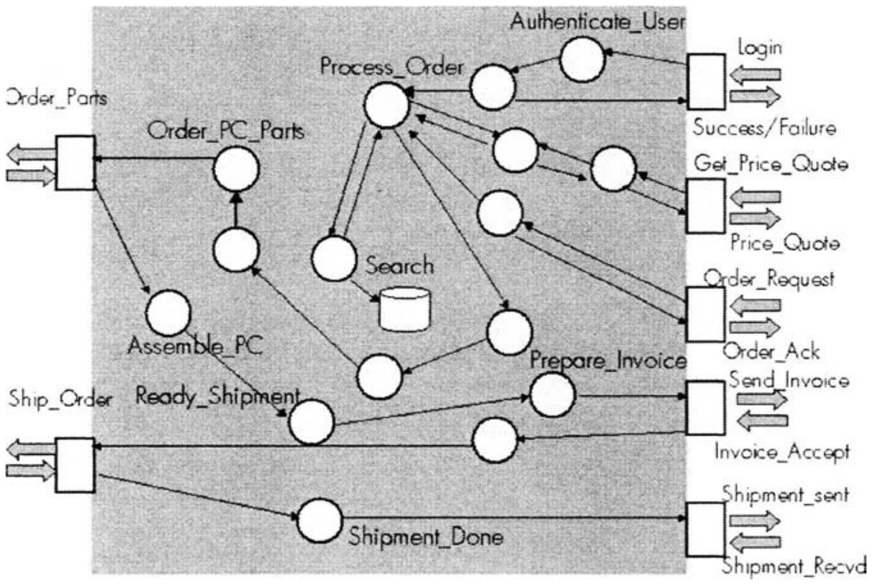


Figure 12: Business processes and Web services in PCMaker.

Figure 13 shows the definition of a sample business process that runs at the back-end of Price_quote operation. The process is initiated when a Get_PriceQuote (PQ) request is received from a client. The process proceeds in three parallel branches to collect quote information from three business partners, then calculates the total cost, and finally sends the total cost to the client. Similarly, there could be many other business processes implemented to handle client requests at the back-end of a Web service.

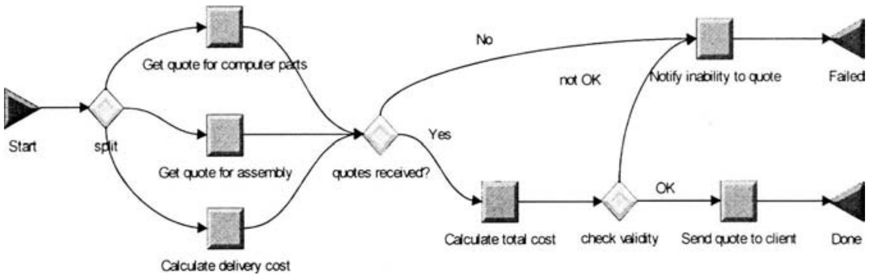


Figure 13: Definition of a sample business process.

Behind all the logical business processes, Web services and operation as described above, the e-business infrastructure needs hardware, software and humans that support the Web service infrastructure. For example, it would require web sites, web server farms, applications servers (J2EE, .Net) and business processes execution platforms (e.g., Process Manager, MQSeries, Web Method).

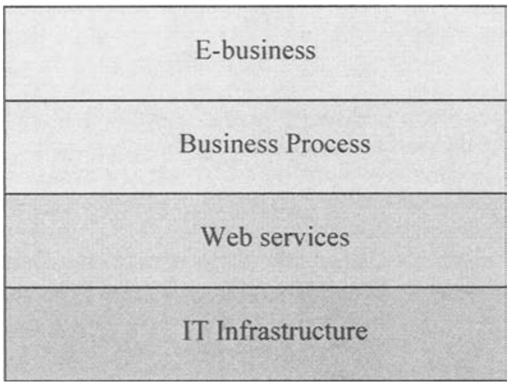


Figure 14: Layers from IT infrastructure to E-business.

Web services being critical to an e-business infrastructure it is important to manage them while undertaking e-business management. Since Web services depend on the IT infrastructure, management of Web services also requires that the underlying IT infrastructure be managed. Once the metrics collected at the IT infrastructure, Web services and business process layers are correlated the management data may be used to either present views to the business managers/analysts through consoles or may be used for managing the enterprise (intra-enterprise) and its relations with its customers (CRM) and other enterprises (inter-enterprise management).

SUMMARY

Web services are emerging as the de-facto mechanism of making services available on the web. In order to enable this scenario a set of standards are being worked upon. These standards range from those related to describing the service (WSDL), discovering the service (UDDI, WS-IL) to business processes that drive the message exchange (WSFL, XLANG, BPEL4WS, ebXML). In order to manage Web services it is thus important not only to understand how these different technologies fit together but also the effect that they have on the current enterprise stack.

Chapter 3

APPLICATION MANAGEMENT AND WEB SERVICES

1. INTRODUCTION

Web Services may be thought of as specialized applications. The body of work that exists in traditional application management is thus quite relevant to the domain of web service management in general. It is thus important to understand the body of work that exists in traditional application management.

Effective application management has been traditionally the main goal of IT managers and IT support organizations. Varied tools and technologies are typically used to bear upon application management. Various criteria can be applied for classifying the space of application management. The stages of an application life cycle, for instance, can be used for this purpose.

An application life cycle is understood as the sequence of defined stages through which an application transitions over its existence. An application life cycle comprises of stages of application creation (production), application deployment (introduction into the system– installation and configuration), operation of the application (usage), maintenance (upgrade, improvements), and the final stage of application phase-out (leaving the system). All these stages and transitions are typically quite complex and require a variety of management processes.

Management per se is not limited to managing networks or system management, as it is often described in technical literature. Application management inherently straddles across "non-computational" domains such as people or processes as is the case in stages of application development, deployment, and operation. Only few areas are covered by management systems and automated support of management operations. Human beings executing management tasks and operations still dominate application management. Multiple examples exist in support of this hypothesis. The development process of an application, and the developers themselves, need to be managed during the application creation stage. During operation, operators or administrators perform most management tasks, assisted by computerized systems that collect, aggregate and present data to them. What is often referred to as application management in most cases only addresses the areas where computer-supported management systems are used, mostly during operation, monitoring and data collection. Event reporting and correlation are typical tasks management systems perform today. At the end, data and alarms are displayed to the operators behind a management console who take decisions and perform corrective actions.

Reducing the discussion of application management to only those areas where management tasks are computer assisted does not reach far enough for a comprehensive overview of application management across the entire application life cycle, as introduced in greater detail in section 2.

For this reason, a generalized view to application management and management processes is introduced in section 3 that encompasses all aspects of application management through all stages of an application life cycle. A uniform view to management across the different areas and aspects related to application management is provided. Such a uniform view allows bridging the gap between computer assisted management processes and tasks that rely on human involvement and execution.

Subsequent sections then discuss management-related aspects in each stage of the application management life cycle in more detail.

2. APPLICATION LIFE CYCLE

Application life cycle is understood as the sequence of stages through which an application transitions during its existence. The general model of an application life cycle is shown in Figure 15.

The outer cycle illustrates the application passing through the stages of creation, deployment, operation, maintenance, and final phase-out. The inner cycle between deployment, operation and maintenance characterizes the main stages during which an application is used. In most cases, maintenance cannot be done directly during operation. Maintenance thus is shown as a separate stage after operation and typically involves deployment and upgrade of components.

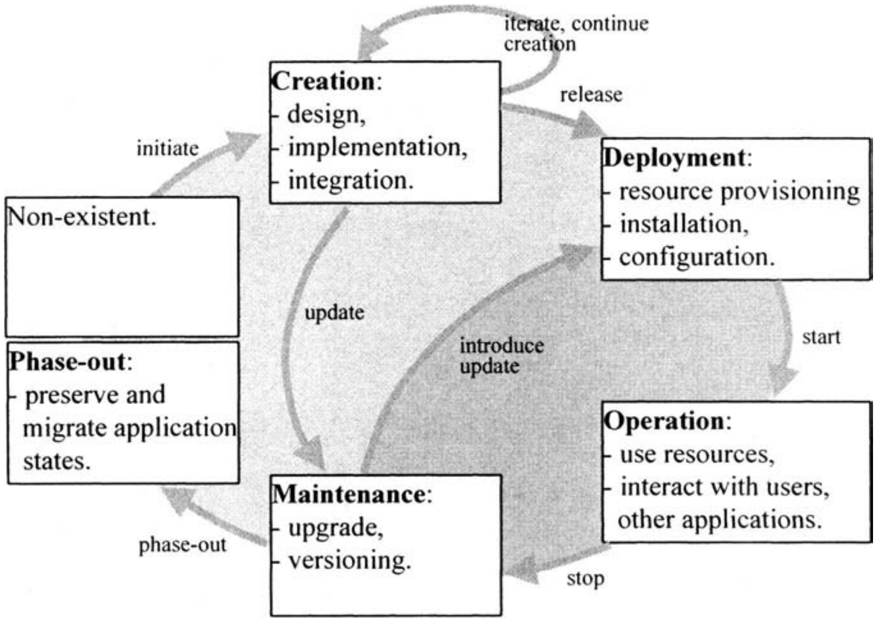


Figure 15: General model of an application life cycle.

Applications are defined, designed, implemented and integrated with other applications during the *Creation* stage. Software development processes characteristic of this stage are requirement analysis, architecture and design, implementation, quality assurance and iterations upon these processes. Creation leads to a new application or leads to a newer version of an existing application. After an application leaves the creation stage for deployment, the creation process typically involves continued improvement of the application, independent of the releases that have been moved to deployment and use. The aspect of continued application creation (development) after releases is shown by the arrow pointing from creation back to the creation stage in Figure 15. Subsequent versions occurring as results of continued creation will update existing applications during their maintenance stages. This is shown by the arrow leading from creation stage to maintenance in Figure 15.

At *Deployment* stage, the application is introduced in its target environment after release. For this purpose, resources for operating the application must be acquired and provisioned. Subsequently, application software and data need to be installed and configured on those resources. The deployment stage leaves the application in a ready-to-go status. Maintenance cycles also typically require installation and configuration of new application components. For this reason, an arrow from maintenance to deployment is shown in Figure 15.

The main stage of an application is *Operation*. At this stage, the actual function of the application is performed, the application is in use. Resources are used, application processes are active, and interactions with users or other applications occur. Operation is also the main stage of application management and the main area where computer-supported management systems are in use.

Upgrades are performed during the *Maintenance* stage. Some maintenance functions can be performed immediately during operation, but most of them require replacement of hardware or software components and a temporary shutdown of the application.

Transitions between Operation, Maintenance, and Deployment must be obeyed carefully since they occur while the application is available to users. Maintenance cycles often require a shutdown of an application and its temporary unavailability. Thus, scheduled maintenance cycles are part of the application life cycle management where users are informed in advance about planned shutdowns. When scheduled maintenance down time is not acceptable for applications, temporary application substitutes must be provided.

At the end of the application lifetime, the *Phase-out* stage migrates application into following generations of applications or into archives.

3. PROCESSES IN APPLICATION MANAGEMENT

As discussed in the introduction, reducing application management to processes and tasks that are assisted by management system (which are themselves applications that underlie the same life cycle pattern) would not reach far enough for a comprehensive overview of application management in all its stages of the application life cycle.

For this reason, a generalized definition of application management is presented in combination with classifying taxonomy that subsequently will be applied to the stages of an application life cycle for discussing application management.

3.1 Generalized View on Application Management

Management in general is the *set of tasks (process),* better still an entirety of processes[2] that is established to assure that a managed domain meets its expectations in a controlled and predictable manner. Expectations need to be defined and formalized for this purpose. Appropriate management processes must be identified with defined responsibilities in the managed domain. Examples of managed domains include components, entire systems, and processes. Managed domains can also include people such as employees, developers, operators, or entire organizations.

A management process, or typically a variety of management processes, accompany a managed domain and interacts with it as shown in Figure 16.

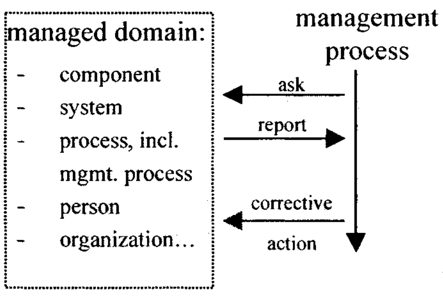


Figure 16: Management process accompanying a managed domain.

In order to fulfill its tasks, a management process needs to interact with the managed domain in a defined manner. Management protocols implement those interactions. Management protocols typically have two purposes:

- The first purpose of a management protocol is the observation of the data collected from probes or sensors by performing measurements and receiving notifications about events and other conditions that have occurred in the managed domain.

  This input data received from the managed domain, in combination with policies representing the management goals and objectives, provide the basis for assessing conditions in the managed domain, and to determine whether it operates within the defined scope of expected

[2] Management processes need not be confused with the business processes. Management processes are workflows that are used to perform application management activities.

behavior or not. For this purpose, observed information must be compared with patterns that define the expected behavior for the managed domain. When deviation is determined, a decision about corrective action in the managed domain must be made.

- The second purpose of a management protocol is to transfer instructions for corrective actions to control points in the managed domain. The control points in turn execute instructions actuating corrective actions in the managed domain.

A management process needs to be created, introduced, executed, maintained, and finally be phased-out just like any other application. This aspect is discussed in more detail in the following section. Subsequent sections then show selected aspects of life cycle stages of management processes:

- definition of a management process at creation stage (section 3.3),
- instrumentation of a managed domain at deployment stage (section 3.4), and
- execution of a management process during operation (section 3.5).

3.2 Management Process Life Cycle

Management is defined as process or an entirety of processes with their own life cycle(s), which defines, when management processes enter a domain, when they disappear and though which stages they transition during existence.

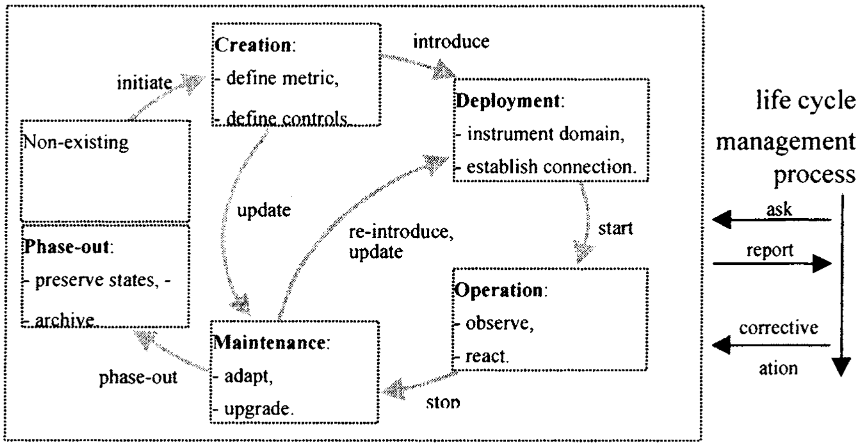


Figure 17: Management process life cycle that is subject to life cycle management itself.

In principle, the same life cycle model shown in Figure 15 for applications applies to management processes. For the discussion here, three selected aspects of a management process life cycle are discussed: definition of a management process, instrumentation of the managed domain, and execution of a management process.

Other life cycle stages of maintenance (adopting management processes to changing conditions), and phase-out (defined termination of management processes) apply as well to management processes but are not further discussed here.

Figure 17 also shows the aspect that management process life cycle is also subject to management process in order to assure defined behavior.

3.3 Definition of a Management Process (at Creation Stage)

A management process usually follows a certain pattern of steps during its existence. The steps a management process performs are defined during the creation stage of the management process life cycle.

The main aspects of defining a management process are:

1. Define the *objective* for and *supportive information* needed by the management process. The objective represents the goal of management or the responsibility of the management process. Supportive information provides the input for assessing whether a managed domain is within or not within the bounds of expected behavior. Individual aspects are:
    a. Define the "expected behavior" in the managed domain for which the management process is responsible.
    b. Derive a formalized information aggregate that represents how the goal of the management process is expressed in measurable and observable data – called *metrics*. These metrics are used to assess the managed domain whether it is within or not within the bounds of expected behavior.
    c. Derive measures that provide input for the metrics.
    d. Define data collection points and event detectors, the data and event paths, data aggregation and event correlations in the managed domain for obtaining the measures for these metrics.
    e. Define the assessment function for these metrics that assesses measures against the formulated metrics.
2. Define the *steps* of the management process that are performed in order to keep a managed domain within expected behavior:

   a. Derive summarize metrics from the base set of metrics, which can be assessed
   b. Define the policy based on which decisions about corrective actions or notifications to higher-ordered management processes are taken.
   c. Define notifications and corrective actions that need to be taken in order to adjust behavior in the managed domain when deviation from expected behavior has been assessed.
   d. Define control points in the managed domain that implement corrective actions.
3. Define the management protocol.
4. Define when and how frequently the management process is performed. What initiates the management process? Several triggers exist: fixed scheduled times, events occurring in the managed domain, instructions from higher-ordered management processes, see section 3.6 on hierarchical management processes.

3.4 Instrumenting the Managed Domain (at Deployment Stage)

In order to be "manageable", the managed domain must be instrumented with data collection and event detecting probes, and with actuators performing corrective actions in the management domain when instructed by the management process. Instrumentation of a managed domain is a main task during deployment stage of a management process life cycle.

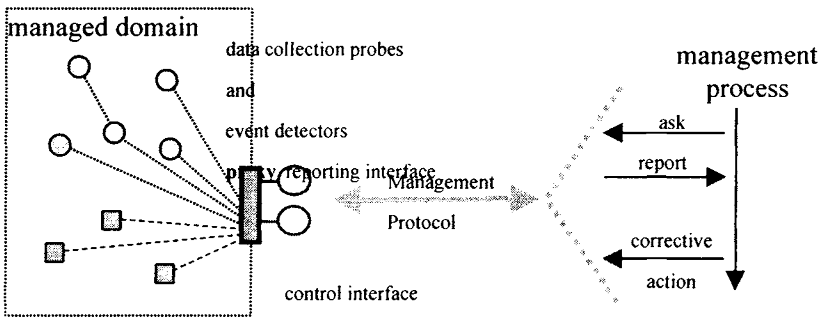


Figure 18: Instrumentation in the managed domain with proxy for the management protocol.

Data collection probes and event detectors as well as actuators are usually not directly connected with the management process. Intermediaries (or proxies) are often introduced in a managed domain for decoupling instrumentations in the managed domain from specific management protocols. Proxies then perform the protocol with the management process. Instrumentations are usually also not capable of performing the full management protocol.

A managed domain may support different management protocols by introducing different proxies. Updating the proxy can accommodate changes and evolution in management protocols, and vice versa, only updating proxies can make changes in instrumentations transparent to management protocols and management processes. All these advantages have made management proxies fairly prevalent.

As shown in Figure 18 management proxies usually provide different interfaces for reporting and control.

3.5 Execution of a Management Process (Operation Stage)

Once the management process has been defined and the managed domain has been instrumented, the management process may be implemented and executed. Remember, management processes are here understood in a broader sense than computerized flows. The "executor" of a management process may be a human being that follows a sequence of instructions defined for a management process. The implementation of a computerized management process means invariably implementing steps in terms of a programming or a flow language.

When management processes are human-driven, meaning operated by people, summarized/aggregated metrics describing expected behavior may be informal (on paper).

When a management process is computerized, metrics must be provided in formal, machine interpretable formats. An example of a formalized description of expected behavior observed by management processes is a service-level agreement or SLA [Halstone 2002].

In order to answer the questions when, how often, and how frequently the management process is performed, the following choices exist:

- It may be performed only once,
- It may be performed multiple times initiated by certain conditions, or
- It may be performed continuously in specified time intervals.

In order to determine that a deviation from expected behavior has occurred, the management process compares the aggregated data obtained from observations with the desired values. When a deviation from expected behavior has been determined, a decision about corrective actions or notification to a higher-ordered management process has to be made. This decision is guided by policy derived from management objectives.

Figure 19 shows the general flow for executing a management process.

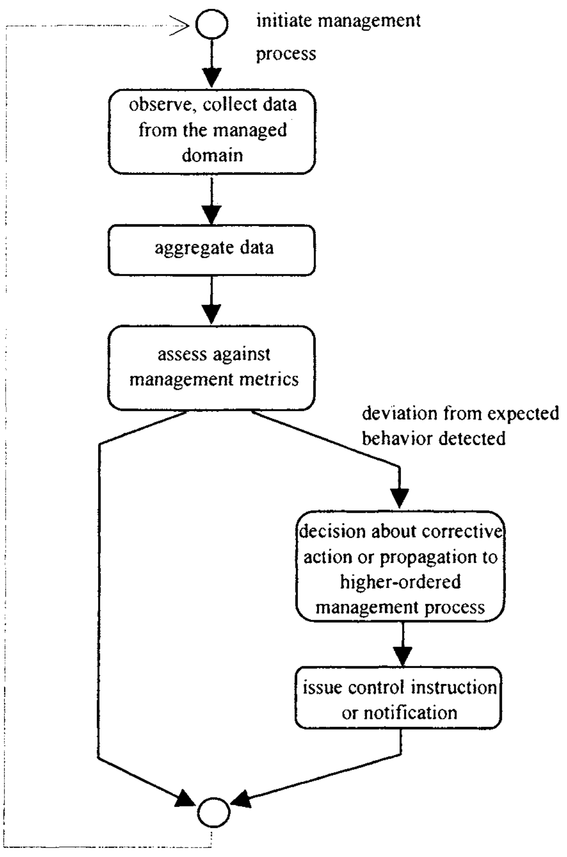

initiate management process
observe, collect data from the managed domain
aggregate data
assess against management metrics
deviation from expected behavior detected
decision about corrective action or propagation to higher-ordered management process
issue control instruction or notification

Figure 19: General flow for executing a management process.

3.6 Management Process Hierarchies

In practical systems, a large number of management processes may coexist, each responsible for certain aspects in the managed domain. Dependencies between management processes often exist. Management processes are often structured hierarchically (refer to Figure 19 where a decision about corrective action or propagation to a higher-ordered management process is depicted). A management process can issue notification to another management process for external correction or further propagation up in the management hierarchy. Complex hierarchies of nested management processes appear. Management processes themselves need to be managed coordinated, adapted to changing needs, etc. The combination of all these issues makes management such a complex arena where well-defined responsibilities and separation of concerns are essential.

3.7 Management of Management Processes

Management processes and supportive management systems must be managed as well, a recursive relationship that needs to be grounded in practice. Figure 20 shows the aspect that management processes themselves are subject to management.

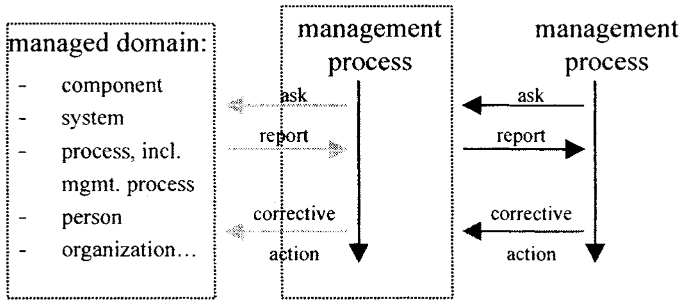


Figure 20: Managing a management process.

4. ASPECTS IN APPLICATION MANAGEMENT

The variety of management tasks in management domains can be classified in areas. One classification has widely been recognized.

Based on initial letters, this classification is also referred to as FCAPS:

Fault	Fault management deals with detecting, analyzing and repairing what is broken.
Configuration	Configuration management deals with issues during application deployment (installation, configuration) and operation (tuning).
Accounting	Accounting management determines who is using what and for what purpose.
Performance	Performance management deals with assuring the specified performance for an application system.
Security	Security management addresses the issue of who can do what in a system. Authentication (proving identity) and, based on authentication, access control are main issues here.

Table 1: FCAPS management aspects.

Classification in Table 1 covers important aspects in management, but there are more aspects to be considered. Another set of aspects are shown in the following table:

System	System management deals with the systems including hardware and software that are involved in performing an application.
Life cycle	Life cycle management focuses on component life cycles of an application as discussed in section 2.
Resource	Resource management assures that quantity and quality of resources for an application are available.
Data	Data management addresses the need to manage application data. Data about customers, processes, and inventories are usually considered quite valuable. Data management prevents losses of data, availability of data, and data representation in formats that are not dependent on specific applications.

Quality	Quality management is important through all stages of an application life cycle. Quality management assures that an application is created, deployed, operated, maintained and phased-out in a predictable and controlled manner.
User	User management deals with representing information about users in a system. User management is the basis for accounting and security management.
License	License management is important in commercial environments in order to meet the terms defined in license agreements with application or system vendors.
Inventory	Inventory management tracks the information about inventories and other valuable IT assts that exist in an organization.

Table 2: Continued classification of management aspects.

Management aspects presented in the two tables can now be associated with the stages of an application life cycle discussed in section 2. The result is a taxonomy that is shown in Table 3.

Subsequently, a second taxonomy is shown in Table 4 which associates stages of an application life cycle with steps of a management process as discussed in section 3.3 where the definition of a management process has been discussed.

4.1 Taxonomy 1: Management Aspects Versus Application Life Cycle

Following table shows in its inner cells interpretations what a management aspect (row) means when associated with a stage of an application life cycle (column)[3].

application life cycle → management	Creation	Deployment	Operation	Maintenance	Phase-out

[3] Double exclamation marks (!!) in cells indicate high importance of the management aspect in combination with the associated life cycle stage. Double hyphen (--) indicate that the management aspect is less relevant to the particular application life cycle stage

aspect ↓					
Fault management	planning, architecture and design failures	unavailable resources, failures in deployment	detect, analyze, repair run-time failures !!	failures during maintenance cycles	prevent application data lost
Configuration management	tools needed for design, development and testing	customization in target domain	adjust configuration to improve operation	adjust configurations during maintenance	--
Accounting management	The (people) time spent in various stage of development.	--	accounting during operation !!	less relevant	--
Performance management	meet time lines, budget, performance of contributors	resource provisioning, capacity planning	ensure run-time performance meets expectations	duration of maintenance cycle	--
Security management	secrecy of designs and code	avoid security leaks when deploying new applications	prevent security leaks and break ins	prevent introducing security leaks during maintenance	prevent leaking or altering data
System management	development and test systems	adjust, tune systems for application	manage systems used for operation	adjust systems for requirements of maintained application	prepare systems for reuse or replacement
Life cycle management	processes for analysis design implementation and test	--	--	versioning, prevent components dependencies	--
Resource management	people are main resource during creation	chose compute, storage, network resources	use of resources, flexing resources based on demands	chose compute resources that meet new requirements	release resources for reuse or replacement
Data management	requirements, designs, code, documents	bring in application data, transform data when needed	prevent losses !!	maintain application data, prevent losses	preserve, migrate application data !!
Quality management	code quality, quality assurance, testing	ensure the deployed application is stable and robust	availability, reliability, QoS assurance	ensure, improve application quality	--
User management	people accessing development and test systems	manage access to deployment systems for deployment engineers	who has access to the application with which privileges	users having temporary access to systems during maintenance	--
License management	licenses needed for development and test	configure license managers, ensures	ensure that license agreements are met during	ensure that maintenance stay in line with license	--

	systems		operation	agreements	
Inventory management	development and test systems	identify resources to deploy an application	resources and facilities where an application operates	replacement of resources and application parts	retire resources or applications

Table 3: Taxonomy associating management aspects with application life cycle.

Table 3 illustrates that management aspects in most cases are relevant throughout all stages of an application life cycle.

4.2 Taxonomy 2: Application Life Cycle Versus Management Process Steps

Another taxonomy can be discussed that associates application life cycle stages with selected steps of management processes (refer to sections 3.3 through 3.5). The following steps of a management process (refer to section 3.3) are considered for the taxonomy shown in Table 4:

1) define expected behavior (functional and non-functional) of the managed domain (step 1.a in section 3.3),
2) derive management metrics (step 1.b in section 3.3),
3) define measures and instrumentation in the managed domain (step 1.c in section 3.3) to provide input for assessment against the management metrics defined in 2),
4) define assessment function that correlates measures against the management metrics (step 1.d in section 3.3) and alert or initiate correction in the managed domain when deviations from expected behavior occur.

The combinations of application life cycle stages with the selected steps of management processes are shown in the following table:

management steps → application life cycle ↓	Define expected behavior	Derive management metrics	Define measures and instrumentations	Define assessment
Creation	requirement analysis, development process	architectural and design specifications, processes and roadmaps, budget	milestones, budgets, deliverables, tests, use cases	does the system meet requirements?
Deployment	define roadmaps, deliverables and milestones	describe platform and environment requirements and dependencies	observe processes for resource provisioning, installation, configuration	verify steps for resource provisioning, installation, and configuration

Operation	availability, reliability, performance	from simple parameter thresholds to complex SLA	instrument the application and associated resources, plan management system	alert or notify when thresholds are passed or SLA are violated
Maintenance	improve functional and non-functional behavior, remedy failures	typically not explicitly defined	maintenance is observed and verified as stages progress	does the evolved system meet the expected behavior?
Phase-out	data preservation	timelines	typically not explicitly defined	obey timelines

Table 4: Taxonomy associating application life cycle with management process steps.

The following sections discuss examples of management processes that are relevant to selected stages of the application life cycle:

- application creation (section 5),
- application deployment (section 6), and
- application operation (section 7).

Discussion will illustrate what management processes need to be defined for each stage and will give examples how management metrics and assessment functions can be provided and which instrumentation is needed in the managed domain. It will also discuss who is carrying out management processes.

5. MANAGEMENT IN APPLICATION CREATION

Application creation encompasses processes of defining applications based on requirements, including architecture and designs of application components, implementation, verification and test. The result is a working version of an application that is released for deployment.

A variety of management processes is involved in application creation. Two examples will be elaborated in order to provide some coverage of the problem space.

5.1 Definition of Management Processes for Application Creation

5.1.1 Example 1: Implementation

Implementation must be performed as a controlled process. Detailed design specifications are prerequisites for implementation. Design specifications are result of precursory design stages.

For Implementation, a plan must be developed how the implementation process can achieve the desired result in a controlled manner.

Following aspects need to be considered:

- Resources (budget determining headcount, equipment, utilities),
- Timelines (roadmap, milestones, deliverables),
- Structure (organization, division of work, component break up, definition of pieces for individual developers, team structure), and
- Dependencies (among pieces and among developers and teams).

The assessment function for managing implementation processes usually is defined in terms of time-line charts where deliverables or milestones are informally described with dates assigned. Time-line charts provide also the basis for measuring and assessing progress during implementation.

5.1.2 Example 2: Assurance, Testing

After implementation, the quality of an implemented version of an application must be assured and verified. This is usually achieved by testing. Testing is another stage in application creation following implementation. Testing has the purpose of assuring the quality of the implemented application by assessing behavior of the implemented code base against requirements. Testing can be complex and need to be organized in form of test plans. Test plans are derived from application requirements and encompass aspects of:

- Functional verification (does the implemented application provide the functions defined in requirements),
- Usability verification (is functionality provided in a usable and expected manner),
- Performance verification (are quantitative requirements met),
- Robustness verification (is the application robust against failures).

Similarly as in implementation, the assessment function for managing assurance processes usually is defined in terms of time-line charts where deliverables or milestones are informally described and assigned to dates and teams. Time-line charts provide also the basis for measuring and assessing progress during the assurance.

Since quality assurance has become more important over time, the International Organization for Standardization (ISO) has introduced a certified standard ISO 9000 [ISO9000] for quality assurance. Application vendors certified with ISO 9000 assure that they use the set of measures and processes defined in ISO 9000 for assuring quality of their software products.

5.2 Instrumentation in the Managed Domain

The managed domain for both examples, application creation and assurance, is primarily people, developers and teams that need to be organized. Instrumentation in the managed domain thus focuses on how individual people (the "managed domain") are informed about expectations, how they are directed, and how progress is reported to (human) managers. Weekly reports have turned out as a reasonable measure for management measuring and assessing progress against the time-lines defined as assessment functions when management processes have been defined.

5.3 Execution of Management Processes

Management processes are performed by (human) managers that perform the roles of defining and refining management processes for their domains of responsibility, structuring work and organizing teams, communicating management processes to team members and assessing progress based on reports from team members.

Tools are used for reporting and creating time-line charts partially automating routine tasks and assisting managers. Management in application creation stage, however, is primarily executed and bound to humans.

6. MANAGEMENT IN APPLICATION DEPLOYMENT

Application deployment follows after application release. Deployment is the process of making an application operational in the context of a customer's environment. Various tasks, associated with deployment, need to be managed. Examples are:

- Allocating resources for the application,
- Installing and configuring the application on those resources,
- Customizing the application to meet specific customer needs, and
- Migrating customer data into the application.

Management processes are again needed to achieve and assure controlled behavior in these domains. Two examples are elaborated for illustration.

6.1 Definition of Management Processes for Application Deployment

6.1.1 Example 1: Installation and Configuration

Installation and configuration of applications can be a complex task. Management processes involve sequences of steps that need to be carried out for installing and configuring an application. Management must encompass domain knowledge by specialists that have critical understanding of complex applications. Specialists primarily carry out deployment tasks and, as human beings, need to be managed themselves with time-lines, deliverables and milestones that define assessment functions for managing installation and configuration processes.

6.1.2 Example 2: Customizing an Application

Applications usually need to be customized for customer environments. Customization, again, is primarily performed by specialists with domain knowledge of the customer's environment and the application.

Again, specialists primarily carry out deployment tasks and, as human beings, need to be managed similarly as in application creation stage with time-lines, deliverables and milestones that define assessment functions for customizing applications.

6.2 Instrumentation in the Managed Domain

As mentioned, application deployment is hardly automated for complex applications and is thus undertaken by humans. Instrumentation for management processes in the domain of application deployment thus is also closely related to managing people and their progress towards achieving a task. Time-line charts are used to assess progress. Instrumentation in the managed domain basically shows as reporting of achievements or problems by individuals to responsible (human) managers who assess progress and make decisions about corrective actions if needed.

6.3 Execution of Management Processes

Human managers mostly execute management processes in application deployment. Some tool assistance is provided for establishing and verifying time-line charts and partial automation of processing and archiving reports obtained from individuals. Centralization und unification of reporting has turned out helpful for executing management processes.

7. MANAGEMENT DURING APPLICATION OPERATION

Application operation is the main domain where automation is achieved and management systems are applied. In contrast to application creation and deployment, application operation is not largely dominated by human involvement.

Three examples have been selected and will be discussed further elaborating management in application operation space:

- Fault Management (section 7.1.1),
- Performance Management (section 7.1.2), and
- SLA Assurance (section 7.1.3).

A section about management systems will provide an overview of how management systems work today and how operators interact with management systems.

7.1 Definition of Management Processes for Application Operation

Management systems for supporting and automating management tasks during operation require definition of management processes that can be formalized and implemented in a management system. Special attention must be paid when human involvement is required for operation and interfaces between management system and human operators are used. Those interfaces often provide critical points for operation and are also critical points of failure. Interfaces are provided in form of management consoles where information from the management system is reported to the operator. This information is used by human operators for making decisions and for actuating corrective actions through control provided as part of the management console.

7.1.1 Example 1: Fault Management

Fault management is probably the most important task in operation management. Fault management deals with the problem of "what to do when some thing is broken". These are circumstances when an application does not meet expectations in quantitative (functional) terms.

Management processes in fault management are usually defined as effect -reaction chains like "when this happens" then "do this". First, fault conditions must be detected. Next, the condition must be classified for proper reaction. And third, the reaction must be executed.

Detection is usually automated by management systems based on probes in the managed domain (in this case the application).

Fault conditions are usually identified when one or more components of an application system are not responding or are not responding in the expected manner. Definition of management processes includes defining what expected behavior of components is and how this behavior can be observed in the managed domain by proper instrumentation and reporting. Faults can be detected in a variety of ways. Components themselves can detect and report abnormal conditions or behavior. Probes external to components can also observe component behavior and detect and report faults. Rules for abnormal conditions or behavior must be defined as part of the management definition process and established in the management systems during instrumentation.

Sometimes, effects of failures in components are not detected as part of component observation directly. This may be the case when components cannot be instrumented. Effects of failures then will appear as unexpected behavior in other components in the system.

Root cause analysis is the process for correlating failure observations in the system with components that actually caused the failure or are the root cause of the failure. Root cause analysis identifies cause-effect chains tracing observed behavior to the component(s), which are the root cause of that effect. Root cause analysis requires complex understanding of the managed domain, the application and the management systems. Root cause analysis is hardly automated today.

Management processes for failure management must address:

- Identification of components, define expected or normal behavior for components in terms of measurable data.
- Define instrumentation inside or in the environment of components for observing and assessing that component behavior is in the bounds of expected behavior that has been defined for a component.
- Define responsibilities and reporting paths when abnormal behavior is detected.

Implementation of management processes is typically realized by customizing components of management systems.

7.1.2 Example 2: Performance Management

Performance management is another important area for ensuring that an application meets expectations. Performance management processes usually are concentrated on the performance critical components. Identifying which components are in the performance critical paths is a first important step for defining management processes for performance management. Defining performance management processes includes as a second important step the definition of metrics in quantitative terms such as measures for component.

Measures of performance critical components can include:

- *Throughput*– how many requests can be processed per time unit,
- *Latency* – what delay occurs between receiving a request and processing it,
- *Response time* – what delay is experienced by the issuer of a request between issue of the request and the receipt of the result,
- *Utilization* – what percentage of capacity is used in a request processing component,
- *Simultaneous requests (concurrency)* – numbers of current or maximum number of requestors (clients, customers) that can interact with an application simultaneously.

Measures defined for performance critical components are then observed in the system by proper instrumentations. Degradation of parameters will cause notification to a responsive control system, which automatically can decide about and initiate corrective actions in the managed domain, or observed conditions will be reported to an operator console, and thus to the operator (using the console).

Performance management also requires processes for collecting statistical data, workloads, and usage patterns and for associating them with individual application systems or sub-components.

7.1.3 Example 3: SLA Assurance

A Service-Level Agreement (SLA) is a more recent form of formally describing expected behavior of application systems or components. SLA definitions include bounds for measurable data that define expected behavior in Boolean or statistical terms. Boolean terms/metrics refers to situations where it can be clearly decided whether a parameter is in bound or out of bound. Simple conditions are used in SLA such as:

response-time < 10 msec.

Any request with a response time larger (or equal) 10 msec is a violation of this condition and subject to reporting.

Since Boolean conditions are very strict and may cause failure reports to be generated even when the system is in normal condition, but may just be facing a sudden hot spot causing a slight delay. Relatively complex formulations are thus often desired for description and observation.

Statistical conditions usually refer to average or mean measures over time intervals eliminating transitional changes being reported. Descriptions are of the form:

*average(response-time < 10, 5*60*1000).*

This expression means that the average response time over an interval of 5 min (5*60*1000 msec) must be less than 0.01 sec.

Depending on capabilities of SLA, any statistical function can be used for expressing statistical SLA terms.

A typical SLA has the following components:

- *Purpose* – describing the reasons behind the creation of the SLA.
- *Parties* – describes the parties involved in the SLA and their respective roles. Examples are issuers and receivers of requests.
- *Validity Period* - defines the period of time that the SLA will cover. This is delimited by start time and end time of the term.
- *Scope* – defines the components covered in the agreement.
- *Restrictions* – defines the necessary steps to be taken in order for the requested service levels to be provided.
- *Service-level Objectives (SLO)* – are the levels of service that both the users and the service providers agree on, and usually include a set of service level indicators, like availability, performance and

reliability. Each aspect of the service level, such as availability, will have a target level to achieve.

- *Service-level Indicators (SLI)* – the means by which these levels can be measured.
- *Penalties* – spells out what happens in case the service provider under-performs and is unable to meet the objectives in the SLA. If the agreement is with an external service provider, the option of terminating the contract in light of unacceptable service levels.
- *Optional services* – provides for any services that are not normally required by the user, but might be required as an exception.
- *Exclusions* – specifies what is not covered in the SLA.
- *Administration* – describes the processes created in the SLA to meet and measure its objectives

An example of a SLA definition is shown in Figure 21 for the following terms: Assuming that myUDC.com and ASP.com have a clause like: At month-end, the availability of the farm allocated to the user myASP.com, measured on the myUDC.com from Mon-Fri from 9AM-5 PM should be at least 99.9%. This can be specified as shown in the SLA definition shown below in Figure 21.

Instrumentation in the managed domain then uses SLA specifications for validation and detecting deviations from expected behavior, expected behavior as described in the SLA. SLA in this sense provides parameterization for instrumentations in the managed domain. General probes can be used instead of specific probes that have one observable metric built in. General probes are tailored to specific measurement tasks by loading an SLA agreement for observation them. SLA definitions can be complex depending on the granularity of how expected behavior is defined.

SLA definitions appear as result of management definition processes. Careful planning is required for establishing complex management systems. Little tool support for creating and maintaining SLA is provided today due to recent emergence of SLA.

SLA definitions and SLA-based management are seen as a key for further automation in system management. Management of SLA is becoming an own area in system management.

```
<sla>
    <slaId>00000159398583</slaId>
    <partnerName>ASP.com </partnerName>
    <startDate>Fri Feb 15 00 :00:00 PST 2004</startDate>
    <endDate>Mon Jul 15 00:00:00 PST 2004</endDate>
    <slo>
        <sloId>1</sloId>
        <dayTimeConstraint>9:5:1:5</dayTimeConstraint>
        <measuredItem>
            <item>
                <constructType>fml.hp.com/farm</constructType>
                <constructRef> My-2-Tier-Farm</constructRef>
                <measuredAt>myUDC.com </measuredAt>
            </item>
        </measuredItem>
        <evalWhen>month-end</evalWhen>
        <evalOn>all</evalOn>
        <evalFunc>Availability:99.9:percent </evalFunc>
    </slo>
</sla>
```

Figure 21: Example of a SLA definition in XML syntax.

7.2 Instrumentation in the Managed Domain

Various techniques exist for instrumentation in managed domains during application operation. Just like the applications, the systems providing infrastructure or necessary functions in the environment of an application must be subject to observation since the functioning of the application depends on those underlying systems as well. Examples are machines on which applications are running, networks connecting applications to users or application components to one another, or other applications or services referred to by an application during processing.

Management systems today cover underlying hardware infrastructures to a large extent for instrumentation and observation. Dependencies of applications to other application are less covered. Recent emergence of SLA also incorporates dependencies among application into observation.

Various standards have evolved over time for purposes of providing common data models, common access protocols and common communication protocols among management instrumentations.

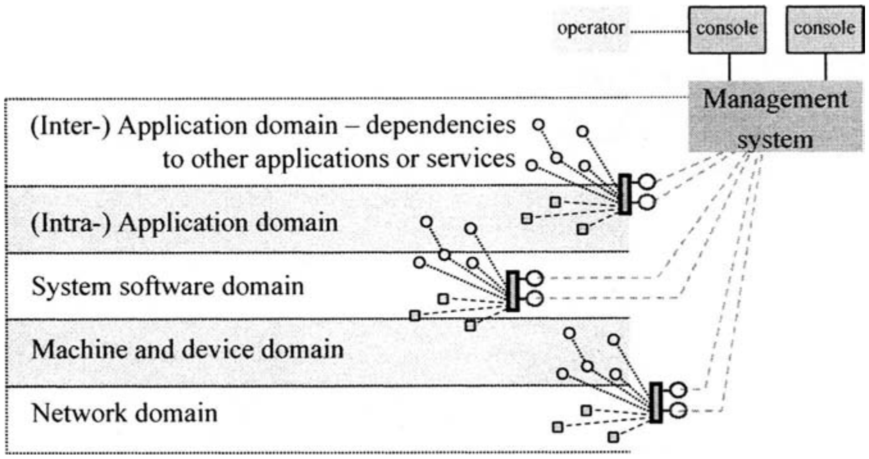

operator
console
console
Management system
(Inter-) Application domain – dependencies to other applications or services
(Intra-) Application domain
System software domain
Machine and device domain
Network domain

Figure 22: Instrumentation in the managed domain of an application.

Standards that are most relevant are discussed in the following sections. Some of these standards are discussed in deeper detail in subsequent chapters in the context of web services.

7.3 General Interaction Pattern for Instrumentation Protocols

The interaction between server and agent is a general. Basic operations are sown in the following table:

get(named-data-element) → value	Initiated by the server, the agent will read the value of the named data element (MIB variable in SNMP) and send the value back to the server in the reply message (also referred to as "pull model").
set(name-data-element, value)	Initiated by the server, the agent will set the named data element (MIB variable in SNMP) with the specified value. No reply is returned. Setting the value of a data element may initiate a control operation in the element. For this reason, the server uses the set method to actuate control operations in a managed element.

set(name-event, handler)	Initiated by the server, the agent will memorize a handler address to be notified when the named event occurs (this operation is not supported in SNMP, where predefined trap addresses are used for notifications, but available in other systems).
notify(event)	Initiated by the agent, the agent will send a notification message (called trap in SNMP) informing the server that has registered an event handler before in the agent, about the occurrence of an event ("push model").

Table 5: Basic operations for instrumentation in a managed domain.

In order to receive an event notification, the management process usually interrupts its current activity in order to react to the event without delay.

The pattern for instrumentation protocols in a managed domain is not bound to SNMP or other management protocols used in management systems. As a pattern, it also applies to "protocols" between acting people, managers and employees as it has been discussed in sections 6.1 and 7.1 for management processes in application creation and deployment.

7.3.1 CMIS and CMIP (OSI Management)

The Common Management Information Protocol (CMIP) is a network management protocol built on the Open Systems Interconnection (OSI) communication model. The related Common Management Information Services (CMIS) defines services for accessing information about network objects or devices, controlling them, and receiving status reports from them.

CMIS/CMIP is designed to run on the OSI protocol stack and has two disciplines. The first is CMIS which defines all of the services available for management. While the second, CMIP, the actual protocol which the services use to gather information. This model allows for management at all levels of the OSI model and is also not a protocol which has unintelligent agents, but instead the agents on the managed nodes are more intelligent than their SNMP counterparts. CMIS provides for one similar actor to retrieve and modify data on the managed stations: CMISE (Common Management Information Services Element) commands. These commands are an extended part of the CMIS services which has three categories:

- Management Association Services
- Management Notification Services

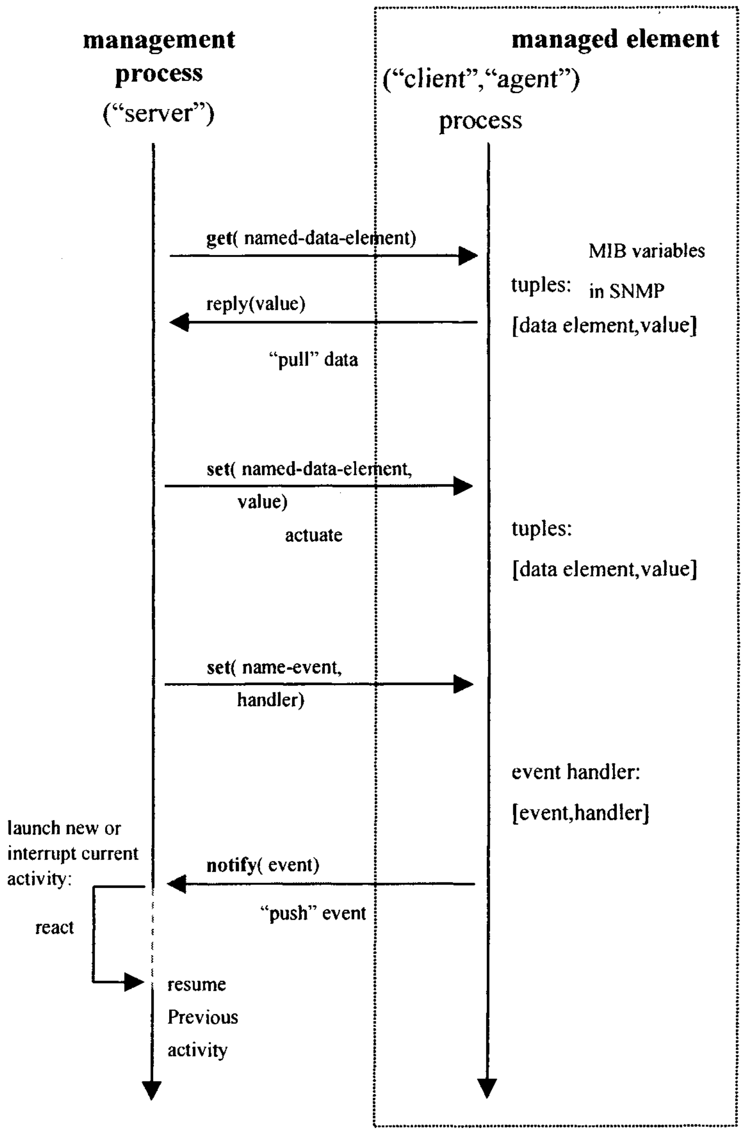


Figure 23: General interaction pattern for instrumentation protocols in a managed domain.

- Management Operation services

Within the Management Association services there are three sub services:

- M-INITIALIZE-generates a connection to a peer CMISE-service-user for network or systems management,
- M-TERMINATE-terminates the connection between peer service users,
- M-ABORT-used when a connection between two peers stops abnormally.

Management Notification Services, like SNMP traps, provides the managed stations the capacity to send events to management stations. This notification is handled via the M-EVENT-REPORT sub service.

Management Operation Services has six sub services that are similar to the SNMP commands. They are:

- M-GET-used to retrieve or gather management information,
- M-CANCEL-GET-used to cancel an M-GET request,
- M-SET-used to change/modify existing management information,
- M-ACTION-invoked by a CMISE-service-user to instruct a peer to perform some specified action (device dependant),
- M-CREATE-used to create a new instance of an object. An example being if there is a new device added to the network. It is possible with this service command to notify, other peers of the existence of the new station via an M-CREATE,
- M-DELETE-this is simply the opposite of the M-CREATE function.

As can be seen from these services there is an inherent intelligence to each of the managed elements or management processes. Drawbacks using CMIS/CMIP are that there are not many implementations of the OSI network protocol nor are there many CMIS/CMIP management systems available. Another problem is that this management protocol needs more resources to operate on network device than SNMP does.

7.3.2 OMG

The Object Management Group (OMG) is an international non-profit organization supported by information systems vendors, software developers and users which develops standards and specifications for object-oriented management environments, originally closely related to the Common Object Request Broker Architecture (CORBA).

The OMG Object Management Architecture provided the overall architecture for OMG standards. It is described in the OMG standard reference architecture which consists of a categorization of basic objects into:

- Object Services,
- Common Facilities,
- Domain Objects, and
- Application Objects.

An OMG object service defines the interfaces and sequencing semantics to support building well-formed applications in a distributed object environment. In non-object software systems, a system's Application Program Interface (API) is defined by a monolithic interface. The OMG Object Services API is modular; particular objects may use a few or many Object Services. By being object-oriented, the OMG Object Services API is extensible and customizable; applications only need to use services they require.

The operations provided by Object Services are made available through the IDL Interface Definition Language (as defined in the OMG CORBA specification) or through proposed extensions to IDL compatible with the OMG Object Model.

8. SUMMARY

Every change in the business environment impacts the IT infrastructure. IT infrastructure and management systems must accommodate change in business needs at lower cost. Management products that have traditionally been focused on network and systems management have begun to transition into higher management layers of application management. The chapter discussed the application life-cycle management and the processes involved in it. It also discussed some of the existing relevant management standards, namely SNMP, CMIP, CORBA, which are discussed later in more detail. Some of these are discussed more details in the following chapters.

Chapter 4

ENTERPRISE MANAGEMENT AND WEB SERVICES

1. INTRODUCTION

The evolution of enterprise computing has had a strong impact on the role of system management. In the mid 1970s, mainframes were the primary source of computing power. With their ability to process large amounts of data and centralize information, mainframes became the workhorse of organizations that could afford them. As companies grew, mainframes became overburdened, and enterprises turned to lower cost minicomputers. By distributing the computing load, mainframe resources were freed for critical jobs.

Over time, the distribution of work across mainframes and minicomputers became common. However, technology continued to advance, and the personal computer began moving into the industry. With their low cost, personal computers could be found on most desktops, a trend that continues to this day. While PCs addressed some of the personal productivity needs of users, they failed to provide the compute power needed by technical and commercial users. During this time, workstations, servers, and client-server computing emerged and revolutionized how many tasks were accomplished.

Client-server and the emerging Web services technologies enable companies to easily distribute information throughout the organization and deliver higher performance and greater system flexibility than legacy systems and PCs. With the ability to be reconfigured quickly, data and applications can be moved to wherever they are needed most. Today, users can tap into distant corporate databases from their desktops or access files and applications on their own systems, even if they are far away.

Client-server and Web services technologies are not only connecting computers, they enabled organizations to do business anytime and anywhere. With the rise of distributed computing, organizations found that they needed a way to tie all of their disparate computing resources together into a coherent enterprise network.

Managing the enterprise involves managing the IT infrastructure (networks, systems), and managing the services thus came to be a critically important task for the enterprise administrators.

1.1 System Management in the Enterprise

System management and administration has long been a difficult issue for organizations. Typically, individual users managed their own desktops, departments managed minicomputers, and data centers managed mainframes. Without corporate standards for resource configuration and management, organizations found it difficult to determine where systems were deployed, how they were employed, and when they needed maintenance.

Over time, administrators assumed the responsibility for desktop configuration and maintenance. System administration became typified by the number of administrators supporting systems for small groups of users. Each group utilized different computing resources, management tools and techniques, and procedures. This diversity resulted in configuration inconsistencies, a high cost of ownership, significant enterprise network complexity, and the need for a large, highly skilled administrative staff.

In an effort to improve efficiency, administrators standardized corporate procedures and began using cross-network tools supporting administrative tasks. Although limited in functionality, these tools helped administrators perform work remotely; they ensured network performance, and simplified monitoring and maintenance of computing hardware. While hardware components could be monitored, administrators still lacked the tools to address software, application, service, and platform configuration and availability to meet the increasing requirements placed on the IT infrastructure.

1.2 Changing Requirements in IT Infrastructure

IT organizations must provide more information faster and with greater reliability and accuracy in order to remain competitive in an increasingly efficient global marketplace. Increased competition and new market opportunities are driving widespread changes in the way organizations use information technology. Organizations are discovering they need new ways to differentiate. Organizations have understood that technology, better data distribution, integrated business processes, and networked communications are essential to improve customer relationships, increase global corporate collaboration, streamline overhead costs, and use information more effectively in order to enhance the value of products and services to customers.

In addition, the changing nature of computing paradigms and technologies, combined with insufficient system management tools, has left IT organizations with environments that are expensive to manage. This struggle continues to worsen as the demand for mission-critical computing grows.

Mission-critical environments and customer demand are raising expectations of acceptable service levels. As a result, distributed computing environments must be more reliable, available around the clock, and easier to diagnose and service. Systems must run continuously for longer periods of time without interruption. Problems must be isolated and repaired without impact on business operation. Disruption of platform availability during routine maintenance must be avoided and critical applications and services must always be available upon demand.

1.3 Enterprise Management

While every organization aims to meet the needs for increasing levels of service, it is necessary to keep associated support costs low. IT organizations must manage the enterprise infrastructure, including desktop and server systems, data storage systems, applications, and the networks that connect them, and maintain high levels of services. To be effective, IT organizations must also reduce complexity and cost. They must lower human costs, and consequently minimize the number of people required to keep the enterprise IT systems operating efficiently.

These demands have left IT organizations struggling with the complexity of the enterprise and the networked systems they use. Rapidly changing trends in IT technology require constant training and enhancements in knowledge and skills, putting additional burden on system administrators to keep pace with technology advancements. While the number of trained personnel is decreasing, the cost of maintaining them is rising. One important way to reduce costs consequently is to simplify the job administrators do most, system management, specifically low-level system management tasks by automating those tasks.

1.4 Role of System Management in the Enterprise

Enterprise management is a complex aggregate of tasks that enables organizations to manage systems, applications, and the networks that connect them. Achieving the continuous operation of servers, desktops, storage subsystems, relational databases, transaction monitors, and complex applications like SAP is difficult and time consuming. While no single application can provide an optimal enterprise management solution that meets every requirement on every platform, organizations can employ applications that are designed to solve common problems that can be adapted and extended to meet the specific needs of a particular organization. Those systems are called enterprise management systems. Prominent examples of widely used products in that area are the OpenView suite from Hewlett-Packard, Tivoli from IBM, and the management products from Computer Associates (CA).

One component of enterprise management, and perhaps the most pressing problem organizations face, is systems management. Focusing on the management of the hardware platform, operating system, and storage components, systems management provides the monitoring, performance, and resource management that is essential to ensure that systems remain operational. Monitoring and adjusting resource utilization allows that application response time and service level requirements can be met. In addition, storing data for trend analysis enables capacity planning and resource management. When systems operate efficiently, organizations are better equipped to make better decisions faster, a factor that can make the difference in an increasingly competitive marketplace.

2. ENTERPRISE MANAGEMENT SYSTEMS

Enterprise management systems refer to infrastructure, applications deployed and operating in that infrastructure and to services comprised of applications.

Enterprise management comprises three major areas:

- Infrastructure Management:
    - Network Management,
    - System Management, and
    - Storage Management.
- Application Management.
- Service Management.

Conventional tools for system management typically do not support the migration from managing small domains of disparate systems, such as data centers or services in those data centers, to managing the integrated enterprise. IT organizations need an integrated set of systems management tools or a platform that offers common services for all enterprise management applications. Such solutions must have a consistent look-and-feel, implement enterprise-wide security procedures, integrate with other management tools, and enable new functionality. Chapter 9 addresses this problem space in deeper detail.

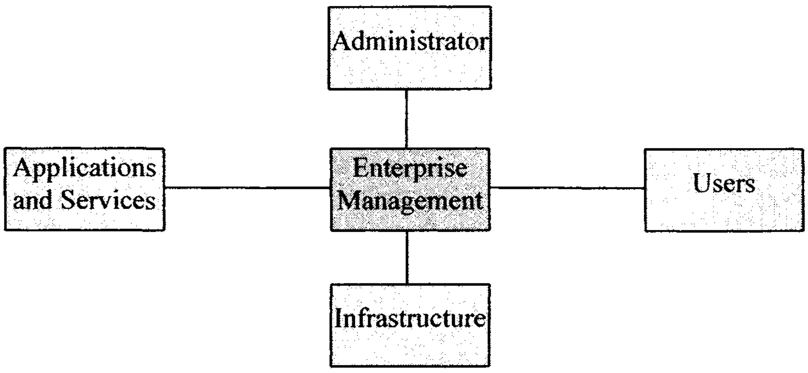


Figure 24: Dimensions of enterprise management.

Key capabilities required by an enterprise management system are:

- Common management platform, scalable from a single system to thousands of server and desktop systems.
- Single point of management, enabling effective use of administrative resources.
- Proactive and automated management of complex and common administrative tasks, reducing the likelihood for errors and helping to ensure availability.
- Configuration flexibility, enabling systems to be configured out of the box to best fit the needs of the environment, as well as providing easy customization for new rules, scripts, actions, etc.

- Management agent architecture, enabling administrators to add functionality and management features.
- Dynamic management agents, enabling functionality to be extended simultaneously at different agent locations in the management system independently of one another on an as-needed basis.
- Active configuration management controls, providing a secure interface for remote dynamic reconfiguration.
- Single event model, enabling information to be shared with multiple consoles for multiple administrators working simultaneously.
- Multiple system support, enabling administrators to monitor and manage systems remotely.
- Predictive failure analysis, enabling administrators to predict potential memory and disk hardware failures on a statistical basis, thereby enhancing decision making and increasing availability.
- Health monitoring, based on a sophisticated set of heuristics that incorporates administrative knowledge, including an intelligent rules-based health monitor that correlates metrics and gives suggested steps for problem resolution.
- Log-file scanning, enabling administrators to search and parse logs and registers for particular status information.
- Logical element grouping, enabling the grouping of systems by geographic location, server role, administrative responsibility, etc.
- Hierarchy and topology viewer, a central management application that displays the hierarchy and a topology map of all the objects that are being managed.
- Automatic discovery of systems, including IP addresses, subnet addresses, hostnames, and object IDs to identify specific types of systems.
- Physical system views, displaying images of hardware components and pointing to components associated events, enabling administrators unfamiliar with a particular platform to quickly determine which components need to be replaced in case of failure.
- Logical system views, presenting a tree hierarchy of managed domains, including all hardware and operating system

components. If an event is associated with a particular component, the logical view will identify its exact location within the logical hierarchy.

- Event and alarm management, providing administrators with the information they need in case of triggered events or alarms.
- Enterprise-wide security measures, such as authentication, data integrity, and access control lists for management of data and active management functions.
- Real-time performance analysis, enabling administrators to isolate potential and existing bottlenecks.
- Standard interfaces and protocols enable integration with third-party management tools providing a complete enterprise management solution across domains that are managed by individual management products and management solutions.
- A common graphical user interface look-and-feel and the ability to manage enterprise IT systems from any console.
- Trend analysis and historical data storage, supporting system performance analysis, configuration fault analysis, system sizing, and long-term capacity planning and forecasting.
- Host application and database fault management, increasing data and service availability through integration with third-party products.
- Firmware and patch management, providing administrators with the ability to upgrade systems when needed.
- On-line diagnostics, enabling hardware and software problems to be predicted, detected, isolated, and resolved, ensuring system and service availability.

2.1 Agent-based Management Infrastructure

Enterprise management technology based on a management agent architecture allows management agents to act in management roles in their respective domains and to execute management code that monitors and controls managed systems by sending requests to them and receiving data. Management agents also have access to critical management information and respond to higher-ordered manager requests. The typical organization of management agents is in form of a hierarchy with individual domains of responsibility.

Utilizing an agent-based approach, enterprise management systems enable local processing, within the domain of responsibility where associated managed objects are located, rather than at a centralized location, as it has been traditionally the case in centralized management systems. By distributing management functions, enterprise management systems provide higher reliability, availability, and serviceability in the enterprise network. Enterprise management systems today may also employ autonomous agents, a technique in which agents are not dependent on other software components and all data collection and processing is done locally by intelligent agents.

Typically based on SNMP, these agents collect and process data locally, and can act on data in order to send SNMP traps, run processes, etc., even if the connection to their next-level managers is lost. These intelligent agents can also initiate alarms, notifications, or specific actions based on collected data or messages through customizable rules and thresholds.

Since no product can meet every need of enterprise management, extensible modules can dynamically be loaded into management agents without disruption of the overall management system. This provides administrators with a flexible and extensible management application framework that can be tailored to a comprehensive enterprise management solution that can be adapted to constantly changing needs.

2.2 Three-tiered Management Architecture

An enterprise management system typically forms a three-tier architecture that provides a high level of scalability and distribution.

- *Console Layer* – the console layer constitutes the user interface of the enterprise management system. The console provides administrators with visual representations of managed objects, as well as the ability to manipulate attributes and properties associated with them. The console should be platform-independent, portable and easily customizable in order to be useful for a variety of computing environments and be adaptable to changing requirements.
- *Management Server Layer* – the management server layer provides typical management services. The management server layer can support multiple consoles, enabling several administrators to view the enterprise network simultaneously. The management server layer consolidates and optimizes multiple console requests minimizing network traffic. Communication with the console layer is routed through a well-defined client application programming interface. This interface can also be used by third-party tools to gain access to data collected by management

agents. All users are authenticated ensuring that only users with administrator roles have access and manage only the systems within their control. Additional server agents provide management services, including topology, event, and data management services.

- *Management Agent Layer* – the management agent layer consists of agents and probes that manage objects such as desktops, servers, storage subsystems, and network hardware and software. The agents typically utilize rules-based technology to determine the status of managed objects. Agents can then generate alarms or perform actions based on detected conditions. Agents can be sophisticated providing auto-management capabilities including predictive failure analysis.

2.3 FCAPS Management in the Enterprise

Typical FCAPS management tasks apply in enterprise management. FCAPS encompasses the domains of Fault, Configuration, Accounting, Performance, and Security management (FCAPS) [Sloman 1996].

Fault Management in the Enterprise

The detection, isolation, and replacement of failed or failing components is essential to ensure that systems, applications, and services remain available. Without on-line diagnostics, administrators must wait until a component fails. Predictive analysis is highly desired. In order to determine the state of components, including the firmware revisions and operating system patches installed, administrators need a comprehensive set of on-line diagnostics that can test and validate hardware, software, and network components.

Enterprise management systems provide administrators with on-line diagnostic capabilities. With enterprise diagnostic systems, system administrators and users can perform runtime diagnostics, isolate faults, and obtain status information on all devices, enabling the resolution of failures.

The on-line diagnostics tools enable administrators to test and validate the configuration and function of server hardware and network devices. Troubleshooting of hardware components must be supported. In addition, administrators can construct a set of actions to be executed automatically by the enterprise management system when diagnostic errors occur reducing the time required to fix problems and increasing system, application, and service availability.

Configuration Management in the Enterprise

Greater use of flexible system architectures and constant changes in products are just two reasons that lead to significant increase in systems integration, scale and complexity. This has produced the need for a configuration management environment capable of supporting many different viewpoints, from strategic planning through to operational deployment and eventual system withdrawal.

Configuration Management Systems include:

- Organizational Components. Identify the organizational units or teams responsible for configuration management activities.
- Development Platform Engineering Team. Identify and describe the integration of release management procedures with informational and technical requirements necessary for release of software for test and production.
- Configuration Management Responsibilities. Describe the responsibilities of the organizational components performing configuration management activities.
- Configuration Item Identification. Identify who is responsible for configuration item identification and describe what will be accomplished.
- Configuration Control. Identify who is responsible for configuration control and describe the procedures for meeting the configuration control.
- Configuration Status Accounting. Identify who is responsible for configuration status accounting and describe the procedures for meeting the configuration status accounting (baselines, records, databases, etc.) requirements.
- Audits and Reviews. Identify the responsibilities involved in audits and reviews and define what will be accomplished during each review.
- Scheduling and Tracking of Activities. Describe the scheduling and tracking of configuration management activities.
- Library Management. Describe the procedures to be used for development library environment management, validation, testing, and release certification.
- Tools, Techniques, and Methodologies. Identify tools and techniques and describe any unique methodologies used to accomplish effective configuration management.

Patch Management can be categorized into four phases: Assessment, Testing, Deployment and Audit.

Security Configuration Management is another important aspect of configuration management in the enterprise. It is discussed under the security topic of FCAPS management below.

Accounting Management in the Enterprise

Accounting management in general determines who is using what, to what extent and for what purpose. Accounting systems in the enterprise are of complex nature.

Accounting consists of a meta-cycle of:

- Defining an accounting model (data and events).
- Defining update cycles for that meta cycle.

The accounting cycle then is driven by the meta model:

- Collect accounting events.
- Aggregate and process them.
- Assign accounting events to accounts.
- Store accounting events in accounts for later inquiry.

Based on these cycles, accounting information is used for a multitude of inquires in order to derive financial statements, general ledger and sub-ledger information providing valuation and recording of financial data as the basis for financial statement reporting, revenue and cost accounting, order and project accounting, and product and service cost calculation determining unit costs for all products and services. Accounting tools are closely tied with Customer Relationship Management (CRM) systems [Dyche 2001].

Typical accounting systems also include a variety of reporting tools. Inventory tools give businesses the ability to categorize entities, items and transactions by location for easier management, tracking and reporting of inventories across multiple locations.

Performance Management in the Enterprise

Keeping the network up and running is a critical and time-consuming task. Administrators need systems that help them to understand network performance. Real-time performance analysis capabilities of enterprise management systems enable administrators to determine system workloads. When system and network reconfigurations are performed or new equipment is added, the immediate impact of those actions can be analyzed and used to guide performance tuning efforts.

To ensure resources are being used to their full potential, IT organizations perform a variety of analysis and planning tasks. With enterprise management systems, administrators can collect and centrally store critical system performance and configuration data for fault analysis, system sizing, and long term capacity planning. An interface to that data, typically maintained in relational databases, enables to use analysis tools and automate and customize data analyses tasks.

Security Management in the Enterprise

Large and small organizations alike rely on trusted security mechanisms to protect their networks and consequently their business functions. Many industries such as engineering, finance, health care, and government, need the highest levels of security and guaranteed privacy for their systems and networks.

Enterprise management systems include a variety of security mechanisms to ensure users are authenticated and granted access only to the areas of the system to which they are permitted. A multitude of security models exists.

Security features of typical enterprise management systems include:

- user or administrator identification and authentication,
- administrative roles and domains,
- access rights of roles in domains,
- encryption as basis of security.

Security configuration management is the ongoing configuration maintenance cycle that is associated with security. Security configuration management tools can fix vulnerabilities before problems occur. Security configuration management is comprised of patch management, system hardening, security operations, and security auditing.

System hardening is the constant evaluation of a company's security architecture and audition of system configurations according to recommendations such as issued by the Network, Security (SANS) Institute or The Center for Internet Security (CIS). Vendors such as Microsoft also provide guidelines such as the Microsoft Security Operations Guides.

Security operations are tasks such as changing passwords for accounts with high authority, periodically alerting certain accounts to change passwords, etc. Without automation, these important tasks will remain undone in most IT organizations.

Lack of security auditing in large organizations will inevitably cause security configurations to “drift” over time. To ensure that security policies are enforced, systems must be audited periodically.

3. INTEGRATED IT SERVICE MANAGEMENT (ITSM)

One origin of Integrated IT Services Management (ITSM) can be found in systems management historically done in large-scale mainframe environments. Through constant refinement over the years these services and functions attained a high level of maturity. ITSM is a methodology, which is comprised of a set of best-practices documents and guidelines, which have been described in the IT Infrastructure Library (ITIL).

Although the UK Government created the ITIL, it is has been rapidly adopted across the world as the standard for best practice in the provision of IT Services. Although the ITIL covers a number of areas, its main focus is on IT Service Management. ITSM provides process-based, integrated services with a focus on business requirements.

The ITIL covers six areas:

- Service Support,
- Service Delivery,
- Planning to Implement Service Management,
- Infrastructure Management,
- Applications Management, and
- The Business Perspective.

ITSM is generally divided into two main areas:

- Service Support and
- Service Delivery.

Together, these two areas consist of various disciplines that are responsible for the provision and management of effective IT services. Those disciplines are discussed in the forthcoming sections.

In today's web-services world, IT organizations need to be increasingly flexible, agile, and cost efficient to be aligned with ever evolving business requirements, for which IT service management is crucial.

ITSM is a business-driven approach to IT management that specifically addresses the strategic business value generated by IT organizations and the need to deliver superior IT service. It is designed to address people, processes and technology issues that all IT organizations face.

3.1 The IT Infrastructure Library (ITIL)

Today, many IT or service provider organizations face the challenge of shifting paradigms from infrastructure management toward service management. The IT Infrastructure Library (ITIL) has become the most widely accepted approach to IT service management in the industry.

ITIL is the basis for ITSM. ITIL provides a comprehensive and consistent set of best practices for IT service management, promoting a quality approach to achieving business effectiveness and efficiency in the use of information systems.

ITIL is based on the collective experience of commercial and government practitioners worldwide. This has been combined into one reliable, coherent approach, which is fast becoming the de facto standard used by many of the world's leading businesses. A wide range of products and services are available to support these initiatives. ITIL products include ITIL books, ITIL Qualifications and Certification, and the IT Service Management Forum (itSMF). Commercial elements provided by companies include consulting, solutions and training.

The itSMF is a non-profit organization, wholly owned and operated by its members. As the official ITIL user organization, it is dedicated to set standards for best practices in IT Service Management.

3.1.1 The ITIL Toolkit

The ITIL Toolkit represents the implementation of the ITIL. It is designed to help guide through ITIL/ITSM. It contains a series of components and resources to simplify, explain and manage ITIL processes.

It comprises a series of resources:

- An ITIL Guide – a detailed and comprehensive introduction to ITIL, targeted at both beginners and experienced practitioners.
- An ITIL Management Presentation – a full presentation on ITIL and service management. It explains ITIL with detailed notes.
- The ITIL Fact Sheets – a reference kit comprising a series of ITIL fact sheets. These cover each of the main ITIL disciplines.
- An ITIL Compliance Assessment Kit – a comprehensive questionnaire set designed to help assess the compliance position with ITIL and identify which areas need attention.
- ITIL Presentation Template – to help to interpret compliance assessments.

3.2 ITSM General Methodology

ITSM and ITIL, upon which it is based, provide an integrated, process based set of best practices to manage IT services, including centralized services and decentralized services. Whereas ITIL defines and documents the best practices, ITSM employs them to meet unique customer requirements and priorities.

A roadmap to IT Service Management starts with an organization strategy and requirements assessment, which determines the organization's requirements both now and in the future. This initial assessment determines the current, existing IT infrastructure, processes, and services. It also provides an understanding of a customer's desired future state of IT and the services that it needs to achieve the goal across the enterprise. The details of this approach are shown in the following diagram.

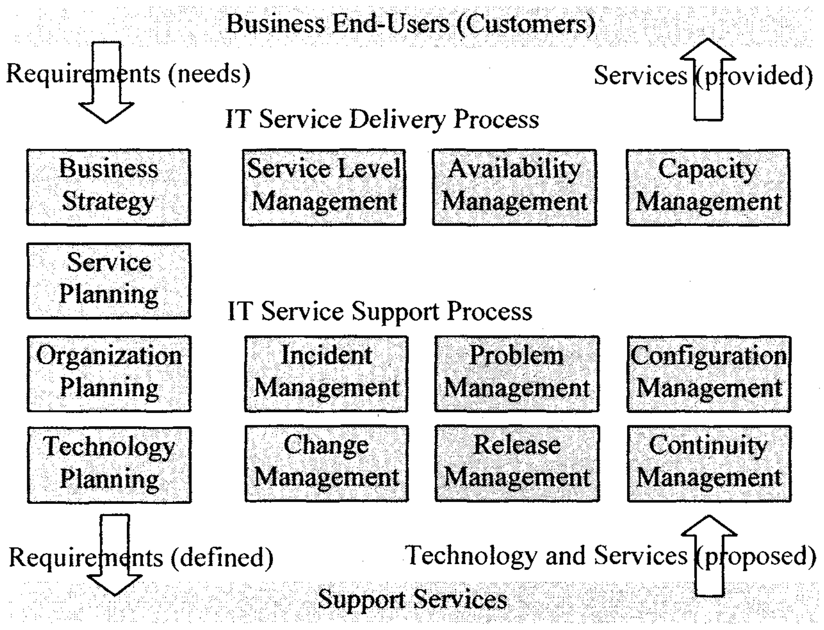


Figure 25: ITSM Methodology.

Requirements Definition Process

The requirements definition process focuses on areas within the organization and IT infrastructure to determine requirements and metrics in a top-down approach starting from business strategy down to technology:

- Business – determines what are the requirements driven by the organization business needs.
- Service – determines what services need to be provided to satisfy those requirements.
- Operational – determines what IT infrastructure is needed to support the services.
- Technology – determines what technology is needed for that IT infrastructure.

The following strategy and planning processes take place within the definition stage of the requirements definition process:

- Business Strategy – how are current business requirements linked to technology infrastructure, what are the processes that provide that linkage.
- Service Planning – how does IT provide services internally and externally for the organization, what are the processes for providing those services.
- Organizational Planning – how does the organization adapt to internal and external business factors that impact the IT infrastructure, what is the impact of how IT is integrated within the organization.
- Technology Planning – how does IT plan its technology infrastructure internally and externally around the organizations business requirements and what models define relationships?

Once the definition of requirements is accomplished, ITIL best practices are employed to develop the necessary IT Services Support and Service Delivery processes. Together this provides an enterprise wide ITSM solution based on ITIL best practices that are tailored to the organization's specific business and IT infrastructure requirements.

3.3 IT Service Management Processes

ITSM encompasses the following areas as basic areas of ITIL: Service Delivery Processes and Service Support Processes.

Service Delivery Processes:

The Service Delivery Process comprises:

- Service Level Management based on Service Level Agreements – maintain and improve the level of service to the organization.

- Availability Management – optimize IT infrastructure capabilities, services, and support to minimize service outages and provide sustained levels of service to meet business requirements.
- Capacity Management – enables an organization to manage resources and strategically plan for future resource requirements.

Service Support Processes:

- Incident Management – the day-to-day process that restores normal acceptable service with a minimal impact on business.
- Change Management – standard methods and procedures for effective managing of all changes.
- Problem Management – the diagnosis of the root causes of incidents in an effort to proactively eliminate and manage them.
- Release Management – testing, verification, and release of changes to the IT environment.
- Configuration Management – physical and logical perspective of the IT infrastructure and the provided IT services.
- Continuity management – the process by which can be ensured that IT Services can recover and continue should a serious incident occur.

Depending on the ITSM consulting methodology that is employed, additional value-added areas can be included. These areas could be separate, but also dependent on those listed above, such as print management, or they could be sub-processes, such as IT strategy development.

3.4 ITSM General Implementation

A typical ITSM implementation encompasses the following stages:

- Determine the current, existing IT infrastructure, processes, and services.
- Develop the desired future state of IT and the services that it needs to provide.
- Architect a roadmap that shows how to get to the desired state from the current state.
- Determine the steps needed to execute the roadmap.

Following those stages, an ITSM implementation can be modeled as a framework with five phases:

- Assessment – determine the current state and begin to collect and understand the metrics for the future desired state.
- Architect and Design – develop a mature design for the future desired state.
- Planning – develop the plans necessary to achieve the future desired state in form of a phased roadmap.
- Implementation – implement and deploy the plans within IT and across the enterprise to approach the future desired state.
- Support – manage, maintain, and improve the future desired state by being able to adaptively integrate enhancements as needed or required.

Within this framework, managing IT as an enterprise wide, service oriented entity typically comprises one or more of the following perspectives:

- People – quantity and quality of expertise and knowledge.
- Processes – IT and organization specific practices, procedures, guidelines, etc.
- Technology – the total logical and physical technology infrastructure consisting of hardware, software, communication networks, applications, data models, etc.
- Organization – internal and external business factors that affect IT and how IT interfaces with an organization.
- Integration – how is IT integrated within the business model, what services does IT provide, how are the services provided, and how are best practices employed within IT.

3.4.1 Service Delivery Processes

Service Delivery is the management of the IT services themselves, and involves a number of management practices to ensure that IT services are provided as agreed between the Service Provider and the Customer.

Service Delivery has been introduced as following disciplines:

- Service Level Management based on Service Level Agreements,
- Availability Management, and
- Capacity Management.

3.4.1.1 **Service Level Management based on Service Level Agreements**

Service Level Agreements (SLA) are fundamental to business continuity. They define minimum levels of availability from key suppliers, and often determine what consequences will be taken in the event of disruption.

In order to define SLA, the following areas are usually considered:

- Agreement statistics – such as what is included within the agreed service.
- Availability – agreed service times, response times, etc.
- Help Desk Calls – number of problems reported, response times, resolution times.
- Contingency – agreed contingency details, location of documentation, contingency site, third party involvement, etc.
- Capacity – performance timings for online transactions, report production, numbers of users, etc.
- Costing Details – charges for the service, and penalties in cases that SLA have not been met.

3.4.1.2 Availability Management

Availability Management is the practice of identifying levels of IT Service availability. All areas of a service must be measurable and defined within the Service Level Agreement in order to measure and validate availability of a service.

Availability is usually calculated based on a model involving the availability ratio and techniques such as Fault Tree Analysis. These techniques include the following elements:

- Reliability – the time for which a component can be expected to perform under specific conditions without failure.
- Recoverability – the time it should take to restore a component back to its operational state after a failure.
- Maintainability – the ease with which a component can be maintained, which can be either remedial or preventative.
- Resilience – the ability to withstand failure.
- Security – the ability of components to withstand breaches of security.

IT Security is an integral part of Availability Management. Some of the above elements are the outcome of performing a risk analysis to identify any resilience measures, identifying how reliable elements are and how many problems have been caused as a result of system failure.

3.4.1.3 Capacity Management

Capacity Management is the discipline that ensures IT infrastructure is provided at the right time in the right quantity at the right price, and ensuring that IT is used in the most efficient manner.

This involves input from many areas of the business to identify what services are (or will be) required, what IT infrastructure is required to support these services, what level of contingency will be needed, and what the cost of this infrastructure will be.

The following elements are inputs into Capacity Management processes:

- Performance monitoring,
- Workload monitoring,
- Application sizing,
- Resource forecasting,
- Demand forecasting, and
- Modeling.

Capacity management is driven from these elements in combination with the capacity plan itself, forecasts, tuning data and service level guidelines.

3.4.2 Service Support Processes

Service Support has been introduced as following disciplines.

- Incident Management,
- Change Management,
- Problem Management,
- Release Management,
- Configuration Management, and
- Continuity management.

3.4.2.1 Incident Management

Incident Management is the resolution and prevention of incidents that affect the normal operation of IT services. This includes ensuring that faults are corrected, preventing any recurrence of these faults, and preventative maintenance to reduce the likelihood of these faults.

The effective practice of both Incident and Problem Management ensures that the availability of IT services is maximized.

3.4.2.2 Change Management

Change Management is the practice of ensuring all changes to configuration items are carried out in a planned and authorized manner. This includes ensuring that there is a business reason behind each change, identifying the specific configuration items and IT services affected by the change, planning the change, testing the change, and having a roll back plan in case that the change results in an unexpected state of affected configuration items.

IT Security must be embedded into the change management process to ensure that all changes have been assessed for risks. This includes assessing the potential business impacts of changes.

3.4.2.3 Problem Management

Problem Management is the resolution and prevention of incidents that affect the normal operation of IT services. Problem management includes:

- Problem detection,
- Problem reporting, and
- Problem resolution.

The Service Help Desk plays an important part in problem management. It is very often the first contact when problems are encountered. The Service Help Desk is a single point of contact for end users who need help.

There are different types of Help Desks, the selection of which is dependent upon what the business requires. Some Help Desks provide a simple call logging function, and escalate calls to more experienced and trained staff. Others provide a high degree of business and technical knowledge with the ability to solve most problems when they are reported.

Service Help Desks embrace the following functions:

- receive all calls and e-mails about problems;
- problem recording;
- problem prioritization, classification and escalation;
- problem solution;
- update the end user on progress;
- communication with other ITSM processes;
- report to management, process managers and customers.

3.4.2.4 Release Management

This discipline of IT Service Management deals with all software configuration issues within the organization. It is responsible for the management of software development, installation and support of an IT organization's services.

Services are often not regarded as tangible assets, which results in inefficient attention and control. Unauthorized software modifications can lead to fraud, viruses, and malicious damage to data files and other severe consequences. It is important that release management procedures be reviewed and impact on services be assessed. Back out plans are essential.

3.4.2.5 Configuration Management

Configuration Management is represented and documented in the Configuration Management Database. This database contains details of the organization's elements that are used in the provisioning and management of its IT services. This database is more than an asset register, as it contains information that is related to maintenance, migration, and experienced problems with configuration items.

The Configuration Management Database holds a much wider range of information about items including hardware, software, documentation, and personnel.

Configuration Management essentially consists of four tasks:

- Identification – specification, identification of IT components.
- Control – management of each configuration item, also specifying who is authorized to manage (and change) it.
- Status recording – this task deals with the recording of the status changes of all configuration items registered in the configuration database, and the maintenance of this information.
- Verification – this task involves reviews and audits to ensure the information contained in the configuration database is accurate.

Without the definition of all configuration items that are used to provide IT services, it can be very difficult to identify which items are used for which services. This could result in critical configuration items being lost, moved or misplaced, affecting the availability of the services. It could also result in unauthorized items being used in the provisioning of IT services.

3.4.2.6 Automated Discovery of Systems and Components

Large organizations often find it difficult to keep track of systems. To be effective, IT organizations must find better ways to keep track of constantly changing and growing configurations. By knowing which systems are employed where, resources can be moved more efficiently as needs dictate. Enterprise management software helps administrators obtain system information through automatic discovery. For each system, IP address, subnet address, and hostname information can be discovered and collected automatically. In addition, object IDs can be used to identify specific types of systems. As a result, operators can monitor how systems are deployed, as well as how the networking infrastructure is employed.

3.4.2.7 Continuity Management

Continuity management is the process by which plans are put in place and managed to ensure that IT Services can recover and continue in case of a serious incident. Continuity management is not only about reactive measures, but also about proactive measures – reducing the risk of severe business impact.

Continuity management is so important that many organizations will not do business with IT service providers if contingency planning is not practiced within the service provider's organization. It is also a fact that many organizations that have been involved in a disaster where their contingency plan failed lost business within 18 months after the incident.

Continuity management is regarded as the recovery of the IT infrastructure used to deliver IT Services. Many IT organizations practice a much far-reaching process of Business Continuity Planning, to ensure that the entire end-to-end business process can continue in case of a serious incident.

Continuity management involves the following steps:

- Prioritizing the businesses to be recovered by conducting a Business Impact Analysis.
- Performing a Risk Assessment (or Risk Analysis) for each of the IT Services to identify the assets, threats, vulnerabilities and countermeasures for each service.
- Evaluating the options for recovery.
- Deriving the Contingency Plan.
- Testing, reviewing, and revising the plan on a regular basis.

Business Continuity Planning and Disaster Recovery Planning.

Business continuity planning and disaster recovery planning are vital activities. The creation and maintenance of a business continuity and disaster recovery plan, is a complex undertaking, involving a series of steps.

Prior to creation of the plan itself, it is essential to consider the potential impacts of disaster and to understand the underlying risks. Following these activities, the plan must be created. The plan must then be maintained, tested and audited to ensure that it remains appropriate to the needs of the organization.

Business Impact Analysis is essential to the business continuity process. This involves considering the potential impact of each type of disaster or event. At a basic level, business impact analysis is a means of systematically assessing the potential impacts resulting from various events or incidents.

After having determined impacts, it is equally important to consider the magnitude of the risks, which would be caused by these impacts. This activity will determine which scenarios are most likely to occur and which should attract most attention during the planning process.

Disaster recovery policies underpin an organization's approach to contingency and disaster recovery. They are usually linked closely with security policies, both addressing the basic requirements to ensure the stability and continuity of the organization. It is essential therefore that they are up to date and remain up to date, are comprehensive, and it is essential to monitor contingency practices to ensure that they are met in practice.

4. MODEL-BASED MANAGEMENT

As systems become larger and more complex, one can expect increasing reliance on automatic or semi-automatic management tools that adapt to constant changes occurring in large and dynamic Enterprise IT infrastructures. Adaptivity of management systems to changing conditions in the existing system, to changing demands, and to changing business needs is the major challenge for Enterprise IT management systems.

In order to allow Enterprise IT management systems to react and to proactively prevent failure conditions, knowledge about the system which is present in human operators must be transferred into machine-usable representations, so-called models, which allow reasoning upon this information, assessing and interpreting it, deriving conclusions and reacting accordingly by issuing management and control instructions to systems in the IT infrastructure in an automated manner.

Models are comprised of:

- Static aspects (data centers, inventories of hard- and software, licenses, staff, etc.),
- Dynamic aspects (current load and utilization conditions, error conditions, etc.).

Several hard problems are inherent to model-based approaches:

- How to obtain models?
- How to represent models?
- How to keep models consistent with the reality they represent?
- How to deal with change?

- How to provide models that not only reflect the current state, but also capture future evolution, anticipation and expectations of future trends, etc.?

In reality it is often hard to obtain an integrated management perspective by model-based approaches. Models are often assembled bottom-up, and presented in a technical form. They are manually maintained by a staff of technical administrators and system architects.

Moreover, complex service hierarchies span many organizations crossing multiple organizational, administrative and jurisdictional boundaries and lead to inter- and intra- organizational dependencies. These dependencies relate to the entirety of resources, systems, tools, applications, people, workflows and processes that are necessary to operate, administer, manage and provision services in heterogeneous environments and thus have a great influence on the management processes of all involved organizations.

Despite the difficulty that is inherent to model-based approaches, there is really no alternative. Today, large IT organizations rely on large troops of IT staff troubleshooting and trying to keep systems operational and keep them up. This approach does not scale. It is a good average in the industry that one IT operator roughly manages 50 servers. With servers becoming cheaper, smaller and denser, the number of servers per IT operator must increase; in fact it must increase faster since there is cost pressure on the IT staff side as well.

This calls for more automation, primarily in low-end system management. Delegating those tasks to management systems requires representation of knowledge inside management systems, which is the core of the model-based management approach.

On the other side, the complexity, which is associated with models coming into systems, must be overcompensated by automated model generation, environment discovery and automated update cycles performed in models.

One major task of service management is to identify and to model dependencies between services, for example using roles, interactions and communication relationships.

In order to address some of the problems associated with service management, a generic service model can be introduced that defines commonly needed service-related terms, concepts and structuring rules in a general and unambiguous way [Hegering 1999]. This service model can be applied to a variety of scenarios and helps to analyze, identify and structure the necessary actors and the corresponding inter- and intra-organizational associations between these actors. Since it also covers the service life cycle, it helps to establish, enforce and optimize information flows between organizations or business units. Service models take a top down view and systematically abstract service and organizational details. This methodology ensures, that functional, organizational and life cycle aspects necessary for service management are considered.

The generic service model addresses the following requirements:

- Generic and abstract service definition: The model gives an abstract definition of a service and thus provides a common understanding for describing services dependencies for a particular scenario or environment.
- Integration of organizational aspects: The modeling approach defines a service as the association between organizations that provide and use services. It allows to model scenarios such as supply-chain and provider hierarchies.
- Separation of service definition and service implementation: The separation of the abstract service description from the corresponding service implementation enables providers to implement services according to their local environment without imposing changes on client services (service virtualization).
- Management as an integral part of the service: the model considers the management of services as an integral part of the service itself.

4.1 Models in Systems Management

IT infrastructures are influenced by several unpredictable factors such as the network performance, temporally varying workloads, and components of services contending for resources. These interactions are difficult to characterize because of their dynamic nature. These factors make it impossible to anticipate all problems during services design, deployment and test.

Model-based management [Pavlou 98] includes formulation and formalization of expected and agreed-upon behavior. This is subject to Service Level Agreements (SLA), with Quality of Service (QoS) expectations such as availability or response times. Operational services must be monitored for potential QoS violations and performance bottlenecks, with effective actions being taken to prevent these problems. Current management systems that monitor thresholds and trigger alarms rely on correct interpretation by the operator to determine causal interactions. This approach does not scale as the number of thresholds and alarms increase. For a scaleable solution, a management system must be able to monitor, diagnose and reconfigure service components to ensure user-level QoS goals. The management system must be proactive and coordinate with existing network management systems.

Model-based reasoning provides a framework necessary for articulating a management system with the above requirements. Model-based approaches allow describing the expected modes of operation of each component in the system. Furthermore, only the properties of interest are modeled. All other details are abstracted away. For example, when managing performance, only response time, visit count, utilization, and interaction characteristics of each component are modeled. Anomalies are detected by comparing measurements with baselines.

The industry is beginning to converge on standards such as DMTF's Common Information Model (CIM) that provide ubiquitous management information models, and on standards such as the Unified Modeling Language (UML) [Booch 1998]. Software design tools and design languages including UML encourage the design of systems as objects and resources, and describing the relationships between them. The emergence of these models and tools enable model-based management since significant parts of the management model can automatically be populated.

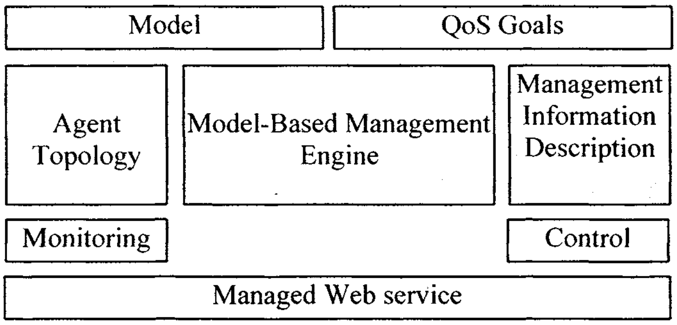


Figure 26: Model-based management system.

4.2 Models for Service Management

Several modeling approaches exist in different branches of the IT industry. Some of them are discussed next.

4.2.1 Common Information Model (CIM)

Distributed Management Task Force (DMTF) defines the Common Information Model (CIM). CIM introduces a management information model that allows integrating the information models of existing management architectures. CIM acts as an umbrella that also allows exchanging management information. The Common Information Model is a conceptual information model for enabling end-to-end management. Until now the focus has been on creating management information models for networks, storage, applications separately and often at odds with each other. Common Information Model has tried to establish an information model that spans various domains. It provides consistent definition and structure to data. The language that CIM uses for modeling is the Managed Object Format (MOF). CIM is independent of any instrumentation and is not tied to any particular information repository.

The CIM Core Model gives a formal definition of a service and allows hierarchical and modular composition of services consisting of other services. However, the focus in CIM is on rather technical details of components than of service composition (pg. 212) and does not include a notion of domains, such as customer and provider.

The Common Information Model (CIM) is a conceptual model for describing a business computing and networking environments with all the managed entities, their states, operations, composition, configuration, and relationships. Model contents are not bound to a particular problem domain or implementation, but address end-to-end management from clients, to servers, and over the network.

CIM addresses FCAPS management (fault, configuration, accounting, performance and security management) and supports the abstraction and decomposition of services and functionality. The information model defines and organizes common and consistent semantics for managed entities.

The organization of CIM is based on an object-oriented paradigm promoting the use of inheritance, relationships, abstraction, and encapsulation to improve the quality and consistency of management data. Object-orientation in CIM is applied in the following dimensions:

- *Abstraction and classification.* To reduce the complexity, high level and fundamental concepts (the objects of the management domain) are defined. These objects are grouped into types of management data (classes) by identifying their common characteristics and capabilities (properties), relationships (associations) and behavior (methods).

- *Object inheritance*. Additional detail can be provided by sub-classing. A subclass inherits all the information (properties, methods and associations) defined for its super class. Subclasses are created to classify the levels of detail and complexity at the right level in the model, and to deliver continuity of management information.
- *Dependencies, component and connection associations*. Being able to express relationships between objects is an extremely powerful concept. Before CIM, management standards captured relationships in form of multi-dimensional arrays or cross-referenced data tables. The object paradigm offers a richer and more expressive approach by that relationships and associations are now directly modeled. In addition, the way that these relationships are named and defined, gives an indication about the semantics of object associations. Further semantics and information can be provided in properties.

4.2.2 CIM Meta Model

The CIM Schema describes the core and common models. CIM is an object oriented model and has capabilities to represent types, and instances alike. CIM defines the notion of *Classes* and *subClasses*. Classes are types while subclasses are subtypes. *Instances* are instantiation of the Classes and subclasses and represent things. *Properties* are attributes and *Relationship* is pairs of attributes.

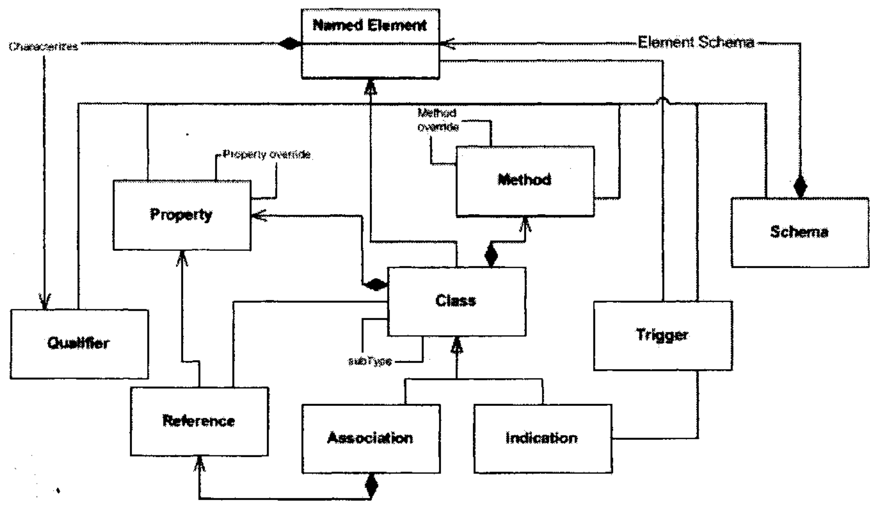


Figure 27: CIM Meta model.

A CIM Class is a blueprint that describes the properties and methods of a particular type of an object. Classes have properties and methods. A Class Name is unique in a particular schema. Properties are unique within a class. A property has a name, a data type, a value and an optional default value. The data types that CIM supports are uint8, sint8, uint16, sint16, uint32, sint32, uint64, sint64, string, boolean, real32, real64, datetime, ref (reference to a class name), char16.

Methods are unique within a class and represent operations that can be invoked. Return type of methods conform to data types supported by CIM. A method signature involves name of the method, return type, optional input parameters, and optional output parameters. Return types must not be arrays.

Qualifiers are used to describe additional information about classes, associations, properties, methods, indications, properties or references. Qualifiers have name, type, flavor, scope, and optional default value.

An Association has references to two or more classes. Associations represent relationships between two or more classes. It is treated as a separate object with references attached to it.

An Indication signifies occurrence of an actual event. They may have properties and methods and be hierarchically arranged. The indications are either life cycle indications or process indications. The life cycle indications signify creation, deletion or modification of a class or a creation, deletion, modification, method invocation, and read access of an instance. The process indications are other notifications that are not related to life cycle.

CIM Specification is described in a language called the Managed Object Format (MOF).

4.2.3 CIM Core and Common Model

The core model comprises of a basic set of classes, properties and methods that provide the basis for modeling of all managed systems. The Core and common models follows the CIM Meta Model. The Core and Common Models are together referred to as CIM Schema. The CIM Common Models capture information related to particular areas of management like systems, applications, networks, devices etc.

The core model lays down a basic classification of elements and associations. The managed element class which is an abstract class forms the base of the hierarchy. The Managed Element class is sub-classed into Managed System Element, Product related, Setting and Configuration related performance and statistical data related classes. Managed system elements are sub-classed further into Logical and Physical Elements.

The common models capture the information that is related to a management area but is technology and platform independent. The Common Models are continuously upgraded and added to.

The CIM Schema v 2.7 defines Common Models on the following

1. Applications: Deals with the structure of an application, life cycle and the transition between states in the life cycle of an application.
2. Database: This schema describes the database system that describes the application software, the logical entity of the database and the database service.
3. Devices: The CIM Device Common Model describes a range of hardware, their configuration information and their data. It covers concepts like sensors, fans and batteries to storage volumes.
4. Events: The CIM Event Common Model covers the aspects of publications, subscriptions and notifications.
5. Interop: The CIM InterOp Model describes the various Web Based Enterprise Management (WBEM) components, namely the CIM Client, CIM Server, CIM Object Manager, Provider. Please see more details in the next subsection.
6. Metrics: The CIM Metric Common Model tries to generalize the concept of transactions through the UnitOfWork concept.
7. Network: The CIM Network Model describes the network systems, network services, logical interconnections and accesses, network protocols (OSPF, BGP), networking technologies (VLAN, Switching/Bridging), Quality of Service (meters, markers, queues) and other related definitions.
8. Physical: The CIM physical Common Models describes the modeling details of Physical elements (elements that occupy space and follow laws of physics). CIM_Racks, and CIM_Chassis are defined for example as physical elements.
9. Policy: Policies are frequently describes as rules that change the behavior of a system. The Policy Model has been developed jointly by IETF and DMTF. It expresses policies as condition action pairs. The <condition> term is a Boolean expression used to specify the rule selection criteria. When for a resource the <condition> term evaluates to true the <action> term is invoked.
10. Support: The CIM Support model deals with standardized way to represent and communicate information, the process of obtaining information, publishing and interpreting support information.

```
//=================================================
===/
// A class example: ManagedElement
//=================================================
===/
[Abstract, Version ("2.7.0"), Description (
  "ManagedElement is an abstract class that provides a common "
  "superclass (or top of the inheritance tree) for the "
  "non-association classes in the CIM Schema.") ]
class CIM_ManagedElement {
  [MaxLen (64), Description (
    "The Caption property is a short textual description (one-"
    "line string) of the object.") ]
  string Caption;
    [Description (
    "The Description property provides a textual description of "
    "the object.") ]
  string Description;
     [Description (
    "  A user-friendly name for the object. This property allows "
    "each instance to define a user-friendly name IN ADDITION TO its "
    "key properties/identity data, and description information. \n"
    "  Note that ManagedSystemElement's Name property is also defined
"
    "as a user-friendly name. But, it is often subclassed to be a "
    "Key. It is not reasonable that the same property can convey "
    "both identity and a user friendly name, without inconsistencies. "
    "Where Name exists and is not a Kev (such as for instances of "
```

Figure 28: CIM MOF description.

11. Systems: CIM System Common Model defines computer system related abstractions. These systems are aggregation entities and are not modeled as collections. They also deal with concepts like systems, files, operating systems, processes, jobs, etc.
12. User: The CIM User Common Models deal with the general contact information related to users, organizations, units of organization etc and the clients of the services as “Users” and the security authentication and authorization related information.

4.2.4 CIM and Web-Based Enterprise Management (WBEM)

Web-Based Enterprise Management (WBEM) is an initiative coupling CIM and Internet standard protocols and encodings (such as XML and HTTP) with CIM. The WBEM architecture includes the notion of a CIM server and various providers of management data, such as instrumentations. The CIM server acts as an information broker between the providers of instrumentation data and management clients and applications. This approach shields providers from management clients and applications.

WBEM consists of a set of management and Internet standard technologies standardized by the Distributed Management Task Force (DMTF) with the goal to unify the management of enterprise computing environments. WBEM provides the ability for the industry to deliver a well-integrated set of standard-based management tools leveraging the emerging Web technologies. The DMTF has developed a core set of standards that make up WBEM, which includes a data model, the Common Information Model (CIM) standard; an encoding specification, xmlCIM Encoding Specification; and a transport mechanism, CIM Operations over HTTP.

WBEM is a set of technologies to unify enterprise management. WBEM is comprised of the CIM model that we described in the previous section, the xmlCIM encoding of CIM elements in XML, and CIM over HTTP specification that enables interoperation across CIM systems. WBEM operations could be simple or batched. The WBEM operationTypes are include data, metadata, Queries and methods. WBEM operations include: GetClass, EnumerateClasses, EnumerateClassNames, GetInstance, EnumerateInstanceNames, GetProperty, SetProperty, CreateInstance, ModifyInstance, DeletInstance, CreateClass, ModifyClass, DeleteClass, Associators, AssoicatorNames, References, ReferenceNames, ExecQuery, getQualifier, SetQualifier, DeleteQualifier, EnumerateQualifiers.

xmlCIM tries to create an XML based grammar expressed in a DTD so as to represent CIM information (classes, instances, methods), and CIM messages for use by CIM protocols. CIM-XML uses xml-CIM as the payload over HTTP transport.

The CIM specification is the language and methodology for describing management data. The CIM schema includes models for Systems, Applications, Networks (LAN) and Devices. The CIM schema will enable applications from different developers on different platforms to describe management data in a standard format so that it can be shared among a variety of management applications. The xmlCIM Encoding Specification defines XML elements, written in XML compatible to defined Document Type Definition (DTD), which can be used to represent CIM classes and instances. The CIM Operations over HTTP specification defines a mapping of CIM operations onto HTTP that allows implementations of CIM to interoperate in an open, standardized manner and completes the technologies that support WBEM.

Specifications include CIM Schema, CIM Operations over HTTP, Representation of CIM in XML, and XML Document Type Definition.

WBEM is a set of technologies to unify enterprise management. WBEM is comprised of the CIM model that we described in the previous section, the xmlCIM encoding of CIM elements in XML, and CIM over HTTP specification that enables interoperation across CIM systems. WBEM operations could be simple or batched. The WBEM operationTypes are include data, metadata, Queries and methods. WBEM operations include: GetClass, EnumerateClasses, EnumerateClassNames, GetInstance, EnumerateInstanceNames, GetProperty, SetProperty, CreateInstance, ModifyInstance, DeletInstance, CreateClass, ModifyClass, DeleteClass, Associators, AssoicatorNames, References, ReferenceNames, ExecQuery, getQualifier, SetQualifier, DeleteQualifier, EnumerateQualifiers.

xmlCIM tries to create an XML based grammar expressed in a DTD so as to represent CIM information (classes, instances, methods), and CIM messages for use by CIM protocols. CIM-XML uses xml-CIM as the payload over HTTP transport.

DMTF has developed a core set of standards that comprise WBEM adding an encoding specification (the xmlCIM Encoding Specification) and a transport mechanism (CIM Operations over HTTP) to the Common Information Model. The xmlCIM Encoding Specification defines XML elements (described in a Document Type Definition, DTD or XML Schema) representing CIM classes and instances. The CIM Operations over HTTP Specification defines how the CIM classes and instances are created, deleted, enumerated, modified and queried. Also, the specification defines a notification and alerting mechanism for CIM events.

4.2.5 TMF – TOM/eTOM

The Tele-Management Forum (TMF) introduced the Telecom Operations Map (TOM) which focuses on the end-to-end automation of communications operations services [eTOM]. The core of TOM is a service framework that postulates a set of business processes that are typically necessary for service providers to plan, deploy and operate their services. These processes are organized using the TMN layering concepts and furthermore detailed to a finer granularity. TOM offers valuable concepts and addresses aspects of service management using business processes.

TOM is a reference model for telecommunications networks. The TOM reference model defines the description of core systems and processes involved in the production operation of telecommunications networks. TOM and eTOM are standards by the Telecommunications Forum (TMF). eTOM is the updated framework that incorporates the complexity of the internet economy and its opportunities. eTOM is proposed as a comprehensive enterprise framework for service providers. eTOM is an ongoing TMF initiative to deliver a business process model or framework for use by service providers and others within the telecommunications industry. eTOM describes all the enterprise processes required by a service provider and analyzes them to different levels of detail according to their significance and priority for the business. For such companies, it serves as the blueprint for process direction and provides a neutral reference point for internal process reengineering needs, partnerships, alliances, and general working agreements with other providers. For suppliers, eTOM outlines potential boundaries of software components to align with the customers' needs and highlights the required functions, inputs, and outputs that must be supported by products.

The eTOM Business Process Framework (BPF) represents the whole of a service provider's enterprise environment. The Business Process Framework starts at the Enterprise level and defines business processes in a series of groupings. The Framework is defined as generically as possible so that it is organization, technology and service independent and support of the global community. At the overall conceptual level, eTOM includes following three major process areas:

- Strategy, Infrastructure & Product covering planning and lifecycle management,
- Operations covering the core of operational management, and
- Enterprise Management covering corporate or business support management.

4.2.6 Parlay / OSA

The Parlay Group [Parlay] is an open multi-vendor consortium formed to develop open technology-independent application programming interfaces enabling Internet and eBusiness companies, independent software vendors (ISVs), software developers, network device vendors and providers, and application service providers to develop applications and technology solutions that operate across multiple networking platform environments.

The Parlay group was formed in 1998. The Parlay group was initiated by a group of operators, IT vendors, network equipment providers, and application developers to support and enable interoperable mobile applications. Parlay defined a number of API specifications. These specifications are standardized with participation of the Parlay Joint Working Group (JWG), which includes the Third Generation Partnership Programs 1 and 2 (3GPP), and the European Telecommunications Standards Institute (ETSI) Services Protocols and Advanced Networks.

Parlay integrates telecom network capabilities with IT applications via secure, measured and billable APIs releasing developers from rewriting those codes again and creating a homogeneous, standardized environment for developing, delivering, measuring, and billing mobile web services.

4.3 Model Creation Process

A top-down approach for model creation follows the object oriented development style (comparable to UML). It is shown in Figure 29.

An inherent characteristic of every service is that it involves two major players: one offering and one requesting and using the service. There is a provider side and a customer side. Both interact to accomplish a service. By examining these interactions, conclusions can be drawn about the service functionality without the need to take the service implementation into account. Therefore, it is important to identify these interactions, which also represent the service life cycle.

As it is impossible to identify every single interaction, an abstraction of these interactions is needed. Classes can be used to group interactions. The life cycle phases lead to a first grouping of interactions. The combination with a functional classification leads to a classification matrix.

As interactions take place between a pair of roles representing e.g., organizational units on both sides, roles are assigned to interaction classes. By examining the identified interaction classes and roles, interfaces can be specified as well as components participating in service provisioning leading to the final step of analysis, developing a service model containing objects and relations on the basis of former identified interactions, interfaces and roles. Finally, the recursive application of this model identification process allows representing entire provider hierarchies [Hegering 2001].

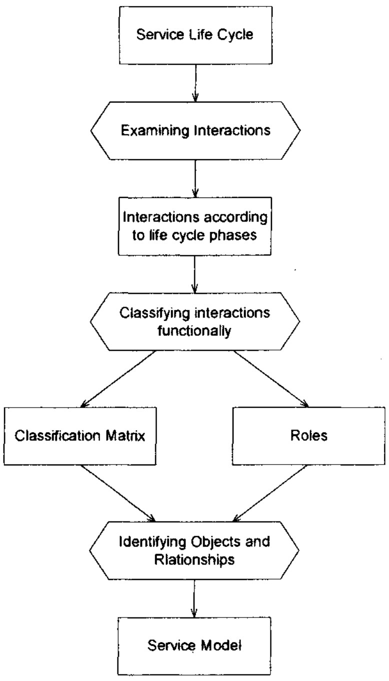

Service Life Cycle
Examining Interactions
Interactions according to life cycle phases
Classifying interactions functionally
Classification Matrix
Roles
Identifying Objects and Rlationships
Service Model

Figure 29: Model creation process.

4.4 A Generic Service Model

It is difficult to define the term service in a universal way, not restricting it to a small set of scenarios. The approach here is based on the top-down methodology presented in the previous section.

As starting point, service is defined as a set of interactions. But this is, of course, not sufficient to determine all aspects of a service. To narrow down the definition, the term service is more precisely defined through the existence of the roles: user, customer as well as provider and through their associations to the service.

These roles and associations are defined in the service model depicted in Figure 30. The understanding of a service must be the same for customer and provider side. The concept of "service orientation" advocates the implementation-independent description of the service from the perspective of the customer side. Side independent aspects can be found in the figure between the two domains symbolizing customer side and provider side. This information is an integral part of service agreements.

The following steps can be used to refine the generic service model.

- A service is composed of one or more (composite) service components. A service component is an independently distributed and addressable piece of functionality and data.
- A service is driven by different classes of actors (using services) with specific use cases (workload profiles, use patterns), which are composed of tasks. Each class of actors may require different QoS criteria.
- Tasks are service-defined pieces of "work" that are of interest to the actors of a service, and to the management system. Tasks. In general, may have three scopes: local, network and remote. The remote scopes of a task are based upon component interactions, and are sub-tasks of the original task.
- Each service component offers different service functionality through its published interfaces.

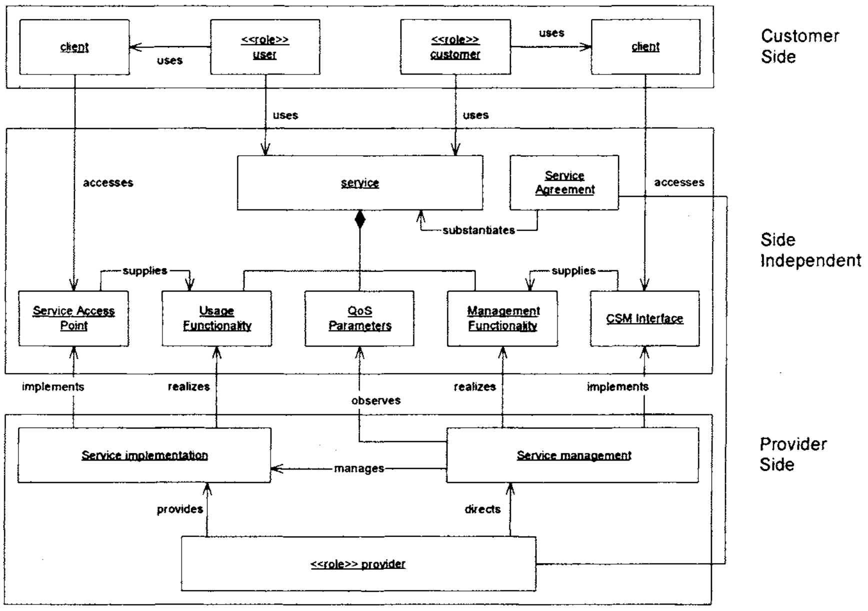


Figure 30: Example of a generic service model.

- Service interactions are based on the use of services. Interactions may occur between different semantic levels of services: methods, interfaces, objects, object clusters, or processes.
- Interactions occur across a logical network providing connectivity for one level of semantic interaction. Each logical network may have specified minimum needs for QoS, which can be represented by configuration rules. The logical network, when mapped to the physical network represents actual node-to-node measurements from network measurement systems.

Side Independent Aspects

According to the main interaction classes, the service consists of usage and management functionality. Both types of functionality must satisfy a set of QoS parameters. These parameters define the minimum required service quality in order to be useful for the customer side. The QoS parameters define qualitative as well as quantitative values.

The usage functionality covers the interactions needed by the user. These interactions represent the actual purpose of the service and interactions beyond the service's purpose that are needed to customize the service according to user's needs and to control the provider's service provisioning. These interactions are provided by the management functionality.

The service agreement substantiates the service by describing the usage and management functionality as well as the QoS parameters.

The information presented so far describes the service used by the customer side and provided by the provider side. To actually be usable, a service interface between these two sides must exist. Service primitives, protocols and physical connectors are represented by the service interfaces.

Interface definitions must also be included in the service agreement (Page 228) to enable the customer side to access the service functionality. The interface is the point where the responsibility of the provider ends. It can be argued whether interfaces are or are not part of the service. When the interface is not considered part of the service, this means that changing or adding interfaces for service access does not result in a different service, though the service agreement has to be changed.

In the same way usage and management functionality have been separated, the interface is split up in a usage interface, often called the Service Access Point (SAP), and a management interface, called the Service

Management Interface (SMI interface) through which the corresponding management functionality is accessible.

Customer Side

On the customer side, in most cases client software is needed to access the service functionality. Such clients are in turn services and allow users and customers to access the functionality at the service access point and the service management interface.

Provider Side

The main task of the provider is to make the service available in the promised terms that are part of the service agreement. This includes all aspects of the service, namely the usage and management functionality to meet the QoS parameters and the interfaces enabling the usability and manageability of the service.

The provider has a service implementation, which realizes the usage functionality of the service and implements the service access point. The service implementation is considered the combination of the entirety of knowledge, staff, software and hardware needed to realize the service's usage functionality and the service access point.

The provider is also responsible for service management. This means, the service is operated in a way keeping it above the agreed quality level, but also optimizes the service operation according to the provider's goals like efficiency and cost.

4.5 Recursive Application of the Service Model

Service models must be able to represent typical service chains with one provider contracting services of another provider, and one service provider acting as a customer of another. In such a scenario, an actor may take up the the user/customer role and the provider role simultaneously. The modeled associations between customer and the provider roles can be reused as associations expressing the relation between provider and a sub-provider. A model of the provider domain results by expanding the provider domain with entities from the customer domain. This model contains the service implementation and service management dependencies.

The service management of recursively composed services will use functionality of the traditional network, system and application management also in a recursive manner. As a consequence, there has to be a management logic controlling management of chained services as well as of sub- service.

4.6 Models for Diagnosis Rules

The following example illustrates how diagnosis rules can be modeled. Each service task can be seen as subject to different QoS goals: either derived directly from a SLA, or derived from the QoS goals of a service interaction. If QoS goals are not specified, nominal values can be obtained from previous observations of satisfactory operation of service.

The ability to correlate performance metrics available from different subtasks is vital to a model of a service task. Based on knowledge of the order in which subtasks occur, the model can diagnose the task performance based on this metric. This correlation of task metrics is necessary for tracing and diagnosing QoS failures within services.

Service tasks have nominal performance characteristics associated with each workload to which they belong. These values are used for detecting performance degradations, by simply comparing them with the current performance of the task, or through a more complex assessment, that takes into account task dependencies. A separate model may be used to adapt these nominal values to changing environments [Matinka 1997].

```
RULE 1 [Task task] meetsClientRespTimeGoal IF
  [task] qosGoals [QoSGoal goal] &
  [goal] respTime [Real target] &
  [task] clientRespTimeAvg [Real respTime] &
  [target] greaterThanOrEquals [respTime] &
  [String currentTime] nowAndEvery [30000].
RULE 2[Task task] performanceBottleneck IF
   [task] violatesClientRespTimeGoal [Real r] &
   ([task] cpu |
    [task] ioBound |
    [task] channelHosed |
    [task] serviceBottleneck [Task service]).
RULE 3 [Task task] serviceBottleneck [Task service] IF
   [task] subTask [service] &
   ( ~[service] meetsClientRespTimeGoal |
     ~[service] meetsBaseLineGoal |
     ~[service] lowLatency).
```

Figure 31: Examples of performance rules for SLA observation.

Figure 31 illustrates performance rules for SLA observation, which are part of a model representing a service's behavior from the provider's point of view. As an example, Rule 1 is a rule that monitors a task's response time every five minutes. The '&' represents a conjunction (logical AND) and '|' disjunction (logical OR).

Once the aggregate response time for a particular task violates its QoS threshold, the diagnostic rules trigger a management event. These rules examine all available measurements for that task in order to identify the cause of the violation. In Rule 2, a task contains a performance bottleneck if its response time QoS goal is violated, or the local (CPU and I/O) network channel for remote services containing bottlenecks.

Other diagnostic rules may combine the service response times according to the order and manner of their invocations.

5. SUMMARY

The evolution of enterprise computing has had a dramatic impact on the role of system management, with system management and administration proving to be a difficult issue for most organizations. With the demand for IT organizations to provide more information faster and with greater reliability and accuracy, IT organizations are seeking for solutions that help them manage the integrated enterprise. Enterprise management systems enables administrators to perform resource, system, and network management, proactively manage systems and processes, automate tasks, and add, remove, or modify functionality based on business needs.

The technologies that have been developed in the context of enterprise management are quite useful in the domain of web services. Especially CIM, ITSM, eTOM, PARLAY have capabilities that can be directly and indirectly leveraged in undertaking model-based management of web services.

PART II

PERSPECTIVES ON WEB SERVICES MANAGEMENT

Chapter 5

MANAGING WEB SERVICES FROM AN E-BUSINESS PERSPECTIVE

1. INTRODUCTION

During the peak of the Internet bubble around 2000 and 2001, the "new economy" was promising unbound growth by transitioning entire industries and all spheres of life into an "e-world" of totally connected people, organizations, and businesses. E-business was understood as business mediated through electronic media, specifically the Internet and the Web. Penetration of Internet and Web access had gathered the critical mass which enabled and ignited the e-Revolution, which is now seen rather as an evolution. This evolution is an ongoing transformation process during which interactions between people and organizations are transformed into ubiquitous information exchange through services created in the Web.

Web services initially emerged as services available on the Web, useful functions that were accessible by users through the Web infrastructure. This kind of Web services were characterized by human-to-service interaction mediated though Web browsers at the client side and Web servers and back-end applications at the service side. Extending the model towards service-to-service interactions required formalization of exchanged data. HTML was designed for browser interactions. HTML was not designed and thus is not suitable for service-to-service interactions. XML brought the required formalization and enabled a second wave of Web services, now between services that existed in the (public or intra-) Web. XML-based Web services have been characterized by the use of HTTP protocol as transport (the same as used in the Web), SOAP (SOAP) as message exchange format, and XML (XML) as universal format for representing data exchanged through messages.

Universal connectivity provided by the Internet and the Web based on standard protocols (IP and HTTP) allowed using Web services technology for business purposes ranging from supply-chain integration to advertising and retail purposes.

From their emergence, for human-to-service interactions and later for service-to-service interactions, Web services had a close affinity to business aspects, business that has been conducted over the Internet and the Web. This affinity to business made Web services special compared to other applications or software. Affinity to business also gave reason why management of Web services [Machiraju 2002] does not only include the technical aspects of running Web services applications and managing those applications. Web services management also closely ties into business aspects. The expectation has been (and is) that Web services management technology can provide the bridge from business perspectives into technical aspects of Web services. Business perspectives include the variety of business models and the ability to evolve and adapt them quickly to changing needs [Ray]. Business perspectives also include measurement of business metrics and validation of the actual business conducted through Web services based on technical measurements gathered and processed by Web services management systems. Business perspectives also include the transition from IT-driven operational and management models to business-driven operational and management models. Realistically, the promise of Web services management to smoothly link the business domain with the technical domain has not been fulfilled. However, progress has been made, progress that significantly expands over traditional applications and management systems.

This chapter provides an overview of the current state of the art in managing Web services, specifically from the business perspective.

This chapter addresses following aspects:

- The Method of Balanced Scorecards,
- Web services and Business Problems,
- The Customer Life Cycle,
- Web services Business Metrics.

2. THE METHOD OF BALANCED SCORECARDS

The method of balanced scorecards was introduced as an approach to strategic management in the early 1990's by Robert Kaplan (Harvard Business School) and David Norton (Balanced Scorecard Collaborative), [Kaplan 1996]. Recognizing some of the weaknesses of previous management approaches, the balanced scorecard approach provided a clearer prescription of what companies should measure in order to balance their financial perspective.

The balanced scorecard is a management system (not only a measurement system) that enables organizations to clarify their strategy and translate them into action. The balanced scorecard provides feedback around the internal business processes and external outcomes in order to continuously improve performance and results.

Kaplan and Norton describe the innovation of the balanced scorecard as follows: "The balanced scorecard retains traditional financial measures. But financial measures tell the story of past events, an adequate story for industrial age companies for which investments in long-term capabilities and customer relationships were not critical for success. These financial measures are inadequate, however, for guiding and evaluating the journey that information age companies must make to create future value through investment in customers, suppliers, employees, processes, technology, and innovation."

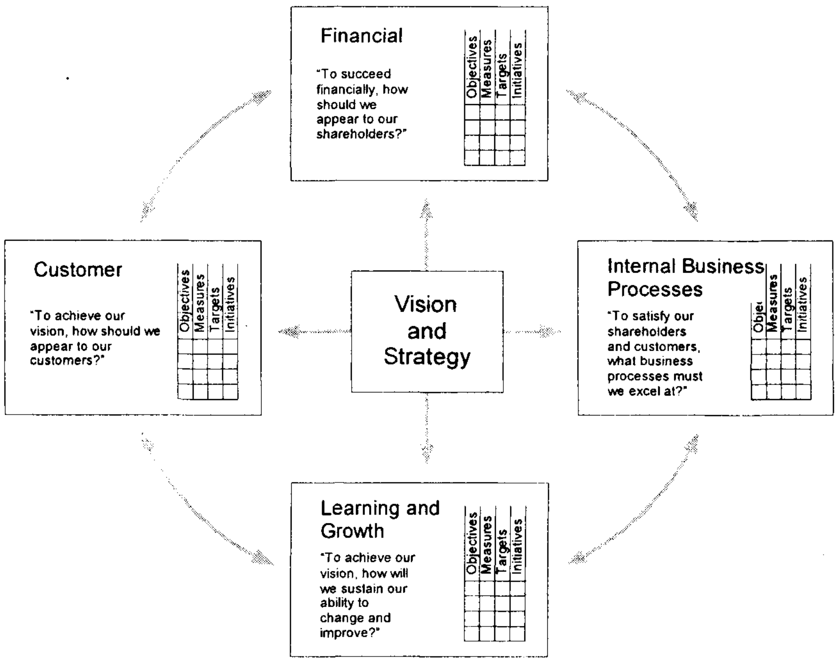
The balanced scorecard suggests that an organization or a system is viewed from four perspectives. The balanced scorecard also suggests the metrics, to collect data and analyze it relatively to each of the perspectives. The four perspectives are:

- The Learning and Growth Perspective,
- The Business Process Perspective,
- The Customer Perspective, and
- The Financial Perspective.

The Learning and Growth Perspective

The Learning and Growth Perspective includes employee training and corporate cultural attitudes related to both individual and corporate improvement. In an organization, people are the repository of knowledge. In the climate of constant technological change, it is becoming necessary to stay in a continuous learning mode. Government agencies often find themselves unable to hire new technical staff, and at the same time they are showing a decline in training of existing employees. This is a leading indicator of brain drain. Metrics can be put into place to guide managers to fund training. Learning and growth constitute the essential foundation for success of any organization.

Kaplan and Norton emphasize that learning is more than training. Learning also includes mentors and tutors within the organization, as well as communication among employees that allows them to get help on a problem when it is needed. It also includes technological tools. One of these tools is the Web and the organization's Intranet.



Financial
"To succeed financially, how should we appear to our shareholders?"
Objectives
Measures
Targets
Initiatives
Customer
"To achieve our vision, how should we appear to our customers?"
Vision and Strategy
Internal Business Processes
"To satisfy our shareholders and customers, what business processes must we excel at?"
Learning and Growth
"To achieve our vision, how will we sustain our ability to change and improve?"

Figure 32: Four perspectives of the method of the balanced scorecard [Kaplan 1996].

The Business Process Perspective

This perspective refers to internal business processes. Metrics based on this perspective allow managers to know how well business is operating, and whether products and services conform to customer requirements (called the "mission"). These metrics have to be carefully designed by those who know these processes best. In addition to the strategic management process, two kinds of business processes can be identified:

a) mission-oriented processes, and

b) support processes.

The Customer Perspective

Recent management philosophy has shown an increasing importance of customer focus and customer satisfaction in any business. If customers are not satisfied, they will eventually find other suppliers that will meet their needs. Poor performance from this perspective is thus a leading indicator of future decline, even though the current financial situation may look good.

In developing metrics for satisfaction, customers should be analyzed in terms of kinds of customers and the kinds of processes for which a company is providing products or services.

The Financial Perspective

Kaplan and Norton include traditional financial data in the financial perspective. With the implementation of a corporate database about the financial situation, it is aimed to automate gathering and processing this information in ways that easily allow creating current views on the financial situation.

2.1 Balanced Scorecard and Management

The balanced scorecard methodology builds on some concepts of previous management ideas such as Total Quality Management, including customer-defined quality, continuous improvement, employee empowerment, and primarily measurement-based management and feedback.

2.1.1 Double-Loop Feedback

In traditional industrial activity, quality control and zero defects have been important. In order to shield customers from poor quality products, efforts were made on inspection and testing at the end of the production line. The problem with this approach is that the true causes of defects could never be identified, and there would always be inefficiencies due to the rejection of defects. Variation can be created at every step in a production process, and the causes of variation need to be identified and fixed. This opens the way to reducing defects and improving overall product quality. To establish such a process, all business processes should be part of a system with feedback loops. The feedback data should be examined by managers in order to determine the causes of variation, the processes with significant problems, and finally fixing that subset of processes.

The balanced scorecard incorporates not only feedback around internal business process outputs, but also adds a feedback loop around the outcomes of business strategies. This creates a "double-loop feedback" process in the balanced scorecard.

2.1.2 Outcome Metrics

"It is hard to improve what you can't measure." Metrics must be developed based on the priorities of the strategic plan. The strategic plan provides the business drivers and criteria for metrics to be measured. Processes are then designed to collect the relevant information based on these metrics and consolidate it into a numerical form for further processing and analysis. Decision makers examine the outcomes of various measurement processes and track the results in order to guide the company.

The value of metrics is in their ability to provide a basis for defining:

- Strategic feedback to show the present status of an organization from many of the four perspectives for decision makers.
- Diagnostic feedback into various processes to guide improvements on a continuous basis.
- Trends in performance over time as the metrics that is tracked.
- Feedback around the measurement methods themselves, and which metrics should be tracked.
- Quantitative inputs to forecasting and models for decision support.

2.1.3 Management by Fact

The goal of making measurements is to permit managers to see a company more clearly and hence to make better long-term decisions. The Baldrige Criteria [Baldrige 1997] reiterate this concept of fact-based management: Modern businesses depend upon measurement and analysis of performance. Measurements must derive from the company's strategy and provide critical data and information about key processes, outputs and results. Data and information needed for performance measurement and improvement are of many types, including: customer, product and service performance, operations, market, competitive comparisons, supplier, employee-related, and cost and financial. Analysis entails using data to determine trends, projections, and cause and effect that might not be evident without analysis. Data and analysis support a variety of company purposes, such as planning, reviewing company performance, improving operations, and comparing company performance with competitors or with best practices benchmarks.

A major consideration in performance improvement involves the creation and use of performance measures or indicators. Performance measures or indicators are measurable characteristics of products, services, processes, and operations the company uses to track and improve performance. The measures or indicators should be selected to best represent the factors that lead to improved customer, operational, and financial performance. A comprehensive set of measures or indicators tied to customer and/or company performance requirements represents a clear basis for aligning all activities with the company's goals. Through the analysis of data from the tracking processes, the measures or indicators themselves may be evaluated and changed to better support such goals.

3. WEB SERVICES AND BUSINESS PROBLEMS

Building and maintaining an effective presence in the Web is not only a major corporate investment, but also a considerable technical challenge. From startups to mid-sized players to industry leaders, companies are looking to build and leverage their online brand in the Web, to gain market share, to complement and accelerate their existing business models, to reinvent their business and revitalize their value proposition.

These stages are shown in Figure 1.

Stage	1	2	3	4
site type	static	dynamic	e-commerce	personalized
Goal	presence	interactive	revenue	customer relationship
complexity	low	medium	high	very high
Impact	small	moderate	significant	transformational

Figure 33: Stages of the Web services evolution.

The Web is both a practical platform for doing business, but also a rich source of information about customer behavior. Its low cost of customer contact makes it ideal for new methods of marketing. The earliest Web sites were standalone attempts to publish useful but relatively static content. Second generation Web sites became more dynamic, with rudimentary customization. The third stage provided e-commerce, and the fourth stage introduced an era of increased online personalization and more effective relationship management.

From an architectural perspective, an early Web site was a simple Web (HTTP) server, communicating with a file system populated with directories of flat HTML documents. It was a single-platform with a single-server in a rather static world. Modern Web sites are groups of interconnected sites consisting of multiple, geographically distributed software and hardware servers, encrypted and non-encrypted areas, and dynamically generated content. There are advertising services, content services, personalization services, registration services, and e-commerce services, working alone or together shaping the user experience. This heterogeneity makes it incredibly challenging to build a complete and coherent view of customer behavior and site activity.

To manage transactions and handle personalization, modern sites need integration with other corporate operational and informational systems. Of course, integration of different applications and data is not new. In the past, integration has focused on direct applications such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), and Data Warehousing (DW) in order to unify key areas of an enterprise. The Web is a new integration platform for applications and data. The Web is able to provide an actionable view of customers' online behavior. This unified online view supports personalization and online interactions, as well as segmentation and other aspects of analyses for decision support and targeted marketing. And while back office systems are hidden from customers through the Web services layer, applications and business functions behind applications are exposing an organization to customers. Customers, on the other side, are just "a click away from competition". The Web thus brings incredible transparency of markets, goods, services and prices at a global scale with almost no latency and at low cost.

This results in a new technical challenge for information technology. The challenge is to build a corporate infrastructure that ties together CRM, e-platforms, and site effectiveness through the right Web analysis. The issues are to act and react quickly, to unify the CRM strategy across channels in order to optimize the entire online business. However, unlike products, pricing, and positioning which competitors can learn about and counter the results of Web analysis can be kept within the organization and be used for competitive advantage. The focus is on information and how to leverage it as a strategic enterprise asset. Part of the enterprises value proposition has always depended on the ability to integrate information assets, and the growth of Web and emergence of Web services are accelerating that.

According to Forrester Research [Forrester 2000], Global 2500 companies spent an average of $41 million on their e-commerce efforts in the year 2000. Corporations continuing investments in improving their online capabilities, specifically also exploiting gathered information using data warehouse applications. They need to understand whether their investments are paying off. The question of Return on Investment (ROI) in IT infrastructure is pressingly important. Corporations had expanded IT infrastructure to accommodate demand from Web services in the past. Now it is the time making use if the gathered information in more clever ways supporting business strategy and business.

Web services management must provide the methodology and the tools for enabling IT customers utilizing information assets understanding business functions conducted and mediated though Web services. To approach these questions, Web services business metrics are becoming the central focus of a Web services infrastructure.

3.1 Key Business Areas Addressed by Web Services

Currently users are attracted to a Web site through advertisements, links, affiliates, searches, or perhaps prior knowledge of a brand. A Web site may further engage user interest through responsive marketing by providing details, conveying a value proposition, making offers, and marketing campaigns.

The goal is to get users to convert which may mean a purchase, a download, an agreement, or a sign up. For the future, the expectation is that they will stay loyal, return for support or to see what is new, bringing cross-sell and up-sell opportunities as well as referrals.

This sequence is called the customer life cycle of a Web site shown in Figure 34. Section 4 provides further refinement of the customer life cycle.

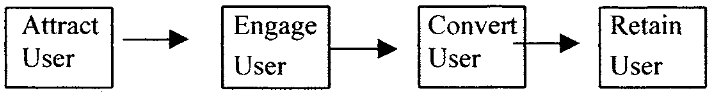


Figure 34: First view on the customer life cycle at a Web site.

The first four business problems correspond to the progression through the customer life cycle. Two additional business problems are important across the customer life cycle: content effectiveness and the ROI on channel relationships.

While every business is unique, certain questions typically arise in each of these areas. Measuring the right Web services business metrics to answer questions, analyzing the results, taking action in response, and then assessing the impact is the path to solving these key business problems through Web services management.

Figure 35 summarizes the six key business problems addressed by Web services management metrics and supporting systems.

1. Users	Understand anonymous user behavior.
2. Marketing	Maximize marketing and advertising ROI.
3. Commerce	Evaluate multi-channel commerce effectiveness.
4. Loyalty	Optimize customer loyalty and life time value.
5. Content	Analyze content effectiveness at complex, dynamic sites.
6. Channels	Assess affiliates and partner networks.

Figure 35: Key business problems addressed by Web services metrics and Web services management.

3.2 Business-to-Consumer (B2C) Domain

Basic purchasing facts about which products have been sold, when, and to whom, provide only a very high-level view at the surface. On the Web, management systems can see below the surface. This is where behavioral ties can be found, user trajectories and paths, interests and focal points, progress and decision processes, interruptions and bailouts, all these within the context of individual users or user segments. Web services business metrics allow looking below the surface, to draw the right connections and guide actions in analysis and response.

Analysis leads to action for each of the questions above. For example, in e-commerce an important question is to determine which site paths are optimal for making a purchase. Knowing the answer can help to guide users onto those paths. It can be made easier to find the right path from a pivotal page on a Web site, or introduce shortcuts that get users faster to their goals. It is important to reduce the risk of diversion once a user is on the right path. It may also turn out that the optimal site path varies by user segment, and that a self-selecting choice can lead users to the most appropriate path for their segment.

There are five major categories of actions Web services metric analysis can lead to:

- Reorganization of a Web site to improve navigability, ease of use, likelihood of purchase, and stickiness.
- Segmentation of the customer base for selective, appropriate, timely communications over multiple channels.

- Using customer profiles, history, and offline context by personalizing content, advertisements, and promotional offers.
- Prediction of future customer behavior (such as likely to convert, next product to buy, likely cart abandonment, likely area of interest) to make the right offer at the right time.
- Alignment with referring services/sites where advertisements are most effective bringing profitable customers in reducing costs and increasing profits.

Understanding users, their behavior and goals, and how to help them achieving their goals is at the core of the analysis-to-action approach. Web services business metrics are a major part of developing and understanding effective scenarios.

3.3 Business-to-Business (B2B) Domain

While the six business problems above apply in principle to both business-to-consumer (B2C) and business-to-business (B2B) operations, especially to web services as they emerge as programmatic back-ends to these web sites, there are additional subtleties in the B2B area.

A B2B marketplace serves both buyers and suppliers. Each party needs to understand the effectiveness of the business model. Buyers want personalized and streamlined transactions, at a competitive price. Suppliers need a window into customer behavior and market dynamics. They want profitable customers who are interested in their goods and services.

In providing suppliers with analytic information, the service provider must assess the performance of the marketplace, the appeal of various products, and the profiles of specific customers and customer segments. Often, it is important to profile user behavior anonymously, prior to registration, and to correlate this information with post-registration actions.

New business models have led to site innovations as well. Business exchanges, auctions, so-called vertical portals, bargain-finding engines, product configurators, and intentions-value networks (e.g., for relocation to a new city) are changing the face of business-to-business interactions. Suppliers are particularly interested in evaluating the business effectiveness of different business models. And through the visibility they provide into corporate inventories, the supply chain, tracking, and billing information, monitoring and metering information is becoming a very important aspect of B2B Web services.

Unlike end-consumer marketing, there is generally more information available about corporations and their representatives. While the concept of a business customer is more complex, given corporate hierarchies and purchasing roles within companies, user behavior is less random, with more personalization possible based on prior behavior. Some corporations go as far as setting up personalized Web pages for each business customer.

4. CUSTOMER LIFE CYCLE

The overall business economy has shifted from process, or product-centric, to customer-centric and hence customer life cycle-centric. The problem in the Web is that customers are invisible without clear definitions of a user or a visit. People come to a Web site, leave footprints, and move on. But those footprints are merely an indication that they were there. Footprints contain the information, but tell little about people who visited a Web site unless visitors are categorized in a spectrum from occasional visitors to loyal customers. It is hard to measure effectiveness of a promotion when its effect cannot be measured [Reiner 2001].

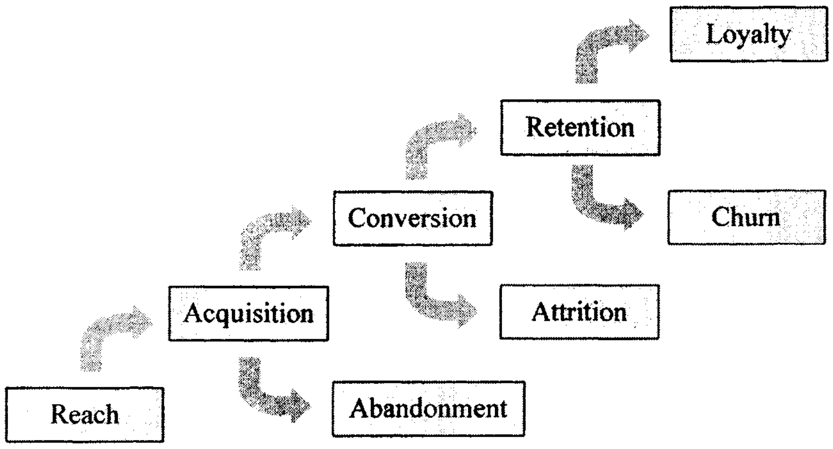


Figure 36: The customer life cycle from initial reach to final loyalty.

The Customer Life Cycle starts with reaching a target audience and progresses towards an established loyal customer base. Of course, along the way, many individual customer life cycles are cut by abandonment and attrition.

The customer life cycle describes the points in the continuum where a Web site:

- Claims someone's attention.
- Brings them into the sphere of influence.
- Turns them into a registered and/or paying customer.
- Keeps them as a customer.
- Turns them into a Web site advocate.

The lines between these various stages depend on the deployed business model. It makes a difference whether a Web site is primarily selling products or services or a typical sale will take 5 minutes or 5 months. It is different whether someone is trying to convince users to purchase a mortgage or register for a personal home page. In a purely retail B-to-C environment, the stages are typically much clearer than in a B-to-B environment.

5. WEB SERVICES BUSINESS METRICS

Traditional metrics deal with corporate value (price to earnings ratio, market capitalization, fixed assets), corporate process management (cash flow, net profits, customer turnover), and financial expectations (market share, book-to-bill ratio, revenue per employee, etc.). Web service based business interactions are shifting focus away from production, distribution, and share of market, and closer towards customer needs, loyalty, and share of budget. Not surprisingly, new Web services metrics are necessary. Web services-enabled companies can reach millions of customers or members; dramatically lower the cost of business, and connecting buyers and sellers more efficiently than ever before. Every Web service or Web site can have an affiliate program, broadening its reach. Banner advertisements, e-mail promotions, and Web site changes are able to affect user behavior in real-time. And users seek a positive experience with personalized treatment.

The new Web services business metrics translate the richly detailed portrait of customer behavior into measurable elements of e-business metered at the source of the Web service. These measures provide tangible, actionable business information that impacts both the online and offline sides of business. With Web services business metrics, the opportunity opens to approach the Web from an objective, systematic perspective. A Web service provider can become a better customer confidante, a better partner in commerce, and a more trusted member of the customer value chain.

From an architectural perspective, there are two interlinked halves. First, raw measurements must be refined into Web services business metrics, and then these results must be processed to be easily understood and applied.

Web services business metrics begins with the hierarchy of Web site activity, moving up from hits to page views to visits to users, and from users to customers to loyal customers. Moving up in the pyramid, as shown in the in Figure 37, means moving up from site design to customer behavior. There is radically more business value at the top of the pyramid after processing the large data volumes at its base.

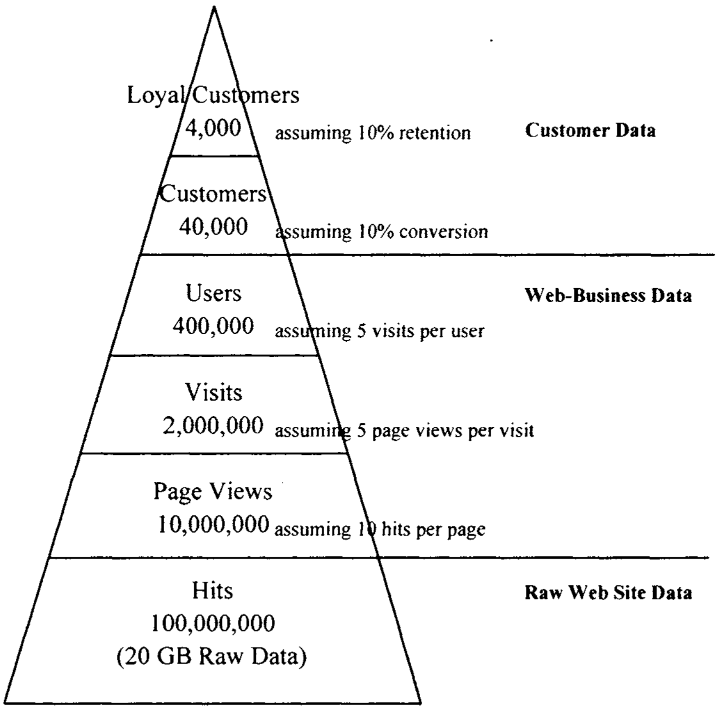


Figure 37: Transitioning from Web site hits to business-relevant customer data.

Whether the end goal for Web site users is pure commerce, exchange of requirements and quotations, registration for a service, joining a community, or just acquiring timely and relevant knowledge, a major objective of Web services business metrics is to track users through their life cycle.

The essential Web services business metrics tracks transitions through an ideal sequence of reach, acquisition, conversion, retention, and loyalty.

But between clicking on a banner advertisement and making a purchase or other action, there are many points where a customer may fall out of the life cycle. Web services business metrics that measure interruptions to the life cycle can help to get a handle on abandonment, attrition, and churn rates, and their relationship to the warning signs of customer behavior.

Another set of Web services business metrics can help segment users by answering the vital question of determining their best customers. Recency, frequency, monetary value, and visit duration are relevant. Yield, or the measurement of first-level goals such as the percentage of users who return, provides an indication of the monetary value of each user. When costs are considered along with customer value, a new set of Web services business metrics appears, such as acquisition cost, cost per conversion, and net yield.

There are many fundamentally new Web services business metrics that provide actionable insights into customer behavior. Stickiness is a measure of site content effectiveness and the ability to hold users attention. Tracking stickiness for specific content areas allows site production staff to quickly identify under-performing pages or sections, while sticky customer segments represent core constituencies and opportunities for deepening relationships. Slipperiness is equivalent to low stickiness for a particular area of a Web site. Focus may be wide or narrow based on how many pages in a site section are touched by an average visit.

The rate of customer movement along the life cycle is known as velocity. The junctures where a customer is exceptionally susceptible to an offer are critical moments. The revolving door ratio is the ratio of visit entries to visit exits for a particular area of the site. Information about so-called black holes, or areas of a Web site where many people are drawn in but few emerge, and so-called worm holes, or consecutive page views between pages that are not directly linked (e.g., how did a user go from the top level product page to the third level support area?). Also the average lifetime of unsolicited pop-up windows, provides essential information.

The personalization index of a Web site is a measure of how well collected customer profile information is used. Customer life cycle value, traditionally calculated from expected purchases, referrals, and margins, is given a new perspective on the Web through the potential cost savings this medium allows. Loyalty value is a comprehensive Web services business metric involving not just purchases, but also including information about visits, referrals, and participation intensity. The freshness factor measures the refresh rate of content compared to visit frequency for areas of content. This tells how often users are seeing fresh content when they visit.

The combination of Web services business metrics with other Web services business metrics, including traditional measures, is a powerful technique for revealing and utilizing information gained from the interactions mediated though a Web site (or a Web service) as a prerequisite for turning this information into a competitive advantage.

The following list shows some further examples of typical business problems paired with Web services business metrics:

- identify the most valuable customers early on using the customer life time value grouped by the type of initial site visit;
- better differentiate the appeal of content across the user spectrum by looking at return rates on Web content grouped by user segments;
- understand how visit behavior is changing over time since the first visit by evaluating visit recency compared to user tenure;
- characterize the impact of increased loyalty on visit and purchase behavior by looking at the depth of site penetration and single visit conversion rates compared for first time users and returning users;
- measure how quickly users are getting where they want to go by assessing the average number of page views to reach preferred content grouped by user segments;
- locate the greatest opportunities for increasing conversion derived from page exit rates compared along the optimal purchase path; and
- distinguish affiliates who refer many users from ones who may refer less users but bring more valuable ones by referring to acquisition cost compared to conversion cost.

5.1 Web Services Business Metrics Classification

The following table summarizes Web services business metrics structured in four major categories [Reiner 2001]:

- Customer Life Cycle Metrics (refer also to section 4),
- Customer Metrics,
- Customer ROI (Return-On-Investment) Metrics, and
- Web Site Behavior Metrics.

Each category names several measured, which can be determined and used of analytical purposes [Cutler 2001].

Customer Life Cycle Metrics	**Customer Metrics**	**Customer ROI Metrics**	**Web Site Behavior Metrics**
Rates for: - reach, - acquisition, - conversion, - retention, - loyalty; *Rates for:* - abandonment, - attrition, - churn.	*Measures for:* - recency, - frequency, - loyalty value: in the: - retail, - high-priced domain, - business-to-business domain, - monetary value, - duration.	*Cost per customer:* - acquisition cost, - cost per conversion, - net yield, - connect rate, - growth rate, - customer ROI and customer net yield.	*Measures for:* - stickiness, - slipperiness, - focus, - velocity, - seducible moments, - optimal site path, - personalization index, - freshness factor.

Figure 38: Classification of Web services business metrics.

5.1.1 Customer Life Cycle Metrics

The customer life cycle has been introduced in section 4. Measures for reach, acquisition, conversion, retention, and loyalty as well as for abandonment, attrition, and churn can be obtained by calculating respective rates comparing to the total number of customers:

$$\text{Rate}(x) = \frac{x}{\text{total number of customers}} \qquad x = \{\text{ reach, acquisition, conversion, retention, loyalty, abandonment, attrition, churn }\}$$

Figure 39: Calculating rates for customer life cycle metrics.

5.1.2 Customer Metrics

The RFM method (Recency, Frequency, and Monetary value) is widely used to describe metrics for best customers. Analysis of these metrics helps answering one of the most fundamental questions in marketing: Who are the best customers? Using past transactions, each customer is viewed simultaneously in three different dimensions:

- *Recency:* Has the customer recently made a purchase or visited the site?
- *Frequency:* How often has the customer placed orders or visited a site historically?
- *Monetary value:* What is the customer's total spending and what is his or her profitability?

Each dimension provides an insight into a customer's purchasing behavior:

- *Recency:* Statistical analysis has shown that customers who have made a purchase recently are more likely to purchase again in the near future.
- *Frequency:* Frequent purchasers are likely to repeat purchasing into the future.
- *Monetary Value:* Customers with high spending in the past are likely to spend again in the near future. This dimension is different from frequency by that it identifies customers who place infrequent but high value orders and, therefore, could be highly profitable.

Dividing customers into a number of segments using clustering methods based on recency, frequency, and monetary value helps to identify and profile customer segments that are not intuitively obvious or visible from other forms of reports and represent significant opportunities.

Recency

Recency is a core measure that describes how long it has been since a Web site recorded an event from a customer (e.g., site visit, product purchase, etc.). Recency is generally considered to be the strongest indicator of future behavior. According to RFM, the likelihood that users purchase tomorrow can be calculated from past experience. Timescales are different depending on products. A loyal luggage buyer may buy a suitcase once every three years. Milk, bread and egg buyers tend to shop weekly.

When browser-based cookies appeared, they were primarily used to welcome people back and let them know how that, for instance, the site had changed since their last visit. Knowing when somebody was last at a site is an important part of user profiling.

As recency diminishes with the time since the last activity or event increasing, the potential for future purchases decreases. Eventually, a pre-determined amount of time lapses and the user can be considered attritioned. In an attempt to reactivate customers, different offers might be targeted to different users. This can be sometimes found in practice. Amazon.com notes the consistency of customer visits and their purchases and sends out an e-mail with gift certificate when customers fall out of their normal purchasing patterns.

Frequency

Frequency is a measure that answers another important question:

"I want to know how many people are coming every day and how many."

Users may visit hourly, daily, weekly, monthly, or other periodic patterns.

Loyalty Value

It is useful to have a metric indicating the loyalty of customers. However, loyalty is very specific to market segments and companies and involves a large number of variables.

For instance, if a customer has an e-mail account at Yahoo and checks it every day, he or she would earn a high loyalty rating from Yahoo. However, if a customer also maintains an e-mail account at Hotmail and checks that one daily as well, then he or she are not really as loyal as Yahoo might think.

Measuring the loyalty of a Web site user means creating an index that can be used daily to see how changes made to the Web site affect customers. A study (Bain & Company) indicated that the average Amazon.com customer must remain loyal for approximately 2.5 years in order to become profitable. Each site should create a loyalty ranking system depending on its goals and its experience.

In the following, three scenarios are discussed of different meanings of loyalty in different domains:

- Loyalty in retail domain,
- Loyalty in high-priced domain, and
- Loyalty in business-to-business domain.

Loyalty in Retail Domain

A customer coming to a florist Web site four times a year may be considered a very loyal customer. A wedding anniversary, Valentine's Day, a spouse's birthday and Mother's Day are the major flower-giving occasions. A one-time-only customer can be encouraged to come back for another holiday as can the customer who only comes twice a year. But the customer who comes four times needs special enticement to increase his or her frequency.

A dollars-off coupon, a bouquet-of-the-month club, or a "buy ten, get one free" offer may appeal to them. Offers can be tested at each level of frequency in order to increase response rates.

Customer loyalty can be measured in purchases. How much does a customer buy? How often does he or she buy? Are they profitable customers?

The formula to calculate loyalty will include the following variables:

- Visit frequency: Scored based on number of visits per month
- Visit duration: Scored based on number of minutes per visit
- Visit depth: Scored based on number of page views per visit
- Purchases per visit
- Number of items purchased per visit
- Total revenue of purchases per visit
- Profitability of purchases per month

If additional marketing programs are implemented, the customer might be evaluated on factors such as:

- Number of referrals per month: Did the customer refer others?
- Value of referrals per month: Did those referrals buy? How much?
- Questionnaire propensity: How willing is the customer to answer survey questions?
- Contest participation: How willing is the customer to participate in contests?
- Reward points program: How willing is the customer to participate in affinity programs?

Loyalty in High-priced Domain

Expensive items create a different pattern of site visits. The occasional click-through at the start of the process initiates a steadily increasing number of visits up to the moment of purchase. If these traffic patterns are properly modeled, they can lead to a clear indication of when the sale may occur.

With this knowledge and some data mining techniques, a company can build predictive models to be more proactive by launching e-mail campaigns, dynamically alter the site or have a sales person call. Manufacturers can use that information to notify their distribution chains about potential sales. Service organizations can watch customer activity to determine the right moment sell. Training departments can track frequency to decide when to offer additional courses.

Loyalty can be measured on a short-term basis. It can also be measured on a longer-term time scale.

If a customer is buying a refrigerator, chances are good that she will not need another one for years. Those customers can be kept in the database for later years. Insurance companies keep information on newborns in their database for decades in order to offer additional auto insurance when they reach driving age.

Loyalty calculations are based on the same variables as in the retail domain, but apply with slight modifications:

- Visit frequency – scored based on visits per decision period and mapped to a decision-making curve. Buyers of one type of product will visit a certain number of times in the first period, a certain amount in the middle of the process, and signal that a buying decision is actively being made when they increase (or decrease) the number of visits in a defined time span.
- Visit duration – scored per session. This is another indication of how close to a decision a visitor may be.
- Visit depth – page views per visit are as important as frequency and duration.
- Site path – shows how well a visitor is following an optimal site path.
- Contact – tells how often a visitor shows initiative by sending e-mail, engage in a chat session or fill in a form on the site. It is important to analyze what kind of questions the visitor asks?

Loyalty in Business-to-Business Domain

Frequency becomes even more important when the relationship between parties is on a longer-term basis. When the Internet is used instead of Electronic Data Interchange (EDI), the pattern of visits and orders can be very informative, as the Web site traffic becomes the pulse of the business relationship. If a steady customer with a predictable pattern of visits changes browsing and buying pattern, this may indicate that human intervention can help increase the spending potential. Frequency information can provide insight into a customer's experience, expose a shift in customer staff, or signal the possibility of increased (or decreased) business.

In the business-to-business domain, the emphasis on selling is replaced with a focus on service. Taking orders and solving problems are most important. Loyalty appears in many shapes and sizes in the business-to-business domain. Loyalty metrics naturally differ greatly between different sites with different business models.

- Visit frequency: In the business-to-business domain, the consistency needs to be observed. Is the customer coming to the site in regular intervals and doing what is expected?
- Visit duration: Can changes in the amount of time it takes a customer to place an order be observed?
- Visit depth: Is the customer looking at products beyond the regular pattern?
- Visit tenure: Time elapsed since first visit.
- Purchase tenure: Time elapsed since first purchase.
- Purchase frequency: Number of purchases per quarter (or month or other fixed interval).
- Total life time spending: Total spending since first visit.
- Visit recency: Time elapsed since most recent visit.
- Purchase recency: Time elapsed since most recent purchase.
- Required clicks to first purchase: Minimum number of clicks required to complete the first purchase in a visit. The first purchase may require more clicks than repeat purchases.
- Required clicks to repeat purchase: Minimum number of clicks required to make a repeat purchase.
- Actual clicks to first purchase: Actual number of clicks until the first purchase was made.
- Actual clicks to purchase: Number of clicks until a purchase.

These variables provide input for higher metrics such as first purchase momentum:

$$\text{First purchase momentum} = \frac{\text{required clicks to first purchase}}{\text{actual clicks to first purchase}}$$

Figure 40: Calculating the first purchase momentum.

and repeat purchase momentum:

$$\text{Repeated purchase momentum} = \frac{\text{required clicks to repeat purchase}}{\text{actual clicks to repeat purchase}}$$

Figure 41: Calculating the repeated purchase momentum.

Monetary Value

The monetary value of a visitor can only be estimated until a purchase is made. A visitor who comes once a day for a week is assigned a much higher probability of purchasing than one who comes once every three months. As soon as a visitor becomes a customer, actual monetary value can be derived from spending and profit margin data. Information can be derived over time, how much does a customer buy per month? How profitable are those sales? What are the characteristics of a high spending customer versus a low spending customer?

Different sites will have different indexes for purchase probability and profitability. Historical ratings of actual customers are of great value. These are the numbers that help to recognize which users are most likely to become profitable buyers.

Duration

As it applies for recency and frequency, the duration of an individual's visits can be a clear signal of intent and a forecast of a possible change in the relationship. Different companies have different goals for the duration of visitors staying on their sites.

A technology company that offers to technologists may want to shorten the duration of each visit because their customers may repeatedly tell that they are busy. Those customers are not coming to the site to find product information quickly. When the pages load faster, when the information is easier to find, when customers can make a decision more quickly, they are happier and appreciate it with more business.

Retailers such as Barnesandnoble.com or Amazon.com want people to stay as long as possible. They want a visitor who looks for something as general as "cooking" to see the wide variety of available products. The visitor will find books, but might also find cooking software, kitchen appliances and cooking games and videos. But once a buying decision has been made, retailers make it as easy and as fast as possible to complete the purchase. Express checkout simplifying and accelerate the purchase process. Obtaining user information from available profiles rather than asking the customer also speeds the purchasing process up. Retailers in the Web observe the same trend as traditional retailers: fast check-out reduces shopping cart abandonment.

In the case of a considered decision purchase for a higher-priced product, the length of a stay may indicate the seriousness of the buyer. If a visitor stops to look at something for only a minute or two, that visitor profiles differently from a visitor who spends an hour in reviewing options.

In combination, the elements recency, frequency, monetary value, and duration paint a detailed picture of how a site is used and where changes can be made to increase its yield.

5.1.3 Customer ROI Metrics

Determining the monetary value of individual customers includes accounting for the associated costs of transitioning them through the customer life cycle, converting them into customers and holding their attention. This cost is reconciled over the customer life cycle starting with the cost of initial contact, through the acquisition phase, and finally through conversion.

Acquisition Cost

The acquisition cost calculation determines the value of a given effort for initial contact over a given period of time. If $25,000 spent on 1,000,000 banner advertisement impressions yields a 0.5 percent click-through rate, the result is 5,000 visits or $5.00 per acquired user.

$$\text{Acquisition cost} = \frac{\text{advertising and promotional costs}}{\text{number of click-throughs}}$$

Figure 42: Calculating acquisition cost.

Cost Per Conversion

If $25,000 are spent on a marketing program, and 5,000 visitors were acquired, and 5 percent of these visitors have been converted, this results in 250 new paying customers. The cost per conversion was $100. This is reasonable for high-priced products such as real estate to high-income professionals, but it is not sustainable for low-priced products.

The cost per conversion is the number that marketing people use to determine the best investment of their promotional budget. Spending $2 million on a Super Bowl ad campaign may seem like a large check to write.

But if the resulting traffic and sales produce an acquisition and conversion cost below alternative means, the cost may not be so alarming.

$$\text{Cost per conversion} = \frac{\text{advertising and promotional costs}}{\text{number of sales}}$$

Figure 43: Calculating customer conversion cost.

Net Yield

Yield is defined as the annual rate of return on an investment, expressed as a percentage. While meant to describe yield on general financial investments, this definition also applies to Web services metrics.

Yield is the measurement of how well a site is reaching its first-level goals: How many visitors have been attracted by promotions? How many of them returned two months later? How often do they return? Answers to these questions determine the shape of the customer life cycle and provide an indication of the monetary value of each user.

$$\text{Net yield} = \frac{\text{total promotion costs}}{\text{total promotion results}}$$

Figure 44: Calculating net yield.

Two examples illustrate how net yield calculations can help in decision-making processes:

- Banner A had a high click rate, but a low conversion to sale. Banner B had a low click rate, but a high conversion to sale. By comparing the net yield calculations for Banner A against Banner B, it can quickly be identified the better-performing banner.
- For a Web-based contest, some users might receive an intermediate start page before the registration page, while others might be directed to the entry form immediately. To determine the optimal path in terms of conversion, the net yield would be calculated by dividing the total number of contest entries by the total entry page visits. This allows determining whether an interstitial start page helps users to understand the contest making form completion more likely, or if it causes abandonment before forms are filled in.

Connect Rate

A promotion's connect rate is useful for identifying potential technical issues in page loading or tracking mechanisms. The connect rate measures the number of visitors who clicked on a banner or link and then successfully have been directed on a targeted page.

$$\text{Connect rate} = \frac{\text{promotional page views}}{\text{promotion click-throughs}}$$

Figure 45: Calculating the connect rate.

Ideally, the connect rate should be equal or close to 1, showing that every visitor who clicked on a promotional banner or link successfully was directed to the subsequent promotion page. If the connect rate falls below 0.80, it means at least 1 out of 5 click-throughs failed reaching the destination page. This might be due to server time outs or visitors clicking the stop or back button. It will affect acquisition numbers.

If the connect rate decreases below 0.80, the impact is substantial. For 5,000 click-throughs, only 4,000 visitors actually visit the target site, resulting in an acquisition cost of $6.25 rather than $5.00. Assuming the same 5 percent conversion rate, now 200 customers convert for a cost per conversion of $125. In this case, a connect rate of 0.80 resulted in 50 fewer paying customers and a $25 premium on cost per conversion.

Growth Rate

The growth rate indicates at what pace a site is growing in terms of visitors and converted customers, and in effect, in terms of yield and sales.

For example, a company has 2,000 subscribers on the first of the month. During the month, 200 new subscribers have been added and 50 subscribers have been lost. By the end of the month, there are 2,150 subscribers. The churn rate (see section 5.1.1) is 50 divided by 2,150, which equals to 2.3 percent.

The growth rate for the month is 200 divided by 2,000, which equals a 10 percent growth rate. Annualized, this means (assuming continued averaging the same performance each month) the company has a 27.6 percent churn rate and a 120 percent annual growth rate.

$$\text{Growth rate} = \frac{\text{customers(end)} - \text{customers(begin)}}{\text{total customers(end)}}$$

(in percent)

Figure 46: Calculating the growth rate (by example of converted customers).

Customer ROI – Customer Net Yield

Net profits are the easiest and most intuitive metric to determine return on investment (ROI). ROI alone does not alone counts for great customers, nor does ROI alone offers insight into maximizing customer life time value.

The following metrics focus on modeling, measuring, and influencing the transition from initial visitors to finally loyal customers.

Every successful sales and marketing organization tracks the sales pipeline. How many initial visitors have been converted? How long did it take to convert them? What were the necessary steps converting them?

If quick-decision, low-priced items are sold, the conversion process occurs during the marketing and advertising process. For higher-priced products, the process may involve a product demonstration or trial, committee meetings, outside consultants and client reference conversations before the sale is made.

The following are the typical stages of a sales cycle:

- Prospect: Someone who has responded to a promotion, expressing interest in making a purchase.
- Suspect: A suspect who fits the profile of current customers. Valuable for targeting advertising and promotional efforts.
- Qualified prospect: A prospect who has been contacted and their need, desire, and ability to purchase have been verified.
- Closing prospect: A qualified prospect about to become a customer.
- New customer: One who has just made a purchase.
- Novice customer: One who is in the process of implementing the product and getting training.
- Unhappy customer: One with a problem to solve.
- Referring customer: A happy customer who is willing to talk to qualified prospects.
- Company advocate: A very happy customer who helps to recruit new customers.

Tracking the life cycle status of a Web site allows customizing and tuning the site for each user. It also provides insight into the customer life cycle.

From a management perspective, these aggregate numbers are the most important result of the tracking illustrated in the table. If it is known that it takes 1,000 suspects to find 100 qualified prospects and that only 10 percent of those will end up as customers, then this information can be used as a sales forecasting tool.

The power of this technique increases after reiterating the model over a period of several months. Rather than relying on estimates, the customer life cycle model can be adjusted as monthly or weekly sales are incorporated into the model making it increasingly accurate over time. Web services metrics are becoming the central to decision points about staffing, quotas, promotions, and production capacity.

Metric	Definition	Value	If improved by 10 percent	Increase in company value
Acquisition				
Visitor acquisition cost	Marketing dollars spent per visitor	$5.68	$5.11	0.7%
New visitor momentum	Increase in number of new visitors in 2Q vs. 1Q	62.4%	68.6%	3.1%
Conversion				
New customer acquisition cost	Marketing dollars spent per visitor	$250	$225	0.8%
New customer conversion rate	Percentage of new visitors who become customers	4.7%	5.2%	2.3%
New customer revenue momentum	Increase in new customer revenue, 2Q vs. 1Q	88.5%	97.4%	4.6%
Retention				
Repeat customer maintenance cost	Operating expenses (without marketing) spent per repeat customer	$1,931	$1,738	0.7%
Repeat customer revenue momentum	Increase in revenue from repeat customer, 2Q vs. 1Q	21.0%	23.1%	5.8%
Repeat customer conversion rate	Percent of customers who become repeat customers	30.2%	33.2%	9.5%
Customer churn rate	Percent of customers who do not become repeated customers	55.3%	49.8%	6.7%

Figure 47: Bottom-line impact of improved retention.

The table shown in Figure 47 is based on a recent McKinsey & Co. report. The table illustrates how improving retention can have significant impact on the bottom line. This report concluded that the cost reductions are considerably less valuable to the company than the opportunities to decrease abandonment and attrition [Cutler 2001].

5.1.4 Web Site Behavior Metrics

The greatest value of a Web service or a Web site is its accessibility. From anywhere in the world, at any time, any prospect or customer can contact a Web service or a Web site. A Web service or a Web site can remember everything about the visitors such as when they arrived, what pages they look at, how long they spend on each page, which products they find the most interesting and more. Privacy considerations are important and must be considered throughout any customer information gathering process.

In the following sections a number of fundamental metrics are presented that are designed to provide insight into customer behavior at a Web site and providing actionable information. These metrics include:

- Stickiness,
- Slipperiness,
- Focus,
- Velocity,
- Seducible moments,
- Optimal Site Path,
- Personalization Index, and
- Freshness Factor.

Stickiness

Stickiness is related to both duration and frequency. Stickiness is a composite measure that captures the effectiveness of site content in terms of consistently holding visitor's attention and allowing them to quickly complete their tasks. In general, sticky sites are considered more effective than sites that are not very sticky. Little consensus has emerged how to calculate stickiness. One is presented here:

$$\text{Stickiness} = \text{frequency} * \text{duration} * \text{total site reach}$$

where:

$$\text{Frequency} = \frac{\text{number of visits in time period T}}{\text{number of unique users who visited in T}}$$

and:

$$\frac{\text{total amount of time spent viewing all pages}}{}$$

$$\text{Duration} = \frac{}{\text{number of visits in time period T}}$$

and:

$$\text{Total site reach} = \frac{\text{number of unique users who visited in T}}{\text{total number of unique users.}}$$

Figure 48: Calculating stickiness.

The following example illustrates a typical stickiness calculation.

A Web site has acquired a total of 200,000 unique users. Over the past month, 50,000 unique users went to the site. These 50,000 users accounted for a total of 250,000 visits (average frequency of 5 visits/unique user for the month), and, during these visits, the users spent a total of 1,000,000 minutes viewing pages on a site.

Therefore:

$$\text{Monthly stickiness} = \frac{\text{250,000 visits}}{\text{50,000 active users}} * \frac{\text{1,000,000 minutes}}{\text{250,000 visits}} * \frac{\text{50,000 active users}}{\text{200,000 total users}}$$

Monthly stickiness = 5 minutes/user.

This stickiness calculation can be applied to entire sites or sections of sites, and can also compare trends between different customer segments.

Slipperiness

Some areas of a Web site have higher customer value when they have very low stickiness, for instance, customer support. In this example, the customer support site is supposed to be slippery, not sticky, meaning that visitors quickly find what they are looking for and get out quickly.

The purchase completion and checkout section of a Web site also is recognized to be better when it becomes more slippery. Every additional click required to make a purchase represents another opportunity for the customer to change his mind and abandon the purchase.

Slipperiness is equivalent to low stickiness. Given the three factors of stickiness, a slippery measure can be seen as the reverse of the stickiness formulas.

Focus

Focus is another concept related to page visit behavior within a section of a site. Assume there are 15 pages in a section. A focused visit may touch 2 or 3 of these, a less focused visit might touch 5 or 6 of them, and an unfocused visit might touch 8 or 10 of them in order to reach the target page.

Therefore:

$$\text{Focus} = \frac{\text{average number of pages visited in a given section}}{\text{total number of pages in the section}}$$

Figure 49: Calculating the measure for focus.

If the average visitor views 3 pages in a section out of 15, then the user's focus in that content section is 0.20. Smaller values for focus are referred to as narrow focus while larger values are called wide focus.

The answer whether wide or narrow focus is better depends on the type of the section and on the user behavior considered as desirable for that section. A sticky area of content is likely to be a good indication, a sticky checkout section may signal an unnecessarily complex checkout process. Narrow focus is good at a customer service area, but eventually not at an online auction section of a site.

Figure 50 relates stickiness and focus in combination for a given section of the site.

	Low Stickiness	High Stickiness
Wide Focus	Either quick satisfaction or disinterest occurs in this section. Further investigation is required.	Either interest on this section, or users are drawn in. Further investigation is required.
Narrow Focus	Locate the correct information quickly.	Browsing indicates a site magnet area.

Figure 50: Relating focus and stickiness.

Since certain combinations are open to multiple interpretations, optimal site path analysis will help to identify reasons when surprises occur in analyzing these metrics. The combination of stickiness and focus for an entire site or a specific section of a site is generally more powerful than stickiness alone.

Velocity

On a more tactical level, tracking the customer life cycle provides the information to improve a Web site and optimize for objectives such as total-life time customer experience (taking a comprehensive set of factors into account).

Velocity is the measure of how quickly a visitor moves from one stage of the customer life cycle to the next. Clocking prospects' qualification process provides the average amount of time that it takes for a member of a given market segment to pass from awareness, through deliberation, to final decision.

With a specific time span in mind, the site designer can test alternative navigation techniques, in combination with the marketing expert experimenting with different offers. Between the two roles, they can identify so-called seducible moments.

Seducible Moments

Seducible moments are junctures where a prospect is exceptionally susceptible to an offer. It might be caused by a rapid purchase button next to a desired product, or it might be a cross-offer at the moment a customer is deciding between two service choices. A seducible moment does not necessarily have to be related to the product. It can be the point where a user must decide to join a discussion group or subscribe to a newsletter. The right encouragement or the right graphic might help to overcome these hurdles. These seducible moments will be different on every site.

Reviewing the profiles of members of a given market segment reveals different attributes of those who slowly decide and those who quickly buy. Adjusting promotional efforts toward those with a higher velocity rating can have direct effect on bottom line sales.

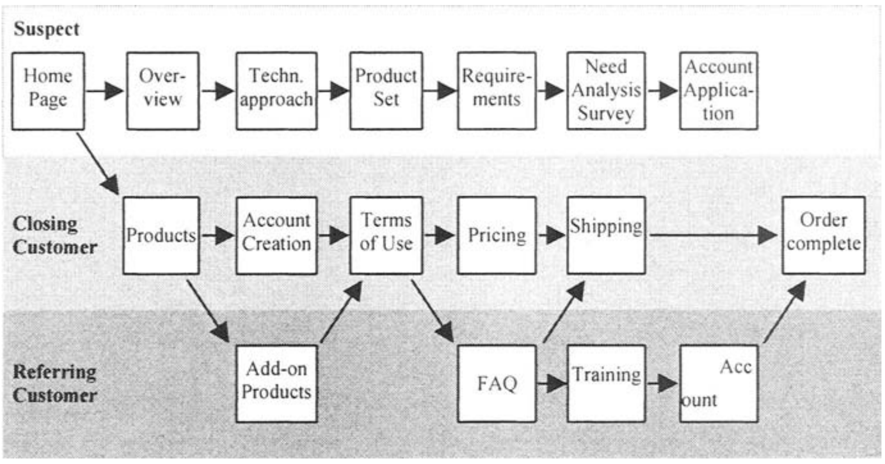


Figure 51: Showing customer's different content at different stages of the customer life cycle.

Optimal Site Path (OSP)

Ford Motor Company found that there were a number of alterative optimal site paths to reach Ford's goal of visitors asking a dealer for a quote on a vehicle. Ford is not selling cars online, but trying to increase the number of people who ask for a formal price quote and make contact with the dealership. The observation was that visitors who viewed the vehicle pages, configured the options to their liking, and selected the audio system and paint color of their choice, did not necessarily ask for a dealer quote. The visitors who turned out to be the most likely to ask for a quote were those who had also reviewed the financing options on the Ford site.

Once Ford had identified this as a critical step in the optimal site path, they were able to encourage visitors to take this step. Financial calculation tools, expanded financing options, and e-mail and Web-based reminders were introduced in order to increase the number of visitors who traversed the optimal site path [Cutler 2001].

Figure 51 shows how an Optimal Site Path (OSP) can expose where each customer resides in the customer life cycle. The boxes in the figure represent either specific pages a visitor must traverse in order to make a purchase or represent subject areas where previous customers spent specific amounts of time. It is not of primary importance whether a path requires 10 minutes or 10 weeks to traverse.

The OSP reveals the viewing habits of customers and makes it easier to classify those who have not yet purchased. It is important information whether those who actually became customers typically looked at the product specifications, the warranty, the success stories, the price and then the licensing. It is likely that other prospects will follow the same path on their way to becoming customers.

Optimal site path information suggests where a Web site should focus promotional efforts. It allows measuring the power of promotional efforts. The impact of changes made to a Web site can be measured. The speed by which new customers are acquired and the effort it takes to keep them can be derived as well.

Personalization Index (PI)

The spectrum of profile data collected from a visitor is virtually unbound. Capturing familiar items such as name, address, phone and e-mail contact, and payment information is typically included in a profile.

Collecting information is one aspect, using it is another. The Personalization Index (PI) is a measure of how effectively customer data are leveraged.

$$\text{PI} = \frac{\text{total number of profile elements in customer interaction}}{\text{total number of pages in the section}}$$

Figure 52: Personalization Index (PI).

If PI is above 0.75, then customer information is efficiently used. It is assumed that a significant number of customizable elements are used. If only two data elements are collected, PI may score 1.00, but in this context it means that only market segmentation is performed rather than personalization. Prospects and customers are only grouped into broad categories. While being simpler to maintain, broad categories are not as powerful as true personalization data based on larger numbers (dozens) of attributes.

When more and more data elements are collected, the better visitors can be classified. At some point, broad segmentation moves towards personalization, where customer strengthened and turned it into a loyal relationship.

If PI is less than 0.30, then more information is collected than used. An untapped reservoir of actionable data about customers is collected. This data, however, is probably getting stale fast.

Freshness Factor

It has long been believed that a Web site must change constantly in order to hold the interest of a given audience. While this is obvious for sites that depend on users returning daily (news, weather, and sports sites as well as portals like MSN), the constant creation of content is expensive and time consuming.

The Freshness Factor measures the impact of dedicating resources to continuous content publishing and renewal. The freshness factor is designed to measure how often content is refreshed versus how frequently users visit the site. This calculation should be performed against individual customer segments since they will be interested in different site sections and will respond to fresh content in different ways.

$$\text{Freshness factor} = \frac{\text{average content area refresh rate}}{\text{average section visit frequency}}$$

Figure 53: Calculating the freshness factor.

Just as the value of individual data elements collected about a specific customer is weighted, the value of content elements based on a number of factors must be weighted as well. The most obvious value is timeliness meaning that current information is delivered to a customer with small latency. If this data element is imported from a stock price reporting service or news feed from a wire service, some of the relevant questions are: How long does it represent valid information? When does it expire? When should a news article be moved over into the archive section?

If content consists of a series of rotating data elements, such as often used for stock quotes, it is important to determine how recently has the data element been shown to a specific user. Different articles carry different weight based on their intended use, also determining their timeliness and duration for which they represent current information. A white paper on choosing the right vendor may have duration of years, while updates on legislative issues are only interesting for one month, for example.

If the freshness factor is less than 1, then, on average, visitors or customers are visiting that section of a site more frequently than its content is updated. They are likely seeing stale content with the consequence of the stickiness parameter decreasing. On the other hand, if the freshness factor is greater than 1, then, again on average, customers see new content each time they visit a site. Stickiness should improve. If the freshness factor reaches above 1.5, then a site or a section of a site is having the risk of wasting resources to create content that is not being viewed.

One should be careful with multi-modal distributions. If half of the customers visit every hour and the other half visit once every 9 hours, then average frequency is 5 hours. This reveals the general necessity of segmenting customers and performing all calculations, including freshness factor, on customer segments rather than global site populations.

SUMMARY

In this chapter we discussed Web services business metrics as they can be measured for traditional Web sites (in the business-to-consumer domain) as well as becoming increasingly important for Web services (in the business-to-business domain).

Historically, it has been seen how business culture at the beginning of the 20th Century shifted from the personal attention of the corner store to mass production by assembly lines and vertical integration. The way to success was to produce more goods, faster and cheaper. In the second half of the 20th Century, the focus shifted to distribution. Wal-Mart mastered transportation and logistics of how mass-produced products could be brought to market faster and cheaper.

At the beginning of the 21st Century, the information age is transforming into the communication age with mining and utilizing the knowledge the Internet and the Web provide for mediating business relationships and delivering services. Knowledge about individual customers can be utilized as it has never before been possible.

The value of knowing customers as individuals is enormous. The Web provides the ability to capture, store and act on information about individual customers as a competitive advantage. With personal information about individual customers Web experience for each user can be customized, buying pattern can be accurately predicted, products and services can be customized meeting specific customers' needs, inventory costs can be lowered, and customer relationships can be maintained at very low cost.

The result is the ability to bind customers into long-term relationships through increased customer satisfaction and increased profits. If vendors are agile and can readily meet new client requirements as they occur, the motivation for those clients to find other vendors will be low.

The opportunity to analyze more than just customer information is rising faster than ever. Today, customer profiles that cover such minimal data as gender, zip code, number of visits and a few preference profile elements. Tomorrow, focus will be more on the interaction rather than the information delivered during the interaction. Customer profiles will include information on how much customers know and how they like to communicate.

The store of information collected while doing business on the Web should be carefully archived and protected, even if this data is not used today. Raw data will allow for deeper and deeper analysis in the future. Even though sites change quickly that six month-old behavior data may not have the same relevance to the current site and infrastructure as current data. Experience from traditional sales and retail tells those patterns reoccur.

Web sites are capturing more and more data every day, performing deeper analytics and data mining and dynamically altering site content based on this analysis. We have discussed some of the most fundamental data and classified them into the metrics presented in this chapter. With the advent of web services not much will change in terms of definition of business metrics. It is the mode of collection of these metrics that will change. The information collected has to be at the level of web service operations, the conversations that these web services undertake with customers and the correlations between these pieces of information.

With systematically collecting, structuring and analyzing the metrics that have been presented, the opportunity exists to approach business in the Web from an objective and systematic perspective in order to become and remain a trusted member of the customer's value chain.

Chapter 6

MANAGING APPLICATIONS AND IT INFRASTRUCTURE OF WEB SERVICES

1. INTRODUCTION

The intra-enterprise management aspects involve managing the underlying network and system infrastructure and also managing the Web service infrastructure and platforms. Web services cannot be adaptive to changing workloads and planned/un-planned downtimes, without managing the underlying network and system infrastructure. Along with managing the network and system infrastructure it is important to manage the Web service infrastructure as well.

1.1 Application View of Web Services

A Web service typically is implemented by various applications in a so-called n-tier architecture. This architecture consists of an inbound connection point, which typically also provides firewall filtering and load balancing capabilities (shown as the load balancer in the figure). Behind the load balancer, a farm of web servers serves basic HTTP requests for static content (static HTML files and images). Web servers provide the horsepower to a web site or a Web services site. They are the direct counterparts for users behind browsers experiencing the service. Load balancer and web server farm form the first tier in this architecture.

Those requests coming from users that are relevant for the Web services application, such as queries in catalogues or orders, are forwarded to the application server tier, which forms the second tier in this architecture. Application logic drives the dialog with users or other Web services.

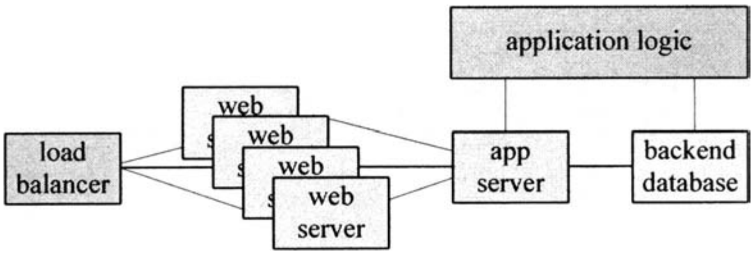


Figure 54: Application view to a Web service: n-tier architecture of a Web service.

The third tier then is comprised of the database backend, which stores all relevant data about service data, catalogues, and users.

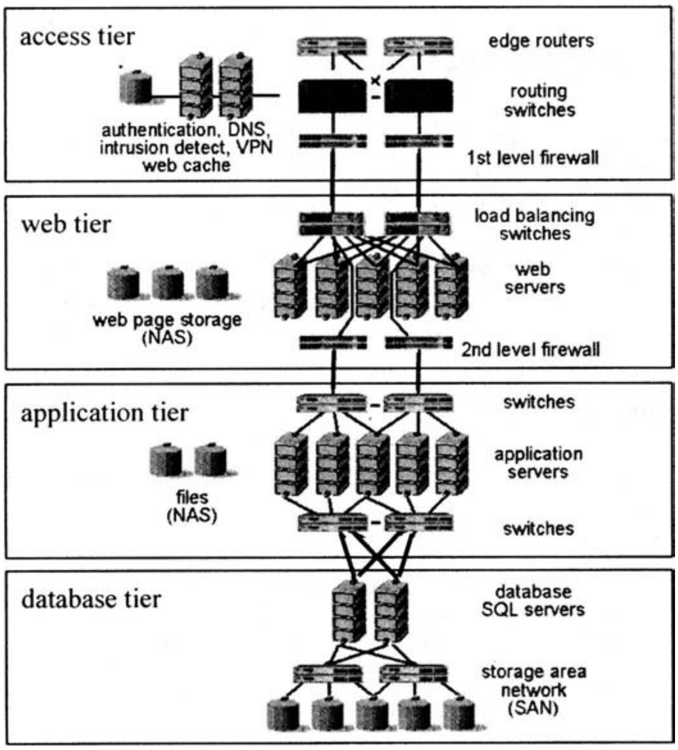


Figure 55: A typical infrastructure view to a Web service.

Monitoring such a system can be done from an infrastructure perspective as "black-box" monitoring by observing the behavior (response times, failure rates, etc.) from outside. Alternatively, components in tiers can also be instrumented and monitored and observed separately. Means to instrument and measure application components are discussed later in this section.

The application view of a Web service is complemented by a so-called infrastructure view with machines, networks, and devices on which the Web services applications reside (servers) or which they use (networks, devices).

The next section discusses the infrastructure view to a Web services system that is comprised of firewall and load balancing devices and servers that are connected through subnets.

1.2 Infrastructure View to Web Services

The following figure shows the infrastructure for Web services. The structure contains three subnets that connect the inbound load-balancing device (named lb in the Figure 55) to an outbound Internet connection. From the load-balancing device, a second subnet connects to the farm of servers operating the web servers.

Static content is directly served from web servers from storage attached to web servers. Separate servers are needed to host application servers and database backend. Those machines are connected to web servers through another subnet.

The reason why multiple subnets are used in the infrastructure is primarily to decouple and isolate components (for reasons of failure isolation and security protection).

Traditionally, tiered architectures have been deployed in data centers manually by selecting and configuring machines and devices (disks, switches, routers). Recently, data center solutions that automate those tasks to a large extent, have appeared. Based in descriptions similar as shown in Figure 55, these data center infrastructure solutions automatically assign resources and configure them properly such that applications such as Web services can be deployed on them.

Examples of such data center solutions are HP's Utility Data Center (UDC), IBM's Blade Center, Sun's N1 (N1), or Microsoft's Dynamic Systems Initiative (DSI).

2. LINKING IT INFRASTRUCTURE MANAGEMENT WITH WEB SERVICES

The following figure shows three levels of management tasks from the point of view of the IT layer.

Business Impact Management

- Align IT management with business objective
- Assure customer service levels
- Improve cost efficiency through IT planning
- Turn data into business information

Event Correlation and Automation

- Understand cross domain dependencies
- Improve availability with fast problem resolution
- Elimination of potential point of failure

Infrastructure and Application Management

- Identify, notify and cure problems at the source
- Auto-discovery of critical resources
- Automated problem resolution
- Response time analysis

Figure 56: Three levels of IT management.

At the lowest level, basic monitoring techniques are applied. At this layer, applications monitor hardware and software, and provide automated corrective actions when possible. With an infrastructure architecture based on monitoring models, monitoring provides a solid foundation for the development of management solutions addressing the complex needs of Web services IT infrastructures.

The next level is event correlation and automation. As problems occur that cannot be resolved at the monitoring level, event notifications are generated and sent to a correlation engine. The correlation engine at this point can analyze problem notifications (events) coming from multiple components, and either automate corrective actions, or provide the necessary information to operators who can initiate corrective actions.

The third level in this structure is called Business Impact Management. It gathers management information from various part of the enterprise, and provides an insight into how a component failure may affect the business as a whole. For example, when a router failure occurs, it is important to understand what line of business applications will be affected and how to reduce the impact of that failure on the business.

Monitoring for Web services Infrastructure involves three major areas:

- *Availability management.* Based on monitoring in the Web services infrastructure guided by monitoring models, or groups of monitoring models, the status of Web service components is constantly checked including the Web servers, the application servers and database backend. The status can be either active (operational) or inactive (non-operational). Monitoring models can be customized adapting monitoring models to requirements.
- *Performance management.* The modules' monitoring models enable to measure and report the performance of virtual hosts run by Web server resources in order to identify bottlenecks and potential problems in the Web services infrastructure. For application server some of the key performance metrics are:
    - Enterprise JavaBean (EJB) performance,
    - database connection pool performance,
    - JVM runtime performance, and
    - Servlets/JSP performance.
- *Operations management.* Monitoring a Web services infrastructure enables managing Web server and application server resources on a scheduled basis. One can start, stop, and restart these servers, check for the status of these servers, and retrieve and analyze trace log files

Monitoring a Web services infrastructure is often based on events and rules about how those events are triggered. Events are issued for those situations than need to be propagated to an operator console.

3. MANAGING THE APPLICATION INFRASTRUCTURE OF WEB SERVICES

3.1 Metric Collection and Analysis in Application Infrastructure of Web Services

A strong Web services business metrics capability is a core component in building the right e-business and CRM architecture. Web analysis is a first order task in the operational infrastructure needed to build and maintain online business.

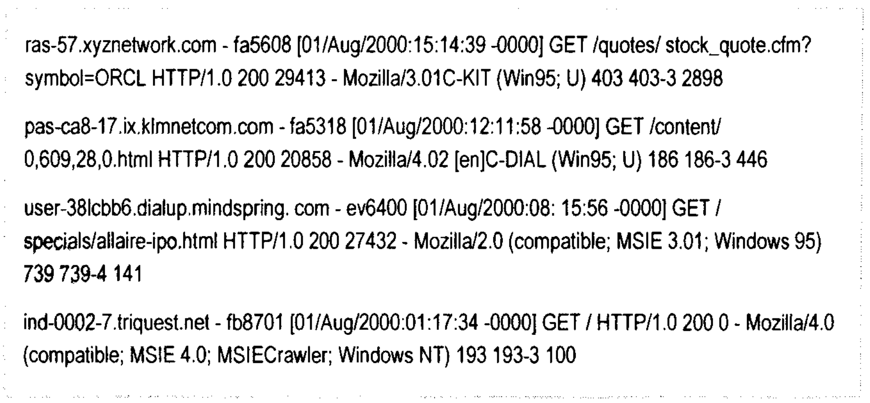
Supportive tools extract, analyze, store, and correlate base information of user behavior (clicks) with other online and offline information. The resulting Web services business metrics is leveraged to measure and improve the financial business performance. Business decision makers have immediate insight into dynamic and interactive reporting and data mining. They can export Web services business metrics to their warehouses and CRM applications.

The six most difficult technical challenges are:

1. deep, complete analysis of online data,
2. easy access to Web services business metrics,
3. flexible interoperability with e-commerce and CRM systems,
4. ready integration with the corporate data warehouse,
5. high performance and scalability, and
6. an open and customizable enterprise-strength solution.

It is essential to invest in a data warehouse solution in order to unify and act on customer information. A flexible Web services business metrics coupled with strong analytic software tools is vital for managing business aspects of Web services. Delivering its value, a Web services business metrics approach works with and enhances the ROI of other investments in the corporate IT infrastructure.

One problem of drawing the link from high-level business metrics to technically measurable data is the gap between information present in measured data at different levels.



```
ras-57.xyznetwork.com - fa5608 [01/Aug/2000:15:14:39 -0000] GET /quotes/ stock_quote.cfm?
symbol=ORCL HTTP/1.0 200 29413 - Mozilla/3.01C-KIT (Win95; U) 403 403-3 2898

pas-ca8-17.ix.klmnetcom.com - fa5318 [01/Aug/2000:12:11:58 -0000] GET /content/
0,609,28,0.html HTTP/1.0 200 20858 - Mozilla/4.02 [en]C-DIAL (Win95; U) 186 186-3 446

user-38lcbb6.dialup.mindspring. com - ev6400 [01/Aug/2000:08: 15:56 -0000] GET /
specials/allaire-ipo.html HTTP/1.0 200 27432 - Mozilla/2.0 (compatible; MSIE 3.01; Windows 95)
739 739-4 141

ind-0002-7.triquest.net - fb8701 [01/Aug/2000:01:17:34 -0000] GET / HTTP/1.0 200 0 - Mozilla/4.0
(compatible; MSIE 4.0; MSIECrawler; Windows NT) 193 193-3 100
```

Figure 57: Raw data collected from HTTP request traces from Web logs.

The sample data shown in Figure 57 below is a simplified example of raw page view data from a Web log. While offline application data is quite structured, online data is fairly unstructured. Online data is largely not easily interpretable without detailed knowledge of the chosen platform, the site design, the architecture, and policies, augmented by the information stored in other logs and associated databases.

Data elements must be extracted from the raw data representing Web activity in a Web log. These data elements include cookies, session information, IP addresses, domain and sub-domain names, time stamps, request types, resource types, dynamic content, user query strings, protocols, status codes, bytes transferred, information about user agents and referrers. Analyzing this data requires a detailed knowledge of the Web site and its internal design. It is also highly dependent on underlying systems and prone to fail when technology used for the Web site changes.

The raw data provide the input for a complex, rule-based process to link requests to visits, and visits to users, and to interpret what is really going on across multiple sites driven from a business perspective with the final goal making right business decisions.

For both anonymous and identified interactions, Web service management systems collect detailed customer behavior data. By recording every click at every moment, the system collects the data points generated by a visit to a Web site available to those who can use them. As a result, Web services systems are able to paint a richly detailed portrait of customers, and can determine how to get more of customers to return, or visitors to be transformed into customers.

The following section discusses an example of Web services metric analysis architecture and the technical aspects associated with it.

3.2 Web Services Metric Analysis

A typical Web services Metric Analyzer contains a high-performance relational database such as Oracle, Microsoft SQL Server, or IBM DB2. It must have high-performance since it will deal with high data volumes for both importing and reporting. The schema of the database is carefully organized as a constellation schema, whose major tables cover the three relevant levels of hits (requests), visits, and users. Page views for any given visit are linked together in order, making it possible to analyze complete sequences of user clicks.

Dimension tables include resources, browsers/platforms, sub-domain/organization, time, referring sites, query string elements (both those from actual user searches and those used to describe dynamically served content), and many other online data elements. Aggregate tables are used to hold pre-computed totals, averages, and other summary information used to greatly speed up queries and interactively drill down information. In addition to reducing response time, aggregates enable effective data management since they can be retained for reporting after the underlying data is deleted.

Typically, a Web services Metric Analyzer can be extended and customized to accommodate additional data from online and offline sources and to adapt to customer requirements.

The major task of the Web service Metric Analyzer is to extract meaningful data from raw technical data. It constructs a comprehensive view of user activity in near real-time. Recognizing the heterogeneous and complex nature of today's Web environments, it is data-source agnostic. It supports data collection through Web server log files, network packet sniffers, and Web server plug-ins and other instrumentation techniques (Pg 187). The choice of Web Data Collector depends on the data that are interested in a given environment for analyzing behavior and the topology of a Web site. While behavioral data traditionally originates from Web server logs, other data sources can be used in conjunction with or instead of log files.

Web data volumes are vast. Large corporate sites attract tens to hundreds of millions of hits per day, and major portals surpass a billion hits per day. Compared to the data processed in traditional customer data warehouses, Web data represents many more interactions, consists of lower level behavioral observations (not just transactions and promotions), changes much more frequently, requires a higher degree of responsiveness, and combines actions from both customers and a larger pool of visitors. Large data volumes are the main reason why scalability is a major issue for effective Web services metric analysis.

Scalability is a multifaceted goal, comprising both Web services Metric Analyzer update and access through Web services Metric Reporting. Supporting systems have to be designed and implemented with high scalability and performance as primary requirements.

Web services metric analysis is unlike the traditional data warehouse problems. Standard warehousing approaches have been developed to load a huge amount of highly structured data into a database in larger periods such as once every month. In contrast, Web services metric analysis is refreshed with up to tens of gigabytes of relatively unstructured new data every day.

Monitoring models provide a systematic foundation for monitoring and archiving data that characterize the behavior and the boundaries of "normal" behavior for a Web service. Monitoring, or more general, management models are also the precursory for tool support and implementing an effective management solution for a Web service.

Business processes are becoming more integrated within the enterprise. IT investments are evaluated by their impact on business value, that is, the profitability of the enterprise, gains in market share, and gains in customer satisfaction. Today's enterprise is far more dependent on the availability of IT service to conduct business. Business managers expect to buy IT services on an as-needed basis. Understanding how business units, partners and customers access and use IT services is critical to managing costs and optimizing their return on IT investments. Business-aware management systems provide a window into how resources such as storage, servers and bandwidth are consumed and which business functions depend on them.

3.3 Rule-based Processing

Sophisticated rules drive Web service metric analysis. It is specifically challenging to effectively process the large volumes and vast quantities of data. Rules fall into different categories:

- where to locate data,
- when to look for it,
- what data to exclude,
- how to recognize users,
- how to define visits,
- what data aggregations to compute,
- how to classify users,
- how to identify user segments, geographically or based on other criteria.

These rules are configurable per site and responsive to the demands of business. Rules represent the knowledge about business aspects into an effective software solution.

For example, all high-traffic sites have DNS resolution turned off in order to improve Web server performance. DNS resolution turns IP addresses into hostnames and meaningful business information (e.g., international traffic vs. domestic, home users vs. corporate users). Configurable caching within the DNS resolution engine speeds up lookups. To help business users to learn more about the names and the geographies than the hostnames can provide, the Web service metric Analysis can incorporate a database that maps sub-domain names into corporate and geographic information. This allows to understand the identities of users and to segment their users by geographic location.

As another important example, user recognition may be based on authenticated user IDs, based on cookies, on hostnames plus browser information, or on specified combinations of these techniques. The combination of techniques may also be specified through rules. An advanced capability is to reconcile visit and hit counts across different user identification methods. Even if the identification method changes during a visit, the change can be tracked.

Protecting privacy is an issue that counters the efforts to collect and gather as much information as possible about users and their activity on a Web site. Users who wish to remain anonymous must be honored and filtered out from the collected data.

3.3.1 Handling Multiple Sites

Larger Web sites are distributed across different geographic locations. They want to do analysis across networks of logical sites or acquired sites. In the publishing industry, for instance, acquired sites are called properties, which is the term used for magazines and other distinct brands. At the database schema level, notions of property and property group must be associated with each page hit. Database-level resource grouping enables analysis by the individual property yet still consider the entire collection as a single entity (meaning that a visit can span properties). Similarly to property-level reporting, analysis can be done across constituent servers or server clusters at different sites.

3.3.2 Web Service Metric Reporting

Web services Metric Reporting can be implemented within a thin-client Web interface and, for instance, installed as Application Server Pages (ASP) on an Internet Information Server (IIS) on NT. The thin client calls the so-called OLAP (On-Line Analytical Processing) [Rohm 2003] interface that in turn reads and writes metadata information and fetches report data from the Web services metric analysis.

Web services Metric Reporting can offer analytic flexibility based on multidimensional data analysis and data visualization. Operators can log in to see reports on Web services metrics, extend them, and create customized reports. They can segment users, content, or any other dimension.

Web service Metric Reporting typically offers visually oriented capabilities to drill, sort, cross-tab, pivot, bin, graph (2D or 3D) and page through report data. Drilling takes the user down into details (e.g., from months to days), up to an aggregated view (e.g., from a geographic state to the whole country), or across (e.g., from visit durations to referring site for a given day). With a cross-tab view, an operator can group data by multiple dimensions (e.g., frequency of visit by user segment). Pivoting allows rearranging the orientation of dimensions on the screen (e.g., flipping rows to columns, or changing the nesting of row dimensions). Binning lets organize data into groups based on value ranges (e.g., visit counts can be divided into ten bins based on the number of pages viewed). Complex formulas, filter data, and define custom groups (or user-defined segments) can be computed based on filtering multiple dimensions.

Reports are typically organized in hierarchical views of folders corresponding to the customer life cycle. Reports can include various metrics. Multiple reports are combinable in HTML in order to be viewable in Web browsers. HTML documents can be customized for report generation.

Often, asynchronous reporting, and cross-user report caching is supported in order to speed up subsequent requests for a report. It is also often possible to directly view the underlying database SQL for a report and to export reports in known formats such as Excel, Lotus, HTML, or plain text.

Web services metric analysis includes interrelated reports, attributes, attribute hierarchies, filters, metrics, custom groups, and documents. Operators and operator groups can be given privileges ranging from narrow to broad. For example, some operators may be restricted to just viewing reports; others may have the ability to view and create new reports.

3.4 Basic Monitoring Functions in Web Services Applications

The basic premise of monitoring an application is to ensure that states of a monitored application component are in a "healthy" state. Whenever the state of the component is not healthy, such as exceeding a monitoring threshold, an early warning signal is sent to the operator for action. The action and resolution of the problem depends on the operator. Typically, events and monitoring data from various sources need to be combined and further analyzed in order to resolve the problem.

As an example is shown how a basic monitoring profile can be created that monitors the general health of the Application server instances. When specific problems occur, additional monitoring is temporarily deployed to further pin-point the problem by gathering and analyzing this additional information. When the problem is resolved, the additional monitoring instrumentation is removed from the system.

The following table shows a monitoring profile for all the Application server instances. This profile contains a description of the steady state threshold during normal operation. Steady state is generally achieved after running the monitor for a while, and deciding about the most reasonable thresholds. The goal is to determine thresholds that will detect an anomaly while masking the normal fluctuation, and not create false alarms. It is advisable to start in an environment with a test system rather than the actual Web service in order to verify functioning of the monitoring system and in order to obtain a first-level approximation of thresholds.

Application component	Indication	Threshold	Cycles (secs)	Occurrence / Hole
application server status	application server down	-	60	1/0
DB pools	connection pool timeout	0	90	9/1
	DB pool average wait time	300 ms		9/1
	percent connection used	80%		9/1
EJB	EJB performance	1200 ms	90	9/1
	EJB exception	30%		9/1
HTTP sessions	live sessions too high	2000	180	9/1
JVM runtime	used JVM memory	80%	60	1/0
thread Pool	active thread load	80%	180	9/1
transaction	transaction response time	1000ms	180	9/1
	transaction time out	3%		9/1
Web application	Servlet/JSP errors	0	90	9/1
	Servlet/JSP performance	800ms		9/1

Figure 58: Example of a monitoring profile for an application server.

The profile shows almost all typical parameters that are used for monitoring an application server. This configuration is also called the monitoring model for the application component. The monitoring model also shows the data that are archived on a disk.

Another advantage of application-level monitoring is the ability to interact with other available monitoring tools via SNMP and other monitoring tools. When the monitoring architecture is not event-based, it can communicate with other monitoring tools via an event emulation layer that acts as a proxy between the monitoring system and other monitoring tools. The main function of an event emulation layer is to mimic the event producer interaction by requesting and collecting the monitoring information from the external tools, and providing it in event-based format to the monitoring architecture.

4. LINKING INFRASTRUTCURE MANAGEMENT TO WEB SERVICE MANAGEMENT

Over time, network and system components have proliferated and have been deployed across wide networks in enterprises. As the complexity of infrastructures increases, so increased the need for management software to handle this complexity, anticipate problems and resolve them before they impact the business.

Managing the components in an IT infrastructure begins with well-managed devices. Servers, disk and tape drives, workstations, printers – all have management software specifically designed to optimize their availability and performance. Management software provides comprehensive configuration capabilities and detailed system information necessary to keep the hardware up and running and reduce the total cost of ownership by automating or simplifying manual configuration processes.

The main areas of infrastructure management are:

- Network management,
- Systems management,
- Storage management, and
- Application management.

Gaining control over IT infrastructure also requires the ability to monitor and manage larger compounds of networks, servers, storage racks, and applications such as databases and other. Management tools are provided by vendors and deliver system-level deployment, configuration and monitoring capabilities. Additionally, management systems such as HP OpenView assist management of an entire infrastructure comprised of a multitude of components by providing visibility into those infrastructure components, along with fault management, performance management, application management and operations management. Through integration with system-specific infrastructure management, they provide on-demand information about the status of each component and present this information to operator consoles in form of consolidated views.

4.1.1 Network Management

Network infrastructures have become more complex than ever. Whether switched IP networks, or providing new services over a converged voice and data network, the need to manage the availability and performance of the network as well as the dependability of IT systems and business have never been greater. Network management systems offer enterprises and service providers the ability to manage voice and data networks through a modular set of products and services that can be tailored to specific needs.

Network management systems are able to:

- Discover the relationships of network devices to each other and map out a topology of the network. Since networks are constantly changing, automated discovery and generation of topology maps are essential functions for a consistent view into an ever-changing network topology.
- Address scales of enterprise networks comprised of hundreds of thousands of components connected through the network.
- Monitor networks for availability. This is achieved through using a combination of industry standards and proprietary methods that are supported by underlying network switches and routers and aggregating and presenting this information in ways that are understandable by administrators.
- Measure the performance of networks and provide detailed insight and analysis regarding network utilization and potential hotspots in a network.
- Integration with management tools from different vendors allows administration from a central location and with a unified view.

Some network management products, such as OpenView's Network Node Manager (OV NNM), have become de facto standards in the industry because of their capabilities, reliability and install base. Those systems follow open architectures that allow customers to integrate and operate hundreds of components.

Key functions of network management systems are:

- Automated network discovery and topology layout.
- Fault management.
- Performance management.
- Predictive management – isolate current and future bottlenecks.
- Problem diagnosis – measure performance and availability of specific network paths.

Characteristics of typical network management solutions (e.g. HP OpenView Network Node Manager product) are:

- Manage layer 2 switched networks and IP multicast networks.
- Support for IP, ATM, and telecommunications infrastructures.
- Problem detection with statistics, alarms, and maps on a display.
- Reduced problem resolution time in complex networks with accurate views and automated problem analysis.
- Enabling viewing the topology of a multi-cast environment and its status.
- Performance insight for networks by application monitors and reports on network protocols and devices.
- Problem diagnosis by providing status and performance information on static and dynamic network paths.

4.1.2 Systems Management

Systems management solutions provide a comprehensive collection of systems management functions that enable to simplify an environment and deliver better service. At the system level, tools are needed to manage components in an environment. System-level management tools provide configuration capabilities and detailed system information that are needed to manage servers, storage devices, clients and printers.

Environments are typically heterogeneous with servers, storage, devices and printers from a variety of vendors. A management solution is needed that monitors, controls and reports the health of the entire environment with all its heterogeneity in order to diagnose and resolve problems, as well as optimize performance and utilization. Multi-platform systems management solutions meet those needs.

Some typical characteristics of system management systems are:

- Multi-platform support by supporting management agents for a variety of operating systems.

- In-depth operating systems management support through plug-ins providing the linkage to applications.
- Management agent hierarchies characterized by:
  - The purpose of reducing data and traffic in case of event storms by solving problems locally,
  - Configurability through scripting and lightweight programming support, and
  - Improved process control and server-agent communication to reduce administration overhead and simplified use.

4.1.3 Storage Management

The speed of performing business on demand requires agility based on availability and access to large amounts of information, at the right time, presented in the right way to the right decision makers. An automated, managed storage network enables this.

Automation gives data and storage administrators the key to exercising control while optimizing utilization and cost.

Key capabilities of automated storage management systems are:

- Enterprise-wide storage capacity management for high availability and proactive planning.
- Interoperability in multi-vendor storage networks, for maximizing performance and return of investment.
- Integrated storage services management for resource allocation and reliability.
- Recovery in case of failure or data loss including instant recovery, system recovery or site disaster recovery.
- Defined provisioning policies and processes for storage capacities.
- Tool support for capacity planning.

Giving storage administrators these kinds of capabilities is the motivation behind storage management products. They offer stability within a business environment that can be volatile and unpredictable.

Characteristics of storage management solutions are:

- increased data availability by:
  - backup, recovery, and instant recovery from volume copies,

  - local and remote data replication.
- increased management efficiency by:
  - centralized storage management,
  - asset inventory,
  - array configuration, asset inventory and problem identification.
- simplified management of storage network resources by:
  - automated management of network storage resources (tape and disk, direct- and network-attached storage, storage area networks (SAN) and heterogeneous storage devices),
  - uniform management console and a common reporting structure.
- improved performance management by:
  - optimized array resource allocation.

Current examples of storage management products are (chosen from the HP OpenView suite):

- Continuous Access Storage Appliance – a storage application platform, enabled by virtualization technology, that delivers local and remote data mirroring, data snapshots and migration across heterogeneous storage devices.
- Device Plug-ins – extends capabilities of the Storage Area Manager through integration with the rest of the IT environment.
- Storage Area Manager – seamlessly integrated software suite for managing and monitoring multi-vendor, direct-attached and networked storage environments.
- Storage data protector – combines both disk- and tape-based data protection across multiple storage architectures, applications, and operating environments to deliver various levels of recovery.
- Storage Media Operations – automates tracking/management of hundreds of thousands of removable storage media.
- Storage Provisioner – capacity management and utilization tool that resolves unpredictable storage demands in complex IT environments.

4.1.4 Application Management

According to recently published industry analyst reports, applications, more than any other components of an infrastructure, are responsible for downtime in critical business systems. As a result, managing the availability and performance of critical business applications has become more important than ever.

Application management systems must provide insight into the structure, availability, and performance of applications. This insight is typically obtained through so-called management plug-ins, instrumentations that are induced into an application environment for observation and actuation of operations received from associated management agents. Each plug-in has application-specific intelligence, understanding the expected behavior and structure of a particular application, and notifying the appropriate management agent when application processes, log files, and other mechanisms indicate a problem with performance or availability.

Application management systems offer application management for nearly every major business application. Additionally, they can be extended to any off-the-shelf application, as well as own applications developed within an enterprise. By integrating application management into management consoles, application administrators can understand the availability of applications, understand related problems in the infrastructure which may impact applications, and assess the impact of application downtime on an overall IT service. Application management systems have become critical pieces of enterprise IT management implementations.

Characteristics of application management systems are:

- Manage performance and availability of a range of applications.
- Automatically instrument agents to manage key attributes of an application.
- Automatic service discovery to improve time to achieve full service management.
- Templates can be built to manage a large variety of applications.
- New Web services management engines allow to:
    - capture Web services transactions,
    - actively manage Web services interactions over WSDL and SOAP,
    - enable provisioning, user profile registration, real-time SLA enforcement, access control, subscription, management, and accounting,
    - identify performance bottlenecks such as provided by transaction analyzing tools.

- Performance management by monitoring agents allows a uniform interface for monitoring, analyzing, and forecasting resource utilization assigned to applications.

5. NEW DEVELOPMENTS IN IT – INFRASTRUCTURE MANAGEMENT FROM THE WEB SERVICE PERSPECTIVE

Enterprise IT organizations continue to adopt the idea of IT service management. Industry standards are enabling more component-based application development and deployment. Application platforms or servers such as BEA Web Logic (BEA WebLogic Platform), IBM WebSphere [Sadtler 2004], or Microsoft Transaction Server (MTS) have become the foundation for Web services-based applications.

Measuring transactions and messages exchanged between Web services-based applications is becoming central for application and services management.

Two main approaches to managing services are discussed:

- IT as a service provider.
- Virtualization as Enabler for Turning Resource into Services.

Enabled by Global Grid Forum (GGF) initiative named Open Grid Services Architecture (OGSA) [Foster, 2002], resources maintained in IT infrastructures are more and more turned into services unifying resources and applications under the same notion of services. The hard line between resources and applications is fading, and the new notion of services is emerging. Consequently, IT management with its facets of network, systems, storage and application management turns into service management, and IT providers turn into service providers.

Virtualization is the technology that enables turning resources into "programmable" services. Virtualization dissolves the hard ties between applications to the machines onto which they are installed and operating. Virtualization introduces a control point between physical resource devices and corresponding virtualized abstractions largely hiding the aspect of heterogeneity.

The virtualization control point can be used for a multitude of benefits:

- improved utilization by sharing underlying physical resources,
- protection and isolation of application domains in resources, and

- temporary suspension of applications from resources without de-installing them or reconfiguring them at reactivation time.

The technology shift caused by virtualization finally enables what is called service management in the following sections.

5.1.1 IT as a Service Provider

Seeing IT as a service provider means, for instance, that IT is planning services that meet required availability, performance and security requirements and which conform to agreed-upon service levels and costs. IT is managing Service Level Agreements (SLA) for both internal and external customers to deliver IT services and manage service delivery at agreed upon quality and cost targets. When a service is disrupted, or performance degrades, IT must not only know which devices are involved, but which services are impacted and which actions will provide the greatest positive impact on the business of its customers. By measuring the results, IT can make its own determination based on empirical knowledge of what actions are appropriate from the business perspective.

Enterprise IT departments as traditional cost centers turn into organizations that can create value and are driven by internal or external customers. Value creation results from making the right decisions based on the knowledge from the business level down to the physical devices level.

It is not enough just to know when something is not working such as knowing when the performance of a service is degrading or not meeting service level objectives. Service management systems must proactively track performance, monitor performance against alarm thresholds and maintain performance data, not only for later analysis, but also for justification and proof of whether or not SLA have been met. Reputation and revenue of an IT services provider depends on proof and evidence that it can meet SLA.

With service management systems, actions can be prioritized by business impact. For example, one can prioritize alarms and events related to a Web server of a customer's online ordering system over a problem in an internal email system. Those priorities must be specified in form of policies and are obeyed by the service management system. Depending on the type of alert, the service management system advises operators to initiate action, or it automatically interacts with the resource virtualization environment to provision resources on the fly that would resolve a problem.

5.1.2 Virtualization as Enabler

By consolidating IT infrastructure while ensuring business continuity, integrating enterprise applications and data, and managing the network effectively, IT service departments move toward a model in which computing services can be delivered on an *as-needed* or *on-demand* basis. IT infrastructure automatically and adaptively matches resource capacity to service demands in real time.

IT infrastructure must match resource capacity to service demands in real time. Virtualization technologies, built-in across the entire infrastructure and data centers, create a compound effect – infrastructure is utilized where and when needed, and reallocated on-demand when business and IT conditions change. By its nature, virtualization reduces inflexibility in the infrastructure so that business processes, applications and services, and the underlying infrastructure can more quickly and easily shift as needed. Virtualization technologies enable infrastructure flexibility by allowing the pooling of IT resources as a single logical entity whose services can be delivered in a consistent manner anywhere, and managed in a consistent manner from anywhere. Virtualization enables more cost-effective and rapid scalability as well as greater deployment flexibility. Virtualization can be extended over time staring with single virtualized resources towards sophisticated virtualization across multiple systems and ultimately across data centers.

Virtualization techniques, built-in over time across the entire IT infrastructure and data center assets, create enable that effect.

Virtualization occurs at two levels:

- Services are virtualizing applications from a customers point of view, customers (or clients) interact with a clean service interface rather than with individual applications, and
- Turning resources into (resource) services decouples serving application's resource needs from underlying physical resource infrastructure.

When an application fails, a corrective action could be to direct client requests at the Service Access Point (SAP) to another application hiding that internal disruption from clients using that service.

Similarly, when physical devices fail, the resource virtualization layer can seamlessly fail-over to other physical resource serving the needs of the application, without noticed by the application. Disk RAID systems are an early example. Servers and networks are now becoming subject to resource virtualization as well.

Both virtualization layers are the essential technical core for service enabling and enabling services management.

With virtualization, infrastructure can be utilized where and when needed, and reallocated on-demand when business and IT conditions change. By its nature, virtualization reduces inflexibility in the infrastructure so that business processes, applications and services, and the underlying infrastructure can migrate more quickly and easily as needed.

Virtualization techniques enable infrastructure flexibility by allowing the pooling of IT resources as one single logical entity whose services can be delivered to any application.

Advantage of virtualization is achieved today by systems and solutions that incorporate these techniques. These systems will create the foundation that will expand over time with more sophisticated virtualization across multiple systems and ultimately across data centers.

Various resource virtualization techniques exist:

- *Virtual server* environments intelligently scale servers for horizontally and vertically scaled applications such as web- or applications servers up and down based on service level needs and workload conditions. This is achieved by continuously monitoring application service levels and adjusting resource allocations as required. Virtual server environments combine the power of server clustering, capacity on demand, server partitioning, and rapid deployment with an intelligent policy-based server provisioning engine.
- *Virtual networks* divide the available network bandwidth into multiple independent and secure channels that can be dynamically and transparently assigned to specific applications without requiring changes to the physical connections paths between network switches.
- *Virtual storage* provides an abstraction layer between physical storage devices and the logical volumes used by applications. Storage virtualization solutions are encompassed in scalable, RAID-based storage architectures. Several levels of storage virtualization are typically offered: server-, array-, and network-attached storage virtualization.
    - *Data center virtualization* combines server, network and storage virtualization for an entire data center. For data center virtualization, HP's Utility Data Center (UDC) is an example. It can provision infrastructure for new applications and services on the fly.

6. NEW CHALLENGES DRIVING INFRASTRUCTURE MANAGEMENT

In the 1980s and 1990s, IT infrastructures were built largely by taking a silo-based approach. Business processes were delivered through custom applications and made available through disparate sets of computers, storage and software. And as those silos have become more complex, larger and larger IT staffs are required to maintain and evolve these systems. While this vertical approach to building IT infrastructures has been the case in the past, it is becoming increasingly clear that it grows harder and harder to gain a return on the investment it requires to architect, build and maintain these silo systems and solutions.

In fact, according to industry analyst META Group, 70% of today's average IT budget goes into maintaining silo-based infrastructures, leaving only 30% for innovation. META Group forecasts that by 2006, as much as 80% of the IT budget will be consumed by maintenance if enterprise management continues unchanged. And by 2010, if companies stay with a silo-based architecture for IT, more than 95% of the IT budget will be consumed by infrastructure and maintenance expenses, leaving very little funds for innovation.

As systems become larger and more complex, infrastructure must evolve from silos of technology to virtualized resource pools that are automatically or semi-automatically managed. Rather than focusing on supporting discrete applications, used by specific business functions or groups, the infrastructure must be organized to support the needs of multiple groups and business needs across the enterprise, subsequently reducing the effort associated with administrating specific business processes and infrastructures. As the infrastructure becomes more adaptive, the enterprise becomes more responsive to change, or *adaptive*.

The deeper integration of IT with business processes has led to a fundamental change in how organizations view IT infrastructure and the services it provides to business units. IT is used to link separate business processes together within an organization and to link suppliers, partners, and customers together outside the corporate firewall. These many linkages have rapidly led to more complex IT solutions, which impact how companies use and view their IT resources.

The following business challenges are fundamental drivers for how the IT infrastructure landscape is changing providing ground for innovation such as an Adaptive Enterprise (HP Adaptive Enterprise):

Increasingly complex IT solutions. The integration of business processes and IT solutions reaches not only one organization, but reaches also into customer, supplier, and partner environments creating complex and interdependent relationships. Enterprise resource planning, supply chain management, customer relationship management, wireless services, and other technologies have created a very expensive legacy IT infrastructure. Managing change keeping IT aligned with business objectives has become increasingly difficult. These infrastructures must be broken up making them more adaptive in a step-wise evolutionary process.

Variable, highly unpredictable workloads. As suppliers, partners, and customers are exposed to a potentially unbound number of users connected through the Internet and the Web, workloads for applications and backend databases are becoming more unpredictable. With a static datacenter infrastructure, each individual resource or application needs to be over provisioned to meet potential demand. As more resources are exposed to the variability of the Internet and the Web and users outside the organization's firewall, scaling enterprise resources is considerably contributing to the rising cost of managing IT.

Growth in the number and variety of systems to deploy and manage. To meet the increasing demands of both business units within the organization and partners outside the company, deploying hundreds, if not thousands, of horizontally scaled servers has become the dominant way to keep pace with the demand for IT resources. Manual deployment and management tasks, as well as the integration challenges of making such a dispersed and varied IT architecture work, create another challenge to organizations' IT operations.

Tremendous cost of downtime. As the management of business processes becomes dependent on the IT resources, and suppliers and customers become dependent on an organization's IT to conduct business, the cost of downtime rises substantially. In some industries, such as financial services, the dependency between IT resources and downtime has been a widely researched and discussed issue. Always available services is becoming a critical necessity.

High cost of managing IT assets. To manage the complexity, security, and variability of IT systems while providing a high level of availability of IT resources, organizations are spending more on IT operations. As a result, organizations are continuing to turn to external service providers to outsource their infrastructure management. In fact, IT outsourcing is the largest category in the IT services market, with expected growth in the future. However, outsourcing such operations only addresses part of the challenge. The organization must still improve the linkage between IT and business objectives, in the outsourced case also in cooperation with the IT service provider.

Current trends turn into visions of large Enterprise Management solution and service providers such as Hewlett-Packard with its vision of the Adaptive Enterprise (HP Adaptive Enterprise) or IBM with its vision of Autonomic Computing (IBM Autonomic Computing).

7. SUMMARY

This chapter introduced the basic notions of application and infrastructure management for Web services.

Metrics have been presented that can be used for build monitoring models for applications or infrastructure components. Monitoring models provide a systematic foundation for monitoring and archiving data that characterize the behavior and the boundaries of "normal" behavior for a Web service. Monitoring, or more general, management models are also the precursory for tool support and implementing an effective management solution for a Web service.

Enterprise management will evolve from infrastructure management to service management. Services will turn management itself into a service, which is needed to operate (managed) services. Services encapsulating and abstracting diversity found in infrastructure is another step towards managing business in conjunction with supporting IT systems. Business-aware management capabilities can be seen today in management products.

PART III

THE PRACTICE OF WEB SERVICES MANAGEMENT

Chapter 7

INSTRUMENTATION OF WEB SERVICES

1. INTRODUCTION

In order to manage Web services it is first necessary to instrument them. This instrumentation may be invasive or non-invasive in nature. Only when the Web services have been instrumented can operational and management related data be obtained from them.

2. INSTRUMENTATION

Instrumentation is used to collect management data. The management data so collected may be used for debugging/testing, visualization/animation through consoles, or for monitoring and control in management systems. Instrumentation consists of implementation code that has to be included with the managed service (either invasively or non-invasively) in order to collect and export models and raw measurements, to send alarms and generate events.

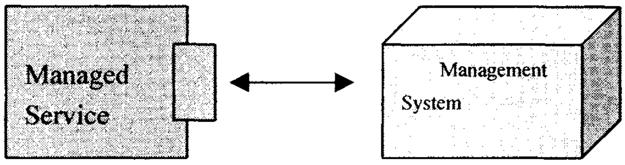
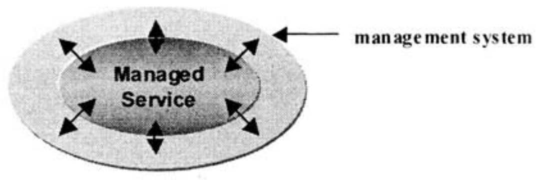


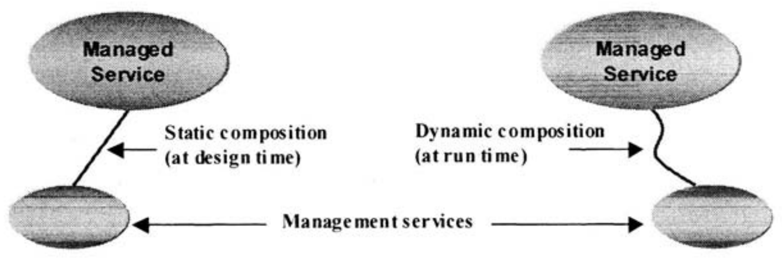
Figure 59: Basic abstraction of managed service and a managed system.

Current enterprise and e-business management systems are tightly coupled with the managed application and are usually constrained by the enterprise and management domain boundaries (e.g., firewalls). However, if the management systems have to manage multiple services that are distributed either within or across multiple enterprises, a decoupled way of obtaining and distributing management information has to be decided upon. There are multiple integration approaches that are possible. The first integration approach usually employs a proprietary set of interfaces and instrumentation points to collect the required measurements and apply the required commands. It also assumes full access to the required information and full control over the service. In the second and third options, the managed service exposes a set of well-defined interfaces that can be exploited by the management system in order to perform its management tasks. The composition could be governed by a pre-negotiated (option b) or just-in-time-negotiated (option c) contract. In the second and third options, the management system could itself be implemented as an outsourced service.

The interfaces, which tie the management system together with the managed service, exchange information using Internet protocols and standards wherever applicable. This facilitates implementing the management system to collect measurements from the various cross-enterprise sources and also to control applications and services that are provided by external suppliers.



(a) Tightly coupled integration


Managed Service
Static composition (at design time)
Dynamic composition (at run time)
Managed Service
Management services

(b) Static integration **(c) Dynamic integration**

Figure 60: Managed service interaction.

Instrumentation may be intrusive or non-intrusive in nature. Non-intrusive instrumentation effectively means not modifying the application in any way for instrumentation. This usually entails instrumentation of the environment itself so that every time a service is instantiated or does something the management data is collected transparently. Non-intrusive instrumentation has been tried in distributed systems like CORBA [Silvano 1995] earlier. Intrusive instrumentation means inserting instrumentation API calls into the application thus modifying it. Insertion of API calls is undertaken to delineate units of works. In this chapter, we examine different instrumentation techniques and present some of the standard approaches.

2.1 Management Information Exposed Through Instrumentation

The management information that the managed service may provide to the management system involves information about:

- **Models:** A set of models representing the structure and configuration of the service, state transitions within the service, and workflow of the supported transactions.
- **Measurements:** Real-time measurements whenever transactions are started and stopped at service and sub-service boundaries, availability heartbeats, contract and offer details, lifecycle state changes, etc.
- **Events:** Asynchronous messages sent whenever errors or important events occur within the service. These could be alarms, traps that the managed service generates.
- **Configuration:** A set of parameters representing current service and environment configuration such as maximum number of users, authentication requirements, machine type, etc.
- **Control points:** Interfaces to change the lifecycle state of the service (e.g., start, stop, suspend, etc.), to improve the availability (e.g., load-balancing, replication, migration, etc.), or to fix certain problems (e.g., rollback, re-index, restart, etc.).

2.2 Manageability and Instrumentation Requirements for Web Services

Web services are federated in nature as they interact across management domains and enterprise networks. Their implementations can be vastly different in nature. Some of the common technologies for implementing

Web services are J2EE, .Net. Figure 61 shows a fictional example of interacting Web services. Employees in a company (workhard.com) use their internal Web service (supplies.workhard.com) to order day-to-day supplies and stationery. The internal supplies Web service in turn uses a supplies marketplace (supplies.marketplace.com) to find the best deals. The marketplace requests bids from two supplies companies (stationery.com and officesupplies.com) to fulfill orders. Both the suppliers use a shipping service (shipme.com) for shipping the ordered products directly to the consumer.

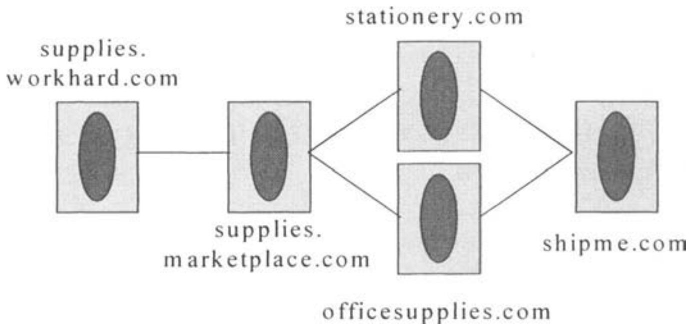
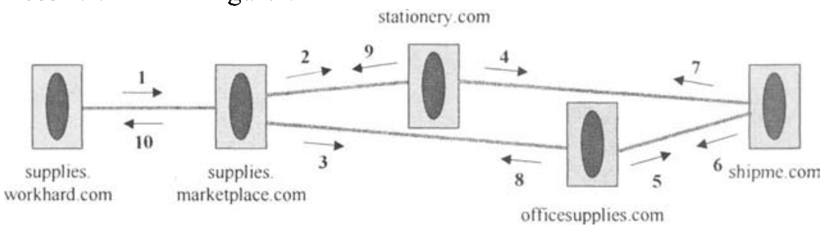


Figure 61: Instrumentation of an example Web service.

When two Web services connect to each other, they have to agree on a document exchange protocol and the appropriate document formats. From then on they can interoperate with each other exchanging documents. SOAP defines a common layer for document exchange. Services can define their own service-specific content on the top of SOAP. The execution of a single business transaction can involve multiple messages being exchanged between Web services. For example, a purchase order transaction that begins when an employee orders supplies and ends when he or she receives a confirmation could result in ten messages being exchanged between various services as shown in Figure 62.



1.	purchase order	6.	shipping confirmation
2.	part of purchase order	7.	shipping confirmation
3.	the other part of the purchase order	8.	order confirmation
4.	shipping request	9.	order confirmation
5.	shipping request	10.	purchase order confirmation

Figure 62: SOAP messages exchanged between Web services.

One such message encapsulated in SOAP is shown in Figure 63. Note that every SOAP message has a clearly defined header (SOAP-ENV:Header) and body (SOAP-ENV:Body). The service-specified content is enclosed within the body of a SOAP message. Headers are typically used to represent meta-information about messages (e.g., message identifier).

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Header>
  <Id>1</Id>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
  <PurchaseOrder>
   <Item count = 100>Postit stick notes</Item>
   <Item count = 200>Stapler</Item>
  </PurchaseOrder>
 </SOAP-7ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 63: SOAP message used in the example.

A typical web-service would get an http request (may contain an XML document) from the parent web-service (if it itself is not the root service) or an ultimate consumer. The request is routed from one of the web server in the web server farm to one of the application modules in the application server farm. Thereafter, business logic is applied to the request and a new request can be directly sent to sub web-services through an http request at this level. The business logic may be implemented in the application server itself or may be a call to a legacy software that implements it. The response from the sub web-service is expected at the listener (attached to the web server farm) that is waiting to receive documents.

Once the corresponding sub web-service responds with a document the response is sent to the business logic, which in turn sends a response to the parent web-service. The communication pattern can vary depending on the implementation. For example, the application logic can send an immediate response to the parent web-service before receiving response from the sub web-service. The final response may be sent later. Also, the final response can be sent directly to the parent web-service by the sub web-service, instead of being routed through the business logic. In addition the format of communication is not symmetric in the web-services world.

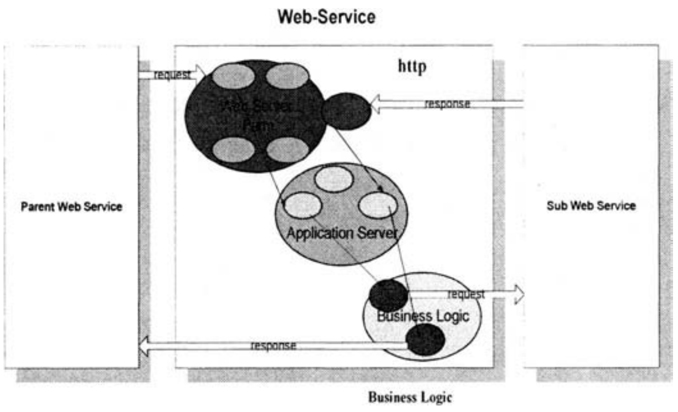


Figure 64: Web service interactions.

A web-service request may not always have a matching response. In some cases, all the participating services are like peers, in which case there is no notion of a request or a response. Some of the message flow patterns that result from this asynchrony are shown in Figure 65. The first example shows a single request resulting in multiple responses. The second example shows a broker-scenario, in which a request is sent to a broker but responses are received directly from a set of suppliers.

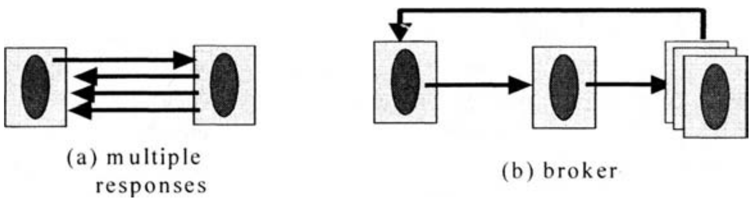


Figure 65: Asynchronous message patterns between Web services.

Management of Web services is a challenging task because of their heterogeneity, asynchrony, and federation. Motivations for end-to-end Web service management arise from the perspective of both clients and service providers. Clients are interested in tracking their requests and in understanding bottlenecks or causes for failure of their requests. Measurements taken at various points along the execution of a transaction are very helpful in analyzing end-to-end quality of the transaction. Service providers that are using other services to provide composite services would like to know how the component services are behaving. By studying and observing their behavior a composite Web service would be able to optimize itself by either changing its component sub services or by instructing the existing component sub services to improve performance.

2.3 Standards in Instrumentation and Manageability

Instrumentation and manageability has been looked into under various domains. Some standard exist in these domain. We will discuss Application Response Measurement, SNMP, and Java Management Extensions (JMX) in this section.

2.3.1 SNMP

The Simple Network Management Protocol (SNMP) is the protocol governing network management and the monitoring of network devices and their functions [Black 1994]. SNMP is described formally in the Internet Engineering Task Force (IETF) Request for Comment [RFC 2571, RFC 3411] and in a number of other related RFC.

SNMP defines two primary elements: a manager and agents. The manager is a process usually connected to a console through which the operator performs management functions on elements in the managed domain. Agents are processes that interface to the actual elements being managed. Bridges, Hubs, Routers or network servers are examples of managed devices that contain managed objects. In addition, applications can have SNMP agents. Managed elements maintain status information about themselves in a so-called Management Information Base (MIB). SNMP is the protocol performed between managers and agents accessing MIB data [RFC 3411].

SNMP agents provide the instrumentation in a managed domain for delivering status data to a management system performing management processes. SNMP agents can answer inquiries for MIB data received from managers. This is typically used for (polling, which means server-initiated) monitoring. A SNMP agent may also initiate sending notifications (events) to the manager through Traps (a push mechanism).

SNMP agents can also alter MIB data on request of the manager. Hardware devices and application components may be sensitive to alterations of MIB data. These alterations are done through set operations. Altering MIB data is used to actuate control operations though the agent.

As specified in Internet Requests for Comments (RFC) and other documents, a network management system comprises:

Network elements -- Sometimes called *managed devices*, network elements are hardware devices such as computers, routers, and terminal servers that are connected to networks.

Agents -- Agents are software modules that reside in network elements. They collect and store management information such as the number of error packets received by a network element.

Managed object -- A managed object is a characteristic of something that can be managed. For example, a list of currently active TCP circuits in a particular host computer is a managed object. Managed objects differ from variables, which are particular object instances. Using our example, an object instance is a single active TCP circuit in a particular host computer. Managed objects can be scalar (defining a single object instance) or tabular (defining multiple, related instances).

Management information base (MIB) -- A MIB is a collection of managed objects residing in a virtual information store. Collections of related managed objects are defined in specific MIB modules.

Syntax notation -- A syntax notation is a language used to describe a MIB for managed objects in a machine-independent format. Consistent use of a syntax notation allows different types of computers to share information. Internet management systems use a subset of the International Organization for Standardization's (ISO's) *Open System Interconnection* (OSI) *Abstract Syntax Notation 1* (ASN.1) to define both the packets exchanged by the management protocol and the objects that are to be managed.

Structure of Management Information (SMI) -- The SMI defines the rules for describing management information. The SMI is defined using ASN.1.

Network Management Stations (NMS) -- Sometimes called *consoles*, these devices execute management applications that monitor and control network elements. Physically, NMS are usually engineering workstation-caliber computers with fast CPUs, color displays, substantial memory, and abundant disk space. At least one NMS must be present in each managed environment.

Parties -- Newly defined in SNMPv2, a party is a logical SNMPv2 entity that can initiate or receive SNMPv2 communication. Each SNMPv2 party comprises a single, unique party identity, a logical network location, a single authentication protocol, and a single privacy protocol. SNMPv2 messages are communicated between two parties. An SNMPv2 entity can define multiple parties, each with different parameters. For example, different parties can use different authentication and/or privacy protocols

Management protocol -- A management protocol is used to convey management information between agents and NMS. SNMP is the Internet community's de facto standard management protocol.

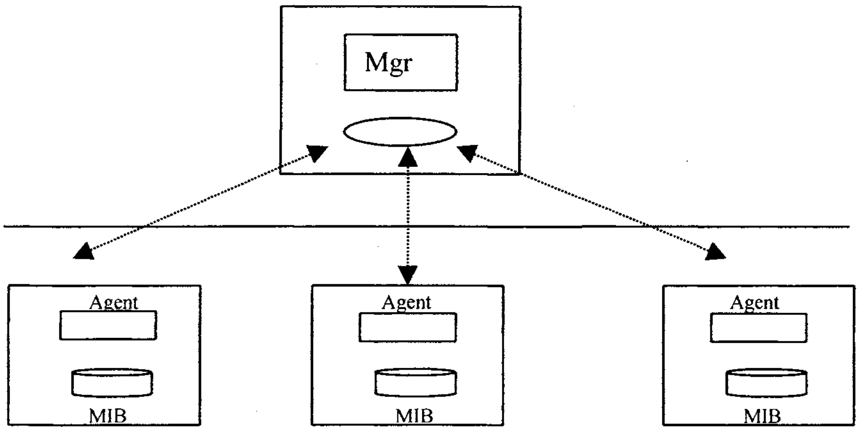


Figure 66: Management with MIB-based management agent.

The Simple Network Management protocol (SNMP) is the de-facto standard for network and system management (Stalling 1999). In SNMP, a management system queries the SNMP agents that implement a Management Information Base (MIB) through get/set operations. The SNMP Agent can send traps to the Management System through the transport layer. Attempts have been made to extend the SNMP to application management by defining relevant MIB. System Application MIB and Application Management MIB are efforts in that direction.

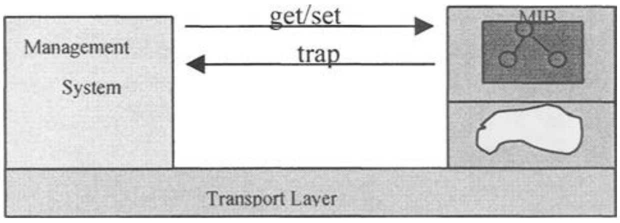


Figure 67: Schema for Simple Network Management Protocol (SNMP).

2.3.2 System Application MIB

The Internet-standard Systems Application MIB, RFC 2287, provides mechanisms to obtain information related to applications without instrumentation. The Systems Applications MIB yields a list of installed applications on each system, the elements that make up each application, and the activity information about each element. The objects in the MIB are arranged in the following groups:

- System Application Installed Group
  - SysApplInstalledPkgTable
  - sysApplInstallElmtTable
- System Application Run Group
  - SysApplRunTable
  - SysApplPastRunTable
  - SysApplElntRunTable
  - SysApplElmtPastRunTable
  - Scalars fpr restricting table sizes
- System Application Map Group
  - SysApplMapTable

The System Application Installed group consists of two tables. These tables are used to determine which applications are installed on the system and what their constituent components are. The first table sysApplInstallPkgTable details the application packages installed on a particular host. The second table the sysApplInstallElmtTable provides information regarding the executables and non-executable files or elements that compose an application.

The group models activities that have been invoked or are running. The tables describe information about the currently executing and past run applications and elements.

The system Application Map group contains a single table, the sysApplMapTable, whose intent is to provide a backwards mapping for determining the invoked application, installed element, and installed application package given a known process identification number.

2.3.3 Application Management MIB

This Application Management MIB specification includes additional attributes that will typically require instrumentation within the managed resource. The sysApplRunElmtIndex is the key connection between these two MIB; it is essential that implementations of this MIB and of the system applications MIB running concurrently on a given platform employ a consistent policy for assigning this value to identify running application elements. The application management MIB:

- Requires cooperation from instrumentation.
- Extends sysAppl MIB.
- Provides service level view.
- Provides info on open channels, transactions, connections.
- Process level status information and control.
- Capabilities to suspend, resume, reconfigure and terminate processes.

Service-level tables

The service-level tables permit the identification of one or more instances of named services on a system, and the association of running application elements to these services. In deciding what should or should not be considered a service, the following factors merit consideration:

- Is there an identifiable set of resources associated with providing this service?
- Is there a reasonably long-lived server or client process?

Transaction related tables

The transaction stream summary table contains per-stream summaries of transaction statistics. Transaction flow statistics table contains statistics broken into both transmitting and receiving counts for requests and responses on each stream. The transaction kind statistics table contains summary information organized by direction, request/response, and transaction kind for each stream.

Former tables

Former channel group, channel, connection, file tables provide information about former groups of channels, channels that have been closed, formerly open channels that were connection, connection-specific historical information

Transaction history tables

Tables provide per-transaction-kind breakdowns for channels carrying transaction-structured flows.

This table provides information for a running application element. Indexed by the sysApplElmtRunIndex, an entry in this table reports useful information on that running element's resource usage. Entries in this table contain:

- current heap usage for this running application element.
- current number of open network connections for this running application element.
- the most recent error status message issued by this running application element.

Running Application element control table:

Variables in this table include:

- suspend/resume control,
- reconfiguration request control, and
- termination request control.

2.4 Application Response Management (ARM)

Transaction measurement is an important aspect of Web service and application management. The ARM API (ARM) is a simple API that applications can use to pass vital information about their transactions. A transaction is loosely defined as any unit of work within the application logic that can be marked with a "start" and "stop". Examples of transactions include:

1. body of a function or method (start at the beginning of the function and stop at the end of the function)
2. a critical section of code (start at the beginning of the critical section and stop and the end of the critical section)
3. the lifetime of an object (start in the constructor of the object and stop in the destructor)
4. a "purchase order" transaction (start when the user initiates a purchase order and stop when the user is given a confirmation number)
5. A "book buying" experience (start when the user logins in to a web site and stop when the book is shipped to or received by the user).

The exact number and nature of transactions are left to the discretion of the application developer. From the above examples, it can be noted that transactions can represent a contiguous piece of code (e.g., 1, 2) or functionality spread across several methods or components (e.g., 3, 4, 5). They can be used to represent system/application logic (e.g., 1, 2, 3) or business logic (e.g., 4, 5). Further, transactions can be short-term (execution completes within seconds or minutes) or long-term (execution takes days or months to complete). However, in all these cases, there should be a well-defined "start" and "stop" that could be inserted into the application logic to mark the boundaries of the transaction.

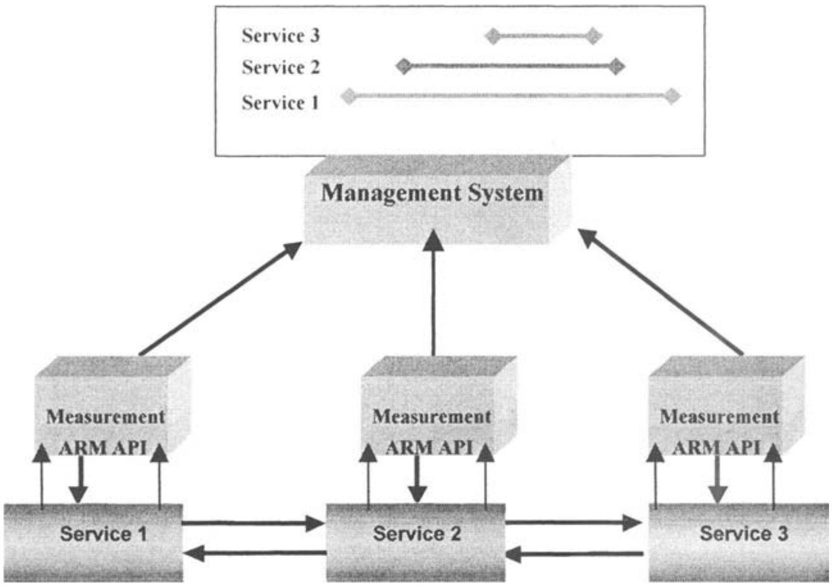


Figure 68: Service correlation with measurements in ARM.

ARM provides APIs that can be used to delimit sections of application code base to monitor time spent in those sections. These APIs correspond to starting and stopping of code sections and assigning handles to them for manipulation by the local ARM Agent. In addition to measuring transactions in isolation, it is often beneficial to relate transactions with each other. The parent-child relationship ties a parent transaction to its sub-transactions. This can be used to measure the parent transaction as a whole and to relate that measure with the measures of each of the sub-transactions. A common use of this is to define a parent transaction as the client-initiated transaction and to define sub-transactions for portions of the parent transaction that execute within server components (Figure 68).

For instance, a purchase order transaction initiated in a browser in an e-commerce application could be defined as the parent transaction while the resulting execution in a web server could be defined as the sub-transaction. The parent-child relationship is established by passing a correlator from the parent transaction to the child transaction. A correlator is a unique identifier that can be used to distinguish one instance of a transaction from another. In the example above, the browser passes a correlator to the web server.

As described in the previous section, web-service undertakes conversations with other web-services, which involves multiple interactions (exchange of documents). These web-services are federated and distributed with varied implementations. In addition, their interactions are asymmetric and asynchronous in nature. The ARM API provides the capability of furnishing of application data in the form of data buffers that are maintained by the ARM library.

```
arm_init ("Application Name", "User Name");
 arm_getid("Transaction A");
 arm_getid("Transaction B");
 arm_getid("Transaction C");

   loop until the program ends
   arm_start(A)
      do some work
       arm_start(B)
         do some work
       arm_stop(B, status)

           arm_start(C)
         loopuntiltransaction ends
         do some work
       arm_update (C)
       end loop
       arm_stop(C, status)
     arm_stop(A, status)
 arm_end
```

Figure 69: Example ARM document.

In ARM, the correlation data is sent to a central correlation application that undertakes the task of linking up the transactions with their component transactions. Since ARM assumes a centralized correlation application, a concern with ARM thus is that tracing any application flows may overwhelm the correlation application and/or the network with the volume of data collected at the agent, and being sent to the correlation application. ARM is also ill suited to the asynchronous and asymmetric interaction model that web-services operate in. ARM is also designed to maintain transaction level data while web-services need conversation monitoring. The ability provided by ARM to relate a sub-transaction to a single parent is also inadequate. In Web services, a single piece of functionality may participate in two or more logical transactions at the same time. This is the n-D correlation problem. Certain extensions may be provided for extending ARM to Web services [Sahai 2001].

Also as in ARM, it is assumed that all the data is sent to a central correlation application, it may not be practical in scenarios where the Web services belong to multiple enterprises. In such a case the management data has to reach the management system across enterprises. Also since the documents exchanged between Web services are all different, passing the correlator for a conversation is a difficult problem. These entail certain changes to the basic premise of ARM.

2.5 Application Response Time Measurement (ART)

Application Response Time Management Information Base (ART MIB) from NetScout extends Remote Monitoring 2 (RMON2) which provides a mechanism of classifying application traffic. The application flows can be identified by the RAT MIB. Initial set of TCP based applications that are supported includes SAP R/3, Baan, PeopleSoft, e-mail, web, database and file-transfer applications. The ART MIB also provides capability for SNMP.

2.6 Windows Management Instrumentation (WMI)

Windows Management Instrumentation (formerly known as WBEM) from Microsoft provides the capability to obtain real time data from hardware components that are based on Windows Driver Model. It enables effective management of PC and server systems in enterprises. WMI complements SNMP and uses CIM. WMI provides a consistent and richly descriptive model of the configuration, status, and operational aspects of Windows operating system. It provides the facility of publishing kernel instrumentation, configuring device settings, providing kernel-side event notification, publishing custom data, allowing administrators to set data security and accessing instrumentation by way of WMI. The WMI Service Development Kit (SDK) is also available. WMI is designed for programmers who use C/C++, the MS Visual Basic Application, or a scripting language that handles MS ActiveX objects. Some amount of COM familiarity is desirable.

2.7 Java Management Extensions (JMX)

Java Management Extensions (JMX) is targeted at resources that are either implemented in Java or are wrapped by Java classes. Such resources that intend to become manageable may use JMX.

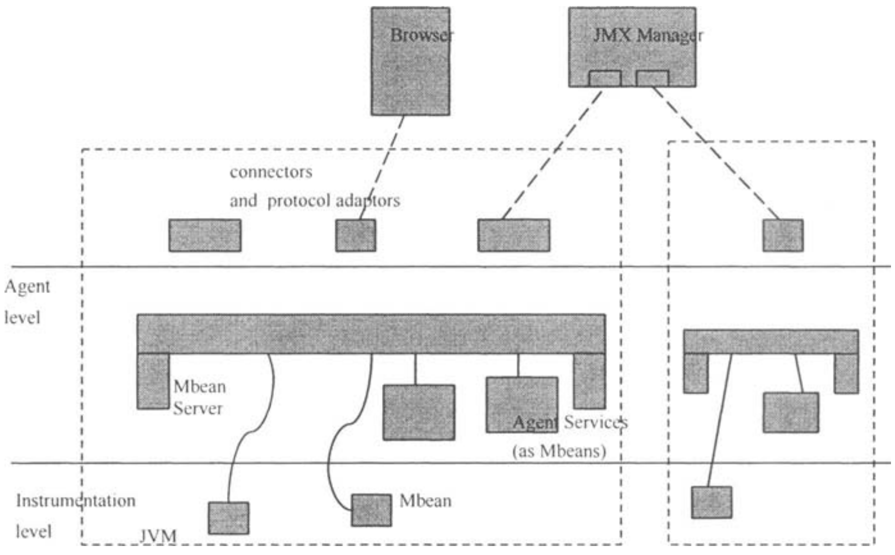


Figure 70: JMX management extensions.

It provides supports for management at the instrumentation level, at the agent level and at the distributed services level as shown in Figure 70. The Managed Bean (Mbean) is used to represent the resources being managed. A Mbean server may be used at the agent level to manage a set of Mbeans. The Mbean Servers may be located on multiple hosts and on different Java Virtual Machines (JVM). The Agent services may also be represented through Mbeans. JMX Manager(s) or Browser may access the Mbean servers through connectors and protocol adaptors

2.7.1 Instrumentation Level

The instrumentation level specification provides means for implementing JMX manageable resources. A JMX manageable resource may be an application, a service implementation, a user, a device etc. JMX manageable resources are automatically manageable by JMX Agents. An Mbean (Managed Bean) is a java object that implements the management interface.

Managed Bean (Mbeans) have the following capabilities for making resources manageable,

- Constructors that initialize the Mbean.
- Valued attributes that can be accessed.
- Operations that can be invoked.
- Notifications that can be set .

There are four different types of Mbeans: Standard, Dynamic, Open, Model. The notification model is defined in JMX so that the Mbeans and the agents may use the notification mechanisms. The specification describes the notification objects, the broadcaster, and the listener interfaces that the objects must implement to use notification mechanisms.

2.7.2 Agent Level

Management Agents monitor, and control JMX manageable resources and expose their information to management systems. The JMX Agent comprises of a JMX Server and set of services. The JMX may be embedded into the same virtual machine as the one being used by JMX manageable resources or may be a stand-alone mediator if the manageable resources are non-java or are located on a different machine.

- Mbean server: enables registration of Mbeans and access to operations of the Mbean.
- Services: The services that the Mbean server provides are related to Notification, Querying, Monitoring, Relationships.

2.7.3 Distributed Services Level

The distributed services layer enables the management applications to interface with the management agents.

- Provide interface for management systems to interact with JMX agents
- Expose management view and consolidate views
- Provide security

2.8 Log File Analysis

The Web services may face hundreds of millions of hits per day. It may support batch import of multiple Web server logs. Due to high data volumes of user hits, it can be necessary that Web Server logs must be rotated putting one log into the hot spot where logged data arrive at high speed, and allowing the other logs to be processed. Web Server log rotation may happen once a day or more frequently. To track visits across multiple, distributed servers, the Web log reader component sews together the information into a chronologically ordered sequence in order to have a full picture of users and visits across servers.

Figure 71 shows the principle of Web services Log analysis. While clients (users in the user-to-service scenario or other services in the service-to-service scenario) are interacting with a Web service, the Web server which is the first tier processing a client request, reports log information about the request in the Web server log file in raw format. The following tier represented by the application server typically extracts more application-oriented information from the request and logs this information in its log file.

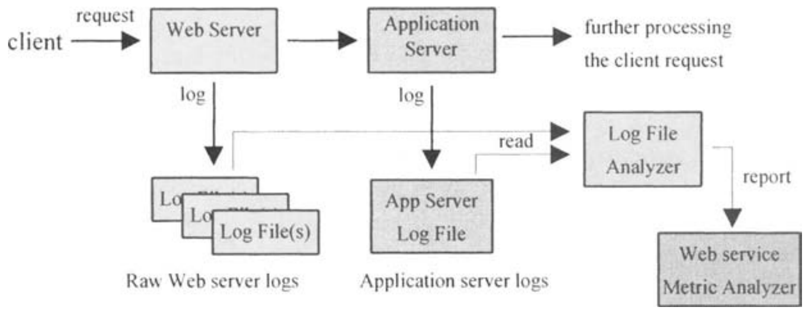


Figure 71: Web services log file analysis.

The Log File Analyzer periodically reads log files, extracts relevant information, which it reports to the Web services Metric Analyzer.

The Web log reader must be flexible in order to access logs automatically across the network and support different log naming conventions. It must also be able to differentiate identically named logs across servers, understand all standard log formats, and in adapt easily to customized formats. While the Web log read process is fully automatable, it can also be run manually to deal with logs that have been delayed or to import historical Web log data.

Given the diverse environments of possible installations, an important Web log reader capability is the automatic detection of log file formats for the leading Web servers. Data from these different sources can be seamlessly sewn together and imported simultaneously. And, as an option, Web hits can be sampled for a given set of users, allowing reduction of the overall data stream while maintaining the validity of paths and visits for the reduced set of Web log information.

2.9 Network Packet Sniffing

An alternative or supplement to importing Web server logs is network packet sniffing. Network packet sniffing is a known method in network management. Network sniffers tap into networks and gather traces of network activity. Network sniffers can trace network activity at different network layers:

- SOAP (Simple Object Access Protocol) layer,
- HTTP (Hypertext Transport Protocol) layer,
- TCP (Transmission Control Protocol) layer,
- IP (Internet Protocol) layer, and
- Physical link protocol layer.

The SOAP (Simple Object Access Protocol) layer transfers XML messages using the SOAP protocol. SOAP specifically is used for service-to-service interactions. SOAP in most cases is based on HTTP. Tracing SOAP flows allows identifying which service interacted with which other service by referring to the SOAP header.

The HTTP (Hypertext Transport Protocol) layer is used as basic transport protocol in the Web. HTTP is user for service-to-service interaction underlying SOAP. HTTP is also used (and originated) in the user-to-service area for serving Web pages in HTML to Web browsers. HTTP traces obtained by network sniffers are helpful to identify which interactions at HTTP level occurred.

The TCP (Transmission Control Protocol) layer underlies HTTP and provides reliable communication between two endpoints (e.g. a server and a browser) with flow control capability. TCP network sniffing is primarily used in network analyzing tools for identifying network problems.

The IP (Internet Protocol) layer underlies TCP and provides the overall connectivity in the various kinds of Internet (public Internet, intra- and extra-nets). IP provides packet-switched network routing based on IP addresses such as 15.9.34.78. IP network sniffing is also primarily used for analyzing network problems and root cause analysis in networks.

The Physical link protocol layer provides physical linkage of devices, network cards, etc. to switches and routers in a network.

Usually located on a dedicated machine on a Web server's network segment, a sniffer can capture the application data contained in the TCP/IP packets streaming past it and output the results in a standard log file format. Because of their location, sniffers can detect low-level network events such as the network disconnect that is sent when a user clicks a link before a page has completely loaded. And because a sniffer monitors the network directly, it simplifies log file management. For example, it eliminates the need for sewing multiple logs when the servers are all on the same network, since this is accomplished by the multi-access nature of the network itself.

Figure 72 illustrates how network packet sniffers can be applied at different layers of the network stack. Typically, sniffing at IP layer only provides raw data, which is typically only helpful for network management. Sniffing network connections at TCP layer provides information about duration of a connection to clients mediated through one TCP connected. HTTP sniffing already provides information about URLs used in the HTTP header, or eventual attachments appended in the HTTP body. Sniffing at higher layers such as the SOAP layer provides detailed insight in exchanged messages between services.

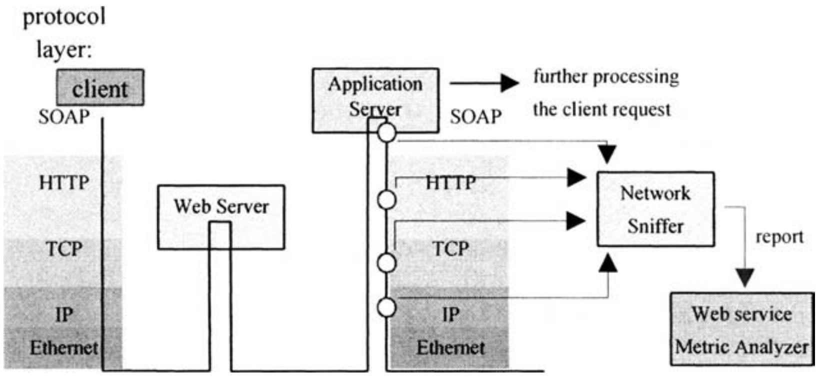


Figure 72: Network packet sniffing at different protocol layers.

The major advantage of network sniffing is that no instrumentation in Web servers or application servers is necessary. However, since the stack of protocols needs to be reconstructed in sniffers, realization of higher-layer network sniffing may become complicated.

Furthermore, in a secure Web environment using SSL encryption, a sniffer cannot decode the application layer of the packet without the decryption key. And sniffers may not be appropriate for hosted environments where other clients may share a LAN but do not wish their data sniffed. Virtual LAN technology can help preventing other client's data being sniffed without their knowledge.

The sniffer can be deployed quickly where desired. It is highly scalable (up to the limits of the Network Interface Card of the machine on which it runs) and writes information out in a standard W3C log format or in an extended log format.

The higher layer network sniffing is applied, the closer obtained data is to business interactions of interacting Web services. Network sniffing at lower layers (physical link protocol, IP) often is of less relevance for business-aspects of Web services metric analysis. It is more relevant for network management.

2.10 Web Server Plug-ins

The third data collection option is to use a Web server plug-in. Even if a site is using a dynamic content engine or an application server, it still needs a Web server. Netscape, Microsoft Internet Information Server (IIS) and Apache account for over 90% of all Web servers. Each of these platforms provides an API or interface in order to extend and modify the information that is logged in the log file. A Web service Metric Analyzer typically has a number of available plug-ins that can be deployed at the used Web server platform.

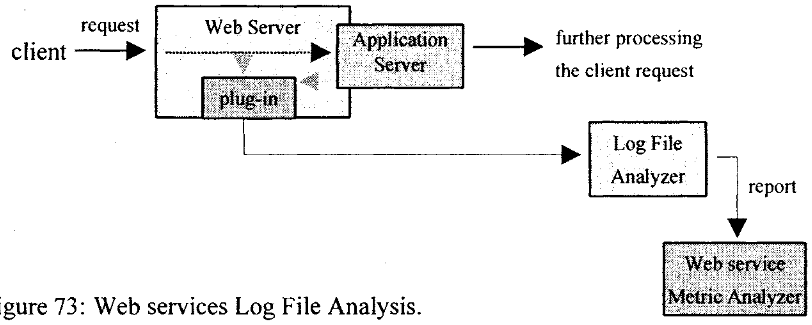


Figure 73: Web services Log File Analysis.

Figure 73 shows how a Web server plug-in is used to extract relevant information form the request routed through the Web server. Web server plug ins can be closely linked with application servers depending on the implementation. Web server plug-ins can be deployed to implement sniffing at HTTP or higher protocol layers.

Web server plug-ins are also useful for tracing sessions a client currently uses to interact with the Web side. Typical Web Server plug-in modules deal with cookies, POST data, and the performance of requests.

A cookie is a piece of information a Web site can leave at the machine from where a client accessed the Web site. This information can be retrieved at later visits and used for various purposes such as:

- re-establish a client's session,
- re-instantiate a client's shopping cart,
- remember time, date and path of first visit.

Cookies in general are a valuable information source. Through reading cookies, information about other (competitive) sites can be extracted without the client becoming aware of it. This information can then be used to customize Web presentations to clients (e.g. highlighting in what regard an offer is better than the offer of the competition).

A Web server plug-in can also track existing visits or user cookies created by other applications. The cookies can then be written to the Web server log file for later analysis. Logging POST data through a plug-in is particularly useful for e-commerce sites and other sites with forms that use POST data. Finally, a server performance plug-in can be used to measure and log how long requests take on a site concluding to an overall user experience visiting a Web site.

- non-standard log file formats,
- treatment of anchors (links within a page),
- treatment of request redirection,
- resource exclusions (such as images),
- reconstruction of visits that span log file boundaries over time,
- exclusion of robots and spiders,
- click-path sampling options by percentage,
- page title or site name lookup,
- use of authenticated user names, or
- use of external referrals to begin a new visit.

With these rules, the basis is provided for gathering and processing meaningful and valuable metrics.

2.11 SOAP Instrumentation

It is necessary to interfere with message exchanges among web services in order to collect information about the interactions with business partners. An acceptable solution should not impose any modifications or limitations on existing web services. SOAP is the standard for web service interactions. A small proxy component that tries to capture incoming and outgoing messages, and records data about the message exchanges, then forwards the captured messages to the actual recipients could be another approach for gathering management data about web service interactions. The most popular implementations of SOAP toolkits share common components, called routers. SOAP routers receive the messages from SOAP clients and submit them to the receivers. SOAP toolkit encrypts the message at the sender site, and decrypts it only when it reaches the receiver's site. A proxy can be easily attached to SOAP toolkit routers with minor modifications to the toolkit. This is the most appropriate way to automatically attach a proxy in order to capture SOAP messages and collect information from those messages. It does not require any modifications to existing web services, and does not require re-compilation of existing SOAP toolkit installations. We used this approach for collecting data from SOAP message exchanges among web services.

In order to correlate individual message exchanges with each other, a notion of Global Flow (GF) can be used. A Globally Unique Identifier (GUID) is used for keeping track of a GF. Every time the proxy component catches a message that is exchanged between web services, it first checks whether a GUID exists. If a GUID does not exist in the message, the proxy inserts a GUID into SOAP header of the message. All web services and other software components propagate the GUID in their communications. Consequently, the proxy components that are attached to SOAP toolkits at business partner sites can easily figure out which SOAP message is sent in the context of which previous messages. These correlators may also be passed as correlator sets as defined in BPEL4WS.

2.12 Handling Dynamic Content

Many Web sites provide dynamic content by constructing and returning pages on the fly. A content management system may take into account the users profile or recent activity. Details about the actual served content resources are often encoded as query information attached to the URL. This enables deeper analysis of the actual information delivered to the user.

An example of this scenario is a Microsoft Active Server Page (ASP) environment, where every URL ends with index.asp. Reporting that every user accessed this URL is not particularly interesting. The content that is delivered, and hence the interesting analysis, is based on the query string attached to the URL. Other dynamic content servers differ in the details of the URL and query string syntax, but all of them are handled similarly. Maintaining aggregate counts based on query strings completes the reported information and makes query string analysis highly efficient.

SUMMARY

While instrumenting Web services, it is useful to identify the important business transactions that have to be measured. It is also important to identify which transactions are created in the context of another transaction. This is important to identify parent-child relationships between transactions. The parent child relationship identification is necessary to trace the end-to-end flow of transactions through distributed client/systems.

It is also important to keep in mind the target audience for the instrumentation data and the corresponding performance degradation that excessive instrumentation may result into. Certain standards like ARM, SNMP, and JMX exist for instrumentation/manageability of applications. The instrumentation data so obtained may be aggregated and used for calculating higher-level metrics. These higher-level metrics may form the basis of Service-Level Agreements (SLA) and may populate the business-level views exposed to business managers.

Chapter 8

MANAGING COMPOSITE WEB SERVICES

1. INTRODUCTION

Web service to Web service Management is important, whether these Web services are all inside the enterprise or in different enterprises. Web service to web service communication could mean a single-step transaction (e.g., a login to a book-selling service), a sequence of related transactions (e.g., logging in, adding books to shopping cart, and checking out), or even business-level processes perceived by the client that may involve manual steps (e.g., the process from ordering books to final delivery of the books). For every step in an interaction, one of the two services initiates the request and the other executes the request. So they execute the traditional roles of consumer and producer. These Web services can take any or both of these roles depending on their interactions with other Web services. When Web services undertake dual roles it leads to Web service composition.

We refer to the service that initiates the request as the *consumer service* and the one that executes the request as the *provider service*. The role of a service could change over the course of an interaction.

From a consumer's perspective, *a provider service is manageable* if the latter offers sufficient visibility and control over itself and over the interactions it executes. For example, a provider that provides information about the progress of a consumer's ongoing interactions or an ability to escalate their speed is more manageable than a provider that does not. From a provider's perspective, *a consumer service is manageable* if it can offer enough information about its service usage back to the provider. Manageability interfaces capture the functionality that should be offered by providers and consumers to each other in order to be manageable.

2. WEB SERVICE COMPOSITION

A consumer service can also afford to choose its provider services. The choice is usually based on the consumer service requirements that can be classified into quality of service and service-specific requirements. This process can be generalized to create Web service optimization logic. The logic can be executed on the management data dynamically obtained from the set of provider services to choose the right set of provider services. This process is similar to a tender process in which a set of providers submit tenders for a job and the consumer makes a qualified decision to choose the right party to do the job.

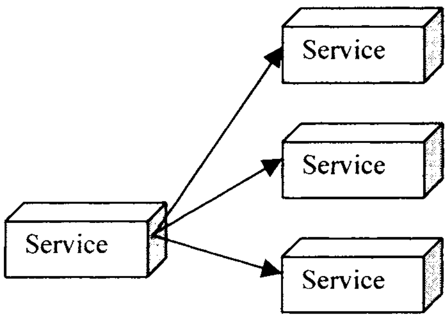


Figure 74: Service relationships.

A service will choose its partners carefully. The decision of a Web service to choose its partners would be based on certain requirements. This optimization process can be generalized to an extent in spite of the fact that it will be guided by the business logic. A service will typically look for its partners with the best performance, availability, reliability, and/or certain business-specific logic (e.g., minimum cost, shortest distance, etc).

A set of requirements can be drawn up:

- Min (response times),

- Max (uptimes),
- Min (failure rates),
- Max (service credibility),
- Min (cost involved).

Depending on the weights attached to the above parameters optimization logic can be created by the Web service for its optimization. A consumer that is interested in performance and cost will associate them higher weights than say availability and reliability. A Web service may not attach too much significance to the rating and may be willing to choose a service with average ratings. This could be a global optimum problem on multiple dimensions.

A set of data structures can be defined. Min, Max, Average, Medium, and Random functions operating on these data structures have to be defined as well. A service optimization decision would thus involve finding the global optimum over multiple dimensions taking into account the weights associated with the parameters like reliability, availability, performance, cost, etc.

3. MANAGING WEB SERVICE TO WEB SERVICE INTERACTIONS

Before Web services can however start interacting with each other a set of requirements have to be fulfilled. These requirements exist in terns of the following:

Web services will execute transactions as a result of these interactions. These transactions have to be of all-or-nothing kind i.e they have to be atomic.

The messages exchanged between Web services have to be sent reliably and may have to be resent if they fail to arrive at the other Web service

These operations have to be secure. Security plays an important role in Web services especially if critical functions have to be carried out through Web services.

3.1 Web Service Transactionality

While Web service Description Language describes the operational interfaces for the Web service, it is important to ensure transactionality for the operations executed by one Web service with another. Especially when web services are used for operations that need to be transactional, it is important to ensure that these operations are committed or rolled back properly. The transactionality for Web service interactions ensures that the transactions are atomic. Usually a coordinator and a coordination protocol is required that agrees to commit an outcome of a transaction. The coordinator determines if there was any failure by asking participants to vote. If the participants all vote saying that they successfully finished a transaction, the coordinator commits all actions taken. If a participant decides that it needs to abort or no response comes back from a participant, the coordinator aborts all transactions. A commit makes the transaction persistent and visible to other transactions. By imposing atomicity, failure and recovery semantics may be applied to transactions. The participating Web service parties can mutually agree on outcomes of transactions.

The 2PC (two-phase commit) protocol is a Coordination protocol that defines how multiple participants reach agreement on the outcome of an atomic transaction. The state diagram in Figure 75 specifies the behavior of the two-way protocol as the exchange of messages between a coordinator and one of its participants.

The state reflects what both sides know of their relationship. Details such as resending of messages or the exchange of error messages due to protocol error are omitted.

The coordinator sends the **Prepare**, **Rollback** and **Commit** messages. The participant returns the **Prepared, ReadOnly**, **Aborted** and **Committed** messages.

The 2PC protocol makes a "presumed abort" assumption to minimize work for normal commit case. Presumed abort means that no knowledge of a transaction implies it is aborted, which allows the following optimizations:

A coordinator can delay logging anything about the transaction until the commit decision.

A participant can terminate the protocol and remove details of the transaction after sending an **Aborted** or **ReadOnly** statuses to the coordinator during phase one. No acknowledgement message from the coordinator is needed.

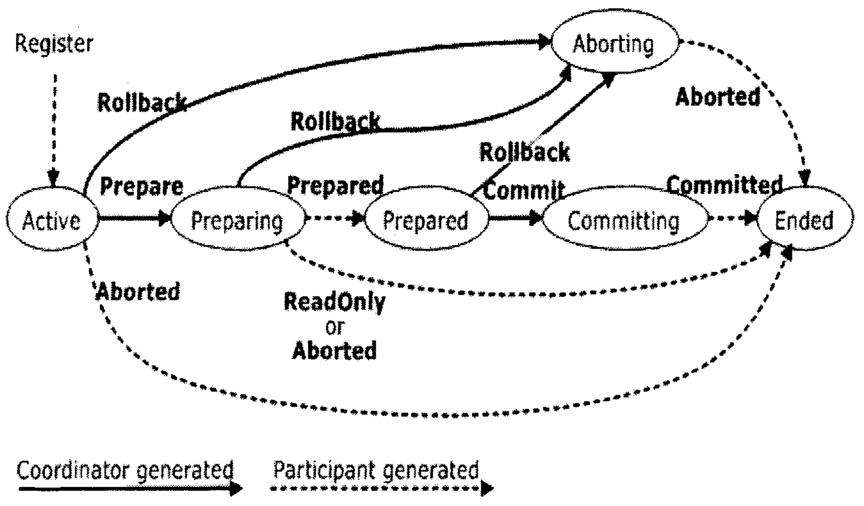


Figure 75: Protocol State Diagram for Two-Phase commit protocol.

After a **Prepared** status is received during phase one, an abort outcome allows a coordinator to forget the transaction after sending the **Rollback** message to its participants. No acknowledgement message from the participant is needed.

Only after a **Prepared** status is received, a commit outcome requires a coordinator to remember the transaction until all **Committed** acknowledgement messages have been received from its participants.

The coordinator initiates the protocol and requests participants to vote by issuing a **Prepare** message. The participant while processing the prepare message, can do either of the following:

It can reply with a **ReadOnly** status, which indicates that it votes to commit and does not need to participate further in the 2PC protocol. In this case, the two-party protocol is in the Ended state.

It can reply with an **Aborted** status, which indicates that it votes to not commit and does not need to participate further in the 2PC protocol. In this case, the two-party protocol is in the Ended state.

It can reply with a **Prepared** status, which indicates that it votes to commit. In this case, the two-party protocol is in the **Prepared** state. A **Prepared** status also indicates that the participant has reliably stored information needed to either commit or abort even if it subsequently fails.

If during the Preparing state the coordinator sends the **Rollback** message, the participant enters the Aborting state.

For the overall transaction, once all **Prepared** reply messages have returned, the coordinator decides whether the outcome for the overall transaction is to commit or abort. It permanently records the decision on stable storage and sends the **Commit** or **Rollback** to all participants, leaving each of the two-party protocols in the same Committing or Aborting state. When each participant has finished committing or aborting, it replies with a **Committed** or **Aborted** acknowledgment.

The state diagram in Figure 76 specifies the behavior of the protocol between a coordinator and one of its participants. The state reflects what both sides know of their relationship. Omitted are details such as resending of messages or the exchange of error messages due to protocol error.

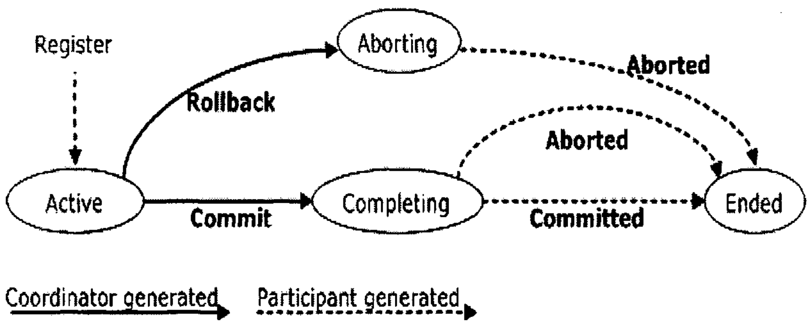


Figure 76: One-Phase Commit Protocol State Diagram.

The coordinator sends the **Commit** and **Rollback** messages.

The participant sends the **Committed** and **Aborted** outcome status messages.

The coordinator begins by sending a **Commit** or **Rollback** request, putting the two-party protocol in the Completing or Aborting state. The only outcome of the Aborting state is for the participant to send an **Aborted** status message. While in the Completing state, the participant can decide on either commit or abort, returning a **Committed** or **Aborted** status message.

The above state diagram did not discuss failure semantics. The **Replay** message can be use by a participant to solicit the transaction outcome from the coordinator after failure, where the participant provides its protocol service port reference. An acknowledgement is not needed, because the coordinator will begin sending protocol messages.

The participant's 2PC service is defined as:

```
<wsdl:portType name="2PCParticipantPortType">
        <wsdl:operation name="Prepare">
            <wsdl:input message="wstx:Prepare" />
        </wsdl:operation>
        <wsdl:operation name="OnePhaseCommit">
            <wsdl:input message="wstx:OnePhaseCommit" />
        </wsdl:operation>
        <wsdl:operation name="Commit">
            <wsdl:input message="wstx:Commit" />
        </wsdl:operation>
        <wsdl:operation name="Rollback">
            <wsdl:input message="wstx:Rollback">
        </wsdl:operation>
        <wsdl:operation name="Unknown">
            <wsdl:input message="wstx:Unknown" />
        </wsdl:operation>
        <wsdl:operation name="Error">
            <wsdl:input message="wstx:Error" />
        </wsdl:operation>
</wsdl:portType>
```

The coordinator's 2PC service is defined as:

```
<wsdl:portType name="2PCCoordinatorPortType">
        <wsdl:operation name="Prepared">
            <wsdl:input message="wstx:Prepared" />
        </wsdl:operation>
        <wsdl:operation name="Aborted">
            <wsdl:input message="wstx:Aborted"/>
        </wsdl:operation>
        <wsdl:operation name="ReadOnly">
            <wsdl:input message="wstx:ReadOnly"/>
        </wsdl:operation>
        <wsdl:operation name="Committed">
            <wsdl:input message="wstx:Committed"/>
        </wsdl:operation>
        <wsdl:operation name="Replay">
            <wsdl:input message="wstx:Replay"/>
        </wsdl:operation>
        <wsdl:operation name="Unknown">
            <wsdl:input message="wstx:Unknown" />
```

```
    </wsdl:operation>
        <wsdl:operation name="Error">
            <wsdl:input message="wstx:Error"/>
        </wsdl:operation>
</wsdl:portType>
```

A party should be prepared to receive duplicate notifications and respond back to the source in a manner consistent with the current state of the target. **Replay** informs a party that the sender is recovering. In response the party should respond back to the source in a manner consistent with the current state of the target.

If a party receives a notification protocol message for an unknown transaction, it should transmit an **Unknown** notification back to the source.

The outcome notification protocol is used by applications to find out when a transaction has completed and what the outcome is.

The state diagram in Figure 77 specifies the behavior of the protocol between a coordinator and one of its participants. The state reflects what both sides know of their relationship. Omitted are details such as resending of messages or the exchange of error messages due to protocol error.

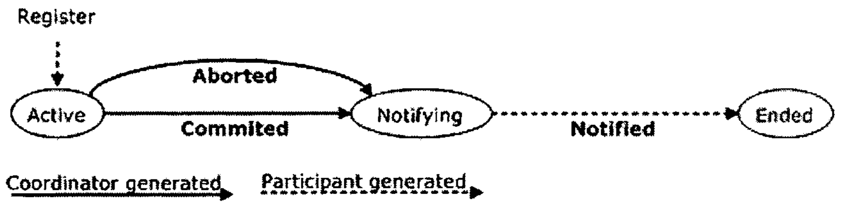


Figure 77: Outcome notification protocol state diagram.

The outcome notification protocol is used by applications to find out.

The coordinator sends the outcome notification indicating either a **Committed** or **Aborted** status. The **Notified** message is a simple acknowledgement that the outcome notification was received. The coordinator must remember the outcome until this acknowledgement is received.

The participant's OutcomeNotification service is defined as:

```
 <wsdl:portType
name="OutcomeNotificationParticipantPortType">
       <wsdl:operation name="Committed">
           <wsdl:input message="wstx:Committed"/>
       </wsdl:operation>
       <wsdl:operation name="Aborted">
           <wsdl:input message="wstx:Aborted"/>
       </wsdl:operation>
       <wsdl:operation name="Unknown">
           <wsdl:input message="wstx:Unknown" />
       </wsdl:operation>
       <wsdl:operation name="Error">
           <wsdl:input message="wstx:Error" />
       </wsdl:operation>
 </wsdl:portType>
```

The coordinator's OutcomeNotification service is defined as:

```
<wsdl:portType
name="OutcomeNotificationCoordinatorPortType">
        <wsdl:operation name="Notified">
            <wsdl:input message="wstx:Notified"/>
        </wsdl:operation>
        <wsdl:operation name="Replay">
            <wsdl:input message="wstx:Replay"/>
        </wsdl:operation>
        <wsdl:operation name="Unknown">
            <wsdl:input message="wstx:Unknown" />
        </wsdl:operation>
        <wsdl:operation name="Error">
            <wsdl:input message="wstx:Error"/>
        </wsdl:operation>
 </wsdl:portType>
```

A party should be prepared to receive duplicate notifications and respond back to the source in a manner consistent with the current state of the target.

Replay informs a party that the sender is recovering. In response the party should respond back to the source in a manner consistent with the current state of the target.

If a party receives a notification protocol message for an unknown transaction, it should transmit an **Unknown** notification back to the source.

3.2 Web Service Reliability

The messages that Web services exchange have to be received reliably. These SOAP messages have to be sent to the receiving Web service with guaranteed delivery, without duplication, and with guaranteed message ordering. The basic requirements are:

- Guaranteed message delivery, at least Once semantics
- Avoiding duplication in a guaranteed manner, At most once semantics
- Guaranteed message ordering within a context delimited using a group id.

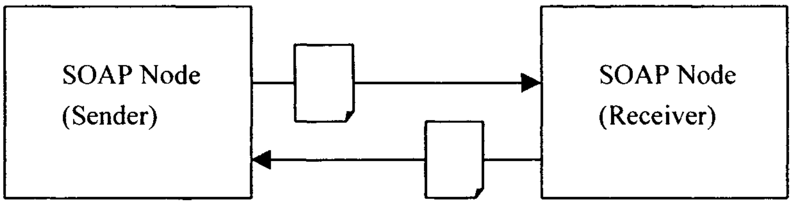


Figure 78: Sender and receiver exchanging SOAP messages.

For a message to be reliably sent from a Sender to a Receiver, both being SOAP nodes, a return message has to be received. The return message may be an acknowledgement of the received message or may be a fault message. All reliable messages must carry a globally unique message Id. The return message is correlated with the original message through message Id. The acknowledgement thus must contain the original message Id. The messages also are time stamped. The time stamp must correspond to the XML Schemas (W3C).

The Sender Node must keep sending the original message with the same message Id until it either gets a reply (containing the message id) or exceeds the number of retries. In the latter case the sender must inform the application layer about the failure.

Both Sender and Receiver must persist the message till the handshaking is over or the time to live element is exceeded. A non-volatile storage memory must be used for persistent messages.

Duplicate elimination must happen at the receiver side. Duplicate messages can arrive because of intermittent failures or routing problems. The receiver must remove duplicate messages with the same message Id.

The Sender and Receiver also must collaborate to provide proper sequencing between messages. The messages are numbered inside a group Id. If messages arrive out of order the Receiver must order them and then make the messages available to the application layer.

A reliable message contains:

- MessageType element,
- ReplyTo element,
- TimeToLive element,
- AckRequested element,
- DuplicateElimination element.

In case of failures, SOAP fault messages have to be generated. The SOAP fault codes have to be extended for reliability elements. They would be of type:

- InvalidMessageHeader,
- InvalidMessageId,
- InvalidRefToMessageId,
- InvalidTimeStamp,
- InvalidTimeToLive,
- InvalidReliableMessage,
- InvalidAckRequested,
- InvalidMessageOrder.

3.3 Web Service Security

Most of the security technology relies on cryptography and the possession of secrets (keys). Alternative technologies such as biometric characteristics may become pervasive over the course of time. Cryptographic algorithms can be classified into those where the same key is used to encrypt and decrypt data and those with different keys. The first category is also referred to as symmetric encryption since the same key is used in both directions implying that this key has to be protected from disclosure. For this reason, algorithms of that category are also referred to as secret-key systems. The most common symmetric, secret key algorithm is DES (Data Encryption Engine). IDEA is another popular symmetric encryption algorithm.

Since distribution of keys is a problem with a high potential of attacks, public-key crypto systems have emerged. Unlike symmetric secret key algorithms, public-key encryption algorithms use two different keys, one (public) key for encryption, and one (secret) key for decryption. Anyone can encrypt data with an individual's public key. But only the individual can decrypt data with its secret key. Based on this principle, a variety of protocols ĉan be established. Mathematically, most public-key systems are based on one-way hash functions. The most popular algorithm is RSA [Rivest-Shamir-Adleman].

Based on cryptographic algorithms, security technology has emerged in networks that also apply to Web services. A general overview over common security technology applied in networks in which Web services are deployed should be given first. After that, new security technology emerging with and being specific for Web services will be discussed in the second part of this section.

3.3.1 Secure Data Communication and Secured Networks

IPsec: IPsec (Internet Protocol Security) is a developing standard for security at the network or packet processing layer of network communication. Earlier security approaches have inserted security at the application layer. IPsec is especially useful for implementing virtual private networks and for remote user access through dial-up connection to private networks.

IPsec provides two choices of security services: Authentication Header (AH), which essentially allows authentication of the sender of data, and Encapsulating Security Payload (ESP), which supports both authentication of the sender and encryption of data. The specific information associated with each of these services is inserted into the packet in a header that follows the IP packet header. Separate key protocols can be selected, such as the ISAKMP/Oakley protocol.

SSL / TLS: The Secure Sockets Layer (SSL) is a commonly used protocol for managing the security of a message transmission over the Internet. SSL has recently been succeeded by Transport Layer Security (TLS), which is based on SSL. SSL uses a program layer located between the Internet's Hypertext Transfer Protocol (HTTP) and Transport Control Protocol (TCP) layers. Developed by Netscape, SSL also gained the support of Microsoft and other providers and became the de facto standard until evolving into Transport Layer Security. SSL uses the public-and-private key encryption system from RSA, which also includes the use of a digital certificate.

TLS and SSL are an integral part of most Web browsers and Web servers. If a Web site is on a server that supports SSL, SSL can be enabled and specific Web pages can be identified as requiring SSL access. However, TLS and SSL are not interoperable. A message sent with TLS can be handled by a client that handles SSL but not vice versa.

VPN: A virtual private network (VPN) is a private data network that makes use of the public telecommunication infrastructure, maintaining privacy through the use of a tunneling protocol such as IPsec or SSL/TLS and additional security procedures. The initial idea of VPN was to give companies networking capabilities at much lower cost by using the shared public infrastructure rather than a private network. In another embodiment, VPN are used to separate intra-organization networks from the open Internet. Authorized users can externally access intra-networks through special gateways from the public Internet. In the space of Web services, this idea evolved into providing collaborating partners an efficient and secure platform for business-to-business interactions.

3.3.2 Digital Signatures

A digital signature (not to be confused with a digital certificate) is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document, and possibly to ensure that the original content of the message or document that has been sent is unchanged. Digital signatures are easily transportable, cannot be imitated by someone else, and can be automatically time-stamped. The possession of a signed message implies that a sender cannot easily repudiate it later.

A digital signature can be used with any kind of message, whether it is encrypted or not, so that the receiver can be sure of the sender's identity and that the message arrived unaltered. A digital certificate contains the digital signature of the certificate-issuing authority so that anyone can verify that the certificate is authentic.

DSS: The Digital Signature Standard (DSS) is the digital signature algorithm (DSA) developed by the U.S. National Security Agency (NSA) to generate a digital signature for the authentication of electronic documents. DSS was put forth by the National Institute of Standards and Technology (NIST) in 1994, and has become the United States government standard for authentication of electronic documents. DSA is a pair of large numbers that are computed according to the specified algorithm within parameters that enable the authentication of the signatory, and as a consequence, the integrity of the data attached. Digital signatures are generated through DSA, as well as verified. Signatures are generated in conjunction with the use of a private key; verification takes place in reference to a corresponding public key. Each signatory has his or her own-paired public (assumed to be known to the general public) and private (known only to the user) keys. Because an authorized individual using his or her private key can only generate a signature, the corresponding public key can be used by anyone to verify the signature.

MD5: Message Digest 5 (MD5) is another widely used digital signature algorithm to verify data integrity through the creation of a 128-bit message digest from data input (which may be a message of any length) that is claimed to be as unique to that specific data as a fingerprint is to a specific individual. MD5 is intended for use with digital signature applications, which require that large files must be compressed by a secure method before being encrypted with a secret key under a public key cryptosystem.

3.3.3 Digital Certificates

A digital certificate is an electronic "access card" that establishes credentials when doing business or other transactions. It is issued by a certification authority (CA). It contains an individual's name, a serial number, expiration dates, a copy of the certificate holder's public key (used for encrypting messages and digital signatures), and the digital signature of the certificate-issuing authority so that a recipient can verify that the certificate is real. Some digital certificates conform to a standard, X.509. Digital certificates can be kept in registries so that authenticating users can look up other users' public keys.

3.3.4 Secure Authentication and Certification

S/PKI: The (Simple) Public Key Infrastructure (S/PKI) provides a means for relying parties (i.e., recipients of certificates who act in reliance on digital signatures verified using those certificates) to know that another individual's public key actually belongs to that individual. Certification Authorities (CA) are those entities that initially authenticate the public key subscriber and issue the certificate for use by the relying parties. Independent verification that a certification authority operates using industry accepted standards for key management and trust relationships among CA.

A Public Key Infrastructure enables users of an insecure public network such as the Internet to securely and privately exchange data using public key cryptography. A PKI offers an integral solution that handles all security issues related to online data transmission. It enables the sending of information in a safe and private way. Later on, we will take a closer look at public key cryptography.

A PKI offers three primary services:

- Authentication - The assurance that someone really is who he claims to be.
- Confidentiality - The assurance that no one but the intended recipient can access the data.
- Integrity - The assurance that data has not been altered while in transportation. It ensures that the message received is the same message that was sent.

A public key infrastructure consists of:

- A certificate authority (CA) that issues and verifies digital certificate. A certificate includes the public key or information about the public key.
- A registration authority (RA) that acts as the verifier for the CA before a digital certificate is issued to a requestor.
- Directories where the certificates (with public keys) are held.
- A certificate management system.

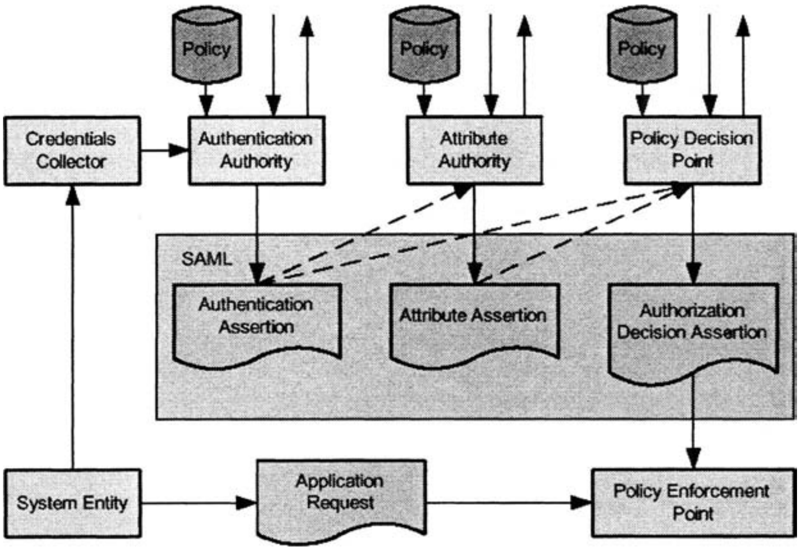


Figure 79: The SAML domain model.

A PKI is built on trust and at the core of this trust is the certification authority (CA). The CA issues digital certificates on which a verified identity is bound to a public key that is trusted by all parties involved. The public key infrastructure assumes the use of public key cryptography, which is the most common method on the Internet for authenticating a message sender or encrypting a message. For all these reasons, the foundation of Web services and e-commerce has to be built on strong security technology enabling trust and full legal recognition as the key business success factors for electronic commerce applications over the Internet.

The current SAML 1.0 Specification Set (29-Mar-2002) [x] consists of following documents: Assertion Schema, Protocol Schema, Bindings and Profiles, Security and Privacy Considerations, Conformance Program Specification, and a Glossary.

One major design goal for SAML is Single Sign-On (SSO), the ability of a user to authenticate in one domain and use resources in other domains without re-authenticating. However, SAML can be used in various configurations to support additional scenarios as well. Several profiles of SAML are currently being defined that support different styles of Single Sign-On and the securing of SOAP payloads. The assertion and protocol data formats are defined in the SAML specification.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <SOAP-SEC:Signature
       xmlns:SOAP-SEC="http://schemas.xmlsoap.org/soap/security/2000-12"
        SOAP-ENV:actor="some-URI"
        SOAP-ENV:mustUnderstand="1">
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <ds:SignedInfo>
            <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026">
            </ds:CanonicalizationMethod>
            <ds:SignatureMethod Algorithm="http://www.w3.org/20/09/xmldsig/dsa-hal"/>
            <ds:Reference URI="#Body">
              <ds:Transforms>
                <ds:Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-26"/>
              </ds:Transforms>
              <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
              <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
             </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>MC0CFFrVLtRlk=...</ds:SignatureValue>
        </ds:Signature>
      </SOAP-SEC:Signature>
    </SOAP-ENV:Header>
</SOAP-ENV:Envelope>
```

Figure 80: Fragment of a digitally signed SOAP header document.

3.3.5 WS-Security

WS-Security provides enhancements to SOAP messaging by providing message integrity, message confidentiality and single message authentication. WS-Security is being developed to work with a variety of security paradigms, namely Kerberos, PKI and SSL.

WS-Security provides support for secure token transfer, message integrity, and message confidentiality. These mechanisms can be individually used or in conjunction with each other. The <Security> header block in the message will provide security specific information to a Receiver. There may be multiple header blocks destined for multiple Receivers. The <Security> header block contains information about the encryption and signing steps that the Sender went through. The Security header element may include Username password tokens. A BinarySecurityToken may also be included. A BinarySecurityToken has an encodingType (say Base64Binary) and a valueType that indicates what the Security Token is (say a Kerberos Ticket). Binary SecurityToken may also contain X.509 certificates.

```
<S: Envelope>
<S: Header>
......
<Security S:actor=" ..." S: mustUnderstand="..">
......
</Security>
.......
```

Figure 81: WS-Security document.

The WS-Security specification builds on XML-Signature specification. The <Security> header block is used to carry a signature compliant with the XML Signature specification within a SOAP Envelope for the purpose of signing one or more elements in the SOAP Envelope. Multiple signature entries may be added into a single SOAP Envelope within the <Security> header block. Senders should take care to sign all important elements of the message, but care must be taken in creating a policy that will not to sign parts of the message that might legitimately be altered in transit.

4. SERVICE LEVEL AGREEMENTS

If dynamic composition of Web services were to become a reality, a number of fundamental issues need to be addressed beyond agreement on document exchange formats. Some of these issues are (a) how will Web services agree on what will be executed by each of the participants? (b) How will Web services agree upon how well (performance, quality, etc) each of them will execute? (c) Who will be responsible for the overall execution or completion? (d) Who will be responsible if there is a failure in the overall execution or completion? (e) How will Web services trust each other? Service level agreements are the corner stone for addressing these issues in Web services. A service level agreement is an agreement between two Web services regarding the guarantees that one of the Web services (provider) offers to the other (consumer). The guarantees are about what transactions need to be executed and how well they should be executed. SLA help divide the responsibilities and risks among Web services.

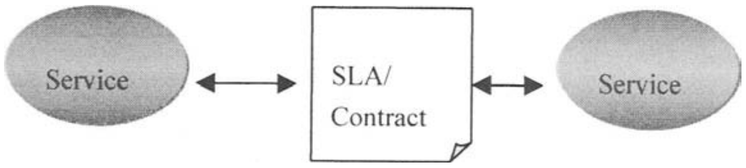


Figure 82: Service Level Agreement between two services.

In a real-world scenario each Web service interacts with many other Web services, switching between the roles of being a provider in some interactions and consumer in others. Each of these interactions could potentially be governed by an SLA. Considering the number of interactions that a Web service typically executes for completing one transaction, and the number of transactions that are executed within a day, the job of manually keeping track of which SLA are violated, why they are violated, and how many times they are violated becomes an overwhelming task. Further, considering the legal and monetary implications in violating SLA, it is in the best interest of a Web service to predict and correct violations before they occur. On the other hand, if there is too much leeway in the specification of SLA, a Web service may not be able to fully capitalize on its capabilities.

Web services SLA management refers to automatic monitoring, enforcement, and optimization of SLA between Web services. One of the enablers for automated SLA management is a flexible but precise formalization of what an SLA is. The flexibility is needed since we neither can completely understand nor can anticipate all possible SLA for all the different types of Web service providers. This will also help create a generic SLA management system for managing a range of different SLA.

The precision is essential so that an SLA management system can unambiguously interpret, monitor, enforce, and optimize SLA.

4.1 Specification of Service Level Agreements

4.1.1 Introduction to SLA

An SLA is typically signed between two parties, which have the role of provider and consumer respectively. A typical SLA [Sturm 2000] includes the following components:

- ***Purpose*** – describing the reasons behind the creation of the SLA
- ***Parties*** – describes the parties involved in the SLA and their respective roles.
- ***Validity Period***- defines the period of time that the SLA will cover. This is delimited by start time and end time of the term.
- ***Scope*** - defines the services covered in the agreement.
- ***Restrictions*** - defines the necessary steps to be taken in order for the requested service levels to be provided.
- ***Service-level objectives*** - are the levels of service that both the users and the service providers agree on, and usually include a set of *service level indicators*, like availability, performance and reliability. Each aspect of the service level, such as availability, will have a target level to achieve. Service Level objectives have day-time constraints associated with them, which delineate their validity.
- ***Service-level indicators*** - means by which these levels are measured. Service Level Indicators (SLI) are the base level indicators.
- ***Penalties*** - spells out what happens in case the service provider under-performs and is unable to meet the objectives in the SLA. If the agreement is with an external service provider, the option of terminating the contract should be built in.
- ***Optional services*** - provides services that are not normally required by the user, but might be required as exception.
- ***Exclusions*** - specifies what is not covered in the SLA.
- ***Administration*** - describes the processes created in the SLA to meet and measure its objectives and defines organizational responsibility for overseeing each of those processes

SLA specification has been researched earlier. Quality management Language (QML) [Frolund 1998] has been proposed for specifying Quality of Service for applications. In QML contracts are instances of contract types. A contract type defines the structure of its instances. In general a contract contains a list of constraints. A constraint consists of a name, an operator and a value. The name refers to a dimension and can also refer to a dimension aspect. A dimension aspect can be a percentile, mean, variance, and frequency. Usually SLA have validity periods (start and end dates) and have SLO that in turn have daytime constraints (e.g. an SLO may be valid only from Mon-Wed, between 6:00 PM-8:00 PM). It is quite essential to capture this kind of information in the contracts. This kind of information is however not captured in QML. In addition there should be a possibility of specifying a wide range of mathematical operations on the measured data. An example would be to guarantee *response time of 5 longest running transactions in last 24 hours < 25ms*. As it does not fall in the allowed dimension aspects it would not be possible to capture this in QML.

Bhoj et al. [Bhoj 98] describe a contract to be defined by a triple (P,M,A), where P is a set of properties, A is the set of assertions and M is the set of methods available on the contract. An *assertion* is an atomic group of statements agreed upon between the parties agreeing to the contract. Statements in an assertion are made up of logical predicates whose values can be uniquely determined. The logical predicates are composed using variables as well as logical operators, quantifiers, set operations and constraints on these variables. An example assertion may be *response time < 25 ms*. We believe however, that assertions have to be unambiguously specified, and the Web services being autonomous should not need human intervention to understand the meaning of SLA, and to monitor and measure their compliance. Also the Web services will sign numerous SLA with multiple parties over time and the SLA management process should be automated as much as possible. An assertion as mentioned above is quite ambiguous. This could mean an instance response time or average response time. Again if it is average response time that is being referred to, is it averaged over every 5 minutes, an hour or 24 hours? It is also necessary to indicate when the averages are calculated. Is it at 6PM everyday?

The assertions thus have to be quite unambiguous. Also the assertions have to be captured in such a way that their meaning is clear to Web services software that handle them, even if they are implicitly understood by humans. It is also necessary to identify components in assertions that are common across SLA so that base modules can be identified and SLA management can be automated.

4.1.2 Rethinking SLA Specifications

4.1.2.1 Precision

To understand the notion of precision in SLA, consider the following example. Imagine a Web service for ordering supplies on the Internet. Assume that "order-supplies" is an operation supported by this Web service. In order to execute this operation, a client should send a "purchase-order" document and should obtain a "confirmation" document in response. Now, consider the following guarantee by the supplier: 95% of the time, the time for executing "order-supplies" will be less than 20 seconds.

While this may seem like a reasonable specification that can be automatically managed on the first look, there are several ways to interpret this:

When? When should a Web service check for the compliance of this SLA? Here are some examples when the evaluation of this SLA can be triggered: (i) whenever a new "order-supplies" execution is completed, (ii) after every 10 "order-supplies" executions, (iii) at the end of the day, or (iv) just before the termination of the SLA. It is trivial to note that if this guarantee holds true in one of these cases, it does not necessarily hold in the rest.

Which? Which inputs should be considered in evaluating the SLA? In order to check whether the above guarantee is upheld or not, one can consider (i) all executions of "order-supplies" since the formation of the SLA, (ii) all executions of "order-supplies" since the beginning of that day, (iii) all executions of "order-supplies" no matter who the client is, (iv) all executions of "order-supplies" initiated by the client with which the SLA is created, or (v) the last 100 executions of "order-supplies", no matter when they happened.

Where? Where is the timing of the execution monitored? It can either be monitored at the client issuing the request or at the service provider handling the request. Usually, clients are interested in SLA that give guarantees from their perspective as opposed to the service provider's perspective. On the other hand, it is quite hard for the service provider to make client-side guarantees since that could be influenced by factors that are outside the control of the service provider (e.g., the documents between the client and service provider often flow through other service providers such as ISPs).

What and How? In the above example, the metric of interest is the response time from the when the purchase-order is sent out to when a confirmation is received. The guarantee is on the 95th percentile. One way to interpret this is that out of every 100 executions, 95 will have a response time of less than 20 s. Another way to interpret this is that 95 out of every

100 executions will have a *mean* response time of less than 20 s. Metrics that are not as well defined like availability, reliability, cost, and quality will further complicate how an SLA should be evaluated.

For a system to automate the management of SLA, all of the above ambiguities should be removed from the SLA specification.

4.1.2.2 Flexibility

There are many kinds of interactions between consumers and providers of Web services – single request-reply interaction, multiple back-and-forth messages, short-lived interactions, or interactions that last for several hours or days. Web services themselves are quite diverse. The metrics that are of interest in an SLA are, many times, quite specific to the Web service. For example, a bookseller would like to provide a guarantee on the number of days it takes to deliver a book. A credit card authorization service provider would like to provide a guarantee on the security of the transmitted information. The vast diversity in the kinds of interactions as well as metrics that are of interest necessitates the need for a *flexible* SLA formalization. One should be able to build an automated SLA management system (or at least a framework for SLA management) using the flexible formalization.

Emerging standards such as Web services Definition Language (WSDL) and Web service Flow Language (WSFL) are creating flexible and generic interaction models for Web services. For example, WSDL introduces concepts such as messages, operations, ports, and end points – which are useful for describing the operations of any Web service. Similarly, WSFL introduces the notion of activities and flows – which are useful for describing both local business process flows and global flow of messages between multiple Web services. So, one way to create a flexible SLA formalization is to build upon these concepts. In other words, one can create a flexible SLA formalization by associating "quality metrics" to the formalizations that are already defined in WSDL and WSFL.

Here are some examples that show how such association can be done.

- Response time of a Web service operation.
- Response time of a flow.
- Security of an operation.
- Number of times an activity is executed in a flow.
- Cost of executing an operation.
- Availability of an end point.

Metrics	Definition	Applicable for Web service constructs	Target specification
Availability	Availability of an entity minus the impact time from any events other than loss of network or system availability	Port , Port Type, ServiceProvider, Endpoint, Flow.	Expressed as percentage or in time
Response Time	The time taken for an entity to complete a client request and return a response	Operation, Activity, Flow.	normally specified as service to complete X% of transactions of type Y to be completed within Z seconds
Throughput	Number in unit of time	Operation, Activity, Flow, Endpoint, Port.	Expressed as tps
Security	The security at different levels need to be agreed upon in the SLA	Service Provider, Port, Operations, Activity, Flow, Endpoint.	Level supported High Medium None
Payment Rate	Rate at which the service/transactions are charged	Operations, Activity, Port, Service Provider, Endpoint, Flow.	The payment rate can be subscription based or can be expressed per transaction type or instance level.
Problem Response	The time required for a client to receive a response after reporting a problem	Service Provider, Port, Endpoint.	1-High Priority[md] 2-Medium Priority[md] 3-Low Priority[md] (all in target time)
Problem Circumvention or Resolution Time	The time required for a client to receive a circumvention or a solution after reporting a problem	Service Provider, Port, EndPoint.	Expressed as times for various categories of problems. High category problems must be resolved faster.
Repeat Trouble Rate	Number of times the trouble is repeated before it is escalated.	Service Provider, Port, Endpoint.	This is usually expressed as a rate
Account set up time	Time taken to create and set up new accounts.	Service Provider, EndPoint, Port.	Expressed as unit of time

Figure 83: Web services problem resolving.

Here are some examples that show how such association can be done.

- Response time of a Web service operation.
- Response time of a flow.
- Security of an operation.
- Number of times an activity is executed in a flow.

- Cost of executing an operation.
- Availability of an end point.

Another way to guarantee flexibility is to identify generic components in typical SLA specifications, which in turn are extensible so that new components can be defined by extending these base components

4.1.3 SLA Specification languages

4.1.3.1 WSML

An SLA typically has a date constraint (start date, end date, nextevaldate) and a set of Service Level Objective (SLO). An SLO in turn has typically a day–time (Mo-We, 6:00PM-8:00 PM) constraint and a set of clauses that make up the SLO. A clause is based on measured data. This is referred to as a *measuredltem*. A measuredltem can contain one or more *items*. A measuredAt element relates to the "where" part of the specification, referring to the side the measurements are taken (provider or consumer). A clause evaluation is done either when an event happens, e.g. say a message arrives, an operation completes or at a fixed time, say at 6PM. We call this an *evalWhen* component. This refers to the "When" component of the specification. Once the evalWhen trigger arrives, a set of samples of measuredltem are obtained applying a sampling function. The *evalOn* component determines the set of samples over which a condition is applied. The sample set is a constrained set of measured data that is obtained by applying the evalOn component. Examples of evalOn components may be a number or a time period, e.g. the 5 longest running transactions, or all the samples for last 24 hours.

```
SLA  = Dateconstraint Parties SLO*
Dateconstraint = Startdate Enddate Nextevaldate
SLO = Daytimeconstraint Clause*
Dateconstraint = Day* Time*
Clause =
MeasuredItem EvalWhen EvalOn EvalFunc EvalAction
MeasuredItem = Item*
  Item  =
MeasuredAt ConstructType ConstructRef
```

Figure 84: Example SLA.

The evalOn component refers to the "which" part of the specification. A condition is thereafter applied on the sample set so obtained. This component is the actual *evalFunc* which is applied on the sample set so obtained from the evalOn component. An example of evalFunc would be average response time function < 5 ms. The evalFunc must be a mathematical function that is expressible in terms of its inputs and logic. The evalFunc refers to the "What" and "How" part of the specification. A sample clause like *At 6 PM the Average response time for the 5 longest running book buy transactions measured on the client side should be < 5 ms* can be broken in measuredItem (item:bookbuy transaction measuredAt:consumer), evalWhen(at 6PM), evalOn function(set of 5 longest running transactions) and the evalFunc (average response time < 5 ms). The evalAction could be notification to the administrator in this case.

4.1.3.2 WSLA

IBM proposed the WSLA specification for specifying Service Level Agreement between Web services (WSLA). WSLA Agreement is specified in addition to the WSDL specification of the Web services.

An SLA in the WSLA language contains three sections: a section describing the parties, a section containing one or more service definitions and the section defining the parties obligation.

The parties involved in a WSLA agreement is always be between two signatory parties, and possibly supporting parties. The supporting parties cannot be held liable for the SLA. A service definition contains one or more service objects. The service definition is for common understanding between the signatory parties of the metric involved. There can be one or more SLAParameters in a service object. SLAParameters define the properties relevant for defining guarantees. The SLAParameter is related to a metric and specifies how it is measured and aggregated. Metrics can be composed out of other metrics. Measurement directives define how parameters are measured by the organization that makes the metrics available. A Function specifies how to compute a metric in terms of other metrics and constants. The obligations that bind parties are of two types: A service level objective and action guarantees. Every obligation has an obliged party. Service providers are obligated for the Service Level Agreements.

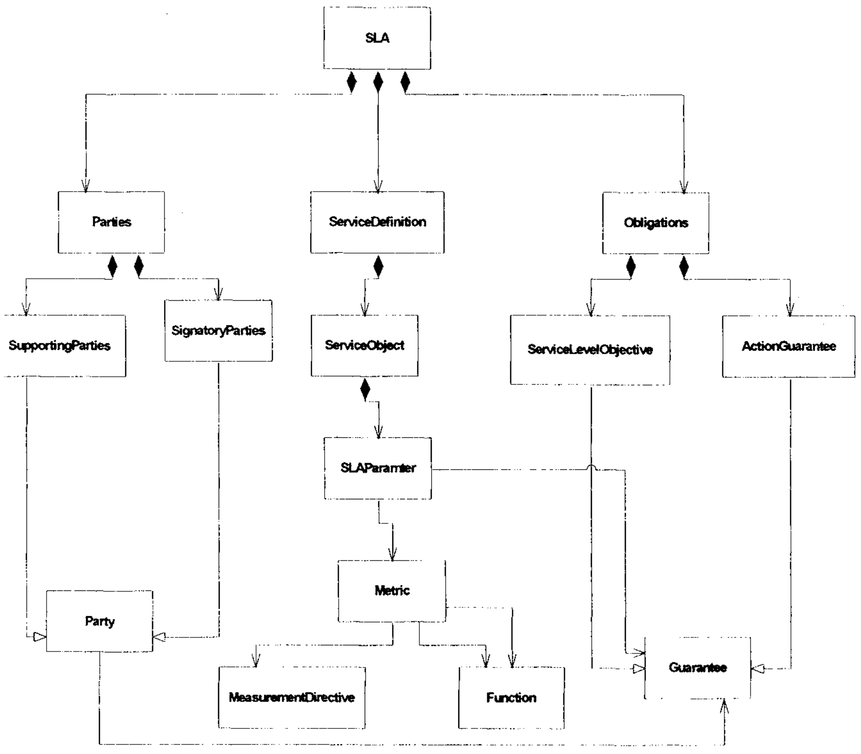


Figure 85: Basic constructs of the Web service Level Agreement (WSLA) language.

4.1.3.3 WS-Agreement

The WS-Agreement is signed between an agreement producer and an agreement initiator (also known as a customer). The *agreement* captures the expected service behavior in potentially domain-specific agreement *terms*.

The Agreement term must define unambiguously what is necessary to satisfy an agreement. WS-Agreement extends another standard called WS-Policy. The WS-Agreement model uses service creation as the negotiation primitive for providing an AgreementFactory interface to agreement providers, inheriting from the FactoryPortType. To support more complex negotiations, WS-Agreement allows agreements to be linked by extensible relationships. An Agreement has two states: satisfied (all its terms have been or being met) or violated (when at least some of its terms are not being met). A special form of satisfaction is the completion.

The AgreementInitiator obtains a handle to the AgreementFactory and negotiates the terms with it. Once the negotiation phase is complete Agreements are created. These Agreements can be monitored by the AgreementInitiator. The consumer uses the application service whose behavior must conform to the Agreements signed between the Initiator and the provider.

```
<xsd:complexType name="AgreementType">
   <xsd:complexContent>
      <xsd:extension base="wsp:PolicyExpression">
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="agremeent" type="AgreementType" />

<xsd:complexType name="TermType" abstract="true" >
   <xsd:attributeGroup
ref="wsp:CompositorAndAssertionAttributes" />
   <xsd:attribute name="Name" type="xsd:NCName" />
   <xsd:attribute                       name="Negotiability"
type="gsa:NegotiabilityType"/>
</xsd:complexType>

<xsd:simpleType name="NegotiabilityType">
   <xsd:restriction base="xsd:QName">
      <xsd:enumeration value="gsa:Fixed"/>
      <xsd:enumeration value="gsa:Negotiable"/>
   </xsd:restriction>
</xsd:simpleType>
```

Figure 86: WS Agreement.

The WS-Agreement model defines two essential portTypes: the gsa: Agreement service and the gsa:AgreementFactory service. An agreement initator is a client of the gsa:AgreementFactory who negotiates by invoking the createService operation with appropriate argument content. The service that is created can have a pre-set termination time. Agreements have their lifetime and can be monitored. Agreements may be composed into further composite Agreements. The Agreement port type has certain service data elements too. The following concepts are captured as the service data elements for the Agreement portType:Creation parameters, Agreement terms, Agreement status.

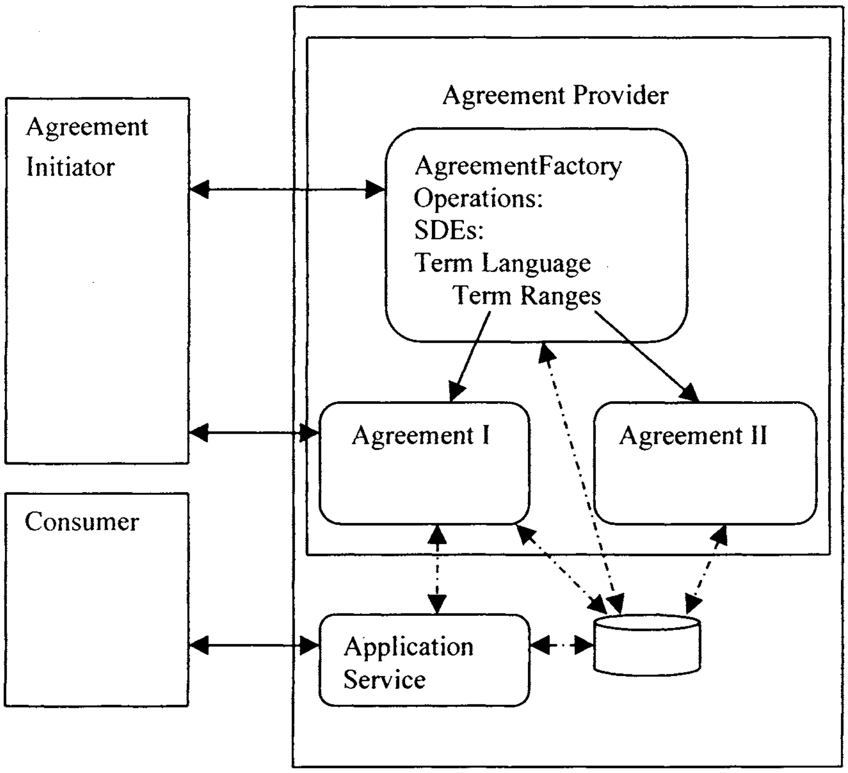


Figure 87: Agreement protocol provider.

5. SLA MONITORING

As minimal human intervention is desirable in Web services it is necessary to create monitoring engine that can take care of a variety of specifications and monitor the necessary management data. The SLA formalizations described above could be used to drive SLA monitoring engines.

To simplify the discussion, we will describe the details of the engine as if it manages a single SLA between two services. Such an engine has then two components – one on the service provider side and one on the service consumer side. Extending our notion to a large number of SLA requires that the engine keep track of the state of multiple SLA simultaneously, and be able to relate each measurement to one or more affected SLA. Extending our notion of two services to a large number of interacting services requires the engine's components to take the dual role of acting as both "service providers" in some SLA and as "service consumers" in some SLA.

5.1.1 SLM Engine

The SLM Process Controller that is part of the SLM engine has to execute the management processes for the SLM engine. These management processes are distinct from the business processes that are executed in the Web services infrastructure as discussed in chapter 2. These management process flows are created and managed for a variety of purpose. These flows may also be defined in WSFL and may expose their own management interfaces to other SLM engines of partner services. These engines can initiate management related conversation with each other. The process controller executes the *SLA monitoring process flow* for undertaking SLA evaluation and reporting. The SLA descriptions then can be used to trigger evaluations of SLO based on time or an event happening [Sahai 2003]

5.1.2 Service Level Monitoring Process Flow

The Service Level Monitoring process consists of the following steps:

1. The process (SLM process) is initiated as soon as an SLA is received as input.
2. Decide where the measurements are to be carried out. This is marked on every service level indicator in the SLA.
3. Decide where the evaluation of the SLA is to be done. The SLA evaluation is carried out at the customer side, if the SLA has items that are all measured at the customer side. Similarly, if all the measured items are measured at the provider side, the SLA evaluation is carried out at the provider side. At the end of evaluation the SLM engines exchange violation report through SLA Violation Report Exchange protocol.

If however, some of the items are measured at the customer side, and some of them are measured at the provider side, then the evaluation has to be carried out at the provider side. This last case, however, requires that the customer-side measurements are transferred to the provider-side.

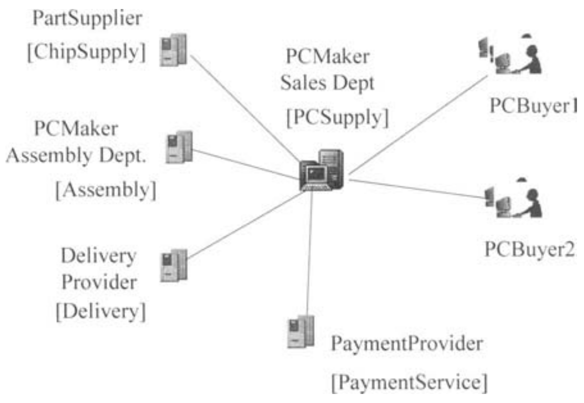


Figure 88: A web service scenario with multiple partners involved

When the evaluation of an SLA depends on measurements from both, the customer-side and provider-side, a measurement protocol is needed for transferring the measurements from the former to the latter.

```
<sla>
  <slaId>2</slaId>
  <partnerName>PcBuyer1.com</partnerName>
  <startDate>Fri Feb 15 00:00:00 PST 2002</startDate>
  <endDate>Mon Jul 15 00:00:00 PDT 2002</endDate>
  <slo>
  <sloId>1</sloId>
  <dayTimeConstraint>Mo-Fr: 9-17</dayTimeConstraint>
  <measuredItem>
  <item>
  <constructType>process</constructType>
  <constructRef>PcMaker.com/Invoice</constructRef>
  <measuredAt>PcMaker.com</measuredAt>
  </item>
  </measuredItem>
  <evalWhen>6PM</evalWhen>
  <evalOn>all</evalOn>
  <evalFunc name ="averageResponseTime" operatior ="LT" Threshold ="6"
    unit ="hours"></evalFunc>
  </slo>
</sla>
```

Figure 89: Example SLA.

Such a protocol should be designed with the following objectives in mind: (a) minimize the amount of data that is transmitted between the two sides, and (b) transfer the data in time for the evaluation of SLA to take place when triggered.

To fulfill these two objectives, the SLA monitoring engines on both sides should agree on (a) what measurements need to be transferred and at what level of aggregation (b) how frequently they should be transferred. and (c) the type and level of aggregation of the measurements. To specify the level of aggregation, typical sampling functions such as count (t), totaled, averaged, movingAvg(lastN), minN, maxN, threshold may be used. In the case when the sampling function cannot be determined all the measurements from the customer-side must be shipped to the provider-side. The reporting frequency depends on the SLA.

```
<sla>
  <slaId>1</slaId>
  <partnerName>PCBuyer.com</partnerName>
  <startDate>Fri Feb 15 00:00:00 PST 2002</startDate>
  <endDate>Mon Jul 15 00:00:00 PDT 2002</endDate>
  <slo><sloId>1</sloId >
  <dayTimeConstraint>We-Thu: 12-17</dayTimeConstraint>
  <measuredItem>
  <item>
  <constructType>process</constructType>
  <constructRef>PCMaker.com/PCDelivery</constructRef>
  <measuredAt>PCMaker.com</measuredAt>
  </item>
  </measuredItem>
  <evalWhen>6PM</evalWhen>
  <evalOn>all</evalOn>
  <evalFunc name="averageResponseTime"operator="LT"  threshold = "6"
    unit="hours"></evalFunc>
  </slo>
</sla>
```

Figure 90: Example SLA

Any web service interaction would involve at least two and often more web services. Here's an example scenario with SLA written in WSML.

The example scenario as shown in Figure 88 has two SLA between PCMaker.com and its customers namely PCBuyer1.com, PCBuyer2.com. The two SLA were namely SLA1 and SLA2. Each SLA has a single Service Level Objective. The first SLA is with PCBuyer1.com. It guarantees that between the dates of 0.2/15/02 and 07/15/02 all the invoice processes from 9-5 and on weekdays will be undertaken in 6 hours. The evaluation is done every day at 6 PM.

The second SLA is signed between PCMaker.com and PCBuyer2.com. It guarantees that between the dates of 0.2/15/02 and 07/15/02 all the PC Delivery processes from 9-5 and on weekdays will be done on an average within 6 hours. The evaluation of the SLA will be done every day at 6 PM.

The SLM engines corresponding to PCMaker.com and PCBuyer1.com have to determine where the evaluations of the SLO take place and at what intervals they share management information.

SUMMARY

In order to realize the vision of Web services enabling inter-enterprise integration it is important to not only standardize on set of technologies but also to ensure that these interactions can be monitored and managed. In this chapter we looked at the requirements and the technologies that are available to enable and manage Web service to Web service interactions. We also discussed some standards that are important to enable web service to web service interactions like WS-Security, WS-Reliability and SLA standards and efforts.

Chapter 9

MANAGEMENT USING WEB SERVICES

1. INTRODUCTION

Management technologies have developed at various layers in the management stack, often to address parts of the pieces of the big management puzzle. These management solutions have been introduced by vendors over the course of time and have not been made interacting with each other. Often the data models, the APIs, their interactions are not compatible with each other.

Efforts are being made to remedy this situation. Web services have been thought of as the unifying technology that may enable uniformization of these interfaces, and interactions between management solutions. Efforts are being made to go one step further to represent managed systems themselves as Web services. Some of the utility computing vendors and the grid community are trying to do exactly that.

2. UNIFORM REPRESENTATION THROUGH WEB SERVICES

Virtualization is a set of transformation processes in the virtualization layer during which associations between virtualized entities and "underlyings" are established and changed (we use the term underlyings to denote any resources represented by the virtualization entities—these resources can be physical or logical, grouped or otherwise constructed). For clients 'above' the virtualization layer, a created virtualized entity exposes its own 'virtual' identity, properties and behavior, while attributes belonging to underlyings remain hidden from the client.

The virtualization layer internally maintains the associations between virtualized entities and related underlyings, often also referred to as the "indirection" to underlyings. Knowledge of these associations allows "seeing" through the virtualization layer and being able to correlate virtualized entities with corresponding underlyings.

Any IT resource or group of IT resources can be a virtualized entity. Examples of virtualized entities include virtual memory providing application processes with the impression of using larger memory than actually available in a machine. Virtual disks of RAID arrays provide the impression of faster and more reliable disks than physical disk devices. Entire virtual machines can be created providing the impression that applications operate on individual machines. Network virtualization allows to fully decouple sub-networks used by different applications from one another by providing individual (IP) address spaces and DNS name spaces.

The Utility Data Center virtualizes storage in form of programmatically attachable disks to machines and networks in form of programmatically connecting selected machines by individual subnets. Virtualization points in the UDC internally are switches for Storage Area Networks (SAN) and Virtual Local Area Network (VLAN). The UDC externally exposes a control interface through which these control points are programmable in a higher-level specification language for resource environments.

Resource virtualization is used for several reasons:

- *insufficient resources* – when underlyings are scarce (e.g. virtual memory, processes as virtual processors, etc.),
- *sharing* – when underlyings need to be shared and using entities should not or cannot be aware of sharing and coordinate accordingly. The virtualization layer then coordinates sharing transparently for virtualized resources (e.g. separate address spaces in operating systems, virtual networks, virtual machines),
- *new properties* – when new properties and behavior of underlyings are desired (e.g. RAID creating disks with better performance and reliability characteristics),
- *transparent failover* – when underlyings fail, the virtualization layer can replace failed parts without exposing applications to the failures.

Similar reasons apply to services virtualizing applications to users and other using entities.

Virtualization in general may have several effects:

- *multiplication* – more quantity of an underlying can be created,
- *independence* – virtualized entities exist widely independently of one another, the virtualization layer takes care of necessary coordination when underlyings are shared,
- *isolation* – virtualized entities are isolated and not aware of one another and of underlyings,
- *protection* – virtualized entities are protected, no virtualized entity can reach into another virtualized entity,
- *decoupling* – virtualized entities are decoupled from underlyings allowing the virtualization layer to change these associations without being noticed by underlyings or virtualized entities,
- *encapsulation* – the virtualization layer encapsulates underlyings and centrally coordinates sharing (in time or space) of underlyings for created virtualized entities,
- *hiding* – underlyings and associations with underlyings are hidden in the virtualization layer and can hence be changed or moved without being noticed from outside. The hiding effect poses a barrier for management systems since it prevents correlating virtual with physical entities (see Section 4 on difficulties of management in virtualized environments).
- *layering* – virtualization separates the layer of underlyings ("physical") from the layer of virtualized entities (layers can be recursive).

3. ROLE OF MANAGEMENT SYSTEM IN VIRTUALIZED ENVIRONMENT

System management is generally understood as the process of maintaining a system in an operational state, and improving and evolving that state towards an objective. One can subdivide operational management in three typical stages:

- assess: data gathering, processing, reporting, presenting, archiving
- advise: reasoning about monitored data, drawing conclusions
- act: issuing control instructions back to managed elements.

A management system consists, on the monitoring side, of probes that monitor assigned managed elements and report to management servers that collect, process, present and archive monitored information. On the actuation side, a management system includes control interfaces of managed elements through which control instructions are issued to managed elements.

From the above definition of management, it becomes clear that virtualization overlaps in some of its tasks with a management system, since the virtualization layer must be aware of the resources available, and changes their state over time. As a consequence, one could imagine virtualization layers relying on management technologies to provide the necessary resource data. In addition, the use of virtualization techniques implies the existence of some kind of 'management system' that translates customer requests into actions within the virtualization layer. Any actions that a management system would like to issue need to be coordinated with the management system that is associated with the virtualization layer.

The above reasoning suggests that management and virtualization are difficult to separate, and may better be dealt with in concert. We argue even further, that a virtualization layer should be looked at as 'just another' management system, one of many possible such systems, each with assess, advice and act capabilities. Together, these management systems form interacting control systems. The remaining questions we address here are: what are the properties of these interacting control systems and how do legacy management systems fit in such future architectures. First, we discuss legacy management systems, and then we address a generalized interface for virtualization layers.

4. ASSUMPTIONS IN LEGACY MANAGEMENT SYSTEMS CHALLENGED BY VIRTUALIZATION

Specifically the hiding effect of the virtualization layer causes problems in management systems since underlyings are not exposed to the management system, and it is hard to track (changing) associations with underlyings. But there are other aspects that fundamentally change some of the assumptions that have been built into management systems.

These assumptions are:

Shared network assumption. Management system and managed system use the same, shared network infrastructure.

Identification of managed elements and management elements (probes, OV spy's, management servers, agents, etc.) is based on domain-wide unique (IP) addresses or hostnames in the underlying shared network.

Managed elements can be reached form the management system from anywhere in the network using known, fixed addresses.

Topology information is obtained (eventually discovered) and modeled in terms of the underlying shared, and rather constant network infrastructure.

In virtualized networks, sub-networks are independent from one another. Since they may have own policies for identification, naming and addressing (based on own IP and DNS address spaces), global identification does not apply per se. Translations between identification, name and address spaces may be necessary.

Managed elements residing in separate virtualized sub-networks cannot easily be reached from other sub-networks. Their addresses eventually must be translated giving them different outer and inner identities, names, and addresses. Management systems must be able to follow translations of managed and management elements in different domains.

Fixed topology assumption. The assumption is that topology information is modeled (eventually discovered) in terms of physical entities in a shared network that will change infrequently.

In a virtualized environment, this assumption only holds for the underlying physical network. Virtual network topologies may occur, change, and disappear frequently, controlled by the virtualization layer and bypassing the management system. Automatically maintaining topology information with known techniques (discovery protocols) is also hard due to virtualized networks. Topology must be related with dynamic information from the network virtualization layer.

Physicality assumption. The assumption is that managed elements physically exist in a network and can eventually be discovered.

Virtual entities are not physical entities. Their existence depends on the transformation process in the virtualization layer and is controlled there. Virtual entities are thus hard to be discovered automatically. They are not responding to discovery protocols and may appear and disappear spontaneously.

Uniqueness assumption. The assumption is that one entity (resource, application, service, etc.) exists only once.

Since virtualization has a multiplication effect, entities may exist multiple times, even under same identity (for purposes of transparent replication, for instance).

Fixed association assumption. The assumption is that applications reside on machines for longer periods, hence monitoring data from machines can implicitly be correlated with the application running on that machine.

In virtualized environments, associations between machines and applications may change frequently. Changes are under the control of the virtualization layer, not the management system.

Single-layer assumption. The assumption is that the management systems only views one layer of resources and applications, only seeing the elements that are visible in that layer.

In a virtualized environment, the virtualization layer introduces a clear and enforced separation between layers of underlying entities and the layer of created (transformed) virtualized entities. These boundaries must be obeyed by management systems. We propose modeling layers as separate management domains taking separate identification, name and address spaces in those domains into account and performing necessary translations for cross-domain interactions.

Highest authority assumption. The assumption is that the management system has highest authority (power of control) in the system. All control is exercised from the management system and its operators.

Virtualization layers have emerged independently from management systems as control points outside of management systems. These control points must be integrated back into management systems reinforcing their authority of control (see section 5).

Full transparency assumption. The assumption is that the management system sees everything in the management domain and has control.

The virtualization layer internally hides associations with underlying entities making them intransparent to the management system,

Furthermore, the virtualization layer can alter associations anytime without notifying any other components in the system including the management system.

Shared infrastructure assumption. The management system, or part of it, uses resources from the same environment as the managed system for its own operation, resources that may have been virtualized as well.

Management interface assumption. The assumption is that manageable elements provide management interfaces (such as SNMP interfaces) that can be reached through a shared network infrastructure and are used to exchange monitoring data and control instructions.

Virtualized entities (resources) usually do not have management interfaces (although they could), making them invisible for management.

5. CONCLUSIONS FOR MANAGEMENT SYSTEMS

Based on the discussion, various conclusions for management systems and for virtualization layers can be drawn:

- Open the virtualization layer for management systems allowing them to access information about associations between virtualized and underlying entities.
- Integrate virtualization control into the management system re-instantiating authority of control of the management system.
- Model layers as management domains. Establish separate management domains for virtualized and underlying entities recognizing the layer boundary between them.
- Provide management interfaces for virtualized resources allowing them to be accessed for management purposes similarly like their underlying counterparts.
- Develop resource (service) models that explicitly incorporate virtualizations.
- Introduce a notion of time when following associations from virtualized entities to underlying physical entities since those associations vary over time.
- Reconsider identification, naming and addressing of managed as well as management elements based on network addresses. Translations across virtual network domains must be taken into account. An alternative is creating a separate system for element identification, naming and addressing that is independent of virtualization.
- Divide topology information into static aspects and dynamic aspects that depend on associations created by the virtualization layer and may change.

6. INTERFACE FOR A GENERIC VIRTUALIZATION LAYER

Due to the large variety, individual virtualization layers or systems are not discussed here. Rather, a pattern for a generic interface between a management system and a generic virtualization layer is presented that can guide construction of resource management systems.

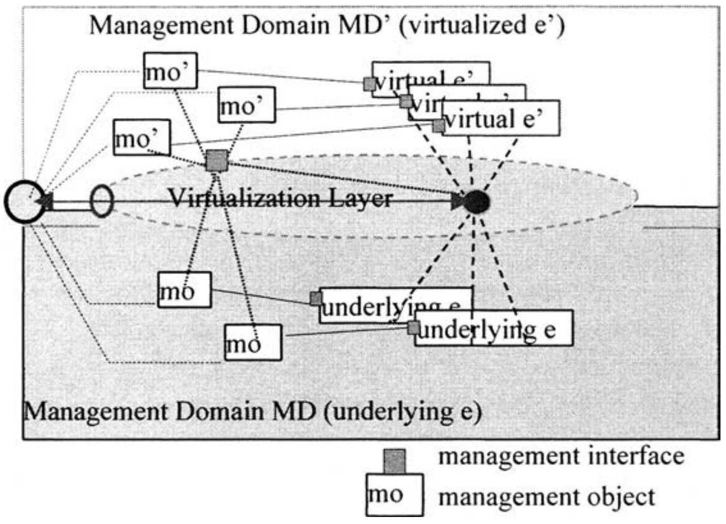


Figure 91: Management in an environment with virtualized and underlying entities.

Figure 91 shows an abstracted scenario with a set of underlying physical entities E (e ∈E) in the lower layer and virtualized entities E' (e'∈E') in the upper layer. E' is created by transformations in the virtualization layer at time t. Associations (n:m) between E and E' are maintained and controlled by the virtualization layer:

assoc(E,E') ⊆ {en × em'},

with en ∈ P(E), em'∈ P(E').

Entities in both layers are subject to management (managed objects) and are accompanied by management objects (mo for e, and mo' for e'). Mo' are *not* virtualized mo. They are instrumentations that have to be brought into management domains separately. In both layers, entities are accompanied by separate management objects that are providing the interface to the management system.

The management system models the partitioning into layers as separate management domains MD and MD' with management objects (mo ∈ MD, mo' ∈ MD').

Since monitoring and control tasks are to be performed upon entities e and e' through management objects mo and mo', management objects are connected with associated entities through a management interface. Examples are SNMP or other management protocols. Management instructions received from the management system are translated in management objects into corresponding interactions with associated entities.

An additional management interface must be provided by the virtualization layer to management objects mo' in the virtualized domain MD' since part of control of virtualized entities is provided by the virtualization layer. Management objects mo in the underlying layer may also have access to the management interface of the virtualization layer. This is not required when this layer should be kept unaware of virtualizations created above.

The virtualization layer itself should be integrated under the control of the encompassing management system, and not exist separately.

6.1 Inner-Layer Management

Since layers established by virtualization are modeled as separate management domains. Traditional management techniques apply with managed (e) and management objects (mo) within each layer.

The only specific property in MD' is that management objects require control over their associated managed entities that is provided by the virtualization layer and hence has to be exercised through a management interface of the virtualization layer.

6.2 Cross-Layer Management

Cross-layer management is a more interesting case where management tasks have to be performed that span across layers of underlying physical entities and created virtualized entities. Examples of such tasks (use cases) include:

- Replace an underlying (e.g. a machine), but take care of arrangements in the virtualized layer (e.g. affected applications running on that machine) before making that replacement.
- Identify which bindings to underlying physical entities have to be resolved when a virtualized entity migrates.
- How can measurements in underlying physical entities be correlated with entities in the virtualized layer such as applications running on a server device at time t.

Since the virtualization layer maintains the associations between underlying and virtualized entities, cross-layer management requires tracking those relationships.

Since association may change, associations are depending on time.

6.3 Time-dependence of Associations

The first two cases only require knowledge about current associations. The third case may also include knowledge about associations of prior times, or even future times when associations have already been determined. Maintaining information about associations in the past would require that records about transitions altering associations would have to be kept in order to recall that information later for any given point in time.

6.4 Association Interface

The association interface plays an important role for cross-layer management. The association interface is attached to the virtualization layer and allows obtaining associations (including attributes of associations) between virtualized entities and underlying physical entities. The interface can be equipped taking time-dependence of associations into account (assumed here).

The association interface consists of two functions. One function is resolving a given underlying entity e into a set of virtualized entities e' to which associations exist at time t. The other function performs in the reverse direction resolving a given virtualized entity e' into a set of underlying entities e to which associations exist at time t:

$f(e,t) \rightarrow \{e'_{assoc}\}$ at time t,

with $e \in P(E)$, $e'_{assoc} \in P(E')$

$f(e',t) \rightarrow \{ e_{assoc}\}$ at time t,

with $e_{assoc} \in P(E)$, $e' \in E'$

(when no time-dependence is supported, the current association is referred to).

Each management domain (MD and MD') has additional functions to resolve (managed) entities into associated management objects (mo):

- $f(e) \rightarrow \{mo_e\}$, with $e \in E$, $mo_e \in MD$,

and the reverse:

- $f(mo_e) \rightarrow \{e\}$, with $e \in E$, $mo_e \in MD$.

With these primitives, association chains between virtualized and underlying entities can be tracked across management domains for cross-layer management purposes.

For example, in order to identify all management objects in MD' that represent virtualized entities depending on a given underlying entity e, the following invocation chain resolves e into desired {mo'}:

- in VL: $f(e,t) \rightarrow \{em'\}$,
- in MD': $\forall e' \in em'$: $f(e') \rightarrow \{mo'_e\}$,
- or combined: $f(f(e,t)) \rightarrow \{mo'_e\}$.

(VL stands for Virtualization Layer.) Determining management objects mo' in the virtualized layer based on entities e from the underlying layer is useful for case 1 in section 6.2 when management operations have to be performed in entities e' in MD' that depend on e in the underlying layer, and when the underlying entity e has to be changed or replaced affecting virtualized entities above.

This example shows how cross-layer management tasks can be performed referring to information about associations in the virtualization layer. The virtualization layer was extended by the proposed interface for this purpose. Similar techniques can be used for translating identities, names and addresses of entities and accompanying management objects between management domains.

7. APPLICATIONS

The virtualization capabilities enabled by web services are being used in the domain of utility computing and grid computing.

7.1 Utility Computing

Businesses are carefully examining all IT expenditures. Many firms are finding that their IT infrastructure is too inefficient and unresponsive to meet the needs of a dynamic business and is not aligned with business needs. The concept of utility computing is to provide IT with the tools to apply computing resources more like an electric utility. In this service-driven model, computing resources are dynamically allocated to meet demands, and systems are increasingly self-managed to maximize flexibility and ease of administration. The result is that IT can be run as a service to the business, and excess capacity can be reduced.

The term *Utility Computing* embraces the vision of accessing and supplying standardized (and commoditized) computational resources as services anywhere and anytime and without a difference in time or location, similarly like other utilities such as water or electricity. Pervasive computing or "on-tap" computing are other term used in this context.

However, there are differences between a water or a power utility and a computational utility. Using power from an outlet only relies on quantitative parameters, no matter where the power has been produced or how it is distributed though the network. In a sense, the power utility system is stateless to the consumer. Computational services typically have state associated. This state needs to be made accessible to a specific computational facility in order to provide the desired service for a consumer. Providing that state to any computational facility is a major technical hurdle for achieving true utility computing.

Behind the vision, two characteristics are associated with utility computing today:

- The technical separation of resources provided in a computational facility (such as a data center) from the state (applications, data) of a particular consumer which only in their combination allows to perform the desired computational services.
- The separation of ownership of the computational facility and the service customer.

The first aspect, the separation of resources from consumers' state is supported (enabled) by virtualization today. Virtualization of resources allows separating state (applications and data) from machines and other parts of a computational facility. The second aspect is organizational, the separation of roles of the owner of computational resources and all associated facilities, such as buildings, networks, and infrastructure, from the consumer (and customer) of computational services. The customer pays for the computational services, which includes the portion for using computational resources and infrastructure. The advantage for the customer is to avoid binding capital in infrastructure, equipment and all efforts for maintenance and evolvement.

Depending on how contractual terms are arranged, the customer may choose from a variety of alternatives such as:

- a fixed rate for a certain contingent of resources used over time,
- *pay-per-use*, determined by actual use or resources, or
- pay for a certain *Quality of Service*.

Rates depending on usage require a metering and accounting infrastructure. Both models also exist in traditional utility environments (for example, local phone service typically provided with a fixed rate in the U.S., and long distance phone service, power or water utilities typically metered and paid as used).

Another advantage of applying the utility model to computing is that the utility provider can share facilities among different customers allowing for more efficient utilization and lower prices. However, sharing in compute utilities introduces security risks that need to be addressed. Again, the problem is related to the state associated to individual customers that need to be protected from undesired access, damage or loss.

7.1.1 The Stages Towards Utility Computing

Three areas define the modern computing industry: the mainframe era, which revolutionized business by automating the back office; the client/server era, which marked the beginning of departmental automation; and the current stage, the network era where computational services are arbitrarily accessible.

Each of these eras has been marked by businesses worldwide embracing new ways of computing to transform their operations. No change has had more impact than the Internet, which itself can be seen in three distinct phases.

Data and Information Utility. In the first phase, companies began offering access to information on simple web sites. Consumers could look up information everywhere from flight information to bank account balances. In most cases, web sites did little more than replicate data that was stored in central systems. Information became more widely available, but for the most part remained static, limiting its utility. This stage can be called Data and Information Utility.

Service Utility. The second stage can be described as integrating data and computation to personalized services. Services such as storing data, collaborating over distances over the Internet provide more than just access to data and can thus be called service utilities.

Business Utility. In the third phase, the Internet became a medium for business transactions. Banks enabled customers to move money among accounts. Airlines allowed online reservations. As companies integrated internal systems and business processes behind the scenes, transactions of all kinds were made possible. Information became more highly actionable.

More than a quarter of all large firms (1000+ employees) and over half of the world's largest companies today can be seen in this second stage. They are building seamlessly integrated end-to-end business processes that allow a wide range of new interactions among their various constituencies.

7.1.2 Utility Computing Infrastructure

Industry analysts suggest that customers' push for utility computing originates primarily from the human costs associated with maintaining large, complex IT environments and the resulting lack of agility. To address this problem large vendors HP, IBM, and SUN, as well as a number of startups have announced initiatives or products that aim for utility computing. Many of these are attempting to improve IT utilization by various virtualization techniques, while others provide configuration and provisioning capabilities, or aim to monitor and manage the IT environment more efficiently.

Utility computing can be defined as the ability to provide complex computing environments on-demand to IT customers. Achieving utility computing is difficult because the needs of enterprise users are complex. Each application running within the enterprise has unique assumptions, each enterprise has different policies that are associated with its applications, and each customer brings a different set of requirements for their application.

Providing infrastructure that supports these diverse requirements is inherently difficult. While techniques such as virtualization or load balancing are necessary for utility computing, they are not sufficient. To be successful, a utility computing system must

- support the design, deployment, and management of arbitrary applications while dealing with their frequently competing requirements for resources.
- It must accommodate both user and operator policies on how infrastructure is used and
- deal with upgrades of both the infrastructure and the applications.
- And it must maintain a high level of automation to reduce errors and manage costs.

Problems must be addressed from the perspective of IT customers. That is, given customer-specified requirements for a complex application, how can the corresponding system specification be automatically created and the system provided to the customer on-demand? The goal is to provide customers and IT service providers with tools that support the entire lifecycle of a computing task, including the design, deployment, operation, and decommissioning of that task while maintaining flexibility, agility, and cost efficiency within the utility through automation.

The term service oriented architecture (SOA) has been introduced to describe the overall approach of building loosely coupled distributed systems with minimal shared understanding among system components. A Web service is considered a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (such as WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. A service-oriented architecture defines a distributed system wherein agents, known as services, coordinate by sending messages.

Utility computing requires a high degree of interoperability among the participants of a utility systems, at the access level as well as at the meta-level, where descriptive information about utility resources are exchanged. Employing a service-oriented architecture for utility systems basically means modeling a system as a distributed system in which all entities are seen as services.

7.2 Web Services in Grid

Grid computing emerged as a paradigm of sharing resources for collaboration and resource usage optimization purposes. A Grid is made up of a finite set of nodes. A node is a system that manages a set of resources. A node may be a single system or a cluster. A resource being managed can be a network, system, or an application. Being mostly used in academic environments, "best-effort" was (and is) a sufficient policy for committing resources to users performing their computational workload.

The Open Grid Services Architecture (OGSA) presents a new vision for the grid. Motivation was that in both, e-business and scientific computing environments services need to be integrated across distributed, heterogeneous, dynamic "virtual organizations" formed from disparate resources within a single enterprise and/or from external resource sharing and service provider relationships. Building on concepts and technologies from the grid and Web services communities, the OGSA defines a uniform exposed service semantics also referred to as the grid service. OGSA defines standard mechanisms for creating, naming, and discovering transient grid service instances. It provides locations transparency and multiple protocol bindings for service instances, and it supports the integration with various underlying platform facilities.

OGSA defines grid service interfaces in terms of the Web Service Description Language (WSDL) with associated conventions and mechanisms required for creating and composing complex distributed systems, including lifetime management, change management, and notification.

OGSA defines the concepts how grid functionality can be incorporated into Web services within and across organizational domains. The notion of a *Virtual Organization* facilitates the workflow of a group of users across multiple domains who share (some of) their resources to solve particular classes of problems.

OGSA itself is structured by a variety of services. All grid service execution or communication relies on basic transport and security functions. These functions are defined within the Grid service specification as binding properties, meaning that a particular service implementation may choose to implement them using any protocol.

The Grid conceptual architecture is presented in Figure 92. The Grid deployment infrastructure is where the resources (applications/software, machines, networks, farms) are provisioned. The OGSA infrastructure which is based on a Web services infrastructure (.Net, J2EE based), provides the basic functionalities that deal with creation through factory, life cycle management, obtaining management related information through manageability interfaces, and other services like notification and invocation. The OGSA meta-services are higher-level functionalities that deal with provisioning/allocation, clustering, policy specification, security and problem determination. Applications can be defined on top of the meta-services. The following figure shows the OGSA basic services built above an infrastructure.

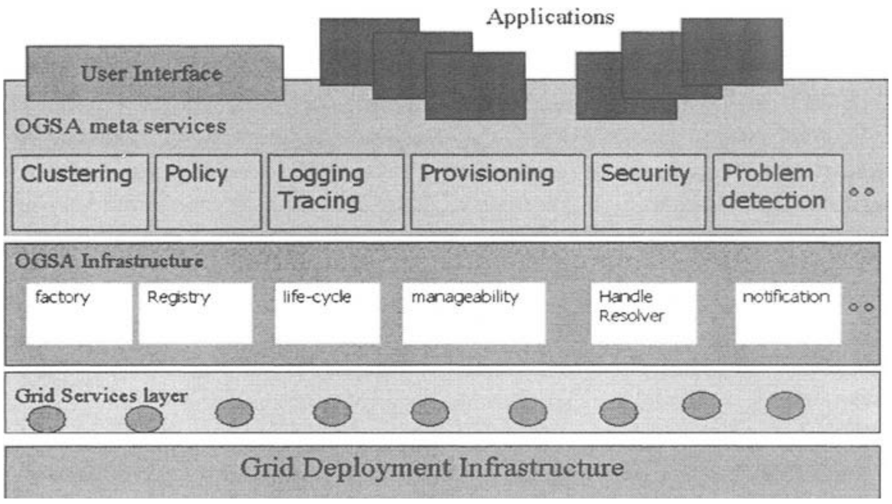


Figure 92: Open Grid Services Architecture (OGSA).

OGSA defines services to be executed within a hosting environment that is similar to such environments in Web services. Standard interface definitions such as those defined within the Grid service specification allow two services to interoperate. They do not address the portability of service implementations. Work is required to define standard hosting environments in order to enable portability. The following are examples: Within a J2EE environment, standardized Java APIs can be defined to allow for portability among OGSA-enabled J2EE systems. Entropia, United Devices, and Condor allow untrusted (and untrusting) desktop systems to participate in distributed computations. A standard "desktop" hosting environment would allow for interoperability among these different systems. The TeraGrid project is defining standard "execution environments" for computers that run scientific applications. These execution environments assume Linux and define conventions for the locations of key executables and libraries, and for the names of certain environment variables.

7.2.1 Basic Interfaces and Behavior

The following enumeration represents a set of interface and behavior definitions that appear particularly fundamental to the creation of interoperable Grid systems. The inclusion of an interface in this list is not in any way a binding categorization.

Common resource models. The OGSA common resource model enables consistent distributed management and access to these resources without having to understand the details of implementation of the resources, whether they are instrumented in CIM or SNMP or MDS/Glue, etc.

Registry and service discovery. OGSA describes a registry service as one of infrastructure services to support the registration, and subsequent discovery, of service instances. One or more standard registry behaviors need to be defined to permit service discovery in various settings.

Handle Resolution. OGSA requires an infrastructure service that enables handle resolution for dynamic invocation of services.

Service domain. In what seems likely to be a common architectural approach, an OGSA-compliant "service" is implemented via a collection of internal services that are managed in some coordinated fashion. Standard interfaces and behaviors need to be defined to facilitate the creation and operation of, and the integration of new services into, such service domains.

Policy. A Policy is a definitive goal, course or method of action based on a set of conditions, to guide and determine present and future decisions. Policies are implemented or executed within a particular context (such as policies defined for security, workload management, network quality of service, etc.). They provide a set of rules to administer, manage and control access to Grid resources. Policy Services are required to provide a framework for creating, managing, validating, distributing, transforming, resolving, and enforcing policies within a distributed environment.

Security. Requirements here are wide reaching. Fortunately, a substantial effort has already started within the OGSA Security WG on an OGSA security roadmap that defines requirements, relationships to other standards efforts (e.g., WS Security) and priorities for early development.

Distributed data management services, supporting access to and manipulation of distributed data, whether in databases or files. Services of interest include database access, data translation, replica management, replica location, and transactions.

These foundation functions and services are discussed in more detail below. Other service not further discussed should be mentioned:

Workflow services, supporting the coordinated execution of multiple application tasks on multiple distributed Grid resources.

Accounting/auditing services, supporting the recording of usage data, secure storage of that data, analysis of that data for purposes of billing, fraud and intrusion detection, and so forth.

Monitoring services support the discovery of "sensors" in a distributed environment. The collection and analysis of information from these sensors is supported as well as the generation of alerts when unusual conditions are detected, and so forth.

Problem determination services for distributed computing, including dump, trace, and log mechanisms with event tagging and correlation capabilities.

Clustering services for grouping and management of distributed peer service instances in order to provide coordinated management actions such as disaster recovery and load balancing, through dynamic join/leave semantics and ordered message and event delivery.

Security protocol mapping services, enabling distributed security protocols to be transparently mapped onto native platform security services for participation by platform resource managers not implemented to support the distributed security authentication and access control mechanism.

Recently Web service Resource Framework has been proposed as the specification for defining the management related interfaces to grid services.

SUMMARY

Just like Web services add complexity to the task of management, it also brings to the table a set of technologies that may simplify the task of management itself. By realizing a set of technologies for Web service description, discovery and invocation it has provided management community the capability to virtualize a wide array of network, system, application components. This has led to the application of Web services technology to the domain of grid computing and utility computing. Both these domains appear quite promising for the vision of Web services being used in management.

Appendix

WEB SERVICES MANAGEMENT PRODUCTS AND SOLUTIONS

1. INTRODUCTION

A wide variety of management systems and solutions exists and has created a mature industry for producing, deploying and operating large IT infrastructures. With the growth in IT systems, accompanying management systems also grew in complexity, diversity and scale.

Today, the market for IT management systems is dominated by large vendors such as IBM (mainly represented by the Tivoli suite) (IBM Tivoli), Hewlett-Packard (primarily with the OpenView suite) (HP OpenView), and Computer Associates (CA). These vendors cover the large space of IT management with comprehensive sets of products and solution portfolios.

A management system for an IT infrastructure is not a monolithic product. It is a portfolio comprised of different products and custom solutions, each covering a different aspect of management of an IT infrastructure. Vendors provide product portfolios from which customers or solution providers can chose for building tailored management solutions. Customizability is a main property of system management products.

As discussed earlier, product portfolios management systems vendors are following the trend from infrastructure and service management towards business-aware management. New management products are being primarily introduced in the higher service and business-aware management layers. Examples from the HP OpenView, IBM Tivoli product portfolios and new startups are discussed in sections 2 , 3, 4, 5, etc.

Due to the complexity and diversity of management product portfolios, we will present in this chapter only a selection of these products that represent the three main layers of management:

- Infrastructure management
- Services management, and
- Business-aware management.

For each management layer, two products from the dominant management product suites, HP OpenView (OV) and IBM Tivoli, are presented in more detail.

2. HP OPENVIEW MANAGEMENT SUITE

The following table provides an overview of the OpenView product and solutions suite. The highlighted products are discussed in more detail in the following sections, two for each management layer of infrastructure, service and business-aware management.

Product	Description
Advanced Security	Operations add-on to manage business critical data over insecure networks.
Cisco Integrations	HP OpenView Smart Way solutions and CiscoWorks Integration.
Continuous Access Storage Appliance	Storage application platform, enabled by virtualization technology, delivers local and remote data replication and migration capabilities across heterogeneous storage devices.

Customer Views for Network Node Manager (NNM)	Integrates with NNM to provide customer-based management of network environments.
Data Protector	Delivers new levels of recovery with a service-driven management approach.
Database Pak 2000	Manage the performance and availability of HP 9000 servers and databases in enterprise environments.
Dynamic Netvalue Analyzer	Real time IP solution that transforms raw usage data into business information.
Event Correlation Services	Correlate events from multiple system layers, integrated with NNM and OV Operations.
Event Correlation Services Designer	Transforms and processes event streams to correlate different event types messages.
Extensible SNMP Agent	Extends SNMP to control basic network devices, critical systems and applications.
GlancePlus	Performance monitoring and diagnostic tool providing immediate system information.
GlancePlus pak	Extends GlancePlus with historical data capabilities of Performance Agent Software.
Internet Services	Provides an integrated view of a network infrastructure.
Internet Usage Manager (IUM)	Open, multi-platform mediation and business intelligence solution.
IT Administration	Centralized user, software and system administration in heterogeneous environments.
IUM CDMA Solution	CDMS data billing mediation and voice switch mediation solution.
IUM GPRS Mediation Solution	A specialized set of extensions that process usage data from network equipment conforming to the General Packet Radio Service (GPRS) specification.
IUM Prepaid Data	A scalable, mediation solution for prepaid data management.
ManageX	Assures availability and optimal performance of Windows NT and Windows 2000.
Network Node Manager	Enables problem detection with statistics, alarms,

(NNM)	and maps of networks on a single display.
Network Node Manager Developer Toolkit	Development tools to create products that integrate with Network Node Manager.
Network Node Manager Extended Topology	Offers reduced problem resolution in complex networks with accurate views and automated problem analysis.
Omnistorage	Client/server-based solution providing virtually unlimited growth of file systems.
Operations Developer Toolkit	Toolkit for integration and instrumentation modules for OpenView Operations.
Operations for UNIX	A distributed solution that monitors, controls and reports on IT environment health (for UNIX).
Operations for Windows	A distributed solution that monitors, controls and reports on IT environment health (Windows).
OS/390 Management	Works with Operations to extend IT environment management to OS 390 domains.
OS/400 Management	Works with Operations to extend IT environment management to OS 400 domains.
Performance Manager/Monitor/Agent	Single interface for monitoring, analyzing, and forecasting resource utilization.
Performance Insight for Networks	Turnkey application that monitors and reports on network protocols and devices.
Problem Diagnosis	Provides status and performance information on static and dynamic network paths.
Reporter	Enables IT to provide timely and accurate reports to prove IT service quality levels.
Service Activator	Automates start and delivery of services offered by internal and external service providers.
Service Assurance/UNIX-based	Enables measurement, monitoring, and troubleshooting across all elements of a customer service transaction (for UNIX).
Service Assurance/Windows-based	Enables measurement, monitoring, and troubleshooting across all elements of a customer service transaction (for Windows).

Service Desk	Implements a helpdesk solution with problem, change, configure, and SLA management processes as a single workflow.
Service Information Portal	Create a portal view to show status information of a customer's IT environment.
Service Navigator	Manage applications and services from a business perspective with graphical views.
Service Quality Manager	Service level agreement and service quality management automation for a service provider.
Smart Plug-Ins (SPI)	Integrated instrumentation components that plug into HP OpenView products for extending a managed domain.
Software Distributor	Centralized UNIX software distribution capabilities.
Storage Accountant	Storage metering and billing for budgeting, financial analysis, and charge-back.
Storage Allocator	Virtualized storage access control and logical storage assignment.
Storage Area Manager	Integrated and centralized storage area management across distributed, multi-vendor storage resources.
Storage Builder	Storage capacity assessment, monitoring, and management.
Storage Management Appliance	A solution for monitoring and management of storage appliances.
Storage Media Operations	Automated tracking and management solution for productively managing the complete life cycle of removable storage media.
Storage Node Manager	Automated device discovery, topology mapping and centralized event monitoring, configuration and troubleshooting for storage systems.
Storage Optimizer	Storage performance assessment, monitoring, and management system.
Storage Provisioner	Capacity management and utilization tool that resolves unpredictable storage demands in complex IT environments.
Storage Virtual Replicator	Server-based virtualization tool that creates snapshots, provides storage administration for

	Windows NT/2000 environments.
Telecom Extensions for OV Operations	Enhances OV Operations with network element and EMS data, includes data collection and alarm management.
TeMIP Expert	HP OpenView TeMIP Expert provides expert system rules that capture the expertise and knowledge within network operations and automate operations processes to solve problems.
TeMIP Fault Management	HP OpenView TeMIP Fault Management and Real-Time Operations is an alarm-handling and event-logging network visualization solution in the TeMIP framework.
TeMIP Framework	HP OpenView TeMIP Framework provides a scalable set of telecom middleware services for applications to communicate in real time via the network and each other.
TeMIP Service Monitor	HP OpenView TeMIP Service Monitor provides Service Impact Analysis for diverse networks and services.
OpenView Transaction Analyzer (OVTA)	Facilitates web application problem resolution by identifying and directing attention to specific performance bottlenecks.
Trend Performance Manager	Repository and database engine that supports performance insight for networks.
Unified Developer Toolset	Using standard-based integration mechanisms, HP OpenView Unified Developer Toolset enables developers to create off-the-shelf applications that integrate with HP OpenView.
Web Transaction Observer	Monitors customer experience on specific URLs by integrating performance metrics.

Table 6: Overview of the OpenView Suite (Hewlett-Packard, 2003).

The following examples of products from the OpenView suite exemplifies the management space from infrastructure, service and business-aware management.

• *Infrastructure Management:*	- Network Node Manager (NNM),
	- Smart Plug-Ins (SPI),

• *Service Management:*	- Internet Usage Manager (IUM), - Service Navigator,
• *Business-Aware Management:*	- OpenView Transaction Analyzer (OVTA).

3. TIVOLI MANAGEMENT SUITE

Underlying the Tivoli management solution set is a group of common services and infrastructure that provide consistency across Tivoli management applications as well as enabling their integration.

Within the Tivoli product family, there are specific solutions that target four primary disciplines of systems management:

- Performance and Availability,
- Configuration and Operations,
- Storage Management, and
- Security.

Products within each of these areas have been created over the years and have become implemented as management solutions in enterprises around the world [Fearn 1999]. With these core capabilities, IBM has been able to focus on building management applications that take advantage of the pillars shown in Figure 93 from infrastructure management to business solutions. A typical business application depends not only on the hardware and networking, but also on software ranging from the operating system to middleware such as databases, Web servers, and application servers and messaging, to the applications themselves comprising a business service.

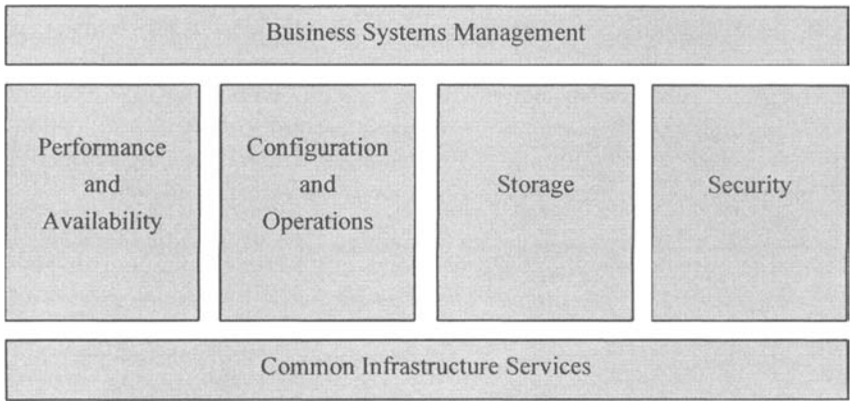

Business Systems Management
Performance and Availability
Configuration and Operations
Storage
Security
Common Infrastructure Services

Figure 93: Tivoli management structure.

The suite of Tivoli solutions allows an IT department to provide management of the entire business ecosystem in a consistent way, from a central site, using an integrated set of tools. By utilizing an end-to-end set of management solutions built on a common foundation, enterprises can manage and follow up with the increasing complexity of IT infrastructure with reduced staff and increased efficiency.

Figure 94 shows a vertical view to the Tivoli Management Architecture. At the lower level shown in Figure 94, monitoring products and technologies exist such as IBM Tivoli Monitoring in combination with resource models. At this layer, Tivoli monitors the hardware and software, and provides automated corrective actions when possible.

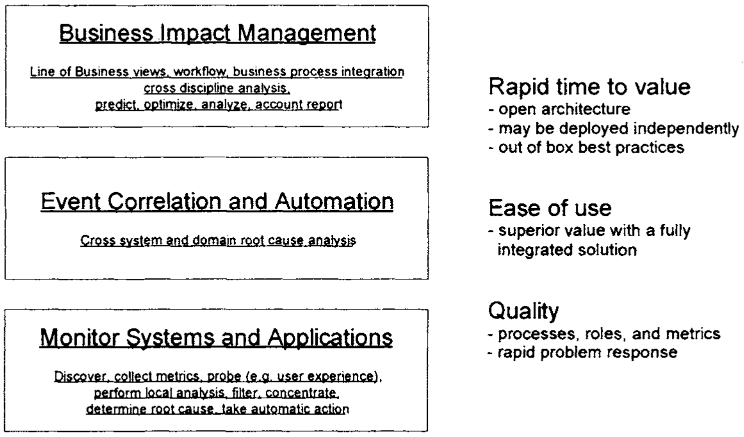

Business Impact Management
Line of Business views, workflow, business process integration
cross discipline analysis,
predict, optimize, analyze, account report
Event Correlation and Automation
Cross system and domain root cause analysis
Monitor Systems and Applications
Discover, collect metrics, probe (e.g. user experience),
perform local analysis, filter, concentrate,
determine root cause, take automatic action
Rapid time to value
- open architecture
- may be deployed independently
- out of box best practices
Ease of use
- superior value with a fully integrated solution
Quality
- processes, roles, and metrics
- rapid problem response

Figure 94: Vertical view to Tivoli Management.

At the next level is event correlation and automation. As problems occur that cannot be resolved at the monitoring level, event notifications are generated and sent to a correlation engine such as Tivoli Enterprise Console. The correlation engine at this point can analyze problem notifications (events) coming from multiple components and either automate corrective actions or provide the necessary information to operators.

The third tier in this structure is called Business Impact Management. It is important to know that a component or a related set of components have failed as reported by the monitors in the first layer. For instance, a router being down could cause database clients to generate errors if they cannot access the database server.

The third layer, Business Impact Management, is the most valuable, as it provides an insight into how a component failure may be affecting the business as a whole. When the router failure mentioned above occurs, it is important to understand exactly what line of business applications will be affected and how to reduce the impact of that failure on the business.

Tivoli Business System Manager (TBSM) and Tivoli Web Site Analyzer (TWSA) provide capabilities of business systems management because it provides the means to understand the business from the perspective of the IT infrastructure.

By sharing a common reporting interface with the other Tivoli components, Web site usage data can be correlated with operational data to discover relationships, which would otherwise be hidden to the organization. Examples of those relationships between the business patterns and the IT infrastructure supporting the business systems are:

- Web site activity as a function of the transaction response times.
- CPU and memory usage as a function of campaign launches.
- User shopping time as a function of database buffer pool utilization.
- Number of referrals as a function of network bandwidth.

In addition, external data such as weather data at various locations, stock quotes, currency exchange rates, and population data may be added to the information warehouse to allow for extended analysis of the usage patterns of a Web site, and the need for IT resources to support the associated business. In the "on-demand" world, these types of analysis are critical to be able to predict when load will peek – and more importantly, why. This will enable better control of resource use and qualified decisions on the trade-off between business and capital investment.

3.1 Tivoli Product Portfolio

Similarly like OpenView, the product portfolio of Tivoli is designed to cover a comprehensive space of management tasks. A summary of Tivoli management products is shown in the following table.

Tivoli Product Category with Products
Network Management:
Tivoli Switch Analyzer
Tivoli NetView
Tivoli NetView for z/OS
Tivoli NetView Performance Monitor
Storage Management:
Tivoli Storage Area Network Manager
Tivoli Storage Manager
Tivoli Storage Manager Extended Edition
Tivoli Storage Manager for Application Servers
Tivoli Storage Manager for Databases
Tivoli Storage Manager for Enterprise Resource Planning
Tivoli Storage Manager for Hardware
Tivoli Storage Manager for Mail
Tivoli Storage Manager for Space Management
Tivoli Storage Manager for Storage Area Networks
Tivoli Storage Manager for System Backup and Recovery
Tivoli Storage Resource Manager
Tivoli Storage Resource Manager Express Edition
Tivoli Storage Resource Manager for Chargeback
Tivoli Storage Resource Manager for Databases
Tivoli SANergy

Monitoring: Tivoli Monitoring Tivoli Monitoring for Applications **Tivoli Monitoring for Business Integration** Tivoli Monitoring for Databases Tivoli Monitoring for Messaging and Collaboration **Tivoli Monitoring for Network Performance** Tivoli Monitoring for Transaction Performance Tivoli Monitoring for Web Infrastructure
Performance and Workload Management: Tivoli Performance Modeler for z/OS Tivoli Workload Scheduler **Tivoli Workload Scheduler for Applications** Tivoli Workload Scheduler for z/OS
System Management: Tivoli System Automation for Linux Tivoli System Automation for OS/390 Tivoli Access Manager for Operating Systems Tivoli Analyzer for Lotus Domino Tivoli Data Exchange Tivoli Configuration Manager Tivoli Remote Control Tivoli Identity Manager Tivoli Intrusion Manager Tivoli License Manager Tivoli Risk Manager IBM Directory Integrator IBM Directory Server
Tivoli Decision Support for OS/390 Tivoli Decision Support Tivoli Enterprise Console

Service Management: **Tivoli Service Level Advisor** Tivoli Web Access for Information Management **Tivoli Web Site Analyzer**
Business-Aware Management: Tivoli Point-of-Sale Manager **Tivoli Privacy Manager for business** **Tivoli Access Manager for Business Integration** Tivoli Access Manager for business **Tivoli Business Systems Manager** Tivoli Business Systems Manager for z/OS

Table 7: Overview of the Tivoli Management Suite [IBM Redbook, 2003].

The following selection of products from the Tivoli suite represents the space from infrastructure management to service and business aware management. These products are discussed in the following sections.

• *Infrastructure Management:*	- Tivoli Storage Manager, - Tivoli Monitoring for Network Performance,
• *Service Management:*	- Tivoli Service Level Advisor, - Tivoli Web Site Analyzer,
• *Business-Aware Management:*	- Tivoli Monitoring for Business Integration, - Tivoli Business Systems Manager.

4. WEB SERVICE NETWORKS

If the commercial Web services are to become a reality and to automate real-world business on the web it is required that the Web service to Web service interactions be secure, manageable, and reliable. In order to do real business on the web these Web services need to specify, guarantee and meet contracts, QoS guarantees amongst each other. This represents the next frontier of Web service management. Some current providers are discussed in the following.

4.1 Grand Central Communications

In the 5.1 Grand Central Communications (GCC) model, one create Web services using existing tools, but rather than send messages directly between consumers and producers, everything passes through a GCC hub. Within that hub, GCC can perform a variety of transport-, session- and presentation-layer transformations and resolve incompatibilities. For instance, a producer and consumer may use different authentication or encryption schemes, or one party may use SOAP over HTTP while the other might send and receive using SMTP or even FTP. A GCC hub is based on IBM's MQ Series software, thereby providing QoS (reliable delivery), non-repudiation and other management features.

4.2 Flamenco Networks

Flamenco Networks (FN) takes a different approach. All parties using the FN network install proprietary software web-services proxy software that runs on any Java VM, dedicated or shared with other functions such as on an HTTP server. The producer's and consumer's proxies communicate directly with one another over the public Internet, not through a hub, but similar to a VPN. The traffic is compressed, encrypted and the proxies provide message queuing. Additionally, there's a management server that handles authentication, non-repudiation and management of the proxy servers. You can either license the management server or outsource this functionality to FN.

4.3 Kenamea

Kenamea focusing on adding network services that enhance the communication between the back end-the servers that support the application and any user device that needs to run the application. Kenamea adds an intelligent message queuing capability to the network that moves beyond the basic request/response nature of the Web to a transactional, event-driven model that can make Web applications more dynamic.

4.4 Talking Blocks

Talking Blocks (acquired by Hewlett-Packard in 2003) provides version management between Web services. They ensure that partners have the same software stack (middleware, language, interfaces, protocols, and versions) to be compatible. They also provide a central hub through which the interaction happens and where partners register their information. The hub ensures compatibility and evolution of the stacks by maintaining details of the versions.

Note that none of these vendors are involved at the application layer. They do not deal with business semantics such as orders or invoices. Furthermore, none of them address the requirements for transactions using either two-phase commit or based on compensating transactions. Until a protocol such as ebXML (ebXML) and BTP, (the Business Transaction Protocol) [Potts 2002] is adopted and implemented, transactions must be managed within the applications. As they do not have application layer semantics they do not know how to manage contracts between Web services.

5. CA UNICENTER

Computer Associates (CA) offers Unicenter for service and IT infrastructure management. Unicenter is a family of modular, integrated solutions addressing issues of managing complexity, maximizing IT staff productivity, and orchestrating resources and services across the enterprise.

The goal of Unicenter is to synchronize the IT resources with the constantly changing and evolving needs of the business. To achieve this, Unicenter provides the following:

- *Deliver IT as a Service* – by mapping IT to the businesses, providing business relevant management tools, and supporting IT as a service business.
- *Make Infrastructure Self Managing* – Unicenter has been extended to allow it to self-deploy, self-manage, and self-heal, all based on the policies defined by a customer. It includes dynamic resource management, and introduces a new concept called service awareness.
- *Service Oriented Architecture* –The new Management Data Integrator facility supports integration of in-house and third-party solutions, and a new role-based visualization capability.

Unicenter offers a broad set of integrated components for managing the health and availability of each aspect of a computing infrastructure. It assesses and manages the cost of these components in the enterprise, and it manages the services delivered to end-users.

6. ACTIONAL

Actional is another company that provides web services management solutions. Actional's Web services management platform is architected to help organizations manage the impact of constant change that is inherent in enterprise Web service networks. Actional delivers visibility, flexibility and active control on a Web services network. This level of control ensures high uptime for monitored Web services while reducing the costs during ongoing Web services management [Actional 2003].

Actional's active Web services management technology includes service brokers, active agents and a centralized management and policy server. It provides users with a complete view and active control of an entire enterprise Web services environment. Actional's approach enables users to understand the service network interdependencies; to know when the service network deviates from its normal operating range; to identify the root cause of problems; and to distribute active policy into the service network to both correct and prevent problems.

Actional's Web Services Management Platform provides centralized components for command, control and analysis across the service network as well as powerful in-network components that enforce policy and act upon in-flight messages.

The Actional Web Services Management Platform product suite includes:

Centralized components:

- Actional Looking Glass Console,
- Actional Looking SOA Planner,
- Actional Looking Glass MyServices,
- Actional Looking Glass Server.

In-network components:

- Actional SOAPstation Service Broker,
- Actional SOAPstation-XD,
- Actional Active Agents.

7. AMBERPOINT

AmberPoint provides a web service management solution for problems that arise because of web services based distributed systems spanning multiple computer systems and geographic locations. Management has to address intra-enterprise domains as well as across enterprises.

There is also a high degree of heterogeneity in Web services itself. Some Web services are built on .NET, some on Java and others are created from legacy assets.

The AmberPoint Web services management solution is characterized by following attributes:

- Fully distributed implementation.
- Non-Invasive, plug-and-manage approach.
- Content- and context-aware instrumentation.
- Active management.
- Dynamic and extensible.
- Performance and scalability.
- Native implementations in .NET and J2EE.

Fully Distributed Implementation. The AmberPoint Web services management solution is based on a distributed architecture for Web services management. Key components are agents, analytical servers and management consoles, all Web services-based and built to operate in a distributed fashion.

AmberPoint Agents are lightweight software applications that are deployed in a Web services network between Web services and their clients. Agents provide the mechanisms needed for instrumenting Web services. Agents enforce user-defined management policies for monitoring, auditing, logging, security, routing, transformation, failover or load-balancing. Agent can manage one or more Web services.

AmberPoint agents are built using standards-based Web services themselves. They form a common runtime environment for all products in the AmberPoint management suite. Once deployed, agents are visible to all AmberPoint management components. AmberPoint agents can be deployed on both .NET and J2EE Web services containers.

Flexible Agent Topologies. AmberPoint agents can be deployed in three modes: as in-server plug-ins, as external proxies and as so-called out-of-band T-Filters. The flexibility of its agent-based architecture allows AmberPoint to bring comprehensive management capabilities to every variety of a Web services system.

Decentralized Analytical Servers. AmberPoint utilizes task-specific servers for analyzing the data acquired by agents. These analytical servers analyze and distribute the workload between agents and servers. While agents capture and aggregate real-time metrics locally, the analytical servers aggregate this data across multiple agents, process the information and maintains the historical view of the data. Analytical servers are built on AmberPoint's agent technology and therefore can be distributed throughout the Web services environment in the same fashion. More analytical servers can be deployed as Web services systems evolve.

Decentralized User Interface. AmberPoint employs decentralized, desktop-based and web-based management consoles for monitoring and managing the Web services environment. AmberPoint user interfaces communicate with distributed agents and servers via Web services. Management consoles can be customized to different roles of users in managing segments of a Web services network.

Concurrent Policy Management. AmberPoint user interfaces and agents have built-in mechanisms for creating, enforcing and managing multiple, independent management policies. Additionally, management policies can be shared and reused. Different users can own role-specific management policies and administer them from their points-of-view. Similarly, different systems can issue different management commands to the AmberPoint management runtime environment.

AmberPoint's agents and analytical servers allow the flexible distribution of workload within an environment. Agents collect and aggregate data locally while analytical servers aggregate data across the Web services environment.

Non-Invasive, Plug-and-Manage Approach. AmberPoint solutions do not require to change the Web services or their clients. They do not require retrofitting code with proprietary controls, inserting proprietary headers or using proprietary APIs. One can deploy AmberPoint's solutions completely independently of Web services.

Content- and Context-Aware Instrumentation. AmberPoint agents contain instrumentation for capturing the operational and business metrics that are necessary for understanding the overall health of the Web services environment. Instruments can be created to track performance (such as the average response times or number of requests per hour) or to track business-related information (such as the number of purchase orders that contain a certain item). Together, operational and business insight derived from this instrumentation allows making decisions such as allocating additional resources to the Web services that process the greatest number of orders, or prioritizing fault handling based on this information – thus enabling aligning IT goals with business objectives.

Active Management. AmberPoint agents not only observe the Web services environment, but can also act upon message content. They can execute customized management actions upon receiving certain request, response or fault messages. Agents can execute actions immediately or on a schedule. Actions can also be triggered by individual instrumentations within agents or executed based on user-defined criteria. Actions can be used individually or combined executing complex management tasks.

8. CONFLUENT

Confluent (acquired by Oblix) overlays its policy-driven management platform over existing assets and infrastructure to consistently enforce the operational policies that are critical to successful deployments, such as security, encryption, service level management, logging, metering and change management.

Confluent's management platform consists of three components:

- Confluent Policy Manager,
- Confluent Monitor, and
- Confluent Enforcement Components—Agents and Gateways.

These modules in combination with Confluent's development environment extensions address the challenges enterprises encounter for application management, from development, to deployment, into operational management and finally into business analysis. Seamless coordination among policy enforcement, monitoring and business analytics enables Confluent's management platform visibility, control and adaptability over an entire suite of applications providing services on which business processes rely.

9. MICROSOFT APPLICATION CENTER

Application Center is Microsoft's deployment and management tool for high-availability Web applications built on the Microsoft Windows 2000 and Windows Server 2003 operating systems.

Microsoft Application Center addresses several requirements.

Microsoft Application Center enables developers and Web site administrators to deploy applications.

Furthermore, Microsoft Application Center also addresses operating applications in clustered environments. A clustered environment comprises a group of servers and turns it into a single, unified resource. By bringing the servers together into a cluster, many servers can be managed as easily as one. The idea of clustering is inherently built into Microsoft Application Center. It:

- Reduces application management complexity. Administrators can construct logical groupings including the contents, components, and configuration of applications. These groupings can be managed throughout the cluster, reducing application complexity.
- Manages many servers as one. When changes are made to a server, Application Center can automatically apply those changes to the other servers in a cluster.
- Streamlines application deployment. Application Center enables migration of applications through the development cycle. It ensures consistency between the stages across developing an application, testing the application, and finally migrating it into production.
- Software Scaling. Software scaling increases the capacity of an application by adding or releasing servers. While hardware scaling requires expensive, specialized servers, software scaling can be achieved using standard off-the-shelf servers. In addition, with software scaling, the relationship of cost to added capacity is close to linear.

Application Center enables creating and running a group of servers as operating a single server.

Its main advantages are:

- Accelerate cluster deployment. Traditionally, software scaling has carried a high barrier to entry, including a high cost in complexity and resources in getting applications to run on multiple servers as a unified resource.
- Enable software-scaled architecture. Application Center enables moving towards software-scaled architectures.
- Scale resources with business needs. Applications can handle increases and decreases in their capacity requirements by adding or removing servers.
- No new application programming interfaces (API) are required. Application Center offers the benefits of software scaling to existing applications, without requiring modifications.

Mission-Critical Availability. Application Center is designed in a way that any server might be brought down without affecting the availability of the overall application avoiding a single point of failure. This means applications can achieve mission-critical availability using off-the-shelf hardware without the need for cost-intensive high-availability hardware.

Microsoft Application Center:

- Ensures Availability. Application Center allows a Website to withstand software and hardware failures without disrupting service.
- Actively monitors performance and health. Application Center provides tools that monitor the cluster and its servers. It also allows views into performance and event-log data for one server or the entire cluster. Administrators can monitor applications remotely using a browser-based console.
- Automates event responses. Application Center can monitor server and application health and can take action in response to particular events and conditions. Automated responses provide faster response time, eliminate risk of human error and allow for higher overall application availability.

10. SERVICE INTEGRITY

Business managers need information faster in order to make decisions and operate efficiently. The Service Integrity Management Solution software monitors, analyzes, alerts and reports key business activities enabling to make timelier, informed decisions. The Service Integrity Management Solution platform helps reducing risks in business by providing visibility into business transactions.

The Service Integrity Management Solution technology provides a detailed understanding into the business context of XML and Web services messages passing through a Web services-based environment. It allows:

- Real-time visibility into service utilization.
- Template based and customizable querying capability for real-time analysis of message data.
- Access to historical information for reporting, graphing, trending analysis, and business impact analysis.

Service Integrity Management Solution operational capabilities include:

- Monitoring transaction performance through multiple tiers of Web services.
- Measuring performance of Web Service invocations at service and method level, and aggregated across a distributed system.

- Reporting SOAP invocation faults.
- Reporting operator defined error conditions.
- Seamless integration with Microsoft Operations Manager and other system management tools for event logging.
- Generating alerts based on historical baselines or service level violations.
- Outputting data to pre-defined logs, dashboard monitors, and reports.

Service Integrity Management Solution requires no costly application, hardware or network infrastructure changes or additions. Deployment and configuration of a Service Integrity Management Solution is supported by:

- Web service-based auto-discovery of components.
- On-the-fly graphical configuration and viewing of logged data.
- Wizards-driven, interactive creation of dashboard views.
- Configurable alerting based on SNMP for integration with existing management tools.
- Web services standards based implementation (XML, SOAP, and WSDL) built upon the same web services framework as the applications it manages.

11. THE UTILITY DATA CENTER (UDC) – INTEGRATED DATA CENTER RESOURCE MANAGEMENT

Large data centers are under increasing pressure to become flexible providers of IT services. This primarily means reducing costs and efforts to adapt IT infrastructure faster to changing business needs and changing demand conditions. This goal has been expressed by vendor visions of *Adaptive Enterprise* from Hewlett-Packard [HP Adaptive Enterprise] and *Autonomic Computing* from IBM [IBM Autonomic Computing].

Traditional IT infrastructure has become too static and too costly for many enterprises. To ensure that sufficient resources are available to support growth, IT planners have traditionally overbuilt and consequently underutilized data center resources and infrastructure. Data center infrastructure has thus evolved into such a complex collection of legacy systems that are hard to manage in a uniform fashion.

Cost of change is the fastest growing factor of infrastructure total cost of ownership. Enterprises have begun overcoming this barrier through consolidation: centralizing and reducing the number of assets. But consolidation does not address the challenge of shifting resources to the services that need them when they need them. This requires a complementary strategy of virtualization. While consolidation aggregates resources, virtualization enables different services to tap those resources when needed.

One way how virtualization is provided for the entire set of enterprise data center resources is the HP Utility Data Center (UDC). Leveraging consolidation, the UDC creates a single virtualized pool of computing resources, including servers, storage, appliances, and networks. All resources are physically wired together (cabled) once at the time when the UDC is installed. All subsequent provisioning is performed through the central control unit of the UDC, which has control over network and storage virtualization.

11.1 Resource Virtualization in the UDC

A so-called storage virtualization fabric connects storage elements to processing elements (servers) via a Storage Area Network (SAN).

Presently, the chief barrier to rapid change is that network resources are tied to physical installations. Excess server capacity or storage allocated to one service cannot be transferred to another without physically picking up and moving the resources, a time-consuming, expensive, and risky task.

The network virtualization fabric allows linking processing elements with storage attached together with private virtual LANs (VLAN). Processing elements connected though a VLAN then host the applications comprising an application environment, such as for a three-tier Web service implementation. VLAN also provide network isolation between different application environments ensuring that application environments cannot interfere with one another by accident or maliciously.

Three types of resources are virtualized:

- *Network Resources* – permitting the programmable rewiring of server machines and devices in a virtual LAN network. Virtual wiring is achieved by programming network switches connecting machines and programmatically connecting or removing machines to or from virtual networks.
- *Storage Resources* – provide logical volumes from storage units (typically RAID systems) containing entire disk images with all persistent states of application environments including file systems, bootable operating system images, application software, application data, etc. With programmability of the storage fabric,

disk images can be made appearing on SCSI interfaces of machines as local disks from where machines boot and applications can be launched.

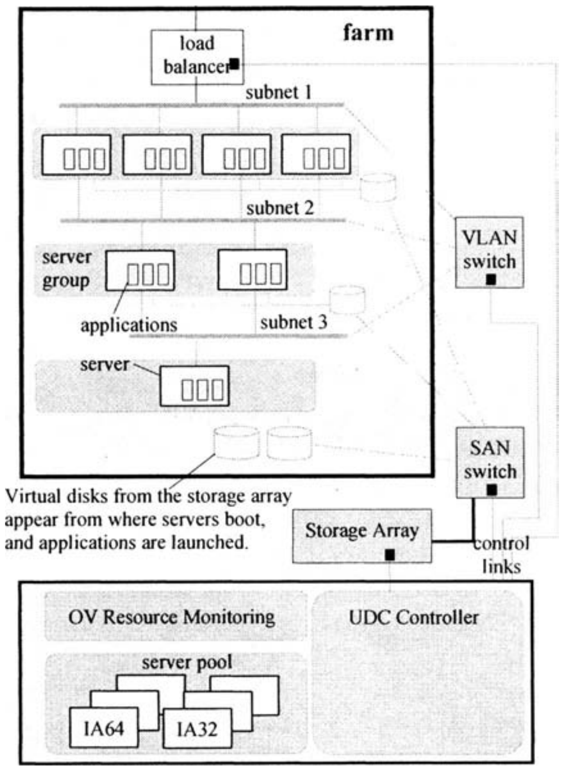


Figure 95: Resource virtualization in the Utility Data Center (UDC).

- *Server Virtualization* – server virtualization occurs in the sense that different (compatible) server devices can be assigned and re-assigned for hosting applications. The UDC makes the decision which machines from its machine pool (meeting compatibility and other constraints) will be assigned to applications. Although virtual machine software can be deployed in a UDC environment, the UDC itself does not provide a virtual machine environment.

11.2 The UDC Management System

The UDC management system controls the entire resource and service environment in combination with management software that ensures operations and monitoring support. The UDC management system allocates and re-allocates ("flexes") server resources within the UDC between applications according to fluctuating workload conditions.

Data center administrators operate the system from a browser-based UDC portal console to design, configure and dynamically reassign data center resources with drag-and-drop style of interaction. No physical changes are required in the data center.

Figure 96 shows the web-based operator console in which an application environment can be configured.

Activating new applications and their associated infrastructure components can thus be achieved significantly faster and at lower cost compared to traditional data centers.

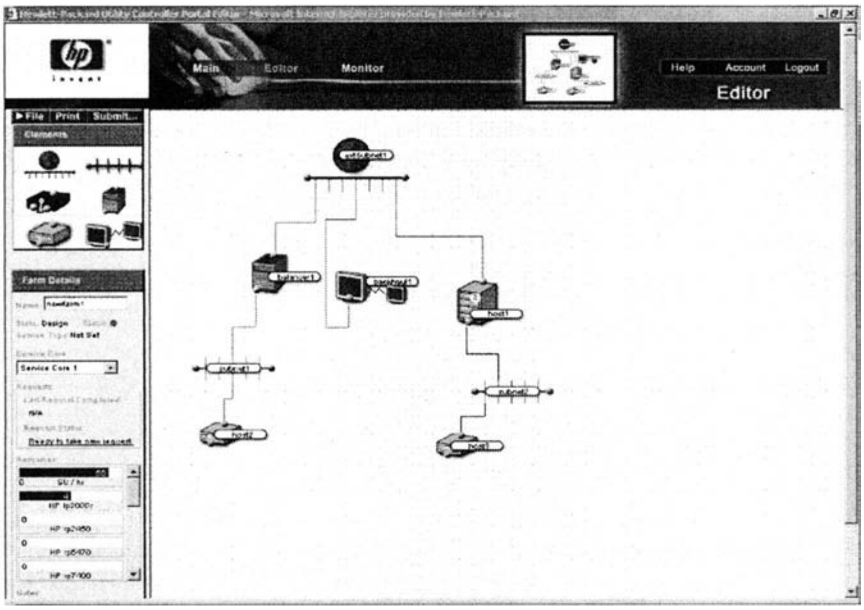


Figure 96: Operator console of the Utility Data Center showing the configuration of an application environment.

Temporarily suspended applications can be reactivated in the range of minutes and do not require reconfiguration or other action in order to be reactivated. For reactivation, servers and network resources are allocated from the resource pool and re-instantiated with the same state that has previously been captured when the application was suspended.

REFERENCES

Chapter 1 - Introduction

Leymann 2000, Frank Leymann, Dieter Roller. Production Workflows. Prentice Hall, 2000.

Cerami 2002, Ethan Cerami. Web Services Essentials. O'Reilly. 2002.

Pitts 2000, Natanya Pitts and Cheryl Kirk. The XML Black Book. Coriolis Technology press, 2000.

Box, Don Box, David Ehnebuske, Gopal Kakivaya. Simple Object Access Protocol (SOAP) 1.1. Available from http://www.w3.org/TR/SOAP.

Gartner 2001, Milind Govekar. Managing Total Business Integration. *Procs. of the Garner Symposium ITXPO*. Cannes, France. Nov. 2001.

BusinessWeek 2002, Kerstetter, J. The Web at your service. BusinessWeek e.biz cover story. March 18, 2002. http://www.businessweek.com/magazine/content/02_11/b3774601.htm.

Daniel Menasce, Virgilio Almeida. Capacity Planning for Web Services. Metrics, Models, and Methods. Prentice Hall. 2002.

Gartner 2004, Whit Andres, Abrams C. Web Services Drive Market Convergence.

Chapter 2 – Overview of Web Services

Andrzejak, A., Graupner, S., Kotov, V., Trinks, H.: Self-Organizing Control in Planetary-Scale Computing, IEEE International Symposium on Cluster Computing and the Grid (CCGrid), 2nd Workshop on Agent-based Cluster and Grid Computing (ACGC), May 21-24, 2002, Berlin.

Chaum, D.: Security Without Identification: Transaction Systems to Make Big Brother Obsolete, Communications of the ACM, Vol. 28, October 1985.

Digital Certificates, CCITT. Recommendation X.509: The Directory - Authentication Framework. 1988.

eCash, http://www.cryptologic.com/faq/faq-ecash.html, 2002.

ebXML http://www.ebxml.org.

Graupner, S., Kotov, V., Trinks, H.: Resource-Sharing and Service Deployment in Virtual Data Centers, IEEE Workshop on Resource Sharing in Massively Distributed Systems (RESH'02), Vienna, Austria, July 2002.

HP Utility Data Center (UDC), http://www.hp.com/go/hpudc, November 2001.

Kim, W., Graupner, S., Sahai, A.: A Secure Platform for Peer-to-Peer Computing in the Internet, 35th Hawaii International Conference on System Science (HICSS-35), Island of Hawaii, January 7-10, 2002.

Kuno H, Sahai A. My Agent Wants to Talk to your Service: Personalizing Web Services through Agents. HPL-2002-114.

IBM Autonomic Computing: http://www.research.ibm.com/autonomic.

Lee T. B. The World Wide Web: Past, Present and Future. http://www.w3.org/People/Berners-Lee/1996/ppf.html.

Microsoft .NET Passport, http://www.passport.com/, 2002.

Millicent, http://www.millicent.com/home.html, 2002.

Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org, 2002.

PKI Forum, http://www.pkiforum.org/, 2002.

RosettaNet, http://www.rosettanet.org, 2002.

Sahai A, Machiraju V, Ouyang J, Wurster K. Message Tracking in SOAP-Based Web Services. IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), April 2002, Florence, Italy.

Sahai A, Machiraju V, Wurster K. Monitoring and Controlling Internet based Services. The Second IEEE Workshop on Internet Applications (WIAPP'01), San Jose, July 2001 (also as HP Technical Report HPL-2000-120).

Sahai A, Ouyang J, Machiraju, V. End-to-End Transaction management for Web Based Services. Third International Workshop on Advanced issues of E-Commerce and Web based Information Systems (WECWIS), June 21-22 2000, San Jose USA (also as HP Technical Report HPL-2000-168).

SAML 1.0 Specification Set, http://www.oasis-open.org/committees/security/docs/draft-sstc-core-29.pdf, 2002.

Semantic Web, http://www.w3.org/2001/sw.

SET Secure Electronic Transactions LLC, http://www.setco.org/, 2002.

Simple Public Key Infrastructure (SPKI), SPKI Certificate Theory, RFC 2693, 1999.

T-Space at IBM. http://www.almaden.ibm.com/cs/TSpaces/. 1999.

The Liberty Alliance Project, http://www.projectliberty.org/, 2002.

The W3C Micro Payment, http://www.w3.org/ECommerce/Micropayments/, 2002.

Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. Available from http://www.w3.org/TR/wsdl.

Weber, R.: Chablis - Market Analysis of Digital Payment Systems, University of Munich, http://chablis.informatik.tu-muenchen.de/MStudy/x-a-marketpay.html, 1998.

Van Moorsel, A. Metrics for the Internet Age-Quality of Experience and Quality of Business. HPL-2001-179. http://www.hpl.hp.com/techreports/2001/HPL-2001-179.html.

WSCL Web Services Conversation Language, http://www.w3.org/TR/wscl10, 2002.

WSFL Web Services Flow Language (WSFL 1.0), Edited by F. Leymann, IBM, May 2001.

XML Key Management Specification (XKMS), http://www.w3.org/TR/xkms, March 2001.

XML Encryption Requirements, http://lists.w3.org/Archives/Public/xml-encryption/2000Oct/att-0003/01-06-xml-encryption-req.html, December 2000.

XML Signature Syntax and Processing, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212, 2002.

ISAPI, Internet Server API (ISAPI) Extensions. Available from http://msdn.microsoft.com/library/en-us/vccore98/html/_core_internet_server_api_.28.isapi.29_.extensions.asp

NSAPI, Netscape Server API (NSAPI) Programmer's Guide. Available from http://developer.netscape.com/docs/manuals/enterprise/nsapi/index.htm.

WSRF 2004, GGF author team, The WS-Resource Framework, http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf.

OGSA 2004, Foster, I., Kesselman, C., Nick, J., Tuecke, S., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, 2002. http://www.globus.org/research/papers/ogsa.pdf.

OGSI 2004, Open Grid Services Infrastructure (OGSI) V1.0, http://forge.gridforum.org/projects/ggf-editor/document/draft-ogsi-service1/en/1,

Foster 1999, Foster, I., Kesselman, C. (Eds.): The Grid – Blueprint for a New Computing Infrastructure, Morgan Kauffmann Publishers, 1999.

W3C-WSA 2003, The W3C Web Services Architecture working group, draft, August 2003. http://www.w3.org/TR/2003/WD-ws-arch-20030808.

Burbeck 2000, Burbeck, S., The Tao of e-business Services, http://www.ibm.com/developerworks/webservices/library/ws-tao, October 2000.

WSRF-Model 2004, GGF author team, Modeling Stateful Resources with Web Services, http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.html.

WS-Resource 2004, Modeling Stateful Resources in Web Services. http://www106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf.

WS-Addressing 2004, Don Box, et al., Web services Addressing, March 2003, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-addressing.asp.

WS-ResourceProperties 2004, Steve Graham, et al., Web Service Resource Properties version 1.1, January 2004, http://devresource.hp.com/drc/specifications/wsrf/WS-ResourceProperties-1-1.pdf.

WS-Notification 2004, Publish-Subscribe Notification for Web Services, whitepaper, http://www-106.ibm.com/developerworks/library/ws-pubsub/WS-PubSub.pdf.

Chapter 3 – Application Management and Web Services

Black 1994, Black, Network Management Standards: SNMP, CMIP, TMN, MIBs and Object Libraries, McGraw Hill Computer Communications Series, 1994.

CIM, Common Information Model (CIM) Standards, Distributed Management Task Force, http://www.dmtf.org/standards/cim.

CMIP, OSI Common Management Information Protocol (CMIP), Specification, 1998, http://www.iso.ch.

CMIS, OSI Common Management Information Services (CMIS), Specification, 1998, http://www.iso.ch.

CORBA, Common Object Request Broker Architecture, Specification, Object Management Group, 1999, http://www.corba.org, http://www.opengroup.org/infosrv/Brand/SPS_pdf/corba.pdf.

DMTF, Distributed Management Task Force, http://www.dmtf.org.

Parlay, http://www.parlay.org.

Halstone 2002, Performance Challenges of Web Services Application Architecture and SLA Management, IDC Report 28298, November 2002.

ISO9000, International Standardization Organization (ISO), Quality Management Standards, 1993, http://www.iso.ch/iso/en/iso9000-14000/iso9000.

OMG, The Object Management Group, http://www.omg.org.

OSI, International Standardization Organization (ISO), Open Systems Interconnect (OSI) standards, http://www.iso.ch.

RFC 2571, Harrington, Presuhn, Wijen, An Architecture for Describing SNMP Management Frameworks, IETF RFC 2571, April 1999, http://www.ietf.org/rfc/rfc2571.txt.

RFC 3411, Harrington, Presuhn, Wijen, An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, IETF RFC 3411, December 2002, http://www.ietf.org/rfc/rfc3411.txt?number=3411.

WBEM, Web-Based Enterprise Management (WBEM) Initiative, Distributed Management Task Force, http://www.dmtf.org/standards/wbem.

Chapter 4 – Enterprise Management and Web Services

Booch 1998, Booch, Rumbaugh, Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 1998.

[Calo 2003] Seaphin Calo, Morris Sloman. Policy Based Management of Networks and Services. V11N3, September 2003.

CIM, Common Information Model (CIM) Standards, Distributed Management Task Force, http://www.dmtf.org/standards/cim.

DMTF, Distributed Management Task Force, http://www.dmtf.org.

Dyche 2001, Dyche, The CRM Handbook, Addison Wesley, 2001.

eTOM 2003, The TeleManagement Forum's Enhanced Telecom Operations Map (eTOM), Michael B. Kelley, Journal of Network and Systems Management, V10N1, March 2003.

Hegering 1999, Hegering, Abeck, Neumair, Integrated Management of Networked Systems: Concepts, Architectures, and Their Operational Application, Morgan Kaufmann, 1999.

Hegering 2001, Hegering et.al., Towards Generic Service Management Concepts — A Service Model Based Approach, In Proceedings of the 7th International IFIP/IEEE Symposium on Integrated Management (IM 2001), 719–732, Seattle, Washington, USA, May 2001.

ITIL, IT Infrastructure Library, http://www.itil.co.uk.

ITSM, IT Service Management, The ITIL and ITSM Directory, http://www.itil-itsm-world.com.

itSMF, IT Service Management Forum, www.itsmf.com.

Martinka 1997, Matinka, Pruyne, Jain, Quality-of-Service Measurements with Model-based Management for Networked Applications, Hewlett-Packard Technical Report, HPL-97-167R1, 1997.

[Pavlou 1998] George Pavlou, Olivier Festor. Network Management Information Modeling. Journal of Network and Systems Management. V6N3, September 1998.

Sloman 1996, Network and Distributed Systems Management, Addison Wesley, 1996.

TMF, TeleManagement Forum (TM Forum), http://www.tmforum.org.

WBEM, Web-Based Enterprise Management (WBEM) Initiative, Distributed Management Task Force, http://www.dmtf.org/standards/wbem.

Chapter 5 – Managing Web Services From an E-Business Perspective

Baldrige 1997, National Institute of Standards and Technology, Baldridge National Quality Program Criteria for Performance and Excellence, http://www.quality.nist.gov/Criteria.htm.

Cutler 2001, Cutler, Sterne, Reiner, Business Metrics for the New Economy, Whitepaper, SPSS Inc., 2001.

Forrester 2000, Schmitt, Manning, Paul, Roshan, E-Commerce Software Takes Off, Forrester Report, March 2000.

Kaplan 1996, Kaplan, Norton, The Balanced Scorecard: Translating Strategy into Action, Harvard Business School Press, 1996.

Machiraju 2002, Machiraju, Sahai, van Moorsel, Web Services Management Network: An Overlay Network for Federated Service Management, Hewlett-Packard Technical Report, HPL-2002-234, 2002.

[Ray 2003] Praddep Ray, Integrated Management from E-Business Perspectives: Concepts, Architectures and Methodologies (ISBN 0-306-47485-9)

Reiner 2001, Reiner, The NetGenesis Enterprise Architecture, Whitepaper, NetGenesis Corp., 2001.

SOAP, Simple Object Access Protocol, Gudgin, Hadley, Mendelsohn, Moreau, Nielsen, , SOAP Version 1.2 Part 1: Messaging Framework, W3C, http://www.w3c.org/TR/SOAP.

XML, Extensible Markup Language, W3C, http://www.w3c.org/XML.

Chapter 6 – Managing Applications and IT Infrastructure of Web Services

BEA WebLogic Platform, BEA Inc., http://www.bea.com.

Darmawan 2003, Darmawan, IBM Tivoli Monitoring for Web Infrastructure: Managing WebSphere Application Server on z/OS, IBM Redbook, April 2003.

DSI, Dynamic Systems Initiative, Microsoft, http://www.microsoft.com/dsi.

EJB, Enterprise JavaBeans Technology, Sun Microsystems, http://java.sun.com/products/ejb.

Foster 2002, Foster, Kesselman, Nick, Tuecke, The Physiology of the Grid – An Open Grid Services Architecture for Distributed Systems Integration, 2002, http://www.globus.org/ogsa.

GGF, Global Grid Forum, http://www.ggf.org.

HP Adaptive Enterprise, Hewlett-Packard, The HP Vision for the Adaptive Enterprise: Achieving Business Agility, Whitepaper, Hewlett-Packard, 2003, http://h71028.www7.hp.com/enterprise/cache/7504-0-0-0-121.aspx.

HP OpenView, Hewlett-Packard, http://www.openview.hp.com.

IBM Autonomic Computing, IBM, Architectural Blueprint of Autonomic Computing, April 2003, http://www-306.ibm.com/autonomic/blueprint.shtml.

IBM Tivoli, IBM, http://www.ibm.com/tivoli.

JDBC, Java Database Connectivity, JDBC4.0 API Specification, Sun Microsystems, http://java.sun.com/products/jdbc.

Lindholm 1999, Lindholm, Yellin, The Java™ Virtual Machine Specification, 2nd Edition, Addison Wesley, 1999.

.NET, Microsoft .NET Framework, Microsoft, http://msdn.microsoft.com/netframework.

MTS, Microsoft Transaction Server, Microsoft, http://www.microsoft.com/com/tech/MTS.asp.

N1, N1 Grid: Managing n Computers as One, Sun Microsystems, http://www.sun.com/n1.

OV NNM, HP OpenView Network Node Manager, Hewlett-Packard, http://www.openview.hp.com/nnm.

Rohm 2003, Online Analytical Processing (OLAP) With a Cluster of Databases, IOS Press, November 2003.

Sadtler 2004, Sadtler, Ganci, Griffith, Hu, Marhas, Will, Jones, WebSphere Product Family Overview and Architecture, IBM Redbook, February 2004.

UDC, Utility Data Center, Hewlett-Packard, http://www.hp.com/go/udc.

WSDL, Chinnici, Gudgin, Moreau, Schlimer, Weerawarna, Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C, http://www.w3c.org/TR/wsdl20.

WSFL, Leyman, Web Services Flow Language, IBM, 2001, http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

Chapter 7 – Instrumentation of Web Services

ARM , Working group, *Application Response Measurement API guide* http://www.omg.org/regions/cmgarmw/index.html.

TIP, Evans, J. Klein, and J. Lyon *Transaction Internet Protocol – Requirements and Supplemental Information, 1998,* http://www.landfield.com/rfcs/rfc2372.html.

Park 1999, J. T. Park and J. W. Baek. *Web-based Internet/Intranet service management with QoS support.* IEICE Trans. Communications., e82-b:11, 1999.

Openview WTO, *Vantage Point Web Transaction Observer* , http://www.openview.hp.com/products/webtransobserver.

Kalbfleish, Kalbfleisch C et al. *Application Management MIB.* RFC 2564, http://www.simpleweb.org/ietf/rfcs/complete/rfc2564.txt.

RFC 2287, System Application MIB. RFC 2287, http://rfc2287.rfclist.org/rfc-2287.htm.

Stalling 1999, Stallings W. *SNMP, SNMPv2, SNMPv3, and RMON 1* and 2. Addison Wesley 1999.

Silvano 1995, Silvano Maffeis, *Adding Group Communication and Fault-Tolerance to CORBA.* In the proceedings of Usenix Conference on Object Oriented Technologies, 1995.

Sahai 2000, Sahai A et al. Message Tracking in SOAP based Web Services. In the proceedings of IEEE/IFIP NOMS 2002.

Chapter 8 – Managing Composite Web Services

W3C , *World Wide Web Consortium's SOAP specification* http://www.w3.org/TR/SOAP.

Sturm 2000, Sturm R, Morris W. and Jander M. Foundations of Service Level Management. SAMS publication. 2000.

Sahai 2001, Sahai A, Durante A, Machiraju V. *Towards Automated SLA Management.* HPL-2001-310.

Daniel 2000, Daniel J, Traverson B, and Vignes S. *A QoS Meta Model to Define a Generic Environment for QoS Management.* Third International IFIP/GI Working Conference, USM 2000. Munich, Germany, September 12-14, 2000. In Proceedings Lecture Notes in Computer Science 1890 titled "Trends in Distributed Systems: Towards a Universal Service Market". Springer Verlag.

Bhoj 1998, Bhoj P, Singhal S, Chutani S. *SLA Management in a federated Environment.* HPL-98-203.

Lewis 96, Lewis D, Bjerring L. An inter-domain Virtual Private Network Management System. In the proceedings of NOMS 96.

Lewis 97, Lewis et al. Experiences in Integrated Multi-Domain Management. IFIP/IEEE International Conference on Management of Multi-Media Networks and Services, Montreal, Canada, 1997.

Hall 1996, Hall J (editor). Management of Telecommunication systems and Services: Modeling and Implementing TMN based Multi-Domain Management, Lecture Notes in Computer Science 1116, Springer-Verlag, ISBN 3-540-61578-4, 1996.

TMN, Telecommunication Management Network (TMN) at ITU-T. Formerly CCITT. http://www.itu.int.

Aurrecoechea 1998, Aurrecoechea, C., Lazar, A.A. and Stadler, R., Open Network Services for Management, IEEE Conference on Open Architectures and Network Programming, San Francisco, CA, April 3-4, 1998.

Long T P, Jong W B, Woon HJ. Management of service level agreements for multimedia Internet service using a utility model. IEEE communications Managezine Vol 39, no.5, May 2001.

Forbath T. Why and how of SLAs [service level agreements]. Business Communications Review, Vol 28. No. 2, Feb 1998.

Chatterjee BS, Sydir M, Lawrence T. Taxonomy for QoS specifications. In the proceedings of WORDS'97, February, 1997.

Lewis L, Ray P. Service Level Management: Definition, Architecture, and Research Challenges. In the proceedings of IEEE GlobeCom'99.

Hauck R, Reiser H. Monitoring of Service Level Agreements with Flexible and Extensible Agents. HP OpenView University Association (HP-OVUA) Plenary workshop, Bologna, Italy, 1999.

Tele Management Forum SLA Management Handbook, GB917, public evaluation version 1.5 , June 2001. http://www.tmfcentral.com/kc/repository/documents/GB917v1.5.pdf..

Samani M, Sloman M. Monitoring of Distributed Systems (A Survey). Imperial College Research Report DOC 92/93. Sept, 1992.

WSLA, IBM, http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

WS-Reliability, http://otn.oracle.com/tech/webservices/htdocs/spec/WS-ReliabilityV1.0.pdf.

WS-Transaction, http://www-106.ibm.com/developerworks/webservices/library/ws-transpec.

WS-Agreement at GGF, http://www.ggf.org.

WS-Security , http://www-106.ibm.com/developerworks/webservices/library/ws-secure.

Frolund 1998, Svend Frolund, Jari Koistinen. Quality of Service Specification in Distributed Object Systems Design. Distributed Systems Engineering Journal 5(4), Dec. 1998.

Chapter 9 – Management Using Web Services

Global Grid Forum Global Grid Forum, http://www.gridforum.org.

Globus Project, The Globus Project, http://www.globus.org.

Globus toolkit, The Globus Toolkit, http://www.globus.org/toolkit.

Foster 2002, Foster, I., Kesselman, C., Nick, J.M., Tuecke, S., The Physiology of the Grid – An Open Grid Services Architecture for Distributed Systems Integration, DRAFT, http://www.globus.org/research/papers/ogsa.pdf, May 2002.

Foster 2001, Foster, I., Kesselman, C., Tuecke, S., The Anatomy of the Grid – Enabling Scalable Virtual Organizations, International Journal of Supercomputing Applications, http://www.globus.org/research/papers/anatomy.pdf, 2001.

Sun Grid Engine, Sun Microsystems, The Sun Grid Engine, http://wwws.sun.com/gridware.

Platform, Platform Inc., http://www.platform.com.

Grid at IBM, IBM Blue Grid, http://www.ibm.com/grid.

GSI, Grid Security Infrastructure (GSI), http://www.globus.org/security/overview.html.

X.509, Internet X.509 Public Key Infrastructure, http://www.ietf.org/rfc/rfc2587.txt.

GRAM, The Globus Resource Allocation Manager (GRAM), http://www.globus.org/gram.

Czajkowski 98, K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, A Resource Management Architecture for Metacomputing Systems. Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, pg. 62-82, ftp://ftp.globus.org/pub/globus/papers/gram97.pdf, 1998.

RSL, RSL specification, http://www.globus.org/gram/rsl_spec1.html.

MDS, The Monitoring and Discovery Service (MDS), http://www.globus.org/mds.

Data Grid, The Globus Data Grid Effort, http://www.globus.org/gt2/datagrid.html.

OGSA at GGF, OGSA Working Group, Open Grid Services Architecture: A Roadmap, http://www.ggf.org/ogsa-wg/, November 2002.

UDC 2001, Hewlett-Packard, Utility Data Center, http://www.hp.com/go/hpudc, November 2001.

FML, Farm Markup Language (FML) Specification, April 2001.

Graupner 2002, Graupner, S., Kotov, V., Trinks, H.: Resource-Sharing and Service Deployment in Virtual Data Centers, IEEE Workshop on Resource Sharing in Massively Distributed Systems (ICDCS-2002), Vienna, Austria, July, 2002.

Andrzejak 2002, Andrzejak, A., Graupner, S., Kotov, V., Trinks, H.: Self-Organizing Control in Planetary-Scale Computing, IEEE International Symposium on Cluster Computing and the Grid (CCGrid), Berlin, Germany, May 21-24, 2002.

Appendix – Web Services Management Products And Solutions

CA, Computer Associates Inc., http://www.ca.com.

Darmawan 2003, Darmawan, D'Amico, Foo, Glasmacher, Nosbisch, Yiu, Tivoli Business Systems Manager Version 2.1 – End-to-End Business Impact Management, IBM Redbook, May 2003.

ebXML, Technical Architecture Specification v1.0.4, February 2001, http://www.ebxml.org/specs/ebTA.pdf.

Fearn 1999, Fearn, Battis, Burghi, Feliga, Generes, Hendry, Langballe, Designing Tivoli Solutions for End-to-End Systems and Service Management, IBM Redbook, July 1999.

HP Adaptive Enterprise, 2003, HP Management Solutions for the Adaptive Enterprise, Whitepaper, Hewlett-Packard, 2003, http://h71028.www7.hp.com/enterprise/cache/7504-0-0-0-121.aspx.

HP OpenView, Hewlett-Packard, http://www.openview.hp.com.

IBM Autonomic Computing, IBM, Architectural Blueprint of Autonomic Computing, April 2003, http://www-306.ibm.com/autonomic/blueprint.shtml.

IBM Tivoli, IBM, http://www.ibm.com/tivoli.

Ishii 1999, Ishii, Castoldi, Del-Cura, Perez, Tivoli Enterprise Performance Tuning Guide, IBM Redbook, November 1999.

Jacob 2002, Jacob, Perez, Darmawan, Manoel, Moeller, IBM Tivoli Monitoring for Business Integration, IBM Redbook, October 2002.

Manoel 2002, Manoel, Baker, Giannelli, Sadie, Weber, Introducing IBM Tivoli Service Level Advisor, IBM Redbook, July 2002.

OV IUM, HP OpenView Internet Usage Manager, Hewlett-Packard, http://www.openview.hp.com/products/ium.

OV NNM, HP OpenView Network Node Manager, Hewlett-Packard, http://www.openview.hp.com/nnm.

OV Service Navigator, HP OpenView Service Navigator, http://www.openview.hp.com/products/servnav.

OV SPI, HP OpenView Smart Plug-ins, http://www.openview.hp.com/products/spi.

OV TA, HP OpenView Transaction Analyzer, http://www.openview.hp.com/products/tran.

Potts 2002, Potts, Cox, Pope, Business Transaction Protocol Primer, OASIS, June 2002, http://www.oasis-open.org/committees/business-transactions/documents/primer.

SNA, System Network Architecture, IBM Systems Network Architecture: Technical Overview GC30-3073, IBM, http://www-306.ibm.com/software/network/commserver/library/publications/csaix_60/dysl1m02.htm#ToC_165.

Tretau 2003, Tretau, Andal, Battaglia, Edwards, Speh, IBM Tivoli Storage Management Concepts, IBM Redbook, August 2003.

UDC, Utility Data Center, Hewlett-Packard, http://www.hp.com/go/udc.

Grand Central Communications, http://grandcentral.com.

Flamenco Networks, http://www.flamenconetworks.com.

TransactPlus, http://www.transactplus.com.

Kenamea, http://www.kenamea.com.

Computer Associates: Unicenter – Managing On-Demand Computing: Aligning IT With the Business, Whitepaper Computer Associates, 2003, http://ca.com/unicenter.

Actional, http://www.actional.com.

Actional 2003, Effectively Managing an Enterprise Web Service Network, Whitepaper, Actional Inc., http://www.actional.com, 2003.

AmberPoint, http://www.amberpoint.com.

Confluent, Confluent Web Services Management Platform, Whitepaper, Confluent Software, 2003, http://confluentsoftware.com.

Microsoft, Microsoft Application Center, http://www.microsoft.com/applicationcenter.

Service Integrity, http://www.serviceintegrity.com.

FIGURES

INDEX