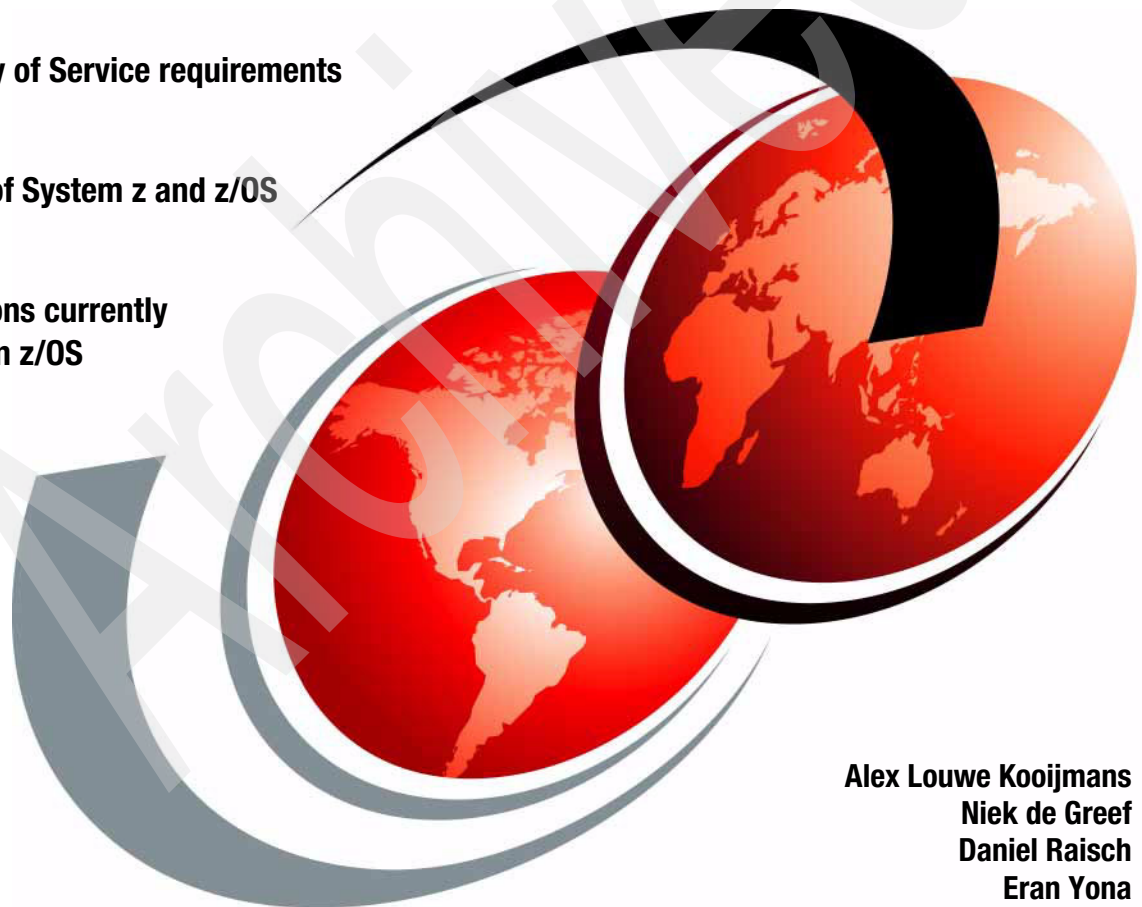IBM

The Value of IBM System z and z/OS in Service-Oriented Architecture

SOA Quality of Service requirements

Strengths of System z and z/OS

SOA solutions currently available on z/OS

Alex Louwe Kooijmans
Niek de Greef
Daniel Raisch
Eran Yona

Redpaper

International Technical Support Organization

The Value of IBM System z and z/OS in Service-Oriented Architecture

August 2006

First Edition (August 2006)

This edition applies to WebSphere Application Server for z/OS Version 6.02, WebSphere Message Broker V6.0 for z/OS, WebSphere Process V6 for z/OS, WebSphere ESB V6 for z/OS, CICS Transaction Server V3.1, CICS Transaction gateway V5 and V6, DB2 UDB for z/OS V8, IMS Transaction Manager, IMS Connect.

This document created or updated on September 5, 2006.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®	CICSPlex®	Rational®
@server®	DB2®	Redbooks™
Redbooks (logo) ™	Geographically Dispersed	RACF®
z/OS®	Parallel Sysplex™	RMF™
z/VM®	GDPS®	System z™
z/VSE™	HiperSockets™	System z9™
zSeries®	IBM®	Tivoli®
z9™	IMS™	WebSphere®
AIX®	MVS™	
CICS®	Parallel Sysplex®	

The following terms are trademarks of other companies:

Java, JSP, J2EE, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

Without doubt, Service-Oriented Architecture (SOA) is one of the most important topics on the agenda of any IT person. SOA involves a new vision of how to design, develop, and manage applications, but also puts new requirements on the underlying infrastructure.

This Redpaper describes the infrastructure challenges that SOA brings to the table and how the IBM® System z™ platform and the z/OS® operating system address those challenges. An effective SOA implementation requires very high Quality of Services (QoS) from the underlying environment, and users demand security, availability, and simplified management of the services. These are fundamental characteristics of System z and z/OS, making them an ideal platform on which to design an SOA.

This paper presents an overview of SOA, describes the SOA reference architecture, and demonstrates how IBM System z and z/OS support the SOA requirements. Finally, it suggests an approach for SOA-enabling existing applications and provides several integration scenarios.

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Alex Louwe Kooijmans is a project leader at the International Technical Support Organization, Poughkeepsie Center. He leads residencies and instructs workshops in the area of Java™ and WebSphere® on z/OS. Alex has worked for IBM since 1986 and this is his second assignment to the ITSO. Before joining the ITSO, he worked in various other roles, such as IT Specialist supporting customers in Europe getting started with WebSphere on various platforms, and Client IT Architect in the financial services sector in The Netherlands. His areas of expertise include Java and WebSphere on z/OS, and integrating WebSphere with DB2®, MQ, CICS® and IMS™.

Niek de Greef is a senior IT Architect from the Netherlands working for Software Lab Services in Hursley, UK. He has 16 years of experience in IT, most of which in the mainframe area. He holds a MSc. degree in Computer Science from University Twente in Enschede, the Netherlands an MBA degree from Henley

Management College in Henley, UK. His areas of expertise include e-business infrastructures and software engineering.

Daniel Raisch is a Senior Certified IT Architect. He has 25 years of experience in IT, mostly related to the mainframe. He holds a degree in Mathematics and Computer Science from Universidade Federal do Rio de Janeiro, Brazil. Daniel works with customers extending core applications to new technologies and has written several redbooks. He can be reached by e-mail at raisch@br.ibm.com.

Eran Yona is an IT Architect from the Israeli Ministry of Defense. He has 14 years of experience in IT. Eran has a BA degree in business from the College of Management in Israel. His area of expertise includes datacenter management and datacenter infrastructure.

Thanks to the following people for their contributions to this project:

Paul DiMarzio
z/OS On Demand Business, Enterprise Integration Strategy

Nigel Williams
IBM Design Center for On Demand Business, Montpellier, France

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

      ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this Redpaper or other Redbooks™ in one of the following ways:

- ► Use the online **Contact us** review redbook form found at:

     ibm.com/redbooks

- ► Send your comments in an e-mail to:

     redbook@us.ibm.com

- ► Mail your comments to:

     IBM Corporation, International Technical Support Organization
     Dept. HYJ Mail Station P099
     2455 South Road
     Poughkeepsie, NY 12601-5400

1

What is SOA and why do we need it?

This chapter provides an overview of Service-Oriented Architecture and defines the business drivers that are contributing to making SOA an important factor in the current business climate.

1.1 Introduction

Service-oriented architecture (SOA) is not the first IT evolution that has come along. We have seen many others in the past, with the e-business evolution being one of the most radical and important ones. The Internet adoption and popularity seemingly happened overnight and suddenly the world was totally different. Due to this IT transformation, the social and business climate entered into a new cycle of wealth generation.

The Internet, e-business and e-commerce shifted the shopping experience from the shop counter to the client's Personal Computer and even mobile phone, and started to remove human intervention from the back-office processes. Everything had to happen faster and with more information. Order confirmations, balance statements, and even mortgage contracts have to be delivered to the client in real time, ideally without human intervention, and definitely without having to wait for lengthy and error-prone overnight batch processes.

Obviously, this transformation imposed tremendous requirements on the complex integration between applications and IT landscapes within an enterprise, and many companies found out that they were not flexible enough to undergo this rapid change in the IT landscape.

We have seen many solutions to address this problem of integration, commonly referred to as *business integration*. *Service-oriented architecture*, referred to here by the abbreviation *SOA*, is a new, better, and as we show later in this paper, more successful solution to the need for integrating a business.

1.1.1 Who should read this paper?

This paper is intended for anyone interested in understanding SOA at an infrastructure level, as well as individuals who need to assess what SOA means for an IBM mainframe environment, and vice versa, what an IBM mainframe can mean for SOA.

In particular, we go into considerable technical detail by providing a deployment model of SOA on the mainframe and show the value and differentiators it brings to the business and IT.

Previous technical knowledge of IBM zSeries®, IBM System z9™ or z/OS is not required to understand this paper. However, if you do have good knowledge in this area, you will derive significant additional value from reading and exploring this paper.

1.1.2 How to read this paper

Figure 1-1 is a guide to how this paper is organized.

The main objective of this paper is to show the value that the IBM mainframe provides to the SOA model from a business and IT perspective. We challenged ourselves to not make the common mistake we saw in most other documents, namely to show the value of the mainframe for any kind of solution. Our goal was to find strengths of the mainframe as they relate specifically to SOA.
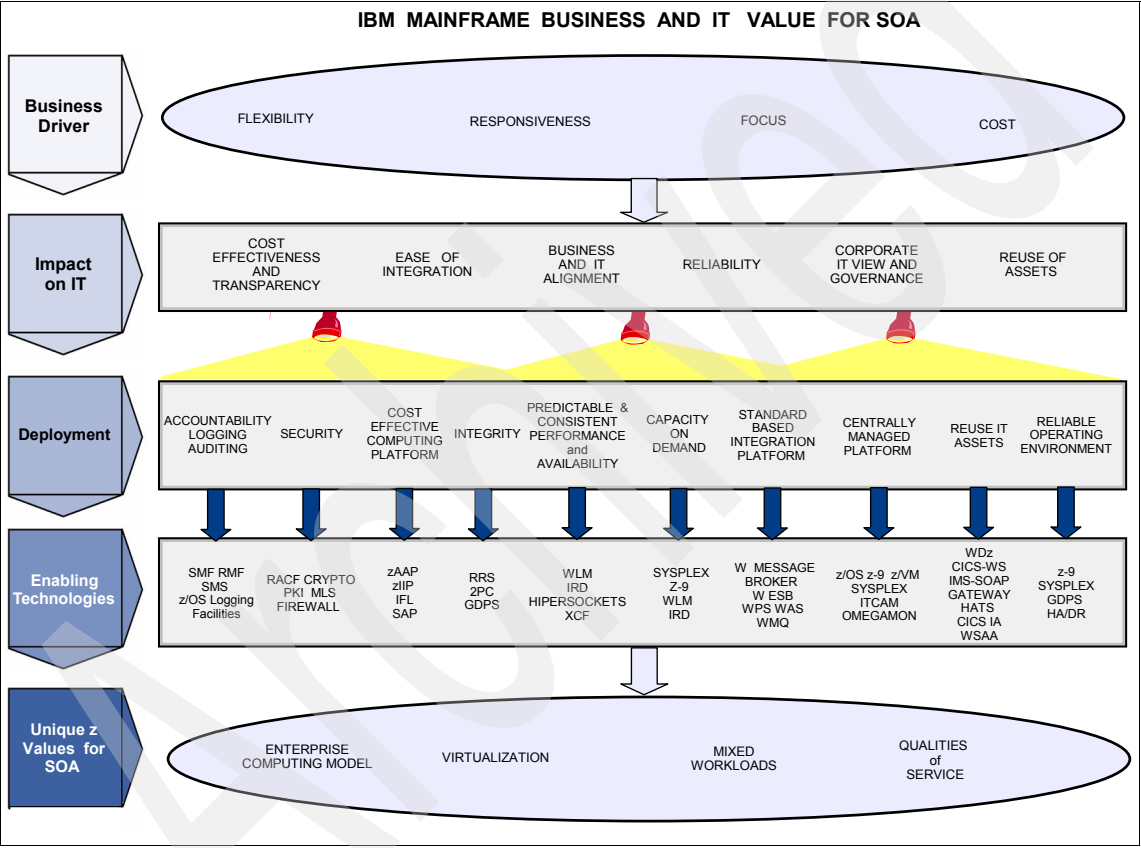
Figure 1-1   Mainframe value for SOA

Our approach to the challenge was to look at SOA-specific issues, beginning with what motivates a business to implement SOA, and developing the rationale up to achieving mainframe-unique values for SOA.

In order to prove our theory, we developed a simple test. We substituted the term *SOA* with any other major solution or model, and if the conclusion we got for SOA

also applied to the other solution, we dismissed it, because we were looking for something exclusive to SOA.

We truly believe we succeeded in our challenge. We expect you will agree after reading this paper. Enjoy!

1.2 Business drivers

Some of the most common questions a CEO is facing today are:

► How can my business be very *flexible* and *dynamic*?

► How can my business take different types of processes, combine them and make a new process, and by that be more *responsive* to the market's dynamics?

► How and where can I and should I cut costs and be more *cost effective*? Does my business need to *focus* on specific customers, products, market segments?

1.2.1 Key success factors and core competencies

Companies are focusing on their *core competencies*. Why?

In order to compete effectively and thrive, companies must focus on their key success factors, imposed upon them by their stakeholders. Businesses must be *innovative*, and improve their product range and customer services to be able to sustain their competitive advantage in today's global, continuously changing business environment.

To achieve these key success factors, organizations must develop and focus on their core competencies (see, for example, Prahalad and Hamel, 1990). This increases their competitive advantage by providing access to new markets and segments and providing products and services that make an important contribution to customer success and that are difficult for competitors to mimic.

As a consequence, companies at any moment need to be able to redesign their business processes to regain focus on core business activities and identify poorly performing processes and non-core processes. This business redesign can initiate business process optimizations with respect to effectiveness and efficiency, and business process outsourcing decisions (see Cherbakov, et. al., 2005). Component Business Modeling (CBM) is a method that business consultants use to create a representation of the business as an organized collection of business components. It provides a foundation for discovering both business inefficiencies and the transformations required to meet strategic goals.

Now we can extend this consideration to IT. In order to support the flexible business model required today, IT must be closely aligned to the business, and the supporting IT systems must possess the flexibility to be reorganized, enhanced, or reduced – with market fluctuations or when the business processes otherwise require this.

We see four major business drivers that influence a business today, as shown in the upper layer of our diagram in Figure 1-1 and shown separately in Figure 1-2 on page 5:

► Flexibility

► Responsiveness

► Cost effectiveness

► Focus

Figure 1-2   SOA business drivers

In the next sections we discuss those business drivers further.

Business flexibility

The market today is very different from the market we saw just a few years ago. Competition between companies has become very aggressive, markets have become global, and products must be improved constantly to keep up with customer expectations. In order to compete, businesses must be able to quickly offer products and services that respond to customer needs.

Companies need to be very flexible with their products and their lines of business, which can be in a constant state of flux. Not only do market changes put strains upon businesses, but mergers and acquisitions are also part of the business landscape today. A company must be able to take a business line that worked smoothly, change part of it, integrate newly acquired business processes, and combine these together into a new business process. All of that needs to be done in a short period of time in order to answer market movements and needs.

In the past, businesses generally had very linear and relatively isolated processes that were designed by the individual lines of business in the organization. This has led to the typical static business silos within enterprises that lack integration. Today's businesses need more integrated business

processes, where processes for a single business unit are composed of several smaller processes, of which some may be unique to a business unit, some may be reused from other business units, and some may be shared with other business units. The organization needs to be able to analyze specific parts of business processes, and change or outsource them, without interfering with the overall processes.

Responsiveness

The ability to respond to the customers' needs must be a main goal in every company. Customers today know what they want, the way they want it, and how much they are willing to pay for it. In today's very open and global market it is easy for customers to switch suppliers when a product offering no longer satisfies them.

On the other hand, these easy shifts of customer demands require companies to be able to respond to increases (and, as a matter of fact, deceases) in demand very quickly. A business has to have the ability to serve more customer during a peak period of time (like a holiday season) without ending up with overstocked reserve resources during non-peak periods.

Responsiveness can be seen also from another point of view: the way a business is ready to respond to the changing business requirements that come from other sources – like legal changes, mergers and acquisitions, new technologies, and so on. You would like to respond to those needs without re-inventing all your business processes.

Cost effectiveness

Of course, the price of the products a company puts on the market must be in line with the company's competitive strategy. If the company is focused on cost leadership, this is an obvious requirement. But for companies with differentiation and focus strategies, there is also nowadays a constant pressure to achieve cost-effectiveness.

Business processes can be very complex. Businesses today are analyzing and redesigning their processes, with the goal to identify and strengthen those processes that provide the company a competitive advantage, optimize process usage in terms of costs and reuse, and finally, to identify processes that can better be outsourced to partners who are better positioned to execute the process for the company. This movement not only can make the business more cost effective, but also more flexible and responsive.

Focus

The complexity of the business process makes it essential to a company's executives to be able to look at and manage the entire process in the company.

They should be able to focus on processes and even on parts of a process to learn about it, check its efficiency, and update it if needed.

On the other hand, a business should have the ability to focus on a specific market segment or type of transaction and give it special care, especially when demand and needs change.

1.3 Impact on IT

In the previous sections we talked about the business drivers; in the following sections we discuss how those drivers impact the IT systems.

Figure 1-3 illustrates what CIOs are facing today. This picture shows the actual application architecture of a retailer in the United States. By the way, in the upper right-hand corner, it says "Page 1 of 2," and in the gray box, it identifies this as version 4. We do not expect you to be able to read this diagram, and it is not our intention to show the exact infrastructure in this figure, but only give a hint about how complex an infrastructure can get in the real world.
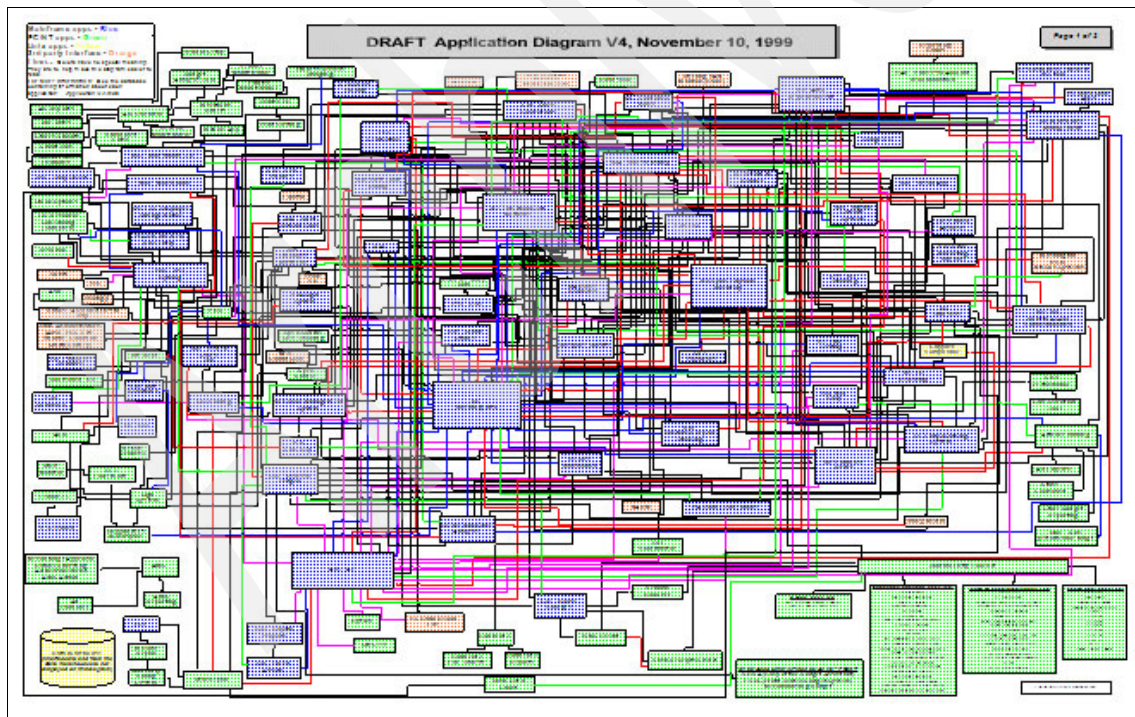
Figure 1-3   The need for infrastructure simplification

Each one of the rectangles in this diagram has a different server underneath it. Every time this retailer puts up a new application, they put up a new server underneath it, which means their servers are very under-utilized – roughly 15 to 35 percent.

In the following sections we discuss the major consequences of today's business requirements on IT and zoom in on the second layer of our diagram, shown in Figure 1-4.
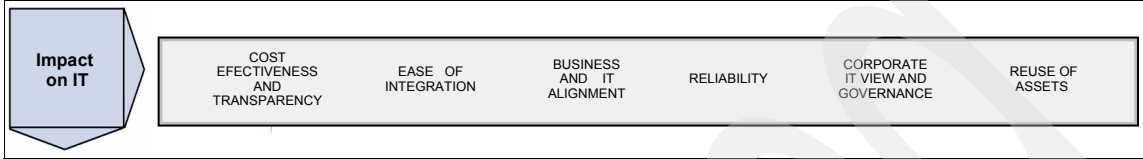
Figure 1-4   SOA impact on IT

1.3.1 Business and IT alignment

Companies today are pushed to increase productivity and cut costs. This task is not an easy one to achieve when the IT systems in the business are expensive, rigid, and unwilling to change. The flexibility to meet new market demands and to seize opportunities before they are lost or taken away by competitors is a must for today's businesses.

To achieve this, companies must examine their business processes and decide which are core functions that will help the business to be successful and to differentiate itself from the competition. The non-core functions can be streamlined or even outsourced to a business partner. By using a value chain that is composed of activities executed by the company itself, and of activities provided by different business partners, each of them focusing on their own core competencies, a business can deliver to the customer a service or a product in a shorter time and with less expense.

To support this strategy, the IT environment has to be very flexible and able to support changes on demand as they are made by the business. The IT systems must be able to communicate and integrate in a flexible way with other IT systems within the company, as well as with systems from partners, suppliers, and consumers in the external environment.

The *business interactions* are an exchange of information between business processes. These processes can be internal processes or external processes. In order to support flexibility in the interaction between business processes, the IT systems that support these processes must be easily adaptable. Because a company may choose to change business interactions, the supporting IT systems must support such changes without the need for extensive application

redesign. By applying techniques like information hiding, encapsulation, abstraction, and applying flexible interface technologies, IT systems can provide this flexibility and adaptability. New technologies allow composable business processes and composable IT system implementations.

In summary, a better alignment of IT with business processes, along with the ability of IT to be more flexible, can do the following:

► Allow IT to participate more easily in the business process outsourcing strategy.

► Reduce complexity of the IT infrastructure.

► Justify IT investments more clearly.

► Give the business a better understanding of what IT does and what its value is.

1.3.2 Cost efficiency and transparency

As described in the previous section, the relationship between the business and IT has changed. By understanding the business processes better in terms of business context, functionality, and performance, IT can provide the means to assess the value of its IT assets and measure productivity. *Componentization* of the business and corresponding IT objects allows a much more precise design, which makes for better solutions, ones that can be developed quickly and cost less in terms of development, operation, and maintenance.

1.3.3 Ease of integration

Realizing a close but loosely coupled relationship between business objects and processes on the one hand and supporting IT systems on the other hand, makes it easier to build, deploy, integrate, and link applications and heterogeneous systems and platforms together across the organization.

The use of well-defined, platform- and vendor-independent standards makes it possible to let components cooperate easily and flexibly. This, in turn, allows for the reorganization of existing business processes and the implementation of new ones in a short period of time. The components in the changed or newly created processes can be composed of multiple technologies provided by multiple vendors, as long as they comply with the industry standards.

1.3.4 Reliability

Re-using objects from variable requesters and having the ability to compose those objects into business processes is driving the IT environment to be very reliable. The requesters should not have to care about the technology behind the

object implementation. Requesters must be able to depend on the agreed service level of the IT components it uses.

Reliability is a quality that must be provided not only in the functionality of the IT architecture; it is even more so an important characteristic of the operating environment on which the architecture is deployed. Reliability refers to:

► Operating platform reliability and stability
The IT systems must be available when the business needs them.

► Predictable, consistent performance
The supporting IT systems must remain responsive even when strong fluctuations in demand occur.

► Integrity
When integrating different systems based on different technologies, and possibly controlled by partners outside the control of the company, integrity becomes very important. Maintaining the integrity of information exchanged between partners and systems is necessary to ensure that information has not been tampered with during the exchange. In addition, the integrity of the business transactions that span all of these systems must be controlled, and this requires support for atomic transaction capabilities.

► Security
The IT systems and infrastructure must provide a level of security in accordance with the business process requirements.

1.3.5 Corporate IT view and governance

Businesses and supporting IT systems must be able to react to changes in the business environment very quickly and take advantage of new business opportunities. As a consequence, it is no longer sufficient to view IT on the level of a line of business or a department. The IT supporting the business processes must be designed and managed as a highly integrated enterprise architecture. The consequence of not having such an integrated view of business and IT will be duplication of business and IT functions, which obviously cost more and are not as effective, and will also lead to future integration problems. When business functions and supporting IT have been separately designed, and at a later stage need to be integrated, this integration will be costly and time consuming.

As described before, in order to increase revenues, the business needs to be more flexible, and adopt new ways to connect between business needs and IT applications. *Governance* of business assets and, as a consequence, of IT assets, plays a more prominent role than ever before.

By governance we mean the overarching view to prioritize and support the enterprise business objectives on a strategic, functional, and operational level.

The *governance model* addresses questions like "What has to be done?", "How is it done?", "Who has the authority to do that?" and "How is it measured?" (Bieberstein et al, 2006).

Even in a well organized and defined organization, unplanned business exceptions can occur. Monitoring the organization as a whole, and having the ability to focus on specific processes and IT systems, and even change them, is very important to business success. The IT systems have to provide those capabilities.

1.3.6 Reuse existing IT assets

Each business has its IT assets. Reuse of those assets, in new and existing processes, instead of inventing, designing, and implementing new ones, makes IT implementations effective and drives down costs. For this to be possible, the components making up the IT ecosystem must be able to communicate in a standardized manner so that they can work the same way everywhere without the need for additional programming.

To be able to rearrange and compose IT components effectively, not only are technical measures required, but also organizational effort is needed to really effectuate the promises of SOA. On the organizational level, strong IT governance and an enterprise-wide vision of IT are needed.

1.4 What does SOA bring to the table?

In the previous sections we have discussed the impact of today's business requirements on IT. Before going into detail on how an SOA addresses those needs, we will first highlight what an SOA actually is.

1.4.1 Defining SOA

Service-oriented architecture is a system architecture in which application components are built as *services*. These services have well-defined interfaces and are loosely coupled, which provides flexibility in interoperability and reuse. Service definitions can be based on a direct mapping of business processes to IT systems. In that way, an SOA can offer closer business and IT alignment than earlier integration architectures.

From a business perspective, an SOA is a set of flexible IT components that can be used to support composable business processes. From a technical perspective, an SOA is a set of IT services that can be called to perform a specified operation.

In other words, SOA and component business modeling (CBM) are two faces of the same coin. CBM presents the business as an organized set of business components. The component that consumes a business service offered by another business component is oblivious to how the provider creates the business service. Service interactions between business components are governed by business-level agreements and contracts, which cover items such as cost structures, service levels, and so on. The information technologies that evolved in support of service orientation is called *Service-Oriented Architecture* (SOA).

So, key to an SOA is the concept of a service, but what is this?

A *service* can be defined as a reusable function that can be invoked by another component through a well-defined interface. The main role of services is to expose important business services in a flexible, easily composed and highly reusable fashion. Services are loosely coupled, which means that they hide their implementation details and only expose their interfaces. In this manner service requesters need not be aware of any underlying technology or programming language the service is using.

The most important characteristics of a service are:

Loose coupling   Services hide implementation details and minimize dependencies with the requesting party.

Reusability   Services must be reusable by definition. Services can be reusable over several lines of business.

Composition   Services can be assembled and coordinated to form new services.

Aggregation   Services can be aggregated to form new functions. Aggregated functions are more specific and therefore less reusable than the services they are composed from.

Genericity   Services can be specific to a particular process or channel, or they can be generic in nature. Generic services are more reusable.

Granularity   Granularity is defined by the amount of data a service processes or by the amount of processing in term of functionality the service implements. Granularity is one of the major design issues in the service design process.

1.4.2 SOA life cycle

Businesses should consider SOA in terms of a life cycle. The cycle begins with gathering requirements and designing business processes. After designing the

processes you should assemble new and existing services from the business process. Then, you deploy the business process into a secure and available environment. In the end you can monitor and manage the process both from a business and IT perspective. The steps are as follows:

Model   Start the model phase by gathering and analyzing the business requirements, which are used to model, simulate, and optimize the business process. In this phase it is important to establish a common understanding between IT and business in order to be sure that the final application is going to meet the defined business requirements.

Assemble   Once you have optimized your business processes, look for existing services in your systems to participate in the business process. (If there is none available, you should create them). Once they are available you can *choreograph* them together into a business process.

Deploy   In this phase you prepare the run time environment for your business process. You need to take into consideration that all key elements of your company are connected and working together. In this phase you should check that your run time environment is well secured, highly available, and scalable.

Manage   After deploying your business process into a secured and scalable environment, start monitoring and managing the environment. This process should be done from both a business and IT point of view. Understanding the outcome of this phase can enable better business decisions and enable continuous improvement.

Process and governance

On the base of these lifecycle stages, you can establish the governance policies which provide guidance and oversight for the SOA project.

1.5 How SOA addresses business needs

Misalignment of IT and business is a pain. When business analysts attempt to optimize, integrate, or outsource business processes, but IT is inflexible, bound to proprietary solutions, and therefore difficult to integrate, this can become an inhibitor for business optimization initiatives.

Implementing an SOA environment can help both IT and the business to be prepared for rapid changes in the business climate. By adopting the SOA approach the business processes become more flexible and responsive to

changes. Reuse of IT assets makes business processes more cost effective. An SOA provides the means to monitor and measure efficiency of business processes and the IT services that support the processes.

Companies that adopt SOA can benefit significantly by:

► Driving down costs by using the business and IT assets more effectively.

► Making the business be ready for changes and by that be more reactive to threats and opportunities.

► Providing the means for measuring business process efficiency, thereby supporting a company in determining its core competencies and developing business process outsourcing strategies.

► Improving understanding about IT functions at a business level, in both functional and operational terms. This allows better judgements and decisions on IT investments.

In the following chapter we present the IBM SOA reference architecture. After we explain the building blocks of an SOA, we go into further detail on how an SOA addresses the business needs and IT challenges coming from those needs.

2

SOA reference architecture

In the previous chapter we described what today's business needs are and how an SOA addresses those needs. We have also seen these business needs impose requirements on an SOA architecture that are not solely defined in functional terms and implemented through state-of-the-art applications, but that will have to be provided by the IT infrastructure.

In this chapter we examine the components that make up an SOA and how they can be deployed so that the non-functional requirements implied by the business needs on the architecture are also addressed.

We begin by illustrating how the view of application integration has developed over time. We discuss IBM's SOA reference architecture, which we then refer to throughout the remainder of this paper. We conclude this chapter with a summary of required deployment characteristics, which are the starting point for explaining the strengths and value of the z/OS platform.

2.1 Evolution of application integration

There are many views of how programming paradigms have evolved towards SOA as we know it nowadays; the view depicted in Figure 2-1 is just one. (See Erl, 2005, for more details.)

SOA has come from two somewhat independent but related developments:

► Evolution of the programming paradigm towards increased reusability.

► Evolution of program interoperability through a shift towards increased integration of applications.

Figure 2-1 illustrates how these factors have evolved towards SOA.
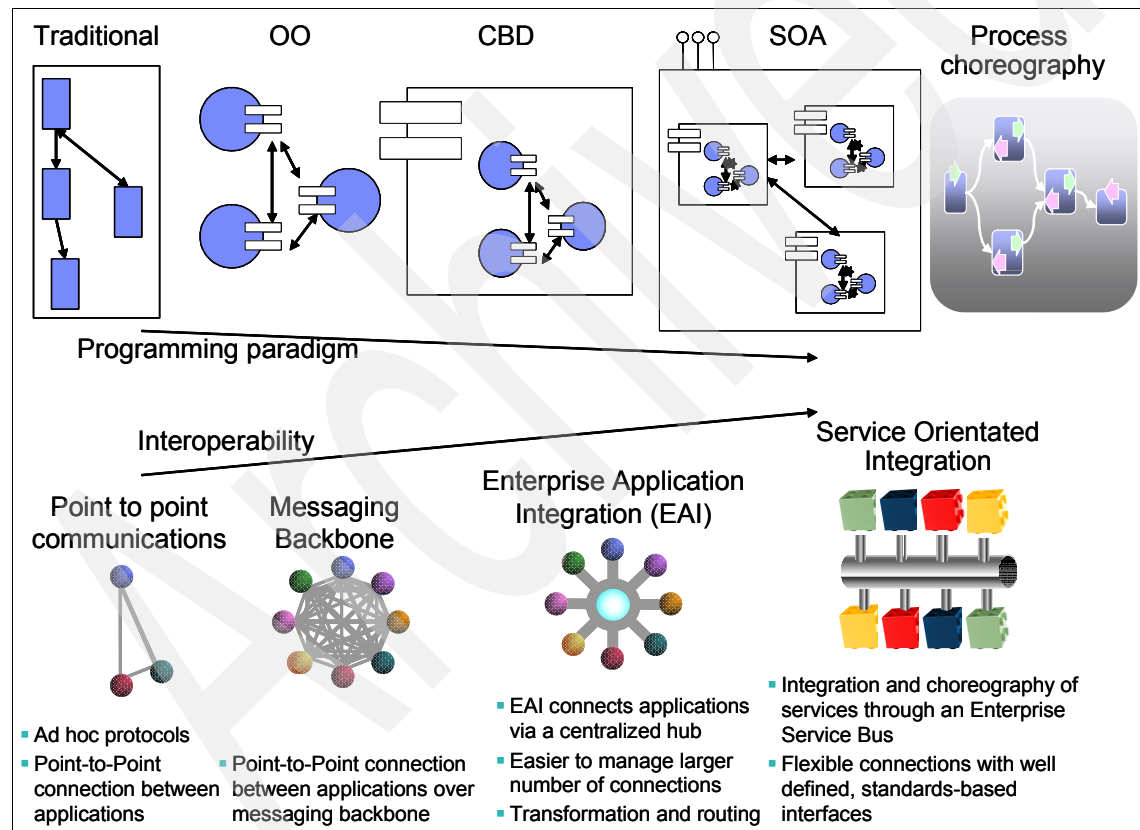
Figure 2-1   Evolution towards SOA

2.1.1 Programming paradigm shift

In the early days programmers were mainly concerned with structuring programs into logical procedures, which were tightly coupled and rarely reused. Communication and resource access was often more complex than the actual business logic implemented.

Modularization brought further program structuring and reuse into play, and the object-oriented programming paradigm provided new capabilities for reuse. Component-based development (CBD) has become the general term for design and programming techniques concerned with reuse and separation of concerns.

Recently IBM has announced *Service Data Objects (SDO)* and *Services Component Architecture (SCA)* as their next steps, and has thereby introduced an SOA programming model that simplifies program construction and enables composite application development.

2.1.2 Interoperability shift

At the same time, program interoperability became an issue as programs increasingly needed to communicate with one another. *Messaging middleware* relieved the programmer of a lot of communication worries and provided a further decoupling of application components. *Message brokering* products were the next step in application integration, providing tools for message transformation and aggregation facilities.

The SOA paradigm provides application designers with standardized intercommunication options and reuse of services at all levels, up to the business service level.

What this development means to application designers is that where they used to spend a lot of time on low-level aspects of system operation and interoperability, they now can focus primarily on the business problem space.

Business process choreography brings a further development into the integration picture, allowing a business modeler to assemble business processes from automated and human services.

2.2 Technologies underpinning an SOA

Before jumping into the SOA reference architecture, we highlight the technologies that can be used today to implement an SOA.

2.2.1 Web services and related technologies

Web services are a set of specifications that define a standard for system-to-system communication. Through Web services, applications can cooperate by invoking well-defined interfaces. The interfaces hide implementation details of the service, making it possible for applications based on different technologies, running on different platforms, to participate in higher level composite applications.

Web services standards use other technologies, such as XML and HTTP, and offer the de facto standards for the implementation of an SOA. The most important Web services standards are discussed now.

The *Web services Description Language (WSDL)* is a series of XML standards that define the interface of each service. The WSDL of a service is highly reusable and should be kept in a place easily accessible for application developers.

The messaging protocol used as part of the Web services standards is called the *Simple Object Access Protocol (SOAP)*. It is a standard for defining the format of a request/reply that is sent to and returned from a service. Note that SOAP does not provide the transport infrastructure. You still need to use a transport protocol underneath. Currently, the SOAP protocol can be used over HTTP or JMS.

There are some protocols that deal with security for Web services. The Web services security specification, commonly named *WS-Security*, is based on a token architecture for secure communications. Built on this base we can find specifications like *WS-Policy*, which defines the rules on how services interact, and *WS-Trust,* defining the trust model of a secure exchange. There are others as well.

The Web services specification also addresses quality of services (QoS) issues. For transactionality there is *WS-Coordination*, *WS-BusinessActivity*, and *WS-AtomicTransaction*. Reliable messaging protocols have been defined in *WS-ReliableMessaging*.

2.2.2 Enterprise Service Bus (ESB)

The communication between Web services in an SOA environment is built on the *Enterprise Service Bus (ESB)*, which is a common distributing network for services to work with. The ESB, as the heart of the SOA environment, has to be very reliable, available, and secure.

An ESB enables applications to create service interfaces for new or existing application functions, either directly, or indirectly through so-called *adapters*.

Besides that, the ESB can transport and route messages between service requesters and providers.

The ESB typically provides the following:

► Support for transport of messages between services over several protocols such as JMS messaging and HTTP.

► Transformation and routing of service requests.

► Event handling.

► (Web) services standards support.

► Support for new applications that are based on services and through that for technologies such as J2EE™, .Net, and so on.

► Support for all required existing applications, programming models, and data formats.

2.3 SOA reference architecture

The IBM SOA reference architecture presented in Figure 2-2 on page 20 includes the building blocks we described in the previous section. In addition to the primary functional building blocks, the architecture includes:

► Supporting building blocks that allow integration with existing and partner systems

► Building blocks providing additional management and monitoring functions

► Infrastructure services to address other non-functional requirements
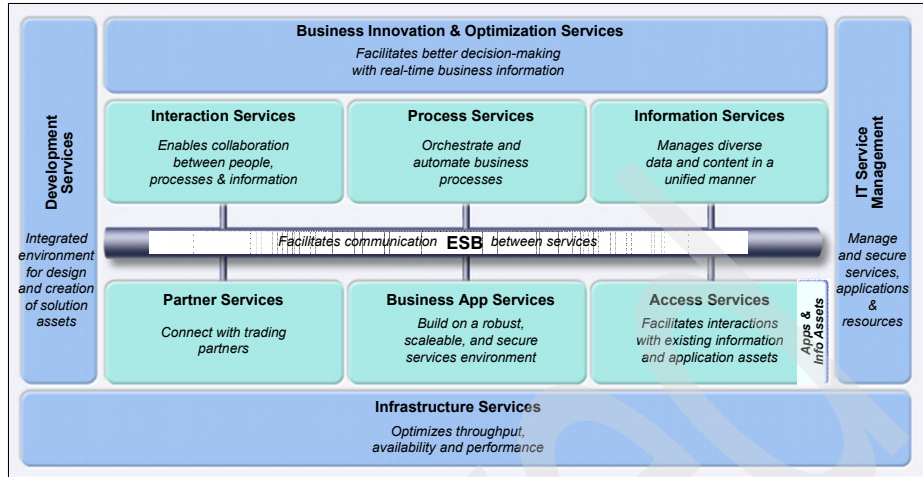
Figure 2-2   IBM SOA reference architecture

The IBM SOA reference architecture depicted in Figure 2-2 describes the key
capabilities that are required for comprehensive, enterprise-wide SOA solutions:

► Development services

► Business innovation and optimization services

► Enterprise Service Bus (ESB)

► Interaction services

► Process services

► Information services

► Access services

► Partner services

► Business application services

► Infrastructure services

► IT Service management services

These building blocks are described in more detail in the following sections.

2.3.1 Development services

Development services are an essential component of any comprehensive
integration architecture. These services enable different people, involved in the
development of a custom application, to develop the custom artifacts based on
their skills, their expertise, and their role within the enterprise.

- Business analysts can analyze business process requirements using modeling tools that allow business processes to be charted and simulated.

- Software architects can model the system structure and behavior.

- Integration specialists can design specific inter-connections in the integration solution.

- Programmers can develop new business logic with little concern for the underlying platform.

The development services enable joint development, asset management, and collaboration among all these people. A common repository and functions common across all the developer perspectives (for example, version control functions, project management functions, and so forth) are provided through a unified development platform.

2.3.2 Business innovation and optimization services

The *business innovation and optimization services* provide monitoring capabilities that aggregate the operational and process matrix in order to efficiently manage systems and processes. These processes are operational IT processes as well as business processes.

These capabilities are delivered through a set of comprehensive services that collect and present both IT and process-level data, allowing business dashboards, administrative dashboards, and other IT-level displays to be used to manage system resources and business processes.

There is a linkage between the development services and the business innovation and optimization services in the ability to deliver run-time data and statistics into the development environment, which allows analysis and iterative process re-engineering through a continuous business process improvement cycle.

2.3.3 Enterprise service bus (ESB)

At the core of the SOA reference architecture is the *enterprise service bus (ESB)*. This architectural construct delivers all the inter-connectivity capabilities required to use services implemented across the entire architecture. The ESB provides the following fundamental services:

- *Transport services* provide the fundamental connection layer.

- *Event services* allow the system to respond to specific events that are part of a business process.

► *Mediation services,* like transformation and validation services, allow loose coupling between interacting services in the system.

The true value of the enterprise service bus concept, however, is to enable the infrastructure for SOA in a way that reflects the needs of today's enterprise: to provide suitable service levels and manageability, and to operate and integrate in a heterogeneous environment.
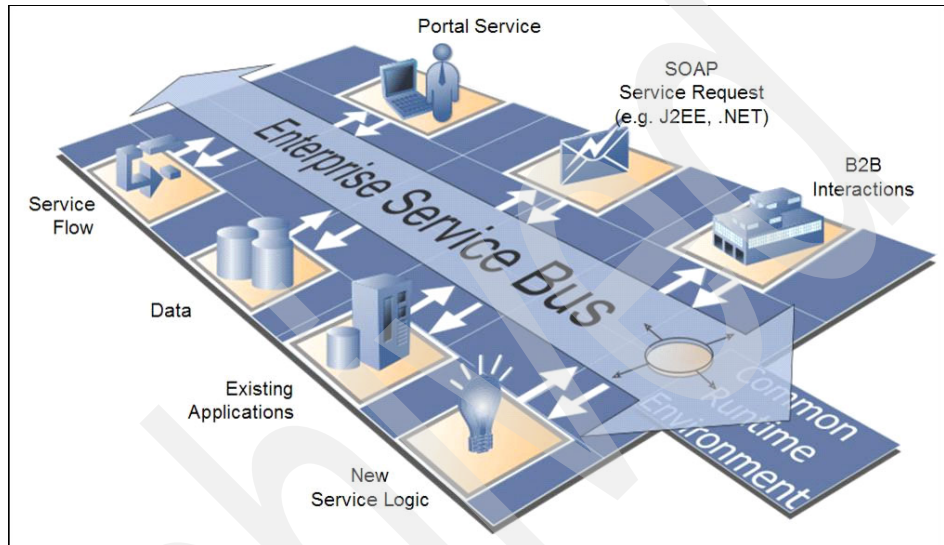
Figure 2-3   Enterprise service bus

The ESB should enable the substitution of one service implementation by another with no effect on the clients of that service. This requires both the service interfaces that are specified by SOA and that the ESB allows client code to invoke services in a manner that is independent of the service location and communication protocol that is involved.

Figure 2-3 shows a high-level view of the enterprise service bus.

The ESB is not the only infrastructure component in an SOA. Although individual scenarios vary, there are other commonly occurring components whose role we should position relative to the ESB:

► The *business service directory*, which provides a taxonomy and details of available services to systems that participate in an SOA. In order to perform routing of service interactions, the ESB obviously requires at least basic routing information, which might be provided by an ESB namespace directory, or by more simple means such as a routing table. However, this routing information is not necessarily the same as the business service

directory SOA component; the role of the business service directory is to provide details of services that are available to perform business functions that are identified within a taxonomy. The business service directory might be an open-standard UDDI directory, or more basic forms might be implemented as a design-time service catalogue, perhaps using collaboration technology. Such catalogues can achieve one of the primary goals of a business service directory: to publish the availability of services and encourage their reuse across the development activity of an enterprise.

► *Business service choreography*, which is used to orchestrate sequences of service interactions into short- or long-lived business processes.

► The *ESB gateway,* which is used to provide a controlled point of external access to services where the ESB does not provide this natively. Larger organizations are likely to keep the ESB gateway as a separate component. An ESB gateway can also be used to federate ESBs within an enterprise.

For more information on the ESB, see the redbook *Patterns: Implementing an SOA Using an Enterprise Service Bus*, SG24-6346.

2.3.4 Interaction services

Interaction services provide the capabilities required to deliver IT functions and data to end users, meeting the end-user's specific usage preferences.

2.3.5 Process services

Process services provide the control and management services that allow integration and automation of business processes that span people, workflows, applications, different systems, and platforms; and meld them into a single, orchestrated application.

2.3.6 Information services

Information services provide the capabilities required to federate, replicate, integrate, analyze, and transform data sources that may be implemented in a variety of ways. These services provide access to the data repositories through various techniques such as accessing stored procedures as Web services, providing standardized interfaces to non-relational data repositories, and other access mechanisms that return "information as a service."

2.3.7 Access services

Access services provide bridging capabilities between legacy applications, pre-packaged applications, and enterprise data stores. These access services

make available the functions and data of the existing enterprise applications, thereby allowing them to be reused and incorporated into functional flows that represent business processes.

2.3.8 Business application services

Business application services provide run-time services required for new application components to be included in the integrated system. These application components provide new business logic required to adapt existing business processes to meet new demands of the enterprise.

2.3.9 Partner services

In many enterprises, business processes involve interactions with outside partners and suppliers. Integrating the systems of the partners and suppliers with those of the enterprise improves efficiency of the overall value chain. *Partner services* provide the document, protocol, and partner management services required for business-to-business processes and interactions.

2.3.10 IT Services management services

IT Services management services provide security, directory, IT system management, service-level automation and orchestration, and virtualization functions. The infrastructure required to support an SOA and the applications cooperating in an SOA is typically dispersed over several platforms and technologies, and is therefore significantly more complex than traditional self-contained applications.

2.3.11 Infrastructure services

Infrastructure services provide functions for scalability and performance, security, and resource virtualization capabilities. These include the hardware and software for deployment of the actual business services and service infrastructure. A production environment usually requires the highest degree of quality of service, and the infrastructure services component provides the QoS.

Many of the infrastructure and IT services management services perform functions tied directly to hardware or system implementations; others provide functions that interact directly with integration services provided in other elements of the architecture through the ESB. These interactions typically involve services related to security, directory, and IT operational systems management.

2.4 Deployment of SOA components

The previous sections have provided an extensive overview of the required and optional building blocks of an SOA.

This section explores the deployment characteristics of an SOA. Deployment in this sense is concerned with determining on which physical nodes a component can best be implemented, and what considerations are important when making those design decisions.

2.4.1 How the SOA reference architecture addresses IT requirements

The translation of building blocks into technical components and real hardware and software is primarily guided by functional and non-functional requirements. The SOA reference architecture addresses these requirements through the definition of building blocks that cover functional and non-functional requirements.

These requirements were summarized in 1.4, "What does SOA bring to the table?" on page 11.

The SOA reference architecture addresses all of these requirements.

Business and IT alignment

Process services provide control over the flow of business processes composed of IT services and human interactions. Process services provide as close a translation of business processes to IT processes as possible, including integration with human processes, and also provide the means to monitor and adjust these processes. Partner services allow integration of business processes with partner processes.

The infrastructure supports this by providing appropriate IT service management services that allow monitoring of the IT services making up these business processes at the right level of granularity. The infrastructure must also provide for scalability, including dynamic resource allocation to processes that need more resources for business reasons.

By providing a close business process to IT service translation, and in addition by providing the means to integrally measure and manage these services and processes both at the business and IT level, an SOA allows for tighter business and IT alignment than we have ever seen before.

Ease of integration

Many existing integration projects are still based on proprietary integration technologies. In order to provide for vendor independence and smooth integration with new or existing platforms, new IT systems and communication protocols must be based on open standards.

Ease of integration is supplied by a standards-based integration platform. This integration platform allows integration of the existing set of diverse IT assets in the organization with newly built applications and business processes. The standards it is based on, like Web services, allow smooth integration with partners and future systems.

The ESB is the backbone for allowing cooperation among IT services through standards. The other building blocks provide access to that services backbone using the same standards, or can rely on conversion services provided by the access services building block, or rely on transformation and conversion services provided by the ESB.

Corporate IT view and governance

The current call for a corporate view of IT and governance arises from the need to align the IT and business value chains. Governance provides the structure to prioritize business objectives and set out the IT strategy.

The need for IT governance is not new in an SOA, but it has become increasingly important since the need for integration of applications threatened to turn the IT landscape into spaghetti (see Figure 1-3 on page 7). However, an SOA can only provide its unique values if sufficient attention is paid to governance; otherwise, SOA will only be a new way to integrate applications. See, for example, Cherbakov et al (2005), and Bieberstein et al (2005).

An SOA provides a number of facilities that support a business-level view of automated business processes. The concept of an SOA and the business process management focus it provides in the process services building block allows business analysts to model business processes and decompose them to automated and manual components. Business processes can be designed and implemented starting from the business perspective and driven by the business. Process services and business process modeling services allow for reuse of existing IT assets in new business processes. Armed with these services, the business manager has the ability to make decisions on a corporate level with regard to reuse of services, and as a consequence, reuse of IT systems supporting those services. Business process monitoring functions allow monitoring of the business processes designed as well as any consequent changes of the processes due to new or changed business requirements.

The IT service management services in the infrastructure provide support for this enterprise model in the form of management and monitoring services of the IT systems in relation to the business processes they support.

IT asset reuse

Existing IT assets, whether existing programs or data, can be reused and therefore be made accessible from standards-based technologies to the composable business process implementation provided by process services. Reuse of the investments made in existing IT assets is therefore provided by access services.

These services must be made as reliable and efficient as possible through platform optimization on the platform where the assets are hosted because reuse of these assets will in practice lead to intensified use of them, thereby increasing the requirements with regards to quality of service.

2.4.2 QoS and other infrastructure requirements

In 1.3, "Impact on IT" on page 7 we discussed the impact of today's business drivers on IT. In the previous section we highlighted how the SOA reference architecture addresses the functional requirements. A number of non-functional requirements are imposed on the architecture that are less explicitly addressed.

Cost effectiveness and transparency can only be delivered if the infrastructure and service management environment provide the means to measure cost of IT on a sufficiently granular level in all different components making up the infrastructure. This imposes special requirements on infrastructure services and IT service management services. These building blocks must provide the means to measure the demand of individual services and components and assemble these figures into business-level reports.

The business process demands a certain *reliability.* The IT systems supporting the business processes must be able to provide sufficient reliability to support the business.

Reliability is a quality that is provided by services in the infrastructure in the form of:

► Predictable and consistent *performance* in accordance with business process requirements. This is especially important in an SOA because the services provided in an SOA can increasingly be accessed by different consumers, and have different characteristics with regard to fluctuations in demand.

► *Capacity* is related to performance consistency.
The IT infrastructure must be scalable when demand requires this, in stressing conditions such as peak transaction volumes, but also in support of,

for example, marketing campaigns that may attract a large number of potential customers and may dramatically increase capacity in both front-end systems and back-end systems throughout the SOA.

► Operating environment *reliability and stability*.
Here we mean technical reliability in the form of high Mean-Time-Between-Failure, and the ability to recover from failures without interrupting the IT operation. In an SOA the availability of a service becomes dependent on all the components making up that service. This will increase the availability requirements for the operating environments on which these services reside.

► *Integrity*.
The loose coupling that characterizes an SOA may have the consequence of loosening control over the atomicity of transactions spanning several services making up a business process. Techniques must be provided to support atomic transaction capability.

► *Security*.
The security implications of an SOA put special requirements on the technology the SOA is deployed on. The IT systems and infrastructure must provide the level of security required by the business process it supports. This comprises all security services at levels in accordance with what the business process requires, such as authentication, authorization, non-repudiation, confidentiality and encryption, privacy, and so on.

2.5 Summary of deployment characteristics

In this chapter we have seen what an SOA deployment environment should provide and how these requirements are addressed by the SOA reference architecture and the operating environment underneath the SOA. The technologies supporting the SOA and the operating environment in which the SOA is deployed must provide:

► A cost-effective computing platform

► Accounting, logging, and auditing facilities

► Consistent and predictable performance

► On demand capacity

► A standards-based integration platform

► A centrally managed platform

► Reuse of IT assets

► A reliable operating environment

- ▶ Integrity
- ▶ Security

These attributes form the third layer in our diagram, as shown in Figure 2-4.
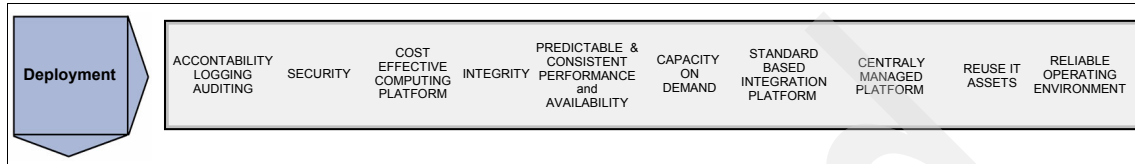
Figure 2-4   SOA deployment characteristics

The non-functional requirements are to be provided by the infrastructure underpinning the architecture. In the next chapter we show how deployment of SOA components on a z/OS platform can provide the required qualities.

3

SOA and z/OS

SOA is an architecture model, independent of platform, technology, and vendor, but when we come to the decision where to deploy SOA applications and the required infrastructure, we have to consider the different aspects and Quality of Services (QoS) provided by each platform and by each vendor.

In this chapter we discuss the particular characteristics and QoS provided by the IBM System z platform and the z/OS operating system when deploying the SOA model.

In the previous chapters we identified the requirements that an SOA must address. In this chapter we show how the required components are mapped to technologies, and how the building blocks can be deployed on z/OS, thereby addressing the non-functional requirements in a unique way.

3.1 How z/OS addresses SOA non-functional requirements

In this section we describe the traditional strengths of the mainframe, focusing on the qualities the mainframe provides, thereby addressing the most important non-functional requirements of most mission-critical IT systems.

3.1.1 Background

The IBM System z has for more than 40 years successfully run the core IT systems of many successful businesses, from medium size to very large. During these years, IBM has consistently invested in the evolution of the mainframe's unparalleled technology. Mainframes have incorporated all the new technologies and computing models that have come around in the marketplace since they were first delivered, from the Centralized model to the Web model, from the Assembler and Cobol languages to Java. Today the mainframe provides all the necessary capabilities to form the backbone in an enterprise Service-Oriented Architecture, and more than ever offers a unique proposition for the heart of an enterprise SOA deployment.

The IBM mainframe has become the computing industry benchmark. Mainframes are in use by thousand of enterprises worldwide, with trillions of dollars invested in applications and skills. The platform hosts a large portion of the total business transactions and data in the world, providing 24x7 availability and other unique qualities.

When we consider the new paradigms and value propositions of the SOA, and at the same time realize the value of existing IT assets and the qualities of the mainframe, the potential of the combination of the mainframe strengths and the SOA concepts becomes very clear. The combination can bring a fast return on investment when transitioning to an SOA, while building the SOA on a platform that provides the high QoS that an SOA requires.

Although the IBM mainframe runs five different operating systems, z/OS, z/VM®, z/VSE™, z/TPF and z/Linux®, in the following section we focus specifically on the z/OS operating system, the flagship mainframe operating system.

3.1.2 Traditional z/OS strengths

The generally recognized z/OS strengths fall into a number of broad categories:

► Centralized computing model
► Security

- ► Manageability
- ► Virtualization and workload management
- ► Reliability
- ► Scalability
- ► Availability
- ► Transaction processing
- ► Batch processing

The z/OS operating system provides a comprehensive set of capabilities and tools out-of-the-box that provide these qualities. The z/OS operating system is able to provide its unique quality of service in combination with System z hardware and Parallel Sysplex® capabilities. In the following sections we discuss each of those qualities.

Centralized computing model

We define a *computing model* as a structure that organizes the way communication and sharing occurs. It describes the topology of interconnected computer resources and how users access these resources.

In the history of commercial computing, we have seen three major computing models: centralized, client/server, and network computing.

The *centralized* model is characterized by a single or few, but large computing nodes. On these nodes a variety of applications are hosted, usually between tens and several hundred, and even thousands is not unusual. These applications all share the same computing resources such as memory, CPU, and disk and tape storage. All users are directly connected to this system.

In the *client/server* model computing resources are spread over a large number of distributed computing nodes that do not share the same resources. Each computing node typically runs a single or just a few applications and the applications communicate over the network. The objective of this model was to offload processing from the central computers which were seen as inflexible, old-fashioned, rigid, and expensive.

The *network* computing model is also known as the *Web* model. Users connect to applications using a Web browser, which is independent of a particular operating system, or hardware. All data, business logic, and applications are brought back to servers and only information required for display is sent to the user device. Small applications, or applets, can be downloaded from the server and run in the local memory of the terminal to improve functionality or performance. The Web model allows users to connect to applications from anywhere over the internet.
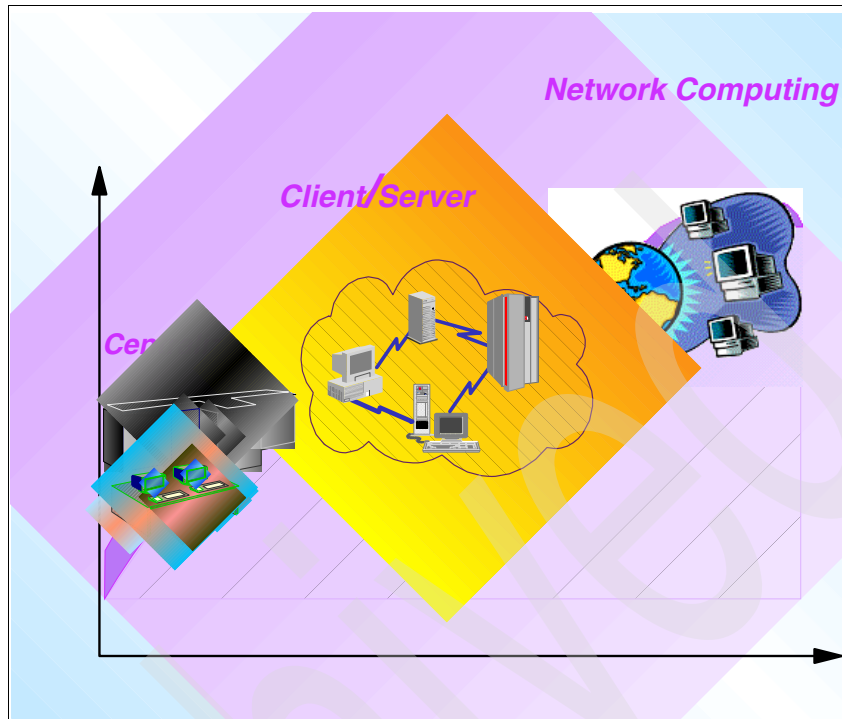
Figure 3-1   Computing models

The centralized computing model was abandoned in the late 80s and early 90s for some good and some not so good reasons. The primary complaints were lack of flexibility and openness, and the high cost of acquisition. At the end of the 90s the centralized computing model made a comeback after new developments in mainframe hardware and software had overcome the disadvantages. Businesses also began to realize the serious manageability complications and security exposures of the client/server model.

Mainframes were the typical representatives of the centralized computing model, which has some clear advantages. The implementation of this model in the mainframe hardware and the z/OS operating system differentiates itself by the provision of integrated *management and control* capabilities to manage the complex workloads running on the platform.

Another key differentiator of the centralized computing model is *data proximity*. When applications are in close proximity to the data they use, this seriously reduces communication overhead, increases security, and minimizes points of failure.

Resource sharing is an additional key differentiator between the models. In a centralized model every resource (storage, memory, processors, I/O channels, and so on) is shared between the different applications. This leads to an overall better utilization of those resources and less idle time. In a distributed model with dedicated resources efficiency is significantly less and idle time higher.

Partitioning and parallel sysplex

Centralizing all components on a single system does not imply that mainframe computing is limited to one single box or to one single operating system. On the contrary, mainframes have the capability of being partitioned into up to 60 logical partitions (LPARs) and each one has the ability to run one of the five z operating systems. There are management tools that allow resource sharing at the box level and at the operating system level.

The z/OS operating system provides the capability of creating a cluster of up to 32 systems, called *Parallel Sysplex*, where a "system" can be a full box or just an LPAR. Most computing platforms today have clustering capabilities, but Parallel Sysplex is a completely different kind of clustering solution because it is capable of sharing every resource between the elements in the cluster, and it is able to dynamically reconfigure, add, or remove resources.

z/OS allows all members in a cluster to share all data, even up to the record level. All other cluster implementations, at best, allow the partitioning of data among the elements of the cluster, and each system can access just the data attached to it.

Parallel Sysplex also provides significant network optimizations for communication across its cluster members. Once a client request reaches the Sysplex Distributor there is no more external network traffic required; all traffic flows over the System z hardware. As a consequence, network latency is kept to a minimum, and typical network issues you normally see in a physical de-centralized infrastructure are inherently absent. Even when the Parallel Sysplex is physically spread over several different boxes, the communication among them flows over high speed fiber optic connections managed by the *Cross Coupling Facility (XCF)*, a specific protocol for those connections, with a magnitude of GigaBytes of transfer rate. Within a physical machine, communication between z/OS images is accomplished memory-to-memory and there is no network protocol faster than that.

Security

z/OS provides deep security integration, from the application level down to the operating system level and the hardware. In this section we highlight a number of major functions in z/OS. For more information, you may want to read Guski, R. et al. (2001) *Security on z/OS: Comprehensive, current, and flexible*, IBM Systems Journal 40, No. 3, 696–720 (2001).

Centralized security

Approximately 30 years ago IBM developed the *Resource Access Control Facility (RACF®)* to provide centralized security functions such as user identification and authentication, resource access control, and auditing for both the operating system and applications running on the system. As a next step, to improve the usability of the system's security interfaces and thus encourage more applications to use consistent system-level security rather than implement their own separate security mechanisms, IBM implemented the *System Authorization Facility (SAF)* within the MVS™ operating system. SAF combined the various security function invocations into a single extensible security mechanism.

This new security structure provided several benefits to encourage use by application programs and components of the operating system itself. Most recently, IBM has again modified SAF to provide additional security interfaces for the UNIX® environment on z/OS.

With the introduction of the System Authorization Facility suite of services, a centralized point was created within the operating system infrastructure through which both operating system components and applications could invoke the services of the z/OS resident security manager.

Auditing and logging

A very important feature of a centralized authentication and access control mechanism is the ability to record and analyze security information. The audit data is essential for ensuring that the customer's installation security policy is being followed. The RACF option of the z/OS Security Server provides multiple ways to specify what security-relevant events are recorded in the audit stream and how that information is reduced and analyzed. RACF provides a wide variety of capabilities and tools to the auditor for data analysis.

Accountability

Accountability in z/OS is achieved by a combination of user authentication and the ability to propagate a user's credentials throughout the application.

The *System Management Facility (SMF)* interface is designed for collecting performance and accounting information. z/OS will collect all accounting and performance data and present this back to the installation through RMF™ and SMF records. These records can be analyzed and aggregated in performance and capacity reports, but can also be used as a base for security analysis and accounting policies.

Cryptography

The System z hardware has two distinctly different cryptographic hardware engines that are supported under z/OS: the CMOS (Complementary Metal Oxide Semiconductor) Cryptographic Coprocessor and the newer PCI (Peripheral Component Interface) Cryptographic Coprocessor (PCICC). The PCICC provides the capability for rapid response to customer requirements that was sometimes difficult to achieve with the CMOS Cryptographic Coprocessor alone.

Network security

Networking and communications security on z/OS is provided by the Communications Server element of z/OS. The Communications Server provides networking and communication services for accessing z/OS applications over both SNA and TCP/IP networks.

Firewall technologies within the Communications Server provide the network-level protections. The Communications Server provides a further, optional layer of protection by supporting multiple TCP/IP "stacks," which can allow customers to configure TCP/IP communications in a way that segregates the external (public) traffic from the internal traffic used in the private network. Thus, the z/OS firewall technologies can provide extra protection for applications using the stack that handles the external communications traffic. Furthermore, the firewall technologies include packet filtering to limit the kind of network requests that can reach a machine, proxy and SOCKS servers to control TCP/IP connectivity, Domain Name Services (DNS), and encryption of IP traffic (IP tunnels, or virtual private networks (VPN)) to allow private communication over the public network. Other security features provided by z/OS and the Communications Server include LDAP, PKI, Kerberos, and openSSH.

Manageability

The z/OS operating system has been designed to manage multiple workloads in a single system image. This nature has from the beginning put the requirement on the operating system to provide the means to manage these complex environments. As a consequence, over the years z/OS has become equipped with an extensive set of management tools for managing and controlling thousands of complex simultaneously running applications, comprising batch and online components, databases, transaction managers, and so on. The management tools included in z/OS have become a very mature set of system management and automation software. Customers over the years have based their sophisticated procedures and automation scenarios on this software.

These tools include:

► SAF (discussed in "Security" on page 35) is the security interface built into the operating system, and RACF is provided for security administration and management.

- SMF and RMF are used for resource management, performance management, capacity planning capabilities, and logging of system and middleware events.

- SMP/E is used for product installation and maintenance.

- DFSMS provides data management software that automatically manages data from creation to expiration. DFSMS provides allocation control for availability and performance, backup/recovery and disaster recovery services, space management, tape management, and reporting and simulation for performance and configuration tuning.

- SDSF provides the ability to monitor, manage, and control the z/OS sysplex.

- System Automation provides the capabilities to increase application availability through policy-based self-healing, and to automate z/OS Input/Output, processor, and systems operations.

- Tivoli® Workload Scheduler (TWS) provides functions to automate, plan, and control the processing of an enterprise's entire production workload.

Virtualization and workload management

Why is System z such a trusted platform for core enterprise systems and applications? One of the answers is *virtualization*. Unlike most other computer systems, the IBM System z is designed from the silicon chip on up for virtualization. By comparison, RISC and Intel® chips are architected to meet computational benchmarks. What they gain in the processing of instructions, they loose in task switching, error handling, data movement, and input-output operations.

IBM mainframes create a virtual environment, not only for existing and new CICS, IMS, DB2, and batch workloads, but also for workloads such as Java (J2EE) applications, Linux, UNIX and GRID applications. This means that an enterprise can run its entire enterprise server workload within a virtual Mainframe environment.

Unlike Intel or UNIX servers, where 5 to 15 percent overall utilization is typical, mainframes typically meet their service requirements at 80 to 100 percent utilization levels. The varying nature of application workloads means that a shared infrastructure is more efficient than a set of dedicated servers. Peak mainframe workloads are dealt with by reducing the allocation of resources to lower-priority background tasks and through dynamic reallocation of resources.
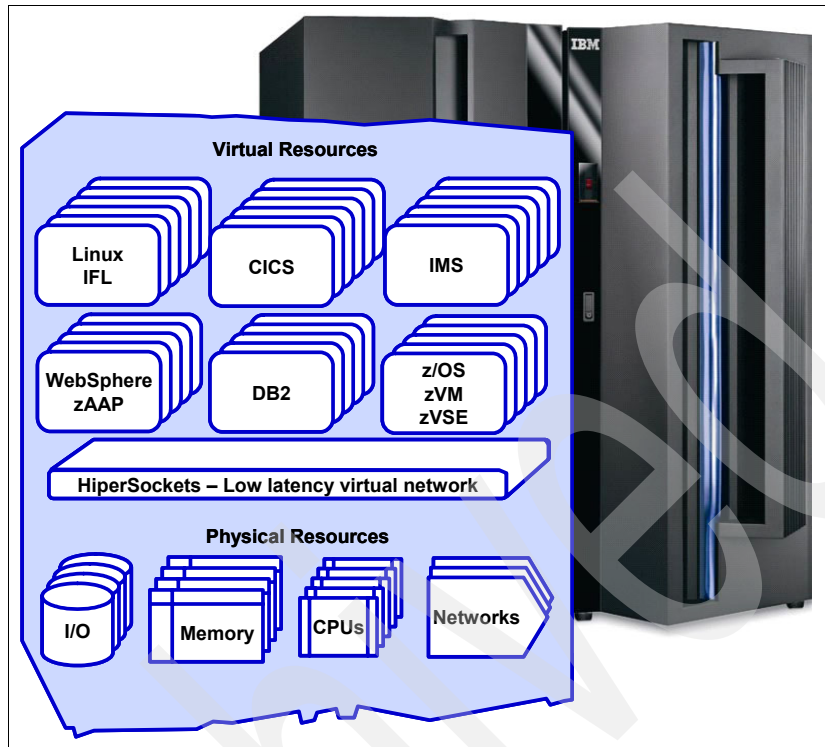
Figure 3-2   Mainframe virtualization capabilities

The virtualization capabilities of IBM Mainframes are diverse (Figure 3-2). z/OS Parallel Sysplex capabilities allow configurations of large numbers of partitions to be run as a single system image sharing resources. z/OS Workload Manager (WLM) controls the distribution of workload over the partitions and prioritization of work within a partition. CPU resources can be dynamically reassigned through Intelligent Resource Director (IRD). WLM and IRD working together provide unique on-demand capacity services. WLM can work with Sysplex Distributor to load-balance workload across the systems in the sysplex, and make the network entry point into the system highly available.

All these capabilities make the mainframe an ideal platform for running mixed workloads of hundreds of different business-critical applications and achieving very high resource utilization. The mainframe manages these diverse workloads while maintaining a consistent performance.

Reliability

System z and z/OS provide an advanced combination of availability and resilience. The combination of redundancy and virtualization delivers

extraordinary levels of application availability and recoverability. The IBM Mainframe architecture protects end users, not only from hardware failures, but from software failures as well. If an application fails, its workload can be picked up by a backup of the same application running on the same physical hardware. On a more granular level, z/OS uses a feature called *Automatic Restart Manager (ARM)* for automatic recovery and restart of services and transactions based on predefined polices.

IBM mainframe software is designed for transactional *integrity* with system-wide two-phase commit. This means that a customer's application data is recoverable, even when hardware and software failures occur. z/OS provides a special set of services for this, called *Resource Recovery Services (RRS)*, which are commonly used by middleware solutions, but which can also be used in custom applications since the APIs are public. When resources in the commit scope are all on z/OS, RRS can provide two-phase commit functionality without the need for distributed transaction technologies such as XA.

Mainframe workloads can be shared by systems running up to 100 km apart, allowing for smooth disaster recovery procedures. And when the time comes to add system capacity, or replace a failed hardware module, those hardware changes can be made without interrupting customer service. Some mainframes run un-interrupted for years, adapting to changing needs, workloads, and applications while continuously in production.

The System z hardware provides a solution with no single points of failure. There is redundancy within the power supply and cooling, there is an inside battery for backup and built-in spare processors and memory. Another hardware advantage is the possibility to do a hardware upgrade or perform hardware maintenance without taking the machine down.

Scalability

Currently, System z hardware can be scaled up vertically to a maximum of 54 CPUs. The z/OS Parallel Sysplex configuration provides additional horizontal scalability. A sysplex can be a cluster of up to 32 z/OS images in different partitions on different System z machines (possibly geographically dispersed) with full data sharing and high availability and recovery.

The machines participating in a Parallel Sysplex can be physically dispersed up to 100 km distance from one another, which gives the capability of having a physically decentralized infrastructure that is logically centralized.

In z/OS, the horizontal scalability is not limited to specific workloads. Any workload can take advantage of the Parallel Sysplex scaling options.

Scalability is fundamental in an on demand world, and the IT infrastructure should be able to scale when business demands more resources. This does not necessarily mean a new server or system image must be integrated to the complex in some way or another, as is required on distributed platforms. Mainframes, as we have seen, have more capabilities to scale on demand. Mainframes are said to *scale in*, while others platforms *scale out*. *Scaling out* means adding resources to a logical system by adding a physical system (or operating system image), thereby increasing the configuration complexity and reducing manageability. *Scaling in* means (automatically) aggregating new resources to the existing infrastructure without increasing its complexity, without demanding redesign of data bases, application patterns, network traffic, and so on, and without the need to bring the system down. This is an additional mainframe differentiator, by design. You can, for instance, add new processors or new storage, or reconfigure the partitions, without any impact on applications, data, and existing infrastructure.

Another key performance and scalability benefit is provided by the *HiperSockets™* virtual TCP/IP network, which eliminates network delays between applications and subsystems running within a mainframe.

Availability

System z has many components to address availability, as does z/OS.

Parallel Sysplex, the clustering solution for the z/OS environment, provides both scalability and availability. A failure of one image in the cluster does not affect the other images, and any specific transaction running on it can be fully dispatched and recovered in any other image, making use of the full data sharing architecture.

For higher availability and disaster recovery purposes, a Parallel Sysplex can be configured in a *Geographically Dispersed Parallel Sysplex™* (GDPS®) mode. There are two GDPS modes:

► *GDPS/PPRC*, a configuration where a Parallel Sysplex is distributed over two sites, connected together at a distance of up to 100 km, with data synchronized and shared continuously. One site (part of the sysplex) is acting as a primary, and the second site is acting as a secondary, in stand-by mode. GDPS controls and automates a full swap to the backup site in case of failures.

► The second mode of operation for GDPS is *GDPS/XRC*, where the distance between sites can be more than 100 km, theoretically without limitation. In GDPS/XRC the sysplex does not span both sites, but instead a full system image is swapped to the alternate site in an emergency situation.

Both modes use *HiperSwap* capability, which provides the ability to activate replicated data in the disaster recovery site without application outage.
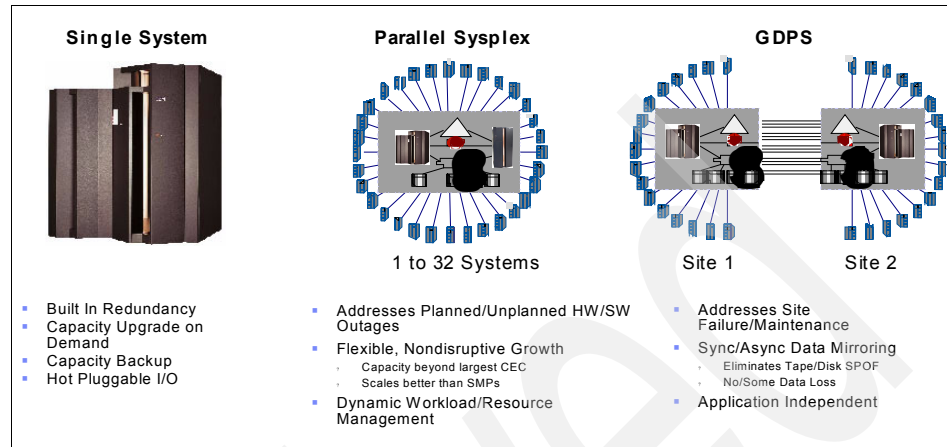

Single System	Parallel Sysplex	GDPS
	1 to 32 Systems	Site 1 Site 2

• Built In Redundancy
• Capacity Upgrade on Demand
• Capacity Backup
• Hot Pluggable I/O
	• Addresses Planned/Unplanned HW/SW Outages
	• Flexible, Nondisruptive Growth
	  › Capacity beyond largest CEC
	  › Scales better than SMPs
	• Dynamic Workload/Resource Management
		• Addresses Site Failure/Maintenance
		• Sync/Async Data Mirroring
		  › Eliminates Tape/Disk SPOF
		  › No/Some Data Loss
		• Application Independent

Figure 3-3   z/OS continuous availability

The System z and z/OS availability features are:

► Unique mainframe clustering technology for maximum up-time (Parallel Sysplex)

► The ability to deploy participating nodes in the Sysplex cluster remotely (Geographically Dispersed Parallel Sysplex)

► The ability to recover virtual Linux servers running remotely (xDR)

► The ability to replicate data real-time at remote locations (PPRC)

Transaction processing

Traditionally, the mainframe has been extremely strong in transaction processing, providing thousands of concurrent online users with the ability to retrieve information and make updates. IBM's transaction management solutions on the mainframe have long been CICS and IMS on the z/OS platform, and TPF as a specialized operating system on the mainframe for airline and financial transactions. WebSphere Application Server for z/OS was recently added to the family of transaction managers on z/OS. Companies have made very large investments in the development of transactional applications, and despite efforts to mimic transaction processing on non-mainframe platforms, these solutions have never provided the scalability and reliability that mainframe transaction processing has.

As we have seen, the z/OS operating system provides transactional services, but this is not limited to applications running in one of the transaction managers.

Resource Recovery Services (RRS) assures integrity by providing transaction and two-phase commit services to any application that requires them.

Batch processing

Another outstanding capability of the mainframe architecture is the running of batch workloads. In the z/OS environment, dedicated subsystems (JES2 or JES3) in combination with special job scheduling software, such as Tivoli Workload Scheduler (TWS), manage batch processing. No other architecture can support batch processing as well as z/OS.

Batch processing is currently still very undervalued. With the requirement to integrate new applications with existing back end systems, batch processing will become more important.

The IBM mainframe is today even more than ever positioned to handle batch processing. With the ability to run batch processes written in the Java language making use of zAAP processors, and with the ability to access back end data either on z/OS or any other server, z/OS has become a very cost-effective platform for running business-critical batch applications.

3.1.3 Additional capabilities of z/OS

z/OS provides a number of additional implementation capabilities for the products running on the z/OS platform. These capabilities significantly increase the Quality of Service to a level above what can be reached on other platforms, while preserving functionality at least the same level.

Openness

The virtualization capability provides the mainframe with openness. The ability to have many different working environments does not mean all of them are the same technology or have the same operating system. Mainframes not only run the traditional operating systems, z/OS, z/VM, z/TPF, and z/VSE, but also support Linux. Once called "proprietary" technology, the mainframe adopted open standards technology as well and is as open as any UNIX-based or Windows® platform. There are no limitations on the mainframe with respect to using standards such as TCP/IP, HTTP, SOAP, and so on.

The power of the openness of the platform is not just in supporting open standards, but in enabling and integrating traditional technologies with the new ones. Core business applications, usually running in IMS or CICS, can be integrated with new style applications using XML and SOAP, thereby bridging the technology landscape evolution and maximizing investments in skill and resources companies had made in the platform.

Cost-effectiveness

Cost is the more sensitive and complex aspect of each analysis. Sensitive because everybody has a different understanding and feeling of what is a reasonable cost, and complex because getting to the total cost of a solution requires a careful analysis of many factors.

There are many studies made by major consulting companies related to cost comparisons among platforms and it is not our intention here to introduce another one. We just want to make some major points to help readers when they make their considerations.

TCA and TCO

Total Cost of Acquisition (TCA) is the buying cost of a certain product. *Total Cost of Ownership (TCO)* is the whole cost involved in a process and it goes much further than just the purchase cost. It considers all the aspects related to installing the product, maintenance, people, training, infrastructure, and so on.

Most companies do not make any distinction between TCO and TCA because lack of knowledge or lack of data makes the full analysis impossible. Various consulting companies that specialize in performing this kind of analysis have come out with very positive results for the System z platform, considering use of the solution over a five year period.

Economy of scale

Because of its centralized model and almost endless scalability, the System z platform provides what is called *economy of scale*, which means that you can add new physical resources (CPU, I/O devices, servers) without the need to proportionally expand the existing infrastructure and the number of people managing it.

People are the most expensive resource in a company, and, in the distributed world, for every additional set of servers or for every additional pool of storage, the amount of human resources required will go up. On the System z platform the number of people required for support is practically flat, and it just increases when very large chunks of additional infrastructure are added.

When talking about scalability, another key differentiator between the System z platform and distributed platforms is the difference between scale in and scale out. Scalability is a major factor in achieving SLAs. Since the rise of the Internet, and consequently of e-business, companies can no longer predict when users surfing the Web will access a corporate Web site and demand IT resources, so systems and applications scalability have become a fundamental factor for e-business success. And here is the point: scalability on System z is accomplished on the same box, what we call *scale in*, while scalability on

distributed platforms is accomplished by adding new boxes, which we call *scale out*.

A solution out-of-the-box

z/OS is not just an operating system, but a package that includes the operating system itself and more than 30 built-in solutions. The full package comes with everything that is required to build a robust IT software infrastructure. Among others components, it comes with security services, communication services, performance and accounting services, compilers, storage management services, an HTTP server, clustering capabilities, and so on. Other operating systems do not provide the equivalent set of services out-of-the-box and companies would have to purchase them as additional products.

IBM flexible charge model

On top of everything we mentioned previously related to cost, IBM has introduced various new ways of charging for mainframe software that have significantly reduced the platform TCO. The most recently added models include specialty processors for Java and DB2 tasks called zAAP and zIIP respectively.

Data proximity

Data proximity delivers additional qualities. When integration logic is deployed on z/OS close to the resources that are being integrated, composition and integration with multiple z/OS resource managers will give optimal performance because data access can be realized cross-memory, requiring no network traffic, and duration of held locks can be minimized. This also significantly increases availability because the configuration will comprise less points of failure. Furthermore, recovery will be much faster in rollback situations. The ability to run transactions using native z/OS services rather than a distributed two-phase commit protocol offers far better performance.

3.1.4 How z/OS extends functionality of software products

In the previous sections we described a number of unique features of the z/OS platform that are available even before installing any additional software products. In this section we describe how these features add QoS to the standard functionality of software products.

WebSphere Application Server, ESB, and Process Server

WebSphere Application Server for z/OS is functionally fully equivalent to WebSphere Application Server on distributed platforms. By taking advantage of WLM capabilities, WebSphere Application Server for z/OS can provide additional qualities in workload qualification and differentiation, and unique scalability and availability. A WebSphere Application Server for z/OS cluster can span all

systems in a sysplex, thereby building on this feature's availability and scalability advantages. z/OS Workload Manager can control starting and stopping of multiple application servers (Servant Regions) in a WebSphere Application Server cluster, depending on the actual workload. WLM can also classify and prioritize incoming workload. Where distributed platforms need special WebSphere XD technology to provide similar functionality, with WebSphere on z/OS this is standard functionality that does not add any complexity to the configuration.

All extended security features provided by z/OS can be exploited in WebSphere Application Server for z/OS. Integration with RACF and other security systems is possible through the z/OS functions and the WebSphere Application Server functions on top of that.

WebSphere ESB, WebSphere Process Server, and WebSphere Portal Server all run on top of WebSphere Application Server for z/OS and can therefore make use of the same unique features offered by WebSphere Application Server for z/OS.

WebSphere MQ and WebSphere Message Broker

WebSphere MQ on z/OS has the unique capability to use *shared queues*. A shared queue is a type of local queue in which persistent or non-persistent messages on that queue can be accessed by one or more queue managers identified to the Parallel Sysplex. Shared queues provide a higher level of availability, scalability, and workload balancing. These are all very important requirements for an ESB.

WebSphere Message Broker runs on top of WebSphere MQ and inherits its QoS. Those capabilities make an ESB on z/OS unique when compared with other alternatives.

CICS and IMS

The z/OS Parallel Sysplex capabilities are extended to the z/OS transaction servers CICS and IMS. By implementing functionality in IMS and CICS, business transactions can exploit the scalability and availability that IMSplex and CICSplex provide.

A CICSPlex® is a set of interconnected CICS regions typically spread over a number of systems in a sysplex, simultaneously processing customer workload and managed from a single point of control.

An IMSplex has similar capabilities for its transaction manager component. The database manager component in an IMSplex can share databases across systems in a sysplex. Within an IMSplex, multiple IMS subsystems are managed as if they are one system.

DB2

DB2 on z/OS takes advantage of several unique z/OS features. The security implementation of z/OS provides integration with RACF at the fine-grained level of row-level security. This feature, called *Multi-Level Security (MLS)*, addresses government requirements for highly secure data that can be shared between agencies on demand, and is adopted now also in commercial environments.

The Parallel Sysplex capabilities of z/OS have made it possible to implement DB2 data sharing configurations. The data sharing function of DB2 for z/OS enables applications that run on more than one DB2 subsystem to read from and write to the same set of data concurrently. DB2 data can thus be shared in a sysplex of up to 60 systems.

A recent announcement is that DB2 workload can be cost-effectively run on a special priced specialty processor, called the *zIIP processor*. The zIIP is designed to help improve resource optimization and lower the cost of portions of eligible workloads, enhancing the role of the System z mainframe as the data hub of the enterprise.

The zIIP is a special processor that will only run specific DB2 workload. The zIIP processor, like the zAAP for Java workloads, does not count for normal licensing charges, thereby making it very cost-effective to run these DB2 workloads on the mainframe.

3.2 Mapping SOA building blocks to z/OS technology

In this section we will describe how the building blocks in the SOA reference architecture as discussed in 2.3, "SOA reference architecture" on page 19, can be implemented with z/OS technology and IBM software products available for the z/OS platform. In the following sections we will go further into detail how z/OS provides unique qualities for an SOA implementation. Refer to Figure 3-4 for an overview of the IBM SOA Reference Architecture with solutions for z/OS.
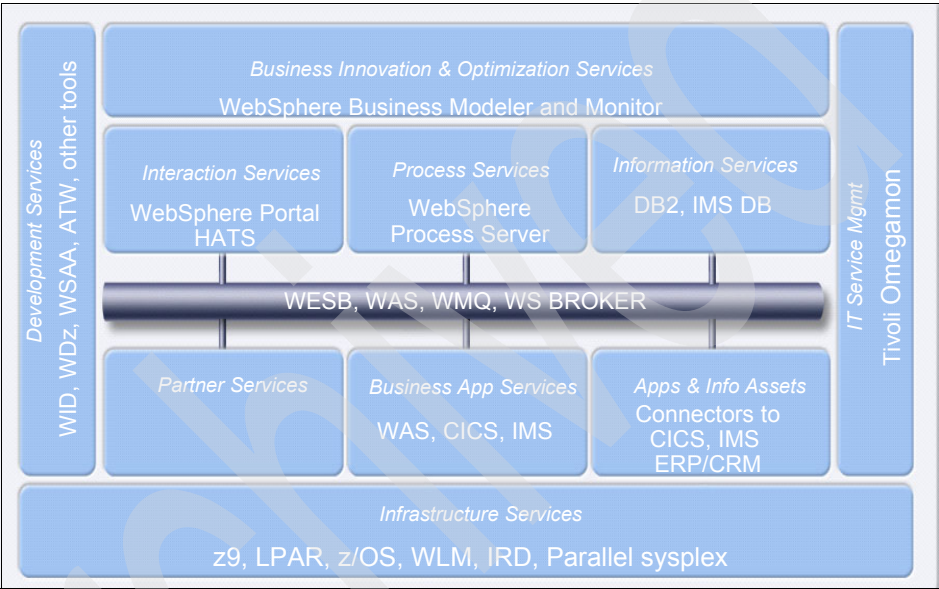

Figure 3-4   IBM SOA reference architecture with solutions for z/OS

Table 3-1 maps the building blocks in the SOA reference architecture to z/OS technologies and products available in the IBM software product portfolio. All these product are available on the z/OS platform, unless specifically noted otherwise in the table.

Table 3-1   Mapping of SOA building blocks to z/OS products

SOA RA building block	Products	Description	Available on z/OS?
Business innovation and optimization services	► WebSphere Business Modeler ► WebSphere Business Monitor	WebSphere Business Modeler and Monitor are integrated products that provide the ability to model business processes, then deploy and monitor the processes and take actions based upon performance indicators, alerts, and triggers. These products typically run on Windows or Linux workstations.	Solutions run on the workstation, but support the z/OS platform.
Development services	► WebSphere Integration Developer ► Rational® Software Architect ► Rational Application Developer	WebSphere Integration Developer provides tools to implement the processes and services defined with WebSphere Business Modeler. Rational Software Architect provides tools to model applications and data, which subsequently can be imported into Rational Application Developer to be implemented. These products typically run on Windows or Linux workstations.	Solutions run on the workstation, but fully support development for the z/OS platform.
IT Service management services	IBM Tivoli and Omegamon suite	These products provide a stable, mature, and functionally rich set of end-to-end system management solutions for the mainframe. For example: ► Tivoli Business System Manager provides end-to-end availability management of the IT infrastructure. TBSM provides the ability to align IT operations to business priorities, thus optimizing IT with business goals. ► Tivoli Omegamon products can improve performance and availability of individual components in the infrastructure, enabling proactive management and tuning. ► IBM Tivoli Composite Application Manager (ITCAM) for SOA can monitor, manage, and control the Web services layer of IT architectures while drilling down to the application or resource layer to identify the source of bottlenecks or failures, and to pinpoint services that take the most time or use the most resources.	Solutions fully support the z/OS environment.

SOA RA building block	Products	Description	Available on z/OS?
Process services	WebSphere Process Server	WebSphere Process Server provides the runtime environment for the processes defined and implemented with WebSphere Business Modeler and WebSphere Integration Developer. Note: WebSphere Process Server is the successor to WebSphere Business Integration - Server Foundation.	Solution runs on z/OS.
ESB	WebSphere ESB and WebSphere Message Broker	The ESB can be built with WebSphere ESB or WebSphere Message Broker, or both. WebSphere ESB provides Web services connectivity, JMS messaging, and service-oriented integration. WebSphere Message Broker is a full-blown ESB including a toolkit and numerous adapters based on WebSphere MQ. Both solutions provide routing, transformation, and event services.	Solution runs on z/OS.
Interaction services	WebSphere Portal Server	WebSphere Portal Server delivers a single, point of personalized interaction with applications, content, processes and people.	Solution runs on z/OS.
Information services	► IMS Database Manager ► DB2 ► WebSphere Information Integrator	These building blocks comprise IBM's premier database management systems IMS DB and DB2. Consolidation of information from different sources is provided by WebSphere Information Integrator, which provides data integration capabilities for business intelligence and business integration: enterprise search, data federation, data transformation, data placement (caching and replication), and data event publishing. **Note**: WebSphere Information Integrator does not run on z/OS, but supports data stored on z/OS.	Solutions available on z/OS.

SOA RA building block	Products	Description	Available on z/OS?
Partner services	► WebSphere Partner Gateway	WebSphere Partner Gateway enables companies to connect trading partners into their business, extending their internal integration beyond the enterprise for real-time business integration.	This solution is not available on z/OS.
Business application services	► WebSphere Application Server ► CICS Transaction Server ► IMS Transaction Manager	WebSphere Application Server allows new J2EE applications and services to be built. CICS and IMS TM provide the ability to extend the existing application portfolio with new applications that can be accessed by and can access other applications through standard protocols such as Web services.	Solutions available on z/OS.
Access services	CICS Transaction Gateway IMS Connect CICS Web services support IMS SOAP Gateway Host Access Transformation Services (HATS)	► CTG provides access to existing CICS programs. The CICS Web services support provides the ability to call Web services from CICS programs as well as the ability to call a CICS program as a Web service. ► IMS Connect provides the ability to call IMS transaction from other environments. ► CICS TS 3.1 has a new implementation for Web services support. CICS can be both a Web service consumer and provider. ► The IMS SOAP Gateway provides the ability to call IMS transactions as a Web service. ► Solution to modernize existing applications running on z/OS.	Available on z/OS.
Infrastructure services	z/OS, z9, LPAR, WLM, IRD, Parallel Sysplex, and so on	The following sections describe the unique capabilities of the mainframe and the infrastructure services it provides to implement the infrastructure services building block.	These are all unique System z or z/OS features.

3.3 Unique z/OS values for an SOA

In the previous sections we demonstrated that z/OS can provide the building blocks to satisfy all functional and non-functional requirements for an SOA deployment. In this section we wrap this up and demonstrate how the combination of all strengths of the z/OS platform provides a unique deployment environment.

In 2.4, "Deployment of SOA components" on page 25 we identified the deployment requirements for a high-demand SOA implementation. In the previous sections in this chapter we defined the strengths of the z/OS platform in general terms. In this section we show how these strengths address the deployment characteristics of an SOA.

Table 3-2   Enabling z/OS technologies for SOA deployment

Deployment requirement	z/OS technology
Accounting, logging and auditing	SAF/RACF SMF RMF z/OS logging
Security	SAF/RACF Cryptographic hardware Network level security MLS
Cost-effectiveness	TCO Manageability/manpower Scalability Integrated management solutions Charge model zIIP, zAAP, IFL specialty processors
Integrity	Transaction processing RRS, 2-PC Parallel Sysplex, GDPS Recoverability, disaster recovery
Predictable and consistent performance	WLM IRD Parallel Sysplex Hipersockets
Standards-based integration platform	All J2EE products available on z/OS, including WebSphere Application Server, ESB, Process Server, Portal Server

Deployment requirement	z/OS technology
Centrally managed platform	z/OS System z Parallel Sysplex Mature management products
Reliable operating environment	System z hardware z/OS stability Parallel sysplex, GDPS

Again, we refer to the diagram that we have used throughout this document and highlight the "Enabling technologies" layer, shown in Figure 3-5.
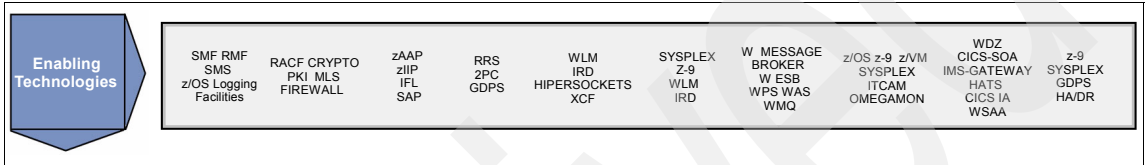

Figure 3-5   z/OS enabling technologies for SOA

3.3.1 Enterprise computing model

In this section we demonstrate how tightly the mainframe paradigm of centralized computing fits the SOA philosophy of an enterprise computing model.

When adopting the SOA model, additional elements should be considered, most of them related to the inherent differences between the centralized and de-centralized computing model.

As we saw in Chapter 1, SOA requires an enterprise-wide view of business processes and IT. The SOA philosophy proclaims advantages through strong corporate IT governance and reuse. The mainframe was made by design to support this view.

As we saw in the previous sections, the mainframe is designed with the primary goal to address enterprise data processing requirements. All its differentiating features—strong security enforcement, the efficient management of multiple and diverse type of workloads, efficient system management, scalability, virtualization, logging, auditing, data sharing, accountability, and so on—were all aimed to address enterprise data processing requirements. The mainframe was designed to run IT as an industrial process. This was basically the mainframe value proposition from the very beginning, and no other platform had the same proposal.

Since the de-centralized and distributed model arose, many attempts have been made to mimic the centralized model, or achieve the advantages of the centralized model. Many technologies have entered the market with these objectives, but so far none have been able to realize the benefits of a centralized computing model in a de-centralized environment. Most of these technologies provide point solutions to increase availability, security, scalability or otherwise, and the actual result is always a significant increase in the number of servers, the number of different software components, the amount of network traffic, the amount of staffing, and as an ultimate consequence, an increase in the complexity of the computing environment. The advantages the software products provide are largely undone by this increased complexity.

For an SOA the problems surfacing in the distributed model described here can become real inhibitors. SOA drives reuse of applications. The services exposing these applications can potentially be reused by any service consumer in the IT ecosystem. This demands a lot of flexibility from the IT environment in terms of differentiation in workload, supported user populations and geographies, availability, scalability, security, and so on. Also, this will drive the IT strategy and moreover the entire IT governance and coordination, from both functional and operational perspectives. Focusing on the operational side of SOA governance, it is clear that without the appropriate capabilities to control resource allocations, vary system capacity, and cross-charge IT costs, such an SOA environment could become uncontrollable technically as well as financially. We demonstrated in the previous sections how z/OS provides unique features to cater to all of the enterprise computing model related issues that an SOA brings to the table.

3.3.2 Virtualization

We saw in Chapter 1 that the flexibility and responsiveness of today's businesses require IT systems that support this flexibility, responsiveness, and reliability in close alignment with business processes. The SOA deployment environment must provide the means to guarantee consistent performance while allowing strong fluctuations in demand through dynamic allocation and de-allocation of resources.

In the previous sections we described a number of technologies that have long existed in the z/OS environment, and also some z/OS technologies that are relatively new that provide exactly these capabilities. Virtualization is one of these technologies.

Where other platforms provide virtualization technologies in a solution-based or product-based manner, z/OS has built-in capabilities for virtualization, and thereby has the capability to virtualize any resource and application running on the z/OS platform. It does this not only by providing the technologies to spread workload over virtualized and real resources, but it also provides the means to

measure which user and application actually use the resources. Through these capabilities z/OS provides the means to allocate IT costs to the actual user of the IT resources. Besides that, z/OS also provides integrated means to manage and control such a flexible environment and keep business processes and IT processes closely aligned.

3.3.3 Mixed workloads

Where virtualization provides the scalability and responsiveness for responsive and flexible businesses, the ability to run mixed workloads is another key factor to support the business drivers for SOA.

IT systems inherently are comprised of on-line and batch-oriented workloads, and usually span several technologies like Web front end applications, middleware layers, and back end components. We showed in 3.1, "How z/OS addresses SOA non-functional requirements" on page 32 how z/OS can manage these different workloads while maintaining consistent and predictable performance. We explained how z/OS can monitor and adjust IT service policies in accordance to business policies. Both capabilities uniquely provide alignment of business and IT and make it possible to monitor and measure the IT system efficiency and contribute to business process efficiency measurements, one of the key promises of an SOA architecture.

3.3.4 Quality of Service

This value of z/OS hardly needs any further emphasis. In the previous sections we described the Quality of Service that z/OS provides and that has made it the platform providing the highest levels of availability, scalability, integrity, and reliability, providing guaranteed service levels and deep end-to-end security integration.

Deploying SOA on z/OS not only brings the SOA functionality to the platform and the platform qualities to the SOA, but what is just as important, z/OS extends the functionality of the SOA components like WebSphere Application Server, WebSphere Process Server, CICS DB2, and WebSphere MQ, and brings a level of quality that no other platform is able to provide.

3.4 Positioning of specific components on z/OS

In this section we summarize the advantages of deploying the most important components in the SOA on the z/OS platform.

3.4.1 ESB on z/OS as the backbone for the SOA

The ESB is the key infrastructure component in the SOA architecture. It is the intermediary through which all service communication runs. It therefore requires the highest level of availability, scalability, security, and performance.

When existing core application functions on the z/OS platform are integrated using the ESB it will need to provide at least the same level of Quality of Service as the back end services it accesses. This is one reason to position the ESB on z/OS. As we saw in "Data proximity" on page 45, co-locating the ESB on the same platform as the back end services provides additional levels of security, availability, scalability, and integrity. This is a second compelling reason to deploy SOA on z/OS. The cost is another factor that will influence the decision where to deploy the ESB. Much of the ESB workload is Java workload that can be off-loaded to zAAP processors, which will lead to a very cost-effective configuration.

We must, however, emphasize that positioning the ESB is not a question of one platform or the other. Although it is possible to let the ESB on z/OS handle all requests, a logical ESB could be spread over multiple platforms. An ESB could be positioned on a distributed server for those services that do not required the high QoS of the mainframe.

3.4.2 z/OS as the Process Engine for core business processes

In most companies the core business applications run on the z/OS platform because these application require the qualities this platform provides. What is more logical than to also run the Business Process Engine component of the SOA, materialized by the WebSphere Process Server product, on the same platform where the services it depends on are deployed?

In fact, the same arguments that apply for the ESB, also apply for the WebSphere Process Server positioning.

3.4.3 z/OS as the platform for core application services

New applications built on the J2EE platform that require the mainframe Quality of Service can be run in WebSphere Application Server for z/OS, thereby taking advantage of the cost-effectiveness of zAAP.

The traditional transaction managers IMS and CICS are also very well positioned for the development of new functionality. Both platforms can participate fully in an SOA by exposing their functionality as Web services, and CICS can also act as a consumer of Web services. This is a very attractive option for the large number of customers that have deep investments in mainframe applications, need the

mainframe for its qualities and facilities like batch processing, and want to reuse these assets.

3.4.4 Leverage existing platform skills

SOA brings new opportunities to the table for existing z/OS skills. As Java grew more and more popular over time, Cobol programmers were more or less challenged to acquire this new skill in order to continue to be competitive in the marketplace. In the SOA model, however, it is not that important anymore in which language a service is written, but much more how easily the service can be integrated. Java still has the most natural support for the standards being used for SOA (such as XML and SOAP), but the gap is closing and Cobol, PL/1, or any other program can be as easily integrated into an SOA as any Java program.

3.5 Summary

In this chapter we have described the actual technologies and features that make IBM System z and the z/OS operating system the best option for an enterprise SOA operating environment. We can now complete the final layer of our diagram, as shown in Figure 3-6 on page 58.
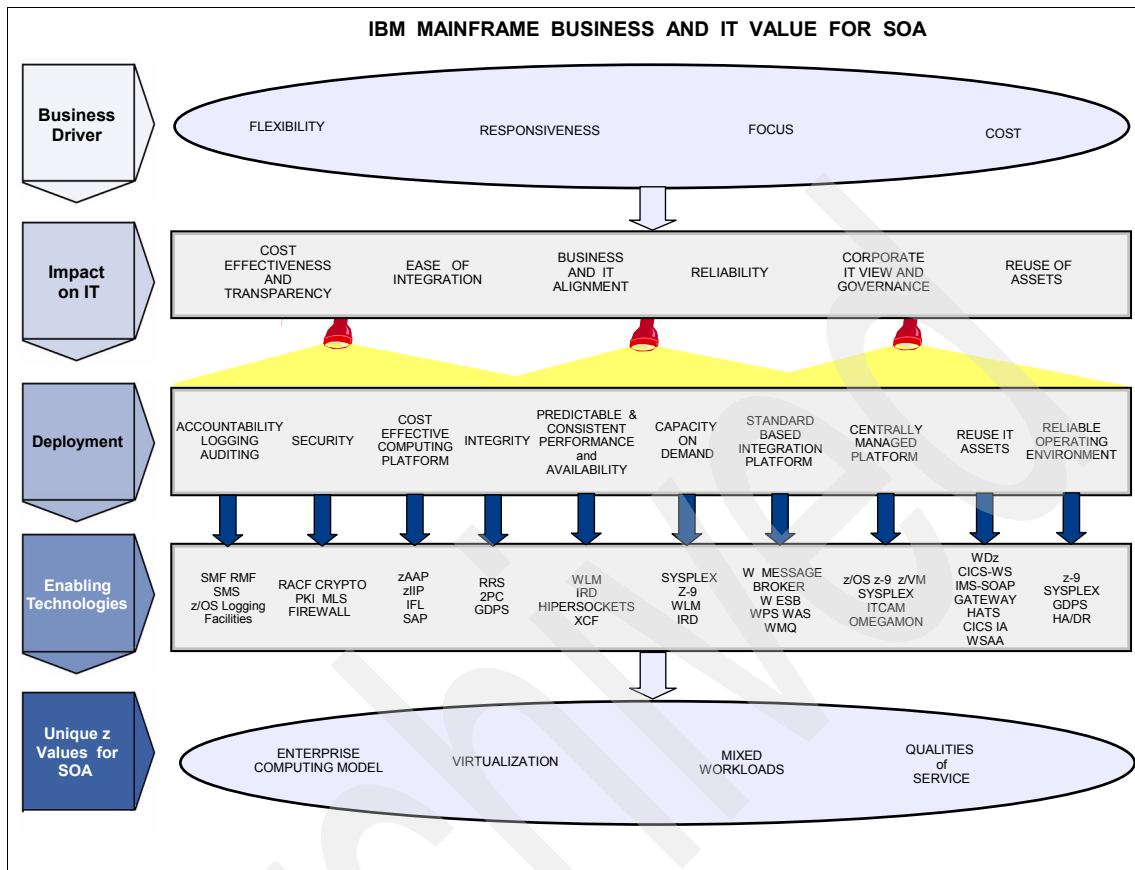
Figure 3-6   Big picture

We can summarize the specific values of the z/OS platform for an SOA implementation as follows.

1. Enterprise computing model

   We have demonstrated that an SOA requires an enterprise computing model. The mainframe was designed to support such a corporate IT governance and a common IT view and it has since then always been the basis for the enterprise computing model for mainframe users.

2. Virtualization

   In order to exploit the business and IT flexibility SOA promises, the infrastructure must support this flexibility too. We have shown that virtualization and on-demand capacity must provide this flexibility and again, these are among the primary characteristics of the mainframe. The

mainframe is better equipped than other platforms to provide this flexibility through its broad set of virtualization capabilities.

3. Mixed workloads

    We have illustrated that z/OS provides right out of the box the ability to guarantee service levels (response time, throughput, and so forth) for specific types of customers and high priority workloads as defined by business needs. There is no other platform that can provide this requirement for your SOA.

4. Quality of Service

    z/OS and the mainframe hardware provide zero-downtime capabilities. The platform can consistently deliver expected service regardless of capacity-constrained environments, unanticipated workload spikes, or failures in applications, system software, or hardware. z/OS can handle unpredictable spikes in mission-critical workload without wasting spare cycles during periods of low and average utilization.

z/OS combines these availability characteristics with the highest security capabilities, integrity, and robustness.

4

The SOA journey on z/OS

In this chapter we discuss the steps required to grow to an SOA and we provide a few sample scenarios.

Implementing a Service-Oriented Architecture is an evolutionary process that does not happen overnight. It is a journey that requires careful planning, and that is accomplished in several stages. In this chapter we discuss an SOA adoption model and how to move from an existing IT model to an SOA model, in granular steps.

4.1 Enablement strategies

As shown in Figure 4-1, there are two principal methodical approaches for the SOA-enablement of the core IT systems: a top-down or a bottom-up approach.
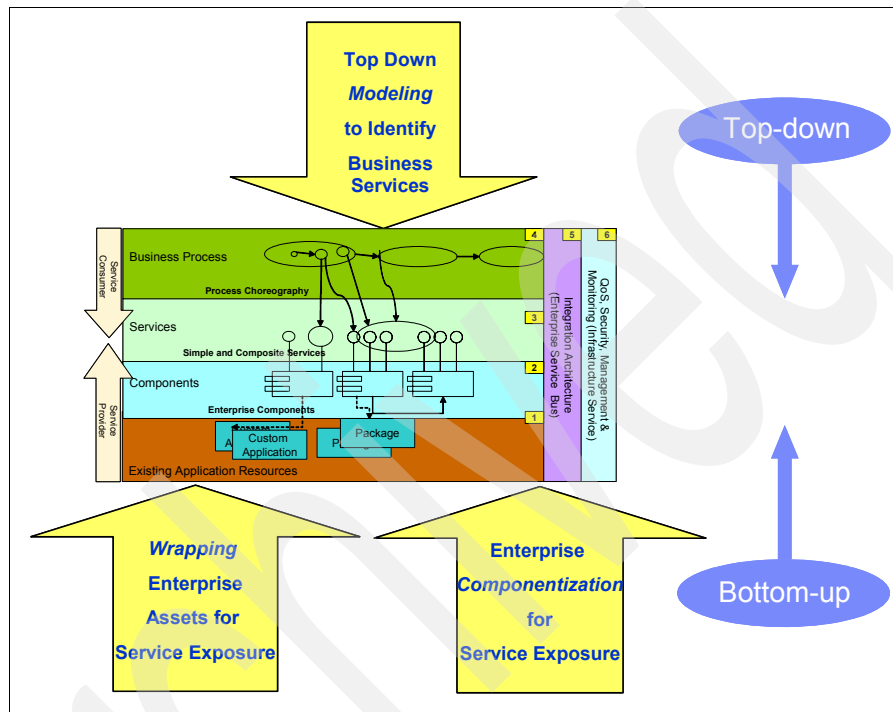

Figure 4-1   SOA enablement strategies

The top-down approach starts at the business level with the modeling of business processes, mapping them to IT processes and systems, and formulating IT requirements. These requirements then define which services can make use of existing applications, which need to be built from scratch, and which services can be created from a combination of the two.

With the bottom-up approach, the existing IT assets are analyzed and it is determined which assets can be meaningfully reused in other processes and applications. The identified functionality may need to be extracted from several existing application components and restructured into new components. Alternatively, the identified IT assets can be wrapped by special programs for the purpose of service exposure.

4.1.1 Meet in the middle approach

In reality, neither the top-down approach nor the bottom-up approach alone will work. In most situations there will already be a number of low-level components available that could be part of a service and at the same time there will be a "new" business view of the services required. So, what will happen in most cases is that an existing component can be mapped "more or less" to a new service, but it will not be quite right. It may need to be generalized, combined with another component, or split up in parts.

So, what we see is that in most cases:

► An existing component cannot be converted into a service that makes sense to the business without making any change to it.

► A new business service cannot be mapped exactly to one existing component.

This is where the "meet in the middle" approach comes in. From one side we have a clean business model with services defined as the business would like to see them, and from the other side we have a collection of existing pieces of business logic that can be technically Web service enabled. The trick is to align those existing components to the business services that have been modeled. This requires some creativity and in some cases an aggregation layer between the process and the business logic components.

4.2 An evolutionary approach towards SOA

Figure 4-2 on page 64 shows the evolutionary path from the current IT environment to an SOA model.
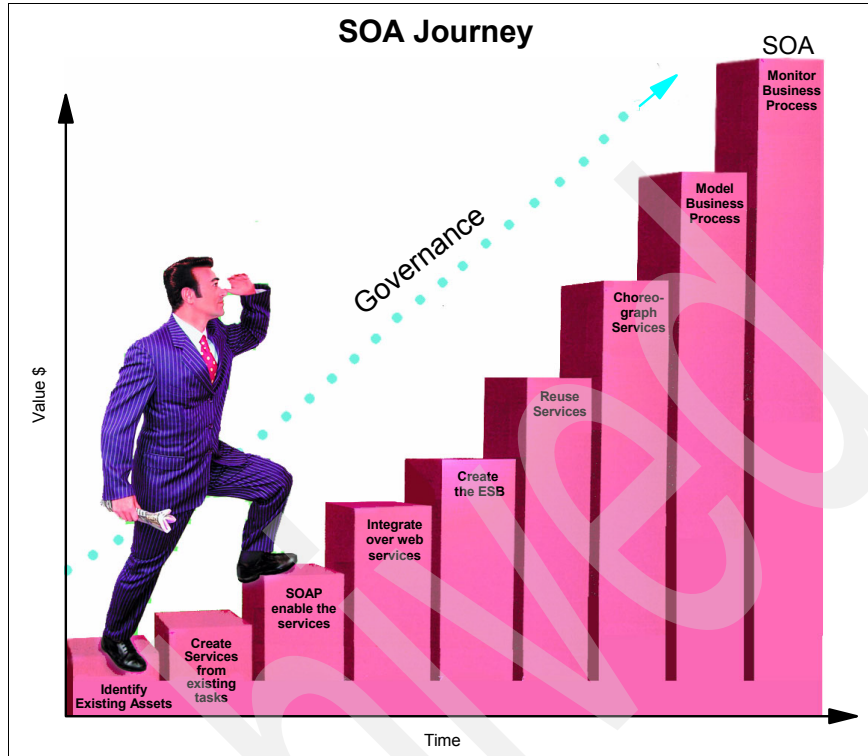
Figure 4-2 The SOA journey

Many businesses rely on mainframes for their mission-critical applications. They have invested significantly in the development of their business applications. These CICS, IMS, and DB2 applications run the vast majority of business transactions executed worldwide. It is these transactions that are becoming eligible components for reuse in a transformation towards SOA.

The steps involved are:

1. Identify existing IT assets.

   In this discovery and identification phase, application repositories are built and existing functionality is analyzed to be leveraged as a reusable asset and service.

2. Create services from existing tasks.

   After identifying functionality that represents reusable business logic, that functionality must be exposed as services.

3. SOAP-enable the services.

Once exposable services have been created, they are made ready for consumption by service requesters, but at this stage we may also want to be able to invoke services from these services. In order to achieve that, the service needs to be SOAP-enabled.

4. Integrate over Web services.

   At this point, a service is ready to integrate and collaborate with other services using Web Services technology such as the SOAP protocol. SOAP can flow over HTTP or JMS using messaging middleware, like WebSphere MQ.

5. Create an ESB.

   To be able to flow an increasing number of service requests in a flexible, secure, and highly available way, a robust infrastructure is required. Also, we still need the capability to have new Web services integrate with existing systems that do not yet support the standard Web services protocols. This requires functionality to perform protocol conversion and data transformation, as well as routing functionality. An Enterprise Service Bus (ESB) provides those functions and has become a hard requirement at this point.

6. Reuse services.

   Reusability is one of the key SOA features and values. An infrastructure is required that facilitates asset reuse by providing a common place for storing service definitions in a *Business Service Repository*.

7. Choreograph services.

   Once there is a considerable collection of services, each one representing a specific business task, business processes can be assembled from this collection of available services in a specific sequence, combining these optionally with human tasks and capabilities for acting upon events. This is what we call *process choreography*.

8. Model business processes.

   This stage and the next one are mostly oriented towards integrating IT with the business and are primarily the terrain of the business analyst rather than the IT specialist. The business analyst defines the business processes in business terms. The analyst models a business process and defines which parts of the process are to be implemented by IT systems and which ones are human tasks.

9. Monitor business process.

   Once the process is in place, a final task is to provide monitoring capabilities to ensure the business process is running as expected and to measure various aspects of the process during execution. This monitoring provides information in business terms that the business analyst can understand. Based on the information business process monitoring provides, the business

analyst can do an assessment on the efficiency of the process, decide on eventual changes to the process, or eventually on outsourcing of the process.

Now that we have provided a high-level overview of an approach to enable an existing IT architecture for SOA, the rest of this chapter goes into the technical aspects of enabling more traditional z/OS technologies and applications for SOA.

4.3 Identify existing assets (Step 1)

In this step a clear insight into the functionality of the existing IT applications is developed. This knowledge may be available and recorded in design and implementation documentation, but more often this type of documentation is non-existent or at least not accurate. Tools must be used in that case to support this discovery phase. IBM provides a number of products in this area. Here we mention a few of these tools, a detailed discussion is outside the scope of this paper.

WebSphere Studio Asset Analyzer can be used to discover existing IT assets and functionality on Windows, AIX®, and z/OS platforms. The product can search a number of languages and environments including COBOL, PL/I, C++, Java, CICS, IMS, and DB2.

The Asset Transformation Workbench can be used to modernize existing code by means of refactoring.

CICS Interdependency Analyzer specifically aims at gathering runtime information on CICS programs and relations between programs.

4.4 Create services, enable and integrate services (Steps 2, 3, and 4)

When the reusable functionality has been identified, this functionality must be made available in reusable components. The functionality may be readily accessible from existing components, for example in existing CICS or IMS programs, but it can also be the case that the functionality is spread out over multiple programs or transactions and may not be as easily made available for reuse.

Depending on the situation, the technology must be selected to expose the existing functionality in the SOA.

In most mainframe environments, core applications run in CICS and IMS. These transaction servers typically expose their applications via 3270 data streams (screens) or via programing language data structures like Cobol Copybooks or PL/I include files. We have several options to enable existing functionality:

1. When the functionality is only available in IMS or CICS screen interactions (online transactions), a solution is required to hide the screen details for the end user and expose this functionality through a Web service interface. The *WebSphere Host Access Transformation Services (HATS)* product has been around for several years; it provides capabilities to Web-enable and Web-servicify 3270 applications.

2. When the functionality is available from a single CICS program, CICS Web services and the IMS SOAP Gateway provide the ability to expose these programs as Web services. In order to service-enable a CICS or IMS program, we have to match these existing programming language structures to an XML schema and create the respective WSDL that is required for use in a Web services environment. IBM has tools to support the tasks of creating Web services from existing application assets, as we demonstrate further in this chapter.

3. When the required functionality is to be provided by a combination of 3270 screen interactions and CICS program invocations, the CICS Service Flow Feature (SFF) can be used to orchestrate these CICS interactions and expose the functionality as a single service. The CICS Service Flow Feature is a standard feature in CICS V3 that allows access to existing CICS transaction and application interfaces (both screen-oriented interfaces and call interfaces) using non-invasive techniques, so that the CICS application assets orchestrated by the service flow do not have to be altered to support the CICS business service flow.

In the following section we go deeper into the details of the technologies mentioned.

4.4.1 WebSphere Host Access Transformation Services

Terminal-driven programs are designed to be invoked directly from an IBM 3270 display station or a similar buffered terminal device. Invocation usually corresponds to a single interaction in an end user dialog, starting with receipt of a message from the terminal and ending with transmission of a response message to the same device.
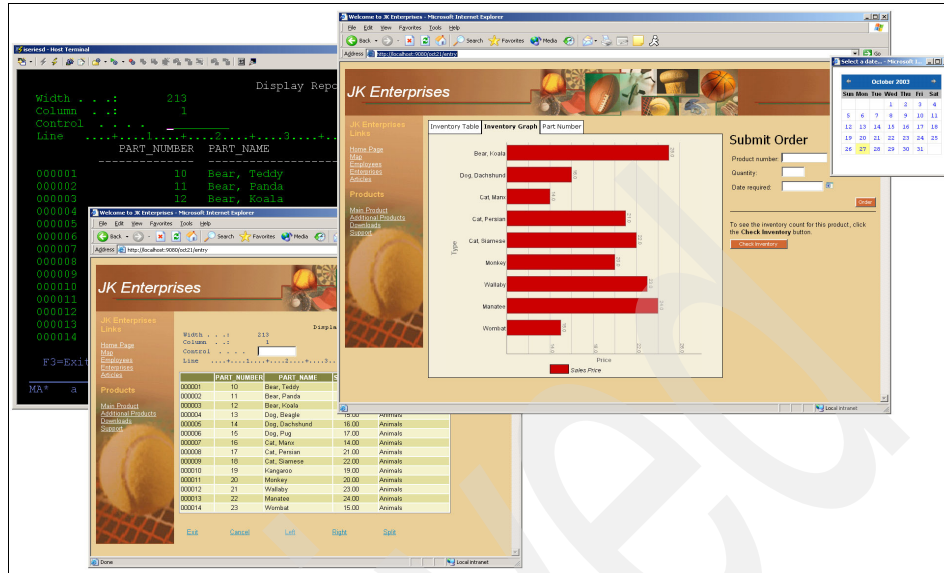
Figure 4-3   WebSphere Host Access Transformation Services

HATS is a solution for Web-enabling 3270-oriented applications. It has been around since the very beginning of the Web era, part of that time under different names. One of the great values HATS delivers is its ability to embed the technology evolution and relieve programmers from the complexity of continuously modernizing existing applications.

HATS first started by just converting 3270 screens to HTML on the fly. Later on, it introduced Javascript support on the generated HTMLs, then introduced JSP™ and Java code for macro development support, and finally, its now has the ability to "Web-servicify" existing JSP code and make it available as a service.

So, it allows a 3270 screen to be exposed as a service and that way reuse this kind of existing mainframe asset.

When using HATS, J2EE applications are created that run inside WebSphere Application Server on z/OS or on a distributed platform.

4.4.2 CICS Web services

In the previous section we showed how we can use HATS to enable applications that are screen-oriented. In this section we look at how CICS callable programs can be enabled. A very elegant solution is provided by the CICS Web services feature.

Web services were first supported in CICS V2.3 with the SOAP feature. With this feature it became possible to call Web services from CICS programs. In CICS TS V3.1 the support for Web services has become an integral part of CICS and provides support for standards such as SOAP 1.2 and WS-AtomicTransaction. With CICS Web services support CICS has the ability to act as a service provider and a service requestor, and it can thereby fully participate in an SOA world.

Development tools

For enablement of CICS programs as Web services, the CICS programming language structure (for example, COBOL) must be matched with an XML schema to create the WSDL describing the Web service. Applications send requests to other applications and receive responses using the CICS COMMAREA interface. Tooling is required to assist programmers and service developers in generating the required constructs.

In order to support this, IBM provides two development tools: the *CICS Web Services Assistant* provided with CICS 3.1, and *WebSphere Developer for zSeries (WD/z)*. The Web Services Assistant provides basic functionality for Web services enablement. WD/z provides a large set of additional customization and optimization options, and in general provides all development facilities needed for the development of applications on z/OS.

CICS Web Services Assistant

CICS Web Services Assistant is a set of batch utilities that can help you to transform existing CICS applications into Web services and to enable CICS applications to use Web services provided by external providers. The assistant supports rapid deployment of CICS applications for use in service providers and service requesters, with a minimum of programming effort. The assistant can create a WSDL document from a simple language structure, or a language structure from an existing WSDL document, and supports COBOL, C/C++, and PL/I. It also generates information used to enable automatic runtime conversion of the SOAP messages to containers and COMMAREAs, and vice versa.

WebSphere Developer for z

WebSphere Developer for z (WD/z) provides a comprehensive set of capabilities for mainframe development, such as Web development, integrated mixed workload, or composite development. WD/z can generate CICS WSBind and language artifacts for conversion between WSDL, XML, and language-specific data formats in a Web services environment, and allows for customization of the generated artifact to the needs of the developer.

CICS as a service provider

CICS V3 supports Web Services sent over the ubiquitous HTTP and WebSphere MQ transports for deployment options dependent on application and IT

requirements. The setup and systems management in CICS is via new resource definitions and configuration files.

How CICS acts as a service provider is illustrated in Figure 4-4. A SOAP request is received by the HTTP listener (called CICS Web support), if the request is sent through SOAP over HTTP, or the WebSphere MQ trigger monitor for CICS, if the request is sent through SOAP over JMS. In both cases the request is passed on to the SOAP for CICS pipeline to process the SOAP architected headers, set up the transaction and security environment, and call the target message adapter with the XML message. The message adapter transforms the XML message into a COMMAREA before calling the business logic. Once the business logic completes, the message adapter creates an appropriate XML message, which is passed back to the pipeline to be wrapped in SOAP headers and returned to the Web service consumer.

Figure 4-4 illustrates the flow and the CICS resources necessary to expose an existing application program as a Web service, thus allowing CICS to function as a service provider.
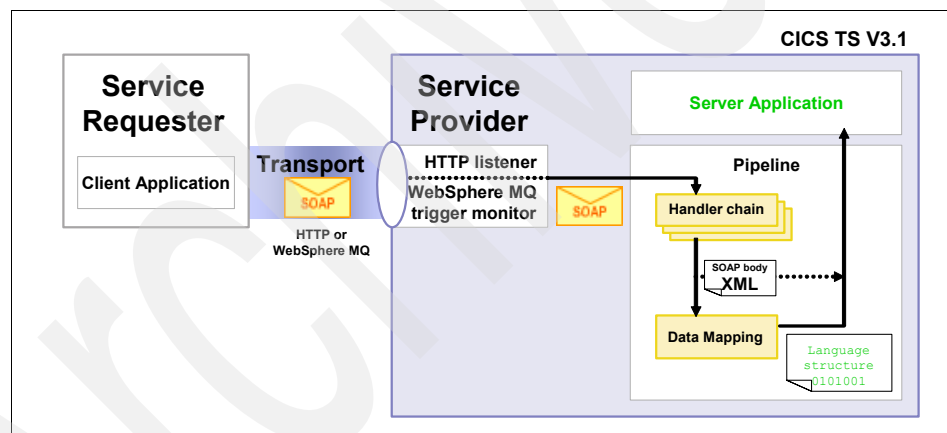

Figure 4-4   CICS as a Web service provider

The CICS Web Services Assistant or WebSphere Developer for z generates a WSDL document and a WSBind file from the language structure or copybook of the business logic application program. The WSDL document is used to create the service requester. The WSBind file is used to tell CICS which program is to be invoked, the interface for that program, and how to map the XML message to the interface.

A resource definition is required to define the transport; both HTTP and WebSphere MQ can be used as transports. For HTTP, a CICS TCPIPSERVICE

definition is required. For WebSphere MQ, a request queue must be defined with a QLOCAL definition.

CICS as a service requester

Figure 4-5 illustrates how CICS operates to invoke a Web service. The WSDL document for the remote service provider is processed through the Web Services Assistant or WD/z. This generates the WSBind file, a language structure which is used as the interface from the service requester program. There is a PIPELINE resource pointing to a pipeline configuration file, and a WEBSERVICE resource.

The service requester program issues an EXEC CICS INVOKE WEBSERVICE command, passing the request language structure via a CHANNEL interface. Information in the corresponding WEBSERVICE resource is used to convert the language structure into a SOAP message. The SOAP message is passed through the pipeline, invoking handler programs according to the pipeline configuration file. It sends the request SOAP message to the remote service provider either via HTTP or WebSphere MQ. When the response SOAP message is received, it is passed back through the pipeline, converted back into a language structure, and passed back to the service requester program.
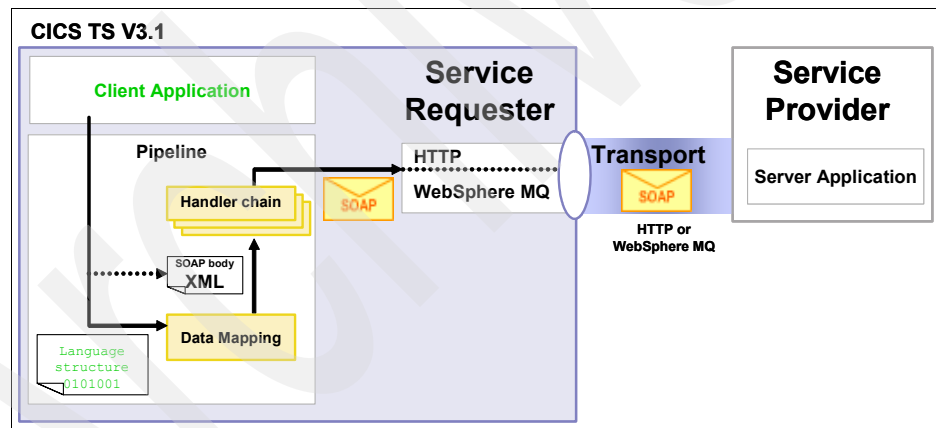

Figure 4-5   CICS as Web service requester

4.4.3 CICS Service Flow Feature

The CICS Service Flow Feature provides the ability for CICS programs to be invoked as services. It is, however, also possible to build Web services in CICS on top of a mix of existing terminal-oriented transactions and callable CICS programs.

One option to create Web services from existing CICS 3270 screen and program oriented application assets is to use the CICS Service Flow Feature. The CICS Service Flow Feature is a business service integration adapter for all CICS applications. It offers both tooling and run-time components. These components enable the creation of CICS business services for integration into an SOA and into business process collaborations.

The CICS Service Flow Feature provides the capability to implement CICS business services by composing a sequence of CICS application interactions using:

- A graphical modeling integrated development environment that enables the creation of CICS business services by composing a flow of CICS application interactions.

- A generation capability that transforms the composed flow of CICS application interactions to form a run-time application, highly optimized for the CICS environment, that retains the inherent qualities of service provided by the existing CICS application implementation.

- A run-time component, known as the CICS Integrator Adaptor, that extends the CICS Transaction Server for z/OS (CICS TS) environment. It offers adapters that exploit CICS interfaces to invoke the CICS terminal-oriented transactions and COMMAREA programs as required by the service flow.

CICS business services built with and hosted in CICS Service Flow Feature expose business function interfaces that can readily be published as Web services in an SOA by exploiting the Web services capabilities of CICS TS V3.1. The CICS application assets orchestrated by the service flow do not have to be altered to support the CICS business service flow.

Additionally, these business services can be integrated as process steps in a business process orchestrated by a business process engine such as WebSphere Process Server.

4.4.4 IMS support for SOA

For IMS the IMS SOAP Gateway is a Web service solution that integrates IMS assets in an SOA. The IMS SOAP Gateway enables IMS applications to inter-operate outside of the IMS environment, through SOAP, to provide and request services, thereby enabling IMS applications to be invoked as Web services.

The development capabilities of WebSphere Developer for z again are used to generate the services description in a WSDL file as a Web service.

Figure 4-6 shows a specific configuration in which different clients connect to IMS applications over the SOAP protocol.
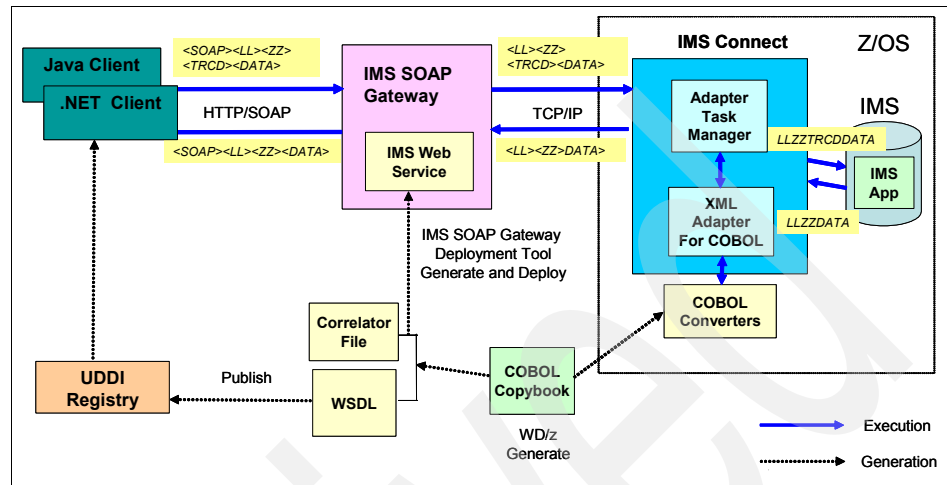

Figure 4-6   IMS SOAP enablement sample configuration

A SOAP client application submits a SOAP request via the HTTP protocol to the IMS SOAP Gateway Server to invoke an IMS transaction. The SOAP message contains the input data for the IMS transaction in XML format.

The SOAP message is received by the HTTP/SOAP endpoint of the IMS SOAP Gateway. It passes the SOAP message to the SOAP Processor. The SOAP Processor opens the SOAP envelope to first retrieve the namespace URI that corresponds to the unique service URN. The URN is used to locate the internal service file to find out the internal service information for running the IMS transaction. Then the SOAP body that contains the input data of the IMS transaction request is also retrieved. Both the internal service information and the input IMS transaction data are passed to the Gateway connector for further processing. Based on the connection and interaction information that is being passed by the SOAP Processor, Gateway Connector builds the corresponding message to IMS Connect. The data portion of the message will contain the payload data from the original message. The message is then sent to IMS Connect via a TCP/IP connection.

After IMS Connect receives the message, the Adapter Task Manager inside IMS Connect calls the XML COBOL adapter to convert the data and send it on to IMS OTMA, which invokes the IMS transaction. The IMS application is scheduled to run and sends the output data back to IMS OTMA and then to IMS Connect.

When IMS Connect receives the output data, the Adapter Task Manager invokes the XML COBOL Adapter again to convert the message back from COBOL

application data structure to XML format. The newly XML-formatted output message is then sent back by IMS Connect to the SOAP Gateway.

The Gateway Connector receives the output message and returns it to the SOAP Processor. The SOAP Processor builds an output SOAP response message containing the output IMS transaction data. The HTTP/SOAP Endpoint then sends this output response message back to the SOAP client application via HTTP.

At the time of this writing, IMS can act only as a Web services provider, not as a requester.

4.5 Create an ESB and reuse services (Step 5)

Now that we have identified our reusable assets and enabled them for Web services, the next step is to remove point-to-point relations and ad-hoc conversions and adapters and commence with the implementation of an ESB that provides the infrastructure to handle mediations, protocol conversions, data transformations, and routing. WebSphere ESB provides these functions on the basis of Web services, but also allows connectivity by joining up with the capabilities of products like WebSphere MQ, WebSphere Message Broker, and WebSphere Event Broker.

We show in this section how an ESB can be created on z/OS by looking at a typical application pattern found on z/OS.

4.5.1 Pattern for WebSphere ESB and CICS integration on z/OS

Current situation

In this scenario we present a virtual company whose current application portfolio comprises thousands of CICS applications. The CICS programs are accessed through 3270 emulators and through MQ messages. The company is running their production systems in a parallel sysplex configuration with CICS running in a CICSplex and DB2 in data sharing mode. MQ uses shared queues to exploit the sysplex capabilities. Figure 4-7 on page 75 shows the configuration. The sysplex facilities are not shown in this picture.

Requirements and solution outline

The company wants to create services based on existing CICS programs. To do that they have decided to implement WebSphere ESB on z/OS. In WebSphere ESB the mediation flow components are deployed, providing the mediation functionality between client requests and service implementations in CICS V3 and WebSphere Application Server for z/OS.
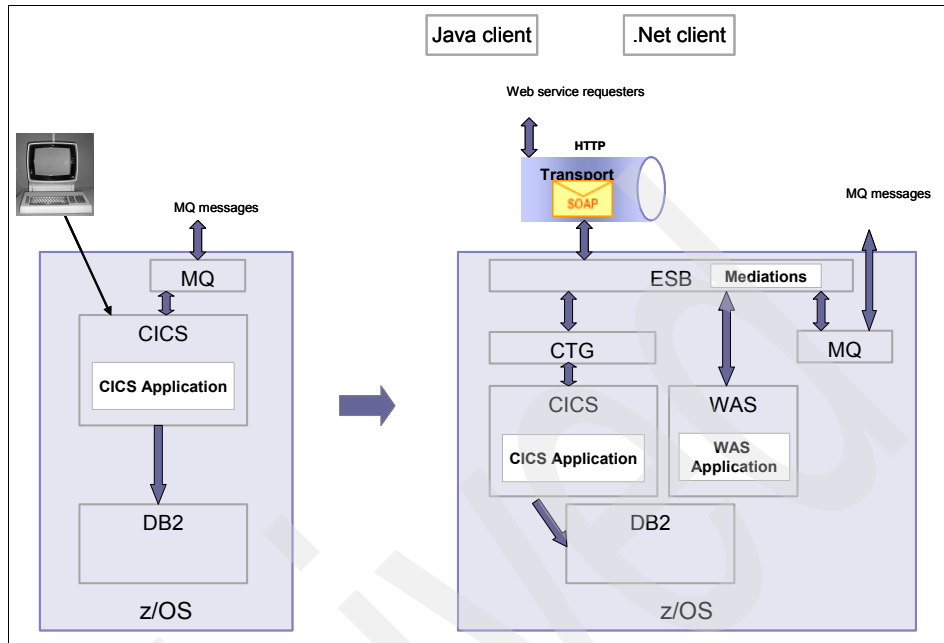
Figure 4-7   ESB on z/OS

The customer has requests from Java and .Net clients on the Internet (different technologies are used by different business partners) that are coming in through the SOAP/HTTP protocol. The requests pass a corporate firewall and are validated in the DMZ (not shown in the picture).

Once the security checks in the DMZ are done, the client request flows to the Sysplex Distributor, which provides load balancing and high availability for inbound network clients reaching the Parallel Sysplex infrastructure on z/OS. Based on predefined rules in z/OS Workload Manager (WLM), Sysplex Distributor identifies the most suitable ESB to receive the inbound request. Sysplex Distributor and z/OS WLM exchange information about the response times of the defined applications, which allows for dynamic adjustments to the routing decisions made by Sysplex Distributor.

For high availability and load balancing two instances of the ESB are configured, one of which is a clone of the other. The ESBs is responsible for data transformation, routing, and data mediation.

In WebSphere ESB so-called *mediation modules* are invoked that transform and route the requests to the target Web services provider, either implemented in CICS or WebSphere Application Server for z/OS.

The ESBs connect with WebSphere Application Server to access services implemented in J2EE applications, and also with the CICS Transaction Server in order to access Web services implemented by applications in CICS. WebSphere Application Server and WebSphere ESB are pictured as two separate boxes in Figure 4-7. In fact, they will most likely be deployed on separate application servers, but in the same WebSphere cell.

The CICS environment is configured in a CICSPlex mode, which provides high availability and load balancing within the CICS configuration. WebSphere Application Server is similarly configured in a Network Deployment configuration, also to provide load balancing, high availability, and fault tolerance.

The invoked CICS and WebSphere transactions access DB2 databases. DB2 is configured in respective sharing modes, which enables full data sharing over the systems included in the Parallel Sysplex.

Choices

For this scenario it was decided not to use the CICS Web services feature to access CICS functionality. CICS programs were unlikely to be reused individually, rather they would be aggregated in mediation scenarios with other CICS transactions and other services to form a logical reusable service. Therefore CICS programs are accessed from the ESB.

Advantages of deployment of the ESB on z/OS

In this section we summarize what we have under the hood that makes z/OS the ideal platform on which to deploy the configuration discussed in the previous section.

▶ Security.
  Depending on the actual client security chosen, end-to-end security can be implemented. Since all components are running in the same system (sysplex), no security context switch is required and all activities performed by the requester are executed under this user's credentials.

▶ End-to-end workload management and load balancing, from the Sysplex Distributor to the back end database, including Web services.

▶ High availability is a key characteristic in this SOA infrastructure, therefore every component is configured in a redundant configuration using sysplex capabilities.

▶ Manageability.
  All components run on the same system (sysplex), which makes system management, problem determination, performance analysis, and capacity planning a much easier task.

► Cost.
WebSphere ESB workloads are dispatched on the zAAP processors, which significantly improves the TCO of the solution and leaves more room for other applications running on general processors.

► IT governance.
Apart from the Web applications running in this environment, this customer can continue running other core applications such as financial reports, HR applications, business intelligence solutions, batch workloads, and so on, all sharing the same resources and data.

► Accountability.
The z/OS facilities SMF and SMS, both z/OS built-in components, have the ability to register every single resource usage and report it for performance and chargeback purposes.

► Optimization.
Elimination of unnecessary network traffic is accomplished through the typical Parallel Sysplex connectivity options.

► Scalability.
Based on the potential business success customers are expecting to have from this flexible, responsive, and open architecture, companies can start to plan having additional business partners join the business. The question a CIO will have is: suppose my business increases by 35%, what is the impact on my IT resources? What should I change in my infrastructure to accommodate it and how long is it going take me to realize that additional configuration? The question is very pertinent, but for the z/OS platform the answer is relatively simple. All that has to be done is to dynamically add more resources.

Conclusion

Now, lets consider having the same architecture and the same QoS on a decentralized infrastructure. If the ESB and WebSphere Application Server are running on other platforms accessing CICS and back end data, how complex would it be? What would be the final TCO? Is this new environment end-to-end manageable?

4.6 Choreograph processes (Step 7)

Choreographing business processes is the next step in the SOA journey. In this section we will present two typical patterns for SOA enablement on z/OS.

4.6.1 Pattern for integrating IMS services into choreographed processes

In this scenario we describe how an existing IMS application infrastructure can be integrated in a business choreography.

Current situation

In this scenario a virtual company's current application portfolio is comprised of IMS/COBOL programs. The IMS programs are accessed through 3270 emulators and through MQ messages. The company is large, runs development, test, acceptance, and production systems in parallel sysplex configuration, and IMS systems are configured in an IMSPlex. MQ uses shared queues to exploit the sysplex. Figure 4-8 shows the configuration. The sysplex facilities are not shown in the picture.
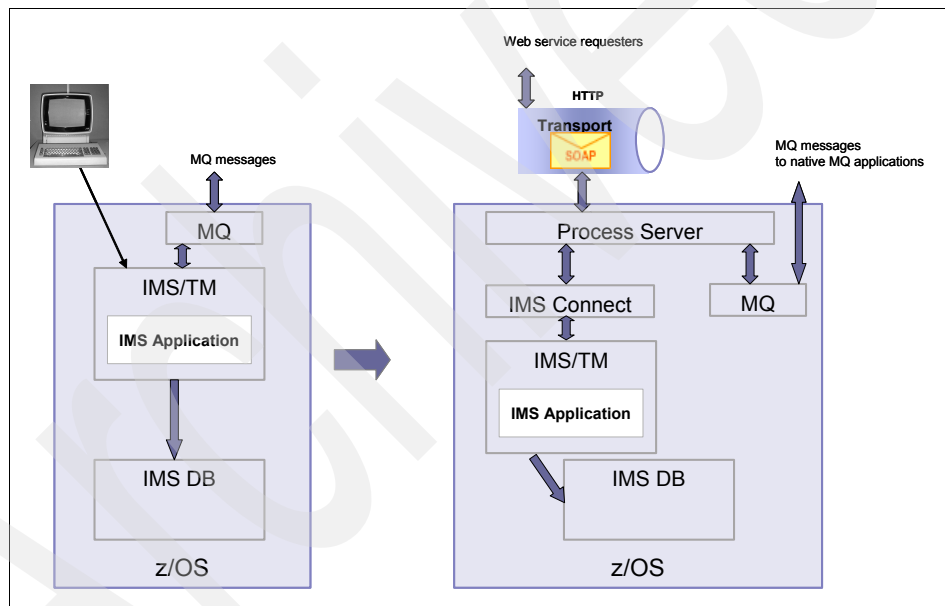

Figure 4-8   IMS and process choreography integration

Requirements and solution outline

This virtual company wants to enable IMS transactions for access through Web services because, among other reasons, it needs to integrate business processes with partners. To accommodate partner integration, a number of existing IMS transactions need to be exposed, likely with some slight changes, as Web services. The company sees other integration issues that will be solved through the implementation of business choreography. They made the logical

choice to run WebSphere Process Server on the same platform where their core applications run because they need the availability and scalability the z/OS platform provides and see big advantages in running the business process server and ESB local to IMS, MQ, and DB2. They also see TCO advantages by running their Java workload on zAAP processors.

The native MQ option needs to be retained for distributed applications using native MQ connectivity that will not be SOAP-enabled (at this stage).

The company wants to reuse their user registry in RACF, without the need for extra administrative procedures for mapping solutions; therefore, direct integration with existing RACF user registry is necessary.

The planned services will require 2-phase commit over MQ, IMS, and DB2 resources.

WebSphere Process Server must provide the same QoS in its business processes as is provided by the current IMS applications, that is 7x24. The Process Server solution must be easily scalable for planned future projects, without the need to increase the number of physical machines or LPARs in the configuration.

Choices

The company chose not to use the IMS SOAP Gateway, mainly because the functionality in the IMS programs needed was not expected to be reused individually, but aggregated with other IMS and non-IMS transaction as new services in WebSphere Process Server. IMS Connect is used to call existing IMS functionality from WebSphere Process Server.

The Web service implementation requires, besides the reuse of functionality in IMS applications, the data mapping facilities in WebSphere Process Server, and message interaction with MQ.

Specific advantages of positioning on z/OS

Besides the availability and reliability advantages we identified in 4.5.1, "Pattern for WebSphere ESB and CICS integration on z/OS" on page 74 of running on z/OS, this solution provides the company with some additional, unique capabilities:

► Two-phase commit over RRS, not requiring distributed transaction mechanisms like XA

► Security integration with RACF without identity mapping solutions and extra user management

► Data proximity, which increases efficiency of data access to DB2 data and CICS transactions

4.6.2 Pattern for integrating CICS services in choreographed processes

In this scenario we describe how WebSphere Process Server can be deployed on z/OS, and how WebSphere Application Server and CICS can participate in the SOA by providing the Web services that make up the business processes. The solution is depicted in Figure 4-9.
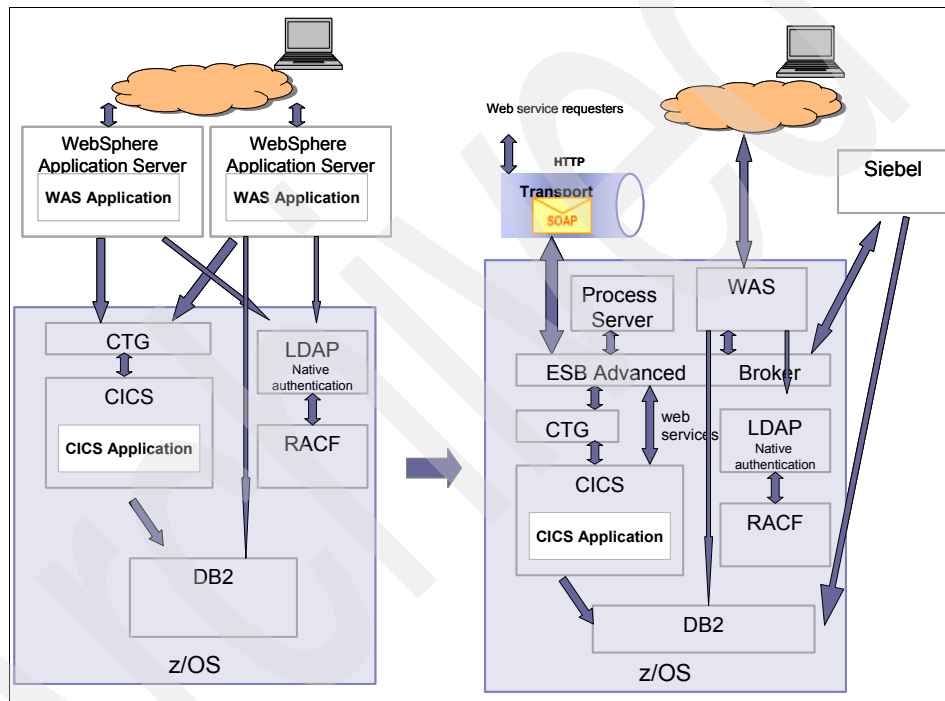

Figure 4-9 Scenario using WebSphere Process Server and CICS Web services

Current situation

This virtual company has a traditional CICS environment. The end users access the application through "green screen" 3270 emulators. The CICS applications are currently integrated with front office applications written in WebSphere Application Server on a distributed platform. WebSphere Application Server on the distributed platform accesses CICS functionality through the CICS Transaction Gateway. WebSphere Application Server uses LDAP on z/OS with native authentication in order to be able to establish a single security context with the CICS and DB2 subsystems on z/OS. WebSphere Application Server uses

DB2 for z/OS as its data store. The configuration exploits a Parallel Sysplex (not shown in Figure 4-9) in which DB2 is configured in data-sharing mode and CICS is clustered in a CICSplex. WebSphere Application Server on the distributed platform is configured for high availability through load balancers on the front end and WebSphere WLM on the WebSphere nodes.

Requirements and solution outline

The company has decided to replaced their current CRM application with a new CRM system (Siebel). Siebel will use DB2 on z/OS to exploit the unique scalability and stability characteristics of DB2 for z/OS. Existing and new applications will access the functionality in the new CRM through Web services invocations. Also, some CRM functionality requires access to existing applications; this will also be realized through Web services. The latter can only be realized because CICS programs can be exposed as Web services using the Web services functionality in CICS. Since the company also needs to invoke Web services from new CICS transactions, such as for new functionality accessing the CRM system, the availability of the Web services feature came in handy.

Another requirement is to be able to integrate human tasks with automated tasks and to choreograph the entire business process as a whole.

The company currently has very stringent availability requirements, which boil down to 7x24, with very limited planned maintenance slots. The new solution must at least provide the same level of availability.

Additionally, this company has a large number of distributed servers, which they find very expensive to manage to provide the availability as needed by the business. They have done research on WebSphere on z/OS and have decided to start consolidating their WebSphere transactions on z/OS because they see major management and availability advantages, and also a TCO advantage because they can exploit the zAAP specialty processor for much of the WebSphere application workload. They have gone through a zPSG exercise and estimated new capacity.

Choices

The company will implement business process choreography through WebSphere Process Server on z/OS because of its QoS advantages. The Process Server solution must be scalable for planned future projects. They will continue reusing LDAP on z/OS with the existing native authentication solution RACF user registry.

It was decided to use Web services in CICS, but also to use CTG for those Web services implementations in the WebSphere Process Server for which SOAP

invocations to CICS are not necessary. This is true in those cases where the functionality in CICS is not found reusable as a stand-alone service.

Specific advantages of positioning on z/OS

Besides the obvious availability and reliability advantages of running on z/OS, this solution provides the company with some unique facilities:

- ▶ Two-phase commit over RRS, not requiring distributed transaction mechanisms like XA
- ▶ Security integration with RACF without identity mapping solutions and extra user management
- ▶ Data proximity which increases efficiency of data access to DB2 data and CICS transactions
- ▶ SOAP access to and from CICS applications without the need for any other middleware solution

4.7 Modeling and monitoring business processes (Steps 8 and 9)

Business modeling and monitoring are key activities in an SOA. Designing and implementing processes and the underlying services is not just a one-shot effort. It is a continuous iteration of aligning the IT to the business and optimizing this IT environment. Therefore, modeling and monitoring are a mandatory part of the SOA formula. In the following sections we further explain those two activities.

4.7.1 Business process modeling

To improve how companies do business, business analysts are responsible for designing and developing new business process models, or adapting existing models. These process models provide a well-defined interface for the transformation of business processes into service-oriented applications and they provide the means for effective business monitoring.

The tooling IBM provides to support business modeling is *WebSphere Business Modeler*. Using this tool a business analyst can graphically model processes across people, partners, and applications. It can be used to redesign processes as business needs change and simulate and validate modeled processes.

The processes defined in WebSphere Business Modeler can subsequently be translated into IT terms through the generation of executable application code that can be further implemented using the development tools that are used to

build choreographed business processes, such as WebSphere Integration Developer.

4.7.2 Business process monitoring

Once the business processes are modeled, we need to monitor and adjust these processes and optimize them. Business process monitoring must provide information on bottlenecks in a process, identify where the process can be optimized (for example, by replacing manual tasks with automated processes), report on key performance indicators in the process, and in general provide insight in process improvements.

IBM *WebSphere Business Monitor* provides the ability to monitor business processes in real time, providing a visual display of business process status, including the following functions:

► Alerts and notifications to key users facilitate continuous improvement of your business processes.

► The customizable dashboard is implemented as WebSphere Portal pages that are visually intuitive, featuring scorecards, key performance indicators, and gauges.

► The dashboard supports multidimensional analyses and reports with embedded business intelligence.

► Customized analytic components monitor existing business processes, as specified by the business user.

► WebSphere Business Monitor puts you in control of reports filtering.

► The Adaptive Action Manager invokes selected real-time actions or sets of actions based on predefined rules and policies.

► WebSphere Business Monitor integrates sophisticated business analysis with business processes.

► Innovation and optimization are enhanced in key business processes.

WebSphere Business Monitor can be integrated with WebSphere Business Modeler Advanced for insightful business modeling, simulation, and analysis.

4.8 Summary

SOA addresses existing business challenges by introducing new concepts to the current way companies manage their business, and the technical infrastructure to support businesses to achieve their objectives.

SOA advocates a service-oriented approach and specifies the architecture that implements this concept.

But, the SOA model goes much further than that: it specifies architecture design elements, open standards-based technologies, and integration patterns.

SOA is all about a new way IT supports the business to be more successful. In order to get the most value out of SOA, corporations should review business strategies and map those to IT architecture. SOA has a huge impact on organizations because it requires strong governance to manage the new model, and because it leverages reusability of assets and higher integration among IT and business units, which again requires each business unit to have a mix of technical and business skills and the willingness to closely cooperate with other business units. SOA represents a shift in the way enterprises acquire and manage IT resources, moving from a departmental to an enterprise-wide model.

In that respect, the mainframe's original design orientation is strongly aligned with SOA concepts, where the main IT objective was to provide enterprise-wide services, sharing of resources, and reusability of assets to achieve economies of scale. Mainframe technology is ready to fully support the SOA specifications and is able to play a very important role in this new paradigm.

The mainframe enterprise computing model, virtualization technologies, the capability to run mixed workloads, and many enhanced Qualities of Services make System z and z/OS the ideal platform on which to implement a Service-Oriented Architecture.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 86. Note that some of the documents referenced here may be available in softcopy only.

- ► *Scaling for High Availability: WebSphere XD and WebSphere Application Server for z/OS,* REDP-3968, DiMarzio, Paul, Watson, Brent, Ryan, Patrick C. (2005)
- ► *Patterns: Implementing an SOA Using an Enterprise Service Bus*, SG24-6346, Keen, Martin, et al. (2004)
- ► *z/OS 1.6 Security Services Update,* SG24-6448, Rayns, Chris, et al. (2005)

Other publications

These publications are also relevant as further information sources:

- ► Bieberstein, Norbert, et al. (2006) *Service-Oriented Architecture (SOA) Compass - Business Value, Planning and Enterprise Roadmap*, IBM Press
- ► Cherbakov, L., et al. (2005) *Impact of service orientation at the business level*, IBM Systems Journal 44, No. 4, 653–667 (2005).
- ► Erl, Thomas (2005) *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall PTR
- ► Thompson, John L. (2002) *Strategic Management - Fourth Edition*, Thomson, London.
- ► Prahalad, C.K. and Hamel, G. (1990) *The core competence of the corporation*, Harvard Business Review, May/June.

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

The Value of IBM System z and z/OS in Service-Oriented Architecture

IBM®

Redpaper

SOA Quality of Service requirements

Strengths of System z and z/OS

SOA solutions currently available on z/OS

Without doubt, Service-Oriented Architecture (SOA) is one of the most important topics on the agenda of any IT person. SOA involves a new vision of how to design, develop, and manage applications, but also puts new requirements on the underlying infrastructure.

This Redpaper describes the infrastructure challenges that SOA brings to the table and how the IBM System z platform and the z/OS operating system address those challenges. An effective SOA implementation requires very high Quality of Services (QoS) from the underlying environment, and users demand security, availability, and simplified management of the services. These are fundamental characteristics of System z and z/OS, making them an ideal platform on which to design an SOA.

This paper presents an overview of SOA, describes the SOA reference architecture, and demonstrates how IBM System z and z/OS support the SOA requirements. Finally, it suggests an approach for SOA-enabling existing applications and provides several integration scenarios.