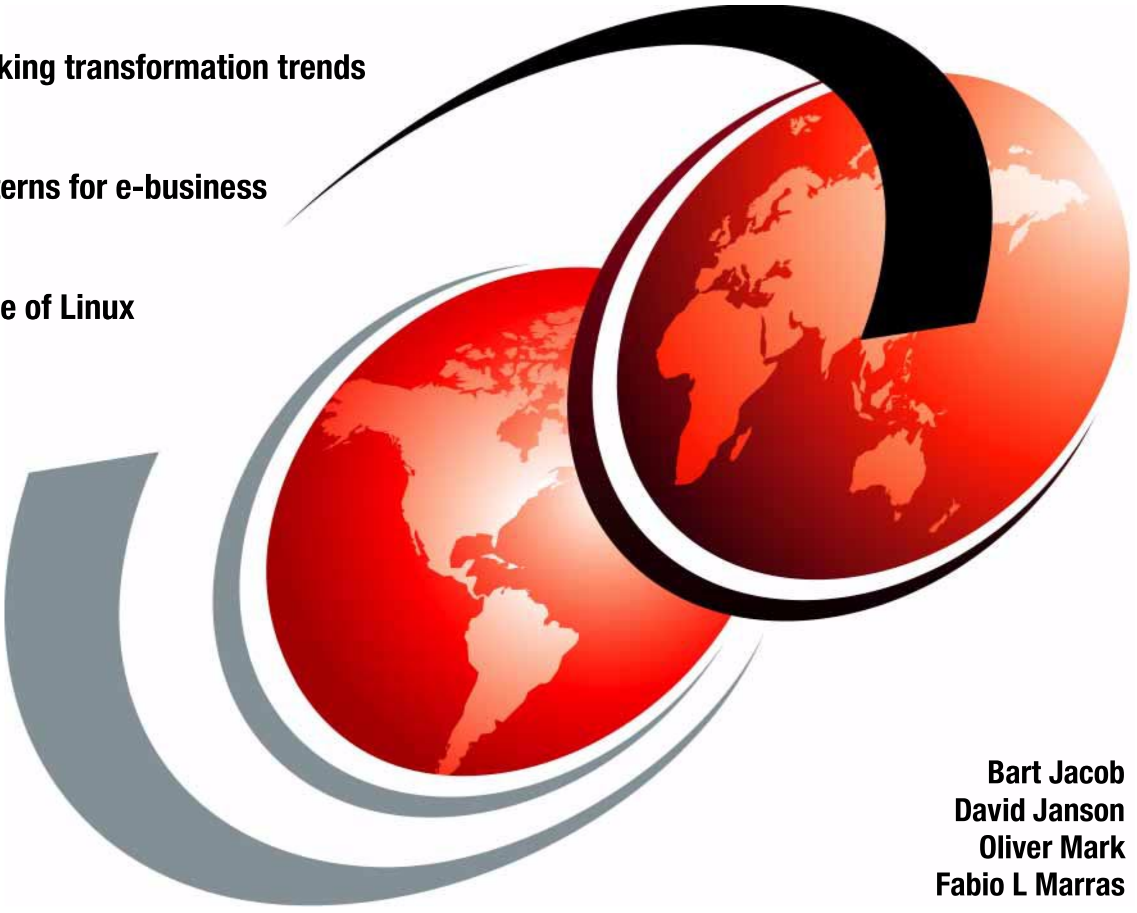
IBM

Linux and Branch Banking

Branch banking transformation trends

An IBM Patterns for e-business approach

The vital role of Linux



Bart Jacob
David Janson
Oliver Mark
Fabio L Marras

Redbooks

ibm.com/redbooks

International Technical Support Organization

Linux and Branch Banking

December 2002

SG24-6909-00

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (December 2002)

Contents

Figures

Notices

Trademarks

Preface

This IBM Redbook addresses two topics that may, at first, seem rather orthogonal. One of these topics is the world of banking, and specifically, the transformation of branch banking. We spend a great deal of time in this book describing the branch banking environment, the transformation that is currently taking place related to branch banking, and various architectures and techniques being used to define the branch bank of the future from an IT perspective.

The second topic is Linux. The phenomenal open source operating system, whose use within e-businesses as a secure and robust platform has skyrocketed since its inception.

This redbook discusses the applicability of Linux to the branch banking environment and why it should be given strong consideration as the basis for branch banking systems of the future.

Chapter 1, “An introduction to Linux” introduces Linux and discusses its applicability as a server and as a client.

Chapter 2, “Branch banking environment” begins our discussion of branch banking, including how it has evolved and why there is such an emphasis in the industry around its transformation.

Chapter 3, “Branch banking requirements” describes the key requirements of a branch banking solution from both a business and an IT perspective.

Chapter 4, “IBM Patterns for e-business overview” introduces IBM Patterns for e-business. These patterns have been successfully used to document the problems and possible solutions for companies becoming e-businesses.

Chapter 5, “Applying IBM Patterns for e-business to branch banking” applies Patterns for e-business to the branch banking environment. Though the solutions we describe are generic, this chapter can be used as a guide for utilizing patterns to define the solutions required for your environment.

Chapter 6, “Linux-based products applicable to branch banking” provides an overview of the facilities available for Linux that can make it a viable solution for branch environments.

Chapter 7, “Scenario for a new branch banking solution” provides a sample scenario showing a branch banking solution of the future.

The team that wrote this redbook

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

Bart Jacob is a Senior Technical Staff Member at IBM Corp - International Technical Support Organization, Austin Center. He has 21 years of experience providing technical support across a variety of IBM products and technologies, including communications, object-oriented software development, and systems management. He has 10 years of experience at the ITSO, where he has been writing IBM redbooks and creating and teaching workshops around the world on a variety of topics. He holds an MS degree in Numerical Analysis from Syracuse University.

David Janson is a certified Consulting IT Architect with the IBM Architecture and Technology Center of Excellence. With over 20 years of experience in the IT industry, the last 7 of which have been with IBM, Mr. Janson's areas of expertise include Enterprise Architecture, e-business Solution Architecture, Application Integration, and software development methods. His special interest in techniques that accelerate the process of architecting and implementing IT solutions has made IBM Patterns for e-business an area of particular focus. Mr. Janson is also an experienced instructor and has had the pleasure of teaching classes on a number of advanced topics related to software architecture and design. He holds Bachelor of Science degree from Southern Illinois University.

Oliver Mark is a certified Consulting IT Architect in IBM Global Services Germany. He has over 12 years of experience in the client/server area, mainly in the banking sector, with a clear focus on OS/2-based end-to-end architectures. He is responsible for all IBM Global Services service offerings for OS/2 and its customers in the EMEA Central Region. He holds a degree in economics and computer technologies. His areas of expertise include OS/2, Warp Server, IBM Communication products, Windows, and Linux, as well as Infrastructure and Systems Management architectures.

Fabio Marras is an IBM I/T Architect in Solutions Financial Services Sector in Brazil. He has 12 years of experience developing solutions for the finance industry. He holds a degree in Electronic Engineering, and his areas of expertise include networking hardware and software, as well as system integration across multiple platforms.

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

Jonathan Adams, IBM Software Group
Margie A. Bachman, IBM Global Financial Services Sector
Larry Baldauf, IBM Software Group
Arthur Cannon, IBM Global e-business Solution Center, Dallas
Jakob Carstensen, IBM Software Group
Jose Carlos Duarte Goncalves, IBM e-Business Technical Sales
Laerte H. Fagundes, IBM Financial Services Sector
Rod Fleming, IBM Global e-business Solution Center, Dallas
David H. Fritz, IBM Software Group
Henry C. Hudson, IBM Financial Services Sector
Catherine McCauley, IBM Global e-business Solution Center, Dallas
John Medicke, IBM Software Group
Thomas Pack, IBM Software Group
Michael Paolini, IBM Linux Integration Center
Bill Reed, IBM Software Group
Jason Salcido, IBM Global e-business Solution Center, Dallas
Timothy F. Sipples, IBM Software Group
Lori Small, IBM Software Group
Lloyd Stearns, IBM Software Group
Pablo E. Suarez, IBM Global Services
Barbara Trainor, IBM Software Group

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 Redbooks to be as helpful as possible. Send us your comments about this 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 Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 003 Internal Zip 2834
  11400 Burnet Road
  Austin, Texas 78758-3493

1

An introduction to Linux

Although we do not start our detailed description of the branch banking environment until Chapter 2, it is important to position Linux within a finance industry context to begin this book.

IBM expects Linux to become a significant platform option for our financial services customers. This is not just speculation. Banks, brokers, and insurance companies are using Linux already, and we expect many others to follow, albeit in generic and niche areas first, before tackling mission-critical transaction-oriented workloads.

With the recent stock market corrections and reductions in retail brokerage volumes adding to market competition, the already intense focus on cost is likely to be even higher, driving further investigation into efficiencies in e-business infrastructure and more cost-effective ways of working. This view is supported by the recent DataMonitor study on IT efficiency in the financial services sector, which found *cost control* to be the number one or two driver of IT strategies in 42% of our customers for 2002. We expect this to further drive Linux investments, both at an individual application level and in server consolidation scenarios.

Let’s start our discussion with a brief overview of Linux.

1.1 What is Linux

Linux is a UNIX-like, POSIX-compliant operating system distributed under the GNU software license, meaning that the operating system and its source code are available for free. Initially developed in the early 1990s by Linus Torvalds along with other programmers around the world, Linux supports all the major Window managers and all the Internet utilities, such as FTP, Telnet, and SLIP. It provides true 32-bit and 64-bit multitasking, virtual memory, shared libraries, and TCP/IP networking. It also has a full set of program development utilities, including C++ compilers and debuggers.

Linux consists of four major components: the kernel, the shell, the file structure, and the utilities.

Although it has been implemented across a variety of hardware platforms, it is most often deployed on Intel-based processors. The Linux kernel is designed to use the special protected-mode features of Intel 80x86 processors. Therefore, Linux makes use of the protected-mode, descriptor-based memory management paradigm and many of the other advanced features of these processors. Linux also exploits the multitasking capability of 80386 protected-mode programming. The Linux kernel supports demand-paged loaded executable so that only those segments of a program that are actually used are read into memory from disk. Likewise, if multiple instances of a program are running at once, only one copy of the program code will be in memory.

The Linux kernel includes many improvements specifically designed for enterprise. The kernel has built-in support for eight-way symmetric multiprocessing, and the TCP/IP stack has been rewritten to increase stack performance and provide better scalability in multiprocessing systems.

The Linux kernel is a monolithic kernel, in that all device drivers are part of the kernel proper. However, Linux does support loadable device drivers that can be loaded and unloaded from memory through user commands. The kernel is capable of emulating 387-FPU instructions itself so that systems without a math coprocessor can run programs that require floating-point math instructions.

Linux users interact with the Linux kernel through their own individual shells. The shell provides an interface between the kernel and the user. Shells provide a command line interface to the Linux kernel; if the user prefers a GUI interface, XFree86, can be installed. XFree86 is a freely available distribution of the X Window System, containing the X server, programming libraries for developing X applications, and documentation. Window managers are also available. The window manager provides the graphical interface that lets users send commands to the Linux kernel by clicking windows, icons, and menus, much as they would on a Microsoft Windows or Mac machine.

1.1.1 Standards

Linux is a Posix-compliant OS, and its constituent subsystems support all relevant ANSI, ISO, IETF, and W3C standards. From the beginning, Linux was developed according to the Posix standard. The standard defines how a UNIX-like system needs to operate, specifying details such as system calls and interfaces. Posix defines a universal standard to which all UNIX versions must adhere. Most of the popular versions of UNIX are Posix compliant. Linux Posix compliance has made it possible for developers to port many popular UNIX applications and utilities to Linux.

Linux provides a complete implementation of TCP/IP networking. The complete range of TCP/IP clients and services are supported. The Linux TCP/IP system provides a standard socket programming interface so that any program that uses TCP/IP can be ported to Linux.

Linux also supports the standard ISO-9660 file system for CD-ROMs. The Linux printing software consists of the UNIX standard *lp* and *lpr* software. Any parallel printer supported by MS-DOS or another operating system should be supported by Linux. Linux supports the full range of serial modems, both internal and external.

In short, the range of interfaces and capabilities that have become standard across virtually all operating environments are also available for Linux and implemented through open source code and based on industry standards. This combination provides a stable and predictable environment for developing, porting, or running business-critical applications.

1.1.2 Minimum operating requirements

According to the Linux specification for Intel-based platforms, the minimum PC requirements for Linux are an Intel 386 computer with about 4 MB of RAM. But to run the graphical interface and to have enough speed and space to run applications, most commercial distributions of Linux now recommend at least a 486 (preferably, a Pentium) machine with 64 MB of RAM and a 600 MB hard drive. Linux requires a minimum of two disk partitions: swap and system. The base Linux root file system uses between 50 MB and 100 MB of disk space. Swap space should be equal to RAM memory, with an absolute minimum of 16 MB recommended. However, the size of the Linux system partition will be larger, based on what the Linux system is used for and the scalability and portability of file systems to other disk partitions.

Although the actual hardware requirements will be dependent on the business and application requirements, it is safe to say that the design and efficiency of the Linux environment allows companies to build robust systems using relatively generic and less expensive hardware.

In summary, Linux, as a full-function, standards-based operating system with relatively minimal hardware requirements, has been getting a lot of attention from businesses looking for less expensive solutions to meet their IT demands.

Of course, Linux distributions are also available for other hardware platforms, including the full range of IBM servers from xSeries to zSeries. For the minimum requirements on platforms other than Intel, please see the relevant Linux distribution documentation.

1.2 IBM and Linux

For IBM, this is all about customer choice and application enabling. IBM is involved in all stages of the Linux phenomenon: contributing code and skills to Linux development itself; delivering Linux-ready hardware platforms in conjunction with established Linux distributors; porting IBM's industry-leading middleware to Linux; providing open-access Linux developer resources; helping customers and business partners implement Linux solutions; and providing support through IBM Multivendor Software Supportline contracts.

1.2.1 Linux as a server

Every IBM Server platform has Linux as an option, either as an alternative to the native operating system, or as a co-resident application environment. IBM provides a wide range of choices for Linux application deployment in the industry. However, this does not mean that IBM expects all applications to be Linux-based and, indeed, new offerings, such as the 64-bit z/OS successor to OS/390, demonstrate that IBM will continue to invest in other environments that support and enable our customers' critical applications.

For customers who just want Linux, the Intel-based xSeries provides native support. The xSeries now also addresses the server consolidation market, as it is possible to partition the x440 system in up to four hardware virtual machines and up to 64 software virtual machines.

The IBM pSeries server provides enhanced capability to execute Linux applications using the Linux operating environment in native mode or in a logical partition (LPAR). IBM AS/400 customers who want to implement Linux capability also can take advantage of the Linux in LPAR support in the iSeries.

Meanwhile, Linux for zSeries extends the capability of Linux for S/390 into the 64-bit world and retains the unique ability to provide massively parallel implementations of thousands of "virtual Linux servers" all within a single enterprise server. This unique capability has been behind a number of significant customer projects among IBM's Linux customer references.

To complement this platform capability, IBM has extended its already broad portfolio of middleware offerings to include Linux on the server. IBM WebSphere, Lotus Domino, DB2 Universal Database, and Tivoli all provide Linux support today as a natural extension of their multiplatform strategies and regard Linux as a *Tier 1* platform for future developments.

1.2.2 Linux as a client

There are still a large number of OS/2-based clients in use by IBM financial services customers, and, therefore, significant interest in whether IBM considers migration to Linux to be strategically viable. For the general-purpose, full-function ("heavy") client environment, the market assessment today continues to be "not yet," although the increasing focus on cost containment in all areas is prompting increased interest in Linux in this context as companies and governments look forward.

However, for specialized environments, such as development desktops and thin-client solutions, particularly in branch environments, Linux may be appropriate and provide an extremely cost-effective alternative. In addition to Linux itself, there are mainstream offerings available, such as the Netscape (Mozilla) browser, the OpenOffice suite (now available with support and documentation for a fee as StarOffice), Adobe Acrobat Reader, and so forth.

IBM has also recently announced the WebSphere Studio Application Developer for Linux and our involvement in the Open Source Eclipse program to enhance the Linux development environment in the future. More information about the Eclipse project can be found at:

http://www.eclipse.org/

At a hardware level, IBM provides a number of Linux-ready ThinkPads, and there are device drivers available for many common components. However, many customer environments include specific devices and programs for which support is not yet available for Linux, and so IBM is working directly with the OS/2 Customer Advisory Council, and individual OS/2 customers within the context of our branch transformation services, to investigate specific migration scenarios and possibilities. As a contributor to these migration scenarios, the Java eXtensions for Financial Services (J/XFS) code has been tested on Linux and can thus provide financial device access to Java applications running in a Linux environment, just as J/XFS does today on OS/2.

1.3 Linux in a business environment

In this section, we describe some of the roles Linux plays in businesses today and address some of the benefits, as well as issues, related to Linux.

1.3.1 Business use

Within corporations, Linux has made significant inroads within the IT infrastructure. Companies have reported considerable success deploying Linux as front-end application servers and in numerous infrastructure environments, such as caching, VPN networking, DNS, and proxy, as well as Web server environments. Linux is being implemented in numerous replicated deployments, such as in-store controllers, kiosks, rental and reservation systems, and thin-client access to databases. Other applications are in technical and scientific computing clusters, digital entertainment, and special effects.

At first, Linux was used to turn older systems with minimum resources into inexpensive Web servers, domain name servers, or e-mail servers. As such, Linux has become a popular development and production platform for many Internet service providers and Web-hosting service providers. Linux brings down the cost of traditionally high-end business computing to the reach of many smaller organizations. Because of its lower initial installation costs, as well as its reputation as a stable operating environment, Linux has been used extensively for implementing Web server farms and other installations requiring a large number of server machines where ease and cost of deployment, as well as availability, are important.

With the porting of IBM middleware, such as DB2, WebSphere, Lotus Domino, and Tivoli management products, to Linux, many businesses are starting to deploy Linux-based systems as application servers, as well as Web servers.

1.3.2 Benefits and risks

Benefits and risks are one of the most important topics to be analyzed when developing a new solution or considering a change to your current environment. In the next sections, we discuss some of the benefits and risks related to Linux.

Open source development

Because developers have access to the Linux source, changes can be made to operating system code to meet the needs of individual applications. With a proprietary operating system, users are dependent on the vendor to make any needed change. If not enough customers request the change to make it profitable, the vendor may refuse to do it. If the vendor agrees to make the change, they may charge for the change or decide not to support it later, or both.

On the other hand, many enterprise users would rather leave changes up to a vendor than make the change themselves and have to share their code with the Linux open source community. The most oft-cited reason given by larger companies with 2000+ employees for not installing Linux is that the proprietary nature of the software their companies depends on precludes them from open source development.

Vendor accountability

Larger organizations tend to prefer proprietary products to open source software because they rely on vendor accountability; they require a vendor that can provide them with complete support coverage, from the system kernel to system configuration. Naturally, they are apprehensive about introducing Linux, whose main support and accountability comes from the open source community. While these companies could look to Linux distribution vendors, as well as to those that market Linux on their machines such as IBM, for support, many argue that the business models these vendors have built around Linux are still untested and may not withstand the long haul. However, others point out that there is no guarantee that companies that build their business models around proprietary operating systems will not fail. What is guaranteed is that no matter what the fate of the open source vendors, users still own the Linux source and do not have to depend on any one company to upgrade it for them. They can do it themselves as part of the open source community.

Management costs

Most IT departments would contend that it does not make sense to use a system management tool that would cost them more than the systems to be managed. That is why many IT managers are turning to open source tools to manage Linux servers. These tools, such as NetSaint and Webmin, are available for free. Managers are also leveraging their UNIX expertise to write their own administration and monitoring scripts.

However, where Linux coexists on a network with other operating system platforms, managing Linux servers separately from other enterprise resources and using different tool kits to do so would actually raise the total cost of owning Linux. In these situations, users should look to their management solutions vendors, such as IBM Tivoli software, to provide the necessary tools.

Reliability, availability, scalability

While the Linux kernel version 2.4 will handle eight-way Intel systems, Linux multiprocessing capabilities are considerably less than that of high-end UNIX and Windows 2000 servers. Therefore, Linux is not always the best choice for high-end applications, such as database installations. However, many users may find that Linux clustering will satisfy their availability requirements at an affordable cost. Using commodity hardware, users can spread the workload across many smaller, less expensive systems.

Companies that have adopted Linux clusters typically have superior in-house skills, with control over their own applications, and are more comfortable with the risks involved in building their own clusters than they are with having to wait for third-party applications to be developed. For those without such skills, IBM offers the Linux IBM @server Cluster 1350 line of pre-assembled server clusters. The Linux IBM @server Cluster 1350 line also includes bundled applications and server-level software, plus integration testing and technical support services. The clusters can be set up with a minimum of four Intel-based servers, which can be expanded to more than a thousand servers. The servers come installed and preconfigured with full systems administration, maintenance, and workload management software, along with server-level software, such as DB2 or IBM WebSphere Application Server and e-business software.

Availability of applications

Linux provides a complete UNIX programming environment, including all of the standard libraries, programming tools, compilers, and debuggers that users have come to expect from comparable UNIX systems. Nevertheless, porting applications to the Linux platform by the major software vendors has been relatively slow, especially in terms of databases, middleware, and legacy Windows applications. With investments in Linux by major computer companies, such as IBM, Oracle, and SAP, this situation is changing quickly. Major applications, such as IBM DB2, IBM WebSphere, Oracle 8i and 9i, and SAP mySAP.com, have been certified to run on one or more Linux platforms.

Application compatibilities among distributions

Even though more applications are being ported to Linux, there is no guarantee that they will run on all Linux distributions. This is because each distribution can have a different version of the same library, making it difficult for developers to know which one to use when porting their application from one Linux platform to another. Applications compiled to run with one version will probably not run under a distribution that uses a different version. What is more, an application that requires one version of a library can overwrite an existing library with a newer version that may be incompatible with existing applications that rely on the former library. Java-based applications provide one possible solution.

Costs of migrating to Linux from current environment

Organizations that have made considerable investments in Microsoft Windows will have to weigh the costs of implementing Linux against any savings in licensing costs. These costs include not only the costs of skills and tools, but also the time it takes to acquire them. While UNIX shops can use the tools and skills they already have to deploy Linux, shops that have made a significant investment in Windows hardware and software and developing their people's skills to support their investment, such as Visual Basic development or system administrators with MCSE certification, may find the conversion costs difficult to justify. However, Linux is already popular in colleges and universities, and most of new graduates have Linux experience.

1.4 Summary

Linux has been around for nearly a decade, and in recent years, it has captured the interest and recognition of businesses as a robust and viable operating environment for many applications.

This chapter has provided a high-level introduction to Linux, discussed some of the ways businesses are utilizing Linux, and highlighted some of the considerations related to Linux.

In the next chapter, we discuss the current state of branch banking and some of the drivers that are causing a branch bank transformation. Because the IT infrastructure to support branch banking may be changing significantly to support this transformation, Linux should be seriously considered as a cost-effective base for these new environments.

2

Branch banking environment

This chapter provides an overview of the current state and structure of branch banking environments. It discusses the business pressures and challenges currently facing branch banking environments. We discuss the latest trends for approaching those issues, both from a business and a technical perspective. We also address the software strategy at the application level to solve the branch banking problems in a broader context of a multichannel solution.

We also cover the organizational structures of such branch bank environments, because any solution that is put in place must meet the functional needs of the individuals that operate and manage the bank branch.

2.1 The state of branch banking

“The retail bank branch is not dead. On the contrary, there are more branches today and more households per branch than ever before”, as stated in the TowerGroup article “The Retail Bank Branch of the Future” published in August 2001. In this same article, the TowerGroup concluded, “Bank customers still prefer to use the physical branch to many automated channels and that, in fact, 92% of all households surveyed had visited a branch in the month prior to being surveyed.”

“Irrespective of the fact that the majority of U.S. consumers are now ready for online banking in technological terms, behaviorally they continue to be slow to migrate away from traditional channels like the local bank branch or call centers,” said Michael Weil, TowerGroup’s managing director of Primary Market Research. “Rather than dropping existing channels, consumers use what they know while slowly adding new delivery channels into the mix. The move to online banking-and ultimately to wireless as a delivery channel for financial services-will continue to evolve.”

So, what we usually see is a single-channel branch accessing data on the central system in parallel to all other channels, as seen in Figure 2-1.

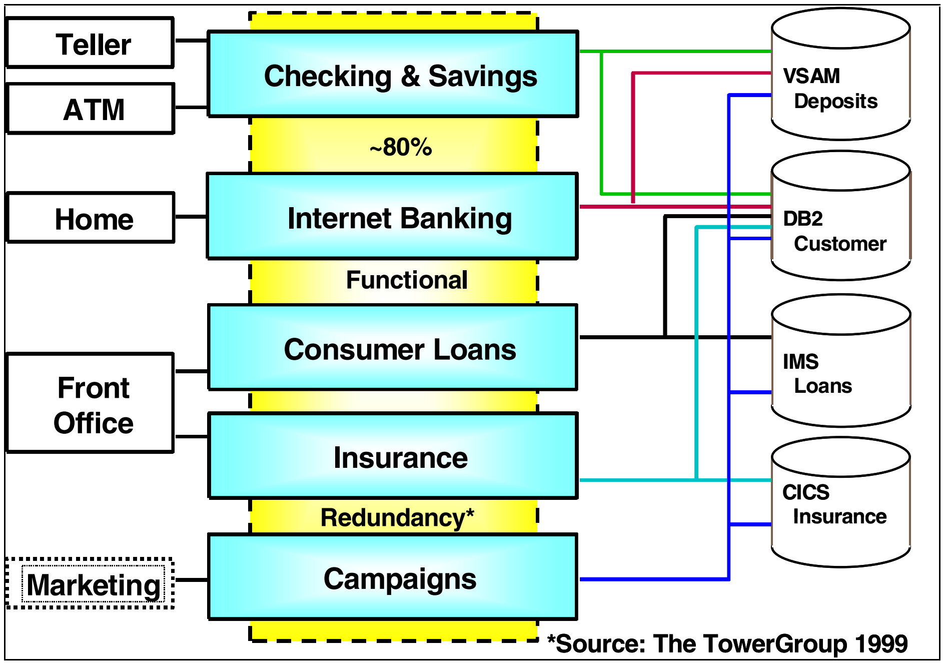


Figure 2-1 Current architecture with isolated channels

2.1.1 Branch banking trends

The banking industry has undergone significant changes in the past several years regarding delivery of retail products and services. A February 2002 article from TowerGroup, “Retail Bank Branch Renewal in the United States: The Market, Technology Directions, and Vendors”, explained “There has been relatively little strategic investment in the retail branch bank infrastructure over the last ten years.” They give four primary reasons for this:

- Emphasis on the continual emergence of alternative delivery channels (Internet, cell phones, PDAs, and so on).
- Existing technology, such as OS/2, which has been the predominant operating system in branch banks for a number of years, has been reliable, stable, and has had a low total cost of ownership. Also, 3270 terminals and 4700 controllers have provided reliable connections to the mainframe.
- There has been little need for advanced technology in the branch, while banks were focused on alternative channels.
- The pain factor involved in changing existing technologies includes the cost of new hardware and software, along with the training required to introduce new interfaces and business processes into the branch. Bank executives have been hesitant to change without a clear ROI for new investments.

However, a number of factors have forced banks to rethink the strategic roles of their branch channel and to consider a major renewal of their existing technologies. The TowerGroup predicts, “In terms of absolute value, the biggest area of global spending over the next five years will be in bank branch renewal.” Reasons for this include:

- The banks’ hope that the emerging call center and Internet channels would lead to the elimination of a significant number of branches has not come to pass. Most have realized that the branch channel will continue to be a significant piece of a multichannel solution. However, due to the slowdown in branch spending and the addition of other channels with a variety of architectures, the IT infrastructure of many banks is neither cost-effective nor suited to the demands of multichannel delivery. As stated in a recent article in *The Banker* magazine, “Integrating internal systems and multiple channels should be at the top of the list of the IT agenda for banks.”
- According to TowerGroup, “The second driver of branch renewal is the ultimate demise of OS/2 as the underlying operating system for branch automation solutions worldwide. Support of OS/2 and the hardware it resides on will become increasingly expensive and difficult to acquire.” Other obsolete technologies, such as 3270 emulation, IBM 4700 controllers, Windows 95/98, and in some cases even DOS or Windows 3.1, face similar issues.

Along with these obsolete technologies, the need for new multichannel services, and the lack of investment and technological readiness combine to make the branch bank the largest channel for investment over the next years.

Also, banks are seeking to derive greater value from the retail branch delivery system through transformation activities. These activities include increasing branch effectiveness, creating financial centers capable of developing and maintaining long-term relationships, and developing flexible branch configurations that can meet the special needs of particular customer segments. Today's retail branch delivery system infrastructures lack the required flexibility and are, at the same time, costly to maintain and operationally intensive.

2.2 Branch technology challenges

In order to deliver the technology solutions that help achieve the business goals of branch transformation, bank IT providers are faced with three key challenges:

- Cost
- Quality of service
- Speed to market

2.2.1 Total cost of ownership

Because cost is a key element of the value equation, banks are concerned that branch transformation solutions can be deployed at an affordable cost. There are two major areas of cost involved in branch transformation: solution deployment costs and ongoing operating costs.

Solution deployment costs include the costs of new software (operating systems, middleware, and applications), hardware (clients, servers, and networks), and people costs (both internal and external) for integration, coding, installation, and education. Costs can be mitigated by reuse of the existing hardware and software infrastructure, minimizing new hardware requirements, and by keeping user interfaces and processes consistent to reduce training requirements.

Ongoing ownership costs include systems management, service and support, upgrade costs for hardware and software, downtime due to defects, user errors, service interruptions, security problems, such as viruses or denial of service attacks, and any resources and downtime involved in deployment of new tools or services. The key to reducing ongoing ownership costs is a manageable infrastructure that reduces people costs and has the flexibility to offer a wide range of choices and options for deployment of new or upgraded solutions.

2.2.2 Quality of Service

Quality of Service (QoS) is important both in reducing costs and in providing high-quality customer service, which can be a competitive differentiator. Issues include security, availability, reliability, performance, and scalability. Security is of increasing concern due to new and more virulent viruses, hacking, and denial of service attacks. Availability, reliability, and performance all contribute to high-quality customer service, and scalability allows the bank to provide increased services without increasing costs or degrading performance. All of these are best achieved by an infrastructure that is flexible enough to allow the customer to make the right choices and trade-offs to optimize each of these Quality of Service parameters based on their particular needs.

2.2.3 Speed to market

In today's competitive environment, banks are looking for a solution that enables them to quickly respond to changing market demands, whether it is by reconfiguring their branches around new delivery models, or rapidly deploying applications to support new revenue-generating products and services. A non-proprietary, open architecture that is flexible and that allows centralized management and deployment of applications is essential to the rapid delivery of new business functionality. An e-business architecture that minimizes the dependency on distribution of code to client systems can provide an optimal solution.

2.3 Branch transformation strategies

Branch banking transformation can be defined as the updating, replacement or re-engineering, or both, of computer systems that support both staff and customers in the branch offices of retail banks.

But, what are banks hoping to accomplish through their investments in the branch channel? Because branches do not seem to be going away, banks are seeking to derive more value from their branches through a combination of initiatives designed to increase the revenue produced while decreasing the cost of delivering services through this expensive, yet necessary, channel.

In order to drive additional revenue from their customers, banks know that they must improve their understanding of their customers' needs and offer them the right products and services at the right price. They also know that they must focus on customer satisfaction and improved service to retain and grow relationships with their most profitable customers.

To do this, banks are focused on a number of different initiatives:

- Transforming the role of the branch from that of a transaction-oriented center to a relationship-oriented center capable of providing a broad range of products and services and focused on developing and maintaining long-term relationships.
- Partnering with local businesses to offer innovative products and services from external partners.
- Creating relationship teams to focus on the complete financial needs and issues of high-value customers, with the branch representative becoming a “relationship manager” and through collaboration, leveraging the collective knowledge and skills of the institution to meet the customers’ needs.
- Developing flexible branch configurations based on the needs and value of specific customer segments.
- Joining their branch and customer relationship management (CRM) initiatives to deliver better customer information to the branch and improve cross-selling at every touch point.

While driving additional revenue is one part of the equation, banks are still acutely aware of the need to reduce branch costs. Because personnel costs account for a significant portion of branch expense, initiatives aimed at improving productivity and effectiveness are key to cost reduction objectives. These can include:

- Re-engineering branch processes, such as teller, sales platform, and lending, or introducing new technologies, such as check capture, check copy, and optimized self-service deposit taking, in order to streamline processes and improve productivity.
- Drive more transactions to lower cost channels and automate low-value, high-volume transactions through information kiosks, ATMs, and teller assist units.
- Empower staff with guided sales tools to decrease the dependency on experts to deliver product and service information.
- Use external suppliers to out source costly operational processes.
- Offer remote access to specialists who can provide sophisticated financial planning and specialized product knowledge in order to mitigate the need for specialists in every branch.

- Improve employee productivity by delivering enterprise information and services online, including broadcast messages, announcements and alerts, HR services and information (intranet), e-Learning, e-mail, collaborative tools, static information (procedures manual, product information, forms, branch/ATM locator), bank directory (phone numbers, people, departments), and dynamic forms.

These and other strategies are placing pressure on the bank IT providers to deliver IT solutions that can not only deliver these capabilities, but also provide a flexible and scalable infrastructure that can quickly support whatever future business initiatives the lines of business may undertake.

2.3.1 Branch software strategy overview

This section introduces some of the basic principles and assumptions for a branch transformation software strategy and why they are important.

Build it to support multichannel

We believe that as banks begin to look at how they will transform their branch delivery channels, many of them will seek to do so in the context of a larger, multichannel delivery strategy, a strategy that enables them to integrate all of their delivery channels and deliver all of their products and services in a consistent manner, regardless of the channel the customer chooses to use.

Why is this important? Because statistics tell us customers are multiple channel users. Therefore, a bank must endeavor to optimize each and every interaction with the customer, regardless of the channel, both to the advantage of the customer and to the bank itself. Such treatment is key not only to improving customer service and satisfaction, but also to solidifying long-term and profitable relationships.

By looking at their branches as part of an overall delivery strategy, banks will be able to tailor branch services to meet the needs of the customers who use the branch, as well as develop a channel strategy that matches service levels to customer value, and set the appropriate goals for branch transformation.

With this in mind, the software strategy for branch transformation is built on the overall software strategy for a multichannel delivery system, with specific emphasis on providing solutions to unique issues related to the branch location and the channels it supports. On the other hand, although they may be thinking strategically, cost pressures and other immediate demands will cause banks to act tactically. The software strategy (in fact, the entire infrastructure strategy) must accommodate this and offer a value proposition for those who choose a more tactical approach to branch transformation.

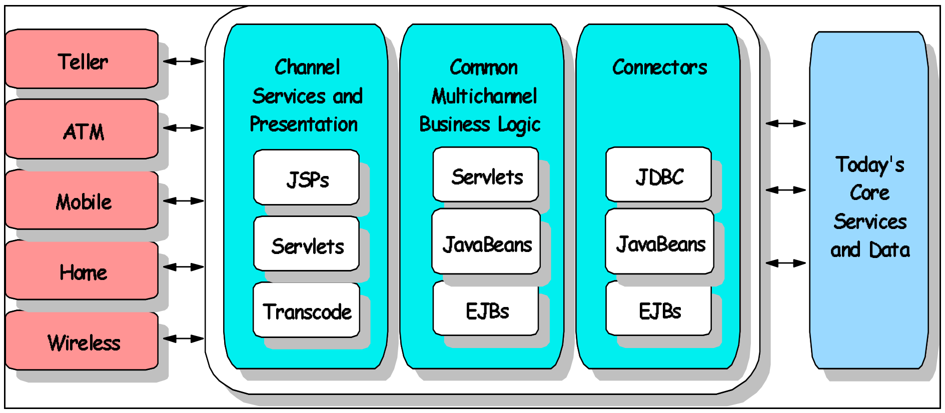

Teller
ATM
Mobile
Home
Wireless
Channel Services and Presentation
JSPs
Servlets
Transcode
Common Multichannel Business Logic
Servlets
JavaBeans
EJBs
Connectors
JDBC
JavaBeans
EJBs
Today's Core Services and Data

Figure 2-2 Built for multichannel

We discuss multichannel considerations in more detail in 2.3.2, “Multichannel context” on page 21.

Build it based on e-business technologies

The nature of the banking business is that the universal tool for the execution of nearly all business processes, marketing, sales, development, production, and distribution of financial products and services, is the information system. No other industry is better suited for e-business than the finance industry.

Most financial institutions first encountered e-business technology when they began to use the Internet to provide electronic home and office banking facilities.

As the second phase of home banking began to take hold in the mid to late 1990s, it quickly became clear that a “fat client” home banking application would present a huge maintenance and support challenge to banks. Banks that had any experience at all managing a large network of PC-based clients deployed throughout their branches were uncomfortable with the prospect of distributing software to millions of client PCs of unknown origin and providing technical support to millions of customers with limited technical skill. Luckily, the World Wide Web began to achieve commercial acceptance, and Internet technologies became popularly available to home users, making it possible to deliver home banking services through a Web browser. Although this did not totally mitigate the support issues, it helped make home banking a viable and cost-effective delivery channel.

When we look at branch networks today, many of the same issues that drove banks to choose the Internet to deliver banking services to home users still exist in the branch network: it is a large, distributed network with users of limited technical skill. Internet technologies have come a long way and have achieved a level of maturity and robustness that make them not just the solution for self-service home banking, but also ideal as the technology infrastructure for the bank’s branch network and, therefore, for the entire multichannel delivery infrastructure.

Moreover, leading banks recognize that a flexible, open, standards-based infrastructure will be essential to helping them retain their competitive edge in a marketplace that increasingly demands innovative business model deployment and rapid response to changing market dynamics.

It is focused on the unique needs of the branch location. Banks have a long history of managing large, distributed technology networks and are keenly aware of the costs associated with managing such a network and supporting a large community of relatively non-technical users. While they value flexibility and openness in their infrastructure, banks are also acutely focused on the need for a secure and stable operational environment underpinning their most visible and highly used channel. So, the challenge is twofold: providing a flexible infrastructure that will enable them to react quickly to change, while delivering a stable, reliable, and manageable operational environment at a predictable cost of ownership.

The adoption of e-business technologies, with server-based applications and thin clients (browser-based), helps to address some of the issues of software distribution, as well as simplifies the support required by internal help desk personnel. In addition, with increased bandwidth capabilities, it is now possible to centralize servers and still deliver satisfactory performance and reliability to the branches. Centralized servers communicating with thin clients can significantly reduce the management burden that results in the high cost of delivering technology to the branch.

On the other hand, we are well aware that centralization of servers and totally thin clients may not be appropriate for all banks. Some banks will not be able to acquire reliable bandwidth in order to sustain the performance and availability they require. While desirable, thin clients may not be possible in the near term as a result of their dependence on existing client/server applications and productivity tools, such as the Microsoft Office, Lotus SmartSuite, and Lotus Notes applications.

Furthermore, the advent and popularization of Internet technologies has propelled security into a new dimension. Older, more proprietary technologies deployed in the branch were relatively immune to some of today’s security risks. But as they deploy open technologies, such as TCP/IP, and widely available, yet vulnerable, operating systems, such as Windows, banks are concerned about the heightened security threat.

All of this is intended to create a stable and secure operational environment that is flexible enough to address the unique needs of each customer and, at the same time, deliver a manageable cost of ownership.

It must support the needs of the builders and the buyers

There are two different categories of clients, and the software strategy must address each one’s requirements:

- Builders

  Those who prefer to build their own applications. Banks that prefer to build typically do so because they do not feel an independent software vendor (ISV) can satisfy their needs or because they consider branch delivery a core competence that they prefer to maintain in-house.

- Buyers

  Those who prefer to buy their applications from best-of-breed ISVs.

Infrastructure, middleware, and tools to support the creation of a multichannel delivery system for banking or to support a specific channel implementation must be considered during the software strategy definition, and it must be based on “de facto” standards to avoid support and continuance problems.

A compelling environment, that is, comprehensive, integrated, and easy-to-use tooling, is of paramount importance to the builder group in order for them to develop these applications quickly and maintain a speed-to-market advantage. Another factor that influences the tooling requirements is that the depth of e-business technology skills in banks is quite variable: some banks have significant depth of skills, while others have only basic capabilities, if any at all. The tooling capabilities, therefore, must cater to the needs of both ends of the spectrum: the power users and the novices.

While tooling is a critical factor to the builders, what about buyers? We believe that tooling is important to them as well, although in an indirect way. In order for the buyers to fulfill their needs with software providers or ISVs, there must be applications to buy. This means that ISVs need tools in order to produce the banking applications that buyers want to buy. ISVs typically have a greater depth of skill in the platforms on which they choose to implement, so power tools are probably more important, but the basic requirements are the same: comprehensive, integrated, and easy-to-use. One significant difference between ISVs and builders is that ISVs typically want to maintain a degree of vendor independence. Therefore, they will be inclined to prefer tools that are open and do not tie them to a specific platform vendor.

2.3.2 Multichannel context

Financial institutions are launching channel integration efforts to improve service, lower costs, and gain a comprehensive view of the customer. But, building a multichannel delivery system is hard work and not inexpensive, particularly for the first channel to be implemented. Like many infrastructure investments, it is difficult to justify the return based on a single channel or application implementation.

Instead, the ROI must be looked at in terms of the long-term flexibility and cost savings delivered and the benefits accrued as a result of reduced complexity and improved speed to market.

The cornerstone of this is an architecture that encourages enterprise-wide reuse of applications and infrastructure components. Figure 2-3 on page 22 depicts what we believe to be the framework or logical architecture of a multichannel delivery system for banking. This is consistent with our customer-centric architecture and the multichannel logical architecture described by the TowerGroup.

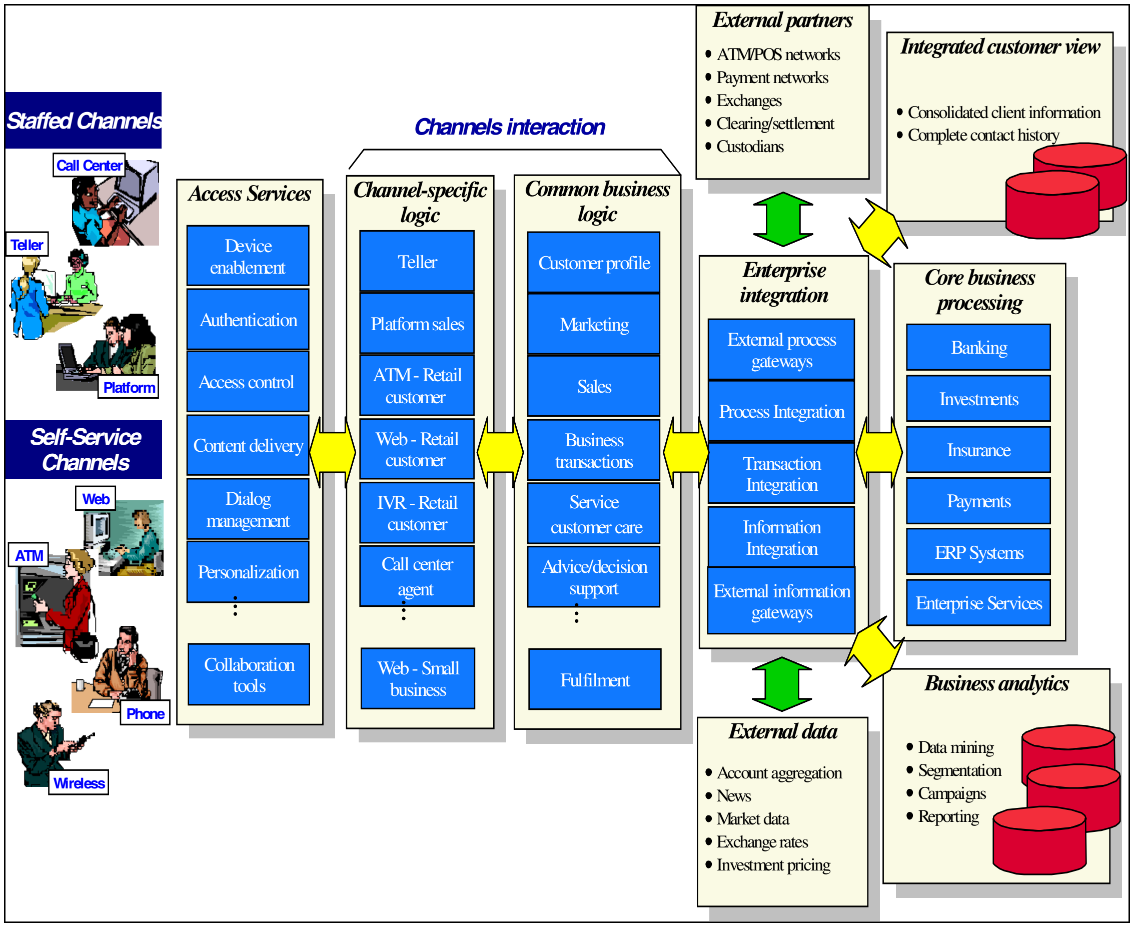


Figure 2-3 Multichannel environment

Integrated customer view

No multichannel solution will be successful unless the infrastructure successfully brings together the disparate data about a customer and delivers a single, consistent view of all that is known about the customer to the applications and users who serve the customer.

This integration of customer data can be physical, aggregating customer data physically by replicating it from existing systems, or logical, aggregating customer data logically, in that it knows where the data is and provides the services to get it, or a combination of both.

Core business processing

These applications, typically known as the back-office or legacy applications, are the systems of record that manage customer accounts, process transactions, and provide enterprise-wide services. The front-office applications (enabled through the multichannel infrastructure) are required to interact with these applications for access to customer financial data and transaction processing capability.

These systems are typically quite fragmented, leading to the fragmentation of customer information and business rules that, in turn, have created some of the channel dissonance in many of today's banking delivery systems. The multichannel infrastructure must continue to leverage the capabilities of these systems through integration, but at the same time, mitigate the fragmentation in service they have created by aggregating customer information and providing a front end with business logic and rules that disguise their inconsistencies.

Business analytics

One of the reasons for building a multichannel delivery system is to be able to optimize each and every interaction with the customer, regardless of the channel through which the interaction occurs. A key to maximizing the effectiveness of a multichannel delivery system in this regard is a set of business analytic capabilities that help analyze customer information, identify customer needs, match products and services to identified needs, and, finally, measure the effectiveness of interactions with the customer.

As Figure 2-3 on page 22 shows, business analytics includes the following types of capabilities:

- Data warehouses and the technologies required to populate warehouses
- Data mining tools to assist in mining the data to determine customer segments and the characteristics that drive customer needs
- Campaign management systems that enable marketing organizations to create, deploy, and monitor the results of campaigns that target products and services to meet the needs of specific customer segments
- Reporting tools that provide feedback to management about various system and customer-related performance indicators

Using a common set of business analytics helps to ensure that the treatment customers receive across all channels is consistent with respect to their needs and the services they are offered and the results reflect the totality of the customer's interaction with the institution.

Enterprise integration

With the growth in the number of channels and demand to deliver more and more functionality through these channels, point-to-point integration with back-office applications and external service providers has proven cumbersome and costly. In order to reduce the complexity of application integration, institutions are looking to implement a common integration framework that enables them to implement integration among applications in an efficient and common way. Tremendous efficiencies in development and testing can be gained, leading to lower cost and increased speed of deployment, by implementing the interfaces required to integrate these applications once on a common integration platform.

As we can see in Figure 2-3 on page 22, there are a number of different integration disciplines required to support the needs of a multichannel delivery system, including:

- Data integration
- Process integration
- Message integration

In addition, the enterprise integration framework needs to support integration with applications and services both within and outside the institution.

2.4 Branch structure

When analyzing the requirements for a new or transformed branch banking system, we first need to understand the structure of a branch. This structure includes the current systems and system types that are in place to support the branch personnel and business objectives.

The current make up of branch automation systems includes devices specific to the tellers and platform personnel, and in some cases, ATMs. How these systems are integrated and used by the bank employees are part of the operational model that is discussed in a later section.

For now, let’s discuss the general system types typically found in bank branches.

2.4.1 Branch systems

Workstations provide an end-user interface to the branch banking system for users, and servers provide connectivity, data access, or other system management functions to the workstations. They are all connected through a physical network, where one or multiple protocols are responsible for fast, stable, and reliable communication.

Teller and platform systems

The goal of branch automation systems is to enable the teller and platform personnel to process customer transactions, access customer account information, and generate reports.

Central systems store customer information, transaction, and account data, and traditionally, branch systems access this information.

Because branch automation systems use component technologies, these systems can make the same information available not only to tellers and platforms but also to other touch points, like ATMs. The goal is to enable platform personnel and tellers to have or provide detailed customer account information and sales prompts for cross-selling and up-selling.

Table 2-1 Some typical functions and transactions available in branch systems

Functions and transactions	
Basic functionality	
► Deposits ► Withdrawals ► Payments ► Transfers ► Official item purchase ► General ledger transactions ► Electronic journal ► Currency reporting ► Monetary instruments ► User log-in and log-out ► Bait list log ► Dual control ► Customer identification ► Customer and account setup ► Stop payments ► Customer and account inquires ► Customer and account maintenance	► Holds ► Night-deposit processing and maintenance cash advances ► Wire transfers ► Savings bond redemption ► Currency and coin orders ► Commercial deposit ► Foreign currency exchange ► Multiple transactions for single customer ► Check order ► Override processing ► Support for over 60 devices and peripherals, such as magnetic image character recognition (MICR) readers, personal identification number (PIN) pad, magnetic strip readers, teller cash dispensers, validators, printers, and other required devices
Administration and management	
► Cash management ► Operator profiles and limits	► Report and statistical updates ► Warnings for out-of-policy conditions
Productivity	

Functions and transactions	
► Sales tools ► Products presentation ► Products selection ► Scripting ► Calculators ► Signature verification	► Web browser and e-mail ► Calendar ► Branch and office locator ► Fee collection ► Fraud verification

Branch servers

When client/server technology made its evolution in the late 1980s, it became a core part of most branch systems. Various features and functions were transferred both from the mainframe to the branch to provide better performance and less WAN dependency, as well as from some PC-based stand-alone applications to a central server to allow multiple users to access and share data.

Also, features such as backup and recovery of data, mail gateways, communication hubs to remote machines or mainframe systems, or both, central printing, and many more are part of a two- or three-tier architecture. Copies of central databases are held for security and accessibility

More recently, due to faster processing, more reliable communications, and the cost of deploying and managing a fully distributed client/server environment the trend has shifted back toward more centralization. For example, Web-based applications are hosted from a single, central server and clustered for availability. In some cases, only core facilities, such as local data and printer sharing, remain on branch servers.

Table 2-2 Examples of server functions in branch banking

Server functions	
Basic functions	
► File sharing ► Printing	► Backup and restore ► User authentication
Communications	
► Database, local as server or remote as gateway ► Proxy server for Web applications	► 3270 host connectivity ► 5250 AS/400 or iSeries connectivity
Systems management	
► User administration ► Central workstation administration	► Hardware and software inventory ► Security administration

Automatic teller machine: Systems

When ATMs were introduced 30 or so years ago, banks viewed these devices as a low-cost alternative to providing routine banking services. Consumers viewed an ATM as a convenient alternative to teller branch banking, because it offered extended access to their accounts, available during extended hours.

While initially these machines were simply automated cash machines, more advanced functions have been added over the years as these terminals have been used more widely and banks have recognized the lower costs of processing routine transactions through these terminals. These machines continue to evolve, incorporating not only additional customer-initiated banking transactions, but also other dispensing capabilities, such as stamps, phone cards, and tickets.

A majority of banks and thrift organizations provide their customers access to their banking accounts through ATMs. These terminals are deployed in nearly all bank branches, as well as many non-bank locations, such as convenience stores, supermarkets, shopping centers, and tourist venues.

Some variations from the traditional ATMs have appeared. Machines with single functions, such as cash dispensers, depository, check dispensers, and others, were developed to be used inside branches. For many banking organizations, this device is viewed as an alternative channel to the branch, while for some other organizations, it is part of the branch channel.

However, the self-service devices were positioned as a complementary channel to the branch at the service level by handling many routine transactions that would have otherwise been handled by the higher-cost teller. Banks are also deploying these terminals to generate fees, to expand services to other geographies, and to increase market presence.

Although additional functions and transaction capabilities have been offered at the ATM, an evaluation of ATM transaction volumes indicates that many consumers still view these devices simply as cash machines. Approximately two-thirds of consumers use the ATM, and for a large majority of these consumers, ATMs represent an essential banking channel that offers convenient access to cash.

2.4.2 Employees

When looking at branch banking transformation and systems that can enable it, it is critical to understand the roles of employees within a branch. After all, the system is being put in place as a tool to make them more efficient and enable them to drive more business. In this section, we describe the typical roles within a branch as a basis for understanding solution requirements.

Branch manager

While in small branches, the branch manager acts both as manager and sales representative, in larger branches, we will find a dedicated branch manager and one or more sales representatives or branch account representatives.

The branch manager directs all sales and service activities, including supervising all personnel, and also develops and maintains relationships with customers and implements policies at the branch level as directed.

Functions

The branch manager functions are as follows:

- Effectively manages all product campaigns and daily sales activities.
- Leads, coaches, develops, supports, and motivates all branch staff to achieve sales and service goals.
- Ensures an environment of teamwork, sales, and goal attainment.
- Directs entire branch operation; supervises all branch personnel; responsible for making sure personnel are trained in their respective areas and cross-trained in other related areas.
- Writes or reviews annual performance appraisals; recommends salary increases; and recommends disciplinary action or termination as required.
- Responsible for branch loan and deposit growth.
- Answers verbal and written inquiries from customers, businesses, realtors, title companies, and builders regarding loan products, rates, and services.
- Originates loan applications and prepares the required disclosures. Analyzes credit requests and submits for approval to management and loan committee.
- Responds to customer inquiries on all bank products.
- Conducts marketing activities for branch by promoting business development among realtors, builders, businesses, and customers.
- Functions as branch security officer, implementing security procedures and ensuring proper security training of employees and coordinating with police agencies.

Assistant sales manager or account representative

In large branches, this role might be split into a true assistant manager and one or more branch account representatives.

The assistant sales manager or account representative performs a multifaceted retail banking role. They manage day-to-day activities of the branch, with the primary responsibility focusing on operations being efficient and effective, assist in the development of the branch staff through coaching sessions, and monitor both branch and individual progress according to goals established by the sales manager. The assistant sales manager or account representative solicits and identifies potential customers through counter transactions, in aisle sales, and incoming telephone inquiries, as well as prospecting target customers. The assistant sales manager or account representative sells appropriate bank products to fulfill customers needs and assists the bank in reaching its growth objectives.

Functions

The assistant sales manager or account representative functions are as follows:

- Sells bank products and services.
- Provides exceptional customer service to all customers.
- Represents the institution in a courteous and professional manner.
- Provides operational support for the branch, processing transactions, balancing, and training on new equipment).
- Ensures proper controls are maintained over all aspects of branch operations.
- Responsible for security and the maintenance of proper cash requirements for branch.
- Reports all operational information, reconciling, balancing, and updating files.
- Supervises and assists in the processing of all transactions.
- Completes quality aisle time requirements and daily sales goals.
- In the absence of the sales manager, responsible for ensuring all sales activities are completed.
- Participates in employee performance appraisals, interviewing of new employees, and scheduling.
- Solicits new business through promotions at the branch, in aisle sales, telephone, and mail. Identifies customer needs and sells appropriate bank products.
- Serves as a mentor and role model for all staff. Assists the individual team members in attaining their individual goals.
- Services existing customers with retail banking needs to include teller transactions.
- Follows appropriate bank, regulatory, and legal requirements.

Branch teller

The branch teller provides superior service to customers while accepting transactions, such as deposits, cashing, checks, and loan payments. The branch teller also recognizes customer needs and suggests appropriate products or services.

Functions

The branch teller functions are as follows:

- Provides prompt, efficient, and friendly service to customers at all times.
- Counts cash and coin while processing deposits, loan payments, or cashing checks according to policies and procedures.
- Post transactions to computer through an online terminal.
- Balances the cash drawer and prepares teller balance sheet on a daily basis, making sure that cash is kept to a minimum and within limits set by the board of directors.
- Sells traveler's checks, certified checks, and personal money orders.
- Processes and records night deposit envelopes and transactions.
- May assist with ATM balancing and supply.

Branch loan officer

The branch loan officer works with customers to process loan applications.

The loan officer’s mission is to provide superior service to customers in the loan area and to recognize customer needs and suggest appropriate products or services.

Functions

The branch loan officer functions are as follows:

- Sells bank products and services.
- Provides exceptional customer service to all customers.
- Represents the institution in a courteous and professional manner.
- Interviews loan customers, assists with applications, and answers basic loan questions. Completes all loan customers and advises on loan decisions.
- Services existing customers with retail banking needs to include teller transactions.

Branch auditor

The branch auditor reviews all internal processes, documents, papers, and numbers. The branch auditor usually does not report to any local management to guarantee objective reviews and findings.

Functions

The branch auditor functions are as follows:

- Reviews of the concurrent audit, RBI inspection, and internal audit reports.
- Examines the delegated authority of branch officials.
- Studies the salient features of books of instruction, especially relating to advances.
- Reviews various control returns sent by the branch to controlling offices.
- Obtains the accounting policies of the bank.
- Goes through important circulars, especially the closing circular.
- Plans the long form audit report.
- Obtains trial balance to identify the areas to be verified.
- Broadly the areas to be verified as Branch Auditors are as follows:
  - Physical verification of cash, security papers, valuable securities, and so on
  - Deposits
  - Advances
  - Sundry assets, suspense accounts
  - Sundry liabilities
  - Inter-branch reconciliation
  - Fixed assets
  - Contingent liabilities
  - Income heads
  - Expenditure heads
- Evaluates the internal control systems to assist in determining the nature and extent of audit procedures.

Branch administrator

The branch administrator is responsible for any kind of technical and operational tasks within the branch network, systems, or machines, such as servers, clients, and ATMs.

Functions

The branch administrator functions are as follows:

- Provides prompt, efficient, and friendly service to customers at all times.
- Provides technical support for the branch and training on new equipment.
- Responsible for hardware and software installation, maintenance, and service.
- Responsible for technical security and the maintenance of IT systems.
- Responsible for uninterrupted availability of all hardware, software, and infrastructure.

2.5 Common branch banking scenarios

This section describes a set of scenarios currently found in various customer environments. It gives and overview of how the business objectives are implemented within a technical architecture of systems, network, and application logic.

Each scenario is described as an overall architecture, including some technical details, such as products and functions used to accomplish business-critical workflow.

2.5.1 Host-centric scenario

This first scenario is a straight forward terminal (mainly 3270 emulation) environment with all business logic and data on a central mainframe. Servers, if they exist in a branch, are only used for gateway functions or basic file and print functionality.

Employees in the branches use 3270 terminals or thin clients for all applications, including mail. The protocol used for this environment is often Systems Network Architecture (SNA).

Transactions are always processed in a core banking solution that runs on the mainframe. There is usually one physical unit (PU) defined per branch with shared logical units (LUs) provided by a communications server or 4700 gateway.

Local servers in the branch are used for local printing queues and as the communications server gateway functions for clients using any type of 3270-emulation.

All business logic resides on the central mainframe. The local clients are used for for presentation services only.

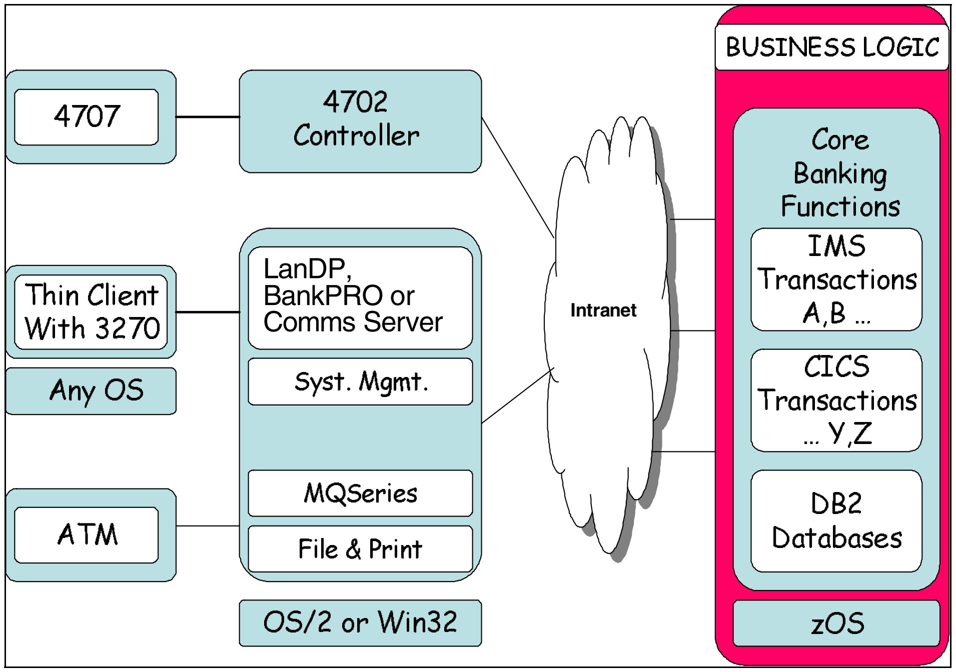


Figure 2-4 Scenario for a host-centric branch solution

2.5.2 Host-centric with local applications scenario

This scenario describes a bank environment with locally installed clients (also called FAT clients) with all transaction-oriented business logic and data on a central mainframe. The branch servers act as communication gateways and file and print servers

The branch infrastructure in this scenario operates a client-based platform based on Intel machines. The operating environment for the branch applications is predominantly IBM OS/2 at present, and branch applications have been developed using VisualAge C++.

Employees in the branches use customer-written or legacy third-party applications to work on business processes and data. Products such as Lotus

Notes are used for e-mail and the Netscape browser for intranet Web functions. The protocol used within the branch is typically TCP/IP. IBM MQSeries (WebSphere MQ) is used to integrate applications between client, server, and host systems, as well as application-to-application communication within the same client.

Transactions are always processed in a core banking solution that runs on the mainframe. For this, there is a PU defined per branch and shared LUs used by the applications. A communications server is used to implement the SNA stack with APIs providing access to LU2 and LU0 applications.

DB2 can be used to store local information for redundancy and availability. It is also used for journalling and the storage for store and forward applications utilizing MQSeries.

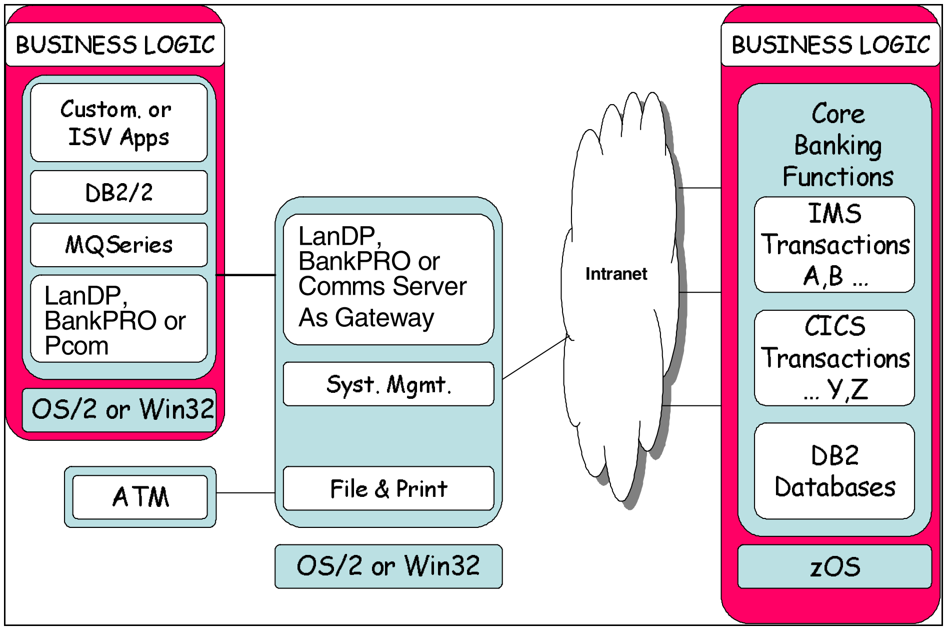


Figure 2-5 Host-centric with local applications scenario

2.5.3 Distributed processing scenario

This scenario describes a bank environment with locally installed clients (also called FAT clients) with business logic spread over local (and possibly regional) servers with data on a central mainframe. This scenario is probably most typical of many large banking environments today.

For the branch infrastructure, banks with this environment operate a client/server platform based on Intel servers. The operating environment for branch applications is predominantly IBM OS/2 at present, and branch applications have been developed using VisualAge C++.

Employees in the branches use Lotus Notes and the Netscape browser in addition to the banking applications. The protocol used within the branch is TCP/IP. MQSeries is used to integrate applications between client, server, and host systems, as well as application-to-application communications inside the same system.

Transactions are always processed in a core banking solution that runs on the mainframe. For this, there is a PU defined per branch, and a couple of LUs defined by each branch, and shared LUs are used by the applications. A communications server is used to implement SNA stack, as well as APIs to provide access to LU2 and LU0 applications.

DB2 is used to store local temporary information and journal and store forwarding messages. There is not any sensitive data maintained in the branch.

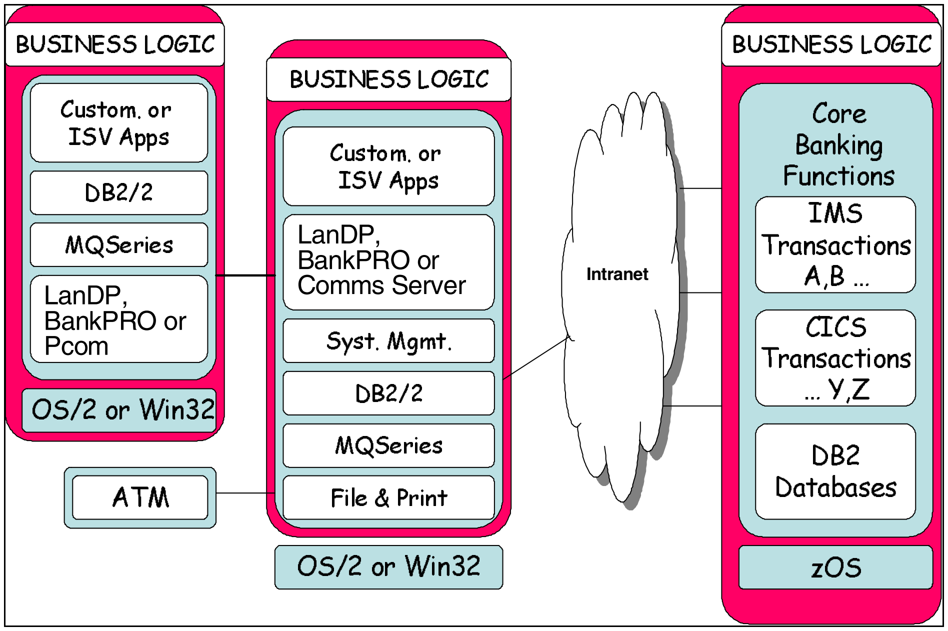


Figure 2-6 Distributed processing scenario

As an example of an application flow, customer advisors in the branch can enter a customer number to obtain a full financial profile. From the local branch application, developed using VisualAge on OS/2, an MQSeries message is generated. The message is sent through the branch-based OS/2 server across the network to the mainframe using SNA protocol. At the mainframe, the workflow application handles message processing tasks, for example, sending an MQSeries message to the CICS-based customer information file (CIF) to obtain general customer information, such as address, telephone number, and a list of accounts held.

Based on the response, the workflow application then generates more messages to obtain more specific account information, such as balance, interest, and service charges. Finally, the workflow application sends a single MQSeries message to the advisor containing all the information relating to the customer from the various systems.

The extensive branch network includes ATMs, point-of-sale (POS) terminals, and workstations. Employees in the branches use Lotus Notes for groupware and e-mail functions. Client/server branch applications have been developed using IBM VisualAge C++.

2.6 Component model for branch banking

This section describes a high-level component model for the major elements of a branch banking system. It provides the functional view of the system that is required to define the necessary infrastructure. It contains two subsections: a component diagram and component definitions.

The component diagram section shows the relative placement of functional components in the system and the high-level connectivity among the components.

The component definitions section contains the name of each component, a description of the services the component provides, and any implementation details known about the component.

Note that some components of the system are based on historical views of the geographic constraints of branch infrastructures. Options for deployment of those components in non-traditional ways are explored where appropriate.

2.6.1 Component diagram

The diagram shown in Figure 2-7 on page 37 represents a functional view of the interaction of the software components within the infrastructure. For completeness, it also shows the points of component interaction with components that are outside of the branch banking system with which the system has to interact. A component diagram normally does not depict the actual deployment of components within the environment. The diagram does show where the components have historically been deployed geographically in a branch system. The model also does not include all of the system components that may be required somewhere within the overall system, for example, Domain Name Service (DNS).

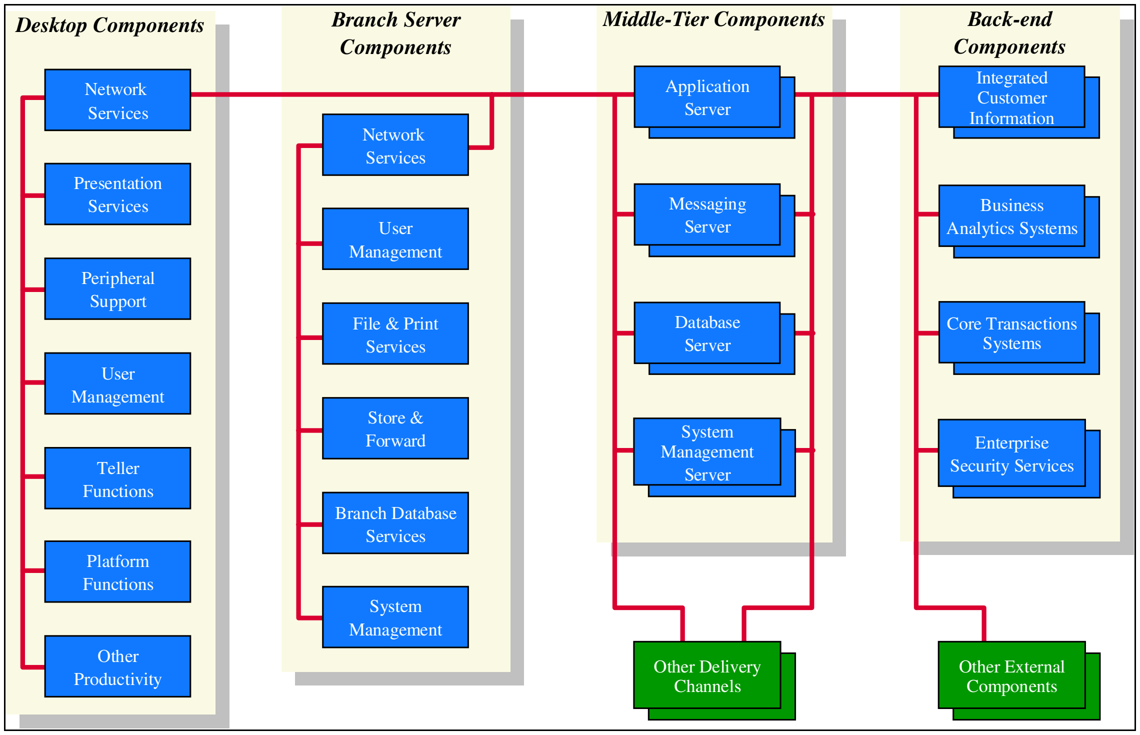


Figure 2-7 Component diagram

2.6.2 Component definitions

Based on the component diagram shown in Figure 2-7, this section provides brief descriptions of these various components.

Desktop components

Desktop components are defined at a conceptual or macro level rather than the micro or detailed design level. A great enough level of detail is given to provide possible trade-offs for *thin* versus *fat* application design models.

Network services

Network services allow the desktop application to communicate with other systems and external components. Both local LAN and external communications from the desktop are through the operating system's TCP/IP services supplemented by local LAN services if required. Communication can be through HTTP or other TCP/IP protocols as dictated by the application model. In addition, a plug-in such as IBM WebSphere Host On-Demand, or a fat client application, such as IBM Personal Communications, can be used to access traditional 3270 type applications.

Presentation services

For a thin client solution, presentation services can be thought of as the Web browser. The Web browser translates the HTML encoded pages that are received through HTTP into a form intended for presentation to the user. Presentation services in a traditional fat client implementation have greater functional capability and thus allows for richer user interfaces. Although applications can be built using either paradigm, traditional teller applications that are keyboard driven with detailed field editing are more difficult to implement using thin clients, although Java applets can provide the needed capabilities.

Peripheral support

Various specialized devices can be used to provide input to the system or output from the system. For example, a bank can use ATM cards and personal identification numbers (PINs) to identify a user when executing transactions in the branch. A magnetic stripe reader (MSR) can be used to read the ATM card and a pin-pad can be used for the customer to securely enter the PIN. In addition, banks can provide a record of a transaction for a customer through a receipt printer or a passbook printer. These devices then become integral parts of the branch system. Support for these specialized peripherals is very much dependent on the application development model that is being used. In so-called fat applications, the application uses device APIs to interface with native operating system device drivers. In thin applications, Java can be used with a standard, such as Java Extensions for Financial Services (J/XFS). More information about J/XFS can be found at:

`http://www.jxfs.com/`

User management and security

The use of a branch banking system by branch personnel (tellers and platform officers) is controlled by a combination of user authentication and role identification. This component provides a functional role in the determination of application function flow, but also a more important non-functional role in controlling access to the system for security purposes. It also provides a set of permissions and limits by which execution decisions are made. The part of the user management component that is deployed at the desktop is usually small and, in many cases, is totally centralized. The availability of user management when communications are lost to a central site can complicate the enterprise management of security.

Teller functions

There are a lot of existing teller applications that can be run on a fat client with communication to back-end and mid-tier systems through either TCP/IP or SNA protocols depending on the operational configurations. Alternatively, new applications can be written using a thin client or browser paradigm. The elements of a teller application include:

- Customer information: Data is pulled from a customer information file based on a key piece of data, such as a relationship number. This information, which includes things such as address and phone number, as well as a list of accounts for the customer, can then be used during customer servicing.
- Account information: This information includes details on an account, such as current balance and transaction history.
- Financial transactions: The transactions that change the financial state of an account (deposits, withdrawals, check cashing, and so on).
- Electronic journal: A log of all transactions that are executed by a teller. The journal may be held on the client machine, at a branch server, or at a regional server. In a completely centralized system, it may consist of a database query using a teller's ID as a key. In a system requiring offline processing, it may also provide the capability to store the transactions locally and forward them to the middle tier or back-end components when the online connection is restored.

Platform functions

Platform functionality deals with providing the customer with information about their existing accounts or services, presenting choices of financial services that a customer may want to purchase, and actually filling out applications or opening new accounts. The platform user needs functions that a teller uses, such as customer information and account information, but also needs a rich set of functions for tracking customer interactions, making sales presentations, and filling out forms. All of these functions are more presentation than transaction driven and therefore lend themselves fairly well to thin client applications that are navigated using a mouse rather than a keyboard. Customer interaction is more casual and conversational, giving the platform personnel more time to assimilate information provided by the system. In addition, the information that needs to be input is more free-form and extensive, which means that interaction with the system is less hurried. Although rich GUI-driven applications using drag and drop have been used extensively in the past, a similar experience using drop-down lists and pre-fills in a browser can be just as intuitive and effective in a less time-driven environment.

Office productivity

Some platform personnel in retail banking branch offices have need for the typical office productivity tools that have become synonymous with the evolution of the personal computer. These applications, e-mail, word processors, spread sheets, and so on, are very often used on the sales platform. There are a variety of applications that can provide these capabilities across different operating systems.

Branch server components

Most banks still believe that a branch server is required to ensure the required availability of the system at all times and to help in managing the end-user desktops. The combination of more reliable branch to data center communications and thin client applications might allow some banks to consider eliminating the branch server, but providing for local file and print sharing, as well as the ability to operate when the central system is not available provide strong reasons to keep a branch server as part of the infrastructure.

Network services

The server's network services component provides LAN communication and any gateway functionality that may be required. An example of gateway capability would be for the provision of SNA communications over a TCP/IP network. In a pure TCP/IP network, this communication function would not be required. A branch server may also provide gateway functionality for a satellite or grocery store branch.

User management

Users must be authenticated, and there must be a control point for determining their role based desktop capabilities. If secure, offline capability is required in the branch, and then this server functionality is a necessity. In addition, the application would have to have access to the user roles in order to determine what limits on functionality should be applied to a given user. One example of this need is for doing *overrides*. A head teller or branch manager may be allowed to override an offline transaction limit for a customer. A person with this higher level of authority would have to be defined as a separate *role*, and the specific characteristics of the role need to be made available for use by the application.

File and print services

Application components may have to make use of these types of services under certain application design models. An example of such a requirement would be the need to share a laser printer among multiple tellers or platform officers. File sharing is also important, and may be critical for employees who move to various machines but still need access to previously created files.

Store and forward

A store and forward mechanism is used in branch systems to ensure that transactions can continue to be accepted in the branch when the connection to the transaction system of record is not available. This is typically due to the loss of communications, but could also be for other reasons. A transaction record is stored in a database, log file, or electronic journal that contains all of the information that would normally be sent to the transaction system. It may also contain a time stamp or sequence number, or both. The forwarding mechanism is an automatic process that runs in the background to forward the transactions to the core transaction system when it becomes available. Although a branch server is a logical place to deploy a store and forward mechanism, some applications deploy this capability in each client. If communications with a regional or centralized server can be guaranteed to the required reliability level, the store and forward mechanism could be centralized. Another option would be to not have any offline capability and only handle transactions when the core transaction system is available to the branch.

Branch database services

Some application models currently use a database in the branch to house data that is required in offline mode. For example, some implementations of an electronic journal might use a database. In addition, a database may be used to store configuration data that is required during offline processing. Branch database services may not be required in more centralized application design models with limited offline requirements.

System management

One of the primary roles for a branch server is the management of end-user desktop configurations. Rather than managing desktops directly, the server can be used to cache desktop configurations. When changes are made, the system management software running on the server can automatically update the desktops.

Middle-tier components

In an e-business architecture, many of the application components of the system can be deployed in centralized servers. In this way, they can be shared by many branches, as well as by other delivery channels. Some of these components are described at a high level in the following.

Application server

An application server is the software component within an e-business architecture where an application's business logic is executed. It acts as an intermediary between the end-user interaction components on the front-end and the data components on the back-end. In a banking branch system, there may be multiple application servers for different business needs.

Messaging server
A messaging server provides a common communications transport layer for all applications within the infrastructure. It can be used to provide routing based on business rules, as well as translation from one message format to another. Use of a messaging server provides an integration point for tying together disparate system components.

Database server
A middle-tier database server provides the branch banking system with application and system data specific to servicing at the branches or other delivery channels. The requirements for this database will be very dependent on the needs of various applications and the design of those applications. Some application design models may not require a middle tier database but obtain all required data from the back-end systems.

System management server
A system management system in the middle tier is used to administer all system management functions for the branch system. Configurations for all desktops in the delivery system are administered through this server. When configurations change, the server is responsible for distributing those changes to the branch server, which in turn, administers those changes to the individual desktops. In other system management designs, this server may be configured to manage desktops directly. In addition to configuration management, the server may also be a collection point for alerts generated by downstream components indicating failures or performance bottlenecks.

Back-end components
The back-end components are not part of the branch banking system. They are enterprise systems that provide services required by the branch banking system. In order to complete the architecture of the branch banking system, all interfaces to the back-end systems need to be well defined and understood.

Integrated customer information
Integrated customer information or customer information files (CIF) hold an integrated view of a customer's total relationship with the bank.

Business analytics
Business analytics can provide analysis of a customer's financial holdings and transaction history to provide guidance on product suitability and financial risk.

Core transaction systems
Core transaction systems provide account-level transaction capability and history.

Enterprise security systems
Enterprise security systems provide the information necessary to authenticate users and determine their system-level permissions.

2.7 Summary

The branch banking environment, which has not changed significantly over the last decade, is starting to go through a major transformation. This chapter has described some of the drivers for this transformation.

In addition, it has provided general information about the structure of branch offices and the types of systems that have typically been in place to support them. We have also provided three different scenarios that represent how many banks operate their branch banking systems today.

If you have been in the banking industry for very long, this chapter may not have provided you with much information that you were not already familiar with. However, it is important to provide this information as a basis for discussions in the upcoming chapters. The next chapter provides some specific information about branch banking requirements, and that sets us up for the remainder of the redbook to discuss a Patterns for e-business approach to designing a solution and understanding where Linux can be an important piece of that solution.

3

Branch banking requirements

In this chapter, we continue our discussion of branch banking by focusing on some of the requirements that will drive the ultimate solution.

We begin with some general objectives and principles. We next look at the requirements from both a business and system context and then look at more detailed functional and non-functional requirements.

3.1 Solution architecture objectives and principles

The solution architecture objectives and principles can be divided into three primary categories:

- Cost-related objectives
- Implementation-related objectives
- Programming-related objectives

3.1.1 Cost-related objectives

The solution architectural objectives must be aligned with IT strategies that have a basis in controlling costs over time. These objectives are as follows:

- **Reduce costs**: A network computing architecture should exploit the network in order to reduce costs. It allows reduction of the computing resources required on the client and supports deployment on network computers, using the network as a vehicle for on-demand distribution of software components. In addition, the architecture supports deployment of reusable business components in a managed server environment.
- **Preserve investment**: An important goal is to preserve the financial institution's investment in the host systems and computing infrastructure, as well as in other new technologies. This makes it important to carefully consider technology selections in order to ensure that they are strategic and will have enduring value.
- **Offer choices**: Allow customers the flexibility to choose their hardware, operating systems, networking systems, databases, communication protocols, and third-party software products. The system must also support flexible distribution of function and data based on the network environment and physical topology.
- **Evolve gracefully**: The system must be flexible and resilient to both business and technological changes. This helps to support rapid application development and to increase competitiveness by improving time to market.
- **Provide manageability**: Once deployed and in production, the system must be easy to manage and resilient to changes in the run-time environment. It must also support remote management. Servers and clients must be able to be managed by a remote site, to solve technical or business issues that can happen during operation.
- **Allow incremental investment**: The system must support the ability to incrementally develop and deploy new business function and technology. In addition, it must support the ability to include new solutions as they become available.

- **Maximize usability**: The system as a whole must be well suited to the needs of its users: not only end users, but also developers and systems management personnel.
- **Maximize reusability**: The system must be constructed in such a way as to maximize reuse of components in all retail delivery solutions. In addition, it must be able to meet the diverse needs of solutions and access channels in financial institutions around the world.

3.1.2 Implementation-related objectives

The solution architecture must be open, scalable, and easy to implement. These principles are related to the architecture objectives and are the basis for the platform selections, programming model specifications, and overall functional requirements of the solution. These objectives are as follows:

- **Open**
  - **Supports industry standards**: An architecture is open when it uses open industry and e-business standards, such as TCP/IP, HTML, HTTP, Java, JDBC, and JavaServer Pages (JSPs), wherever possible. These standards provide a solid foundation and make it easier to use available proven components instead of building custom ones and to change vendors and implementations to satisfy changing business requirements. Industry standards tend to be strategic and have longer life spans because of the high levels of investment and commitment involved with creating them.
  - **Extendable and customizable**: An architecture must be built to be extendable and customizable at many different layers or channels. This means it can be used in a wide range of situations and can accommodate specialized requirements that are specific to an individual channel, customer, country, or region.
  - **Provide insulation**: The solution architecture must isolate and abstract interactions with other systems to insulate architecture-based applications from the specifics of other systems. In a global solution, this is essential to provide the flexibility to adapt to many diverse environments, particularly different host systems and databases.
  - **Preserve investment**: The principles listed above ensure the preservation of customer investments. The solution architecture safely preserves the investments in current hardware, software, operating systems, network, communication infrastructure and protocols, and back-end subsystems of the customer environment.

- **Scalable**
  - **Logical tiers**: The benefits of a logical tiers architecture, such as the Network Computing Architecture (see the *Network Computing Framework Component Guide*, SG24-2119), are well known. The Network Computing Architecture is logical in that it specifies that the presentation layer must be decoupled from the business logic, which must be decoupled from the data access layer, but it does not specify how to physically deploy the tiers. Although this approach is a form of isolation, it also provides scalability by allowing each of these layers of the system to change independently of the others. That is, the platform selections and design of each layer can change without impacting the rest of the system. This architecture also requires that the presentation layer be thin to realize the goals of network computing. This means that workstations with a small amount of physical memory and no virtual memory can download and execute the application.

    The main objective of the solution architecture is to support the model of a multiple-tier network computing application while also allowing engagement teams to implement solutions based on other application models, such as a two-tier fat client application.
  - **Replaceable components**: Components are packages of system function with established interfaces and a predetermined execution environment. As long as a component is within its required execution environment, and it interacts with other system components through its public interfaces, it is replaceable with minimal effort. This construction enables high levels of reuse and allows the system to evolve without causing large ripple effects. It also allows the implementation of components and their execution environments to vary to meet performance or scalability requirements.
  - **Enterprise topology independence**: This notion extends the idea of a logical tiers architecture so that not only are the tiers independent of physical location, but system components are independent of any specific physical topology. This makes the solutions highly flexible for deployment in different environments by allowing customers to configure the system as needed to achieve the scalability desired for their environment.
- **Easy to implement**
  - **Visual programming**: Where possible, framework-based solutions use visual programming to assemble the application from parts. This technique is particularly effective in developing application screens and rapid assembly of graphical user interfaces.

– **Separate analysis from design**: Analysis should be a separate process from design and have its own distinct work products. Solutions of this product suite should use analysis to form an entirely logical representation of system function that is independent of technology or implementation. This helps to retain the value of earlier development effort even if the implementation must change entirely.

– **Development methodology**: A methodology for guiding the development process in an engagement project to make solution implementation easier and the deployment faster must be provided.

– **Transaction-oriented**: Most branch products require a solution in which an enterprise-centric, back-end system executes most of the application business logic, and the front end of the solution, running in a delivery channel, must behave as a transaction posting engine to run the transactions in the back-end system. A well-defined architecture must address this type of requirement, especially in a high-volume transaction processing environment.

– **Development effort**: An easy way to implement new transactions or applications is based on the parameter's externalization, so business operations behave differently depending on their specific set of external parameters. This enables solutions to deliver new functions and capabilities without requiring new coding, simply by adding new external parameters to the system.

3.1.3 Programming-related objectives

Development of new business applications or re-engineering existing applications often focuses on the Java programming language and on mature Internet technologies creating reusable components within an overall framework for your application. This ensures that component-based applications can be deployed with confidence as integral parts of robust production systems, while a framework provides consistency and completeness over your components.

Application development needs to be based on a shared central repository that supports component reuse and extensive parameterization of object definitions. Its design hides technical issues from solution designers, which allows them to focus on business function rather than on the underlying technical details.

These features create benefits in the areas of project completion time, intermediate- and long-term cost-effectiveness, and readiness for future changes, improvements, and evolution.

For example, IBM WebSphere Business Component Composer provides these features.

These objectives are as follows:

- **Reduced risk**: Components are a fast and competitive way to solve your application needs, but being fast and competitive does not mean that you are left exposed to risk.
  - **Proven product**: Components are a mature foundation for products developed for software applications for the financial services sector.
  - **Systems work together**: The extensive use of open computing industry standards (including Internet standards) protects against incompatibilities between systems.
  - **Protection against obsolescence**: The inherent flexibility and the ability to update component-based applications protects these applications from becoming obsolete.
  - **Fast response to the business environment**: The application development environment allows quick changes to applications in response to changing business conditions.
  - **Build it right the first time**: The application development environment is required to support teamwork, and this, in turn, promotes dialog and sharing of ideas; fewer details will be overlooked.
  - **Preserve stable IT infrastructures**: Existing systems that provide reliable service can continue to be used through communications components. These components provide connectivity to legacy systems.
- **Faster time to market**: The development approach that a framework of components promotes is designed to shorten development cycles and flatten the learning curve for the project team. The objective of this approach is to effectively save development effort, improve consistency, and reduce the time to market for all delivery channels.
  - **Shortened development cycles**: A framework provides an environment that supports rapid application development by exploiting the benefits of component reuse. It does this by promoting the extensive use of object-oriented techniques and a high degree of application object parameterization.
  - **Ready-to-use components**: A framework of components provide a set of prebuilt infrastructure components with well-defined interfaces. The components are immediately available for development and are ready to be incorporated into delivery channel applications. A project team needs only to learn how to use them, not how to build them.

- **Parametric application definition**: The development model is based on a centralized object repository where all the relevant information about the application is maintained. With this model, adding a new function to a framework-based application usually involves little more than using the Development Workbench to add definitions to the Development Workbench Repository.
- **Flattened learning curve**: A component-based framework usually hides the underlying technical details of the framework. This reduces the amount of time and effort needed by a project team to learn the framework features and how to use them to deliver a solution. The development model creates a clear separation of roles that allows project team members to focus on their specific tasks.

3.2 Business context

The context diagram, shown in Figure 3-1 on page 52, covers the following essential elements of the branch banking system and its external interfaces:

- Customers request services and support, as well as receive those services. Both activities can involve one or more channels
- Tellers provide services and support to customers on one of the other channels. Therefore, they might no longer be a traditional teller behind a desk, but also a virtual teller in an online bank branch.
- A platform officer provides services and support within the branch or central side, so they are more or less back-end support staff for front-end processes. These can also be online brokers, loan officers on the phone, and so forth.
- Concierge or support staff in the branch is both for convenience of customers and tellers or officers in the branch, like any administrative person.
- Enterprises can be a receiver or requester for services from or to a branch or a bank overall.
- Business partners provide services beyond the core scope of the branch for customer convenience and portfolio completeness.

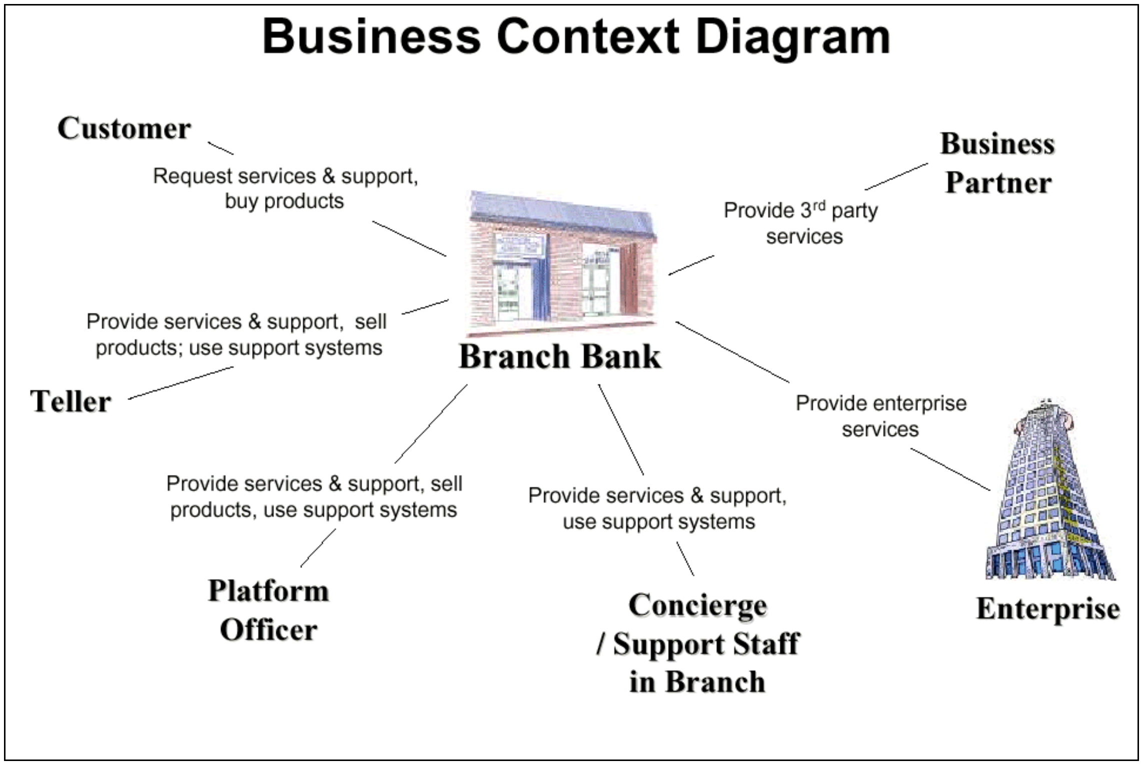

Business Context Diagram
Customer
Request services & support, buy products
Business Partner
Provide 3rd party services
Branch Bank
Provide services & support, sell products; use support systems
Teller
Provide enterprise services
Provide services & support, sell products, use support systems
Provide services & support, use support systems
Platform Officer
Concierge / Support Staff in Branch
Enterprise

Figure 3-1 Business context diagram

The following lists, although not comprehensive, provide a summary of the types of business tasks that must be taken into account when defining the requirements for a branch banking solution.

Simple tasks

Simple tasks are defined by two characteristics, one is the involvement of a single person, either a teller or a branch sales representative, and the other is a relatively straight forward transaction.

Examples of simple tasks include:

- Deposit of withdrawal for one account, where a teller enters the account number and the amount being transferred into the system
- Clearing a check at the teller’s desk
- Automatic money withdrawal on an ATM
- Money transfer between two accounts within the bank or within the country

- Selling a simple bank product on an automated sales point
- Querying information for a bank product
- Credit line verification by the account representative or the branch manager

Complex tasks

Complex tasks are defined as tasks that either require more than one person to work on, for example, the 4-eye principle or 2-key security, as well as any kind of multistaged application processing. This could be either real time or asynchronous using message queueing or workflow processing. These processes can have the following different types of required interactions:

- Human to human: A teller requires approval from the branch manager for a certain transaction.
- Multiple information sources: A process can only be completed by gathering information from multiples sources or writing multiple results back to a set of targets.
- Business to business: A good example would be an alliance between the bank and an insurance company, where both resell each others products and have access to each others systems for information and data exchange.

Examples for complex tasks are as follows:

- Opening a new account that requires the approval from branch management, as well as clearance from a central authority for checking the credit history of that person.
- Any large withdrawal that might fall under money laundering regulations requires approval on-site and potentially from the central side and is recorded and tracked.
- A more complex investment plan or loan application that may involve opening additional, special accounts, financial checks, or batch process calculations.
- An alliance between the bank and an insurance company, where both resell each others products and have access to each others systems for information and data exchange.
- Portfolio management requires information from multiple sources to be queried, processed, and put together to extract a portfolio recommendation for a specific customer or customer requirement. Also, managing this portfolio may require access to various accounts, even across banks.
- Credit card approval usually involves a central clearance application to process the request and approve the credit of the requesting person.

3.3 System context

The context diagram, shown in Figure 3-2, covers the following essential elements of the branch banking system and its external interfaces:

- End users (tellers and platform personnel)
- The branch banking system composed of:
  - Desktops in the branches and back office
  - Server (optional)
  - Infrastructure components
  - Regional and centralized application and infrastructure components
- The back-end resources and systems represent bank systems of record or utility functions with which the application must interface in performing its mission. These include customer information systems, transaction processing systems, and business analytics.
- Bank enterprise security services provide authentication and access control to the applications.

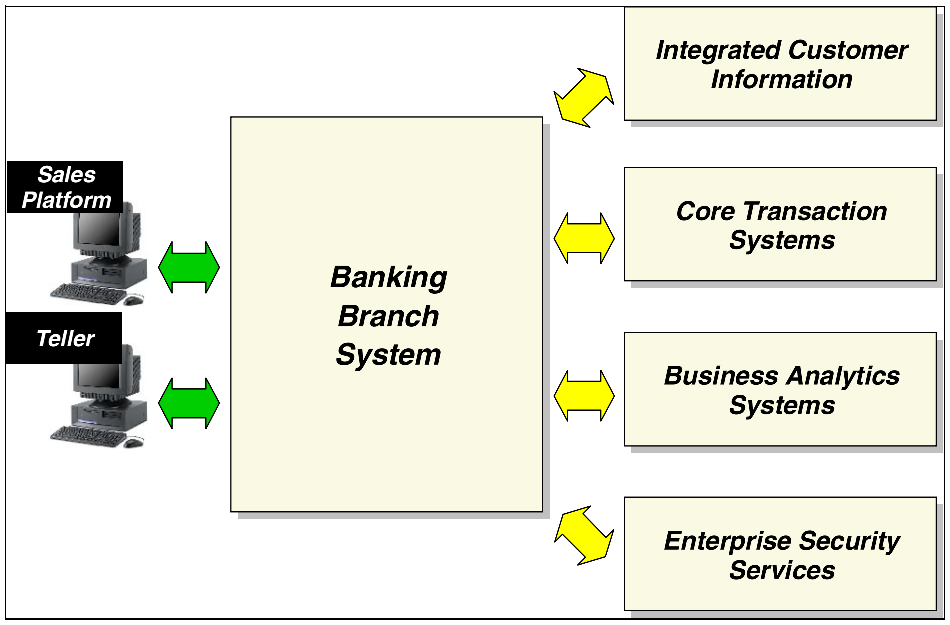

Sales Platform
Teller
Banking Branch System
Integrated Customer Information
Core Transaction Systems
Business Analytics Systems
Enterprise Security Services

Figure 3-2 System context diagram

3.4 Functional requirements

The functional requirements of a banking branch system are those customer service functions required to open accounts, make inquiries on accounts, and provide transactional services. Branch banking requirements are usually split into two broad categories: teller and platform.

Teller functionality includes taking deposits of cash and checks, providing cash and check withdrawals, paying loans, and so on. Platform functions include opening accounts, taking loan applications, and providing information about products and services. Although platform functionality is of more value to the bank because of the emphasis on bringing in revenue, teller functionality is somewhat more mission critical, because it involves a need for a consistent level of service where and when the customer requires it.

Teller functionality is well defined, and the application requirements are well known and fairly static. Platform requirements, on the other hand, vary from bank to bank and tend to change over time as new ways of providing information and selling to customers evolve. Office productivity tools, such as word processors and spreadsheets, may handle some of these requirements. Tellers and platform officers, as employees of the financial institution, may also have a need for indirect applications, such as HR, e-mail, and general information. Most of these can be handled by iintranet access through a Web browser.

In addition to the actual application functionality, other functional requirements include the ease and speed of changing or adding new functions and a requirement for providing consistent service across multiple retail delivery channels. These requirements have led to the recent innovations in providing multichannel functionality that can be reused in all others delivery channels.

3.4.1 Operational considerations

Operational considerations deal with qualities or constraints that an IT system must satisfy. These include service level requirements, such as reliability, availability, serviceability, security, scalability, performance, workload management, and systems management. This section provides some early details about the architecture elements needed to ensure that the non-functional requirements of the banking branch system are met.

Reliability, availability, and serviceability

The availability expectations of a system relate to how many hours in the day, days per week, and weeks per year the system is going to be available to its users and how quickly they should be able to recover from failures. In order to meet the reliability, availability, and serviceability (RAS) objectives, there should be no single points of failure in essential system nodes. All application servers, database servers, messaging servers, and back-end servers would be deployed using redundancy, load balancing, and clustering technology. In the branch, the availability of users to work at any desktop would ensure that desktops would never be a single point of failure. If a desktop failed, the user could just move to a machine that is not in use.

Persistent data should not be stored on desktops except for logging or backup purposes. Requirements for being operational in the face of multiple failures (for example, client and server or server and network) would be costly and may not be a justifiable expense. Servers are potentially a single point of failure and that is why they should be used in backup scenarios with primary operation through redundant centralized servers, where the cost can spread over more users.

Because of the redundant nature of the middle-tier and back-end system nodes, most planned maintenance can take place while the rest of the system remains online and available to end users. Desktops and other branch system nodes can be maintained during non-business hours.

Security

The primary requirements for security must be achieved naturally in the branch system architecture. It must have a locked down desktop that only permits access to applications that are controlled by the server and the system administrator.

In the preferred mode, users would not be able to load their own applications into the desktop. Any disks loaded by the user would be for data backup and would not be executable. Internet access would either not be allowed or would be accessible through a proxy server with sufficient network *farewell* controls. File transfers initiated by the user would also not be allowed. In addition, the branch system must be customized to the role or roles allowed for a specific user, and therefore, access to information and transactional systems would be tightly controlled.

Optionally, all communications between the branches and the regional or central data centers can be encrypted to ensure data confidentiality, and message authentication codes can be attached to all data communications to ensure that data is not altered.

Performance

A performance model must be used as a method of sizing the physical components of an IT Infrastructure. The performance model attempts to capture a combination of the application behavior and the non-functional requirements. The elements of a performance model include node performance, application performance, network performance, and database performance. These elements are explored more deeply as the architecture is detailed to the specification and physical implementation levels.

System management and maintainability

System management and maintainability refers to the maintenance and administration of the branch software and hardware systems. It involves configuration management, application management and maintenance, and monitoring of all system nodes and application components for early detection of failure or performance deterioration.

Configuration management

Configuration management deals with the tracking and deployment of hardware nodes and software components across the enterprise. For a large financial institution's branch system, this can be a large undertaking involving 5,000 locations, more than 75,000 hardware nodes, and multiple system and application software footprints.

Application management and maintenance

This is oftentimes viewed as part of configuration management. This involves the ability to update both system and application software components to meet changing business requirements, such as adding new functions and changing data formats. Application changes are made by generating new *packages* and including them in revised desktop definitions. System changes can be made using tools that are available to update the systems. In a branch system, the deployment of the updates becomes very difficult. Specialized system management components and nodes may need to be defined when driving the architecture to the specification and physical level in order to simplify the deployment process.

System monitoring

Agents that regularly poll the site and measure its response time will perform system monitoring. The system needs to be integrated to the centralized monitoring system so that branch systems-related alerts would be distributed in the same manner as other error conditions in the rest of the bank's system.

3.5 Non-functional requirements

The non-functional requirements of an IT system specify those aspects of the system that are not directly related to the application functionality. The primary non-functional requirements that need to be dealt with are service level requirements, such as capacity and performance (volumetric), availability, security, and system management.

Other typical non-functional requirements include properties to ensure portability and maintainability and any system constraints. Typical system constraints include business constraints that the system must satisfy (for example, geographical location), existing technical standards, and technical "givens" that constrain the system architecture and design (for example, existing IT infrastructure).

The non-functional requirements detailed in the following sections form the basis for the trade-offs and decisions made in the architecture and design of a representative retail banking branch infrastructure. They directly affect the choice and number of various components and thus the overall cost of the infrastructure.

3.5.1 Cost of operation

The costs, both for operations and technology, are one of the major drivers for decisions on all levels within a branch bank environment. While some of the following requirements have only implicit influence to overall costs, the total costs for the new solution need to meet business requirements. It is not unusual that the return on investment is required to be less than 12 months.

A 1996 study by Gemini Consulting found that the cost to banks of financial transactions varies widely, depending on how they are accomplished. The study does not account for up-front costs, such as installing a PC banking system or building a bank branch office. [1]

[1] http://news.mpr.org/features/199608/01_catlinb_banking/banktranscript.htm

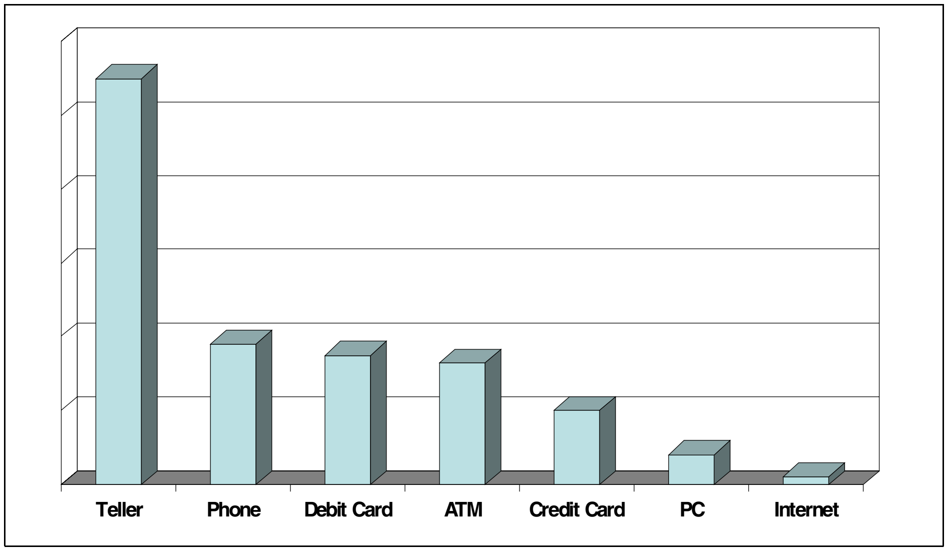


Figure 3-3 Estimated bank costs per transaction

3.5.2 Capacity, performance, and scalability

In a distributed system, such as a branch system, the emphasis is not so much on the capacity of individual components in the branch, but on how the branch components integrate to the up-stream components. With the current state of technology for most application designs, performance is usually more affected by things such as communications bandwidth and the number of transactions aggregated from all branches than the performance of an in-branch server or end-user desktops.

3.5.3 Reliability and availability

Branch banks need to be available on a typical business schedule, usually five days a week, eight hours a day, excluding holidays. There may be extended hours one day a week or shortened hours on Saturday. During the time a branch is open, information systems need to be available essentially without failure. The total system uptime, including the ability to process some transactions when communications to back-end systems are not available, must approach 100%. If the branch cannot accept transactions in some manner, then it might as well close. Therefore, contingency plans and systems must be in place to cover any system component failure or temporary loss. For purposes of this document, the required maximum system failure recovery time is assumed to be few hours, and the required system restart time is assumed to be less than minutes. Because the branches are open only during scheduled business hours, maintenance is easily scheduled during non-business hours.

The middle-tier and back-end systems that support the branch channel must have reliability and availability characteristics that support the branch requirement. Because these systems also support other channels, such as the Interactive Voice Response (IVR), call center, and Internet, they are normally designed for virtual 24x7 availability with the required reliability and redundancy to achieve those goals.

3.5.4 Security

The main security requirements of a banking branch system are as follows:

- Local and remote user authentication
- Functions authorization based on roles
- Data confidentiality
- Protection from malicious or fraudulent attacks

Detailed security requirements can be very specific to a particular bank, because the ways in which security is implemented in the enterprise will vary widely. Other considerations, such as single signon and the need for data encryption, may also be dependent on specific policies, procedures, and network implementation.

3.5.5 System management

As with any IT system, the architecture needs to accommodate the ability to monitor all system components and respond to outages using automated procedures to the greatest extent possible. In the case of a branch system, there are some key specific requirements for achieving this goal, as follows:

- The ability to update both system and application code automatically
- The ability to restore a failed system to its pre-failure state
- The ability to add new applications to previously installed systems
- The ability to detect, diagnose, and correct problems quickly with minimal human intervention

The distributed nature of a branch system presents cost challenges for system management, because the number of end-user desktops is high, and there may be communication limitations in deploying common system management tools.

3.5.6 User and desktop management

In a distributed system, such as a branch system, the administration of the environment might not be local to the branch, so a remote configuration of users, roles, applications, and desktop settings is required. This goes along with the ability to provide a consistent desktop for a user logging on to different physical machines, or multiple users logging on to a single machine at different times, a roaming user.

Furthermore, a consistent look and feel needs to be accomplished, so users would not necessary realize which underlying platform their client machines are running on.

3.5.7 IT standards and existing IT infrastructure

IT infrastructure and IT standards constraints are very specific to individual banks or financial institutions. Common constraints are the choice of operating systems and hardware platforms, specific database products, network topology, security products and processes, and enterprise system management products. When drilling down to the specification and physical implementation architecture levels, the existing standards and infrastructure present constraints that will oftentimes affect the available choices for functional components or operational nodes. At the conceptual level, some typical choices have been made in this document that may or not be correct for a particular bank implementation.

3.5.8 Geographic constraints

Branch systems have the same constraints in virtually all geographies. Although urban settings in a specific country or region may have better infrastructure than cities in other parts, branch offices of large financial institutions may exist in rural areas with limited or cost prohibitive access to some communication services. Therefore, the architecture will have to accommodate data communication reliability and availability limitations. Specific implementations of the architecture, however, may be able to take advantage of superior communication capability.

3.6 Change cases

Any potential future requirements that the system may have to support should be evaluated and validated to ensure that the architecture is able to accommodate them. IBM defines *change cases* as possible future changes to a system. In this document, change cases are used to indicate areas in which both functional and non-functional requirements may change for different banks, or areas in which emerging technologies may allow significant change in the system infrastructure.

Proper focus on potential change cases ensures that the system is easy to extend, easy to maintain or port, robust in the face of change, and quick to develop. The focus is on what is important and likely, rather than what is possible. Change cases try to predict change. Such predictions rarely turn out to be exactly true. Users, sponsors, suppliers, developers, and other stakeholders determine the properties of a system. Changes can arise from many sources, for example:

- Business drivers: New and modified business processes and goals
- Technology drivers: Adaptation of the system to new platforms, integration with new components
- Changes in the profile of the average user
- Changes in the integration needs with other systems
- Scope changes arising from the migration of functionality from external systems

Because change cases are predictive, they are themselves subject to change. Were they 100% certain, they would, by definition, be deferred requirements.

3.7 Summary

This chapter has described some of the requirements for a branch banking system from several different perspectives. These requirements will be used as input when we look at IBM Pattern for e-business that apply to branch banking solutions.

Whether you use Patterns for e-business or another methodology to architect your solution, these requirements and ways of looking at them are still applicable. As shown in Chapter 6, “Linux-based products applicable to branch banking” on page 117, Linux can play a key role for building an infrastructure that meets these requirements.

4

IBM Patterns for e-business overview

IBM Patterns for e-business are a collective set of proven architectures that have been compiled from more than 20,000 successful Internet-based engagements. This repository of assets can be used by companies to facilitate the development of Web-based applications. They help an organization understand and analyze complex business problems and break them down into smaller, more manageable functions that can then be implemented using low-level design patterns.

4.1 Introduction to Patterns for e-business

As companies compete in the e-business marketplace, they find that they must re-evaluate their business processes and applications so that their technology is not limited by time, space, organizational boundaries, or territorial borders. They must consider the time it takes to implement the solution, as well as the resources (people, money, and time) they have at their disposal to successfully execute the solution. These challenges, coupled with the integration issues of existing legacy systems and the pressure to deliver consistent high-quality service, present a significant undertaking when developing an e-business solution.

In an effort to alleviate the tasks involved in defining an e-business solution, IBM has built a repository of patterns to simplify the effort. In simple terms, a *pattern* can be defined as a model or plan used as a guide in making things. As such, patterns serve to facilitate the development and production of things. Patterns codify the repeatable experience and knowledge of people who have performed similar tasks before. Patterns not only document solutions to common problems, but also point out pitfalls that should be avoided. IBM Patterns for e-business consists of documented architectural best practices. They define a comprehensive framework of guidelines and techniques that were actually used in creating architectures for customer engagements. Patterns for e-business bridge the business and IT gap by defining architectural patterns at various levels, from Business patterns to Application patterns to Runtime patterns, enabling easy navigation from one level to the next. Each of the patterns (Business, Integration, Application, and Runtime) help companies understand the true scope of their development project and provide the necessary tools to facilitate the application development process, thereby, allowing companies to shorten time to market, reduce risk, and more importantly, realize a more significant return on investment.

The core types of Patterns for e-business are as follows:

- Business patterns
- Integration patterns
- Composite patterns
- Application patterns
- Runtime patterns and matching product mappings

When a company takes advantage of these documented assets, they are able to reduce the time and risk involved in completing a project.

For example, a line-of-business (LOB) executive who understands the business aspects and requirements of a solution can use Business patterns to develop a high-level structure for a solution. Business patterns represent common business problems. A LOB executive can match their requirements (IT and business drivers) to Business patterns that have already been documented. The patterns provide tangible solutions to the most frequently encountered business challenges by identifying common interactions among users, business, and data.

Senior technical executives can use Application patterns to make critical decisions related to the structure and architecture of the proposed solution. Application patterns help refine Business patterns so that they can be implemented as computer-based solutions. Technical executives can use these patterns to identify and describe the high-level logical components that are needed to implement the key functions identified in a Business pattern. Each Application pattern would describe the structure (tiers of the application), placement of the data, and the integration (loosely or tightly coupled) of the systems involved.

Finally, solution architects and systems designers can develop a technical architecture by using Runtime patterns to realize the Application patterns. Runtime patterns describe the logical architecture that is required to implement an Application pattern. Solution architects can match Runtime patterns to existing environment and business needs. The Runtime pattern they implement establishes the components needed to support the chosen Application pattern. It defines the logical middleware nodes, their roles, and the interfaces among these nodes in order to meet business requirements. The Runtime pattern documents what must be in place to complete the application, but does not specify product brands. Determination of actual products is made in the product mapping phase of the patterns.

In summary, Patterns for e-business captures e-business approaches that have been tested and proven. By making these approaches available and classifying them into useful categories, LOB executives, planners, architects, and developers can further refine them into useful, tangible guidelines. The patterns and their associated guidelines allow the individual to start with a problem and a vision, find a conceptual pattern that fits this vision, define the necessary functional pieces that the application will need to succeed, and then actually build the application. Furthermore, Patterns for e-business provides common terminology from a project’s onset and ensures that the application supports business objectives, significantly reducing cost and risk.

4.2 The Patterns for e-business layered asset model

The Patterns for e-business approach enables architects to implement successful e-business solutions through the reuse of components and solution elements from proven, successful experiences. The Patterns for e-business approach is based on a set of layered assets that can be exploited by any existing development methodology. These layered assets are structured in such a way that each level of detail builds on the last. These assets include:

- Business patterns that identify the interaction between users, businesses, and data.
- Integration patterns that tie multiple Business patterns together when a solution cannot be provided based on a single Business pattern.
- Composite patterns that represent commonly occurring combinations of Business patterns and Integration patterns.
- Application patterns that provide a conceptual layout describing how the application components and data within a Business pattern or Integration pattern interact.
- Runtime patterns that define the logical middleware structure supporting an Application pattern. Runtime patterns depict the major middleware nodes, their roles, and the interfaces between these nodes.
- Product mappings that identify proven and tested software implementations for each Runtime pattern.
- Best-practice guidelines for design, development, deployment, and management of e-business applications.

These assets and their relation to each other are shown in Figure 4-1 on page 67.

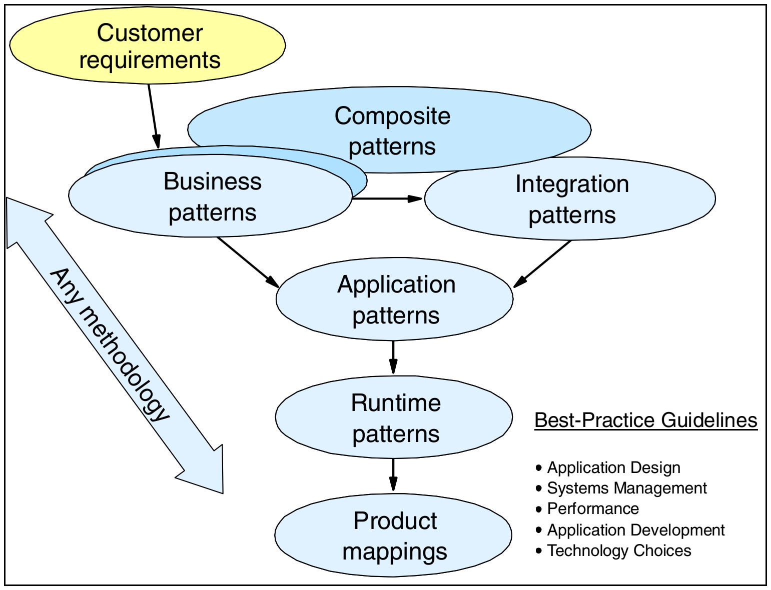


Figure 4-1 Patterns layered asset model

Patterns for e-business Web site

The Patterns for e-business Web site provides an easy way of navigating top down through the layered Patterns for e-business assets in order to determine the preferred reusable assets for an engagement.

For easy reference to Patterns for e-business, refer to the Patterns for e-business Web site at:

http://www.ibm.com/developerWorks/patterns/

4.2.1 How to use Patterns for e-business

As described in the previous section, Patterns for e-business are structured in such a way that each level of detail builds on the last. At the highest level are Business patterns that describe the entities involved in the e-business solution. A Business pattern describes the relationship between the users, the business organization or applications, and the data to be accessed.

Composite patterns appear in the hierarchy shown in Figure 4-1 above the Business patterns. However, Composite patterns are made up of a number of individual Business patterns and at least one Integration pattern. In this section, we discuss how to use the layered structure of the Patterns for e-business assets.

There are four primary Business patterns, as shown in Table 4-1.

Table 4-1 Business patterns

Business patterns	Description	Examples
Self-Service (User-to-Business)	Applications where users interact with a business via the Internet	Simple Web site applications
Information Aggregation (User-to-Data)	Applications where users can extract useful information from large volumes of data, text, images, and so on	Business intelligence, knowledge management, Web crawlers
Collaboration (User-to-User)	Applications where the Internet supports collaborative work between users	E-mail, community, chat, video conferencing, and so on
Extended Enterprise (Business-to-Business)	Applications that link two or more business processes across separate enterprises	EDI, supply chain management, and so on

It would be very convenient if all problems fitted nicely into the four Business patterns, but reality says that things will often be more complicated. The patterns assume that all problems, when broken down into their most basic components, will fit more than one of these patterns. When a problem describes multiple objectives that fit into multiple Business patterns, Patterns for e-business provide the solution in the form of Integration patterns.

Integration patterns allow us to tie together multiple Business patterns to solve a problem. The Integration patterns are shown in Table 4-2.

Table 4-2 Integration patterns

Integration patterns	Description	Examples
Access Integration	Integration of a number of services through a common entry point	Portals
Application Integration	Integration of multiple applications and data sources without the user directly invoking them	message brokers, workflow managers

These Business and Integration patterns can be combined to implement installation-specific business solutions. We call this a Custom design.

We can represent the use of a Custom design to address a business problem through an iconic representation, as shown in Figure 4-2.

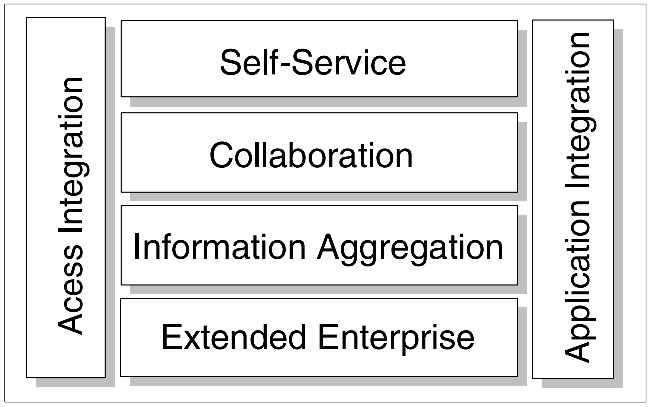


Figure 4-2 Pattern representation of a custom design

If any of the Business or Integration patterns are not used in a Custom design, we can show that with lighter blocks. For example, Figure 4-3 shows a custom design that does not have a mandatory Collaboration business pattern or an Extended Enterprise business pattern for a business problem.

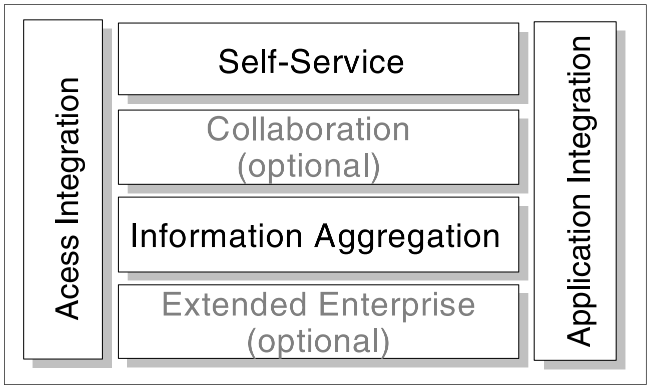


Figure 4-3 Custom design

A Custom design may also be a Composite pattern if it recurs many times across domains with similar business problems. For example, the iconic view of a Custom design in Figure 4-3 can also describe a Sell-Side Hub composite pattern.

Several common uses of Business and Integration patterns have been identified and formalized into Composite patterns. The identified Composite patterns are shown in Table 4-3.

Table 4-3 Composite patterns

Composite patterns	**Description**	**Examples**
Electronic Commerce	User-to-online-buying.	► www.macys.com ► www.amazon.com
Portal	Typically designed to aggregate multiple information sources and applications to provide uniform, seamless, and personalized access for its users.	► Enterprise intranet portal providing self-service functions, such as payroll, benefits, and travel expenses ► Collaboration providers who provide services such as e-mail or instant messaging
Account Access	Provide customers with around-the-clock account access to their account information.	► Online brokerage trading apps ► Telephone company account manager functions ► Bank, credit card, and insurance company online apps
Trading Exchange	Allows buyers and sellers to trade goods and services on a public site.	► Buyer's side - interaction between buyer's procurement system and commerce functions of e-Marketplace ► Seller's side - interaction between the procurement functions of the e-Marketplace and its suppliers
Sell-Side Hub (Supplier)	The seller owns the e-Marketplace and uses it as a vehicle to sell goods and services on the Web.	► www.carmax.com (car purchase)

Composite patterns	Description	Examples
Buy-Side Hub (Purchaser)	The buyer of the goods owns the e-Marketplace and uses it as a vehicle to leverage the buying or procurement budget in soliciting the best deals for goods and services from prospective sellers across the Web.	www.wre.org (WorldWide Retail Exchange)

The makeup of these patterns is variable in that there will be basic patterns present for each type, but the Composite can easily be extended to meet additional criteria. For more information about Composite patterns, refer to *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, et al., ISBN 1931182027.

Selecting Patterns and product mapping

After the appropriate Business pattern is identified, the next step is to define the high-level logical components that make up the solution and how these components interact. This is known as the Application pattern. A Business pattern will usually have multiple Application patterns identified that describe the possible logical components and their interactions. For example, an Application pattern may have logical components that describe a presentation tier for interacting with users, a Web application tier, and a back-end application tier.

The Application pattern requires an underpinning of middleware that is expressed as one or more Runtime pattern. Runtime patterns define functional nodes that represent middleware functions that must be performed.

After a Runtime pattern has been identified, the next logical step is to determine the actual product and platform to use for each node. Patterns for e-business have product mappings that correlate to the Runtime patterns, describing actual products that have been used to build an e-business solution for this situation.

Finally, guidelines assist you in creating the application using best practices that have been identified through experience.

For more information about determining how to select each of the layered assets, refer to the Patterns for e-business Web site at:

`http://www.ibm.com/developerWorks/patterns/`

4.3 Summary

This chapter has provided an overview of Patterns for e-business. Patterns for e-business is a process for looking at a problem space and requirements and applying proven Business, Integration, Application, and Runtime patterns to design a solution.

The next chapter investigates how these patterns can be applied to a branch banking environment.

5

Applying IBM Patterns for e-business to branch banking

The previous chapter, IBM Patterns for e-business overview, establishes the overall context of Patterns for e-business. This chapter builds on that background information by exploring:

- How Patterns for e-business apply to the transformation of banking at the branch and enterprise levels
- How Linux can be used to implement Patterns for e-business based solutions

Linux and the Patterns for e-business can be used in defining Linux-based banking solutions in two contexts:

- At the branch bank level: Branch banks face near-term issues for which Patterns for e-business provide alternatives that fit within an existing environment while positioning the bank for the future.
- At the enterprise level: Patterns for e-business provide a context and road map for transforming a bank's enterprise architecture and show where Linux can play a key role.

5.1 Starting to use Patterns for e-business

Patterns for e-business let us describe both the problem or problems we want to solve and the solution or solutions that can be used to address those problems. For example, Business patterns let us characterize the types of problems we want to solve in simple but powerful terms. After we have described the scope and nature of the problem or problems we want to solve, Patterns for e-business leads us to solution alternatives in the form of Application and Runtime patterns. These patterns can be implemented through the use of well-known combinations of technology components (the Product mapping alternatives that are associated with Runtime patterns). At each point, we will need to make decisions about which pattern alternatives are the best choice for the situation, and for that, we need some key information as input to the decisions.

5.1.1 Describing the problem or problems we want to solve

The first step in applying Patterns for e-business is to establish which Business or Integration patterns, or both, are a good fit, in other words, which ones best represent the problem or problems we are trying to solve. To make that choice, we need to look at the *problem space*, identifying key business and IT requirements. Those requirements will let us make decisions about which Business and Integration patterns apply.

The Business and Integration patterns describe common characteristics that are found in a broad range of business processes across multiple industries. One of the goals in defining these patterns was to apply an 80/20 rule: Assure that these patterns could be applied to a majority of business processes in order to address a majority of a business's needs. Because of this, we can be confident of finding a good match for a majority of banking business process scenarios.

5.1.2 The solution or solutions alternatives

After we have described the problem space and determined which Business and Integration patterns apply, we will be ready to start looking at solution alternatives, in the form of Application patterns, for each of the Business and Integration patterns we have selected. The solution alternatives described in Patterns for e-business also identify technology component Product mappings that can be used to implement instances of the Runtime patterns. Once again, we have some decisions to make, and we will need to collect and summarize some key information as input into those decisions.

The Patterns for e-business Application patterns, Runtime patterns, and Product mappings are based on frequently recurring themes found in the functional and operational aspects of successful system architectures. In most cases, these patterns are reflected in a large number of successfully implemented systems. In some cases, we will also find what are referred to as *emerging* patterns that are considered valuable alternatives, but may be too recent to have been widely implemented yet. The key difference between the standard patterns and emerging patterns is the number of times the pattern has been implemented in the field. Although emerging patterns have been used less frequently than standard patterns, and therefore cannot be considered mainstream patterns yet, their capabilities and initial track record make them likely candidates to move beyond the early adopter stage and into the mainstream solution pattern category.

5.1.3 Gathering and summarizing requirements and drivers

We can look for the information we need for our pattern selection decisions by focusing on the following areas where we can gather customer requirements:

- Business context

  This area of focus looks at user profiles and business processes so that we can summarize customer requirements and identify key business drivers, both of which are important in determining which Business patterns apply. The business drivers will also help us evaluate Application pattern alternatives.

- User interactions

  This area of focus looks at how users interact with each other, another key to identifying which Business patterns apply.

- IT context

  This area of focus looks at application and technology portfolios to establish a high-level context at the branch and enterprise levels. This will identify any significant technical considerations that could influence the solution and identify key IT drivers. The IT drivers will also help us in a future step when we evaluate Application pattern alternatives.

- Interactions among users and components in the IT systems

  This area of focus looks at the ways in which users make use of IT services and how components that provide those services are deployed and interact with each other (and with external systems). This helps identify which Business and Integration patterns apply.

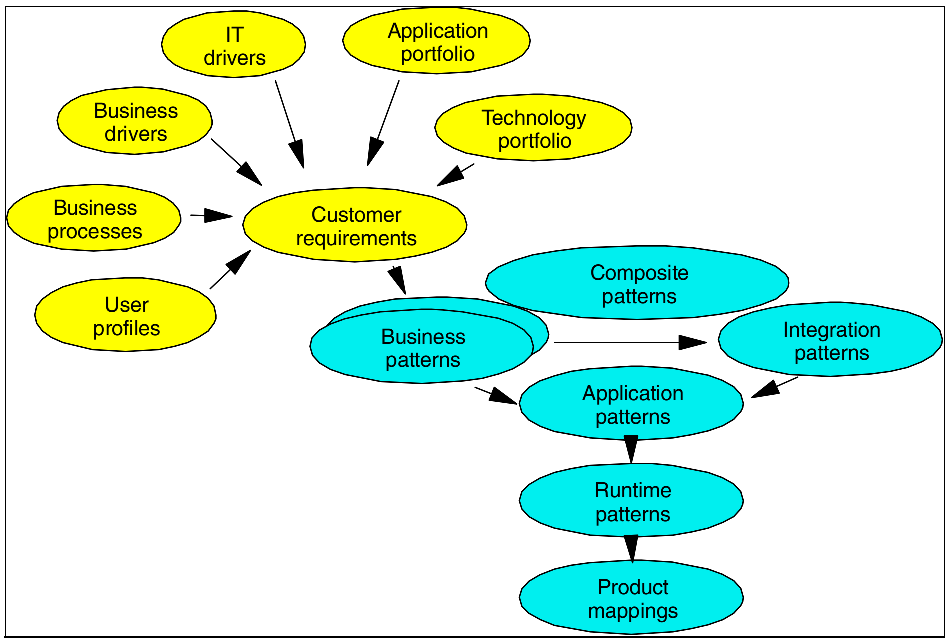


Figure 5-1   Some important sources for customer requirements

The customer requirements that are found in these focus areas define and establish the scope of the problem or problems we want to solve and lead us to conclusions about which Business and Integration patterns apply. If multiple patterns apply at this level, we could end up with a Composite pattern or Custom design (we will look more closely at this later in this chapter). Figure 5-2 on page 77 shows the relationships between Business and Integration patterns. After we have taken a look at our requirements, we will modify this figure to highlight the patterns we have that best describe our specific situation.

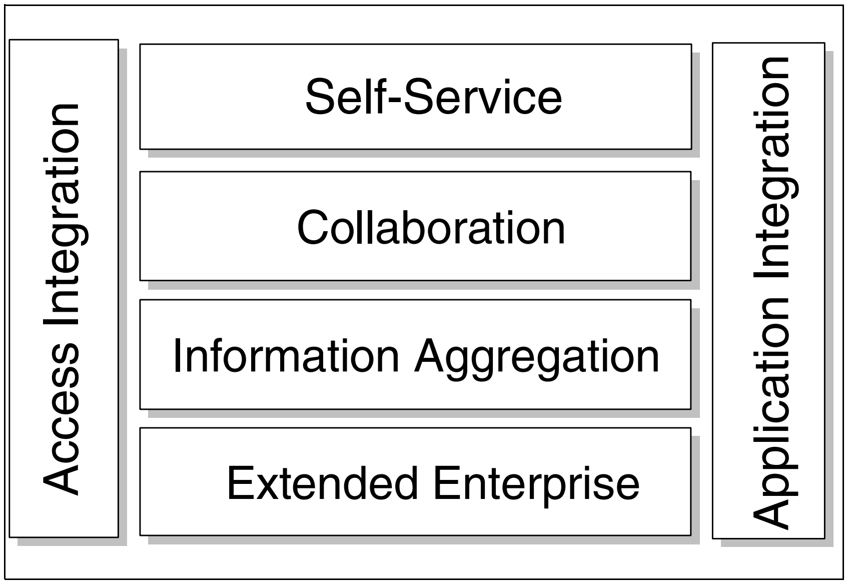


Figure 5-2 Business and Integration pattern relationships

After we have described the problems we want to solve, establishing which Business and Integration patterns apply (as well as some possible Composite patterns or Custom designs), we can move on to the patterns that describe solutions: the Application and Runtime patterns. The business and IT drivers that were collected during our analysis of customer requirements (as described in Chapter 3, “Branch banking requirements” on page 45) will be a key input to this next decision making process.

5.2 Business context

A wide variety of types of users appear as we study banking scenarios. Some examples are as follows:

- Customer
- Branch manager
- Loan officer
- Teller
- Vault supervisor
- Administrative assistant

We can see that *both internal and external users* make direct use of a bank’s automated business services. For external users (typically a Business-to-Customer, B2C, scenario), this could take the form of a customer using an ATM, using a bank's Internet site, or placing a call to a customer service representative or voice response unit (VRU) to inquire about their account status. In the future, a bank might also want to make these services available through a mobile hand held device. For internal users (a Business-to-Employee, B2E, scenario), this could take the form of a teller using a terminal or workstation during face-to-face transactions with a customer, or a loan officer entering a loan application using their desktop workstation.

These interactions tell us that there are a number of *self-service* scenarios where a user can directly access automated IT services. We can also see that it is highly desirable for these IT services to be made available through multiple channels or modes of delivery:

- Desktop workstations
- ATMs or kiosks
- Home PC browser
- IVR unit
- Call center

Of special interest to banks is the need to support the mobility of users, while providing a personalized and secure set of IT services. For example, some branch employees need to be able to perform their work from any of a number of workstation locations. At whichever location they find themselves, they need a specific, tailored set of application services presented through a consistent user interface.

This is often accomplished today through the use of technology that logs a user in and loads a user-specific desktop onto their workstation. This approach supports the use of a range of application architectures, from stand-alone and thick client application styles, to multi-tier distributed thin client topologies. As older applications are replaced by a predominance of server-centric applications, the need to provide a heavy-weight client machine will decrease, and along with it, the need to deploy applications to the client desktop. That trend will make it increasingly easier to provide application services at a wider range of locations and through a wider range of devices.

In addition to the self-service types of interactions we have described, there are also many examples of collaboration-style interactions within a branch banking environment. For example, a large or unusual withdrawal may require approval from the branch manager. Similarly, opening a new account may require collaboration between the account representative and credit checking personnel.

To summarize, one of the generalizations we can make is that we have users that are both internal and external to the organization. We also see that users interact with automated IT services through a number of channels (desktops, ATMs, IVRs, call centers). We also see that users interact with each other in face-to-face situations, and also through IT services that support both immediate and deferred collaboration.

5.3 IT context

Financial institutions span a broad range of sizes in terms of both their customer base and geographic distribution. This leads to a range of enterprise infrastructure topologies, some of which are more centralized, while others are more distributed at a regional and branch level. One repeating theme is that the bank branch continues to play a key role. The importance of the customer facing business processes carried out at bank branches means that the underlying IT services need to be as available as possible so that key business transactions can take place when the customer is there at the branch.

Here are some key IT requirements and drivers:

- IT services that need to be available through multiple delivery channels
- Flexible support for mobile or roaming users
- Access security controls
- Personalized presentation of IT services
- Need to accommodate varying network infrastructure capabilities
- Need to maximize the availability of services, in some cases providing a period of reduced services as an alternative to a period of no services
- Emphasize application and technology strategies that minimize the total cost of ownership
- Emphasize application and technology strategies that reduce application complexity and maximize the inter-operability and reuses of application services
- Emphasize application and technology strategies that provide flexibility in deploying IT services across a range of distributed or centralized topologies

The distributed nature of branch banking adds special challenges and influences the architectural decisions made for key banking applications. One important aspect of this is the need to provide several levels of autonomy so that if one component of the system is not available, other components can continue to operate and deliver at least some level of function. One example of this is the need to allow an ATM to dispense cash up to a pre-set limit (but not perform a balance inquiry) even if the enterprise systems it integrates with are not available, or to take a deposit at a teller station even if the branch server is temporarily offline.

The architecture of a bank's IT systems will benefit from being flexible and adaptable in terms of how its application and technology components are deployed at a branch, regional, or enterprise level to match the characteristics of its organization and infrastructure capabilities.

5.3.1 Application and technology portfolio

A wide range of applications are in use today, and we are likely to find a variety of application topologies and user interface styles:

- Monolithic stand-alone applications that run on a client desktop (for example productivity applications, such as word processors and spreadsheets).
- Applications with components that have been partitioned and distributed, operating at a branch level, in some cases integrating with enterprise applications. These are traditional two-tier client/server applications that have significant application components deployed on the client desktop which interact with other application components on a local server, or a database on a local server.
- Traditional host-based applications accessed through a terminal style interface, either on a dedicated terminal device, or through a terminal emulator that is loaded onto a client desktop.

Individual applications may also be difficult to integrate and share data with the following:

- Individual applications may have been implemented to support a specific business process and may not be designed to interoperate. Wherever that characteristic exists, there is a lost opportunity to combine the underlying application services in a new way to meet new competitive situations.
- The information underlying these applications may be stored in separate, independent data stores. This can limit the bank's ability to exploit valuable information across product lines, making it more difficult to manage a complex portfolio or to identity worthwhile cross-selling opportunities.

Cases where applications or their data cannot be easily integrated or combined imposes limits on the speed with which a bank can implement new or enhanced services. This affects a bank's competitive position.

Branch clients

Some of these application characteristics, combined with the need to support user mobility, leads to the use of technology that lets a user's desktop follow them to whichever client machine they are working at. This can involve loading that machine from the operating system on up to the applications, based on a user's role and preferences.

The range of application topologies requires a robust client machine, which imposes limits on the ability to deliver IT services through a wider array of channels. This can become a significant inhibitor to extending IT services to customers over the Internet.

Branch servers

Branch servers have evolved capabilities to provide a personalized locked down desktop, as well as host server-side application components and databases, host connectivity, and traditional file and print services. These features are aimed at the need to support a robust client desktop, as well as traditional two-tier client/server applications. Over time, some of these capabilities must be retained, such as the ability to provide a personalized, secured interface to application services, while others can be shed, such as the capability of remotely loading an operating system and applications to a client desktop (as application architecture evolves to be client independent).

5.3.2 Transformation strategies

Over time, a bank transformation strategy is aimed at increasing the range of integrated application services available to users, while also increasing the range of channels through which these services are delivered. To accomplish this, application architecture and the underlying technology infrastructure architecture need to follow a strategy to evolve away from dependence on a specific client or server configuration and deployment topology and toward consistent adoption of flexible, open standards and alternatives.

Bank transformation will most often take place in stages, with each step delivering specific short-term benefits, while being aligned to support a longer-term strategy. While the first steps are likely to be driven by urgent near-term pressures, they need to be viewed in the context of the overall transformation strategy. Patterns for e-business are particularly useful in accelerating the identification of both near-term and strategic solutions, which increases the likelihood that short-term, as well as long-term, objectives can be met.

5.4 Pattern selection

Reviewing the customer requirements will lead us to specific Business and Integration patterns. For a business as complex as we find in the banking industry, many patterns are likely to apply, resulting in a Composite pattern. Business and IT drivers will then lead us to specific Application and Runtime patterns.

5.4.1 Selecting Business and Integration patterns

Patterns for e-business *Business patterns* fall into four categories. Each of these covers generalized characteristics of common business process requirements. Let's see where bank transformation and business process requirements apply in this area:

- **Self-Service**: The user-to-business interactions we have found, such as customers using an ATM, or a teller using a teller workstation, indicate that the solution involves the Self-Service business pattern. At the branch level, the ability of a bank employee to move from one location to another and have their personal set of capabilities follow them is an important factor in maximizing efficiency. Over time, customers and bank employees will more than likely have an increasing number of automated business processes made directly available to them in a self-service mode, making this Business pattern of increasing importance.
- **Collaboration**: User-to-user interactions can be found in business processes ranging from opening an account, loan origination, customer inquiries, and large withdrawals. For example, a branch manager may need to authorize a withdrawal that is more than some specified amount. This indicates that the Collaboration business pattern applies. Because user-to-user collaboration can improve the efficiency of an organization, as well as increase the number of opportunities for the bank to interact with their customer base, it is likely that the number of collaboration scenarios and the importance of this pattern will increase.
- **Information Aggregation**: Business processes, such as portfolio management, cross-sell marketing, and business performance measurement, require the collection and analysis of information from multiple sources. This indicates that the Information Aggregation pattern will apply. As more and more information is aggregated, new opportunities to exploit this valuable resource are likely to be found, leading to the need for increasingly sophisticated approaches to information aggregation.

- **Extended Enterprise**: Some business processes cross enterprise boundaries. Collecting credit history during loan origination, transactions involving securities, and fund transfers are examples of business processes that can involve other business partners. This indicates that the Extended Enterprise business pattern applies. This is also an area that is likely to increase in importance as banks adopt more *e-banking*. For example, e-banking provides integrated management of customer-related data, which had traditionally been managed by a number of separate systems, in its customer information system. Therefore, related departments can share and analyze accurate customer data. Furthermore, the analytic CRM system provides market and customer trend analysis to allow the bank to conduct more effective, proactive marketing activities.

Integration patterns fall into two categories: *Access Integration* and *Application Integration*. Each of these basic patterns can be subdivided into more discrete collections of services or characteristics. Let's see where our requirements lead us in this area.

The *Access Integration* pattern is subdivided into four areas of common services:

- **Device Support services**: When we considered the Self-Service business pattern, we discovered that it is essential for a bank to be able to provide its services through multiple channels. This increases the number of touch points the bank can have with its customers, as well as providing flexibility in delivering services to its employees in a variety of contexts. The Access Integration pattern Device Support services can be used to provide or extend a bank's multichannel capabilities.
- **Presentation services**: Providing a common look and feel is a key to achieving high employee productivity, as well as in establishing a bank's brand image with its customers. Both of these are important requirements that can be met through the Access Integration pattern Presentation services.
- **Personalization services**: Bank employees often need the ability to move from location to location as they carry out their work. The Access Integration Personalization services addresses the need to have a user's personal set of capabilities follow them from one desktop to another.
- **Security and Administration**: Security is one of the most important requirements in a banking environment. The Access Integration Security and Administration services supports a need to define and administer role-based policies and to have those policies applied as a user moves from access point to access point.

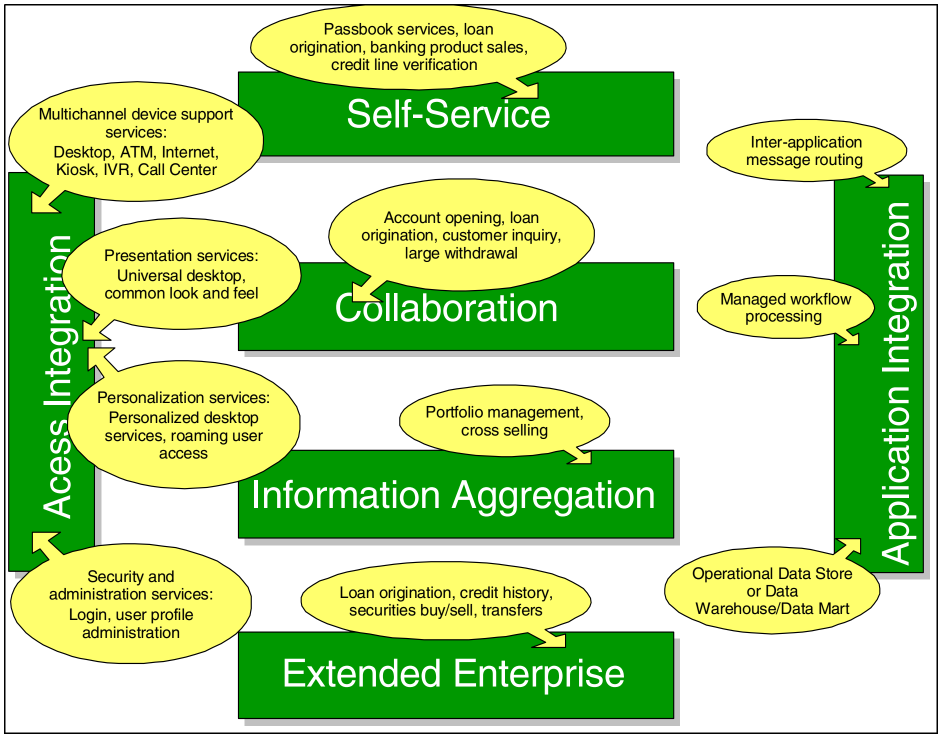
The *Application Integration* pattern can be viewed in several dimensions. One of the key aspects is whether the integration is aimed at invoking application processes or accessing data. Another aspect pertains to whether the application-to-application interactions are synchronous or asynchronous. Still another important aspect is whether the integration uses a point-to-point or multi-point topology. Lastly, application integration scenarios can be characterized in terms of message flow: do messages simply pass through an integration point, does message transformation or enrichment take place, or are messages being routed. Let’s see where process-focused and data-oriented Application patterns apply in this area:

- **Process-focused**: The common services provided by process-focused Application patterns are protocol adapter, message handler, data transformation, decomposition/recomposition, routing/navigation, state management, security, local business logic, and business unit of work management. The five process-focused Application patterns are Direct Connection, Aggregator, Transactional, Broker, and Managed Process.
- **Data-oriented**: Data-oriented Application patterns can be characterized based on topology (centralized or decentralized), as well as by database affinity (homogeneous/same vendor, multi-vendor relational/all ODBC or JDBC, heterogeneous structured/some non-relational but all have a structured layout, structured with non-structured/integrating both structured database content with unstructured content). The four data-focused Application patterns are Propagation, Replication, Operational Data Store (ODS), and Federated Repository.

One thing that is clear from our analysis of the bank transformation and business process requirements is that a bank’s individual IT services need to be integrated in ways that support multichannel access and simplified and consistent user interfaces. When we consider a bank’s need to interact with its business partners, we see additional points at which applications need to integrate, and in this case, integration occurs across enterprise boundaries. The bottom line is that the Application Integration pattern clearly applies to our solution. Deciding which of these patterns and pattern dimensions apply requires consideration of both business and technical requirements. We highlight specific cases of application integration in the following sections.

5.4.2 Composite pattern

We can see that each of the Business and Integration patterns applies in some way to bank transformation. Figure 5-3 on page 85 highlights requirements that led us to select each of these patterns.



Passbook services, loan origination, banking product sales, credit line verification
Self-Service
Multichannel device support services: Desktop, ATM, Internet, Kiosk, IVR, Call Center
Inter-application message routing
Account opening, loan origination, customer inquiry, large withdrawal
Presentation services: Universal desktop, common look and feel
Collaboration
Managed workflow processing
Acess Integration
Application Integration
Personalization services: Personalized desktop services, roaming user access
Portfolio management, cross selling
Information Aggregation
Security and administration services: Login, user profile administration
Loan origination, credit history, securities buy/sell, transfers
Operational Data Store or Data Warehouse/Data Mart
Extended Enterprise

Figure 5-3 Where Business and Integration patterns apply

Our resulting Composite pattern at this level is shown in Figure 5-4 on page 86.

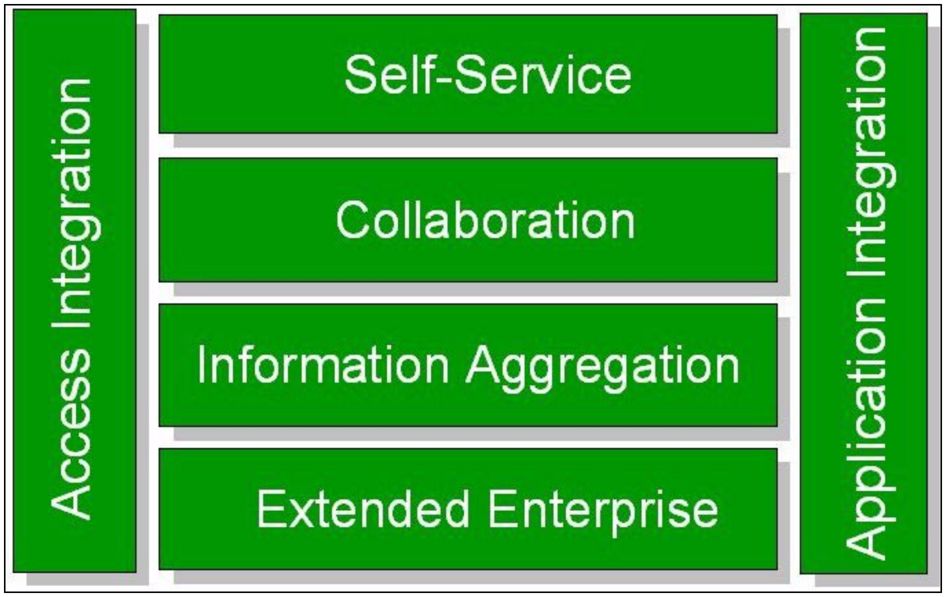


Figure 5-4 All Business and Integration patterns apply

At the end of this chapter, we look at customer loyalty and e-banking scenarios where a Composite pattern can be applied.

Before we look at the Composite patterns scenarios, we want to identify suitable Application and Runtime patterns and decide how they can be applied to our solution. Because we have a special focus on the branch bank in this redbook, we will look most closely at the Self-Service business pattern and the associated Application and Runtime patterns. The Self-Service business pattern also provides a springboard for a discussion of the role of Access and Application Integration patterns.

5.4.3 Application and Runtime patterns

Under each of the Business and Integration patterns we have identified is a group of Application patterns. The Application patterns describe generalized logical application topologies: the “shape” of applications that address a Business or Integration pattern’s requirements.

Business and IT drivers are a key input to selecting Application patterns. As described in 2.2, “Branch technology challenges” on page 14, the key challenges to be addressed include:

- Speed to market: The adaptability of a bank’s solutions are key to being positioned to respond to competitive pressures and changing marketplace demands. Adaptation could take the form of rapid reconfiguration, or rapid deployment of new or enhanced applications.
- Total cost of ownership (TCO): Branch transformations must be affordable; a key consideration is to establish a plan for when specific components will be replaced: which components must be replaced in the initial step, which are more easily retained, addressing the expenditure rate.
- Quality of Service (expressed in terms of non-functional requirements: security, availability, reliability, performance, scalability, so these are really IT drivers).

Self-Service application and runtime patterns

There are seven Self-Service application patterns, some of which could be used to provide immediate solutions, while others support longer term objectives:

- Stand-Alone Single Channel
- Directly Integrated Single Channel
- As-Is Host
- Customized Presentation to Host
- Router
- Decomposition
- Agent

The following sections discuss each of these Self-Service application patterns and identifies Runtime patterns that can be used to implement them

Single Channel, limited integration patterns

The Stand-Alone Single Channel application pattern is a single channel pattern that is useful when there is no immediate need to integrate the application with other systems, and where the time-to-market driver is especially significant. The Directly Integrated Single Channel application pattern is another single channel pattern, but supports point-to-point integration with other systems. This pattern can be used to reduce the latency of business events and maximize the reuse of legacy systems. This pattern can also represent an evolutionary future step for the Stand-Alone Single Channel pattern.

The As-Is Host application pattern extends the reach of legacy applications that use a terminal-style user interface to new channels. This pattern allows users to access legacy applications through a thin client browser instead of a specialized terminal device or a heavier weight desktop terminal emulator. The Customized

Presentation to Host application pattern can be used to provide a new user interface for legacy applications. Multiple legacy applications can be given a new, more efficient, and consistent user interface, as well as making the legacy applications available through a thin client browser.

Multichannel, complex integration patterns

To address multichannel and complex integration scenarios, we look to a second group of Self-Service application patterns. The Router application pattern is focused on addressing multichannel delivery of services. It accomplishes this by providing a standard set of application services that can be requested from one of a number of different delivery channels, such as browser, Kiosk, IVR, or call center. A request from any channel is routed to the appropriate application system for processing. This approach eliminates the need for point-to-point integration for each individual channel.

The Decomposition application pattern builds on the Router pattern by adding the capability of being able to break individual requests from a channel into multiple requests to enterprise applications, and then recombining the individual responses into a single response back to the originating channel. This enables a more sophisticated set of services that combine the capabilities of multiple applications.

The Agent application pattern further extends the Router and Decomposition patterns by adding its own intelligence to the responses from multiple enterprise applications. It can transform a group of enterprise applications that are not capable of directly integrating with each other into what could be considered a new federated application that is more than the sum of the individual application services.

The Agent pattern can, for example, evaluate information returned from multiple enterprise applications and apply business rules to arrive at a cross selling recommendation that no individual application would be able to make.

Stand-Alone Single Channel pattern

In this section, we describe the Stand-Alone Single Channel application and runtime patterns.

Application pattern

Figure 5-5 on page 89 illustrates the Stand-Alone Single Channel application pattern.

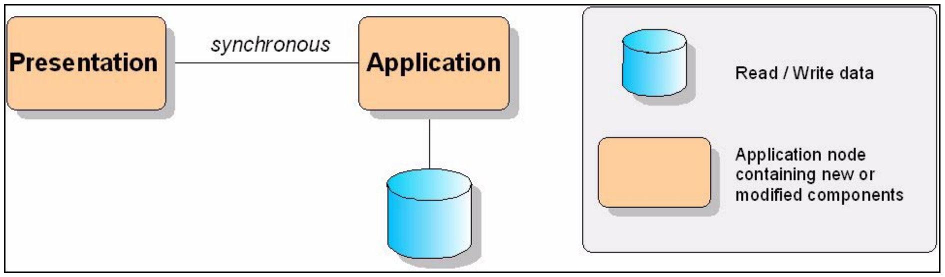


Figure 5-5 Stand-Alone Single Channel application pattern

This pattern typifies the separation of presentation services from application services. In an e-business context, this is often implemented through very sparing use of presentation logic on the client.

For situations where there is an urgent time-to-market pressure to deliver application services through new access channels, this application is one of the fastest way to implement new services. One of the key features of this pattern is that it is not dependant on integration with other systems, which eliminates what can be one of the more challenging aspects of integrating new systems. This also means that the application services could be deployed to servers at a branch, region, or enterprise level, depending on the bank's specific requirements.

In a bank transformation context, this pattern could be used to move *monolithic applications* that currently can only run on a robust client desktop with a specific operating configuration to a thin client, server-oriented architecture. This would not only reduce the need for high-powered desktop hardware, it could also extend the reach of the application to any device with a standard Internet browser capability. This pattern could be implemented in several ways, depending on the type of application under consideration.

For applications that are *stand-alone applications*, such as desktop productivity applications (for example, a word processor), the application could be hosted on a central server using technology, such as Citrix Winframe or Connectix VirtualServer. In this case, the application runs on a server instead of the desktop. The presentation of the application's user interface is handled through a thin client component, which is much less dependant on its operating environment than the application itself, thus extending the reach of the application to a wider range of devices and users. This approach can also be used for client/server applications where there is a thick client component, and where there is no time or interest in migrating the application to a multitier e-business based architecture.

For traditional *two-tier applications,* where there is an interest in migrating to a multitier, e-business based architecture, the Stand-Alone Single Channel application pattern provides one option. As long as the application does not require integration with other applications, this pattern can be used.

Several benefits accrue from using this approach. First of all, the application can be re-architected using a Model View Controller (MVC) based application framework. By applying this separation of concerns approach to the application architecture, the application not only can be moved to an e-business environment, immediately making it available through additional device channels, but perhaps more importantly, the underlying application services can be made available through a more generalized interface, which positions the application for wider reuse and easier integration in the future.

Runtime pattern

Figure 5-6 illustrates the Stand-Alone Single Channel runtime pattern.

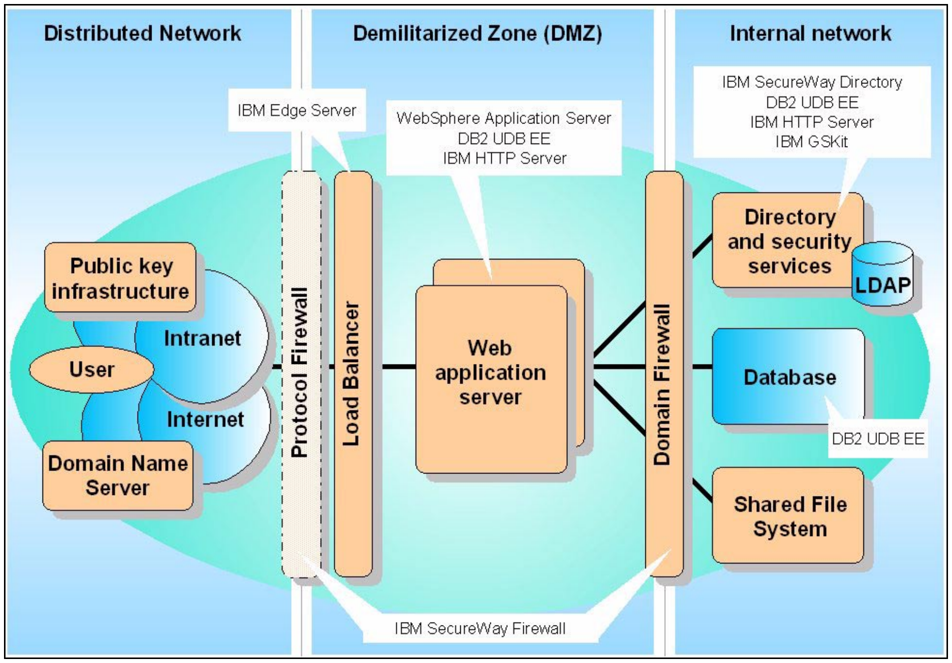


Figure 5-6 Stand-Alone Single Channel runtime pattern

The client node presentation services can be implemented through Java script, Java applet, or only HTML. On the server side, a JSP can be used to encapsulate any other dynamic presentation-related logic. Application logic that interprets requests from the browser and delegates the request to a business model layer and business logic components are housed in separate components, such as servlets, Java classes, or JavaBeans (in other, more sophisticated patterns, Enterprise JavaBeans could be used, along with various application integration techniques).

The *emerging* Runtime pattern shown in Figure 5-6 on page 90 could be used as the basis for implementing this approach and demonstrates how this Application pattern could be implemented on a Linux server platform. This pattern is considered emerging because some of the products have only recently become available on the Linux platform and have not yet achieved as wide use as they have on other platforms.

Table 5-1 Pattern summary: Stand-Alone Single Channel

Key characteristics	► Single channel at a time ► No integration with other applications
Commonly used technology components	► Linux server ► WebSphere Edge Server ► IBM Developer Kit for Linux ► WebSphere Application Server AE ► DB2 UDB EE ► IBM HTTP Server ► IBM Access Manager for e-business

Directly Integrated Single Channel pattern

In this section, we describe the Directly Integrated Single Channel application and runtime patterns.

Application pattern

Figure 5-7 on page 92 illustrates the Directly Integrated Single Channel application pattern.

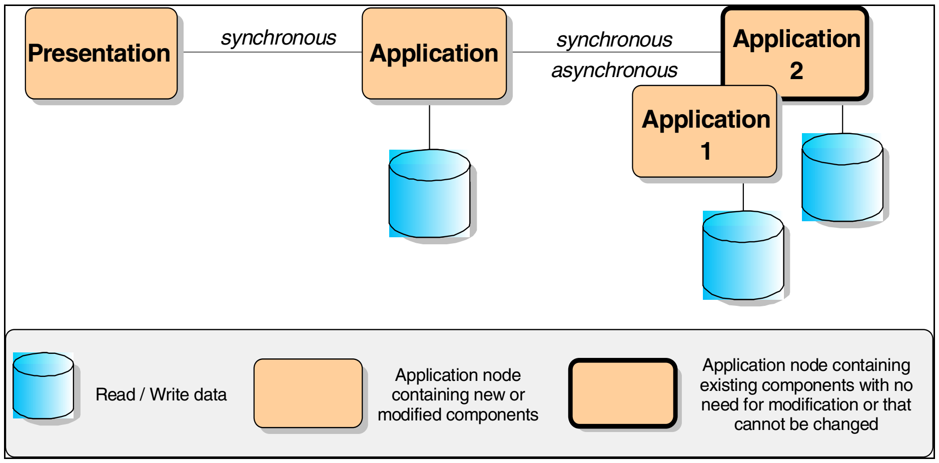

Presentation
synchronous
Application
synchronous
asynchronous
Application 2
Application 1
Read / Write data
Application node containing new or modified components
Application node containing existing components with no need for modification or that cannot be changed

Figure 5-7 Directly Integrated Single Channel application pattern

This Application pattern extends the Stand-Alone Single Channel pattern to provide an approach for applications that can use a *point-to-point connection* with one or more back-end applications (or data, or both). As a Single Channel pattern, it could be used to implement any one of the delivery channels.

The simplicity of this pattern is achieved by limiting the scope of the pattern to a single channel and to relatively simple integration techniques. While this makes the pattern easier to implement than some of the other patterns, the trade off is that it is also limiting, not being a multichannel solution (see “Router pattern” on page 98, “Decomposition pattern” on page 101, and “Agent pattern” on page 104), and usually does not interact with more than one enterprise application in any individual transaction (see “Decomposition pattern” on page 101 and “Agent pattern” on page 104). There may or may not be a need for modification of the back-end legacy applications depending on their current capabilities (note that there is one back-end application node that represents an unmodified application, and another node that represents an application that would be modified to support integration).

Runtime pattern

Figure 5-8 illustrates the Directly Integrated Single Channel runtime pattern.

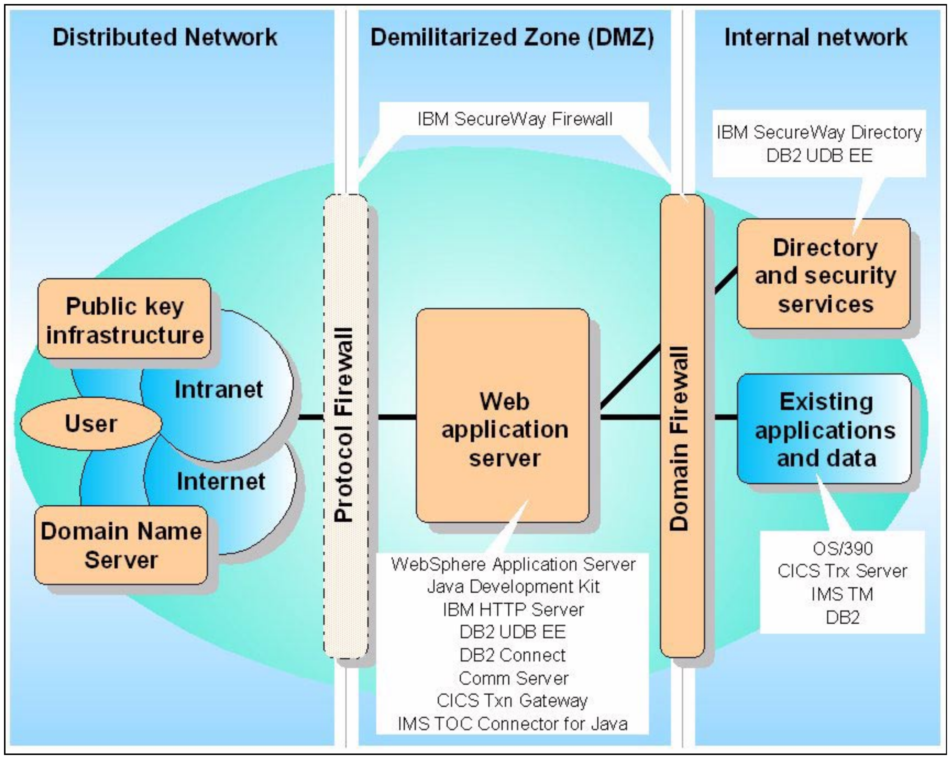


Figure 5-8 Directly Integrated Single Channel runtime pattern

This pattern adds *connector components* to enable integration between the e-business application and previously existing legacy applications. These connectors are *point-to-point*, meaning that request and response conversations between the e-business and legacy applications do not go through any intermediate dynamic routing or transformation nodes, resulting in a less complicated infrastructure architecture. Although a load balancing node is not explicitly shown in this pattern, one could easily be added (see Figure 5-6 on page 90).

Table 5-2 summarizes this pattern's key characteristics and technology components.

Table 5-2 Pattern summary: Directly Integrated Single Channel

Key characteristics	► Single channel ► Point-to-point integration with individual applications
Commonly used technology components	► Linux server ► WebSphere Edge Server ► IBM Developer Kit for Linux ► WebSphere Application Server AE ► IBM HTTP Server ► IBM Access Manager for e-business ► DB2 Connect Enterprise Edition for Linux ► eNetworks Communications Server ► CICS Transaction Gateway ► IMS Connector for Java (runtime classes)

As-Is Host and Customized Presentation to Host patterns

In this section, we describe the As-Is Host and Customized Presentation to Host application and runtime patterns.

Application patterns

Figure 5-9 and Figure 5-10 on page 95 illustrate the As-Is Host and Customized Presentation to Host application patterns.

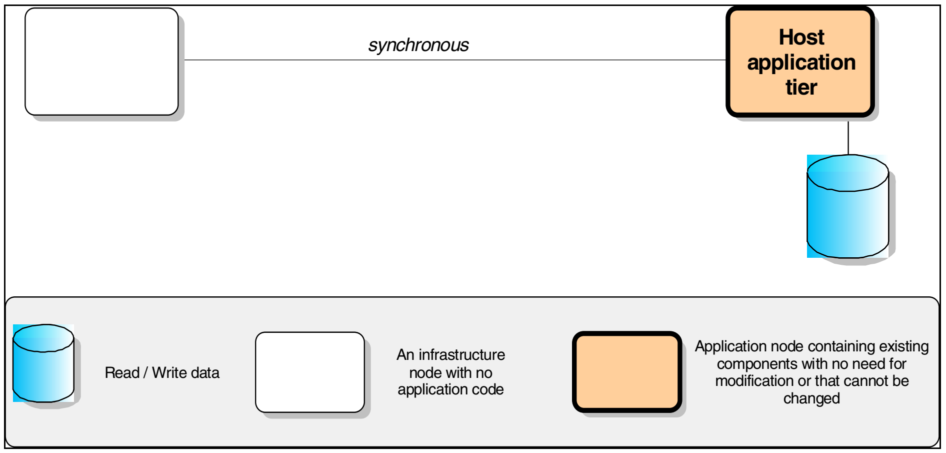

synchronous
Host application tier
Read / Write data
An infrastructure node with no application code
Application node containing existing components with no need for modification or that cannot be changed

Figure 5-9 As-Is Host application pattern

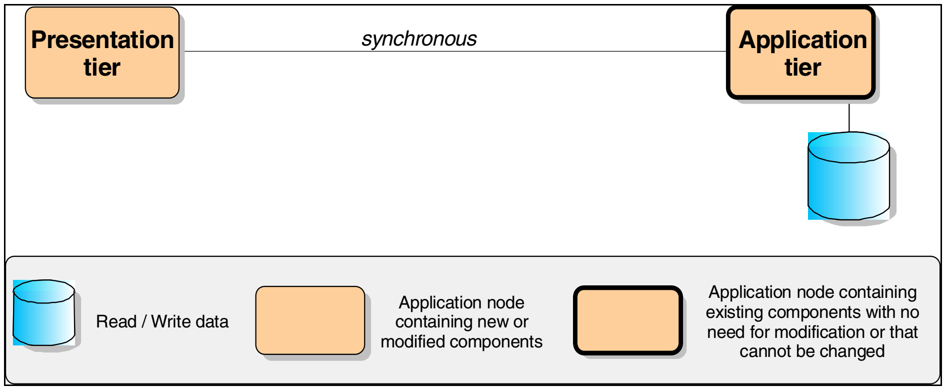

Presentation tier
synchronous
Application tier
Read / Write data
Application node containing new or modified components
Application node containing existing components with no need for modification or that cannot be changed

Figure 5-10 Customized Presentation to Host application pattern

These two patterns enable browser access to applications that have a terminal-oriented user interface, thus extending the reach of the applications to new locations and lighter weight client devices. These patterns work well in situations where the host application needs to be left unchanged (hence the application tier node with the bold border).

These two patterns represent two different solution approaches. The As-Is Host pattern provides the user with the original user interface design, while moving the terminal emulator function off of the client. The Customized Presentation to Host pattern can be used to provide a redesigned, more Web-like user interface, making it possible to simplify or streamline the user interface, and also making it possible to provide a more consistent user interface across a range of terminal oriented applications.

In the *As-Is Host pattern*, an infrastructure node with no application code provides terminal emulation functions on a server, presented through a browser, eliminating the need for the client to have a terminal emulator installed or running on it. Moving the terminal emulation function upstream from the client provides several benefits:

- The host application can be accessed from new devices, with simplified client node requirements. The ability to use a *zero footprint* client lowers the cost of the client node and simplifies its configuration.
- The host application can be used at new locations, for example, where browser access to the enterprise network is available, but a server with a host communications gateway component might not be.

- Placement of the server node that hosts the terminal emulation function becomes an easier decision than may have been the case for the communications gateway needed for a thick client-based terminal emulator. The presentation tier server could be located at a branch, region, or central location.

The *Customized Presentation to Host pattern* adds intelligence to the presentation tier to provide the capability to reformat host screens. Like the As- Is Host pattern, this pattern is useful when the host application is not going to be (or cannot be) modified and will be used as is. This pattern can be used to mask the complexity of host applications, and it can also be used to give multiple host applications a common look and feel. Key benefits from using this pattern are as follows:

- All of the benefits of the As-Is Host pattern also apply to this pattern.
- The underlying applications can take on a look and feel that is consistent with other Web-based applications.
- Multiple terminal-oriented applications that may have very different user interface designs can be given a consistent user interface design.
- The improved, more consistent user interface reduces training time and cost.
- The usability improvements can lead to increased productivity and reduced turnover.

Runtime pattern

The topology for these two patterns are similar to Figure 5-8 on page 93. However, instead of needing all of the connection technology components highlighted in that topology, we would use components that simplify the transformation of host screen data streams. The base technology components are as follows:

- A client that supports a Java-enabled browser: This could be a lightweight desktop running any operating system that supports the browser requirements.
- A Linux server running: WebSphere Application Server.
- The underlying implementation of the As-Is Host pattern is based on a server-hosted terminal emulation product: IBM WebSphere Host Publisher. (At the time of this writing, IBM WebSphere Host Publisher is not supported on Linux.)

- The underlying implementation of the Customized Presentation to Host pattern uses some form of screen scraping mechanism to transform the host screens into a more user friendly interface. The following off-the-shelf products are available that can significantly reduce the effort to implement this pattern:
  - Client Access Family V5R1
  - Host Access Transformation Server (HATS); 3270, 5250 through a browser with a Web-like look and feel

Table 5-3 summarizes this pattern's key characteristics and technology components.

Table 5-3 Pattern summary: As-Is Host, Customized Presentation to Host

Key characteristics	► Make terminal-oriented host applications available through a browser. ► As-Is Host pattern moves terminal emulations off the client, leaving the original user interface design unchanged. ► Customized Presentation to Host also moves terminal emulation off of the client, and in addition, transforms the original user interface design into a new user-friendly design.
Commonly used technology components	► Linux server. ► WebSphere Edge Server. ► IBM Developer Kit for Linux. ► Web Sphere Application Server AE. ► DB2 UDB EE. ► IBM HTTP Server. ► IBM Access Manager for e-business. ► IBM WebSphere Host Publisher (As-Is Host) (At the time of this writing, IBM WebSphere Host Publisher is not supported on Linux.) ► Host Access Transformation Server (HATS) (Customized Presentation to Host).

Router pattern

In this section, we describe the Router application and runtime patterns.

Application pattern

Figure 5-11 illustrates the Router application pattern.

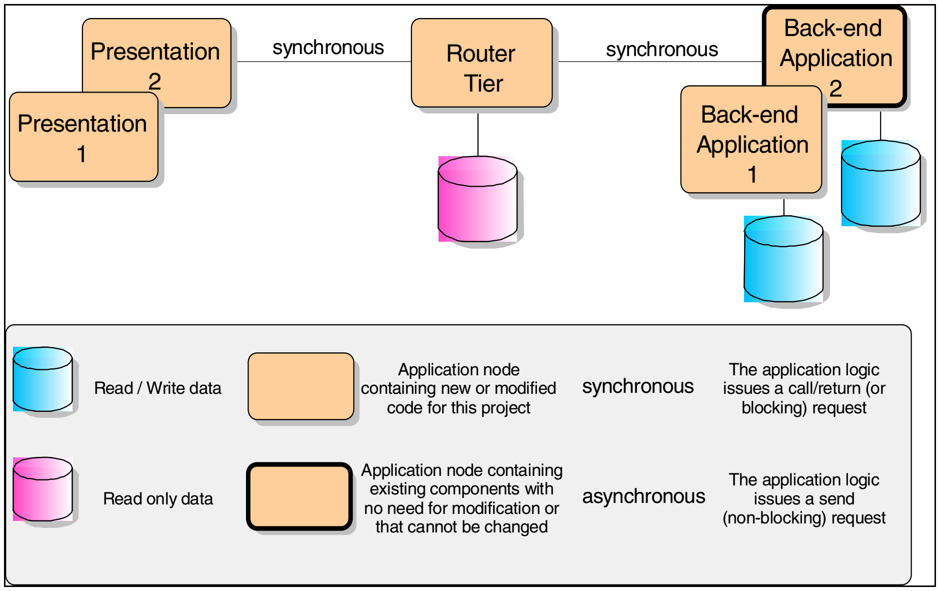

Presentation 2
synchronous
Router Tier
synchronous
Back-end Application 2
Presentation 1
Back-end Application 1
Read / Write data
Application node containing new or modified code for this project
synchronous
The application logic issues a call/return (or blocking) request
Read only data
Application node containing existing components with no need for modification or that cannot be changed
asynchronous
The application logic issues a send (non-blocking) request

Figure 5-11 Router application pattern

This Application pattern represents a fan out on the presentation side to support multiple channels (that is, uses a hub and spoke configuration instead of having each channel provide its own point-to-point integration with business applications). The multiple presentation nodes represent different access channels, such as a browser, IVR, or ATM.

The router tier supports multiple channels on the presentation side and can route requests from any of those channels to the appropriate enterprise application. The router does not fan requests out on the application side (as does the Decomposition pattern). Each incoming request is routed to one target application.

The synchronous connection on the right is logical, not physical, and the implementation could use a non-blocking messaging technique in addition to other synchronous techniques, in any case, on the presentation side, the interaction appears to the user to be synchronous. The router can perform protocol conversion (for example, HTTP to RMI) and session concentration, in addition to request routing. Router can also enforce role-based security rules.

The Router pattern also uses the Access Integration and Application Integration patterns.

An example of how the Router pattern can be used would be to support access to client account information through the Internet, kiosks, VRU, call center, and teller stations. A limitation of this pattern is that it provides a one product or account at a time view (see "Decomposition pattern" on page 101 and "Agent pattern" on page 104 for a consolidated view).

Runtime pattern

More than one runtime variation exists for this Application pattern. One approach is to use a message broker for integration. For example, WebSphere MQ can provide message transport between the router node and back-end applications (as well as between the router and channel specific applications). WebSphere MQ Integrator can provide intelligent routing and message transformation. Figure 5-12 shows a commonly used approach based on the WebSphere MQ product family.

Another emerging pattern approach is based on the user of an object request broker. In this case, the Integration Server node would be populated with WebSphere Application Server Enterprise Edition, and connectors, such as WebSphere MQ, to support integration with back-end applications.

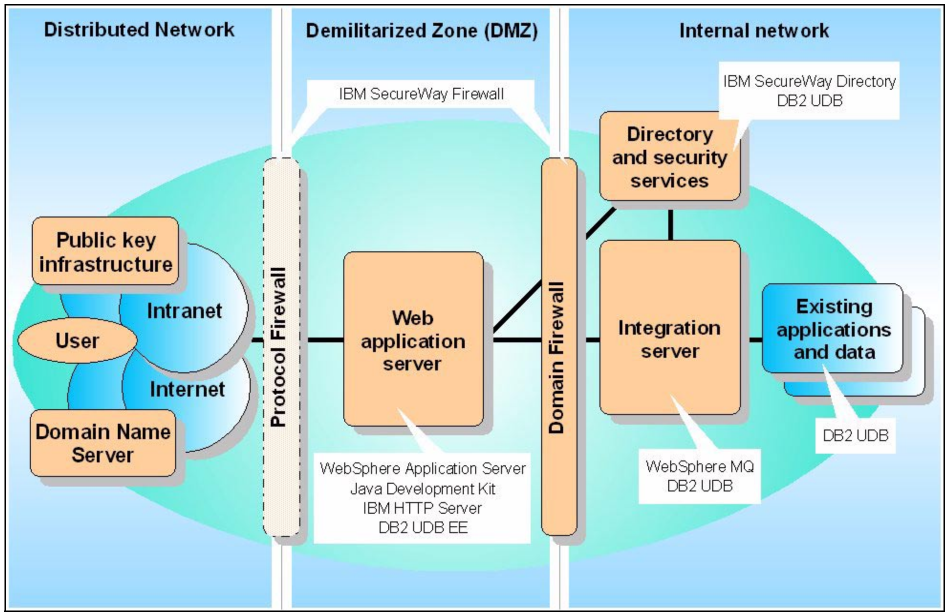


Figure 5-12 Router runtime pattern

Table 5-4 summarizes this pattern's key characteristics and technology components.

Table 5-4 Pattern summary: Router

Key characteristics	► Make application services available through multiple channels. ► Move to a hub and spoke topology that provides back-end application services to any channel without the channel needing to know details of the back-end application locations or message formats. ► Intelligent request routing from a channel to the appropriate application system. ► Message transformation of requests and responses. ► Session concentation: the router maintains a reduced set of sessions with back-end applications, instead of each channel needing to maintain its own set. ► Enforce channel- and role-based security policy. ► Eliminate the need for each channel to support dedicated point-to-point connections to back-end application systems.
Commonly used technology components (message broker based)	► Linux server ► WebSphere Edge Server (optional) ► IBM Developer Kit for Linux ► WebSphere Application Server AE ► DB2 UDB EE ► IBM HTTP Server ► IBM Access Manager for e-business ► WebSphere MQ ► WebSphere MQ Integrator ► WebSphere MQ classes for Java

Decomposition pattern

In this section, we describe the Decompoition application and runtime patterns.

Application pattern

Figure 5-13 illustrates the Decomposition application pattern.

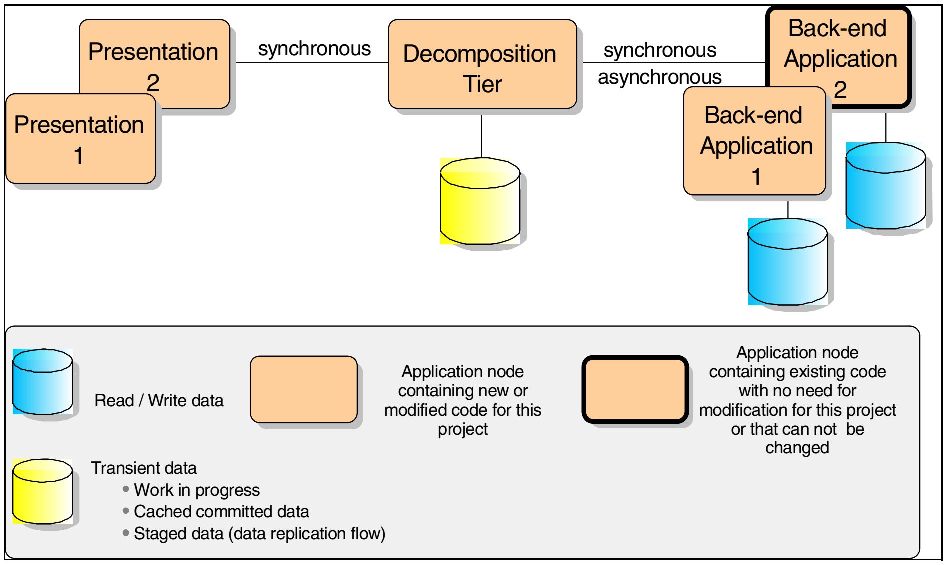

Presentation 2
Presentation 1
synchronous
Decomposition Tier
synchronous
asynchronous
Back-end Application 2
Back-end Application 1
Read / Write data
Application node containing new or modified code for this project
Application node containing existing code with no need for modification for this project or that can not be changed
Transient data
Work in progress
Cached committed data
Staged data (data replication flow)

Figure 5-13 Decomposition application pattern

The Decomposition application pattern extends the Router pattern to enable processing of more sophisticated requests. As with the Router pattern, this pattern can handle requests from multiple channels. It goes beyond the Router by being able to break down a request from a channel into multiple requests to back-end applications, and then assembles each back-end application response into a single response back to the original requester.

This pattern is based on the Router's hub and spoke architecture and extends it by adding intelligence to decompose then recompose requests and responses. One client request becomes multiple requests to individual enterprise applications (referred to as *fan out*), and each of the individual results are collected into a single response back to the client (referred to as *fan in*). A local work-in-progress database may be used to hold intermediate results that are needed to represent process state and to assemble the final response back to the channel, as well as transformation and routing business rules. Although the decomposition of an inbound message (fan out) requires intelligence in the integration server, an even more challenging aspect of this pattern is the need to collect, correlate, and assemble the individual responses (fan in).

An important benefit of this pattern is that a consistent consolidated customer view becomes available through any of multiple channels. Without the capabilities of this pattern, each channel would need the intelligence to know which back-end applications provide specific information and services. The Decomposition pattern concentrates this intelligence in the interrogation node, which greatly simplifies the design of individual channels.

Runtime pattern

Similar to the Router pattern, several Runtime patterns could be used to implement the Decomposition application pattern. Figure 5-15 on page 104 represents a commonly used approach based on the use of a message broker.

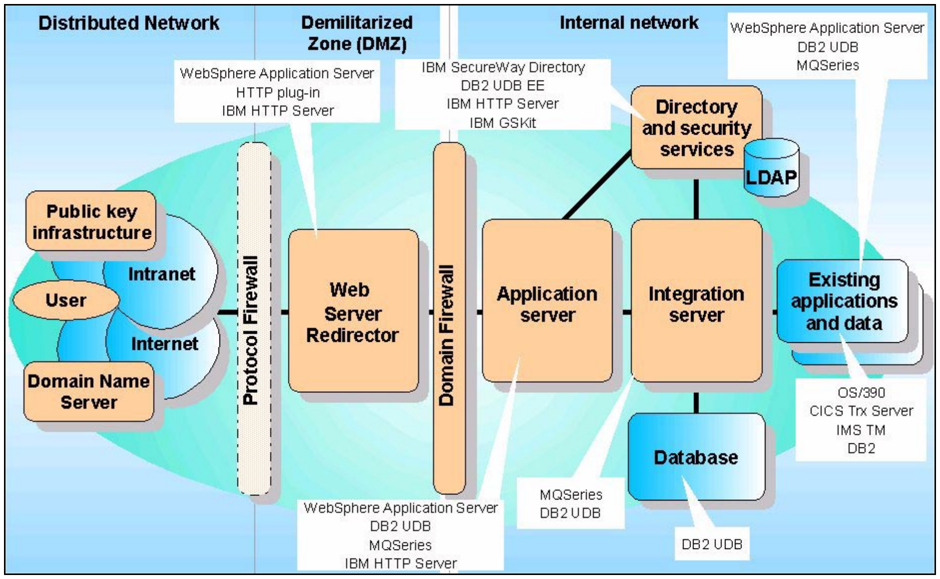


Figure 5-14 Decomposition runtime pattern

Table 5-5 on page 103 summarizes this pattern's key characteristics and technology components.

Table 5-5 Pattern summary: Decomposition

Key characteristics	► Make application services available through multiple channels. ► Move to a hub and spoke topology that provides back-end application services to any channel without the channel needing to know details of the back-end application locations or message formats. ► Intelligent request routing from a channel to the appropriate application system. ► Message transformation of requests and responses. ► Session concentation: The router maintains a reduced set of sessions with back-end applications, instead of each channel needing to maintain its own set. ► Enforce channel- and role-based security policy. ► Eliminate the need for each channel to support dedicated point-to-point connections to back-end application systems.
Commonly used technology components (message broker based)	► Linux server ► WebSphere Edge Server (optional) ► IBM Developer Kit for Linux ► WebSphere Application Server AE ► DB2 UDB EE ► IBM HTTP Server ► IBM Access Manager for e-business ► MA88 support pack (MQ JMS classes) ► WebSphere MQ ► WebSphere MQ Integrator ► IA72 Mixers Aggregator plug-in (fan out and fan in support) ► WebSphere MQ classes for Java ► WebSphere Application Server EE

Agent pattern

In this section, we describe the Agent application and runtime patterns.

Application pattern

Figure 5-15 illustrates the Agent application pattern.

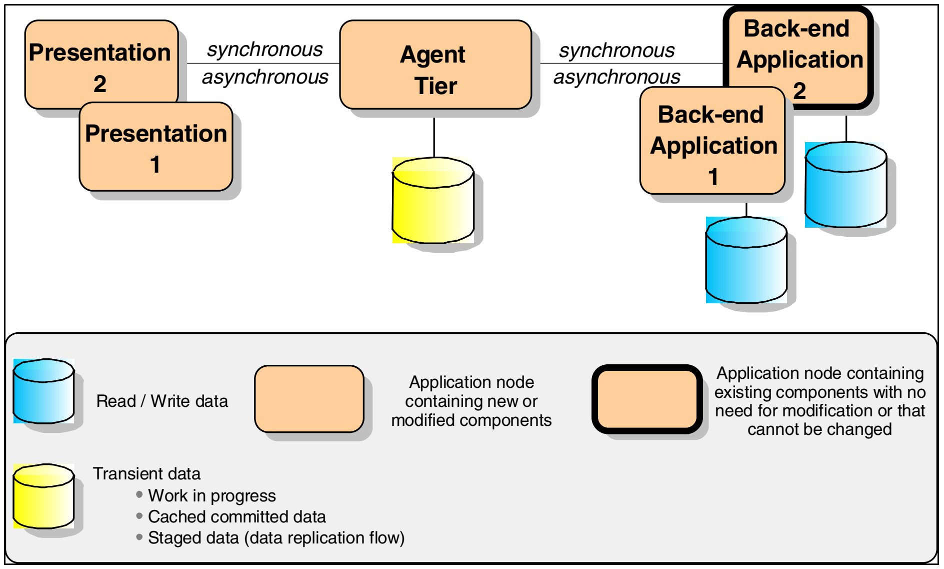


Figure 5-15 Agent application pattern

This Application pattern extends the Router and Decomposition patterns. It provides a unified, customer-centric view that can be exploited for mass customizing of services and for cross-selling purposes. The Agent pattern makes intelligent use of work-in-progress cached data to push content to users based on analysis of the data.

A variation of the Agent pattern uses an Operational Data Store (ODS) instead of relying on individual requests to return the required data.

Examples:

- E-commerce sites with targeted marketing
- Call center representatives being prompted with cross-selling scripts
- Work-in-progress database holding long running transaction data (for example, a loan application staged until submitted) before committing changes

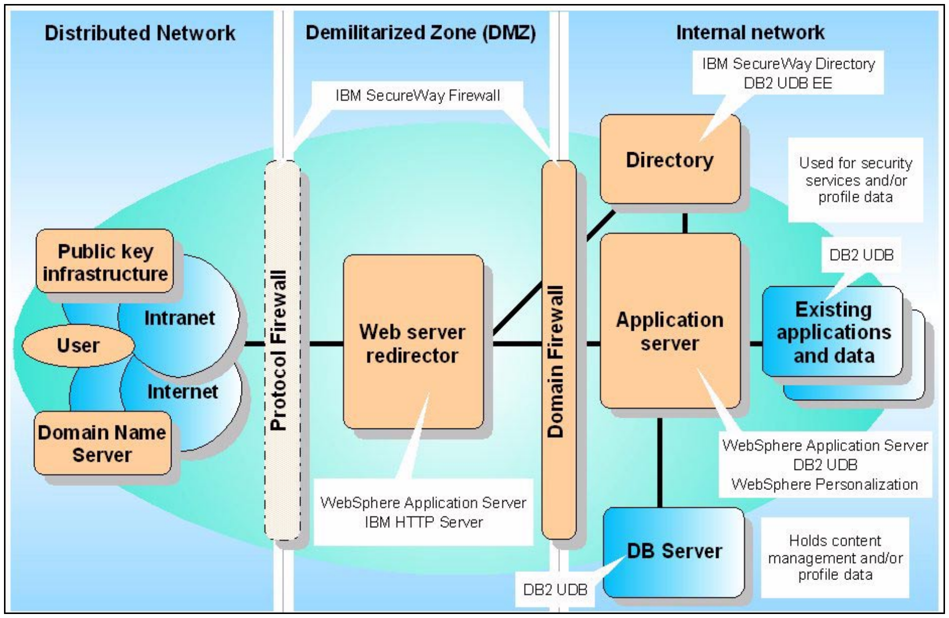


Figure 5-16 Agent runtime pattern

Products that can be deployed on Linux:

- See Stand-alone Single Channel for base products
- WebSphere Personalization Server for Multiplatform 4.0
- WebSphere Personalization Server 3.5
- WebSphere Extended Personalization Offering 4.0 (WAS, WSAD, WS Personalization for Multiplatform, WS Site Analyzer)

5.4.4 Customer loyalty in the financial services industry

The intense, enterprise-wide focus on customer loyalty initiatives by many financial institutions points to an emerging truth: The ultimate goal is to retain and grow a base of dedicated customers. That is easier said than done, especially when you have separate applications and systems that address single business issues. The challenge is even more daunting when you consider that customers now expect more and have more choices than ever before. There is little room for error.

By integrating all data into a single view, data can be transformed into accessible customer information to allow delivery of consistent, customized services over any channel. A solution should provide the tools that enable the vendor to treat customers as individuals and deliver value on their terms: what they want, when they want it. Customer analytics determine the projected lifetime value of a customer to the organization so that service levels can be matched to individual customer profitability levels.

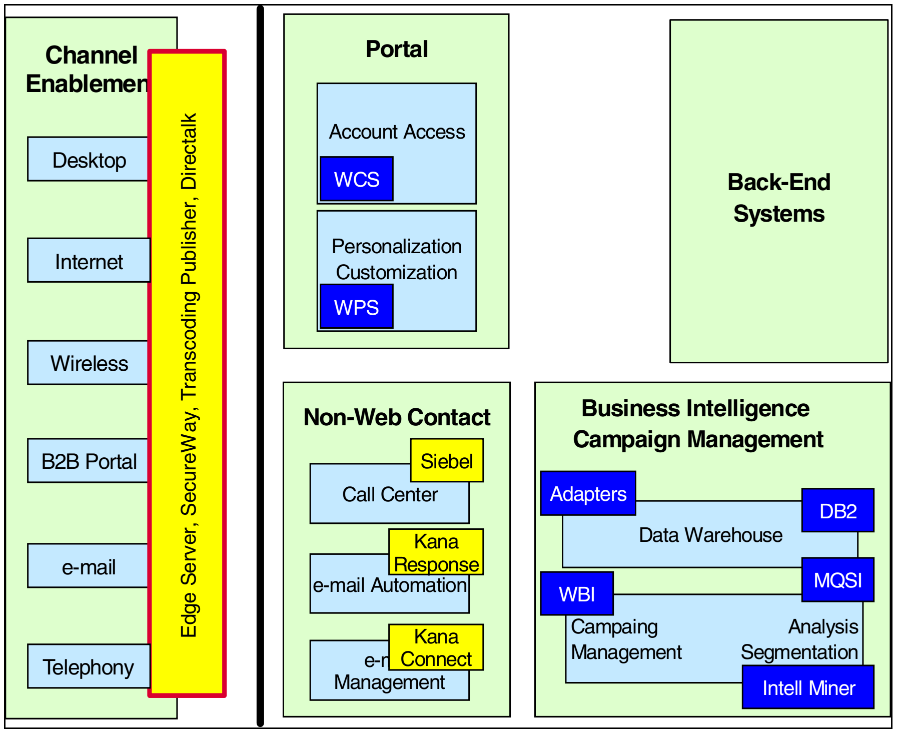


Figure 5-17 Customer loyalty architecture

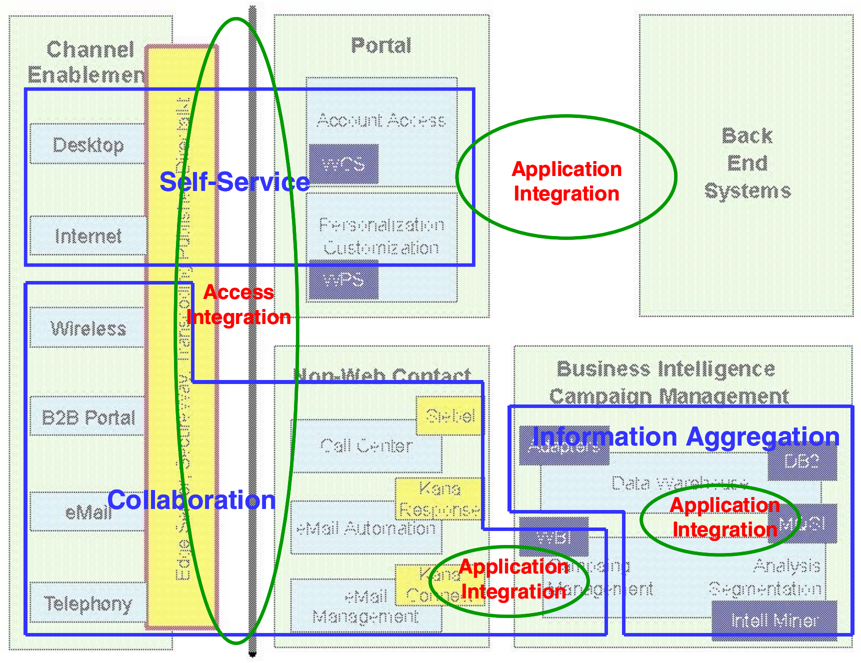


Figure 5-18 Patterns of customer loyalty

Patterns identified in customer loyalty are shown in Table 5-6.

Table 5-6 Customer loyalty patterns

Patterns	**Application Patterns**
Information Aggregation	Data Mining
Information Aggregation	Population - Multistep
Information Aggregation	Information Access - Read Only
Information Aggregation	Mining/Analysis
Collaboration	Store and Retrieve
Application Integration	Managed Process
Application Integration	Broker
Application Integration	ODS
Access Integration	Single Sign-On and Role-Based Access

Patterns	Application Patterns
Access Integration	Personalized Delivery
Access Integration	Pervasive Device Access
Customer Portal composite pattern	
Employee Portal composite pattern	
Account Access composite pattern	

Business processes in customer loyalty

The following describes the businesses processes:

- Cross-industry processes:
  - Customer sales analysis
  - Multichannel campaign
  - Product sales analysis
  - Customer loyalty analysis (acquisition, retention)
  - Order entry service
  - Order entry management
  - Customer life cycle
- Financial industry processes:
  - Insurance claim
  - Inquiry claim
  - Add first notice of loss
  - Policy quote (specialization of order entry service)
  - Add bill payment
- Solution specific processes:
  - Real-time campaign management

5.4.5 Composite pattern: e-Bank

Less than ten years after the emergence of the World Wide Web, e-business is radically reshaping retail financial services. In particular, the rapid growth of online trading has shown how e-commerce can reduce prices, increase convenience, and add value in ways that can build market share rapidly for firms that understand this new medium.

The stage is now set for the rapid expansion of Internet-based retail banking services. In the next three years, IBM expects e-commerce to rise exponentially as competition among online personal finance providers accelerates, boosting demand for online lending, bill payment, and investment services. Amid the proliferation of new Internet financial services and products, online consumers are searching for clarity and convenience. Retail banks can meet this compelling need by building low-cost, interactive banking platforms that allow their customers to electronically simplify and integrate a broad range of personal financial services.

e-Bank is a customer-oriented system. For example, e-Bank provides integrated management of customer-related data, which had traditionally been managed by a number of separate systems, in its customer information system. Therefore, related departments can share and analyze accurate customer data. Furthermore, the analytic CRM system provides market and customer trend analysis to allow the bank to conduct more effective, proactive marketing activities.

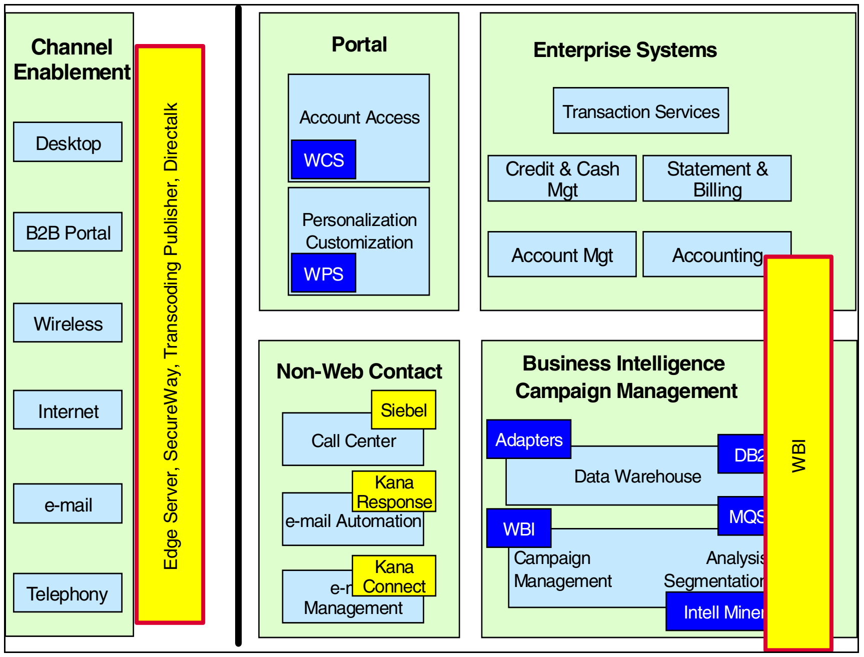


Figure 5-19 e-Bank architecture

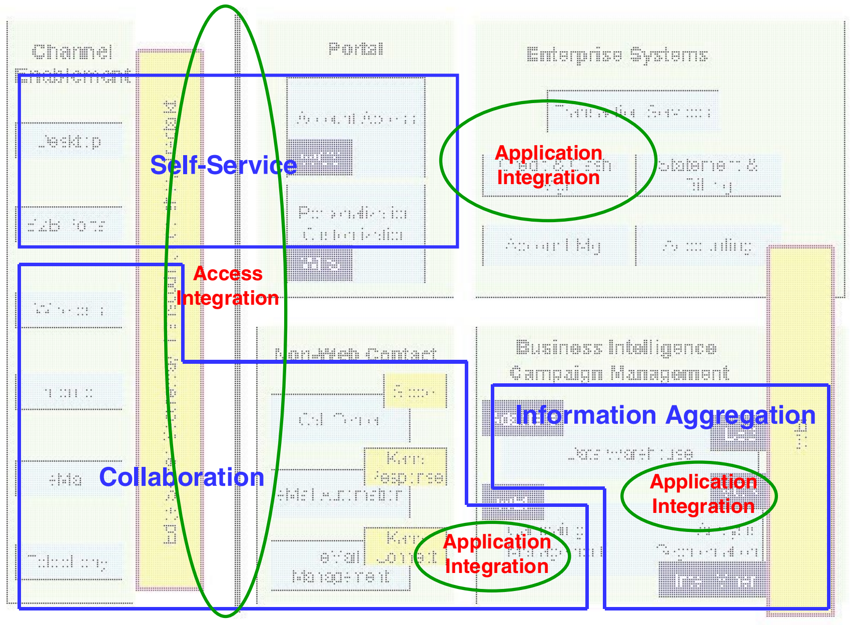


Figure 5-20 Patterns of e-Bank

The patterns identified in e-Bank are shown in Table 5-7.

Table 5-7 e-Bank pattern

Patterns	**Application Patterns**
Information Aggregation	Population - Multistep
Information Aggregation	Data Mining
Information Aggregation	Extensive User Update
Information Aggregation	Information Access - Read Only
Collaboration	Managed Collaboration
Extended Enterprise	Document Exchange
Application Integration	Propagation
Application Integration	Broker
Application Integration	Transactional
Application Integration	Managed Process

Patterns	Application Patterns
Application Integration	Operational Data Store
Access Integration	Pervasive Device Access
Access Integration	Single Sign-On and Role-Based Access
Access Integration	Personalized Delivery
Customer Portal composite pattern	
Employee Portal composite pattern	
Account Access composite pattern	

Business processes in e-Bank

The following describes the businesses processes:

- Cross-industry processes:
  - Customer sales analysis
  - Customer account inquiry
  - Order entry soft/hard
  - Order entry services
  - Order management soft/hard
  - Order management services
  - Multichannel campaign
  - Product sales analysis
  - Product life-cycle management (add, change, delete, replace)
  - Customer life-cycle management (add, change, delete)
  - Request for quote
  - User access management (add, change, delete)
  - Accounts receivable
  - Accounts payable
  - Customer service request
  - Customer loyalty analysis (acquisition, retention)
  - Online billing and statement with payment
- Financial industry processes:
  - Credit and cash management
  - Add bill payment
  - Account management
  - Electronic transfer
  - Stop payment
  - Transaction services
  - Balance inquiry
  - Account inquiry

  - Deposit account statement inquiry
  - Credit card statement closing
  - Deposit account transaction inquiry
  - Card account transaction inquiry
  - Bank account transaction image inquiry
  - Interest rate inquiry
  - Bank account taxation inquiry
  - Foreign exchange rate inquiry
  - Stop check add
  - Stop check cancel
  - Stop check inquiry
  - Stop check audit
  - Stop check synchronization
  - Funds transfer add
  - Funds transfer modify
  - Funds transfer status modify
  - Funds transfer cancel
  - Funds transfer inquiry
  - Funds transfer audit
  - Funds transfer synchronization
  - Recurring transfer model add
  - Recurring transfer model modify
  - Recurring transfer model cancel
  - Recurring transfer model inquiry
  - Recurring transfer model audit
  - Recurring transfer model synchronization
  - Check order add
  - Deposit book order add

5.4.6 WSBCC and Eontec

IBM and business partner Eontec have collaborated to combine WebSphere Business Components Composer (WSBCC) with the Eontec financial components. In the resulting solution, WSBCC multichannel business operations can launch a request to an Eontec financial component that can, in turn, use WSBCC Communication Services to integrate with other applications and data. Together, Eontec and WSBCC can be used to create a multichannel application that supports the Self-Service business pattern.

Eontec uses a J2EE-based architecture including stateless session beans and entity beans that implement bean-managed persistence. This approach allows the entity beans (and the underlying Eontec framework) to adapt to alternative mechanisms to persist information, such as a database or through the use of other application services. The Eontec framework’s persistence layer uses services of the Eontec Financial Process Integrator component to integrate with services of enterprise applications. The Financial Process Integrator uses WSBCC Communication Services to access a larger framework of enterprise applications.

For example, consider a request from a client received through the WSBCC multichannel presentation layer that invokes an Eontec component that needs to populate itself with data that resides in another (for example, legacy) application. The Eontec component will make a request to the Eontec persister layer, which will be passed to the Eontec Financial Process Integrator. The Financial Process Integrator uses its routing configuration information to determine which system or connector needs to be used to access the data and what transformation needs to occur before sending the request using WSBCC Communication Services, as well as after a response is received. These characteristics fit well with the description of the Self-Service Decomposition application pattern.

The Self-Service Decomposition application pattern has one basic Runtime pattern associated with it, as shown in Figure 5-1 on page 76. This Runtime pattern emphasizes the separation of concerns between application functions provided by the Application Server node and integration functions provided by the Integration Server node. A literal interpretation of this Runtime pattern would typically lead to deployment of application services on one physical server and integration services on another, separate, physical server. A less strict interpretation of this pattern as a *logical* view could allow for the possibility of deploying both application and integration services on a single physical server.

The former, literal approach emphasizes hub and spoke style integration services that are made available to a wide range of applications. This explicit tiering is optimized for loose coupling between application services and for providing a generalized set of enterprise-wide integration services. A physical hub and spoke configuration also maximizes the *session concentration* characteristics of the Decomposition Runtime pattern, typically reducing the total number of connections that need to be established with other existing applications. Figure 5-21 on page 114 is a representation of this approach.

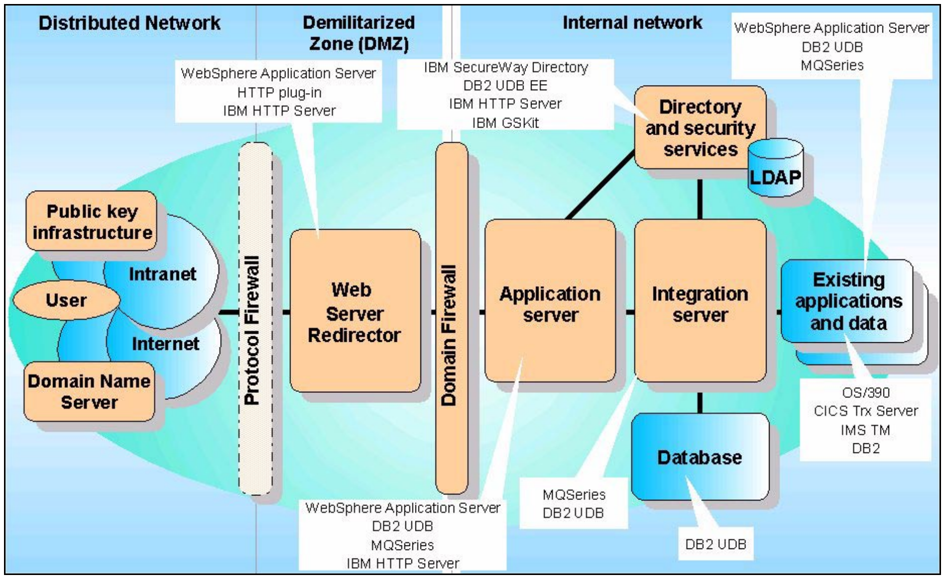


Figure 5-21 Runtime pattern for Self-Service Decomposition Application pattern

A less strict interpretation of the Decomposition pattern could be implemented using integration services that are more tightly coupled to an overall application service framework and which could be deployed on the same node as the application services. This approach could result in an infrastructure topology that is likely to be easier to implement and less complex. This simplicity could potentially give up some ground in the areas of the level of generalization of the integration services and in session concentration. WSBCC standardized Communication Services reduces these concerns. A revised view of the Decomposition Runtime pattern is shown in Figure 5-22 on page 115. A notable feature of this revised topology is a single logical node labeled *Application & Integration Server*. This topology is very similar to the Self-Service Directly Integrated Single Channel runtime pattern seen previously in this chapter.

While WSBCC can support both interpretations of the Decomposition runtime pattern, the latter approach, where application services and integration services are deployed on the same physical server, is a commonly used approach to WSBCC deployment. When implementing this approach to the Runtime pattern, the server configuration would be similar to the revised view of the Decomposition Runtime pattern shown in Figure 5-22 on page 115.

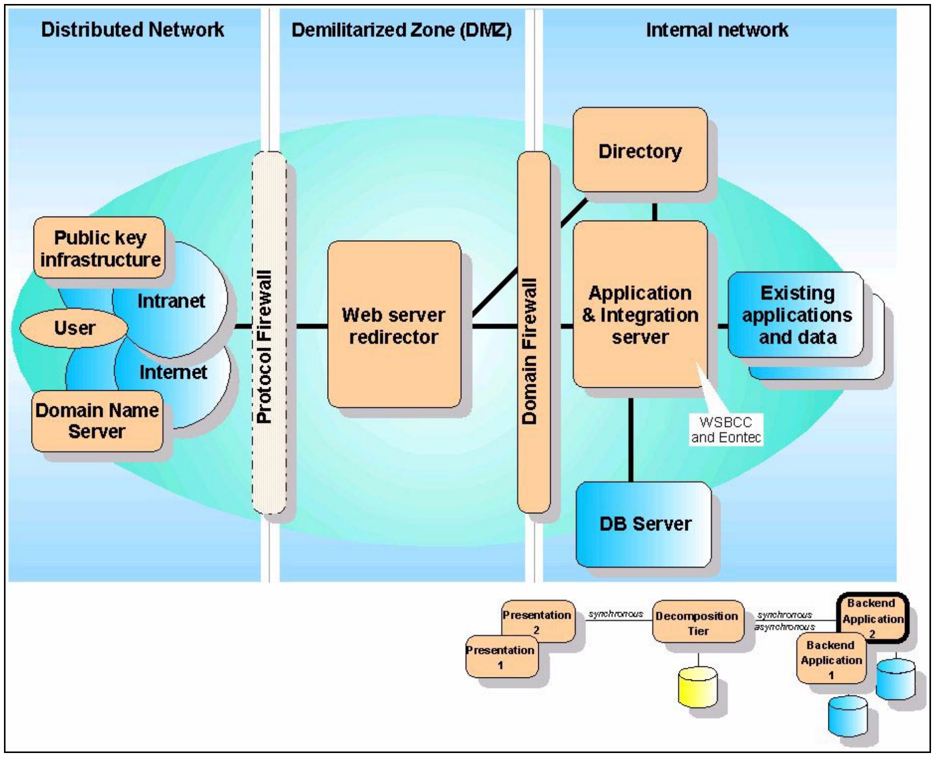


Figure 5-22 Revised view of the Runtime pattern

As is common with many architectural decisions, there are some trade-offs to consider with regard to these deployment alternatives. The physical hub and spoke integration services configuration in Figure 5-1 on page 76 is missing at a physical level in Figure 5-2 on page 77 but are preserved at a logical level. This is an important observation to consider. Regardless of the number of servers used to deploy WSBCC and Eontec, from the standpoint of the application/integration servers and existing applications, consistent integration techniques are used. One of the main differences between the two approaches will be in the number of connections that need to be established between the application/integration servers and existing applications. One of the important similarities in the two approaches is the use of a consistent set of integration techniques. In the case of a branch-level deployment, each branch's application/integration server would establish its own connections to existing applications. In a centralized deployment scenario, the number of connections would be driven by capacity requirements and the degree of horizontal scaling being employed.

So far, the discussion of WSBCC and Eontec has focused on options for the deployment of application and integration services. WSBCC also provides flexible deployment of presentation-related components. Presentation components can be deployed at execution time in a thin client approach using Java applets, or can be deployed using a zero footprint client approach by keeping presentation components on the server side. This allows Eontec financial application services to be made available on client machines with a wide range of capacities and physical characteristics.

The flexibility in deployment options across all tiers combined with built-in financial application services makes WSBCC and Eontec a valuable alternative to consider as part of a banking transformation strategy at the branch or enterprise level.

5.5 Summary

This chapter has described in detail how Patterns for e-business can be applied to branch banking, resulting in various Runtime patterns and Product mappings. Most of the Product mapping examples use Linux where applicable.

In the next chapter, we describe some specifics of how Linux applies to branch banking and the various roles for which it is well suited.

6

Linux-based products applicable to branch banking

Now that we have described the branch banking environment and Patterns for e-business in detail, you may have an architecture in mind that will suit your specific needs. Eventually, you will need to start creating the product mappings and making the choices for the operating environments across your infrastructure.

In this chapter, we provide a summary of the various products and facilities available for Linux environments. This may help you evaluate which systems within your environment can be, and maybe should be, running Linux.

6.1 Linux in branch banking environments

In Chapter 3, “Branch banking requirements” on page 45, we discussed what kind of components and services are required for a branch banking solution. In this section, we discuss how Linux can address these minimum requirements.

There are many products and solutions currently available for Linux, and this number is growing daily. In the following sections, we describe just a small subset of these solutions. Those that we do discuss are often used (and required) within an IT infrastructure built to support branch banking. Many of these facilities are provided natively with Linux distributions, while others are add-on packages that are available for Linux environments.

6.1.1 Network services

Let’s start by discussing some of the network services that are required. Most of these services have been provided by Linux since its inception and are proven and stable.

File Transfer Protocol (FTP)

FTP is the simplest way to exchange files between computers on TCP/IP networks. FTP is an application protocol commonly used to download programs and other files to desktops or servers from other servers.

Network File System (NFS)

NFS is a client/server application used to share files between machines on a network as if the files were located on the client's local hard drive. Linux can be both an NFS server and an NFS client, which means that it can export file systems to other systems and mount file systems exported from other machines.

Network Information System (NIS)

NIS is a naming and administration system for networks. Using NIS, each host client or server computer in the system has knowledge about the entire system. A user at any host can get access to files or applications on any host in the network with a single user identification and password. NIS is similar to the Domain Name System (DNS), but somewhat simpler and designed for a small network and intended for use on local area networks.

NIS uses the client/server model and the Remote Procedure Call (RPC) interface for communication between hosts. NIS consists of a server, a library of client programs, and some administrative tools. It is often used with the Network File System.

Domain Name System (DNS)

DNS is the way that Internet domain names are located and translated into Internet protocol addresses. A domain name is a meaningful and easy-to-remember “handle” for an Internet address.

Because maintaining a central list of host names and their related IP addresses would be impractical, the lists of domain names and IP addresses are distributed throughout the network in a hierarchy of authority. There is probably a DNS server within close geographic proximity to your access point that maps the domain names in your network and that handles requests or forwards information to other servers as needed.

Firewall

A firewall is a set of related programs, located at a network gateway server, that protects the resources of a private network from users from other networks.

Basically, a firewall, working closely with a router program, examines each network packet to determine whether to forward it to its destination. A firewall also includes, or works with, a proxy server that makes network requests on behalf of workstation users.

Hypertext Transfer Protocol (HTTP)

HTTP is a set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on TCP/IP networks. Relative to the TCP/IP suite of protocols, HTTP is an application protocol. There are many HTTP server implementations available for Linux.

Hypertext Transfer Protocol Secure (HTTPS)

HTTPS is secure HTTP utilizing encryption.

Apache HTTP Server

The Apache HTTP Server Project is an effort to develop and maintain an open source HTTP server for modern operating systems, including UNIX and Windows NT. The goal of this project is to provide a secure, efficient, and extensible server that provides HTTP services in sync with the current HTTP standards. Apache has been the most popular Web server on the Internet since April of 1996. For more information about Apache, see:

http://httpd.apache.org/

IBM HTTP Server

IBM HTTP Server powered by Apache is based on the Apache HTTP Server. IBM has enhanced the Apache-powered HTTP Server, for example, adding SSL for secure transactions and offering full support, when part of the WebSphere bundle.

IBM Communications Server for Linux

IBM Communications Server for Linus provides a multiprotocol gateway and provides workstation communications services. It enables personal computers to communicate with System/390 and AS/400 hosts and other personal computers. The Enterprise Extender feature allows SNA applications to operate across a TCP/IP network.

For client/server and distributed applications, IBM Communications Server includes support for peer-to-peer networking (APPN) network node and end node. IBM Communications Server also supports a rich set of application programming interfaces.

IBM WebSphere Host On-Demand

WebSphere Host On-Demand gives your users a simple way to reach critical host data, without requiring any software to be installed on the client. WebSphere Host On-Demand uses Java technology to help open the doors to your host data directly from your browser. This Web-to-host connectivity solution helps provide secure Web-browser access to host applications, so you can take existing host applications to the Web without programming. With support for TN3270E, TN5250, VT52, VT100, VT220, VT320, VT420, and CICS Transaction Gateway access, users can have a single interface to their key host data. Because it is Java technology-based, its interface has the same look and feel across various types of operating environments. It also provides a default GUI to help simplify the experience for users unfamiliar with traditional green screens.

IBM WebSphere Host Access Transformation Server

IBM WebSphere Host Access Transformation Server (HATS) gives you all the tools you need to quickly and easily extend your legacy applications to your customers and employees. HATS makes your 3270 and 5250 applications available through the most popular Web browsers, while converting your host screens to a Web-like look and feel. HATS enables you to extend your host application to the Web within a single day of installing the software without any changes to your host. HATS is a zero footprint Web-to-host solution; the only software needed on the client is a Web browser.

IBM WebSphere MQSeries

WebSphere MQ enables application integration by helping business applications to exchange information across different platforms by sending and receiving data as messages. They take care of network interfaces, assure *once only* delivery of messages, deal with communications protocols, dynamically distribute workload across available resources, handle recovery after system problems, and help make programs portable.

The MQ Adapter allows data to be passed between MQ messages and applications. Data can then be sent to, or received from, any supported platform. MQ provides a single, multiplatform application-programming interface. A key factor is time-independent processing. This means that messages are stored reliably for later delivery, even if one of the recipients is temporarily unavailable.

IBM WebSphere Edge Server

WebSphere Edge Server distributes application processing to the edge of the network under centralized administrative and application control. It includes:

- **Application off load**: Shifts the burden of serving composed, personalized, dynamic content from the application server to Edge Servers placed at the network edges by off loading back-end servers and peer links.
- **Content distribution**: Deploys published Web content to caches and *rehosting servers* throughout the network.
- **Caching**: Improves response time by off loading back-end servers and peer links as a forward, reverse, or transparent proxy. It caches and invalidates dynamic content generated by WebSphere Application Server, including JavaServer Pages (JSPs) components and servlets.
- **Load balancing**: Through features such as Network Address Translation (NAT), Network Address Port Translation (NAPT), Proxy and Kernel-level Content Based Routing (CBR), and others. It can improve server selection, load optimization, and fault tolerance.
- **Security**: Can be centralized using Tivoli Access Manager. With the Edge Server's Caching Proxy configured to use the Tivoli Access Manager Plug-in, the Access Manager authorization engine ensures only authorized users can access cached and non-cached resources. This integrated solution helps ensure control at the edge of your network.

IBM Directory Integrator

IBM Directory Integrator is an integration development and deployment tool that helps customers quickly and effectively integrate application-specific directories to provide a consistent, enterprise-level view of directory information.

With some built-in connectors, an open-architecture Java development environment to extend or modify these connectors, and tools to apply logic to data as it is processed, IBM Directory Integrator can help you by:

- Synchronizing and exchanging information between applications or directory sources.
- Managing data across a variety of data repositories.
- Providing a consistent directory infrastructure that can be used by a wide variety of applications ranging from security and provisioning to Web services.
- Offering a consistent view of additional directory-based information, such as product and pricing data, to applications beyond traditional user identity and passwords.

6.1.2 User management and security

Because most branches have a need for local processing when the communications link to the central site is unavailable, there is a need to have some level of user management that can be enforced at the branch level. This section describes some of the offerings in this area.

Samba

Samba provides Windows Common Internet File System (CIFS) and SMB functionality for UNIX clients and servers and can be configured to provide a Windows NT like Primary Domain Controller for the Branch.

It’s based on the common client/server protocol of Server Message Block (SMB) and Common Internet File System (CIFS). Using client software that also supports SMB/CIFS, for example, Windows and OS/2 products, an end user sends a series of client requests to the Samba server on another computer in order to open that computer's files, access a shared printer, or access other resources. The Samba server on the other computer responds to each client request, either granting or denying access to its shared files and resources.

Pluggable Authentication Modules (PAM)

Pluggable Authentication Modules can be used to allow the system administrator to set authentication policies for PAM-aware applications without having to recompile authentication programs. PAM does this by utilizing a pluggable, modular architecture. PAM modules can be developed to authenticate users against a centralized database.

When used correctly, PAM provides many advantages for a system administrator, such as the following:

- A common authentication scheme that can be used with a wide variety of applications.
- PAM can be implemented with various applications without having to recompile the applications to specifically support PAM.
- Great flexibility and control over authentication for the administrator and application developer.
- Application developers do not need to develop their program to use a particular authentication scheme. Instead, they can focus purely on the details of their program.

Kerberos

Kerberos is a secure method for authenticating a request for a service in a computer network. Kerberos lets a user request an encrypted *ticket* from an authentication process that can then be used to request a particular service from a server. The user's password does not have to pass through the network.

Snort

Snort is a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching and matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, common gateway interface (CGI) attacks, Server Management Block protocol (SMB) probes, operating system (OS) fingerprinting attempts, and much more. Snort uses a flexible rule-based language to describe traffic that it should collect or pass and a modular detection engine. Snort has a real-time alerting capability, with alert mechanisms for syslog, a user specified file, a UNIX socket, or Win Popup messages to Windows clients using Samba smbclient.

Lightweight Directory Access Protocol (LDAP)

LDAP is a proposed open standard for accessing global or local directory services over a network. LDAP is very useful because it was designed to support propagation across LDAP servers throughout the network, much like the Domain Name Service (DNS). DNS servers help to connect computers to one another based on domain names or the type of service requested from a domain. Without DNS servers, host names could not be translated into IP addresses, which are required for TCP/IP communication. In the future, LDAP could provide the same type of global access to many types of directory information. Currently, LDAP is more commonly used within a single large organization, such as a bank, for directory services.

LDAP is a client/server system where the LDAP client connects to an LDAP server and either queries it for information or provides information that needs to be entered into the directory. The server either answers the query, refers the query to another LDAP server, or accepts the information for incorporation into the directory, based on the permission of the user.

IBM Directory Server

IBM Directory Server provides a powerful Lightweight Directory Access Protocol identity infrastructure that is the foundation for deploying comprehensive identity management applications and advanced software architectures, such as Web services.

IBM Directory Server and the new directory and metadirectory services provide capabilities that are essential for companies that need to deploy Web services and user provisioning. A directory is a database used to store and retrieve user, resource, and other key network information. Metadirectory software aggregates directory information that may be stored in directories or databases provided by multiple software vendors and running on a variety of operating systems.

IBM Tivoli Access Manager for e-business

IBM Tivoli Access Manager is a policy-based access control solution for e-business and enterprise applications. It uniquely addresses the challenges of e-business security, enabling new and rapidly scaling e-business initiatives to reach new markets and customers. It also addresses managing growth and complexity, controlling escalating management costs, and directly tackles the difficulties of implementing security policies across a wide range of Web and application resources. Tivoli Access Manager for e-business helps companies by reducing deployment time and cost for new e-business applications.

Tivoli Access Manager for e-business lets organizations control both wired and wireless access to applications and data and provides single signon (SSO) for authorized users. Tivoli Access Manager for e-business integrates with e-business applications to deliver a secure personalized e-business experience for authorized users. Tivoli Access Manager includes integrated security for key customer relationship management (CRM), enterprise resource planning (ERP), and supply chain management (SCM) e-business solutions, as well as enhancements for securing J2EE-conforming applications running on WebSphere Application Server. Tivoli Access Manager for e-business provides partners, customers, suppliers and employees with secure access to business-critical applications and data for highly available and scalable transactions.

6.1.3 File and print services

Branch offices almost always require some level of local file and printer sharing. Again, there are many Linux-based products for this area. In this section, we describe a few of the more popular solutions.

Samba

Samba is a very popular open source server package where a Linux system can provide file and print services, also known as shares. Samba implements the Server Message Block (SMB) protocol, a complex protocol, developed by IBM in 1984, that provides file and print services for Windows, OS/2, and Linux client desktops. Some application components make use of these types of services under certain application design models.

Samba is used by many people around the world on a daily basis in a production capacity. Perhaps this is because of Samba's open source nature, the fact that it ships with Linux distributions, and because of the Samba team's efforts to supply a product that can work seamlessly in an otherwise Windows-based environment.

For further information about SAMBA, see:

`http://www.samba.org`

IBM Tivoli Storage Manager

IBM Tivoli Storage Manager protects your data from hardware failures and other errors by storing backup and archive copies of data on offline storage. It scales to protect hundreds of computers running a variety of operating systems across a wide range of hardware ranging from laptops to mainframes. They can be connected through the Internet, WANs, or LANs. The Tivoli Storage Manager’s centralized, Web-based management, smart-data-move and store techniques, and comprehensive policy-based automation all work together to minimize data protection administration costs and the impact to both computers and networks. Optional modules allow business-critical applications that must run 24x365 to utilize the Tivoli Storage Manager centralized data protection with no interruption to their service.

IBM Linux Technology Center Omni drivers

The Omni printer driver provides support for over 400 printers using the Ghostscript framework. In addition, it provides a model for dynamically loading printer drivers, creating new devices by editing device description files, and simplifies new printer driver development by allowing for the sub-classing of previous device features.

6.1.4 Store and forward

Store and forward services provide support for operations that cannot be processed online because there is no back-end connectivity, perhaps due to an unexpected interruption, and also for batch processing operations.

Typically, store and forward services are built into applications or application frameworks rather than as a separate product. The products described in the following sections can support applications requiring store and forward capabilities.

IBM WebSphere Business Components Composer

WebSphere Business Components Composer (WSBCC) is a strategic element of both multichannel and branch transformation solutions for the retail banking industry. It brings high value to financial institutions looking to renew retail delivery channels as part of larger initiatives to improve operating efficiency and profitability. By enabling financial institutions to build and deploy an infrastructure that delivers consistent, flexible business logic and customer service across all retail delivery channels, this offering enhances a bank’s ability to understand and satisfy unique customer needs.

WSBCC provides a set of components designed specifically for these types of retail banking applications. By using WebSphere Business Components Composer's proven component-based middleware framework, retail banks can build integrated applications to support multiple customer channels and branch transformation efforts.

IBM Bank Teller Business Components

The IBM WebSphere software branch transformation solution is enabled by IBM Bank Teller Business Components. These components provide the flexibility and adaptability to implement a Java teller system based on a set of proven and high-performing components. The components enable an environment that allows customers to rapidly build and deploy teller transactions in a manner that supports transaction reuse across other channels. Implemented by a number of global financial institutions, the WebSphere branch transformation solution provides a solid foundation for leveraging WebSphere capabilities across the enterprise.

IBM Local Area Network Distributed Platform (LANDP)

LANDP has been designed and developed primarily for retail banking. It is a tried and tested solution that is used by a large number of customers worldwide, including some of the world's largest banks. LANDP includes the following:

- Client/server environment: Supports multiplatform networks running a combination of Linux, Windows, OS/2, and DOS communicating over NetBios or TCP/IP.
- Communications servers: Supports different communications links and protocols to run host applications.
- Data management servers: Provides support for the storage, retrieval, and updating of data in LANDP workgroups.
- Database servers with multiple access modes, a shared-file server, an electronic journal, and a store-and-forwarding server cover a wide range of data management requirements.
- Financial device servers: LANDP can be used to access and share financial devices.
- Common Application Programming Interface: Used to develop applications for LANDP environments. The location of services in a workgroup is managed by LANDP. This means that when you develop an application, you do not need to know where the server that supports the service is located.
- Migration support: LANDP cross-platform capabilities enable the same applications and services to be easily migrated across many different platforms.

6.1.5 Database services

Some application models currently use a database in the branch to house data that is required in offline mode. For example, some implementations of an electronic journal might use a database. In addition, a database may be used to store configuration data that is required during offline processing. Branch database services may not be required in more centralized application design models with limited offline requirements.

Red Hat Database

Red Hat Database is the open source database tested and verified for use with Red Hat Linux Advanced Server and Red Hat Linux. It is powered by an enhanced version of PostgreSQL and its ISO image is freely downloadable from the Red Hat download center at:

`http://www.redhat.com/apps/download/`

MySQL

MySQL is one of the world's most popular open source database systems, designed for speed, power, and precision in mission critical, heavy load use.

The software from MySQL AB that you can download from the MySQL Web site is licensed under the GPL license and is provided “as is” and is without any warranty:

http://www.mysql.com

PostgreSQL

PostgreSQL is a sophisticated object-relational DBMS, based on POSTGRES, developed at the University of California at Berkeley Computer Science Department, supporting almost all SQL constructs, including sub-selects, transactions, and user-defined types and functions. It is the most advanced open source database available anywhere. Commercial support is also available.

IBM DB2 for Linux

The DB2 advantage is simple: pure power at a lower cost than any other e-business-ready Linux database on the market. With DB2, you can access, manage, and analyze all forms of information across the enterprise, and on all major platforms. With Linux becoming a major platform for database and application servers, DB2 for Linux gives you a robust, easy-to-manage database that offers high performance, complementing the stability and reliability of Linux.

DB2 for Linux supports and embraces open standards, including Java and XML, and integrates with many open source products, such as Apache, PHP, Perl, and Python. DB2 UDB is the most scalable database in production today that can manage mission-critical data on a single PC, SMPs, clusters, as well as the mainframe for Linux.

IBM Informix Dynamic Server

IBM Informix Dynamic Server (IDS) provides the mission-critical availability, reliability, scalability, and data-intensive transaction management capabilities that a growing number of organizations rely on to succeed in e-business. It also provides world-leading extensibility features.

Because business logic can be embedded in the database and fully integrated in IBM Informix Dynamic Scalable Architecture (DSA), IDS can solve business problems that few other systems can. It can also provide uniform access to client information, regardless of where it resides, through leading distributed database and virtual table capabilities, thereby allowing queries to encompass a wide range of databases, from flat files to other IDS and non-IDS databases.

6.1.6 Application servers

In today's e-business world, application servers play a critical role. Linux provides a stable and affordable base for running application servers, especially in a large and distributed environment such as branch banking.

Tomcat

Developed by members of the Jakarta Project, Tomcat is a servlet container based on Java Servlet 2.3 technology and a JavaServer Pages (JSP) 1.2 implementation. There are other products like it, but Tomcat is the official reference implementation for these two technologies, and one of the few freely available application servers that offers support for JSP 1.2.

Tomcat is developed in an open and participatory environment and released under the Apache Software License. Further information about Tomcat can be found at:

`http://jakarta.apache.org/tomcat/`

IBM WebSphere Application Server

WebSphere Application Server is a Java technology-based Web application server, integrating enterprise data and transactions with the e-business world. It provides a rich, e-business application deployment environment with a complete set of application services, including capabilities for transaction management, security, clustering, performance, availability, connectivity, and scalability.

WebSphere Application Server is the core Web services and J2EE compatible application server enabled with industry-leading qualities of service and an array of flexible deployment configurations to meet the needs of stand-alone, multi-server distributed, and highly dynamic decentralized distributed enterprise environments.

6.1.7 Messaging servers

Messaging is the exchange of messages (specially-formatted data describing events, requests, and replies) to a messaging server, which acts as a message exchange program for client programs.

IBM WebSphere MQ

For additional information about WebSphere MQ, see "IBM WebSphere MQSeries" on page 121 or the WebSphere MQ Web site at:

`http://www-3.ibm.com/software/ts/mqseries`

6.1.8 Systems management

In a geographically distributed environment such as branch banking, it is important to have the facilities and tools available to remotely manage the workstations and servers in this environment. Again, there are a variety of management facilities and products provided by or available for Linux.

Telnet, rsh, and rlogin

Telnet is the way you can access someone else's computer, assuming they have given you permission. More technically, Telnet is a user command and an underlying TCP/IP protocol for accessing remote computers.

The **`rsh`** command lets you share processors and execute commands on remote systems. With **`rsh`**, it is a simple to ask a command to run on any computer on the network, for which you have access. You can have the command's output printed on your screen, directed to a local file, or directed to a remote file.

The **`rlogin`** command allows an authorized user to log on to other Linux machines (hosts) on a network and to interact as if the user were physically at the host computer. Once logged on to the host, the user can do anything that the host has given permission for, such as read, edit, or delete files, and stop or start process.

Secure Shell (SSH)

SSH is a protocol for creating a secure connection between two systems. Common methods for remotely logging on to another system through a shell (Telnet, rlogin, or rsh) or copying files between hosts (FTP or rcp) do not encrypt data that is sent over the connection between the client and the server and should be avoided. Instead, you should only connect to a remote host using a secure shell or an encrypted virtual private network. Using secure methods to remotely log on to other systems will decrease the security risks for both your system and the remote system.

NetSaint Network Monitor

NetSaint is a program that monitors hosts and services on network. It has the ability to e-mail or page you when a problem arises and when it gets resolved. NetSaint is written in C and is designed to run under Linux, although it should work under most other UNIX variants. It can run either as a normal process or as a daemon, intermittently running checks on various services that you specify. The actual service checks are performed by external plug-ins that return service information to NetSaint.

Webmin

Webmin is a Web-based interface for system administration for Linux. Using any browser, you can set up user accounts, network services, file sharing and printing, and so on. Webmin consists of a simple Web server and a number of programs that directly update Linux system files.

IBM Tivoli systems management software

IBM Tivoli software provides a comprehensive and complementary set of products that provide end-to-end management of IT environments. Enterprises can use Tivoli applications to manage everything from the mainframe, middle-tier servers to desktops and even PDAs.

Tivoli solutions are available for managing everything from the network infrastructure to middleware to applications. They have embraced Linux as both a platform to manage, as well as a platform on which to run the management applications. They have product-specific modules for managing many of the key software components required for an e-business, such as DB2, WebSphere, and Domino, and they are managed in a platform-independent way.

The Tivoli products fall into four broad categories of systems management:

- Performance and Availability
- Configuration and Operations
- Security
- Storage Management

Through the integration of the various products, Tivoli helps you to deploy, operate, secure and manage the performance and availability of your systems and data across all major platforms, including Linux.

IBM Desktop On-Call

Desktop On-Call is a remote control software product that uses a unique concept: you do not need to install software on the guest PC because you can control the host PC from another PC with an ordinary Web browser using a Java applet. It enables you to control your desktop PC remotely from virtually any PC at any location in the world.

Desktop On-Call is very helpful in many situations for banks. You can access files and other important applications. If you have a problem with an application program on your desktop PC, you can fix it from a remote PC. When the host PC is accessed through the Web browser, the Java applet program is dynamically downloaded to a Web browser and draws an image of the host PC desktop. The Java applet program then starts to communicate with the native daemon program. This enables you to control your host PC from a remote location.

IBM Director

IBM Director is a comprehensive workgroup hardware manager designed for use with IBM xSeries servers, PCs, and notebooks.

Working with IBM Director, the Advanced System Management service processors in xSeries servers are the key to hardware problem notification and resolution. They provide the system administrator with complete remote management of a system, independent of the server status. The processors simulate a computer within a computer, keeping the server up and available for your business-critical applications.

6.2 Summary

We have described a variety of Linux-based products and facilities that provide the needed infrastructure for a branch banking environment. Most of these have proven track records and provide the stability and performance you require. Having all of these capabilities (and more) available to run in Linux environments is helping drive the movement towards using Linux as a less expensive and low-risk option for many banks and other companies.

As you develop your own solution architecture using IBM Patterns for e-business as described in Chapter 5, “Applying IBM Patterns for e-business to branch banking” on page 73, you can use the information provided in this chapter to start identifying the product mappings required for your chosen Runtime pattern.

7

Scenario for a new branch banking solution

In 2.5, “Common branch banking scenarios” on page 32, we describe some common IT infrastructures for supporting current branch banking environments. In this chapter, we describe some scenarios related to the branch banks of the future. These scenarios are based on the trends we describe in Chapter 2, “Branch banking environment” on page 11, Patterns for e-business, and on the applicability and availability of Linux-based solutions for these environments.

7.1 Scenario overview

The challenge is to provide an architectural road map for a branch banking infrastructure that addresses today's business challenges and the services of tomorrow. So, the infrastructure of a new branch bank world must integrate with all other channels and become a core part of those channels as well. As a reminder, it must have the characteristics to support:

- A full range of services (self service to personal assistance)
- Advisor collaboration
- Wireless connectivity
- Drive-through traffic at the branch
- Increased focus on customer relationship management

Therefore, the goals for the overall solution include, but are not limited to, the following:

- Converged network infrastructure
- Flexible, role-based components:
  - Teller
  - Platform officer
  - Branch administration
- Enabled for collaboration
- Integrated with and enhancing the ATM channel
- Integration with the enterprise:
  - Operational banking systems
  - CRM
  - Wealth management

A potential scenario in a branch banking environment with integrated channels could be as follows:

1. A customer uses a mortgage calculator on the Web.
2. Later, the customer enters branch, makes a deposit, and receives a printed receipt.
3. During the interaction with the teller, the teller extends an offer to the customer to discuss mortgage refinancing with an expert (based on business intelligence knowledge of the mortgage calculator usage on the Web).

4. The customer makes an appointment for a meeting in the branch with a financial expert (through remote conferencing) to begin in 15 minutes and waits in the branch coffee shop until the time for appointment.
5. A concierge (notified on PDA) comes to get the customer when the expert is ready.
6. The customer has a remote conference with a financial expert regarding mortgage refinancing, using aggregation by portal to share customer information. The customer expresses an interest in home improvement (kitchen remodeling) and requests funds from refinancing to cover the remodeling.
7. The customer completes a pre-filled loan application with the financial expert.
8. A few days later, the customer goes to a Web-based ATM and withdraws cash.
9. The ATM informs the customer that the loan has been approved and offers to schedule the loan closing in one of the following ways: choosing from a list of available appointment times or requesting a customer service representative call for loan closing.
10. The customer accepts the call-back option.
11. During the loan closing, a platform officer extends an offer for a discount on home improvement services provided through a business partner and offers to schedule a home visit by a remodeling representative, which the customer accepts.

As outlined in Chapter 5, “Applying IBM Patterns for e-business to branch banking” on page 73, we have basically touched all four business patterns.

Table 7-1 Matching examples to Patterns for e-business

Step	**Self Service**	**Collaboration**	**Aggregation**	**Extended Enterprise**
1	**x**			
2	**x**			
3	**x**			
4	**x**			
5		**x**		
6		**x**		
7		**x**		
8			**x**	

Step	Self Service	Collaboration	Aggregation	Extended Enterprise
9			**x**	
10			**x**	
11				**x**

7.2 Scenario solution

Having fully understood all functional and non-functional requirements and using a methodology, such as Patterns for e-business, we might create an infrastructure such as the one shown in Figure 7-1 on page 137.

We have created an integrated, multichannel, e-business-enabled application framework that can be deployed to the branch. Business logic was moved off the client and away from legacy applications toward an open standard, Java-based infrastructure, ready to run on any platform supporting a browser and a Java Virtual Machine, including devices such as Web-based ATMs.

All business logic resides in a component-based implementation that can easily be hosted either centrally or distributed to regional or even branch servers for speed, performance, or availability, based on network reliability and other requirements. The new middle-tier contains the connectors to any back-end system. There might be systems already in place today or systems that may be created in the future. By using an architecture that supports pluggable connectors, we have a flexible environment that can keep up with changing requirements.

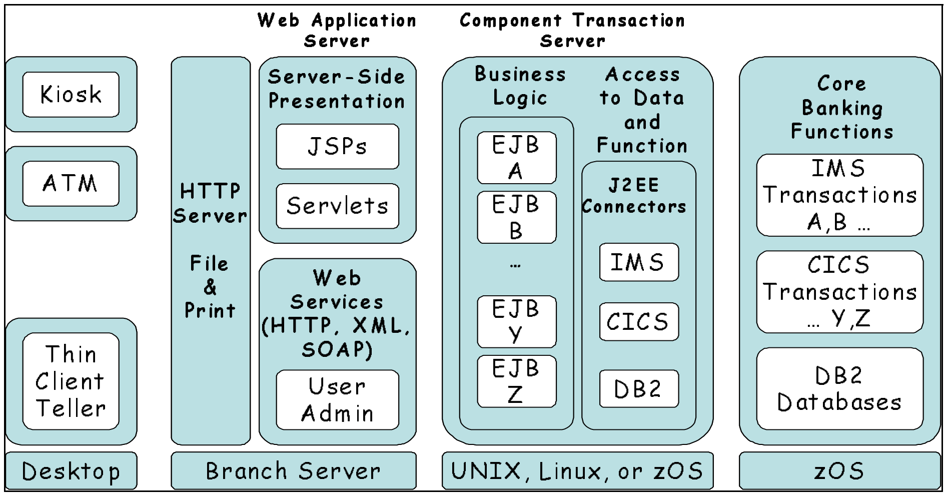


Figure 7-1 Step 3: Create a new infrastructure

Having this overview of a branch-focused system in mind, we need to draw the whole picture and allow all other channels to participate in the end-to-end architecture.

Let's focus a little more on the specific functions we need to support all business requirements and all patterns we found earlier. The more complete infrastructure might be as shown in Figure 7-2.

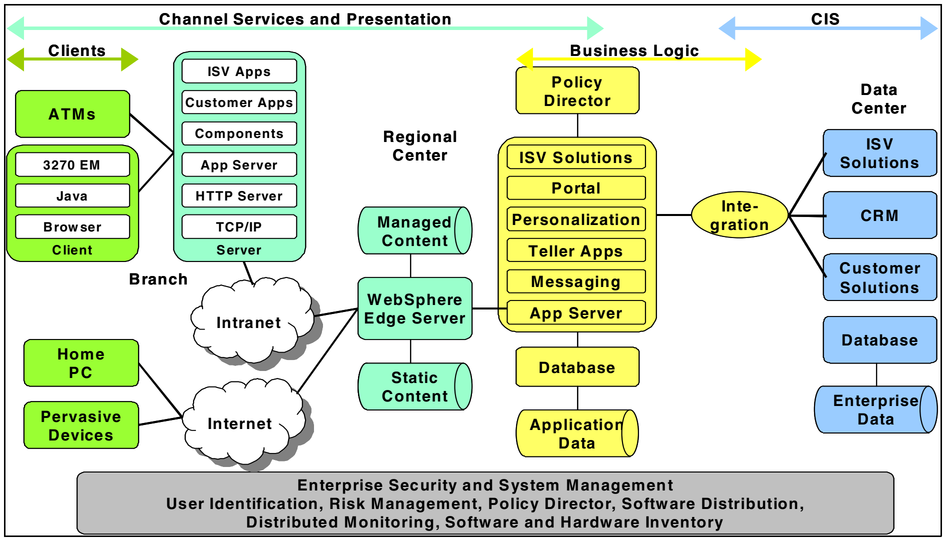


Figure 7-2 Future infrastructure outline

From this Runtime pattern, we would next perform a product mapping to the functions shown in the figure. One possible product mapping is shown in Figure 7-3.

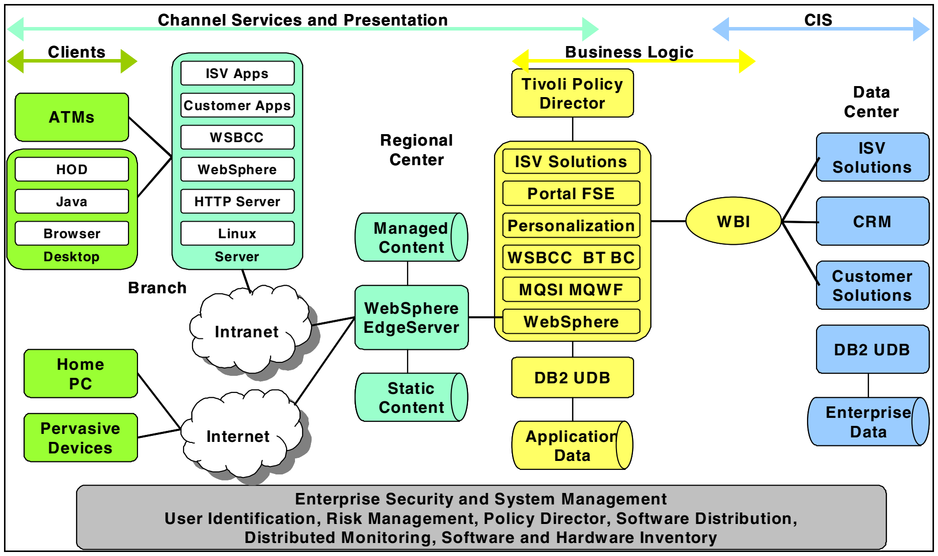


Figure 7-3 Future infrastructure product mapping

Because one of our goals and assumptions was to evaluate Linux as a viable platform to host some of these functions, we need to verify how much of this will run in a Linux environment. Table 7-2 lists the products we have highlighted and their supported platforms.

Table 7-2 Linux products overview

Product	Linux	Windows	OS/2
Host On-Demand	x	x	x
Java and browser	x	x	x
HTTP server	x	x	x
WebSphere	x	x	(x)
WSBCC	x	x	
MQSeries	x	x	x
DWSBCC BT	x	x	
Portal FSE	x	x	

Product	Linux	Windows	OS/2
WebSphere Business Integrator	x	x	
Tivoli Policy Director	x	x	

It looks like we have a match! Of course, there are many considerations for choosing the operating platform that will be use, but support for the required infrastructure products and facilities is not an issue for Linux.

Given its price, performance, stability, and support by organizations, such as IBM, Linux will play a key role in this environment.

7.3 Summary

The scenario outlined in this chapter describes, at a high-level, how one can use the information presented earlier in this redbook to look at the branch banking requirements and apply Patterns for e-business to drill down to the Runtime and product mapping levels. In this simple, but representative, case, we have shown how Linux can and should be considered as a viable and attractive alternative for branch banking infrastructures.

A

IBM Software for Linux

Linux is ready for your business!

It is no surprise that Linux is the world's fastest-growing operating system: it is open, stable, and easily scalable. IBM Software is also powerful, scalable, and based on open standards, such as Java, XML, and Web services.

Following are some products from IBM that were available during the writing of this redbook. To find the latest information about IBM Software for Linux, see:

http://www.ibm.com/software/linux/

This site is constantly changing as new and additional support for Linux is provided by IBM Software.

DB2 for Linux

DB2 enables you to access, manage, and analyze all information across the enterprise, whether your data resides on Linux, UNIX, Windows, or many other platforms.

Linux is becoming a major platform for database and application servers. DB2 for Linux gives you a robust, easy-to-manage database offering high performance, stability, and reliability. Together, DB2 and Linux are the building blocks for success in e-business.

Further information can be found about DB2 for Linux at:

http://www.ibm.com/software/data/db2/linux/

DB2 Universal Database products

The following are DB2 Universal Database products for Linux.

DB2 Universal Database Personal Edition

Provides a database management system for the desktop.

http://www.ibm.com/software/data/db2/udb/edition-pe.html

DB2 Universal Database Workgroup Edition

Multi-user database for applications and data for workgroups on PC-based networks.

http://www.ibm.com/software/data/db2/udb/edition-we.html

DB2 Universal Database Enterprise Edition

Multi-user database for complex configurations and large databases.

http://www.ibm.com/software/data/db2/udb/edition-ee.html

DB2 Universal Database Enterprise-Extended Edition

High performance to support large databases, offering greater scalability in clustered servers.

http://www.ibm.com/software/data/db2/udb/edition-eee.html

Note: All of the DB2 Universal Database products are available for IBM xSeries and other Intel-based servers. In addition, DB2 Universal Database Enterprise Edition is available for IBM zSeries.

DB2 application development

The following are DB2 application development solutions for Linux.

DB2 Personal Developer's Edition

Tools needed to develop desktop business tools and applications for DB2 Universal Database Personal Edition.

http://www.ibm.com/software/data/db2/udb/edition-pde.html

DB2 Universal Developer's Edition

Tools for developing client/server applications for DB2 Universal Database on any supported platform.

`http://www.ibm.com/software/data/db2/udb/edition-ude.html`

DB2 XML Extender

Provides new data types that allow customers to store XML documents in DB2 databases.

`http://www.ibm.com/software/data/db2/extenders/xmlext/index.html`

DB2 Net Search Extender

Adds fast full-text retrieval to DB2 applications.

`http://www.ibm.com/software/data/db2/extenders/netsearch/index.html`

DB2 Intelligent Miner Scoring

Enables users to do data mining in real-time applications with a simple SQL call.

`http://www.ibm.com/software/data/iminer/scoring/index.html`

Informix

The following is an Informix solution for Linux.

Informix Dynamic Server

Database on Linux for Informix applications. It is available on IBM xSeries and other Intel-based servers.

`http://www.ibm.com/software/data/informix/`

Connectors

The following are connectors for Linux.

DB2 Connect

Provides a connection to DB2 systems and is available on IBM xSeries, zSeries, and other Intel-based servers.

`http://www.ibm.com/software/data/db2/db2connect/`

IMS Connect

Provides a connection to IMS systems and is available on IBM zSeries.

`http://www.ibm.com/software/data/db2imstools/imstools/imsconnect.html`

CICS Transaction Gateway

Connection to transactions on CICS systems and is available on IBM zSeries.

http://www.ibm.com/software/ts/cics/ctg/index.html

WebSphere for Linux

IBM WebSphere for Linux can help propel your company into the next generation of e-business. Based on open, industry-accepted standards, WebSphere makes it easy to collaborate and strengthen relationships with customers, suppliers, and trading partners. For more information, see:

http://www.ibm.com/websphere/

Application server

The following is an application server for Linux.

WebSphere Application Server V4.0 Advanced Edition

Adds support for full Java 2 Enterprise Edition (J2EE) and Web services (SOAP, UDDI, WSDL). It is available on IBM xSeries, zSeries, and other Intel-based servers.

http://www.ibm.com/software/webservers/appserv/version40.html

Application integration

The following are application integration solutions for Linux.

WebSphere MQ

Connects applications to each other. It connects almost everything to almost anything, to form one efficient and coherent enterprise-wide or community-wide system to streamline your processes and improve time to market. It is available on IBM xSeries, zSeries, and other Intel-based servers.

http://www.ibm.com/software/ts/mqseries/messaging/

WebSphere MQ Everyplace

Extends WebSphere MQ to mobile workers using laptops, PDAs, and smart phones. It is available on IBM xSeries and other Intel-based servers.

http://www.ibm.com/software/ts/mqseries/everyplace/

Industry solutions

The following are industry solutions for Linux.

WebSphere Business Components Composer

Provides a set of tools, Java components, and services that accelerate the building of multichannel banking applications that access transactional systems. Composer also enables branch transformation initiatives within retail banking, including the construction of branch applications and teller systems.

http://www-3.ibm.com/software/webservers/components/

IBM Bank Teller Business Components

IBM Bank Teller Business Components provide a set of application enabling components specialty tailored to allow rapid deployment of branch teller transactions and solutions. Bank Teller Business Components are designed to integrate seamlessly with Business Components Composer in the WebSphere software branch transformation solution.

http://www-3.ibm.com/software/webservers/components/bankteller/

IBM Linux Technology Center Omni drivers

The Omni printer driver provides support for over 400 printers using the Ghostscript framework. In addition, it provides a model for dynamically loading printer drivers, creating new devices by editing device description files, and simplifies new printer driver development by allowing for the subclassing of previous device features.

http://oss.software.ibm.com/linux/

Development tool

The following is a development tool for Linux.

WebSphere Studio Application Developer

Provides customers with an integrated development environment for building, testing, integrating, and deploying Java 2 Platform, Enterprise Edition (J2EE platform) applications that rapidly grow and adapt to meet the most stringent business demands.

http://www.ibm.com/software/ad/studioappdev

Presentation

The following are presentation solutions for Linux.

WebSphere Transcoding Publisher

Dynamically adapts Web content and optimizes it for delivery to pervasive devices.

http://www.ibm.com/software/webservers/transcoding/

WebSphere Personalization

Customizes Web sites to suit the interests and needs of site visitors.

http://www.ibm.com/software/webservers/personalization/

Deployment

The following is a deployment solution for Linux.

WebSphere Edge Server

Provides an integrated solution for load balancing, static and dynamic caching, application offload, content distribution, enhanced security, and transactional Quality of Service, all under centralized administrative and application control.

http://www.ibm.com/software/webservers/edgeserver/

Integration

The following are integration solutions for Linux.

WebSphere Host On-Demand

Extends host applications to browsers and is available on IBM zSeries.

http://www.ibm.com/software/webservers/hostondemand/

IBM Screen Customizer

Provides graphical user interfaces for IBM zSeries and iSeries host screens. It is available on IBM xSeries and other Intel-based servers.

http://www.ibm.com/software/network/screencustomizer/

Lotus Domino for Linux

A successful company's most valuable assets are its people. Harness the brainpower of your team with Web-enabled solutions that allow smart collaboration between employees, customers, partners, and suppliers. Using Lotus Domino for Linux, you can create a connected community with easy access to ideas.

Additional information about Lotus products for Linux can be found at:

`http://www.lotus.com/home.nsf/welcome/domino`

Collaboration

The following is a collaboration solution for Linux.

Lotus Domino

Provides collaboration, messaging, group calendaring, discussion, and custom applications. It is available on IBM xSeries and other Intel-based servers.

`http://www.lotus.com/products/r5web.nsf/webhome/nr5serverhp-new`

Workflow

The following is a workflow solution for Linux.

Lotus Workflow

Provides access to workflow management services and tools for Web applications.

`http://www.lotus.com/products/domworkflow.nsf`

IBM Tivoli software

Tivoli solutions for Linux provide you with the tools you need to simplify the management of your technology infrastructure. Using Autonomic Computing, Tivoli automates mundane management tasks, freeing up your IT staff to focus on the more important challenges facing your business.

Tivoli software is Linux friendly, providing your company with an array of cross-platform management tools that cut costs, speed return on investment, and improve IT responsiveness. The same tools and controls administrators use today to manage existing systems are now extended to cover newer Linux servers, integrating them with your current IT infrastructure.

To explore the Tivoli product family, see:

http://www.tivoli.com/

Security

The following are security solutions for Linux.

IBM Tivoli Access Manager for e-business

Provides end-to-end security for e-business, including Web single signon, distributed Web-based administration, and policy-based security. It is available on IBM zSeries servers.

http://www.tivoli.com/products/index/access-mgr-e-bus/

IBM Tivoli Access Manager for Operating Systems

Provides end-to-end access control, application-level data protection, and centralized security policy management for the IBM WebSphere MQ environment. It is available on IBM xSeries and other Intel-based servers.

http://www.tivoli.com/products/index/access-mgr-operating-sys/

IBM Tivoli Identity Manager

Centrally coordinates the creation of user accounts and automates the workflow for approval and the provisioning of resources.

http://www.tivoli.com/products/index/identity-mgr/

Storage

The following are storage solutions for Linux.

IBM Tivoli Storage Manager

Automates data backup and restore functions, supports a broad range of platforms and storage devices, and centralizes storage management operations.

http://www.tivoli.com/products/index/storage-mgr/

IBM Tivoli Storage Manager Enterprise Edition

Delivers an integrated data-protection solution for large enterprises, including LAN-free, SAN-enabled data transfers, hierarchical storage management, and disaster recovery functions.

http://www.tivoli.com/products/index/storage-mgr-enterprise/

Performance and availability

The following are performance and availability solutions for Linux.

IBM Tivoli Enterprise Console

Provides a centralized point of control that keeps your IT staff in close and efficient control of, and provides automated corrective actions to, events happening across all systems and networks. It is available on IBM zSeries, xSeries, and other Intel-based servers.

http://www.tivoli.com/products/index/enterprise-console/

IBM Tivoli Monitoring

Provides monitoring for essential system resources to detect bottlenecks and potential problems and to automatically recover from critical situations. It is available on IBM zSeries, xSeries, and other Intel-based servers.

http://www.tivoli.com/products/index/monitor/

Configuration and operations

The following are configuration and operations solutions for Linux.

IBM Tivoli Workload Scheduler

Automates, monitors, and controls the flow of work through your enterprise's entire IT infrastructure on both local and remote systems. It is available on IBM zSeries, xSeries, and other Intel-based servers.

http://www.tivoli.com/products/index/scheduler/

IBM Tivoli Workload Scheduler for Applications

Manages enterprise scheduling from a single point and integrates with solutions such as mySAP.com. It is available on IBM xSeries and other Intel-based servers.

http://www.tivoli.com/products/index/scheduler-apps/

IBM Tivoli Configuration Manager

Delivers an integrated solution for deploying software and for tracking hardware and software configurations across an enterprise. It is available on IBM zSeries, xSeries, and other Intel-based servers.

http://www.tivoli.com/products/index/config-mgr/

Related publications

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

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 152.

- *EJB Development with VisualAge for Java for WebSphere Application Server*, SG24-6144
- *Enterprise Business Portals with IBM Tivoli Access Manager*, SG24-6556
- *IBM CBConnector Cookbook Collection: CBConnector Bank User Guide*, SG24-5121
- *Network Computing Framework Component Guide*, SG24-2119

Other resources

This publication is also relevant as a further information source:

- *Patterns for e-business: A Strategy for Reuse*, by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, George Galambos, IBM Press, 2001, ISBN 1931182027

Referenced Web sites

These Web sites are also relevant as further information sources:

- IBM Software - General

  `http://www.ibm.com/software/`
- IBM Software - Linux

  `http://www.ibm.com/software/linux/`
- IBM Linux Technology Center

  `http://oss.software.ibm.com/linux/`
- Lotus Domino software

  `http://www.lotus.com/home.nsf/welcome/domino`

- IBM Tivoli Software

  `http://www.tivoli.com/`
- Appache HTTP server project

  `http://httpd.apache.org/`
- Appache Jakarta Project - Tomcat application server

  `http://jakarta.apache.org/tomcat/`
- Eclipse open source project for an integrated development environment

  `http://www.eclipse.org/`
- J/eXtensions for Financial Services (J/XFS)

  `http://www.jxfs.com/`
- MySQL open source database project

  `http://www.mysql.com`
- IBM Patterns for e-business

  `http://www.ibm.com/developerWorks/patterns/`
- Red Hat Linux

  `http://www.redhat.com/apps/download/`
- Samba open source file SMB and CIFS project

  `http://www.samba.org`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

`ibm.com``/redbooks`

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

Linux and Branch Banking

Linux and Branch Banking

Branch banking transformation trends

An IBM Patterns for e-business approach

The vital role of Linux

The banking industry is undergoing a major transformation to e-business, supporting a multichannel model for its delivery of services. As part of this transformation, there is a major focus on the branch IT infrastructure.

In this very competitive environment, banks are looking for solutions that are cost effective, provide a high quality of service, and allow them to speed new products and services to market. In addition, they need to have a flexible and open environment in order to absorb the inevitable changes that occur over time.

At the same time, Linux is gaining popularity and credibility as a robust and stable operating environment for many business-critical functions.

This IBM Redbook surveys the current trends in branch banking, describes in detail an IBM Patterns for e-business approach to designing the branch infrastructure of the future, and provides the reader with an understanding of how and where Linux can play a key role in branch banking infrastructures.

SG24-6909-00 ISBN 0738428337