chapter 1

Systems Architecture

The first requisite for success is the ability to apply your physical and mental energies to one problem incessantly without growing weary.

Thomas Edison (1847–1931)

Systems architecture can best be thought of as both a process and a discipline to produce efficient and effective information systems.

It is a process because a set of steps is followed to produce or change the architecture of a system.

It is a discipline because a body of knowledge informs people as to the most effective way to design.

A system is an interconnected set of machines, applications, and network resources. Systems architecture unifies that set by imposing structure on the system. More importantly, this structure aligns the functionality of the system with the goals of the business.

Systems architecture encompasses the infrastructural layer of Figure 1–1.

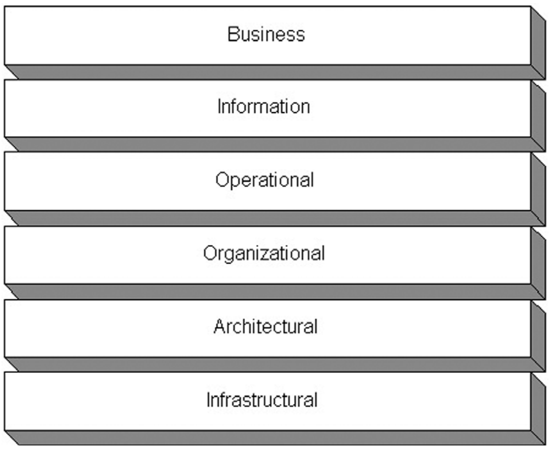


Figure 1–1
Enterprise architecture model.

The basic purpose of systems architecture is to support the higher layers of the enterprise architecture. In many companies, the software and hardware represent a significant portion of the enterprise's total assets. It is important that enterprise architects do not equate their duties with the objects, the applications, or the machines that comprise their domain. The fundamental purpose is to support and further the business objectives of the enterprise. Hardware and software objects are fundamentally transient and exist only to further the purposes of the business.

Systems architecture is also used as part of the process of keeping the enterprise architecture aligned with the business goals and processes of the organization. It is important to understand the technical details of the infrastructure and the applications running within it but also to have the knowledge to participate in the process of architectural change with the enterprise architectural team. That involves the following:

- Defining the structure, relationships, views, assumptions, and rationales for the existing systems architecture and the changes in relationships, views, assumptions, and rationales that are involved in any changes required for moving from what is to what is desired.
- Creation of models, guides, templates, and design standards for in use in developing the systems architecture.

Canaxia Brings an Architect on Board

Let us set the stage for further discussion by focusing on Canaxia. As part of Kello James's engineering of Canaxia's enterprise architecture, James has brought on board an architect, the first one in Canaxia's history.

We are sitting in on a seminal meeting with some of Canaxia's C-level executives and Myles Standish, the new architect. He is being introduced to a semi-skeptical audience. It will be Myles's task in this meeting to establish the value to Canaxia of systems architecture and, by extension, a systems architect. As part of his charter, Myles was given oversight responsibility for recommended changes in the physical infrastructure and for the application structure of the enterprise. That means he will be part of the architecture team as Canaxia evolves.

Myles begins his remarks by giving a quick rundown of what his research has uncovered about the state of Canaxia's systems architecture. The discussion proceeded as follows:

- Myles noted that a large part of Canaxia's fixed assets, like those of most large corporations, is in the applications and machines that comprise Canaxia's systems architecture. The corporations in Canaxia's business environment that best manage their systems architectures will thereby gain a competitive advantage.
- He noted that the majority of the budget of the information technology (IT) department is swallowed by costs related to Canaxia's infrastructure. At the present time, most of these costs are fixed. If Canaxia wishes to gain any agility in managing its information technology resources, it will have to put into its systems architecture the necessary time, effort, and attention to sharply reduce those costs.

- Major business information technology functionality at Canaxia, Myles pointed out, is divided among silo-applications that were, in some cases, created three decades ago. Here, Myles was interrupted by an executive vice president who asked, "You are not going to tell us that we have to rewrite or replace these applications are you? We have tried that." Myles firmly replied that he had no plans to replace any of the legacy applications. Rather than replace, he was going to break down the silos and get the applications talking to each other.
- The terrorist attack that completely destroyed the World Trade Center in New York City has put disaster recovery on the front burner for Canaxia. Systems architecture issues are crucial to a disaster recovery plan (DRP). (The systems architecture aspects of data recovery planning are discussed later in this chapter.) As part of his presentation, Myles briefed the business side on the systems architecture section of a disaster recovery plan that had been developed by the architecture group.
- Myles then discussed the issues surrounding storage. He asked the executives if they realized that, at the rate that Canaxia's data storage costs were rising, by 2006 the cost of storage would equal the current IT budget. The stunned silence from the executives in the room told Myles that this was new information for the executive team. Myles then said he would also ask them for their support in controlling those costs and thereby gaining a further competitive advantage for Canaxia.

Data storage and the issues surrounding it are discussed later on in this chapter.

- The recent flurry of bankruptcies among large, high-profile companies has led to the imposition of stringent reporting rules on the top tier of publicly owned companies in the United States. These rules demand almost real-time financial reporting capabilities from IT departments. Canaxia is one of the corporations subject to the new reporting stringencies. Myles bluntly told the gathered managers that the current financial system was too inflexible to provide anything more than the current set of reports and that the fastest those reports can be turned out is monthly. Myles next stated that accelerated financial reporting would be a high priority on his agenda. This brought nods from the executives.
- Myles also brought up the issue that visibility into the state of Canaxia's infrastructure is an important component of a complete view of the business. Myles intended to build a reporting infrastructure that allowed line managers and upper-level management to have business intelligence at their disposal so they could ascertain if the system's infrastructure was negatively impacting important business metrics, such as order fulfillment. In the back of his mind, Myles was certain that Canaxia's antiquated systems architecture was the root cause of some of the dissatisfaction customers

> were feeling about the auto manufacturer. He knew that bringing metrics of the company's infrastructure performance into the existing business intelligence (BI) system was an important component in restoring customer satisfaction levels.

The cumulative impact of thousands of low-level decisions has encumbered Canaxia with an infrastructure that doesn't serve business process needs and yet devours the majority of the IT budget. In many cases, purchase decisions were made just because the technology was "neat" and "the latest thing." "Under my tenure that will stop," Myles stated firmly. "We are talking about far too much money to allow that. Moving forward, how well the project matches with the underlying business process, and its return on investment (ROI) will determine where the system infrastructure budget goes."

Then the subject of money—the budget issue—came up. "You are not expecting a big increase in budget to accomplish all this, are you? No money is available for budget increases of any size," Myles was informed. He responded by pointing out that the system infrastructure of an enterprise should be looked at as an investment: a *major* investment, an investment whose value should increase with time, not as an object whose value depreciates. The underperforming areas need to be revamped or replaced. The areas that represent income, growth, or cost-reduction opportunities need to be the focus.

After the meeting, both Kello and Myles felt greatly encouraged. Both had been involved in enterprise architecture efforts that had faltered and failed. They both knew that the prime reason for the failure was an inability to engage the C-level managers in the architecture process and keep them involved. This meeting made it clear that, as far as Canaxia's top management was concerned, the issues of increased financial reporting, disaster recovery, and cost containment were on the short list of concerns. Myles and Kello knew that they would have the support they needed to build their architecture. They also knew that that architecture would have to translate into fulfillment of Canaxia's business concerns.

Architectural Approach to Infrastructure

It is important to take a holistic view of all the parts in a system and all their *interactions*.

We suggest that the most effective path to understanding a system is to build a catalogue of the *interfaces* the system exposes or the *contracts* the system fulfills. Focusing on the contract or the interface makes it much easier to move away from a preoccupation with machines and programs and move to a focus on the enterprise functionality that the systems must provide. Taken higher up the architecture stack, the contract should map directly to the business process that the system objects are supposed to enable.

When done properly, creating and implementing a new systems architecture or changing an existing one involves managing the process by which the organization achieves the new architecture as well as managing the assembly of the components that make up that architecture. It involves managing the disruption to the stakeholders, as well as managing the pieces and parts. The goal is to improve the process of creating and/or selecting new applications and the process of integrating them into existing systems. The payoff is to reduce IT operating costs by improving the efficiency of the existing infrastructure.

Additional Systems Architecture Concerns

The following are some considerations to take into account when building a systems architecture to support the enterprise:

- The business processes that the applications, machines, and network are supposed to support. This is the primary concern of the systems architecture for an enterprise. Hence, an architect needs to map all the company's business processes to the applications and infrastructure that is supposed to support it. This mapping should be taken down the lowest level of business requirements. In all likelihood, a 100 percent fit between the business requirements and the implementation will not be possible.
- Any part of the overall architecture that is being compromised. Areas that often suffer are the data and security architectures and data quality. The data architecture suffers because users don't have the knowledge and expertise to understand the effects of their changes to procedure. Security architecture suffers because security is often viewed by employees as an unnecessary nuisance. Data integrity suffers because people are pressured to produce more in less time and the proper cross-checks on the data are not performed.
- The stakeholders in the systems architecture. The people involved, including the following:
  - The individuals who have created the existing architecture
  - The people tasked with managing and enhancing that architecture
  - The people in the business who will be affected positively and negatively by any changes in the current systems architecture
  - The business's trading partners and other enterprises that have a stake in the effective functioning and continued existence of this corporation
  - Any customers dealt with directly over the Internet

The needs and concerns of the stakeholders will have a large impact on what can be attempted in a systems architecture change and how successful any new architecture will be.

- The context that the new system architecture encompasses. In this text, *context* refers to the global enterprise realities in which systems architects find themselves. If you're a company that is limited to a single country, language, and currency, you're the context will probably be relatively simple. Canaxia is a global corporation, and its stakeholders speak many different languages and live in a variety of time zones; thus its corporate culture is a complex mix of many regional cultures. This diversity is, on balance, a positive factor that increases Canaxia's competitiveness.

- The data with which the systems architecture deals.

See Chapter 11, Data Architecture, for full details about data architecture considerations.

Working with Existing Systems Architectures

All companies have not had a single person or group that has had consistent oversight of the businesses systems architecture. This is the situation that Myles found himself in. Canaxia's systems architecture wasn't planned. The company had gone through years of growth without spending any serious time or effort on the architecture of its applications, network, or machines. Companies that have not had a systems architecture consistently applied during the growth of the company will probably have what is known as a *stovepipe architecture*.

Suboptimal architectures are characterized by a hodgepodge of equipment and software scattered throughout the company's physical plant. This equipment and software were obtained for short-term, tactical solutions to the corporate problem of the moment. Interconnectivity was an afterthought at best. Stovepipe architecture is usually the result of large, well-planned projects that were designed to fill a specific functionality for the enterprise. These systems will quite often be mission critical, in that the effective functioning of the company is dependent upon them. Normally these systems involve both an application and the hardware that runs it. The replacement of these systems is usually considered prohibitively expensive. A number of issues accompany stovepipe architecture:

- The systems don't fit well with the business process that they are tasked with assisting. This can be due to any of the following:
  - The software design process used to build the application imperfectly capturing the business requirements during the design phase
  - The inevitable change of business requirements as the business's competitive landscape changes

See Chapter 5, Methodology Overview, Chapter 6, Enterprise Unified Process, and Chapter 8, Agile Modeling, for more depth on software development processes.

- The monolithic design of most stovepipe applications making it difficult to rationalize discrepancies between business processes and application functionality.
- The data not integrating with the enterprise data model, usually due to vocabulary, data format, or data dictionary issues.

See Chapter 11, Data Architecture, for a more complete discussion of the data architecture problems that can arise among stovepipe applications.

- Stovepipe applications definitely not being designed to integrate into a larger system, be it a result of enterprise application integration (EAI) or supply-chain management.
- It is difficult to obtain the information necessary for business intelligence or business process management.
- The marriage of application and hardware/operating system that is part of some stovepipe applications creating a maintenance and upgrade inflexibility that results in stovepipe applications having a high total cost of ownership (TCO).

Many times it is useful to treat the creation of enterprise systems architecture as a domain-definition problem. In the beginning, you should divide the problem set so as to exclude the less-critical areas. This will allow you to concentrate your efforts on critical areas. This can yield additional benefits because smaller problems can sometimes solve themselves or become moot while you are landing the really big fish. In the modern information services world, the only thing that is constant is change. In such a world, time can be your ally and patience the tool to reap its benefits.

Systems Architecture Types

Few architects will be given a clean slate and told to create a new systems architecture from scratch. Like most architects, Myles inherited an existing architecture, as well as the people who are charged with supporting and enhancing it and the full set of dependencies among the existing systems and the parts of the company that depended on them. It must be emphasized that the existing personnel infrastructure is an extremely valuable resource.

Following is an enumeration of the systems architecture types, their strengths, their weaknesses, and how to best work with them.

Legacy Applications

Legacy applications with the following characteristics can be extremely problematic:

- A monolithic design. Applications that consist of a series of processes connected in an illogical manner will not play well with others.
- Fixed and inflexible user interfaces. A character-based "green screen" interface is a common example of a legacy user interface (UI). Interfaces such as these are difficult to replace with browser-based interfaces or to integrate into workflow applications.
- Internal, hard-coded data definitions. These data definitions are often specific to the application and don't conform to an enterprise data model approach. Furthermore, changing them can involve refactoring all downstream applications.

- Business rules that are internal and hard-coded. In such a situation, updates to business rules caused by changes in business processes require inventorying applications to locate all the relevant business rules and refactoring affected components.
- Applications that store their own set of user credentials. Application-specific credential stores can block efforts to integrate the application with technologies to enable single sign-on and identity management.

While many legacy applications have been discarded, quite a few have been carried forward to the present time. The surviving legacy applications will usually be both vital to the enterprise and considered impossibly difficult and expensive to replace.

This was Canaxia's situation. Much of its IT department involved legacy applications. Several mainframe applications controlled the car manufacturing process. The enterprise resource planning (ERP) was from the early 1990s. The inventory and order system, a mainframe system, was first created in 1985. The customer relationship management (CRM) application was recently deployed and built on modular principles. The legacy financials application for Canaxia was of more immediate concern.

Canaxia was one of the 945 companies that fell under Securities and Exchange Commission (SEC) Order 4–460, so the architecture team knew it had to produce financial reports that were much more detailed than the system currently produced.

The existing financials would not support these requirements. The architecture team knew that it could take one of the following approaches:

- It could replace the legacy financial application.
- It could perform extractions from the legacy application into a data warehouse of the data necessary to satisfy the new reporting requirements.

Replacing the existing financial application was looked at long and hard. Considerable risk was associated with that course. The team knew that if it decided to go the replace route, it would have to reduce the risk of failure by costing it out at a high level. However, the team had other areas that were going to require substantial resources, and it did not want to spend all its capital on the financial reporting situation.

The data extraction solution would allow the legacy application to do what it is used to doing when it is used to doing it. The data mining approach buys Canaxia the following:

- Much greater reporting responsiveness. Using a data warehouse, it is possible to produce an accurate view of the company's financials on a weekly basis.
- More granular level of detail. Granular access to data is one of the prime functions of data warehouses. Reconciliation of the chart of

accounts according to the accounting principles of the enterprise is the prime function of legacy financial applications, not granular access to data.

- Ad hoc querying. The data in the warehouse will be available in multidimensional cubes that users can drill into according to their needs. This greatly lightens the burden of producing reports.
- Accordance with agile principles of only doing what is necessary. You only have to worry about implementing the data warehouse and creating the applications necessary to produce the desired reports. You can rest assured that the financial reports that the user community is accustomed to will appear when needed and in the form to which they are accustomed.

In terms of the legacy applications that Myles found at Canaxia, he concurred with the decision to leave them intact. In particular, he was 100 percent behind the plan to leave the legacy financial application intact, and he moved to implement a data warehouse for Canaxia's financial information.

The data warehouse was not a popular option with management. It was clearly a defensive move and had a low, if not negative, ROI. At one point in the discussion of the cost of the data warehouse project, the architecture team in general and Myles in particular were accused of "being gullible and biased toward new technology." This is a valid area to explore any time new technology is being brought on board. However, the architecture team had done a good job of due diligence and could state the costs of the project with a high degree of confidence. The scope of the project was kept as small as possible, thereby substantially decreasing the risk of failure. The alternative move, replacing the legacy application, was substantially more expensive and carried a higher degree of risk. Also, it was true that bringing the financial data into the enterprise data model opened the possibility of future synergies that could substantially increase the ROI of the data warehouse. In general terms, the good news with legacy applications is that they are extremely stable and they do what they do very well. Physical and data security are usually excellent. Less satisfying is that they are extremely inflexible. They expect a very specific input and produce a very specific output. Modifications to their behavior usually are a major task. In many cases, the burden of legacy application support is one of the reasons that so little discretionary money is available in most IT budgets.

Most architects will have to work with legacy applications. They must concentrate on ways to exploit their strengths by modifying their behavior and focus on the contracts they fulfill and the interfaces they can expose.

Changes in any one application can have unexpected and unintended consequences to the other applications in an organization. Such situations mandate that you isolate legacy application functionality to a high degree. At some point, a function will have been decomposed to the point where you will understand all its inputs and outputs. This is the level at which to externalize legacy functionality.

Client/Server Architecture

Client/server architecture is based on dividing effort into a client application, which requests data or a service and a server application, which fulfills those requests. The client and the server can be on the same or different machines. Client/server architecture filled a definite hole and became popular for a variety of reasons.

Access to knowledge was the big driver in the rise of client/server. On the macro level, with mainframe applications, users were given the data, but they wanted knowledge. On the micro level, it came down to reports, which essentially are a particular view on paper of data. It was easy to get the standard set of reports from the mainframe. To get a different view of corporate data could take months for the preparation of the report. Also, reports were run on a standard schedule, not on the user's schedule.

Cost savings was another big factor in the initial popularity of client/server applications. Mainframe computers cost in the six- to seven-figure range, as do the applications that run on them. It is much cheaper to build or buy something that runs on the client's desktop.

The rise of fast, cheap network technology also was a factor. Normally the client and the server applications are on separate machines, connected by some sort of a network. Hence, to be useful, client/server requires a good network. Originally, businesses constructed local area networks (LANs) to enable file sharing. Client/server architecture was able to utilize these networks. Enterprise networking has grown to include wide area networks (WANs) and the Internet. Bandwidth has grown from 10 Mbs Ethernet and 3.4 Mbs token ring networks to 100 Mbs Ethernet with Gigabit Ethernet starting to make an appearance. While bandwidth has grown, so has the number of applications chatting on a given network. Network congestion is an ever-present issue for architects.

Client/server architecture led to the development and marketing of some very powerful desktop applications that have become an integral part of the corporate world. Spreadsheets and personal databases are two desktop applications that have become ubiquitous in the modern enterprise.

Client/server application development tapped the contributions of a large number of people who were not programmers. Corporate employees at all levels developed some rather sophisticated applications without burdening an IT staff.

The initial hype that surrounded client/server revolution has pretty much died down. It is now a very mature architecture. The following facts about it have emerged:

- The support costs involved in client/server architecture have turned out to be considerable. A large number of PCs had to be bought and supported. The cost of keeping client/server applications current and of physically distributing new releases or new applications to all the corporate desktops that required them came as a shock to many IT departments. Client/server applications are another one of the reasons that such a large portion of most IT budgets is devoted to fixed costs.

- Many of the user-built client/server applications were not well designed. In particular, the data models used in the personal databases were often completely unnormalized. In addition, in a significant number of cases, the data in the desktop data store were not 100 percent clean.
- The huge numbers of these applications and the past trend of decentralization made it extremely difficult for an architect to locate and inventory them.

The rapid penetration of client/server applications into corporations has been expedited by client/server helper applications, especially spreadsheets. At all levels of the corporate hierarchy, the spreadsheet is the premier data analysis tool. Client/server development tools allow the direct integration of spreadsheet functionality into the desktop client. The importance of spreadsheet functionality must be taken into account when architecting replacements for client/server applications. In most cases, if the user is expecting spreadsheet capabilities in an application, it will have to be in any replacement.

In regard to client/server applications, the Canaxia architecture team was in the same position as most architects of large corporations in the following ways:

- Canaxia had a large number of client/server applications to be supported. Furthermore, the applications would have to be supported for many years into the future.
- Canaxia was still doing significant amounts of client/server application development. Some were just enhancements to existing applications, but several proposals focused on developing entirely new client/server applications.
- One of the problems that Myles has with client/server architecture is that it is basically one–to–one: a particular application on a particular desktop talking to a particular server, usually a database server. This architecture does not provide for groups of corporate resources to access the same application at the same time. The architecture team wanted to move the corporation in the direction of distributed applications to gain the cost savings and scalability that they provided.

The architecture team knew that it had to get some sort of grip on Canaxia's client/server applications. It knew that several approaches could be taken:

- Ignore them. This is the recommended approach for those applications that have the following characteristics:
  - Are complex and would be difficult and/or expensive to move to thin-client architecture.
  - Have a small user base, usually meaning the application is not an IT priority.
  - Are stable, that is, don't require much maintenance or a steady stream of updates, usually meaning that support of this application is not a noticeable drain on the budget.

- ○ Involves spreadsheet functionality. Attempting to produce even a small subset of a spreadsheet's power in a thin client would be a very difficult and expensive project. In this case, we suggest accepting the fact that the rich graphical user interface (GUI) benefits that the spreadsheet brings to the table are the optimal solution.
- Put the application on a server and turn the users' PCs into dumb terminals. Microsoft Windows Server, version 2000 and later, is capable of operating in multi-user mode. This allows the IT department to run Windows applications on the server by just providing screen updates to individual machines attached to it. Putting the application on a server makes the job of managing client/server applications several orders of magnitudes easier.
- Extract the functionality of the application into a series of interfaces. Turn the client/server application into a set of components. Some components can be incorporated into other applications. If the client/server application has been broken into pieces, it can be replaced a piece at a time, or the bits that are the biggest drain on your budget can be replaced, leaving the rest still functional.

Service-Oriented Architecture (SOA) is a formalized method of integrating applications into an enterprise architecture. For more information on SOA, see Chapter 3, Service-Oriented Architecture.

- Client/server applications are still the preferred solution in many situations. If the application requires a complex GUI, such as that produced by Visual Basic, PowerBuilder, or a Java Swing application or applet, many things can be done that are impossible to duplicate in a browser application, and they are faster and easier to produce than in HTML. Applications that perform intensive computation are excellent candidates. You do not want to burden your server with calculations or use up your network bandwidth pumping images to a client. It is far better to give access to the data and let the CPU on the desktop do the work. Applications that involve spreadsheet functionality are best done in client/server architecture. Excel exposes all its functionality in the form of Common Object Model (COM) components. These components can be transparently and easily incorporated into VB or Microsoft Foundation Classes (MFC) applications. In any case, all new client/server development should be done using interfaces to the functionality rather than monolithic applications.
- Move them to thin-client architecture. At this time, this usually means creating browser-based applications to replace them. This is the ideal approach to those client/server applications that have an important impact on the corporate data model. It has the important side effect of helping to centralize the data resources contained in the applications.

The Canaxia architecture team decided to take a multistepped approach to its library of client/server applications.

- About a third of the applications clearly fell in the "leave them be" category. They would be supported but not enhanced.
- Between 10 and 15 percent of the applications would no longer be supported.
- The remaining applications are candidates for replacement using a thin-client architecture.

One matter that is often overlooked in client/server discussions is the enterprise data sequestered in the desktop databases scattered throughout a business. One can choose among several approaches when dealing with the important corporate data contained in user data stores:

- Extract it from the desktop and move it into one of the corporate databases. Users then obtain access via Open Database Connectivity (ODBC) or (JDBC) supported applications.
- For the client/server applications that are moved into the browser, extraction may be necessary.

While client/server architecture is not a good fit for current distributed applications, it still has its place in the modern corporate IT world. We are staunch advocates of using interfaces rather than monolithic applications. As we have recommended before, concentrate on the biggest and most immediate of your problems.

Thin-Client Architecture

T*hin-client architecture* is one popular approach to decoupling presentation from business logic and data. Originally *thin clients* were a rehash of time-share computing with a browser replacing the dumb terminal. As the architecture has matured, in some cases the client has gained responsibility.

As noted, client/server applications have substantial hidden costs in the form of the effort involved to maintain and upgrade them. Dissatisfaction with these hidden costs was one of the motivating factors that led to the concept of a "thin client." With thin-client systems architecture, all work is performed on a server.

One of the really attractive parts of thin-client architectures is that the client software component is provided by a third party and requires no expense or effort on the part of the business. The server, in this case a Web server, serves up content to users' browsers and gathers responses from them. All applications run on either the Web server or on dedicated application servers. All the data are on machines on the business's network.

Thin-client architecture has several problematic areas:

- A thin-client architecture can produce a lot of network traffic. When the connection is over the Internet, particularly via a modem, round-trips from the client to the server can become unacceptably long.
- The architecture of the Web was initially designed to support static pages. When the next request for a static page comes in, the server doesn't care about the browsing history of the client.
- Control of which browser is used to access the thin-client application is sometimes outside the control of the team developing the application. If the application is only used within a corporation, the browser to be used can be mandated by the IT department. If the application is accessed by outside parties or over the Internet, a wide spectrum of browsers may have to be supported. Your choices in this situation are to give up functionality by programming to the lowest common denominator or to give up audience by developing only for modern browsers.
- Users demand rich, interactive applications. HTML is insufficient to build rich, interactive GUI applications. When developing thin-client applications, the development team must strive to provide just enough functionality for users to accomplish what they want and no more. Agile methods have the correct approach to this problem and do no more than what clients request.
- Data must be validated. The data input by the user can be validated on the client and, if there are problems with the page, the user is alerted by a pop-up. The data also can be sent to the server and validated there. If problems arise, the page is resent to the user with a message identifying where the problems are.
- The most common Web development tools are usually enhanced text editors. More advanced Web rapid application development (RAD) tools exist, such as DreamWeaver and FrontPage; however, they all fall short on some level, usually due to the facts that they only automate the production of HTML and they give little or no help with JavaScript or with the development of server-side HTML generators such as Java server pages (JSPs) and Java servlets.
- Server resources can be stretched thin. A single Web server can handle a surprising number of connections when all it is serving up are static Web pages. When the session becomes interactive, the resources consumed by a client can rise dramatically.

The keys to strong thin-client systems architecture lie in application design. For example, application issues such as the amount of information carried in session objects, the opening database connections, and the time spent in SQL queries can have a big impact on the performance of a thin-client architecture. If the Web browser is running on a desktop machine, it may be possible to greatly speed up performance by enlisting the computing power of the client machine.

Data analysis, especially the type that involves graphing, can heavily burden the network and the server. Each new request for analysis involves a network trip to send the request to the server. Then the server may have to get the data, usually from the machine hosting the database. Once the data are in hand, CPU-intensive processing steps are required to create the graph. All these operations take lots of server time and resources. Then the new graph has to be sent over the network, usually as a fairly bulky object, such as a GIF or JPEG. If the data and an application to analyze and display the data can be sent over the wire, the user can play with the data to his or her heart's content and the server can be left to service other client requests. This can be thought of as a "semithin-client" architecture.

Mobile phones and personal desktop assistants (PDAs) can be considered thin clients. They have browsers that you can use to allow interaction between these mobile devices and systems within the enterprise.

The rise of the browser client has created the ability and the expectation that businesses expose a single, unified face to external and internal users. Inevitably, this will mean bringing the business's legacy applications into the thin-client architecture. Integrating legacy applications into thin-client architecture can be easy or it can be very difficult. Those legacy applications that exhibit the problematic areas discussed previously can be quite challenging. Those that are fully transactional and are built on relational databases can often be an easier port than dealing with a client/server application.

Exposing the functionality of the legacy systems via interfaces and object wrappering is the essential first step. Adapting the asynchronous, batch-processing model of the mainframe to the attention span of the average Internet user can be difficult. We discuss this in greater detail later in this chapter.

Using Systems Architecture to Enhance System Value

As software and hardware systems age, they evolve as maintenance and enhancement activities change them. Maintenance and enhancement can be used as an opportunity to increase the value of a system to the enterprise. The key to enhancing the value of systems architecture via maintenance and enhancement is to use the changes as an opportunity to modify stovepipe applications into reusable applications and components. This will make your system more agile because you will then be free to modify and/or replace just a few components rather than the entire application. In addition, it will tend to "future proof" the application because the greater flexibility offered by components will greatly increase the chances that existing software will be able to fulfill business requirements as they change.

The best place to begin the process is by understanding all the contracts that the stovepipe fulfills. By *contracts*, we mean all the agreements that the stovepipe makes with applications that wish to use it. Then, as part of the enhancement or maintenance, the stovepipe externalizes the contract via an interface. As this process is repeated with a monolithic legacy application, it begins to function as a set of components. For example, some tools consume COBOL copy books and produce an "object wrapper" for the modules described in the copy books.

Messaging technology is a useful technology for integrating legacy functions into your systems architecture. With messaging, the information required for the function call is put on the wire and the calling application either waits for a reply (pseudosynchronous) or goes about its business (asynchronous). Messaging is very powerful in that it *completely* abstracts the caller from the called function. The language in which the server program is written, the actual data types it uses, and even the physical machine upon which it runs are immaterial to the client. With asynchronous messaging calls, the server program can even be offline when the request is made and it will fulfill it when it is returned to service. Asynchronous function calls are very attractive when accessing legacy applications. They can allow the legacy application to fulfill the requests in the mode with which it is most comfortable: batch mode.

To be of the greatest utility, object wrappers should expose a logical *set* of the underlying functions. We think you will find that exposing every function that a monolithic application has is a waste of time and effort. The functionality to be exposed should be grouped into logical units of work, and that unit of work is what should be exposed.

What is the best way to expose this functionality? When creating an object wrapper, the major consideration should be the manner in which your systems architecture plan indicates that this interface or contract will be utilized. The effort should be made to minimize the number of network trips.

Wrappering depends on distributed object technology for its effectiveness. Distributed objects are available under Common Object Request Broker Architecture (CORBA), Java, .Net, and by using Messaging Oriented Middleware (MOM) technology. Distributed objects rely upon an effective network (Internet, intranet, or WAN) to operate effectively.

In addition to increasing the value of a stovepipe application to other segments of the enterprise, exposing the components of that application provide the following opportunities:

- The possibility of sunsetting stagnant or obsolete routines and the incremental replacement of them by components that makes more economic sense to the enterprise.
- The possibility of plugging in an existing component that is cheaper to maintain in place of the legacy application routine.
- Outsourcing functionality to an application services provider (ASP).

If the interface is properly designed, any application accessing that interface will be kept totally ignorant of how the business contract is being carried out. Therefore, it is extremely important to design them properly. The Canaxia architecture group has an ongoing project to define interfaces for legacy applications and important client/server applications that were written in-house.

The best place to start when seeking an understanding of the interfaces that can be derived from an application is not the source code but the user manuals. Another good source of knowledge of a legacy application's contracts is the application's customers, its users, who usually know exactly what the application is supposed to do and the business rules surrounding its operation. Useful documentation for the application is occasionally available. Examining the source code has to be done at some point, but the more information gathered beforehand, the faster and easier will be the job of deciphering the source.

Network Protocols

A network protocol is a set of agreements and specifications for sending data over a network. Many network protocols are in use today. Let's dive into a quick overview of protocols used to help you become familiar with this domain to allow you to make informed infrastructure decisions.

TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) is the network protocol that has the widest use in industry:

- TCP/IP protocol stacks exist for all operating systems currently in use.
- It is an extremely robust and reliable protocol.
- It is routable, which means that it can be sent between disparate networks.
- It is available for free with all operating systems. Even Netware, the home of the once very popular network protocol SPX, offers TCP/IP in addition to its proprietary protocol.
- An extremely robust suite of security functionality has been developed for communication via TCP/IP.
- Internet communication by and large uses TCP/IP. Several other network protocols can be and are used over the Internet. They will be discussed later in this section. However, the Internet communication protocols HTTP, HTTPS, SMTP, and FTP all use IP.
- Web services use TCP/IP.
- Messaging protocols use TCP/IP.
- All remote object communication, such as DCOM, Java RMI, and CORBA, can use TCP/IP.

Subnets

Subnets are segments of a network that can communicate directly with each other. A person on a subnet can communicate directly with all the other computers on that subnet. Networks are split into subnets by the subnet mask part of the TCP/IP properties. Subnets simplify network administration and can be used to provide security.

Computers on different subnets require routers to communicate with computers on other subnets.

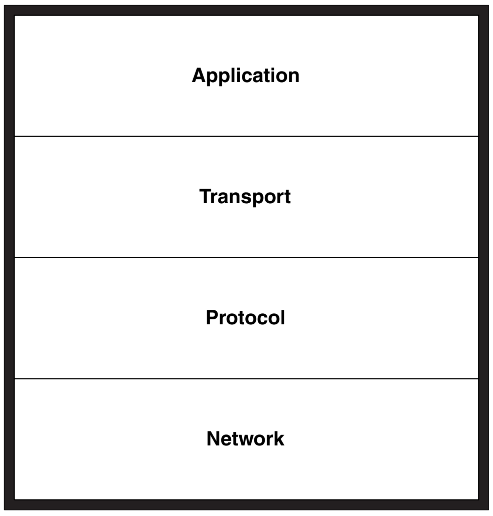
TCP/IP is a suite of protocols built on a four-layer model:

- The first layer is the network interface. These are the LAN technologies, such as Ethernet, Token Ring, and FDDI, or the WAN technologies, such as Frame Relay, Serial Lines, or ATM. This layer puts frames onto and gets them off the wire.
- The second layer is the IP protocol. This layer is tasked with encapsulating data into Internet datagrams. It also runs all the algorithms for routing packets between switches. Sub-protocols to IP are functions for Internet address mapping, communication between hosts, and multicasting support.
- Above the IP layer is the transport layer. This layer provides the actual communication. Two transport protocols are in the TCP/IP suite: TCP and User Datagram Protocol (UDP). These will be explained in greater detail in the next section.
- The topmost layer is the application layer. This is where any application that interacts with a network accesses the TCP/IP stack. Protocols such as HTTP and FTP reside in the application layer. Under the covers, all applications that use TCP/IP use the sockets protocol. Sockets can be thought of as the endpoints of the data pipe that connects applications. Sockets have been built into all flavors of UNIX and have been a part of Windows operating systems since Windows 3.1, see Figure 1–2.

The transmission protocol is the actual mechanism of data delivery between applications. TCP is the most widely used of the transmission protocols in the TCP/IP suite. It can be described as follows:

- TCP is a packet-oriented protocol. That means that the data are split into packets, a header is attached to the packet, and it is sent

Figure 1–2
TCP stack.



off. This process is repeated until all the information has been put on the wire.

- TCP is a connection-oriented protocol in that a server requires the client to connect to it before it can transmit information.
- TCP attempts to offer guaranteed delivery. When a packet is sent, the sender keeps a copy. The sender then waits for an acknowledgement of receipt by the recipient machine. If that acknowledgement isn't received in a reasonable time, the packet is resent. After a certain number of attempts to transmit a packet, the sender will give up and report an error.
- TCP provides a means to properly sequence the packets once they have all arrived.
- TCP provides a simple checksum feature to give a basic level of validation to the packet header and the data.
- TCP guarantees that packets will be received in the order in which they were sent.

UDP is used much less than TCP/IP, yet it has an important place in IP communication because of the following:

- UDP broadcasts information. Servers do not require a connection to send data over the network via UDP. A UDP server can sit and broadcast information such as the date and time without regard to whether or not anyone is listening.
- UDP does not have the built-in facilities to recover from failure that TCP has. If a problem, such as a network failure, prevents a particular application from receiving the datagram, the sending application will never know that. If reliable delivery is necessary, TCP should be used or the sending application will have to provide mechanisms to overcome the inherently unreliable nature of UDP.
- UDP is faster and requires less overhead than TCP.
- Typically, UDP is used for small data packets and TCP for large data streams.

Other Protocols

ATM is a high-speed network technology like Ethernet. All data streamed over ATM are broken down into 53-byte cells. The constant size of the cells simplifies switching issues and can provide higher transmission capabilities than 10 or even 100 Mbs Ethernet. TCP/IP establishes a connection between sender and recipient machines, but it doesn't establish a circuit, a fixed set of machines through which all the traffic will go for the duration of the session. TCP/IP allows network conditions to modify the actual physical route that packets will take during a session. This makes TCP/IP robust and self-healing in the face of network problems. In the case of video or voice transmission, it is best to have a connection between the communicating parties, and ATM can provide that. TCP/IP can be sent over ATM.

Another protocol that has emerged is Multiprotocol Label Switching (MPLS). MPLS competes with ATM in that it allows the establishment of labeled paths between the sender and the receiver. This ability to establish paths has made MPLS of interest to the creators of virtual private networks. Unlike ATM, it is relatively easy with MPLS to establish paths across multiple layer 2 transports like Ethernet and FDDI. It also outperforms ATM and offers some very useful path control mechanisms. Like ATM, MPLS uses IP to send data over the Internet.

For further information, see the MPLS FAQ at *www.mplsrc.com/mplsfaq.shtml*.

The emergence of Gigabit Ethernet implementations has provided enough raw bandwidth to allow TCP/IP over Ethernet to compete with ATM for video and other bandwidth-intensive applications.

The big question to consider about TCP/IP is whether to utilize IPv4, which was released way back in 1980, or to move toward IPv6. IPv6 has many features of importance to the infrastructure of an enterprise:

- Virtually unlimited address space.
- A different addressing scheme that allows individual addresses for every device in an enterprise everywhere in the world.
- A fixed header length and improved header format that improves the efficiency of routing.
- Flow labeling of packets (Labeling packets allows routers to sort packets into streams, making TCP/IPv6 much more capable than TCP/IPv4 of handling stream-oriented traffic such as VOIP or video streams.)
- Improved security and privacy (Secure communications are compulsory with v6, meaning that all communications exist in a secure tunnel. This can be compelling in some circumstances. The increased security provided by IPv6 will go a long way toward providing a Secure Cyberspace.)

Secure Cyberspace

The Internet was designed with a highly distributed architecture to enable it to withstand such catastrophic events as a nuclear war. The infrastructure, the net part, is secure from attack. However, the structure of the Internet provides no security for a single node, router, or computer connected to the Internet.

Unprotected computers are the key to one of the greatest current threats to the Internet: denial-of-service attacks. The others are worms and email viruses that eat up bandwidth.

The U.S. government has outlined a national strategy to protect the Internet. For details, see the draft paper entitled "National Strategy to Secure Cyberspace" at www.whitehouse.gov/pcipb.

From a hardware and operating system point of view, converting to v6 can be very close to cost-free. This is based on the facts that all the major operating systems have v6 IP stacks built into them and that adding v6 support to devices such as routers only requires a software update.

The exhaustion of the IP namespace is another issue of concern. It is a fact that the current IP address space is facing exhaustion. Innovations such as network address translation (NAT) have postponed the day when the last address is assigned, buying time for an orderly transition from IPv4 to v6.

Network Address Translation

Network address translation is a technology that allows all computers inside a business to use one of the sets of IP addresses that are private and cannot be routed to the Internet. Since these addresses cannot be seen on the Internet, an address conflict is not possible. The machine that is acting as the network address translator is connected to the Internet with a normal, routable IP address and acts as a router to allow computers using the private IP addresses to connect to and use the Internet.

IPv6 contains numerous incremental improvements in connecting devices to the Internet. The enormous expansion of the naming space and the addressing changes that make it possible for every device on the planet to be connected to the Internet are truly revolutionary for most enterprises. For Canaxia, not only can every machine in every one of its plants be connected to a central location via the Internet, but every sub-component of that machine can have its own connection. All sections of all warehouses can be managed via the Internet. With a secure Internet connection, dealers selling Canaxia cars can connect their sales floors, their parts departments, and their service departments to a central Canaxia management facility.

The conversion of an existing enterprise to IPv6 is not going to be cost-free. Older versions of operating systems may not have an IPv6 protocol stack available, perhaps necessitating their replacement. In a large corporation, ascertaining that all existing applications will work seamlessly with v6 will probably be a substantial undertaking. In theory, IPv4 can coexist with v6. But we say that if you have to mix IPv4 with v6, you will have to prove that there are no coexistence problems.

We recommend the following be used as a template when contemplating conversion to IPv6:

- If you are a small or medium-sized firm without any international presence, wait until it is absolutely mandatory. In any case, you will be able to get by with just implementing v6 on the edge of the network, where you interface with the Internet.
- If you are a multinational corporation or a firm that has a large number of devices that need Internet addresses, you should formulate a v6 conversion plan. Start at the edge and always look for

application incompatibilities. Of course, you should have exhausted options such as using nonroutable IP addresses along with NAT.

- If you do decide to convert, start at the edge and try to practice just-in-time conversion techniques.

Because of its size and international reach, Canaxia has formulated a strategy to convert to IPv6. The effort will be spaced over a decade. It will start at the edge, where Canaxia interfaces with the Internet and with its WAN. New sites will be v6 from the very start. Existing sites will be slowly migrated, with this effort not due to begin for another five years. As applications are upgraded, replaced, or purchased, compatibility with IPv6 will be a requirement, as much as is economically feasible. Canaxia's architecture team is confident no crisis will occur anytime in the next 10 years due to lack of available Internet addresses.

Systems Architecture and Business Intelligence

If your enterprise runs on top of a distributed and heterogeneous infrastructure, the health of the infrastructure can be vital to the success of your business. In that case, the architect will need to provide visibility into the status of the enterprise systems. The ability to provide near real-time data on system performance to business customers is critical. Following is an example:

- The Web server was up 99 percent of the time, which means it was down 7.3 hours last month. What happened to sales or customer satisfaction during those hours? If the customers didn't seem to care, should we spend the money to go to 99.9 percent uptime?
- The Web server was up, but what was the average time to deliver a page? What did customer satisfaction look like when the page delivery time was the slowest?
- How about the network? Part of the system is still on 10 MBits/second wiring. How often is it afflicted with packet storms? With what other variables do those storm times correlate?
- Part of the service staff is experimenting with wireless devices to gather data on problem systems. What was their productivity last week?

In terms of quality attributes, the following figures for uptime relate to availability:

- *99 percent uptime per year equates to 3.65 days of downtime.*
- *99.9 percent uptime equates to .365 days or 8.76 hours down per year.*
- *99.99 percent uptime means that the system is down no more than 52 minutes in any given year.*

It should be possible for a manager in marketing, after receiving an angry complaint from an important client, to drill into the data stream from systems underpinning the enterprise and see if there is a correlation between the time it took to display the Web pages the customer needed to work with and the substance of his or her complaints. Perhaps several of the times that the Web server was down coincide with times when this customer was attempting to do business. Without data from all the possible trouble spots, it will be impossible to pin down the real source of the problem.

In most organizations, the tools to monitor the health of networks, databases, and the Web infrastructure are not plugged into the enterprise's overall BI system. They are usually stand-alone applications that either provide snapshots into the health of a particular system or dump everything into a log that is looked at on an irregular basis. Often they are little homegrown scripts or applications thrown together to solve a particular problem, or they are inherited from the past when the systems were much simpler and much less interdependent.

Start with the most important performance indicators for your system. Perhaps they are page hits per second or percent bandwidth utilization. Integrate this data into the BI system first. Perhaps the segment of your enterprise infrastructure architecture aligns with one of the company's business units. Fit the performance metrics to the unit's output and present it to the decision makers for your unit.

The planning stage for a new system is an excellent time to build performance metrics measurement into its architecture. New systems often are conceived using an optimistic set of assumptions regarding such metrics as performance and system uptime. When that new system comes online, you will be grateful for the ability to generate the data to quantify how accurate the initial assumptions were. If problems crop up when the system becomes operational, you will have the data necessary to identify when the problem lies at your fingertips.

Service Level Agreements

Service level agreements (SLAs) are a formalization of the quality attributes of availability and performance that have been externalized in a document. As infrastructure and applications become more vital to businesses, they are demanding that the providers of those assets guarantee, in writing, the levels of performance and stability that the business requires. SLAs are a manifestation of how important certain IT functionality has become to modern business processes.

The crucial part of dealing with an SLA is to think carefully about the metrics required to support it. If you are required to provide a page response time under 3 seconds, you will have to measure page response times, of course. But what happens when response times deteriorate and you can no longer meet your SLA? At that point you had better have gathered the data necessary to figure out why the problem occurred. Has the overall usage of the site increased to the point where the existing Web server architecture is overloaded? What does the memory usage on the server look like? For example, Java garbage collection is a very expensive operation. To reduce garbage collection on a site running Java programs, examine the Java code in the applications. If garbage collection is bogging down your system, the number of temporary objects created should be reduced.

The bottom line is this: When planning the systems architecture, you will have to think beyond the metric in the agreement to measuring all the variables that impact that metric. We suggest that the introduction of SLAs in your organization be looked upon as an opportunity rather than a threat and as an excellent tool to force the organization to update, rationalize, and above all integrate its infrastructure measurements. If a new project does not have an SLA, you might want to write up a private SLA of your own. The discipline will pay off in the future.

Systems Architecture and Storage

The vast majority of businesses data storage costs are increasing exponentially. The costs are both in the physical devices and the personnel costs associated with the individuals needed to manage, upgrade, and enhance the storage devices.

One problem with storage costs is that the real cost to the enterprise is hidden in hundreds or thousands of places. Most businesses upgrade storage a machine and a hard drive at a time. This results in scores or hundreds of small, low profile purchases that never show up as line items on the IT department's budget, see Figure 1–3.

The important point is not that you adopt a particular architecture but that you understand the benefits and trade-offs associated with the different architectures.

The conventional approach is to attach the storage devices, usually hard drives, directly to the machine. This is direct attached storage (DAS) and is an excellent choice for situations in which tight security must be maintained over the data in the storage device. Its downside is that it is expensive to upgrade and maintain. Upgrade costs arise more from the costs related to having personnel move applications and data from older, smaller storage devices to the newer, bigger device than from the costs related to the storage device itself. In addition, this storage mechanism makes it difficult to establish exactly what the storage costs of the organization are and to manage those costs.

Storage area networks (SANs) offer basically unlimited storage that is centrally located and maintained. Using extremely high-speed data transfer methods such as FDDI, it is possible to remove the storage media from direct connection to the computer's backplane without affecting performance. SANs offer economical enterprise-level storage that is easy to manage and to grow, but implementing a SAN is a large operation that requires thorough architectural analysis and substantial up-front costs. However, the ROI for most SAN enterprises is very compelling. SANs should not be considered unless it is possible to establish the very high-speed data connection that is required.

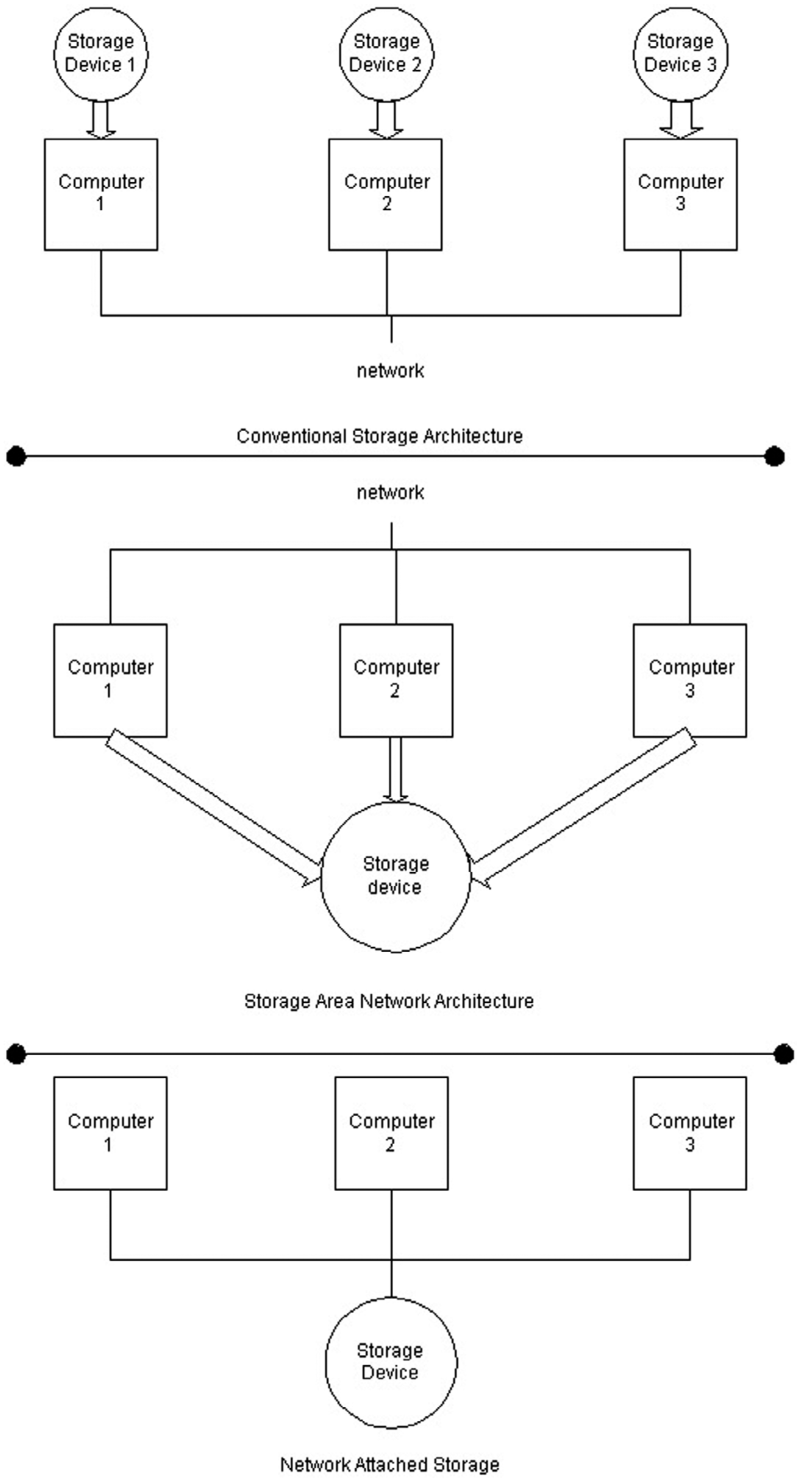


Figure 1–3
The three storage architectures.

If the machines to be connected are geographically remote, different storage architecture is required.

Network attached storage (NAS) devices are low-cost, very-low-maintenance devices that provide storage and do nothing else. You plug a NAS into the network, turn it on, and you have storage. Setup and maintenance costs are very low; the only cost is to relocate the existing, attached storage to the NAS. These devices are perfect for increasing storage to geographically distributed machines.

The major determinant for choosing between NAS and SAN should be the data storage architectural design. The following will help you choose between the two:

- Disaster recovery strongly favors SAN because of the ease with which it can be distributed over large distances. As of this writing, fiber can extend up to 150 kilometers (95 miles) without amplification. This means that data can be mirrored transparently between devices that are 300 kilometers (190 miles) apart.
- Distributed performance is better with SANs.
- The following functionality favors SANs:
  - Very large database applications
  - Application server duties
- File storage definitely favors NASs.
- Ease of administration is a tie. In their own ways, both storage solutions are easy to administer. With a NAS, you plug it into the wall and into the network and you have storage. The work lies in partitioning the storage, moving data onto the device, and pointing users to the device. SANs require quite a bit of up-front design and setup, but once online, adding storage is very simple and more transparent than with a NAS.
- High availability is pretty much a tie. The SAN can mirror data for recovery, and the NASs all have hot-swappable hard drives in a RAID 5 configuration.
- Cost favors NASs. Initial costs for an NAS storage installation are about one-fifth the cost of a similar-sized SAN storage installation. However, the ongoing costs of a SAN are usually less than for a similar NAS. This is due to the following factors:
  - Staffing costs are less due to the centralization of the storage devices. This makes the task of adding storage much quicker and easier.
  - Hardware costs are less. Less is spent on storage devices. Less disk space is needed with a SAN because they are more efficient in the utilization of storage. Normally, SAN storage will average 75 to 85 percent utilization, while utilization for some NASs storage devices will be 10 to 15 percent.
  - LAN infrastructure spending is less. NASs are accessed over the network, and backups are usually conducted over the network. This increase in network traffic can force costly network upgrades.

Canaxia has a SAN that hosts the company's mission-critical databases and data. Portions of that data are mirrored to data centers physically removed from the operation center itself. In addition, each company campus has a central file-storage facility that consists of large numbers of rack-mounted NAS arrays. Most application servers are connected to the SAN. Storage requirements continue to grow at a steady rate, but the cost of administrating this storage has actually dropped as the data have been centralized onto the SAN and various NASs.

Most of you will use a mix of the three architectures. It is important that this mix be planned by your architects and be easy for managers to cost and to justify.

While disaster recovery planning is not about providing backup of a company's data, the storage architecture chosen can make the task of protecting and recovering the enterprise data storage materially cheaper and easier. It will take significant effort to make direct attached storage (DAS) as robust in the face of disaster as distributed storage architecture, such as a SAN or a NAS with remote mirroring. Quick (a few hours) recovery is possible with a SAN architecture.

Disaster recovery is discussed in greater detail later in this chapter.

Systems Architecture Aspects of Security

Ensuring enterprise security is a wide-ranging operation that touches on almost every area of a business. As such, it has to grow out of extensive interactions between the company's management and the architecture team. One of the issues that has to be on the table first is how to build a security architecture that increases a company's competitive advantage. Since security is an enterprise-wide endeavor, the input of the entire architecture team is required. It is important to adopt a graduated approach and apply the proper security levels at the proper spots in the enterprise.

Effective enterprise security consists of the following:

- Effective, well thought out, clearly communicated security policies.
- Effective, consistent implementation of these policies by a company staff that is motivated to protect the business's security.
- A systems architecture that has the proper security considerations built in at every level.

When discussing enterprise security, one of the first matters to discuss is what needs to be protected and what level of protection is appropriate for that particular item. A useful rule of thumb can be borrowed from inventory management. Proper inventory management divides articles into three levels: A, B, and C. The A level items are the most valuable, and extensive security provisions are appropriate for them. Trade secrets, credit card numbers, and update and delete access to financial systems are just a few of the items that would be on an A list. B items need to be protected, but security considerations definitely need to be balanced against operational needs. C items require little or no security protection. As a rule of thumb, 5 percent of the list should be A items, 10 to 15 percent should be B items, and the rest should be C items. A similar analysis is appropriate for the security aspects of an enterprise's systems architecture. The data assets are located on machines that are part of its systems architecture. In addition, elements of the systems architecture, such as the network, will most often be used in attacks upon the business's vital data resources. The following is a list of the major classes of security attacks, in rough order of frequency, that must be considered when building the security part of the systems architecture:

- Viruses and worms
- Attacks by dishonest or malicious employees
- Destruction or compromise of the important data resources due to employee negligence or ignorance
- Attacks from the outside by hackers
- Denial-of-service attacks

Viruses and worms are the most common IT security problems. The vast majority of the viruses and worms that have appeared in the last few years do not actually damage data that are resident on the computer that has the virus. However, in the process of replicating themselves and sending a copy to everyone on a corporate mailing list, viruses and worms consume a large amount of network bandwidth and usually cause a noticeable impact on productivity. For the last couple of years, the most common viruses have been email viruses. They exploit extremely well-known psychological security vulnerabilities in a business's employees. In addition, some person or group of persons in corporate infrastructure support will be required to monitor computer news sites daily for the appearance of new email viruses and to immediately get and apply the proper antivirus updates to, hopefully, prevent the business from becoming infected by this new email virus. Unfortunately, the current state of antivirus technology is such that defenses for viruses can only be created after the virus appears.

The majority of attacks against corporate data and resources are perpetrated by employees. To protect your organization against inside attack, utilize the following:

- User groups and effective group-level permissions
- Effective password policies to protect access to system resources
- Thorough, regular security audits
- Effective security logging

Assigning users to the proper group and designing security policies that completely, totally restricts access to only the data and resources the group requires to perform its functions is the first line of defense against attacks by insiders. Don't forget to remove the accounts of users who have left the firm.

For those enterprises that still rely on passwords, another area for effective security intervention at the systems architecture level is effective password policies. In addition to mandating the use of effective passwords, make sure that no resources, such as databases, are left with default or well-known passwords in place. All passwords and user IDs that are used by applications to access systems resources should be kept in encrypted form, not as plain text in the source code. Passwords are the cheapest method of authentication and can be effective if strong passwords are assigned to users, the passwords are changed on a regular basis, and users don't write down their passwords. Users can reasonably be expected to remember one strong password if they use that password on at least a daily basis. Since a multitude of applications are password-protected in most enterprise environments, we recommend that enterprises standardize on Windows operating system logon user name and password as the standard for authentication and require every other application use the Windows authentication as its authentication. This can be via the application making a call to the Windows security service or by the application accepting a Kerberos ticket from the operating system.

Unfortunately, the vast majority of enterprises have a multitude of applications that require a user name and password. As a result, most users have five, ten, even fifteen passwords. Some passwords will be used several times a day, some will be used a couple of times a year. Often the corporation cannot even standardize on a single user name, so the user will have several of those, too. In a situation such as this, most users will maintain written username and password lists. The justification for this situation is always cost: The claim is made that it would cost too much to replace all the current password-protected programs with ones that could get authentication from the Windows operating system. When faced with this argument, it is useful to document the time and money spent on resetting passwords and the productivity lost by users who are locked out of an application that they need to do their job. We expect you will discover costs in the range of $10 to $20 per employee per year. Those figures, coupled with the knowledge of the damage that could occur if one of the password lists fell into the wrong hands, might get you the resources necessary to institute a single sign-on solution.

Devices such as smart card readers and biometric devices can authenticate using fingerprints and retina scans. These devices allow extremely strong authentication methods, such as changing of the password daily or even changing the user's password right after they have logged on. The cost of equipping an entire enterprise with such devices is considerable, and they only get you into the operating system. If the user has ten password-protected programs to deal with once he or she is logged in, the device has not bought you much. Protecting A-level assets with one of these devices makes excellent sense.

For the majority of employees and the majority of businesses, a single sign-on solution with the user expected to memorize one strong password every couple of months is adequate security and the most cost-effective solution.

You need to know where you are vulnerable and what attacks can exploit these vulnerabilities. If you have not run a security audit, make plans to do so. Large enterprises may want to engage outside consultants to audit the entire firm. As an alternative, some software packages can effectively probe your system for vulnerabilities. You might consider building up a security cadre of individuals in your organization. They would be tasked with knowing and understanding all the current hacker attacks that can be mounted against systems of your type and with running any security auditing software that you might purchase.

Finally, effective, tamperproof audit systems will allow you to detect that an attack has occurred and will provide you with the identity of the attacker. This is where scrupulously removing expired user accounts is important. If Bob Smith has left the company but his user account still exists, that account can be compromised and used in an attack with you having no clue as to who really perpetrated it.

In any case, the cost of implementing the patch to the vulnerability has to be balanced against the seriousness of the threat and the probability that someone in your organization would have the sophistication necessary to carry out such an assault. Lock the most important doors first.

Inadvertent damage to or compromise of business data by well-meaning or ignorant employees causes substantial business losses annually. While this type of damage has no malicious component, the results are the same as for a malicious attack. Inadvertent damage can be prevented in a variety of ways. Training is first and foremost. When new corporate applications are being brought online, it is crucial that people who will be using them are thoroughly trained. Effective training provides a positive ROI. After an application is in regular operation, it is important that experienced operators be given the time and resources to train and mentor new operators. However, even the best training programs are not 100 percent effective. It is also important to make sure that people are in the proper roles and that the security parameters of these roles are so crafted that people are given the least opportunity to do damage without restricting their ability to carry out their assigned functions. Audit trails are useful for establishing exactly what was damaged or compromised. After that has been ascertained, it is the role of the backups that have been applied to this data that will allow you to recover from the situation.

Hacker attacks are high-profile events that invariably become news items when discovered. It is difficult to accurately judge the size of business losses caused by hacker attacks from the Internet. In any case, you have to take them seriously. All companies can expect to experience hundreds of low-level "doorknob rattling" attacks, such as port scans run against them, in the course of a year. On the positive side, the vast majority of technical hacker exploits are wellknown, and the measures necessary to defeat them are standard parts of all companies' Internet defense systems. The major response that hacker attacks will prompt is you keeping all your machines both inside the firewall and in the DMZ 100 percent up to date on security patches. Any stratagem that you can devise to streamline the application of security patches will give your company a strategic advantage over a company that does it all by hand or ignores the problem completely.

Of the attacks that can occur from the outside—from the Internet—denial-of-service (DOS) attacks are the least common but potentially the most destructive. DOS attacks involve the recruitment of large numbers of outside computer systems and the synchronization of them to flood your system with such a large number of requests that it either crashes or becomes unable to respond to legitimate service requests by your customers. Fortunately, due to the large-scale effort involved on the part of the attackers, DOS attacks are rare and are normally directed against high-profile targets. DOS attacks are extremely difficult to combat, and it is beyond the scope of this book to discuss methods to deal with them. It is important, though, that if your company is large enough and important enough to be a possible target for a DOS attack, you begin now to research and to put into place countermeasures to fend off a DOS attack.

All security mechanisms, such as encryption and applying security policies, will have an impact on system performance. Providing certificate services costs money. By applying agile architecture principles to the design of this security part of your systems architecture, you can substantially mitigate these impacts. To be agile in the area of systems architecture security design means applying just the right level of security at just the right time. It means being realistic about your company's security needs and limiting the intrusion of security provisions into the applications and machines that make up your systems architecture. Just enough security should be applied to give the maximum ROI and no more.

As a final note, do not forget physical security for any machine that hosts sensitive business data. There are programs that allow anyone with access to a computer and its boot device to grant themselves administrator rights. Once more, we recommend the A-, B-, and C-level paradigm. C-level data and resources have no special physical security. B-level data and resources have a minimal level of security applied. For A-level data and resources, we suggest you set up security containment that also audits exactly who is in the room with the resource at any time.

Systems Architecture and Disaster Recovery

Disaster recovery planning is sometimes referred to as "business continuity planning." DRP for the enterprise system infrastructure is a major systems architect responsibility.

The purpose of DRP is to produce a set of plans to deal with a severe disruption of a company's operations. DRP is independent of the modality of the disaster (flood, fire, earthquake, terrorist attack, and so on). It involves the following:

- Identification of the functions that are essential for the continuity of the enterprise's business.
- Identification of resources that are key to the operation of those functions, such as:
    - Manufacturing facilities
    - Data
    - Voice and data communications

- People
- Suppliers
- Etc.

- Prioritization of those key resources
- Enumeration of the assumptions that have been made during the DRP process
- Creation of procedures to deal with loss of key resources
- Testing of those procedures
- Maintenance of the plan

DRP is normally a large project that involves input from every unit in the company. The sheer size of the effort can make it seem prohibitively expensive. DRP is largely about money. Perfect disaster protection will cost an enormous amount of it. A disaster coupled with inadequate DRP can destroy a company. However, the DRP process will be like any other company process in that the size and cost of the DRP effort will be adjusted to fit available company resources. To provide the best DRP for the available resources, it is vital that the analysis phase correctly do the following:

- Identifies all the key resources
- Accurately prioritizes the key resources
- That it provide accurate costing for protection of those resources
- That it provide accurate estimates of the probability of the destruction of the key resources

Given an accurate resource analysis, the IT/business team can work together to build an affordable DRP. When costing the DRP program, do not neglect the maintenance of the plan.

DRP planning for most enterprises will involve the coordination of individual DRP plans from a variety of departments. We focus on Canaxia's DRP and the process of its creation.

Initially, the disaster planning team (DPT) was created and tasked with creating a DRP process. Canaxia's architecture group was a core part of that team. It quickly became obvious that Canaxia required a permanent group to maintain and administer its DRP. As a result, the Department of Business Continuity (DBC) was created.

Following are some of the critical Canaxia resources that the DRP identified:

- Engine manufacturing facilities. All engines for the Canaxia line of vehicles were manufactured in one facility.
- Brake assemblies. Canaxia used a unique brake system available from a single supplier. That supplier was located in a politically unstable region of the world.
- Data. Of the mountain of data that Canaxia maintained, 10 percent was absolutely critical to daily functioning. Furthermore, 45 percent of the data was deemed irreplaceable.

- Voice and data communications. Canaxia depended on the Internet for the communication necessary to knit together its far-flung departments.
- The enterprise resource planning (ERP) system that totally automated all of Canaxia's manufacturing and supply activities.
- A core group of IT employees that had done the installation of the ERP system. Loss of the manager and two of the developers would have a severe negative impact on the functioning of the ERP system.

Once critical resources had been identified, they were divided into three categories:

- Survival critical. Survival critical resources are those absolutely required for the survival of the company. Normally, survival critical activities must be restored within 24 hours. Data, applications, infrastructure, and physical facilities required for the acceptance of orders and the movement of completed vehicles from manufacturing plants are examples of survival critical resources for Canaxia.
- Mission critical. Mission critical resources are resources that are absolutely required for the continued functioning of the enterprise. However, the company can live without these resources for days or weeks.
- Other. This category contained all the resources that could be destroyed in a disaster but were not deemed to be survival or mission critical. If they were replaced, it would be months or years after the disaster.

Within each category, the team attached a probability that the resource would become unavailable and a cost to protect it. All the assumptions used in the analysis phase were detailed to allow the individuals tasked with funding the DRP to do a reality check on the numbers given them.

Documenting existing procedures so that others could take on the functioning of key Canaxia employees was a substantial part of the DRP. Canaxia, like many other enterprises, had neglected documentation for a long time. That neglect would have to be rectified as part of the DRP. In many cases, producing documentation decades after a system was built would be an expensive task.

Every DRP will have to enumerate a set of tests that will allow the company to have confidence in the DRP. Running some of the tests will be expensive. Once more, the test scenarios have to be rated as to cost, probability of resource loss, and criticality of the resource. Testing is a good place to examine the assumptions from the analysis phase for accuracy.

A DRP will have to be maintained. As systems and business needs change, the DRP will have to be revised. In some cases, the DRP needs can drive business decisions. Canaxia made the decision to move to a generic brake assembly that could be obtained from a variety of sources. Canaxia's brake performance was one of its technical selling points, and management knew that moving to a generic brake would not help sales. Canaxia also established a company rule that strongly discouraged getting into single-source supplier relationships in the future.

We recommend an iterative approach to DRP. First do the top of the survival critical list all the way through the testing phase. Use the experience gained in that process to make the next increment more focused and efficient. Trying to do the entire job in one big effort risks cost and time overruns that can cause the process to be abandoned before anything is protected.

Conclusion

There are three important aspects to emphasize about systems architecture:

1. It must align with the business goals of the organization.
2. It must provide what the stakeholders need to perform their functions. This is a two-way street. The architecture team should take responsibility to establish communication with systems architecture stakeholders and to understand their issues.
3. The software and hardware infrastructure of an enterprise is a major asset that must be managed to provide the greatest return on that investment.

The following should be considered as systems architecture best practices:

1. Know the business processes that the systems architecture is supporting. Know them inside and out.
2. Keep support of those business processes first and foremost on your agenda. Know what the business needs and keep the business side aware of your accomplishments.
3. Know the components in your systems architecture: all the machines, applications, network connections, and so on.
4. Instrument, your system. That is, install monitoring and measurement systems that allow you to find the problem areas and the bottlenecks.
5. Attack the cheap and easy problems first. That will build and help maintain credibility and trust with the business side of your company.
6. Prioritize and be proactive. If you are not constrained to produce immediate results, identify the most important systems architecture problems and attack them first, even before they become problems. Good system measurements are key to being able to identify problem areas in your systems architecture and to establish the most effective means to deal with them.
7. Know all your stakeholders. The people aspects of your systems architecture are at least as important as the machines.
8. Only buy as much security as you need. Give up on the idea of becoming invulnerable. Prioritize your security issues into A, B, and C lists.

As architects team with the business side to align systems with business processes and keep them aligned as business needs evolve, they will get the opportunity to develop skills in understanding the business and working with financial algorithms and calculations.