CHAPTER 6

Baseline Description

6.1 Introduction

As we have already seen in Chapter 3, the internal processes of most strategic methods (such as the business strategy, the ISSP, and so on) involve a baseline phase. This is a key activity in providing the foundation for further strategic work. Without a good knowledge of the current state of affairs, it is impossible to determine a reasoned direction. You may remember that later phases in most strategic methods include a migration assessment—the TOGAF architectural development method (ADM) has the migration planning phase. This migration phase requires that the organization understand the current (or baseline) state, the end state, and the gaps between the two that represent the migration.

This chapter takes a look at the TOGAF ADM baseline description phase. As shown in Figure 6.1, this is the second major ADM phase. In this chapter, we describe the properties of this phase and conclude with the baseline description for our example organization to demonstrate the concepts.

The baseline description phase is the second major phase in the TOGAF ADM. As stated in TOGAF, its objective is to:

> Build a high level description of the characteristics of the current environment. This is necessary as the description documents the starting point for architectural development, and lists the interoperability issues that the final architecture will have to take into account.

It is key for the architect to develop a solid understanding of the current technology environment. This establishes the necessary basis for the next phase (the target architecture). The TOGAF inputs and outputs for this phase are shown in Figure 6.2. The inputs have been generated

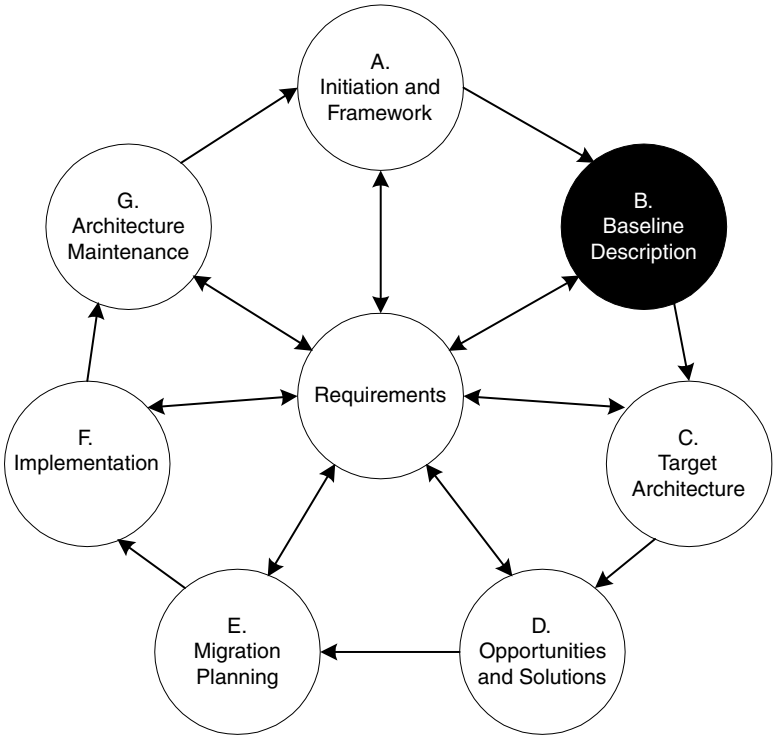

FIGURE 6.1. ADM, with baseline description highlighted.

from the initiation and framework phase, while the outputs are required for the target architecture phase.

The initiation and framework phase will have delivered a full context for the architectural project. This will include both organizational and project contexts. This phase will gather information from various sources so that the current environment (systems, technologies, and processes) can be described. Sources may include

- IT operational documentation
- Individual business systems design documents
- Environmental (e.g., network, security) documentation
- Interviews with development and operational staff
- Interviews with key users and management

The output of this phase is version 2 of the business architecture that

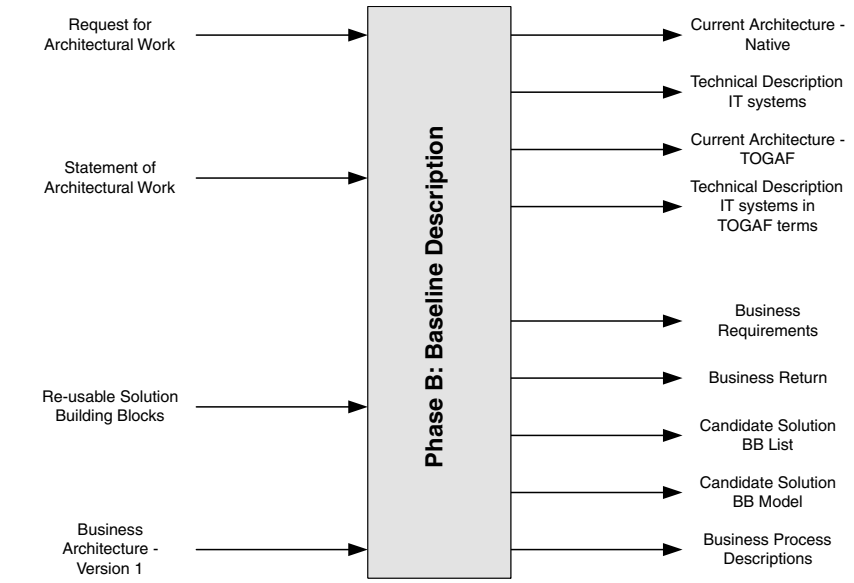

FIGURE 6.2. Baseline description wiring diagram.

extends on the initiation documentation collected during the previous phase.

We do not intend to use fully the TOGAF concept of building blocks (see the next section), yet TOGAF calls for initial building-block specification during this phase. In their place we intend to introduce a concept we call "super services." As we descend further into this chapter, the notion of services will be introduced more fully, and Chapter 9 is dedicated to a full description of super services. In short, a super service is a clustering of technical (typically not business) functionality that collects and extends the base TOGAF foundation platform services. During this phase the architect may note that higher-layer technology services are being developed within business systems. Additionally, strategic business requirements may dictate technology services that blur the line between platform and business system. These technical functions should be captured, and their placement considered. Figure 6.3 shows a slightly altered view of the standard TOGAF inputs and outputs. Here, we show the information-gathering inputs to this phase. Also, the building blocks have been removed and are instead replaced by an initial view of the service portfolio.

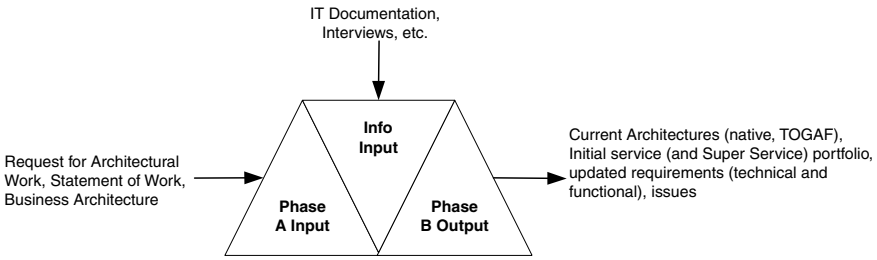

FIGURE 6.3. Phase B – adjusted inputs and outputs.

6.2 Where Is TOGAF Appropriate?

It is important to understand how TOGAF is holistically positioned against the various different types of architecture. Due to its framework structure, TOGAF can be used as an architectural development tool at a number of architectural levels, as follows (Figure 6.4 summarizes the characteristics of using TOGAF at differing levels):

- TOGAF can be used at an enterprise architectural level. Using TOGAF, we can define the organization's enterprise technical architecture. This represents the high-level use of TOGAF and is the subject of this text.
- TOGAF can be used to describe architectures that represent the specification and integration of various systems. This is a medium-level treatment of TOGAF.
- TOGAF can be used to architect individual business systems. This is the lowest-level, most specific use of TOGAF.

TOGAF competes with an increasing number of other approaches as the described architecture becomes increasingly solution-specific. Due to its structure and defined artifacts, we believe that TOGAF is most effective at the enterprise level and that this was its original intent.

There are a wide variety of methodology choices when the desire is the architecting of individual systems (or even components). Methods such as Rational Unified Process™ (RUP) combined with Unified Modeling Language (UML) provide detailed approaches, notions, artifacts, and tools for the description of application architectures. In this area, TOGAF does not provide a specific or detailed enough framework to currently compete.

The relationship between TOGAF and application architectures is important in the success of enterprise architectures. We believe that TOGAF-produced enterprise architectures provide the necessary tech-

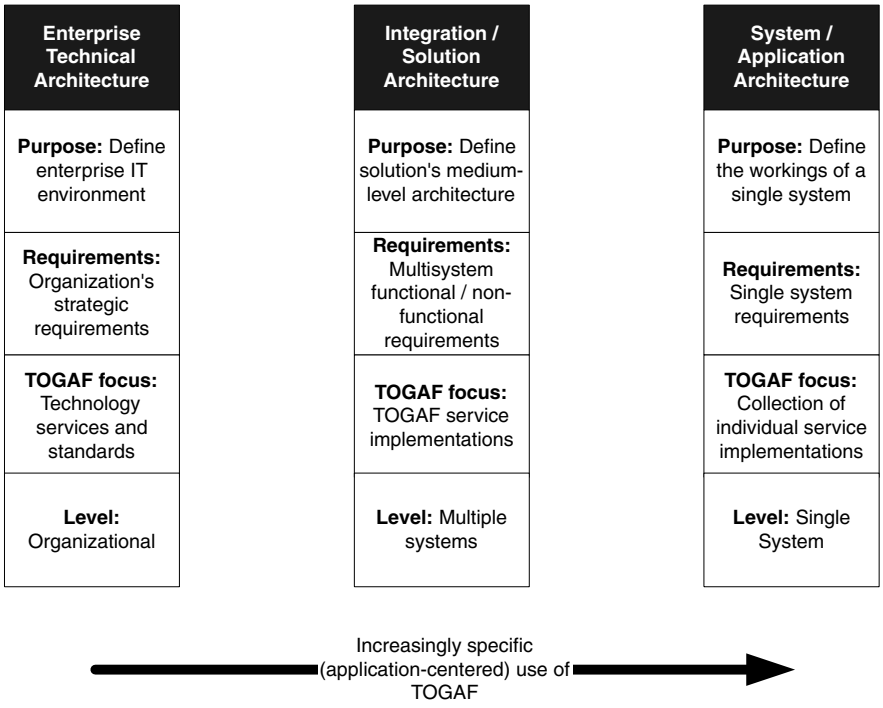

FIGURE 6.4. Characteristics of TOGAF use.

nology governance and base constructs (such as the standards information base) with which to architect individual business system projects. Furthermore, applications may use TOGAF service notations to describe system-wide features such as interoperability, scalability, and manageability (known as the –abilities). Ensuring that individual systems accurately follow the organization's technical architecture is essential. However, we do not believe TOGAF will compete with other methods in the architecting of individual systems.

With the advent of e-business, the pressure is now for organizations to consider not only the Internet-facing systems themselves but also the entire back-end integration required to extend business processes to external parties (e.g., customers, partners, suppliers). This increases the requirement to describe and architect the integration environment fully before embarking on individual e-business systems projects. Describing the (medium-level) components of this integrated environment suits a TOGAF approach. Using the concept of services, it can be valid for the architect to map out the integration environment in full TOGAF terms. Such a use is analogous to a "mini enterprise-wide" approach. Using

TOGAF artifacts at this level allows the effective description of the essential interface implementations, the technologies adopted for these interfaces, and the platform-quality aspects to be achieved by the final deliverables. TOGAF provides a complete approach for capturing the overall business requirements for the integrated environment and ensuring that they are met. Used in this way, TOGAF essentially describes the solution architecture. We have used a TOGAF approach effectively in a number of organizations at this level.

The third potential use of TOGAF is for the definition of the corporate technical architecture; that is, the technology framework within which the organization will deliver its technology. In this book we concentrate specifically on this use of TOGAF. The technical architecture is a component of the strategic-planning cycle, as discussed in Chapter 3, and defines the organization's strategic technology direction, the platform on which business functions sit. At this level, architectural development is more general and increasingly high-level. Development is impacted less by the requirements of individual systems and more by the overarching requirements of IT and the business. Indeed, architectural development is less about defining the architecture for an individual system but is more focused on the architecture to be applied in a general corporate sense.

One of the contemporary components of TOGAF is the concept of building blocks. Defined in TOGAF, building blocks are characterized as follows:

- A building block is a package of functionality defined to meet the business needs across an organization.
- A building block has published interfaces to access the functionality.
- A building block may interoperate with other, interdependent building blocks.
- A good Building block has the following characteristics:
  - It considers implementation and usage and evolves to exploit technology and standards.
  - It may be assembled from other building blocks.
  - It may be a subassembly of other building blocks.
  - Ideally, a building block is reusable, replaceable, and well-specified.
- A building block may have multiple implementations but with different interdependent building blocks.

A building block is therefore simply a package of functionality defined to meet business needs. The way in which functionality, products and custom developments are assembled into building blocks will vary widely across organizations and business applications. Every organization must decide for itself what arrangement of building blocks works

best for it. Systems (in the wider sense) are built up from collections of building blocks, so building blocks must interoperate. Wherever that is true, it is important that the interfaces to a building block be published and reasonably stable.

TOGAF suggests that building blocks can be defined at various levels of detail, depending on the current stage of the architecture development method. For instance, at an early stage, a building block can simply consist of a collection of functionality such as a customer database and some retrieval tools. Building blocks at this functional level of definition are described in TOGAF as architecture building blocks (ABBs). Later, real products or specific custom developments replace these simple definitions of functionality, and the building blocks are then described as solution building blocks (SBBs).

The Open Group is continuing to develop the concept of building blocks in TOGAF. In its current state, we have found that using TOGAF building blocks for enterprise technical architecture development can be somewhat confusing in intent. In their current state, it is difficult to rationalize how building blocks can be put to use in defining an organization-wide architecture, where we are not concerned as much with the functional make-up of business systems, rather the technology makeup of the platform they will operate on. At this level of architectural work, the finest grain of functionality (both business and infrastructural) considered tends to be the business system (note that in some cases more detailed analysis may be required to understand core subsystems for service reuse) and services. Typically, business systems are defined in the business system architecture and augmented with domain-specific application models. Services are already a major facet of TOGAF. It therefore does not appear necessary, or correct, to describe these aspects again in the technical architecture with building blocks. Viewing business systems with the aim of identifying reusable functionality that can be migrated to "the platform" (i.e., into a technical service) is a key objective of the technical architecture; however, building blocks may not necessarily be the best method of achieving this aim.

TOGAF also suggests ". . . applying the architectural building block method introduces application space into the architectural process. This is the means of linking the services view, which addresses functionality that must be considered on an enterprise basis, with the applications view, which may or may not address global functionality." Certainly, this approach has some merit, especially at a more detailed architectural level. However, at the enterprise level we are less concerned with application functionality and more concerned with the technology environment in which the applications operate (i.e., the platform and its services).

At the enterprise technical level, we are particularly interested in the

technology services either provided by or used by the business systems. This form of technical functionality is readily encapsulated in the TOGAF concept of *services*—there is no need to describe them as building blocks. In fact, it is possible to consider a service as having the same characteristics ascribed to building blocks. Services have a greater level of synergy with the development of the technical architecture than do building blocks. Services describe the *platform* or infrastructure components of the corporate technical environment; for instance, network, integration, or transaction management typifies platform services. One of the primary focuses of the enterprise technical architecture is to define and describe all the key technical services that make up the IT environment. Services (and the actual technologies or standards that implement them) provide the platform on which business systems are implemented.

As described earlier, the concept of building blocks is a reasonably new introduction to TOGAF. It is our opinion that as they evolve their use will become clearer. The reader is encouraged to follow their development. However, we have chosen to make only limited reference to building blocks within the text. This does not corrupt the way we are using TOGAF to define the organization's technical architecture, it merely demonstrates its flexibility. TOGAF is a framework, and as such it can be molded in many ways to achieve the goal of producing a technical architecture; it need not be applied prescriptively. Architects are encouraged to extend or subtract from the framework to ensure that it fits the requirements of the organization and the architectural program. In this way, TOGAF is equally suitable for a medium-sized organization wishing to carry out a two-week "architectural" exercise or a large corporate or government department requiring a detailed technical architecture.

Our relegation of building blocks does have an impact on our description of the phases of the TOGAF ADM. The development of architectural and solutions building blocks forms parts of the ADM phase outputs. We will continue to faithfully present the TOGAF inputs and outputs for each phase (and in some cases this may include building blocks).

6.3 Describing the Current Systems

The main objective of this phase is to form an understanding of the organization's current IT environment. We do this for a number of reasons:

- An architectural project is rarely based on a "green fields" situation. There are always "legacy" systems and technologies. The target ar-

chitecture (defined in the next phase) will have to augment, replace, or maintain aspects of the current environment. The validity of the target architecture will depend on just how the current environment is treated.

- To understand the issues with the current environment. The target architecture is a product of the current environment, identified issues, and the gaps between the target and the current environment. The current environment may never have been put under an "architectural microscope." Stepping back and viewing the individual systems holistically against the rest of the IT environment—not to mention the business strategies—can throw new light on system issues not apparent at first.
- To catalog the technologies, standards, and products that make up the individual systems. This allows the architect to understand areas of technology overlap, where multiple technologies provide the same service, and technology gap.
- To understand how business systems are integrated.

There are a number of ways to describe the current systems. The first, and most intuitive (to the organization, that is), is in a native form. This means that current system descriptions are based on the notations that already exist in the organization. Most organizations will have voluminous documentation describing systems. This material can be easily summarized to provide the necessary baseline descriptions.

A second method is to view the current system structures in TOGAF terms. This means translating the documentation already held on the current systems using techniques that are native to TOGAF, such as services and reference models. This approach has some value as well because it means the artifacts produced can transition smoothly into the next ADM phases, which use a pure TOGAF style.

Cataloging Current Systems

Business systems (and infrastructure) consist of a wide variety of complex interworking components that make up the whole. In analyzing the architecture of a system, it can be difficult to appreciate its full extent. However, understanding its breadth and depth are key to the architectural process. That is not to say that the system needs analysis of the detail of every subsystem or interface but rather that the key components—or to put it another way, the significant architectural components—are well-understood.

It can sometimes be difficult to operate at a level above system detail. For instance, application architects will wish to develop an in-depth understanding of an application's internal stuctures. The job of the tech-

nical architect is to understand some aspects of this detail but not to spend endless hours descending into system design minutiae. The aim is to consider aspects of the system that have bearing on the enterprise view of technology. In essence, extract the TOGAF artifact components from systems, which includes:

- Technologies, such as specifications
- Standards adopted either by the system independently or through a more holistic organizational mandate
- Interfaces (such as APIs and exchange formats) that may be both industry-provided or proprietary
- Products, particularly infrastructure or service-related ones
- Quality aspects, such as scalability, availability, and security
- Policy and procedures

As an example, an application architect may be interested in the functionality of a Web portal component that transforms XML via XSL into a device-specific delivery channel (such as HTML or WAP). The Technical Architect is interested in XML, XSL, HTML, and WAP as key standards (technologies and interfaces) used within the application, and will wish to understand what products have been used to implement them and the conceptual frameworks under which they will be implemented. This is a recurring mantra for describing the technical architecture—standards, frameworks, and governance. It is therefore critical to be able to separate the significant components that are relevant to the technical architecture without delving into unnecessary detail.

Obtaining the Information

The cataloging of current systems can be a time-consuming and laborious process. It is therefore important to ensure that the scope of the assessment is fully understood and that only pertinent information is gathered.

Obtaining the necessary information requires that the correct people be approached. Normally it is a good idea to understand the IT group's hierarchy, including key positions and personnel. Furthermore, IT management (such as the CIO or departmental managers) will be able to list the "people in the know." Establishing an informal communications web with subject matter experts will speed the process. Note the members of all IT steering groups, including the IT strategy group (the strategy group may, of course, be the steering group for the technical architecture work). These key people are not only necessary for information gathering purposes but are important quality-assurance resources, architectural advocates, and change agents.

There are a number of strategies that can be used to obtain the necessary information, including:

- Interviews
- Library or documentation research
- Surveys

We find that the most productive method of gathering information on the current systems is through direct interviews. Although identifying the subject matter experts can be difficult, generally the detailed knowledge they can bring to the assessment will be invaluable (documentation can seldom provide the level of information gained through these interviews). Consider approaching the application architects, the network specialists, infrastructure support personnel, and help desk personnel. Also, implementation project managers can have reasonably detailed technical knowledge of the entire solution.

Regardless of how the information will be finally described (e.g., native or TOGAF), it is important to prepare a template of sorts, listing the areas of prime interest, to keep information gathering from being sidetracked.

As an interview technique, it can be worthwhile to predetermine key questions that delve into as many facets of the system as possible. For instance, use:

- *Fact finding* questions such as "How does that work?" or "What does that component do?"
- *Leading* questions such as "Did you consider any alternate technologies here?" or "What quality aspects do you consider important?"
- *Exploring* questions such as "Why was that used?" or "Was integration a problem?"
- *Perception* questions such as "Did you all agree with that architecture?" or "What do you think about the way the system is managed?"

Always close by summarizing the important facts extracted from the interview. Use language such as "My understanding is that . . ." or "I think that . . . is this correct?"

Most IT groups have a large amount of documentation describing their current systems. Although the quantity may be high, extracting useful information of a strategic architecture nature may be somewhat of a challenge. Always search out the original design documents. If possible discover the methodology applied to the implementation. This can tell the architect the types of design documentation that may be available. For example, an object-oriented project will inevitably provide some sort of software architecture document. When viewing documentation, it is important not to be focused entirely on one subject—say the application architecture or the network design—even if this is the most

readily accessible. Always attempt to gather a cross section of information affecting all of the services provided by the application. Operational documentation, such as service-level agreements, can also provide a valuable insight into the quality aspects of the application platform such as performance, capacity, and availability.

Surveys are another approach to gathering information about the current systems. Compared with face-to-face contact, this can sometimes be a suboptimal approach. However, it may be unavoidable where the people with the information are more inaccessible (e.g., the international offices of a large corporation or from remote vendor organizations). Additionally, surveying can be less time-consuming for the architect, and can lend an air of formality. The structure of the survey is critical because there is seldom an opportunity to clarify answers. Structure the survey to match the selection method of describing the systems. Using a TOGAF approach can be more beneficial in this respect because it defines formal technology taxonomy. Always test the survey on a readily accessible system specialist before submission.

Native Views

One of the most effective ways of describing the current system is in native terms. This means that the architect will use the vocabulary and taxonomy already used by the IT group to describe the current environment. By their very nature, native views are embodied in the current system documentation and culture of the group. Using a native vocabulary, the architect can easily extract relevant information without the need to transform it into a neutral or common industry format (such as TOGAF). The advantage of native views is that they should be instantly recognizable and understandable by the organization.

The business (and infrastructure) systems are the very reason the IT environment exists. Large organizations will have countless systems, and it is important to understand the scope of the assessment before starting. This would be defined in Phase A: Initiation and Framework. The business systems architecture, if available, should also provide a list of all of the major systems and may have also prioritized them in order of importance. Begin with the core systems. Understand as much as possible about the structure of these systems. Consider the technologies used both to build and deploy the systems. Describe the physical topology of the systems, including the placement of clients, servers, and other important physical components. Attempt to collect any capacity, performance, and scalability information; the support teams will be a principal source of information here.

It is also imperative to view how business systems interoperate. Most major corporate systems will exchange information in many ways. Even the lowly batch-generated flat file (or even manual data re-entry) is an

interchange, and its details should be captured. Consider all methods of integration, such as component-to-component interfaces, asynchronous message passing, or flat files on removable storage. Understand the communications protocols used to support information exchange and the format of information exchanged (i.e., how the syntax of the information is described); some may use comma-separated values whereas others are described in XML or EDIFACT. Understand, also, libraries that provide integration support and any applicable ADIs.

The infrastructure should not be forgotten. This consists of the technology that supports the business systems while being reasonably isolated from them. Areas to consider include:

- The network. Include both the wide area and local area networks in the assessment. Consider any external or partner networks that are significant (e.g., the Internet or any extranets). Understand structure or topology, the protocols supported, and current and future capacity assessments. Also catalog significant network equipment, such as routers and switches.
- The systems and network management environment. This includes all of the components used to manage the business systems and the infrastructure. Organizations without a discernable systems management implementation are likely to still have components in the field that need cataloging. Areas to look for include management protocols and information formats, products, application instrumentation techniques, and the overall capability (including the physical reach of management).
- E-mail, directory, file sharing, and other infrastructure applications. Most organizations have a significant investment in these infrastructure applications. They are important to the integrity of the entire environment and support vital user-productivity functions. Consider products, protocols, formats, and physical topologies.
- Security environment. Typically, security is implemented within individual business systems. However, an organization may also have an enterprise-wide solution supporting a number of systems. Consider the protocols, technologies, cryptography algorithms, and products used to implement this service. Also define how the individual systems integrate with the security environment.

One other aspect that may be interesting is the opinion of the IT professionals and other business users. During interviews, this sort of opinion will be voiced. Whether it is perception or whether there is some quantifiable justification, anecdotal comments may prove invaluable later when assessing the current environment. Always attempt to substantiate key evidence, especially if it will be used later to provide input into target architecture decisions.

6.4 TOGAF Terms

Using TOGAF terms to describe the current systems can be more time-consuming if the organization (and architect) is not familiar with its structure and vocabulary. We provide a brief, generic discussion on what constitutes "TOGAF terms" in this section. Without TOGAF experience, it is unlikely that any systems within the organization will be fully described in TOGAF (or proto-TOGAF) terms, so the architect would need to translate native terminology into the TOGAF taxonomy.

As the organization becomes more familiar with TOGAF terminology—after all, the final technical architecture will be described in this way—it will be increasingly likely that new business systems projects will begin using TOGAF vocabulary for describing structure at the high level. The same goes for the enterprise architect. The greater use made of TOGAF, the more likely it will be that system descriptions will be easily seen in TOGAF terms. In fact, the TOGAF way is not in any way exceptional, or wildly academic, but rather it uses a nomenclature that should be immediately familiar to most in IT.

Sooner or later the current system assessment will be translated into TOGAF terms. This translation could occur during this phase of the ADM because it will be required as an initial input into the target architecture phase, or a translation can be applied during target architecture development. However, it is best that it occurs at this point, and therefore we would encourage architects to migrate their thinking into TOGAF mode during the description of the current systems.

Chapter 8 takes a detailed look at the processes involved in converting the organization's current system view in TOGAF terms, a key step in moving into the target architecture development stages. In this section, we provide some of the base constructs of the TOGAF way.

A Service Portfolio

What TOGAF components can be used to describe the current systems? As has already been described, TOGAF is an architectural framework, and as such it can be applied in any number of ways to the description of the enterprise architecture. TOGAF is constructed from a number of important parts, some of which have already been presented and others that will come to light in more detail in the chapters to come. Primarily, the main components of TOGAF are:

- The architectural continuum
- The architectural development method (ADM)
- The foundation architecture
- A standards information base (SIB)

- A taxonomy of views
- A governance framework

Applicable TOGAF components at this stage include the foundation architecture (specifically, the generic services), the concept of the SIB (rather than its TOGAF implementation), and architectural views. We provide an initial description of these in the remainder of this section.

The Technical Reference Model

The technical reference model (TRM) is a component of the TOGAF foundation architecture (this exists at the left-most end of the architectural continuum; see Chapter 4). The foundation architecture is a collection of generic technology services, defined by The Open Group, that provides the basis for refining the organizational services. The services described are intentionally generic so that they can be used as a basis for further, organization-specific, work. Some organizations will find that the generic services completely describe their technical architecture, whereas others will need to augment, replace, add, or delete services. The process of evolving the TRM services follows the architectural continuum (moving from left to right, from generic to organization-specific). The process for controlling this evolution is the architectural development method (ADM).

The TRM simply provides a catalog of all of the technical functions (or services, to use the TOGAF vocabulary) required to support business systems. It presents this in a visual representation (i.e., a reference model) and as a taxonomy, thus providing a method of describing the collection of technologies required to support the organization's business (and infrastructure) systems.

Key definition: **TOGAF taxonomy.**
Defines TOGAF-specific terminology and provides a coherent description of the components and conceptual structure of an IT environment and its set of services. The TOGAF taxonomy is depicted graphically through the Technical Reference Model.

An important concept for understanding exactly how the enterprise technical architecture differs from, say, an application architecture, is the *application platform*. The TRM—its catalogue of services—exists within the application platform. The platform defines the set of services required to support all custom-developed and package applications. The technical architecture is primarily interested in the platform, for it is the crux of the organization's IT environment and therefore represents its technical strategy. Using this basis, the definition of the platform comes

before the definition of individual applications; that is, applications plug into, and use the services of, the platform, not the other way around.

For instance, the application will require data management services (most likely via a database of some type). The contents of the database (i.e., the tables, attributes, rows, and columns) and the data model are application-specific; these exist in the TOGAF notion of an application. However, the database access mechanisms (such as the SQL API) and the formats of requests and responses transacted between the database and the application exist in the platform. At a foundation level, such database mechanisms exist as part of the data management services service. The fact that, once in the platform, they represent non application-specific services is the reason why they would affect the organization's technical direction. Generally, the organization will mandate to the application designers just what technologies will be used by the application to gain access to data management services.

The technical architecture is all about setting and controlling platform standards. But what about all those negative aspects associated with standards, in particular the affect on application architect and developer innovation? It is true that one of the central themes behind the enterprise services is commoditization and standardization. But standardization in this context has noble aims—the tackling of the endemic concept of "not invented here." How many security services does an organization need? Certainly not one per application. It is no secret that the more technologies delivering the same function the greater the total cost of IT to the organization. Regardless IT persists in reinventing technical functions because the existing functions were not built by the current faction. At some times this may be perfectly justifiable; however, in many circumstances it is merely a whim of the project team. Application development is all about business function, not building technical infrastructure.

Certainly, standardization can stifle innovation. However, the technical architecture is not targeted at the areas of an application architecture that respond to innovative practices—the domain-specific parts. The technical architecture attempts to identify those functions (or services) that exist within applications that can be considered commodity and not specific to an individual application (i.e., cross-domain). We believe that extracting such services from applications and standardizing these in the platform enables the application architects to continue to innovate and ensures the IT environment to meet its own strategic goals.

Figure 6.5 provides a high-level view of the TOGAF TRM and the application platform. Notice that the application uses the functionality within services of the platform via an interface (actually, a number of interfaces) of some type. Additionally, the application platform inter-

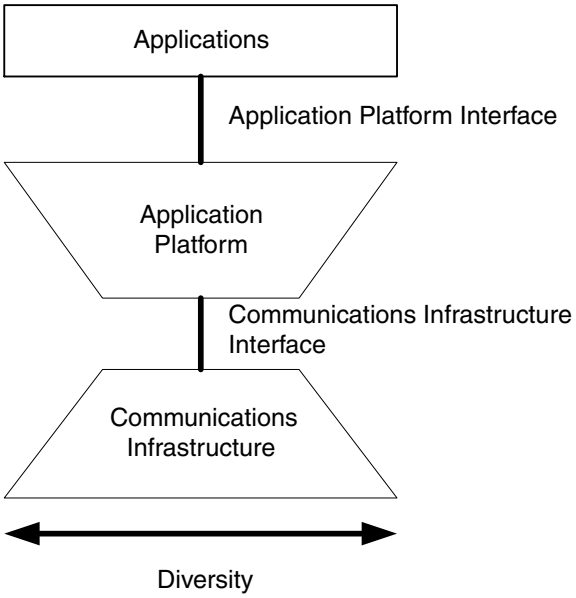

FIGURE 6.5. High-level view of the TRM.

faces directly with the communications infrastructure. The communications infrastructure consists of the physical elements that make up the networking environment, including network hardware and software and physical communications links. Note the fact that the application is sheltered from the physical network implementation by the services provided in the platform. This is a key strategy of the technical architecture.

The platform and service approach stress key architectural goals for the organization's technical architecture:

- *Portability*. Although a change in business requirements will necessitate some change to the application (possibly replacement), if the interface between the application and the platform service is standardized in some way, it is perfectly possible to replace a platform service without a major effect on the application.
- *Interoperability*. The ability of applications to interoperate through the use of common services and common interface semantics is a key driver for any organization wishing to use IT effectively.
- *Commonality*. The ability to recognize common functions within the IT environment, possibly derived from individual business systems, which should be "promoted" to the platform and used by all systems.
- *Architectural gap*. Understanding what platform services an organization should have and what they currently possess provides for a

robust gap-analysis process. Additionally, determining areas of technology overlap is easily described using this service approach.

- *Isolation.* It is increasingly important for contemporary applications to be isolated from the machinery of the IT environment. "Engines," such as middleware, are becoming increasingly complex. The days where business application developers coded directly to the network (say, with sockets APIs) are gone. Moving complex items such as object (or component) transaction monitors into the platform and providing a "simple" API is the aim of the TOGAF application platform and the TRM.

In turn the architectural goals allow the organization to achieve strategic goals such as lower TCO, improve information integrity, reduce time-to-market.

The application platform, as depicted in Figure 6.5, should not be viewed as a physical entity. It is purely conceptual and does not suggest a particular application or deployment style. However, most all applications can be conceptually described using this model. Consider, for instance, the client/server model. Client/server topologies (whether they be two-, three-, or n-tier) are characterized by multiple machines distributed around a network performing various application functions. Each application function (for example, supported within the web, object/component, or database server) will require the use of platform services. The functions will be sufficiently distributed to require access to a communications mechanism of some type. Figure 6.6 demonstrates a method for describing a two-tier client/server model in TRM terms. As can be seen, platform services exist to support all of the client or server components of the business system. Additionally, the application platform provides (and isolates) the application with communications services.

We have presented the generic construction of the TOGAF technical reference model (TRM) and the concept of the application platform. We have also seen how the platform (and hence the application) is made up of distinct technology functions (or services). To complete the TRM, the categories (or types) of technology services provided by the platform must be overlaid.

There are many ways to describe platform services. The TOGAF foundation set is just one approach. Others are available from alternate standards bodies, such as IEEE. Individual organizations may also have their own taxonomies. There is no leader in this area but rather several parallel approaches. It is entirely up to the individual organization to adopt what it sees as sensible. We use the TOGAF taxonomy in this book for a number of reasons:

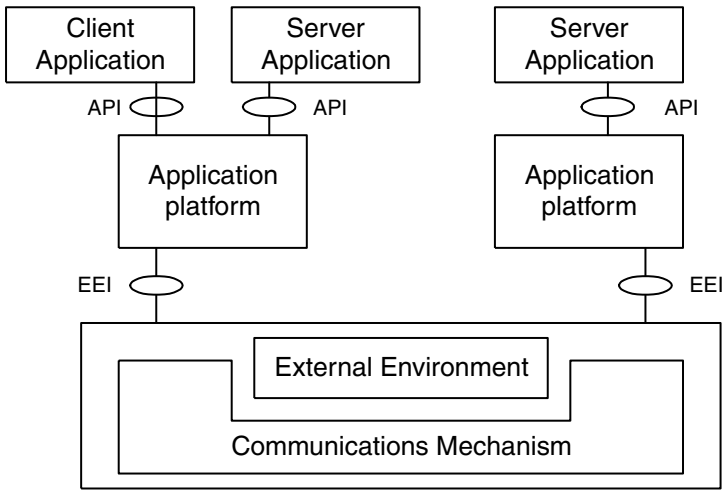

FIGURE 6.6. TRM for the client/server model.

- *Completeness.* The generic service categories are reasonably complete for most uses and provide an excellent basis to begin viewing the IT environment architecturally.
- *Extensibility.* Through the concept of the architectural continuum and the architectural development method, the generic TOGAF services can be modified, new services added to produce the organization's TRM.
- *The SIB.* A key advantage of the TOGAF services is that The Open Group has already defined an "organizational architecture" based directly on these services. The Open Group's standards information base provides an information source of all The Open Group's endorsed industry standards based on the its service taxonomy. This can be beneficial if the organization adheres to the standards provided by The Open Group.
- *Super services.* The TOGAF TRM easily supports our notion of "Super services"—architecturally significant services that use the foundation platform services.

The TRM was presented initially in Chapter 4 and is shown again in Figure 6.7. This figure focuses on the application platform showing the TOGAF foundation services. This taxonomy sits at the foundation architectures position on the architectural continuum (see Chapter 4 for an introduction to the architectural continuum).

The key conceptual components of The Open Group's TRM are:

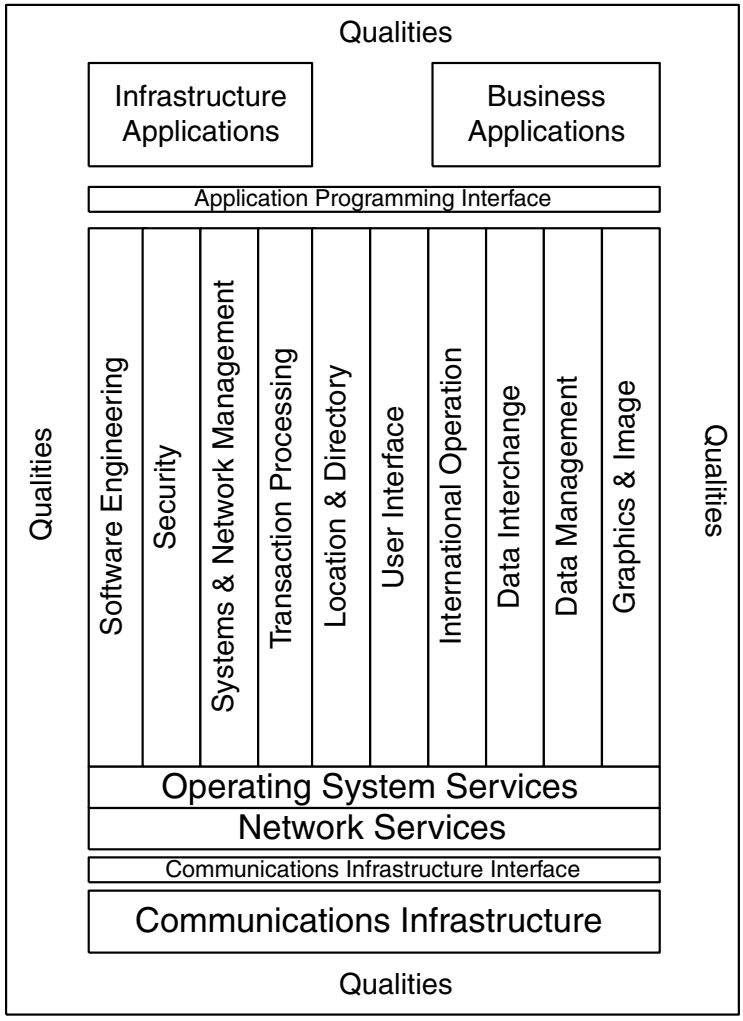

FIGURE 6.7. Detailed TRM showing service categories.

- Application software
- Application platform
- Communications infrastructure
- Two service interface types: the application programming and the communications infrastructure interfaces

Using this model it is possible to describe almost any IT system. Furthermore, it encapsulates all possible technology services required by all

systems in the IT environment (at least at the foundation-level). Any difference is unlikely to exist with the conceptual components but rather in the contents of the components. The TRM is critical to the enterprise technical architecture. Once complete the organizational TRM will represent the technology makeup of the organization and in essence define its technical direction. Let's consider in more detail the components of the TRM.

The Application Software

The application software component can be separated into two major categories:

- *Business applications*. These are the applications that directly support the business strategy and operational requirements. They are typically specific to a particular industry or vertical market. Examples include manufacturing, human resources, and business-to-business systems.
- *Infrastructure applications*. These are support applications, which provide the business with generic functions typically nonspecific to a particular industry. Examples include electronic mail, workflow, knowledge management, and so on. These applications are more likely to be common off-the-shelf products (COTS) and rely heavily on the underlying platform services.

The application programming interface (API) allows the application software to make use of the extant platform services. A single API may provide access to multiple services, or more than one API could be required to access a single service. This will depend on the technologies that will implement the services (selecting technologies to implement services is dealt with in later chapters).

The nature of the interfaces between applications and the platform has a huge bearing on the success of the technical architecture. Typically, the more rigorous the definition of this interface, the more likely the IT environment will support portability, interoperability, commonality, and isolation. The Open Group endorses and certifies a large number of robust API standards at this layer. It is the responsibility of each organization to determine how it will use industry standards. This point will be covered in greater detail in later chapters.

The Application Platform

We have already discussed the application platform at a high level. At this point we take a detailed look at its makeup. The application platform consists of a number of related services that provide support for the application software. TOGAF provides the architect with an initial baseline of services that will meet most general application software needs.

In TOGAF terms, the service baseline exists as part of the foundation architecture, at the foundation end of the architectural continuum. This is merely a starting point for architectural development. One of the underlying concepts of the architectural development method is to transition the application platform from the foundation architecture to an organization-specific architecture by augmenting, deleting, modifying, or adding services.

Coarseness of services is not specified, implying that the architect can adopt a granularity that is appropriate for the organization. For instance, a service may be as fine as an object (although this is unlikely to provide the requisite strategic intent), a component, a subsystem, or an entire system.

Physically, services can exist anywhere within an individual system; for instance they can be developed as part of a specific system, or they can be made more generic and used by more than one system (such as application frameworks). Conceptually, the TOGAF notion of platform services is those that have been moved out of the application and into "the platform" and hence standardized by the IT environment. This is the crux of the organization's technical strategy. Extract as many "common" services into the platform will ensure reuse and deliver to the organization's IT strategy. Platform services are then provided to the application or system through standard[22] interfaces (such as an application program interface—API).

Table 6.1 describes some of the key characteristics of platform services. This characterization is valid for all services identified during the architectural process, not just foundation services.

Key definition: **Services.**
Platform services are the crux of the organization's technical architecture. They define the functions the IT environment will provide all corporate applications.

Each service defined at the foundation level is not a specific technology (or product), rather it is a technology categorization. As the architectural is developed, the services will be realized by actual technologies. Again, at a foundation level, technologies are defined for these services in The Open Group's Standard Information Base (SIB). In the same way the services transition between the foundation and the organizational architecture, the technologies and the SIB will do likewise.

Throughout this text, we will refer to service categories as a descriptor for the high-level service areas pictorially described in the TRM and

22. The term "standard" used here does not imply an interface defined by an open standards body but merely an interface that is consistent for all corporate systems.

Table 6.1. Characteristics of platform services.

Service Characteristics
They primarily exist within "the platform"; that is, they tend to be generic, system-centered (as opposed to business-function-centered), and usable by cross-domain business applications.
They are encapsulated bundles of functionality.
They are accessed (by the application) through a published and stable interface (either based on standards, de facto, or proprietary specifications).
They typically interface with other services, providing a hierarchy (or embedding) of rich platform functionality. Some services interface directly with the external environment (i.e., network).
They can be replaced with other implementations (products, custom developments) without affecting the application as long as the interface is preserved.
They meet the service-quality requirements defined by the organization; for instance, performance or scalability.
The interfaces and internal machinery must adhere to the organization's architectural principles.
They typically do not implement organization-specific or domain-specific functionality (functions that are specific to an organization's business requirements).

service subcategories as the medium-level classification of individual services. Generically, when we use the term service we refer to both of these concepts. The following list shows the TOGAF foundation set of service categories and subcategories (the collection of which is called a service portfolio).

Key definition: **Service portfolio.**
The service portfolio is the collection of all services and their implementations that make up the organization's IT environment (or, in other words, the platform).

- Data Interchange Services
  - Document generic data typing and conversion services
  - Graphics data interchange services
  - Specialized data interchange services
  - Electronic data interchange services
  - Fax services
  - Raw graphics interface functions
  - Text processing functions
  - Document processing functions
  - Publishing functions

- Video processing functions
- Audio processing functions
- Multimedia processing functions
- Media synchronization functions
- Information presentation and distribution functions
- Hypertext functions

- Data Management Services
  - Data dictionary/repository services
  - Database management system (DBMS) services
  - Object-oriented database management system services
  - File management services
  - Query processing functions
  - Screen generation functions
  - Report generation functions
  - Networking/concurrent access functions
  - Warehousing functions

- Graphics and Imaging Services
  - Graphical object management services
  - Drawing services
  - Imaging functions

- International Operation Services
  - Character sets and data-representation services
  - Cultural convention services
  - Local-language support services

- Location and Directory Services
  - Directory services
  - Special-purpose naming services
  - Service location services
  - Registration services
  - Filtering services
  - Accounting services

- Network Services
  - Data communications services
  - Electronic mail services
  - Enhanced telephony functions
  - Shared screen functions
  - Video conferencing functions
  - Broadcast functions
  - Mailing list functions
  - Distributed time services

- Distributed data services
- Distributed file services
- Distributed name services
- Remote process (access) services
- Remote print spooling and output distribution services

- Operating System Services
  - Kernel operations services
  - Command interpreter and utility services
  - Batch processing services
  - File and directory synchronization services

- Software Engineering Services
  - Programming language services
  - Object code linking services
  - Computer-aided software engineering (CASE) environment and tools services
  - Graphical user interface (GUI) building services
  - Scripting language services
  - Language binding services
  - Run-time environment services
  - Application binary interface services

- Transaction Processing Services
  - Transaction manager services

- User Interface Services
  - Graphical client/server services
  - Display objects services
  - Window management services
  - Dialogue support services
  - Printing services
  - Computer-based training and on-line help services
  - Character-based services

- Security Services
  - Identification and authentication services
  - System entry control services
  - Audit services
  - Access control services
  - Nonrepudiation services
  - Security management services
  - Trusted recovery services
  - Encryption services
  - Trusted communication services

- System and Network Management Services
  - User management services
  - Configuration management (CM) services
  - Performance management services
  - Availability and fault management services
  - Accounting management services
  - Security management services
  - Print management services
  - Network management services
  - Backup and restore services
  - On-line disk management services
  - License management services
  - Capacity management services
  - Software installation services
  - Trouble ticketing functions

In the same way that the application software uses the platform services via APIs, the services within the platform may make use of each other in a similar fashion. Some inter-service interfaces may be defined and published, whereas others may use private or unexposed interfaces. The valid interaction between services is a vital component of the integrity of the IT environment.

An interesting characteristic of services is the delineation from application software. In the past, we may have seen applications that implemented their own network services or e-mail services, or even developed their own programming environment. This usually occurred when such services were either not available on the platform or possibly not functional enough for the application. However, as services became more ubiquitous and more common to many applications, they slowly began to migrate into the platform, where the application merely has to issue an API call to use the service. An example of this migration is the network services. In the past programmers typically dealt directly within the TCP/IP stack (possibly at a socket level) to exchange information between distributed application components. Modern software platforms provide much higher layer interfaces to the network layers (such as Object Request Broker interfaces, WWW APIs, and others) isolating the programmer from the details of the network.

Analogous to biological evolution, this technology migration process is continually occurring. As new technologies appear, they are typically provided directly within applications. The style of the technology will then dictate how it will evolve. If they are seen as usable by a wide range of applications (i.e., if there is a large market into which they can be developed), they will become increasingly common (possibly even standards), and a greater vendor population will wish to get on the gravy

train. At some point in this process, these technologies will be considered commodity and will be "formally" considered part of the platform. Other technologies that remain niche will continue to exist in high-cost, low-penetration applications and remain outside the platform. Workflow and decision support are examples of "niche" services that are still predominantly specialized and used by niche organizational applications. For both of these technologies, increased penetration is required to support their move to commodity status and hence to the platform. The advent of Web-technology based workflow may indicate the beginning of this process.

This brings us to a discussion on super services. We introduced this concept earlier in this chapter, mentioning that they are a collection and extension of the current platform services. A super service shares common themes with both infrastructure applications and services. We see them as a further refined (or even summarized) version of these two TOGAF concepts. For example, take the notion of object-oriented or component-oriented technologies. Most of the base technologies that provide object-oriented services already exist under the foundation services categories—they include transaction services, network services, data interchange services, and so on. However, clustering these services under an object-oriented super service can provide a convenient and useful categorization in its own right. In this way the platform can support a layer of services—the base (foundation) and the super services. The clustering of services can prove very useful when it comes to assessing products. For example, the current application-server market tends to sell an entire bundle of services as a single package. Assessing the technology against the relevant super service can aid the selection process. Super services are dealt with in detail in Chapter 9.

There is one further service category that is worthy of a mention at this point; namely, service qualities. These are the often-overlooked "nonfunctional" aspects of the architecture. Qualities consist of those operational and strategic characteristics of both individual systems and the entire technical environment that represent an organization's view of a quality IT environment. They include such components as (and are generally known as the –abilities):

- *Availability*—including manageability, serviceability, performance, reliability, recoverability, and locatability
- *Assurance*—including security, integrity, and credibility
- *Usability*—includes international operation
- *Adaptability*—including interoperability, scalability, portability, and extensibility

Qualities relate both to the application platform services and the application themselves, which is why they are represented in the TRM. In

some respects, individual quality components are in conflict with each other. For example, manageability can affect security, or reliability can affect performance. However, it is important to be able to recognize these qualities in the current environment so that a complete analysis is achieved. We have used these quality aspects to lead the definition of an organization's strategic architectural principles. For example, an organization should have a definitive statement on portability. Is this a key factor? If it is, this will be a significant driver when selecting technologies and products. Another example is manageability. The quality of the environment would be severely affected if an organization had a limited understanding of how the environment should be managed. This is not just applying management technologies (which would be defined in the systems management service) but how these technologies are applied. Service qualities tend not to be technologies in their own right but rather they may be requirements, policies, guidelines, quality assurance checks, and so forth.

Communications Infrastructure

The communications infrastructure is now a ubiquitous part of all organizations' IT environment, whether it be a simple Internet connection or a worldwide corporate network. The communications environment provides the physical components to connect application platforms together. There is virtually no business system today that does not in some way rely on a networking medium. In fact, this reliance on the network will only increase as microdistribution of business system components continues to advance. This has had a significant impact on the network. In the past, the network was a strategic (and costly) corporate asset; today, it is becoming increasingly commoditized

Accepting this change in focus of the network, the TRM models the physical network as a service to the common platform services (not connected directly to the applications themselves). The infrastructure itself is the collection of network links, software, and hardware that make up the physical networks. Generally, the network service provides the technologies and interfaces that integrate with the physical network.

Commodity communications infrastructure interfaces exist between the platform and the network. Used by the platform services, these interfaces enable the higher-layer services to interact with the physical network medium.

It may be difficult to see where the network services stop and the physical network starts. For example, is ATM a network service or a physical network? There is no hard and fast rule on how the architect should pigeonhole the technologies within the TRM. Our call here is that ATM makes up the physical network as it is implemented within the

hardware and software of the network, and the network service will include a technology that supports the formats and protocols for an application to use the "ATM network." However, this sort of classification is up to the individual. The only advice is to be consistent.

Views

Architectural views are an excellent approach for analyzing both a current environment and the target environment. We will be describing views in more detail in the target architecture chapters (and dedicate an entire chapter—Chapter 7—to a discussion on views), but a brief description of them here will demonstrate their usefulness in this phase.

Views are essentially slices through the architecture at different points. Typically, if we were to ask IT people to describe the makeup of a particular business system, their perception will be "tainted" by their particular expertise. For instance, the database developer will provide a different analysis of the system than the OO designer or security administrator. These perspectives are merely different views of the system, and each is important in building up a full picture of the system and environment. There are a number of views that can be taken of the current systems in this phase. The architect may consider only a subset important or that additional views should be added. The point here is, again, to ensure that a complete picture of the systems is gained before moving to the next phase. We have found the following views to be useful.

- Functional view. This could be obtained from the business systems architecture, if available.
- Builder's view. Takes a software development focus.
- Security view. Builds an understanding of the security requirements and implementation.
- Computing view. Includes how the hardware and software components are distributed within the organization. This view can also include a Communications view.
- Management view. Looks both at the quality aspects and how management is implemented.
- User view. Considers the user's interaction with the environment.

However, do not be constrained either by the views outlined here or those in TOGAF. Formulating alternate views that reflect the nature of the assignment or problem is encouraged. For instance, a commercial view may be important, so might an organizational view. In this area, there is no wrong answer; whatever is useful in providing input into the current environment analysis should be used.

6.5 Current System Issues

Balancing architectural decisions is a key facet of the development of the technical architecture. Understanding the constraints imposed on the architecture, determining necessary tradeoffs, and factoring sensitivities are tools we will use later when analyzing the target architecture. However, this analysis is also important in understanding the makeup of the current environment. Reviewing the various "architectural" or strategic processes available for describing the IT environment, it is noted that current state analysis plays an important role in understanding the problems to be solved. For instance:

- The information architecture considered how the current information needs were being met. Analysis of the current information uses and the organization's requirements identified information gaps. Gaps meant that additional business systems might be required to fulfill them.
- The business systems architecture determined business system information deficiencies. Additionally, review of the current business systems may have led to a decision to divest in favor of a new system.

The technical architecture analyzes the current environment determining both the technical gaps and technical deficiencies. As always it is important that this analysis be done in context with both the business and IT strategic directions (including the information and business systems architectures). For instance, there is no point in analyzing a system that the business systems architecture has already doomed to replacement or to recommend extending the internally managed network to more sites if the organization has decided to centralize its operations and attempt to compete on cost.

When describing and analyzing the current systems, the architect should therefore always be reflecting back on the business and technical strategies. This is the main purpose of the initiation and framework phase—establishing the strategic context.

There are a number of good analysis methods available, but the most important of these is the architect's experience and knowledge. Although a fairly limp phrase, applying the concept of "best practice" effectively is the secret of analysis in the context of the technical architecture, the context being the macro view. This experience manifests itself as knowledge of:

- Common IT industry practice and likely directions
- Trends in technology areas

- Access to vendor organizations, or trusted vendor analysts
- Knowledge of the specific vertical market and its associated technologies
- Knowledge of the business strategies

While collecting information on the current environment, the architect will be assessing this against his or her knowledge and experience. The objective is to uncover areas of the current environment that appear ineffective, misguided, or lacking focus. Typically, for an experienced architect these areas of concern will be fairly obvious. Most organizations tend to have similar problems, with only the scale and the technology being different. For instance, the following are common generic problems:

- Technology overlap—more than one technology filling a service role.
- Technology gaps—a defined service function that is not supported by technology.
- Policy gaps—colloquial guidance that needs to be institutionalized.
- Integration issues—applications and service that do not inherently interoperate.

The approach then is to further analyze these affected areas. Firstly, compare them with the business and technical strategies to understand the scope or likely effect of the problem. A technical deficiency that relates directly to a strategic objective will have a far higher impact than one associated with a purely operational area of the business.

Avoid digging deeply into the architecture of the individual systems, unearthing problems with application internals. This is the responsibility of the application architect and has little bearing on the overall environment unless an application service is being considered for migration into the platform. Although the application architect can describe the problems with the architecture of the system, the technical architect must view the individual system as part of an entire technical environment. The architect should be thinking in terms of the boundary of the application and how it interacts with the environment. This is one of the important concepts of the services portfolio and why the application itself is represented merely as a single block in the TRM.

We have found it useful to analyze the systems and the environment in alignment with the organization's specific (TOGAF) service portfolio. The first architectural project obviously will not have this artifact available, in which case using the TOGAF foundation service portfolio will suffice. This approach is constructive because it ensures that the architect does not miss key service areas when assessing systems. Tackling systems purely from the point of view of the information available has

the effect of creating gaps in the analysis merely because the information is not there. Therefore, viewing each system in terms of the entire suite of platform services—viewing slices through the system if you will—ensures that no information is missed. For each system, and the enterprise environment, consider completing a table using criteria similar to those defined in Table 6.2.

There are a number of format methods available that the architect may find valuable in current system assessment. The TOGAF specification refers to two methods for analyzing architectural views:

- IEEE P1471—Recommended Practice for Architectural Description.
- The Architectural Tradeoff Analysis Method (ATAM) from the Software Engineering Institute and Carnegie Mellon University.

There are significant research papers on the ATAM, and its method provides a robust process driven approach to discover architectural problems based on the tradeoff of various different system attributes. The basic process is summarized in Figure 6.8.[23] ATAM provides a micro process to analyze system architectures in detail. It fits easily into TOGAF as an additional support tool for the overall architecture development method.

The key point of the current system analysis is to describe the issues that exist with the current environment. This provides the focus for the architectural work to follow. In essence, it provides the fuel for technical change. Once the issues have been discovered, the final step is to align them with the business and technical strategies so that they may be prioritized. This prioritization step ensures that the target architecture is concentrating on technical change that provides real business benefit rather than for technology's sake alone.

Table 6.2. System assessment criteria.

System: <system or environment name>				
Service Category	Summary of Technology that Provides Service	Issues Noted	Effect on Arch Principles, IT Strategy	Tech Assessment
<service cat name>				
<service cat name>				

23. For more information on ATAM, see www.sei.cmu.edu/ata.

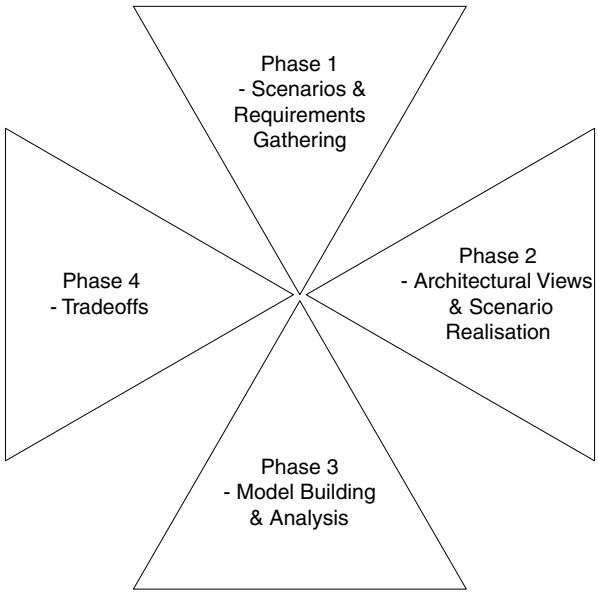

FIGURE 6.8. Summary of ATAM.

6.6 Outputs

The final stage of the baseline phase is to ensure that all of the necessary documentation is up-to-date. The baseline will need to be documented in a form that allows for easy reference during later stages. Typically, such a document will describe each system and each infrastructure component in individual sections, stating necessary facts and analysis about each. All issues identified and described within these sections should be collected and summarized in the assessment section, including all rationales for highlighting issues. Each issue should be linked with either business or IT strategies that it affects. In TOGAF terms, this document is known as the technical architecture document. This document is a logical construct only; how it is physically described is a matter for the organization to decide. It is carried through the entire ADM, augmented at each phase, and is provided as the final key deliverable. This phase also adds to the business architecture. Again, this is a logical document only and consists primarily of information from the business systems architecture and the architectural TOR.

The final documentation tasks update the constraints, assumptions, or requirements documented during the first phase (initiation and

framework). Typically, this will require updating the TOR or the architectural statement of work, if necessary. Remember that it is likely that the baseline activities will uncover further constraints that must be captured, and these are most likely to be technical rather than organizational. Be aware that should these new constraints have significant impact on the target architecture, they should be discussed with the architectural governance board with the aim of prioritizing their impact.

6.7 Example

Introduction

This section continues CFL's architecture program with the baseline phase. We do not provide a complete treatment of the CFL baseline but merely examples of some approaches in describing the current system baseline.

The CFL architectural program has progressed through the initiation phase as described in the previous chapter. The foundation documents describing both the project (the architectural statement of work) and the CFL business environment (the business architecture) have been produced. In this phase, the baseline, the next step is to delve into the current environment to provide the foundation information for the target architecture phase. During this phase, we will also update the "Phase A" deliverables if necessary.

Current System Background

Our first step, after reviewing the initiation documentation, is to map out a plan to capture the necessary current system information. We already knew from the ISSP that the CFL IT environment was separated into two streams, the legacy and the contemporary environments. The legacy environment appears to be characterized by stable management and mainframe-like disciplines. Gaining the necessary information should be reasonably straightforward here. The contemporary environment appears chaotic. Finding the key systems is going to be a challenge, as will be finding the correct people to talk in order to gain this information.

The ISSP also provided us with some key areas of focus, for instance:

- Total cost of ownership (TCO). The CFL systems and environment have a high cost-to-value ratio. This ratio must be reversed either by lowering cost or increasing value (or both).

- The Internet. The ability of the organization to implement its e-business strategy is hampered by the lack of an Internet-ready environment.
- Systems management. User complaints and systems unreliability must be redressed through more proactive management.
- Technology duplication. A number of infrastructure applications are duplicated and do not cooperate. This also leads to TCO problems.
- Security. There is little control over the security of information within the environment.
- Reporting. Current reporting methods are not timely and are costly to produce.

The business systems architecture also provides some valuable insight into our areas of focus. The key points gleaned from this are:

- The need to support global operations
- The ability to reduce the cost of transactions
- The ability to provide CFL with the capability to develop faster solutions for future business problems
- The need to improve customer convenience
- Highlighting the major issues associated with duplication of core CFL information, including customer and product
- That retention of the core CFL systems was acceptable

The e-business strategy, a key foundation for the future of CFL, must also been considered.

Our information gathering plan looks like this:

- Make a detailed assessment of the "corporate" systems located at HQ, interviewing system support personnel (including help desk), system owners, and vendors where appropriate.
- Construct a questionnaire to be sent to remote office support personnel (or vendors) or office managers if no on-site support is provided.

We have decided to use native views to describe the entire environment. This is likely to facilitate timely gathering of information on the legacy (HQ-based) systems. Also this will not place any undue overhead on the remote offices when it comes to answering the questionnaire.

CFL's Current Systems

Network

We obtained the definitive network diagram from the Operations group, a summary of which is shown in Figure 6.9. Although it does not show

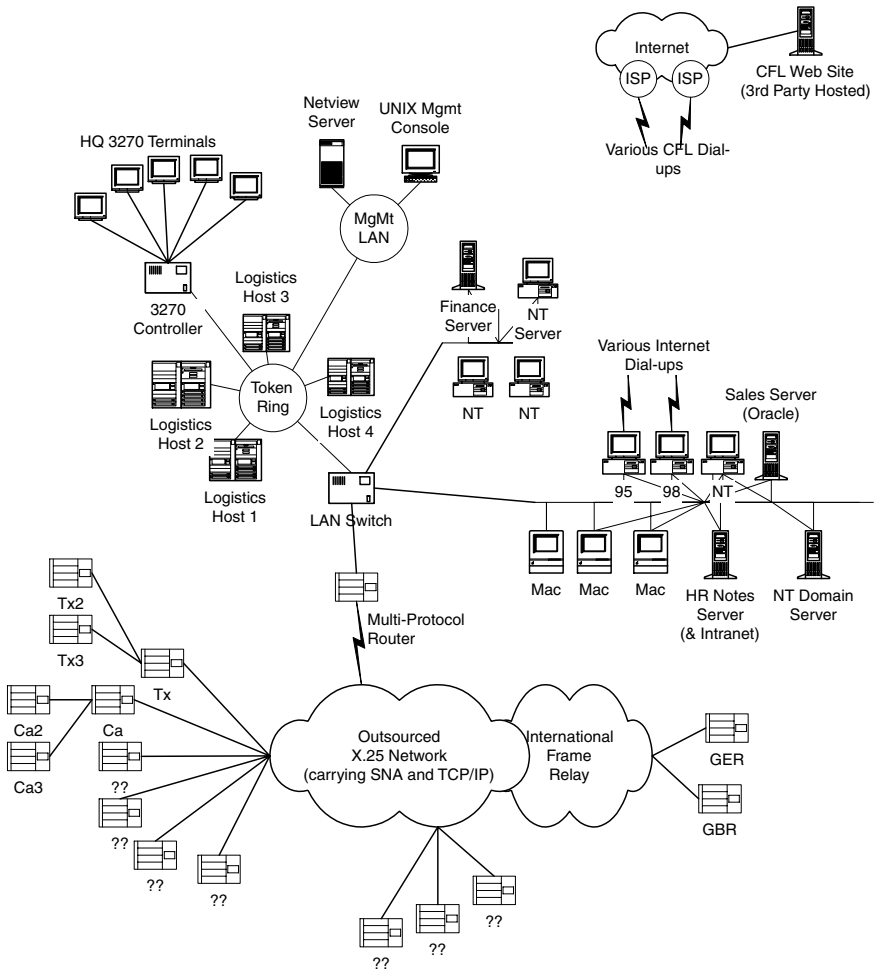

FIGURE 6.9. Network topology.

a large amount of detail, it does allow us to understand the general environment. Additionally, it is interesting to note that the view of the environment is HQ-centric. From HQ's perspective, control appears to terminate at the regional routers, as in this diagram.

Coupled with this diagram, we were also able to obtain a detailed set of documents describing:

- The network hardware and software, including versions
- A list of all protocols supported

- Detailed service-level agreements, and vendor contracts
- Detailed configuration information
- A schematic of the Netview network management topology, including the properties of all managed objects (most of these consisted of the network equipment and the AS/400 systems)

Logistics System

The Logistics system is the key corporate business application, managing the manufacturing and materials requirements. Although it can be considered legacy, there appear to be no current plans to replace it (see the business systems architecture in initiation and framework). CFL continues to maintain considerable expertise in-house with both the application itself and the AS/400 technologies. We interviewed the systems analyst, the technical support leader, and a key business user to gain a better understanding of the system. Its key technical characteristics include:

- Mainframe-based architecture—"green screen" clients connected to several AS/400s. All business logic, data access logic, and data are maintained on the AS/400s.
- The system was developed in COBOL 74. The database has recently been ported to DB/2.
- Presentation is provided in 3270 data streams. Although there remain a number of terminals at HQ, all other regions use 3270 emulation software on PCs to gain access to the system.
- Logistics supports only SNA for client terminal access. However, recently a TCP/IP stack was installed on all hosts to allow for network management via Netview.
- The system has rigid response time and availability requirements. In summary, all on-line transactions must execute in 2 seconds or less, and the system (measured at the AS/400 end) must exhibit greater than 99.95% uptime.
- There are four AS/400s. One production on-line machine, one product batch and reporting machine, and two development/testing machines that double as hot-standby systems. Data are replicated between systems nightly.

Sales System

Information on the Sales system was gained by interviewing Development group developers, the Sales VP, and a key system user. Additionally, the data model provided for the information architecture was also analyzed with the corporate information architect.

The Sales system, developed in-house, is a recent addition to the enterprise applications portfolio. An analysis of CFL customers, which included a satisfaction survey, indicated that customer knowledge was

ineffectively managed by regional sales teams. CFL regularly held misleading or outdated knowledge about customer activities (most of which had to come from the Logistics system via paper output). The CRM initiative was designed to increase the accuracy of customer information in the field, better manage promotions, and track customer interaction with CFL.

A "computing view" was obtained from the Development group. This is shown in Figure 6.10. Salient points are:

- The Sales system is a two-tier Oracle Forms application, modeled in and partially generated from Oracle Designer.
- There are at least two "static" PCs in each office that have access to the system. The limited network bandwidth (and SNA priorities) and the two-tier nature of the system limited the number of users that could be connected at any one time. This has been alleviated marginally through the use of Oracle's Multi-Threaded Server; however, the bottlenecks continue to be an issue.
- Many of the Sales users have a personal copy of the application running on their laptops. The laptop version consists of a sophisticated replication controller that allows salespeople to download core information about their customer to their CRM-lite database. They are able to work offline, including updating customer information. At regular intervals, they are encouraged to upload changed information to an offline replica database on the Sales server. This database contained significant rules that check for inconsistencies between the replica's information and that held on the online system.
- Laptop users can use a number of techniques to replicate information, including dial-up, Internet, and docked on the corporate network. The replication controller uses FTP to transport the information.
- The system does not make use of any management tools. All systems management is carried out by a local Development group resource. Any problems with a salesperson's laptop are either fixed over the phone or the laptop is sent back to HQ.

Interfaces

Gaining an understanding of the interfaces between systems proved to be difficult. We could not identify a single owner for the interfaces in general, and there were no intersystem contracts describing the interfaces. Most interfaces are run within the Operations group via standard batch jobs. Almost all required physical intervention to move generated files between systems. There were few error-recovery mechanisms in place. All systems (apart from HR) maintained a subset of customer information. Frequently, the view of a customer was different in each

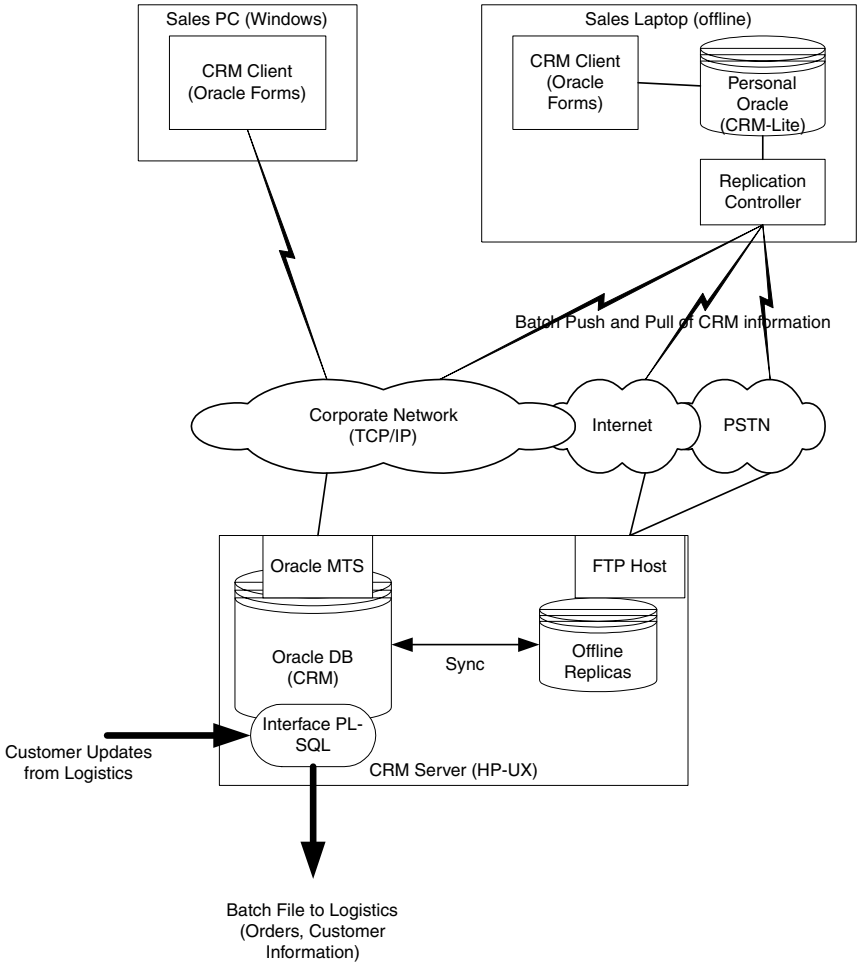

FIGURE 6.10. Sales system computing view.

system. During an interview with the Operations Manager, she indicated that the interfaces were "a mess"—each new interface requirement caused another point-to-point solution and exacerbated the problem. The help desk noted a number of irate customer calls relating to the inaccuracy of CFL's information due to interface difficulties.

We sketched a high-level diagram of the interface architecture, shown in Figure 6.11, from information provided by a number of support people in both the Operations and Development groups. We could not discover a detailed description of each interface (interchange formats, scheduling requirements, and so on).

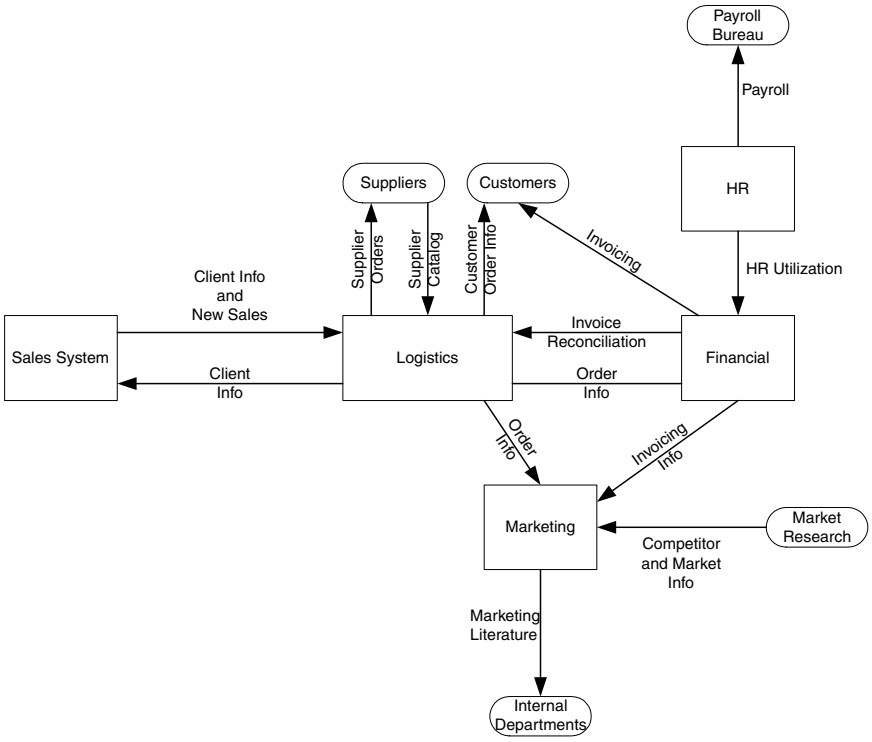

FIGURE 6.11. Interface topology.

Regional Systems

We constructed a number of different questionnaires to support the discovery of the regional IT environments. Detailed and specialist technical questions were distributed to the known regional support personnel, the objective being to understand as much as possible about the technology environment. More general questions were sent to regional managers to gather an understanding of both the business systems regularly used at the site and the opinions of the users as to the effectiveness of the systems.

Figure 6.12 shows an example of infrastructure information received from the Texas regional HQ. This information highlighted that there was considerably more technology deployed within the regions than suspected, including a growing number of dialup Internet users. Additionally, although no new "corporate applications" were uncovered, the survey unearthed a large number of small regional applications supporting key regional functions. Many of these small applications were taking

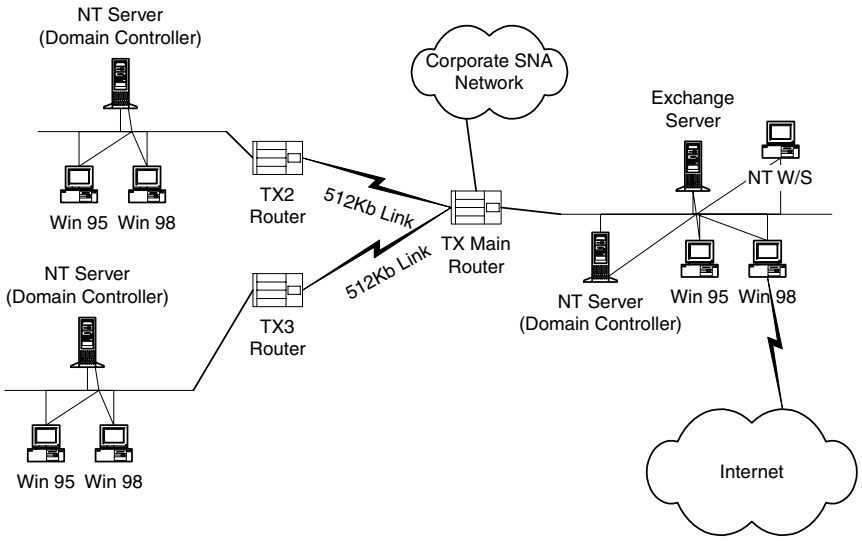

FIGURE 6.12. Texas system infrastructure.

information feeds from corporate systems and presenting them in a more usable form for the users.

Current System Assessment

In analyzing the current systems, we discovered and confirmed many of the problems described in the ISSP. Given our in-depth analysis, we were able to document them to a greater extent and understand some of the areas where changes would be necessary.

There were a few areas identified in the current system assessment that were not picked up in the ISSP. These were intersystem integration and security. This section looks at these two issues in more detail.

Additionally, through the interview process, we deduced further architectural constraints that were not already captured. The architectural TOR was updated to reflect this new information. The constraints identified were:

- IT management has just signed a hardware and operating system support contract with IBM for support of the Logistics system and the Logistics systems management environment (Netview). The contract runs for 4 years.
- The Marketing department is actively pursuing direct Internet-based customer ordering. This is in response to a competitor's efforts in

Service Category Information Interchange
Summary of Technology that Provides Service
• Mostly batch integration between systems • Some ad hoc file transfer used • Sales system supports a custom developed replication systems
Issues Noted
• Unreliable • No feedback mechanism • n to n interface methods • Requires significant manual intervention • Costly to maintain integrity
Effect on Arch Principles, IT Strategy
• Management costs are high—negative effect on TCO • Leads to information duplication across systems—corrupts corporate architecture • Error prone—affects interface quality • Reduces ability to change systems in a timely manner • Limited the ability to integrate e-business applications into the corporate environment and therefore reduce supply chain costs • Overall: major negative effect on achieving strategy
Tech Assessment (hype curve)
• N/A

this area. This service has been pitched to customers (mostly large supermarkets) for initial release in 6 months.

6.8 Summary

In this chapter, we concentrated on baselining the current environment. This is a key activity in understanding the systems and infrastructure components that make up the current environment and how they align with the business, IT, and architectural directions. We looked at mechanisms for describing the current environment, including in both native and TOGAF forms. From the TOGAF perspective, we introduced in more detail the concept of a service portfolio and its relationship with the standard technical reference model. Finally, using CFL, we provided some insight into what a baseline might look like.

In the next chapter, we take a more detailed look at architectural views.

Service Category Security
Summary of Technology that Provides Service
• Security services specific to each application (e.g. Oracle DB security for Sales, RACF for Logistics) • NT security provided in sites that have implemented NT domains • Netware security provided in sites that have implemented Novell • Lotus Notes public-key-based security • Plus a number of others
Issues Noted
• User credentials duplicated in a large number of security repositories • Users must remember a number of different passwords and user IDs to gain access to systems • There is no overarching security policy governing security implementation, and therefore each system implements security to its own requirements • There are little or no audit mechanisms to ensure that information security is maintained and that no attacks have occurred • Ad hoc Internet connections have the ability to compromise internal security—there are no secure gateways preventing this • The security environment is not robust enough, and does not have sufficient functionality to support e-business initiatives
Effect on Arch Principles, IT Strategy
• Unable to assure the integrity of CFL information or environment • Limits the ability to implement Internet-based applications and supply chain integration • Increases TCO due to maintaining numerous user repositories and handling help desk calls relating to forgotten passwords • Reduces system and environment usability • Overall: major negative effect on achieving strategy
Tech Assessment (hype curve)
• Most technologies providing security services are mature