CHAPTER 10

Management Integration: Putting the Pieces Together

As we saw in earlier chapters, managing a network involves a great variety of functions—from monitoring devices in the network to provisioning services, from diagnosing networking problems to planning for optimum network performance, from detecting security breaches to assessing the impact of planned network maintenance on existing services and customers.

One of the challenges in network management—indeed, some would argue, the "holy grail" in network management—lies in providing operational support infrastructure and management systems that are integrated. This means that all management functionality that is required for everything that needs to be managed is provided in one holistic solution, as opposed to providing the functionality in multiple, separate parts that essentially form separate islands. Having multiple management islands can cause many problems that could be avoided with an integrated solution: Data needs to be maintained redundantly and can run out of synch, training cost increases for operational staff that needs to be familiar with a multitude of systems, and management tasks fall through the cracks.

In this chapter, we take a closer look at the challenges that are associated with integrated management. Recognizing what those challenges are is the first step in confronting them successfully. The chapter also discusses some techniques that can be used to tackle those challenges and some of the trade-offs that they involve. In the course of the discussion, we start putting together many of the pieces from the earlier chapters.

Here are some of the things that you will learn by reading this chapter:

- Get to know many of the factors that make management integration a challenge, which is a prerequisite for being able to deal with them successfully
- Understand how the different management dimensions encountered in earlier chapters can be used to approach management integration
- Recognize trade-offs between platform- and component-based integration approaches, and between tight and loose management integration
- Learn about approaches that can help you reduce the complexity of management tasks you might face

The Need for Management Integration

In practice, the diversity of things to be managed and the diversity of functions needed for management easily lead to a diverse set of management applications that are used to manage a network. Management integration aims to provide an operations support environment in which management functionality is seamlessly integrated and holistic, end-to-end management support is provided.

Before we get into what the challenges of management integration are and how those challenges can be successfully approached, let us start by taking a look at the various reasons why management integration is of such importance. To that end, let us set the stage by discussing the benefits that are to be gained from management that is integrated compared to management that is not. We also explain why management integration is not just a technical problem. As the saying goes, "Beauty lies in the eye of the beholder." Similarly, what constitutes management integration lies in the eye of the beholder—or, rather, the issues that need to be addressed as part of integrated management depend in part on the perspective of the party involved.

Benefits of Integrated Management

Having management that is integrated—as opposed to management that is based on a piecemeal approach that consists of multiple management "islands"—is important for many reasons that include the following:

- It helps ensure that management tasks do not fall through the cracks. Management tasks that are supported by a holistic, integrated operational support environment do not need to rely as much on manual procedures and leave little to chance, compared to management tasks that are not supported by such an environment.
- Integrated management systems and holistic operational support environments (from here on, simply referred to as integrated management infrastructure) reduce the need for training and increase the pool of available personnel that can carry out operational tasks. With integrated management systems, operators need to be well versed in fewer systems and machine interfaces.
- Integrated management infrastructure facilitates management of the management itself—that is, of the management systems and management network that need to be managed in addition to the production network itself. Management environments that are not integrated require much more manual administration, supervision, and intervention to keep network operations running smoothly.
- Integrated management infrastructure eliminates (or at least reduces) the need for operators and network administrators to enter redundant data. For example, in a nonintegrated management environment, information about which network elements need to be managed

and how to reach them (IP addresses, user credentials) frequently has to be entered multiple times, potentially into every management application that needs to know about the network elements. The result is lower operations efficiency: After all, entering this information takes time and effort. Even worse, it is error prone.

- In the same vein as the previous point, integrated management infrastructure reduces or eliminates the need to keep the same data redundantly in multiple locations, such as in separate management applications. Maintaining redundant data can be an issue, even when it does not have to be entered redundantly, because that data can potentially run out of synch. When it does run out of synch, cleanup can be a mess.

- Integrated management infrastructure helps reduce the management load on the managed network. In nonintegrated management environments, different applications might all need to query a managed device for the same management information. This not only wastes bandwidth on the management network, but it also causes avoidable CPU cycles spent by the device responding to management queries instead of passing network traffic. This additional load can be quite significant, particularly when applications rely on frequent polling.

- Integrated management infrastructure makes it easier to have management information available whenever and wherever it is needed, sometimes in conjunction with applications where it might not be expected at first. For example, the same data about existing network inventory might need to be accessed for the following very different purposes:

  — Network planning (determine additional needs based on what is already available in the network)

  — Network monitoring (know what to monitor)

  — Service provisioning (determine network equipment that needs to be configured to carry an instance of a service)

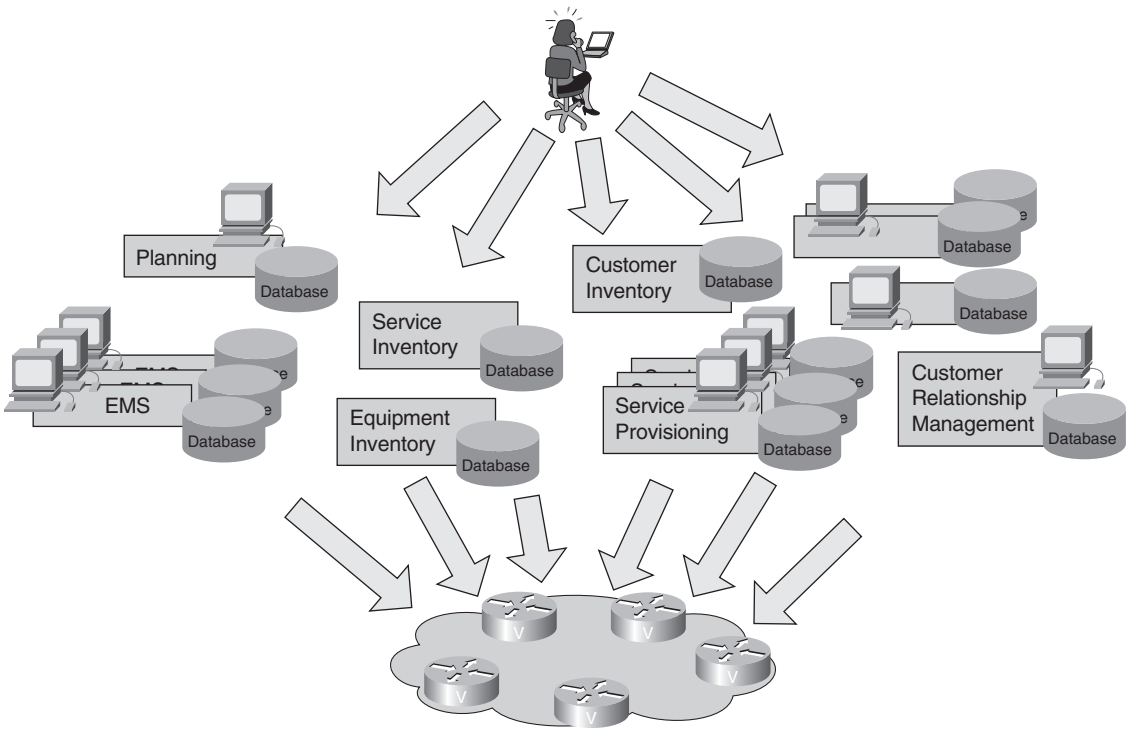
  — Help desk (be able to trace a problem with the level of service that a user is experiencing to possible culprits in terms of network equipment that might cause the problem)

- Integration can help feed itself, in the sense that management infrastructure that is already integrated will be easier to integrate with other parts of the business if the need arises, fostering further integration. There will be only one system with (hopefully) one well-defined interface to interact with instead of a hodgepodge of different components with all kinds of interdependencies.

In other words, management that is integrated results in management that is also much more efficient than it would otherwise be. Compare the situation depicted in Figures 10-1 and 10-2. In Figure 10-1, the operator has to deal with a multitude of different systems, each with its own user interface and database. Clearly, this situation is intimidating, if not overwhelming. In addition, the

devices in the network will be hit with requests from multiple directions. In Figure 10-2, this situation has been replaced by one that is much simpler. The complexity has been absorbed by a single integrated management system—elusive perhaps, but the subject of the management integration quest.

Figure 10-1 *Nonintegrated Management, All Too Often Management's Reality*



Nontechnical Considerations for Management Integration

This book deals mainly with management technology, so our discussion of management integration focuses on its technical aspects. However, it should be mentioned that management integration is not only a technical problem involving management systems and applications. There is also significant organizational dimension that involves the structure of the network provider organization that manages the network. In fact, the issues with management integration at the technical level mirror in many ways the issues that can occur at the organizational level, and the approaches to dealing with those issues need to take similar aspects into considerations, as illustrated in Figure 10-3.

Figure 10-2 *Integrated Management, Management's "Holy Grail"*

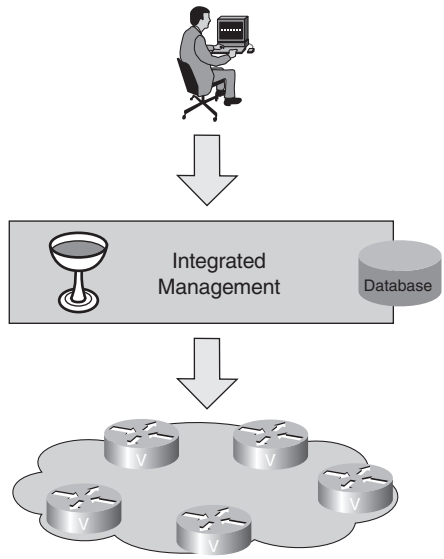
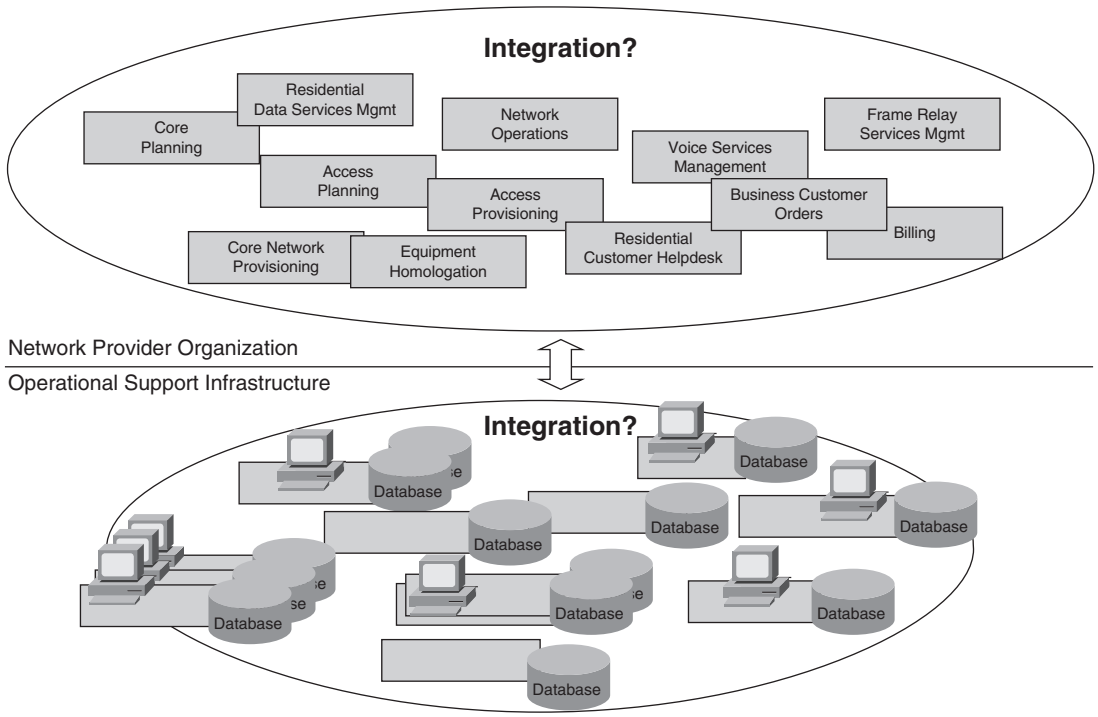


Figure 10-3 *Duality of Management Integration Problem: Organization vs. Operational Support Infrastructure*



For example, imagine a situation in which the responsibility for network operations is distributed across several groups and none of them feels responsible for a particular problem that comes up during the running of the network. This spells trouble for the provider of the network. The same is true if different groups want to use the same piece of equipment for different purposes but fail to communicate with each other, resulting in the groups breaking each other's network configuration.

These might be drastic examples. However, tasks falling between the cracks and mutual finger pointing between groups are problems that are not unheard of in network provider organizations. Integrated management implies that there is also a need for the different organizations that are running the network to be "integrated" and complement each other to function efficiently.

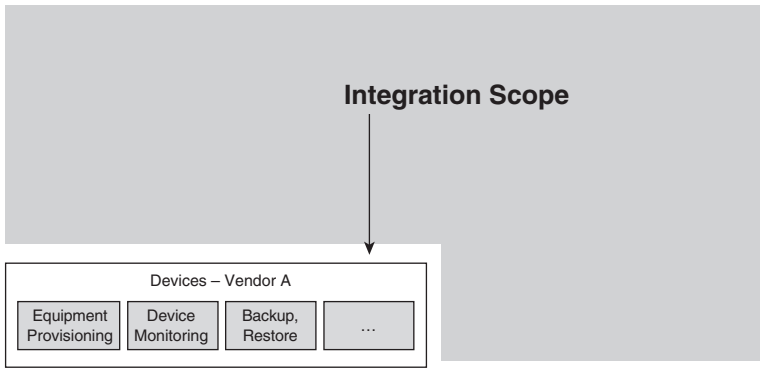
Different Perspectives on Management Integration Needs

Let us now take a look at who has an interest in integrated management and why—in other words, the different perspectives from which integrated management can be approached. The main difference between the perspectives is the scope of what management integration entails. We start with the perspective of the equipment vendor for whom integrated management has the most constrained scope. After that, we proceed to enterprises and service providers for whom the scope of what management integration entails becomes increasingly larger.

The Equipment Vendor Perspective

The question of what to provide for integrated management is typically answered in a fairly straightforward manner by equipment vendors, particularly if they are concerned with shipping only a few types of devices. In general, the vendor's customers expect to be offered what amounts to an element management system, integrating various element management functions for a device, as Figure 10-4 shows. For better or worse, the vendor's management offering is essentially considered an extended equipment feature. (This is actually a two-way street because the equipment vendor's customers generally expect and demand just that. In too many cases, the people who make purchasing decisions are separate from the people who later need to run the network.) Of course, significant differences exist between different customers and market segments, but, in the end, this is what things boil down to in many cases.

Figure 10-4 *Typical Management Integration Scope—Equipment Vendor*



As a result, for the equipment vendor, integrated management means the need to provide an integrated management application that will allow that vendor's devices to be remotely configured, audited, and monitored; their configurations to be backed up and restored; and device images to be upgraded. The embedding of those applications into a larger operational context is generally left to the vendor's customers and to independent software vendors and system integrators.

There are exceptions, of course. Some equipment vendors view themselves as end-to-end solution providers, not point product providers. In that case, they could be treating management as a business in its own right and essentially play the dual role of an equipment vendor and of a systems integrator and application vendor. They realize that the ability to efficiently operate and manage the network is a significant factor in the economic equation for their customers. But for equipment vendors who treat management de facto as an extended feature of the equipment that they sell, this is as far as integrated management goes. And who is to blame them? From their perspective, this makes perfect business sense and allows them to focus on their core business and competence.

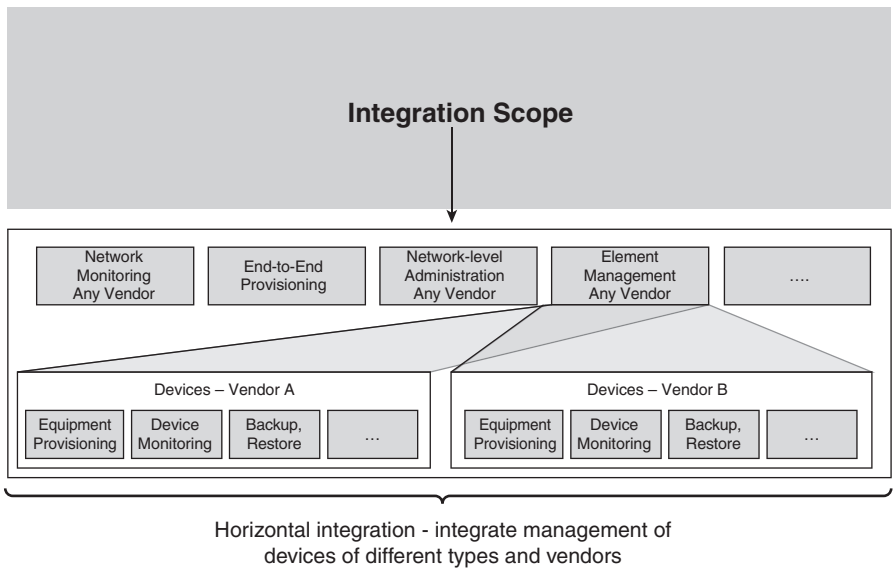
Of course, even with a relatively limited integration scope, equipment vendors do need to realize that in some scenarios, their management offering needs to be embedded into a larger operational context that might require integration with other systems. We get to these scenarios in the next subsections. An important aspect, often at least as important as the functionality of the management applications themselves, is therefore that the equipment vendor provides open and well-documented interfaces as part of the management applications that will facilitate such an integration to take place.

The Enterprise Perspective

The scope of integrated management becomes larger in the case of a small to medium-size enterprise that needs to manage a network that includes a wide range of different types of devices from different vendors. In such environments, typically a variety of element management systems (EMSs) are deployed, each addressing the management needs for equipment of different vendors and each used independently of one another.

This might work for a little while, as long as the domain that is to be managed and the number of different types of equipment in the network remain limited and are still easily overseen. However, inevitably, at a certain point, the need for further integration arises. The individual vendor-dependent EMSs need to be either replaced by a vendor-independent system or complemented by systems that integrate certain EMS functions for the network. Figure 10-5 depicts this increased integration scope. For example, it might become necessary to introduce an application to monitor the overall health of the network from a single place, which provides a view of the health of all the devices in the network and of all alarms that are occurring. The alternative to having to keep an eye on different vendor-specific applications becomes simply unacceptable at a certain point. In essence, this would require an operator to sit on a swivel chair to be able to easily switch among multiple terminals that each run a different application. This type of integration—or lack thereof—is generally referred to as "swivel chair integration"; you already encountered this in Chapter 1, "Setting the Stage." Even if the operator were able to access the applications from one terminal that runs multiple clients, it would not improve the situation by much—and not just because of the limited real estate that is available to display information on a computer screen.

Figure 10-5 *Typical Management Integration Scope—Medium-Size Enterprise*



As the enterprise gets larger, the need for further integration arises. For example, requirements such as the need for a central inventory of networking equipment might enter the picture, or the need for an integration of employee databases and network user accounts so that user accounts are automatically established or revoked when employees join or leave the company. In addition, requirements might arise for network management capabilities that go beyond individual network elements but that concern the network as a whole —for example, this could be the capability to manage and monitor end-to-end connectivity between enterprise locations.

The Service Provider Perspective

Management integration takes on a whole new level of complexity for a telecommunications service provider that has to manage literally thousands of different types of systems, devices, and applications supplied by dozens of vendors, used to provide not just one type, but a great variety of different services.

In such an environment, many different tools have to be integrated that are provided by different suppliers and equipment vendors—each of which might be "fully integrated" from their limited perspective. However, how to integrate element management is only one of many considerations because it will be impossible to manage the network one device at a time. Management at the network management layer to manage end-to-end connectivity is no longer an option, but becomes instead indispensable.

In addition, entirely new management functions that did not occur at the level of the enterprise have to be dealt with, each needing to be integrated with the rest of management. Here are some examples, along with a description of specific requirements they impose on integration:

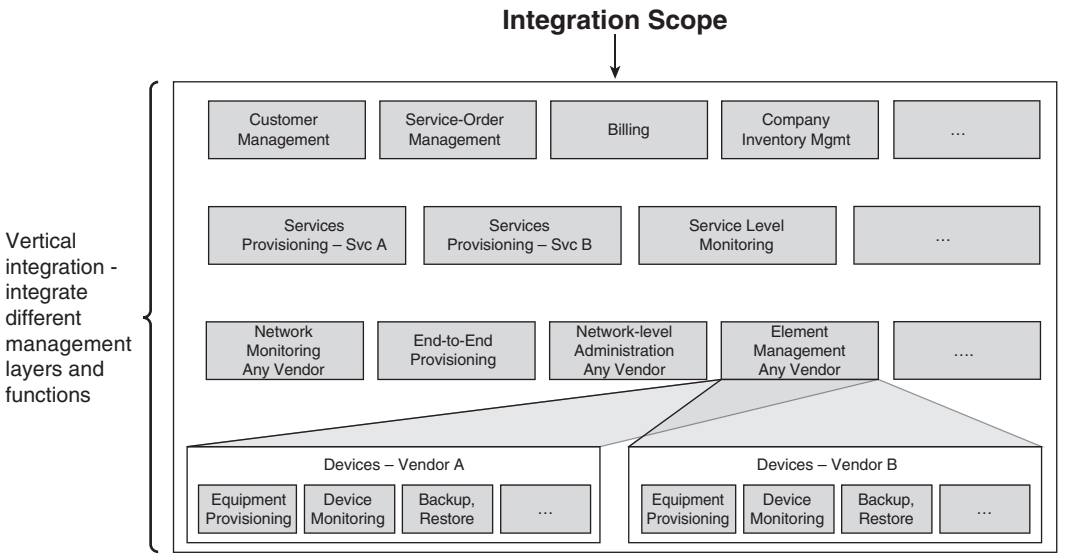
- **Billing for services**—Of course, some IT departments in enterprises might also "bill" departments internally for communication services. However, the significance of billing for a service provider puts this function in the spotlight. Billing typically requires the capability to relate information about service orders, customers, and the network itself (such as information on network addresses and ports) to be able to attribute accounting records to the right customers. Accordingly, billing imposes significant requirements on management integration because much of this information is also related to management functions other than billing and is potentially maintained by different management applications.

- **Monitoring of service level agreements provided to customers**—(Service level agreements are contractual obligations regarding the quality of the service being delivered—more on that in Chapter 11, "Service Level Management: Knowing What You Pay For.") The association of observed network outages with services and end users that will be affected by them needs to be facilitated. This requires integration between applications for the management of services and service level agreements with applications for network monitoring.

- **Management of service orders and integration of management of the network with the management of the service provider's supply chain**—For example, in fulfilling a service order, the service provider must ensure that the necessary network equipment is already deployed or, if it has not, that it is on back order and scheduled to be delivered and deployed by the time the service order becomes due. This requires integration among inventory, service order management, and provisioning systems.
- **Integration of network operations with other business functions**—For example, this might include customer care so that help-desk personnel has access to current network status when responding to customer inquiries.

The list goes on. In summary, service management and business management functionality becomes a critical component of integrated management for service providers. This means that the scope of management integration is no longer just "horizontal," calling for the same management functionality for increasing numbers of managed devices and device types to be integrated. Instead, the need for integration becomes "vertical" as well, having to integrate entirely new categories and layers of management functions, as Figure 10-6 shows.

Figure 10-6 *Typical Management Integration Scope—Service Provider*

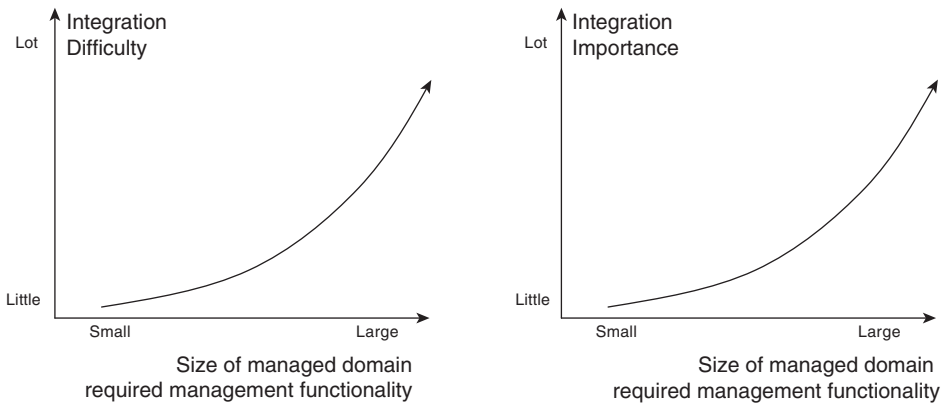


Integration Scope and Complexity

As you have just seen, the scope of what needs to be integrated as part of management depends on the perspective. It can be observed that as the scope of integration grows, its complexity grows in two dimensions simultaneously: horizontally and vertically. Horizontally, the number of devices

and network technologies that need to be integrated increases. However, with growing network size and complexity, it is no longer sufficient to merely integrate and add more systems while still providing essentially the same functionality. Instead, complexity also grows vertically at the same time and makes it necessary to deal with functions of the upper layers in the TMN hierarchy. New phenomena emerge that require new management functionality to be dealt with, phenomena that are not really an issue in operations of a more limited scope. Just as it becomes more important to address management integration, it becomes more difficult to do so, as Figure 10-7 shows.

Figure 10-7 *Correlation Between Integration Importance and Difficulty*



As a network provider, when you are looking for an integrated management solution, you might find that there is nothing available on the market that can do it all. At this point, you need to ask yourself whether you indeed need all the systems and tools in your management infrastructure to be fully integrated, or whether you can afford to break up your management tasks into several distinct areas, each of which can be integrated on its own.

If you are a small to medium-size business, you will most likely rely on commercial off-the-shelf management software to manage your network, perhaps augmented by some home-grown management scripts. However, as an alternative, you might decide to outsource some of your management and buy many of your communication services as a "managed service" from an outside managed service provider instead of providing those services yourself by managing your own network. Of course, any service provided by a service provider is "managed." What is different about "managed services" is that they constitute services that are provided by networking equipment on your own premises; you could choose to entirely manage those services yourself, unlike other services where this choice does not exist, such as perhaps global long-distance telephone calling that does require relying on not just your own but a service provider network.

If you are a large service provider, the option to completely outsource your management goes away. Otherwise, it would imply repositioning yourself to effectively become a reseller, not a provider of communication services. (Having said that, even a service provider typically outsources some aspects of management, such as the processing of billing information, and relies for some service aspects on other service providers, such as for additional international transmission capacity.) However, custom development to achieve management integration becomes a feasible option because you are able to achieve economies of scale that a small network provider cannot attain. You might not do all the custom development yourself, of course, but you can contract a systems integrator for that purpose.

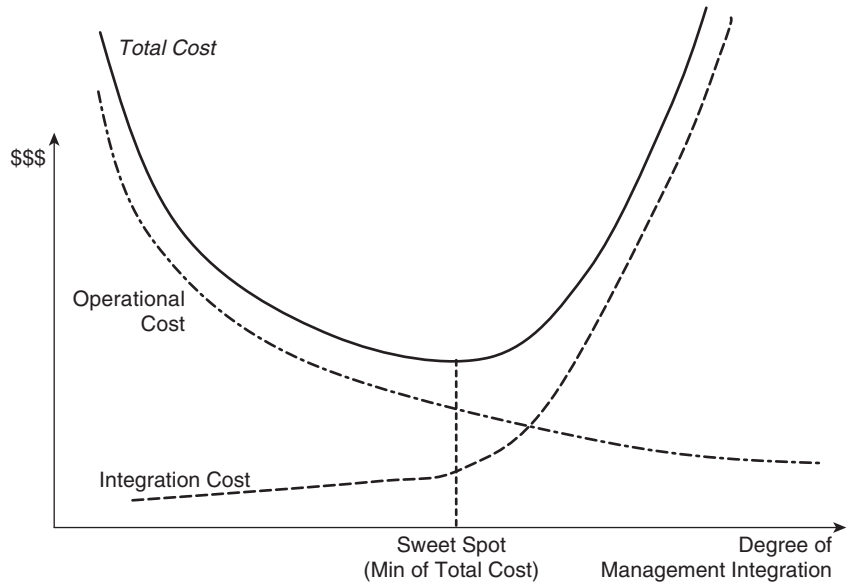
Even so, attaining complete integration of all management functionality for your entire network still presents a formidable challenge, and an expensive one. If instead you decide to break up your overall management task into a number of more loosely coupled pieces and focus on integrating each one of them individually, you can better attain a solution and significantly reduce the total cost. In effect, instead of multiplying complexities of integrating individual subtasks with each other, you are merely adding them up and can deal with one problem at a time. (See also the section entitled ”Quantifying Management Integration Complexity.”) This leaves you with a significantly lower price tag overall than trying to develop the all-integrated solution.

Of course, you need to tread carefully either way: If you break up your management task too much, you are left with a scattering of different tools and a fragmented, piecemeal approach to network management in which things are run inefficiently, important tasks fall through the cracks, and, in the worst case, the functioning of your communications infrastructure is severely compromised, putting your business at risk. This is the very situation that you need to avoid. The challenge accordingly lies in finding the “sweet spot.” That would be the point at which the sum of the cost of providing an integrated management infrastructure and the cost of operating the network using that infrastructure is at its minimum, as Figure 10-8 shows. Note in the figure that the sweet spot does not have to coincide with the point where operational cost and integration cost are equal.

Management Integration Challenges

Several factors make management integration a challenge. They have to do with the different dimensions along which integration occurs: On one hand, management functions need to be integrated across the managed domain—the different devices, networking technologies, and services that need to be managed and to which the same management function applies. On the other hand, different management functions have to be integrated as well. This leads to interesting challenges from a software-engineering perspective.

Figure 10-8 *Sweet Spot of Management Integration Benefit and Cost*



In the subsections that follow, both integration dimensions are discussed, in an attempt to capture what causes the complexity that makes management integration a challenge. We also discuss how management integration complexity can be characterized and quantified according to a set of equations. Understanding the factors in those equations and how they can be influenced is an important prerequisite for making smart choices that simplify management integration.

Managed Domain

As mentioned earlier, the challenge of providing integrated management is compounded by the fact that, just as the importance of integrated management grows, the complexity of what needs to be managed—that is, of the managed network or the "managed domain"—is also increasing. For one, this phenomenon is related to scale—that is, to the number of network devices. More important, it is related to their heterogeneity.

Heterogeneity refers to the fact that different types of devices need to be managed—different device models, devices of different vendors, devices for different networking technologies, and so on. Compare this to homogeneous environments in which all devices to be managed are alike, which obviously simplifies the management task. Increasing heterogeneity implies that network management needs to be capable of dealing with a greater variety of management interfaces. The same device and network features might have to be managed in different ways. For example, one device might offer Simple Network Management Protocol (SNMP) support and a special SNMP MIB to manage certain features, whereas a second device might also support SNMP but provide

a different MIB for the same purpose; a third might offer only a set of command-line interface (CLI) commands.

Also, in general, different types of devices differ in the features that they offer, even if the devices play a similar role in the network. Of course, different features each impose their own requirements for management. For example, one device might offer a feature that allows a manager to configure a policy for connection admission control. The feature allows network managers to specify how many connections may be in progress until the device should deny additional connection requests. This avoids connections cannibalizing each other in their competition for resources, such as processing power or bandwidth. Another device might offer no such feature. Integrated management infrastructure needs to accommodate management for all the different feature variations that could be encountered.

As a second example, even when two devices offer the same networking feature, they might need to be managed differently. For example, the level of granularity at which the feature needs to be managed could differ. Consider the case of a network device that supports voice services in the role of a so-called voice gateway. One parameter that can often be configured concerns which voice codec should be applied. (The voice codec determines how the audio is encoded as binary data when sent over the network; different codecs make different trade-offs between required bandwidth and sound quality.) Whereas one device might allow the codec to be configured on a per-port basis, another device might allow it to be configured only for the entire device or for a group of ports instead of each individual port. Again, all these differences need to be accounted for by a solution for integrated management.

Another aspect of heterogeneity concerns the number of different services that are provided over the network and need to be managed. In the past, a network was often dedicated to a single service, such as phone service or Internet dial-up service. Today networks are increasingly converged, which means they support a wide variety of services that all use the same network infrastructure. For example, the same network might support data services and telephone (Voice over IP) service.

On one hand, this is excellent news as far as network management is concerned: Where formerly several networks needed to be managed, now there is only one. In fact, reducing management cost is one of the main motivations for network convergence. However, the fact that multiple services need to be supported simultaneously can lead to some management challenges of its own. Converging of services also requires a certain degree of integration in the management of those services, whereas previously they could be managed completely independently of one another. The reason for this is that, in a converged network, different services compete for the same networking resources—something that management needs to address properly.

Consider the example of two services that both want to use the same port. Managing both services separately in a nonintegrated manner using separate management applications could lead to the same port first being assigned to one service by one service management application, then

reassigned to another service by another service management application, breaking the other service. Both applications keep track of their use of networking resources but are unaware that they could conflict with one another because each is competing for resources with another service—this aspect was not an issue in a nonconverged network.

A second example involves competition for Digital Signal Processing (DSP) resources. Imagine a device with DSP resources that are limited so that they allow the device to provide one service with traffic encryption and another service with data compression, but not both at the same time. Again, if both services are managed independently and are unaware of one another, conflicts between them can arise.

Software Architecture

As mentioned earlier, business realities often dictate that network providers deal with different management systems and tools that they then try to fit under one umbrella. However, let us briefly assume that we could start with a clean slate. Could we simply build an integrated management application that does it all? Intuitively, this would be the first approach to take. Let us pursue this possibility for a moment to see what it would entail and what difficulties we might encounter. This will give us insights about the challenges that need to be dealt with from a software-engineering perspective. There are other and more effective approaches to management integration; nevertheless, the lessons learned here should prove helpful.

Challenges from Application Requirements

For starters, our integrated management system needs to be capable of providing all management functionality that is required. Of course, as we have seen in earlier chapters, so many functions are involved in managing a network that being able to provide them all through a single system seems unlikely. Even if it were possible to do so for one particular environment, different network providers have different needs, making the task of building something where one size fits all (and that would be customizable enough to make it work in different settings) daunting indeed.

Our integrated management system also needs to be highly scalable, to support very large network deployments. After all, if multiple separate systems had to be deployed because of lack of scale, with each system managing a different part of the network, the resulting management infrastructure could hardly be called “integrated.” Of course, at a certain point it might be necessary to add processing horsepower, perhaps in the form of additional server hosts, but this should not affect the users of the management system, the network operators; in fact, it should be transparent (that is, unnoticeable) to them. However, another aspect of scalability is easily overlooked: Scalability also applies to the other direction. Management systems often need to be capable of also supporting very *small* network deployments in a way that is not cost prohibitive. This way, as the network grows, the need to change or migrate operational support infrastructure in midstream can be avoided.

Then there are the obvious challenges from needing to deal with the complexity of the network itself that we just discussed. This implies that a management system's software architecture needs to be capable of supporting myriad different underlying entities that need to be managed, each with its own unique characteristics and variations in management interfaces.

Hence, the integrated management system must be very easy to extend and maintain regarding the systems that it manages. Importantly, it should be capable of decoupling the release cycle of the application software itself from the release cycle of the entities that it manages—that is, the managed devices. In other words, every time a new type of equipment is added to the network and needs to be supported by the management system, a new management software release should not be required.

Of course, this type of requirement is well known in other areas, such as computer operating systems. Consider, for example, the need to add support for new printers. This is generally accomplished through printer drivers that contain instructions on how to translate between a generic application programming interface (API) that applications use to print and a device-specific command set. In general, new drivers can be dynamically added without requiring a new release of the operating system, or even a restart. Our integrated management system should support an analogous capability. Of course, a management driver in general turns out to be significantly more complex than a printer driver.

Challenges from Conflicting Software Architecture Goals

Another difficulty arises from the fact that different management functions can impose conflicting requirements on the software architecture that the integrated management system is to be based on. At times, those requirements could be difficult, if not impossible, to reconcile in a single system. This point is a little more subtle than the previous ones yet is just as important, so let us elaborate further on it.

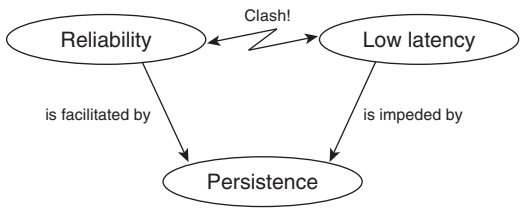
In Chapter 1, we mentioned that one of the challenges in building management applications lies in the wide range of architectural properties that they require. For example, for a provisioning application, reliability is of utmost importance. The application needs to make sure that provisioning commands indeed had the desired effect and must go to extra lengths to ensure that. After all, if services do not work as anticipated, customer satisfaction will plummet, revenue will be lost, and additional cost will be incurred to diagnose and fix things. At the same time, how quickly the management transaction executes is of lesser concern—a customer generally does not perceive a difference of a few minutes in the time that it takes to turn up a new phone service to be critical.

On the other hand, for an alarm monitoring application tasked with monitoring the network for alarms, the biggest concern is to minimize the amount of time that passes between the instant a problem occurs and the instant the network manager is alerted. Time is money, and in some cases,

network outages can cost thousands of dollars every second. Consider, for example, the consequences when a flight-booking system or a network that connects bank tellers goes down. Information that is related to the alarm, such as which users might be affected, has to be very quickly identified, and state changes need to be quickly propagated through the system. Delayed delivery of alarms may be unacceptable because the state of the network changes rapidly, and receiving updates that are late can be almost as bad as receiving no updates at all.

None of these applications is trivial to build, per se. Try to combine all of them into one integrated system, and you are faced with a problem that can be quite difficult to solve. Application properties such as minimized latency and optimized reliability can have different implications on the underlying software architecture and need to be addressed in different ways. At a minimum, they differ on where architectural emphasis should be placed. Excellent judgment is required on how to make the necessary architectural trade-offs to arrive at a system that is well balanced overall. The example of a provisioning application and of an alarm monitoring application that we just encountered should illustrate this point, which Figure 10-9 also depicts.

Figure 10-9 *Conflicting Architecture Goals*



In the case of the alarm monitoring application, special consideration needs to be given to rapid state propagation and event delivery, which might have to be traded off with reliability. For example, for the sake of performance, it might be undesirable to persist into a database every event or equipment state change that is observed because this could slow things down. On rare occasions it is of course possible that an application module fails and some transient information about the network device is lost. However, with some effort, it is generally still possible to recover all important information through special exception-handling procedures, such as by auditing the device for its current state. Although this is not ideal, it is still much more acceptable than the alternative of more sluggish application responsiveness.

The provisioning application, on the other hand, could well make the trade-off the other way: Precedence needs to be given to reliability and persisting what is being done every step along the way. This ensures that services are reliably turned up, even in the case of failures along the way. For example, it might be unacceptable to accidentally issue the same provisioning command for a second time, which might accidentally occur in a scenario in which an application has to be unexpectedly restarted and it is not clear how far a sequence of previous provisioning steps had progressed—an ill-defined network state might otherwise result. Taking the necessary precautions

and logging things into a database every step along the way takes priority over trying to send commands to network elements and processing their responses in the fastest possible manner.

Hence, important calls have to be made for the software architecture of our integrated management system. For example, how often should management information be persisted in the database—every time a change occurs, for maximum robustness, or only periodically, for improved performance? Should state changes be propagated to interested applications using callback functions in synchronous fashion, interrupting them in what they are doing, or is it better to follow a more asynchronous model in which applications are expected to actively retrieve them?

Eierlegende Wollmilchsaun and One-Size-Fits-All Management Systems

To summarize the gist of the discussion to this point, trying to address the needs of multiple management applications in a single system inevitably leads to situations in which the best that can be accomplished might be a compromise—hopefully acceptable, but not the best that could be achieved for each function individually.

For all practical purposes, the likelihood of being able to build a truly comprehensive integrated application that fits everybody's needs is, well, slim. The German language has a term for it: *eierlegende Wollmilchsaun,* referring to an egg-laying, wool-giving, milk-giving pig—a farmer's dream, except for the fact that, unfortunately, it doesn't exist.

So what should be done? In the case of the eggwoolmilkpig, farmers generally revert to the second-most-desirable option: They get some chickens, a sheep, a cow, and a pig instead. Of course, this conglomeration of animals takes a little more effort to tend and uses up a little more space than a single animal. However, at the end of the day, they accomplish the same purpose.

In the case of management systems, why not build multiple applications that each serve a particular purpose and simply make sure that they can work well together? Or if applications have already been built, just use existing applications and perhaps provide some "glue" and additional auxiliary functions that allow them to work together. This is indeed what needs to be done, and we investigate corresponding approaches in the next section. But before we do, let us summarize what we have discussed so far and get a handle on how we can express the complexity of management integration a little more formally.

Quantifying Management Integration Complexity

One could state in a simplified manner that the inherent complexity of providing integrated management for a given managed domain is the product of several factors. This can be expressed as follows:

*Management integration complexity = scale complexity * heterogeneity complexity * function complexity*

We take a closer look at each of those factors in the following subsections.

Note that the factors that contribute to management integration complexity are multiplied by one another, not merely added. For them to be added, the factors that contribute to the overall complexity would have to be independent of one another. This would be the case only if it were sufficient for the management infrastructure to consist of a number of systems that simply exist side by side. In that case, the complexity of providing the management infrastructure would be much lower—it would be merely the sum of its parts. On the other hand, to truly integrate the parts, a multitude of interdependencies need to be addressed, the complexity of which is more appropriately expressed as a product, not a sum.

Scale Complexity

Scale complexity results from the size of the domain that needs to be managed—in other words, the sheer number of devices in the network. The management task is a lot simpler for a small business that has a handful of equipment to manage than for a telecommunications service provider with hundreds of devices in the core of the network and tens of thousands of access devices. Scale does matter. Nevertheless, it is perhaps the least of the factors that determine overall management complexity. Scale complexity could be simply defined as a function of the order of magnitude of the number of managed pieces of equipment or, to account for the different scale and complexity of the equipment itself, of the number of ports in the network that needs to be managed. This is expressed in the following equation:

Scale complexity = f(#ports, #devices)

Heterogeneity Complexity

Heterogeneity complexity defines the degree of uniformity, or lack thereof, of what is to be managed in an integrated fashion. Clearly, if only a single type of router that is deployed across the network needs to be managed, the task is a little simpler than in the case of 10 different equipment types from five different vendors. Accordingly, the heterogeneity complexity of a network can be roughly defined as the product of several factors that are functions of the following:

- The number of vendors whose equipment is deployed in the network
- The number of networking technologies that are converged in the same network (for example, DSL, cable, and fixed wireless for a set of access networks, or MPLS and ATM for a backbone network)

- The average number of different types of devices that are used for each networking technology
- The average number of different versions of each device type (for example, different software images reflecting different revisions in the equipment's operating system)

Expressed as an equation, this yields the following:

*Heterogeneity complexity = f(#vendors) * f(#technologies) * f(#device types) * f(avg # of device versions)*

The factors that contribute to heterogeneity complexity are once again multiplied, not added, for reasons analogous to those that were outlined for management integration complexity as a whole.

Function Complexity

Function complexity results from the variety of different management functions that are to be integrated. Two factors determine function complexity: the number of distinct functions that need to be integrated, and the depth with which you want integration to occur. This is characterized in the following equation—again, a product, not a sum, for analogous reasons as for the previous equations:

*Function complexity = f(#management functions) * f(integration depth)*

The fact that the number of functions that are to be integrated is a factor in determining integration complexity should intuitively be clear. A system that integrates functionality from multiple applications tends to be significantly more complex than the sum of the complexities of each application being realized in its own respective system. The number of management layers that need to be addressed in integrated fashion can also be factored in here. Addressing application management, service management, and network management each through their own respective system is significantly simpler than trying to have them all seamlessly integrated.

The depth of integration is a little harder to capture. It has to do with the extent to which functions are really integrated. It is possible for integration between management functions to be very shallow, reflecting the fact that they are internally realized through separate and largely independent applications. For example, two management functions that are only shallowly integrated might share their user administration and be launchable from the same screen but have little else in common. The look and feel of their graphical user interfaces (GUIs) might still differ, data and internal databases might not be shared, interfaces that are exposed to other applications might be entirely disjointed, and access to managed devices might not be coordinated. Deep integration, on the other hand, might make it virtually impossible for users to distinguish where one application ends and another begins, or even whether multiple applications are involved at all. Obviously, deep integration is much harder to achieve than shallow integration.

Approaches to Management Integration

The integration problem is generally too large to be tackled all at once. Often there is a trade-off to make: Does management infrastructure have to be integrated all the way, thereby making management more efficient but more expensive? Or is it acceptable not to integrate certain aspects, to lower management integration complexity and cost, at the expense of operations that might be a little less efficient as a consequence? If a decision is made to settle for less than full integration, the decision of where to make the cut should be carefully evaluated. The cut should be made in a place where a high reduction in management integration complexity and cost results, yet operations efficiency is minimally impacted.

This needs to be approached in systemic fashion, one aspect at a time. At this point, it helps to remember that there are different dimensions in network management, which we discussed earlier in Chapter 4, "The Dimensions of Management." One possibility for integration is to take a look at these dimensions and decide which dimension is the most crucial for integration. This provides the starting point from which to tackle integration.

For example, in one particular case, it might be decided that integration is most important along the lines of management functions (integrated alarm management for the entire network, for example). At the same time, integration of different functions with each other (alarm management and inventory management, for example) might be of lesser importance. This integration approach could be a good fit in case of a network provider organization that is primarily grouped along functional responsibilities instead of different network technologies, for example. This way, the management infrastructure is integrated for the function that each group is primarily interested in. In the example, the fact that other management functions are not as well integrated is deemed less critical because an individual group will not be concerned with those other management functions and, hence, will not be affected as much by a lack of integration. (Having said this, some coordination between the groups may still have to occur and should be supported by the management infrastructure.)

Integration might also start with any of the other management dimensions, for example:

- The network technology being managed—for example, addressing management integration separately for the core router network, for the DSL access network, and for the mobile wireless network
- The layer of the network being managed, separating management of the physical transmission network from the switching and routing networks
- The management layer, integrating service management on one side and element management on the other

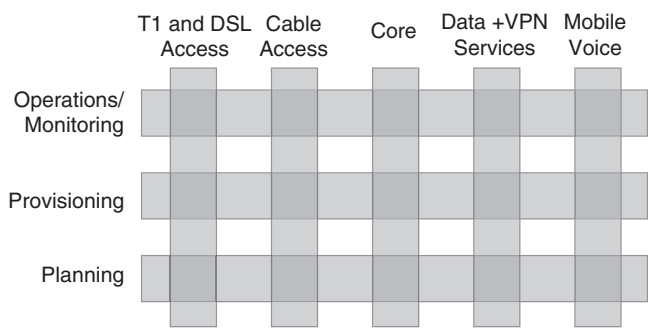
The choice of which dimension to use as you begin integration is significant for management integration. When you are finished integrating along the first dimension, you can start taking on the second dimension that reflects your second priority. You might never get to the last dimension and your least priority, but you will end up with management that is integrated to a degree that should suit you well.

Adapting Integration Approach and Network Provider Organization

One aspect that will likely influence a network provider's approach to management integration is how its operations organization is set up. The larger the network that needs to be managed, the more important it becomes how the organization that is responsible for managing the network is structured. After all, the management organization itself is an important part of how networks are managed; hence, integrated management does not stop at the technical infrastructure but needs to take the organizational dimension into account as well. In addition, because the management infrastructure is supposed to support the organization in the most effective manner possible, the way the organization is set up profoundly impacts the integration that needs to occur for the systems and applications that make up the management infrastructure.

The questions that need to be answered when setting up a management organization to run a network are very similar to the questions that concern technical issues. For example, should management be organized primarily along function, with one group responsible for equipment deployment and provisioning, another for network planning, and a third for monitoring and maintaining the network? Or would an alternative organization be preferable that revolves around network technologies, with one group responsible for the time-division multiplexing (TDM) voice network, a second for access networks, and a third for the IP and Multiprotocol Label Switching (MPLS) core? Figure 10-10 depicts a matrix of functions that have to be supported for a fictitious network provider organization.

Figure 10-10 *A Roles and Responsibilities Matrix for a Fictitious Network Provider*

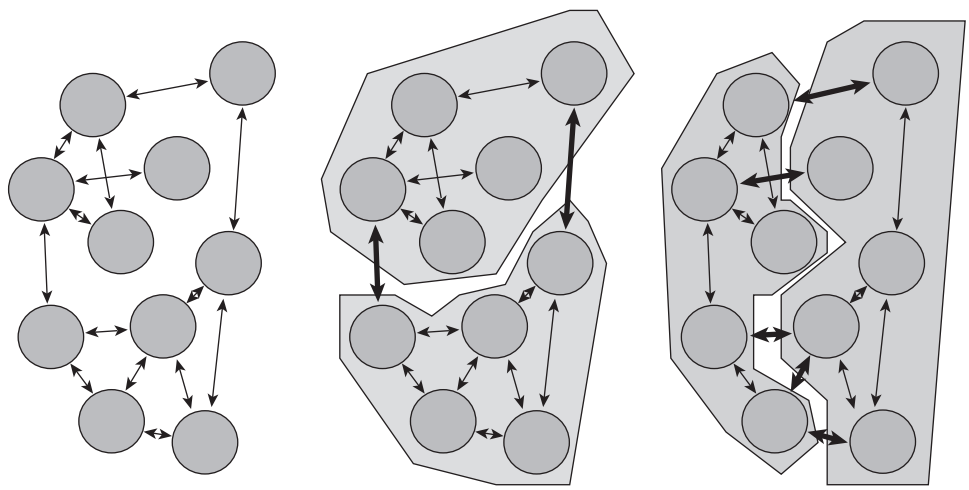


Different network providers might answer these questions differently. Regardless of what particular organization results, a couple rules should be observed:

- It needs to be clear who is responsible for what. Accountability must be as clear as possible; otherwise, the ability to react appropriately during times of crises will be jeopardized. Situations must be avoided in which different groups point fingers at each other, each claiming that it is not responsible for problems that have arisen.
- The number of interactions and interdependencies between groups should be minimized. When trying to divide the overall task of running a network, the division that results in the greatest independence among the resulting pieces generally is preferable. The fewer interdependencies exist, the more robust the overall organization will be against failures in particular functions, and the more efficient it will turn out to be.

  Figure 10-11 depicts this principle in a simplified manner as a graph optimization problem: Assume a network provider organization with different functions and interdependencies, depicted as a graph on the left side of the diagram. The organization decides to split network operations across two groups, each of which is to be responsible for roughly the same number of functions. Two possibilities to organize are depicted as organizational cuts A and B, respectively. Cut A is clearly preferable over cut B because it involves fewer interdependencies.

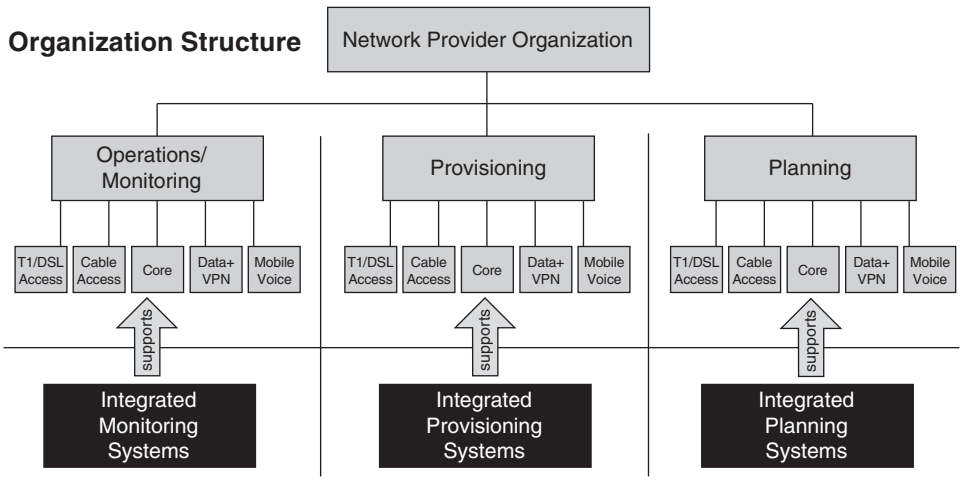
Figure 10-11 *Approaching Reduction of Organizational Interdependencies from Graph Theory*



- Where dependencies between different groups exist, processes need to be clearly defined that define the "rules of engagement" between the groups. Those processes need to be accompanied by clear communication paths. As in the case of establishing who is responsible for what, this increases accountability and reduces the chance of finger pointing.

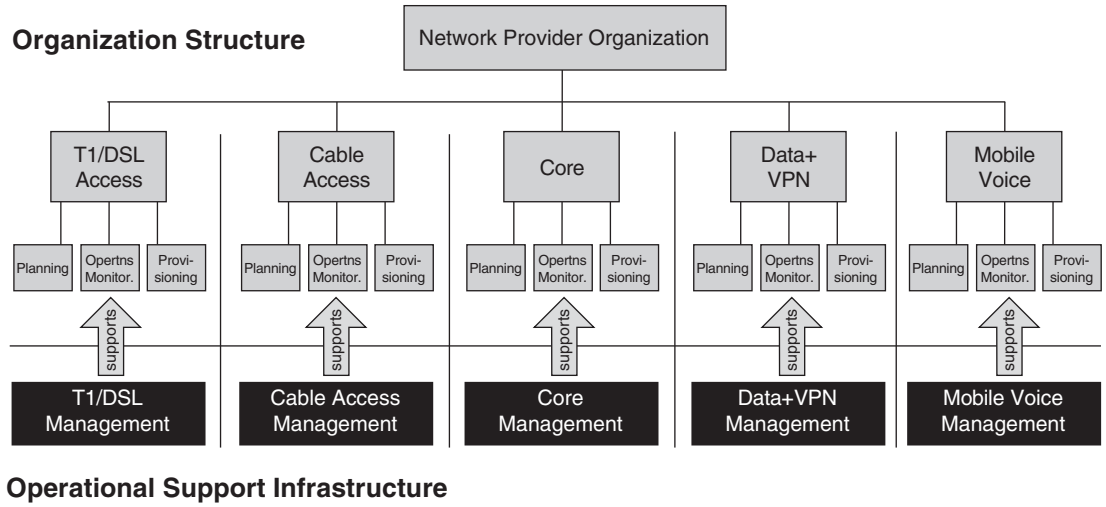
Regardless of the particular way in which the network provider organization is structured, integration of the technical management infrastructure should go hand in hand with it. The management infrastructure needs to support the network provider organization in the best way possible and, to a certain degree, reflects its structure. To this end, Figures 10-12 and 10-13 depict two alternative organizations to support the roles and responsibilities matrix from Figure 10-10; one is organized along support function, the other along supported services and technology. Note in each case the differences in terms of operational support infrastructure.

Figure 10-12 *Organization and Operations Support Infrastructure Along Functional Lines*



For example, if different groups are responsible for equipment provisioning and for monitoring, very little can be gained from trying to integrate provisioning and monitoring systems. However, much can be gained from trying to integrate the provisioning applications for equipment from different vendors, and alarm management applications from different vendors. By the same token, if different groups are responsible for the core and access portions of the network, management systems do not have to integrate management of the entire network, which also relaxes some scaling requirements. In that case, more is to be gained by the integration of different management functions, limited to just this particular portion of the network. Likewise, the interfaces and dependencies among the different systems of the operational support infrastructure tend to reflect the interfaces and dependencies between the organizations that own them.

Figure 10-13 *Organization and Operations Support Infrastructure Along Technology and Service Lines*



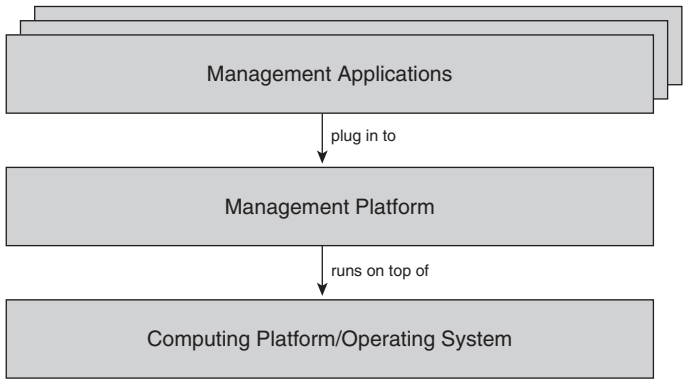
Platform Approach

A common approach to management integration is to integrate management applications using a *management platform*. A management platform is a software system that provides common infrastructure services for management applications, along with a base set of management functions that are typically intended for monitoring a network. Management platforms are often accompanied by software development kits that facilitate the development of additional functionality. Additional management applications are available, sometimes offered by independent third parties, that "plug into" and integrate with the management platform. There are several popular management platforms and accompanying product suites on the market today.

At its core, a management platform constitutes a piece of management middleware that resides on top of a computer operating system and below management applications that build on top it, as shown in Figure 10-14. The management platform provides the "integration glue" for the management applications. In this sense, it can be thought of as a "management system operating system." The idea is that every management application and every management tool that the network provider needs should be capable of running on the same management platform. The management platform, along with the applications that run on top of it, therefore becomes the only system the network provider will ever need. The management platform effectively becomes a "Jack of all trades" that is integrated by design—integrated at least to a similar degree as applications running on a desktop PC, which use the same GUI widgets, the same file system, the same device drivers, the same internal communication mechanisms. By adopting the principles of an operating system

that allows multiple applications to be provided and run semi-independently on top of it, the platform approach avoids the pitfalls of a monolithic one-size-fits-all management system.

Figure 10-14 *Management Platform Concept*



For readers who are interested in the software-engineering aspects of management platforms, the following subsection dives briefly into the common management application infrastructure that a management platform typically provides. Subsequently, we discuss application functionality that a management platform typically provides out of the box.

Common Platform Infrastructure

Like a computer operating system, a management platform provides a set of infrastructure that management applications commonly need and that developers of applications for the platform consequently do not have to build themselves. This infrastructure goes beyond building blocks that are incorporated into application code in the form of software libraries. Instead, it involves common services that live in their own process space during runtime. Applications that need to use those services interact and exchange information with the platform components that provide those services. Platform infrastructure includes also mechanisms that applications can use to coordinate, communicate, and exchange data with one another—think of Object Linking and Embedding (OLE) in operating systems.

Infrastructure in management platforms typically includes (but is not limited to) the following:

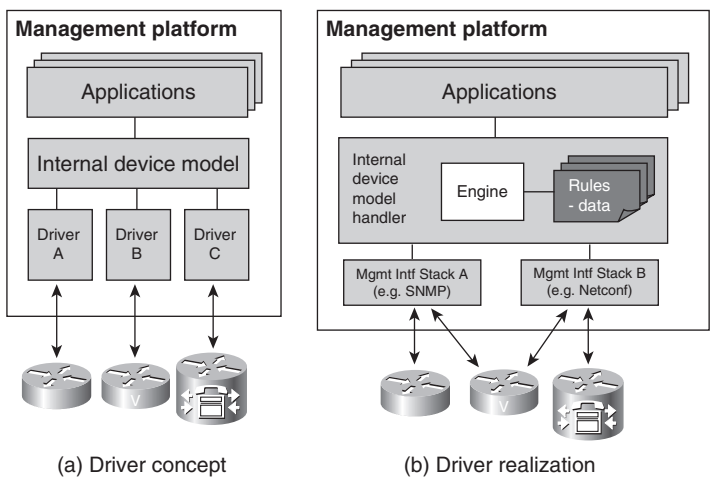
- A common database for storing information.
- Common system-administration facilities, including user administration, error logging, and database backup and restore.

- Device communication services, typically provided through some kinds of device communication handlers that allow applications to exchange requests and responses with the managed devices in a way that is easy to use and program against, and that will encapsulate any required communication protocol stack.
- Internal directory and name-resolution services, allowing applications to (for example) resolve the names of network elements and managed objects to the network addresses and identifiers used to communicate with them.
- Network-discovery services, identifying devices in the managed network domain.
- Management information caching and network inventory, storing and maintaining information about managed devices in the database, from device credentials that are needed to "log on to" a device and maintain a management session with it, to information about the physical and logical configuration of a device as well as the software image and operating system version running on it.
- As an extension of the management information cache, services to keep track of the current alarm state of managed devices, including the highest severity level of any alarm condition that has not cleared, or the reachability state of the device (which indicates whether the device is responsive to management requests).
- Event-distribution and -registration facilities, allowing different application modules that are interested in events from the network to subscribe to them, without requiring managed devices to send events to each application separately.
- Application component registries, allowing application modules to announce themselves to the system and possibly to exchange information with each other.
- A common help system.
- A GUI framework to facilitate development of applications with a common look and feel, possibly including facilities that enable drag-and-drop between different application functions.
- Application programming interfaces (APIs) and possibly a software development kit (SDK). The APIs and SDKs provide common software building blocks, such as GUI widgets, which allow the management platform to be extended and new applications to be developed for it while leveraging its existing infrastructure.

Also, as in the case of a computer operating system, the infrastructure includes facilities that allow applications to make use of new peripheral devices, which, in this case, means to manage new equipment. It is generally a requirement to be capable of adding support for a new piece of equipment dynamically, without requiring a new management platform software release (and, in

some cases, without even needing to stop and restart the platform). Typically, this is achieved by management software that is data driven. This means that device-specific application logic is not hard-coded into the algorithms of the management platform. Instead, where device specifics come into play, the management platform is provided with a set of device-specific driver data in a separate file or data store that can be loaded during runtime. The driver data includes information about how to communicate with the device—for example, what management interfaces and MIBs it supports, the syntax of commonly used CLI commands, perhaps even a graphical icon that tells the platform how to represent the device on a GUI. The management platform contains software that can interpret the data as needed to derive what it needs to do to perform its functions—for example, how to communicate with the device and audit it, how to interpret event messages sent from the device, and how to render a graphical depiction of the device on a screen. With operating systems, such capabilities are known as device drivers or plug-and-play. Figure 10-15 depicts the analogous concept for a management platform.

Figure 10-15 *Device Driver Concept*



(a) Driver concept (b) Driver realization

In Figure 10-15, part (a) depicts device drivers as a concept—applications can operate on an internal model of the device, with device specifics mapped to this model through means of drivers. (Note that in the context of management applications and platforms, other terms are generally used, such as device adapters or model handlers—the underlying idea, however, is the same.) Part (b) of the diagram depicts how such drivers might be realized. The component realizing the internal model contains a software engine that interprets driver data that contains rules for how to map the internal device model onto operations on the device. Actual communication with the devices occurs through a set of management protocol stacks such as SNMP, Netconf, or CLI over which the operations are carried as specified in the driver rules.

There are some inherent limitations that govern how far such capabilities go in management platforms. Support provided by the platform generally extends to providing basic discovery of the device—for example, understanding how to audit its basic hardware and networking configuration, as well as monitoring its alarm state. It might also allow users to browse the device's management information. The platform might not be capable of understanding what the information means, but it is still a useful capability because an end user can make sense of it. However, more comprehensive management support requires capabilities beyond that: For example, new devices might have new features with very specific management requirements that require additional management application logic and very specific knowledge about the device. Also, services might have to be mapped in a very specific way to the new device's configuration, which again requires a deeper understanding of the device.

Typical Platform Application Functionality

Like most computer operating systems, a management platform comes with a set of applications that are built in and ready for use by network managers. As with computer operating systems, the boundary between which applications are part of the platform itself and which are separate applications that build on top of it can be fleeting. As mentioned, separate applications can be offered by the same vendor that supplies the management platform, or they can be offered by independent third parties.

Examples of base applications that come with a management platform include the following—as before, this is a noninclusive list, and specifics vary with each product:

- Network topology views, along with capabilities to underlay the view with maps and to animate icons to indicate the state of managed devices, to provide network managers with an overview of the network
- MIB browsers and CLI terminals, enabling network managers to interact with devices and explore their management information
- Alarm viewers, enabling network managers to view the current list of alarms in the network
- Basic alarm correlation and filtering functions

These basic capabilities can be accompanied by a wide array of additional functions that cover a vast spectrum, from network-performance analyzers to functions that allow distribution of new software images to managed devices, from expert systems for network diagnostics to workflow engines, from special-purpose applications for the management of storage-area networks to special-purpose applications that assist with the planning of physical wireless access point topologies.

Custom Integration Approach

Another approach to management integration is to avoid relying on a management platform, but to instead custom-integrate multiple systems and applications that are otherwise independent. The resulting operations support infrastructure is often also called a "solution." (One might wonder, a solution for what? The answer to this question is, of course, a solution for the problem of how to provide an integrated management infrastructure.) Hence, a solution consists of multiple management systems and applications that are integrated to work together and collectively form the operations support infrastructure that is used to manage the network.

The following subsections discuss the challenges that are associated with this approach, and considerations and techniques that can be used to address those challenges.

Solution Philosophy and Challenges

Contrary to applications that run on a management platform and rely on the management platform's infrastructure, each of the systems and applications that is integrated in a solution incorporates sufficient infrastructure to be capable of running on its own. They constitute a set of loosely coupled solution components, which coordinate with each other as required to exchange data and provide an integrated management experience. Where a management platform provides off-the-shelf integration, custom integration is pursued in the case of a solution.

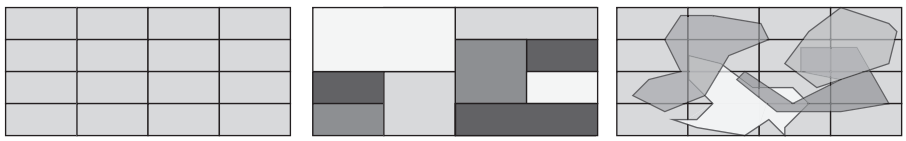
The promise of the custom integration approach is that it might better fit the particular needs of an operations support organization than would be possible with a one-size-fits-all management platform. Importantly, it allows the use of best-of-breed components for different management functions without restricting the users to those components that happen to be available on a particular management platform. Where a management platform might resemble a shark, a solution resembles a school of tuna that can take on the silhouette of a shark when needed but that is more nimble and flexible otherwise.

Ideally, the components in a solution should fit together and complement each other like Lego blocks. For this to occur, each component must have a well-defined scope of functionality. A component generally also needs to offer a northbound interface, allowing other applications and components on top of it to "flow through" operations to the network. (The term *northbound* interface originates from the fact that the interface is generally driven by applications that reside at a higher layer in a management hierarchy and that in diagrams are therefore drawn on top [or, if the diagrams were a map, to the north] of the application that provides the interface.) Likewise, a solution component can interface southbound with components of lower management layers and can collaborate with other systems and applications to the east and west as needed.

The trouble is that, in reality, different applications that need to be converted into components of a solution rarely fit so easily. Multiple issues must be overcome—for example:

- Most of the time, components are not metaphorically shaped like Lego blocks, but they resemble shapes that are closer to potatoes or horse radishes that, while wholesome, are hard to fit together. Figure 10-16 depicts this situation. The fact that collectively they might have functional gaps that need to be filled is only the most obvious of the issues. Just as important, the functionality of different components might overlap, which can also be a problem: In a management solution, just as in an organizational structure, it is important to have clear responsibilities and accountability. If the same function can be performed from multiple places, certain complications could arise: It requires synchronization between the parties involved, the same tasks may have to be tracked across different components, and in case things go wrong, the responsible component might be hard to identify.

Figure 10-16 *Custom Integration—Theory and Practice*



(a) Management territory to cover

(b) Theory: Lego block integration

(c) Practice: Amorphous shapes

In addition to issues of functional coverage, components might not fit together easily. In general, even if components offer comprehensive interfaces, making them work together requires significant effort up front and is a far cry from plug and play.

- Components are likely to require duplicated infrastructure, which makes solutions harder to deploy. Because most of the applications and systems that are used as components also can function independently of any other particular system, they might require different computer operating systems or database management systems. When this is the case, they need to be deployed on separate hosts, which can significantly increase the footprint of the management infrastructure versus what would otherwise be required. Hosts cost money and take up space. Just as important, this increases the effort that is required to administer and maintain the management infrastructure itself.

- It is harder to create a seamlessly integrated user experience than with a platform approach because, in many cases, components come with their own screens and look-and-feel. This is one reason why custom-integrated, component-based solutions are often not an ideal fit in environments where human users are expected to interact with (multiple) applications. Solutions are better suited for scenarios in which humans do not need to interact with most components of the solution components directly (except, perhaps, under special circumstances), but where instead applications interact with other applications through their

electronic interfaces and APIs. Another good fit for solutions is environments in which organization boundaries match component boundaries—that is, environments in which each distinct group or network manager would be using only one of the applications directly anyway. Integration in such scenarios can focus on the exchange of data between the applications using their electronic interfaces and APIs, without concern for human user experience.

- The solution as a whole is harder to sustain and keep updated. Changes in server host infrastructure impact potentially many applications, unless components are not integrated with regard to their use of infrastructure, which leads to other issues, as described earlier. For example, if an organization wants to upgrade to a new revision of an operating system or a database, all applications need to support the new revision before a migration can occur. Likewise, if new devices or networking services need to be supported, support needs to be potentially added in multiple places simultaneously. This can significantly compromise the nimbleness that is supposed to be achieved through the solution.

Considerations for Top-Down Solution Design

Let us investigate a few aspects that should be considered when designing a solution architecture for an integrated operations support environment that is made from different collaborating components. The first aspect concerns ownership of different pieces of management information and the establishment of which component in the solution is in charge of what function. The second aspect concerns how the solution interacts with the managed devices and the establishment of clear chains of command and escalation chains. The third concerns the use of "umbrella" components such as workflow engines to facilitate collaboration in a top-down fashion.

Who Owns What

We mentioned earlier the importance of establishing clear roles and responsibilities in network management. This is true for groups in a network provider organization as well as for applications and systems that are to interact as part of a holistic operational support infrastructure.

Establishing roles and responsibilities in an integrated management solution entails defining which component owns what piece of data or management information and is responsible for maintaining it. As a general rule, there should be exactly one owner for each piece of data.

It might be worth keeping all data in a common repository so that applications always know where to find the data they need. The repository would be conceptually centralized, although, of course, it might (and, indeed, should) internally be realized in distributed fashion, with appropriate replication and mutual backup mechanisms in place to achieve scaling and reliability objectives. Any component can retrieve data in the repository as needed. However, the responsibility to maintain the data could be distributed across different components. For example, one

element management system could maintain information about network equipment of a particular vendor or equipment type, a service order system could maintain information about services that customers ordered, and a network management system could maintain data about the provisioned connections in the network. Separate repositories also could exist for different types of information, such as a network inventory and a service inventory. In some cases, information in the inventory can be seeded (that is, initially be populated) by human administrators who define what equipment needs to be managed.

Individual systems and applications that are part of a solution can still cache data as needed. However, the location of the authoritative copy of information and who maintains it must never be in doubt; this must be one of the first things to be clarified in any solution architecture. This appears to be self-evident, but it must be mentioned because it truly helps to avoid issues regarding synchronizing replicated information; this can be among the most difficult problems to solve in integration between otherwise independent systems, and it simplifies getting to the root cause of problems if things ever go wrong.

Chains of Command

Another related issue concerns how components in the solution interact with the managed devices in the network. When following a TMN-like hierarchy, upper-layer systems that need to interact with the network, such as to provision a service, instruct lower-layer systems to carry out requests, such as to configure a port on a device, until finally requests reach the network element. The responses are propagated back up accordingly.

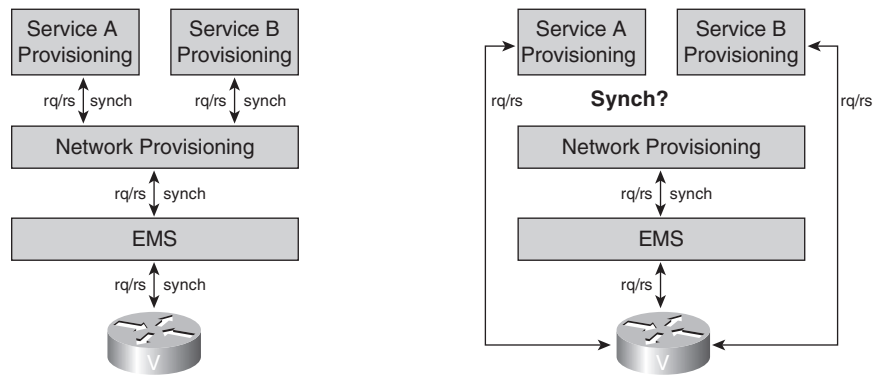
A major advantage of this architecture is that a clear hierarchy makes it easy to keep things in synch. The element management system at the lowest management layer can coordinate the requests that are intended for the device. If the element management system is responsible for keeping the solution's network inventory updated about the device, it will be capable of doing so very easily because all requests lead through it.

A disadvantage of this architecture is that it can lead to inefficiencies because requests have to trickle down multiple layers. This can negatively impact performance. In addition, in many cases, there is no real advantage in having these layers for purposes of information aggregation and abstraction. This can be the case, for example, when a service provisioning system already has a very precise notion of a device-specific parameter that needs to be touched to manage a service, such as the echo cancellation and codec settings for a voice port. From the perspective of the systems on top, the added value that the lower-layer management systems provide (for example, in terms of hiding intricacies of the management interfaces of different types of devices) is at least partly offset by aspects that make the job of provisioning more difficult. For one, more "rainy day" scenarios need to be accounted for—more possibilities exist for things to go wrong because multiple components requiring multiple interactions are involved. In addition, a dependency is introduced on the lower-layer component—let us not forget that this component could be a

full-fledged management system in its own right. For example, if it is late to market and does not support the required device type and interface in time, the service provisioning system will not be capable of performing its job in the meantime.

For these reasons, there will always be a temptation to have components in a solution bypass other components to interact with the network directly, as shown in Figure 10-17. In those cases, care needs to be taken to not violate the earlier principle of needing clear owners for certain responsibilities. If a component is bypassed, it can no longer coordinate access to the underlying systems and network devices, which can lead to resource-contention issues. It could also be harder for the component to maintain the management data that it is responsible for. Of course, in the case of the provisioning example, it can be argued that an element management system might still be capable of maintaining the inventory information of "its" devices even if bypassed by provisioning applications. The reason is that the element management system will have synchronization mechanisms with the network elements in place that will inform it of any changes to the configuration that are made. Still, bypassing components is an issue that should not be done lightly. Careful consideration needs to be given to the benefits of having a clear chain of command versus taking some pragmatic shortcuts that might or might not be necessary. After all, the cost of synchronizing different applications after the fact could offset the presumed efficiency gains.

Figure 10-17 *The Temptation of Bypassing Management Chains of Command*



(a) Clear chain of command, straightforward synchronization, inefficient "waterfall" command propagation

(b) No clear chain of command, efficient command issuance, synchronization between systems lost

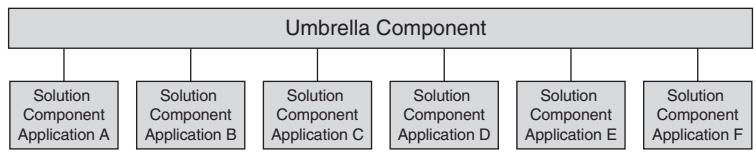
Umbrella Coordination

Finally, if there are many activities that involve interactions among multiple components, introducing an umbrella system should be considered—a system that can act as a central coordinator. This is called an "umbrella" because it acts as a common front end to users, shielding network managers and external systems from needing to individually interface with the various

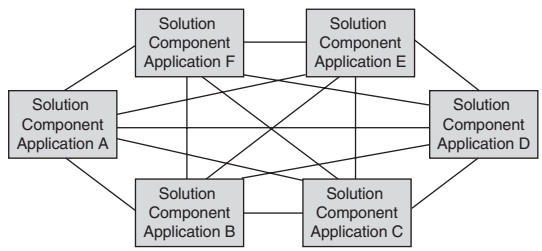
components underneath the umbrella. An example of umbrella systems is a workflow system that takes care of interactions with users or external systems at the front end, while accessing and coordinating different components at the back end, as needed.

The main advantage of having a central coordinator is that it focuses integration activities on one system. It tends to result in simpler solution topologies when compared to solutions in which components are effectively treated as equals. Such solutions involve a larger number of mutual dependencies that resemble a mesh, which consequently also involve a greater number of integration steps. The simplification in integration steps in many cases justifies the cost associated with introducing the additional umbrella component. Figure 10-18 depicts the difference between umbrella and mesh integration.

Figure 10-18 *Integration Topologies—Umbrella vs. Mesh Integration*



(a) Umbrella integration



(b) Mesh integration

Component Integration Levels and Bottom-Up Solution Design

As mentioned earlier, integration between components is not something that is binary—something is either integrated or it is not. Instead, integration occurs at different levels—integration can be shallow or deep. This is good news: It means that it is often possible to start with very simple steps to begin integrating different applications and systems, and successively deepen the integration later.

Integration touches on different technical aspects of the systems to be integrated. Many of these aspects can be approached independently of one another. For example, integration of databases has little effect on the integration of user interfaces (or lack thereof), and vice versa. Here are some

ways in which integration between different systems can be successively deepened along several lines in a "bottom-up" fashion:

- User interfaces are the most obvious area of integration. At the most shallow integration level, two applications' user interfaces can run on the same client, viewable on the same computer monitor. As a next step, each application makes functions of the other application accessible from its own user interface. For example, an alarm management application might enable a user to invoke a device view that an element manager provides. As an initial step, it might simply be possible to invoke the device-viewing function from a drop-down menu in the alarm management application, which invokes a screen from the element management application, from which the user can navigate to the desired device. With increasing integration, the user should be able to invoke the view for a particular device from the displayed alarm information itself, without needing to navigate to the device first. Ultimately, it might get to the point that both applications share the same look and feel. Deep integration is achieved when the user can no longer distinguish which function belongs to which application.

- Databases are a key to deep integration. The databases of applications that are to be integrated need to become successively more aligned with increasing levels of integration. As a first step, different management functions should use the same database management system infrastructure and backup/restore utilities without actually sharing schemata (that define what data is to be stored and how) or the data itself. As integration progresses, applications should start aligning their schemata, up to the point that they can actually start sharing each other's data. Deep integration is achieved when the applications share all data and no longer keep individual copies of data just for their own use.

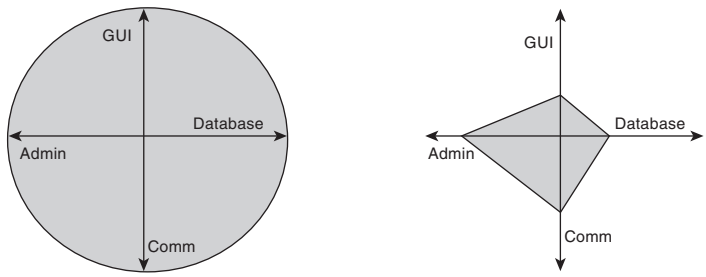
- Communication with the managed devices should become increasingly coordinated between applications as integration deepens. Without integration, each application will communicate and synch up with the managed devices individually. They discover and audit the devices separately and on their own. They may also subscribe to the same events, requiring managed devices to send events redundantly even if the management functions reside on the same host. As integration starts to occur, applications first start converging on using the same communications stack, then successively coordinate their communication needs; for example, they share events that are received from the device and discover each device only once for all the integrated applications. Deep integration is reached when managed devices are never hit twice for the same purpose.

- Other areas affect the depth of integration, including the electronic interfaces that are exposed to other applications, user administration, help functions, and even less technical aspects such as product documentation and installation and upgrade procedures.

Integration can hence be considered an *n*-dimensional integration space, with database, GUI, northbound interfaces, and so on each constituting different and independent integration

dimensions. To characterize the degree of integration, an integration profile can be formed by indicating the degree of integration on an axis for each integration dimension and then connecting the resulting coordinates. Figure 10-19 depicts this. Full integration between different systems can be characterized in terms of a shape that approaches a circle, whereas integration that is less than perfect leads to differently shaped integration profiles.

Figure 10-19 *Integration Dimensions and Integration Profiles*



(a) Management integration dimensions

(b) A management integration "profile"

The Role of Standardization and Information Models

In Chapter 4, we discussed the role of standards in network management. It should be mentioned that standardization also plays a big role in integration among different components. After all, the intent of network management standards is to facilitate interoperability between different systems, which facilitates their integration. Standardization concerns not only interfaces between managing and managed devices, but also interfaces between different management systems. We discussed this in conjunction with the TMN hierarchy, in which you can have systems acting as managers and as agents at different layers of the hierarchy. Organizations such as the TeleManagement Forum (TMF) aim specifically to standardize interfaces between the different systems that are involved in the management of large networks. The fundamental idea is always the same: By building toward a standardized interface, different systems can achieve at least a basic level of interoperability right out of the box, without requiring additional custom development effort. At a minimum, this makes integration easier than it would be otherwise.

Beyond communication protocols and services that are provided across interfaces, a significant consideration for integration concerns management information. If different applications are based on the same abstraction of the underlying managed domain, there is no need to translate between different information models, which is where a good deal of the integration effort lies. Having a common semantic understanding among all involved parties of the managed domain provides a powerful model to achieve deep, "true" integration. Therefore, modeling is often at the heart of integration efforts.

In practice, it turns out to be quite difficult to achieve standardization of management information. The following are some of the obstacles that need to be overcome:

- A model needs to be reasonably rich and complete for its defined scope to be useful. It is difficult to formulate a model in a manner that is at the same time generic enough that it does not break easily when applied to different situations, many of which might not be foreseen at the time the model is first defined, and specific enough that the model is still efficient to use.

  Often it is most pragmatic to settle on a well-defined least common denominator, which compromises between seeking consistency on one hand and keeping the model easy to use and not bogging down applications on the other hand. In other words, it focuses on the 20 percent of the information that is required for 80 percent of interactions between systems that need to be integrated. It is also helpful if a model has a well-defined scope and focuses on a very specific and clearly bounded problem.

- It can be a challenge to maintain standardized models and keep them up-to-date. Managed technology evolves quickly, while standardization traditionally moves slowly.

- To be successful, a standardized information model must be defined in a way that makes it is very easy to understand, use, and implement. If the model becomes too complex, there is a risk of it becoming hard to learn and use. In addition, the information model must provide an obvious benefit not just for integrators who want to use the interface, but for the interface developers as well, to achieve the necessary degree of acceptance.

Of course, as with all things that get standardized, in many cases there are different standards to choose from, which runs somewhat counter to the purpose of standardization. Here, as in networking itself, Metcalfe's law applies, which states that the value of a standard increases with the square of the number of systems that support it.

Containing Complexity of the Managed Domain

Management integration is a key factor in simplifying the management task for network provider organizations. However, management can be simplified in other ways. These involve containing and avoiding management complexity in the first place instead of dealing with it once it occurs. To keep management complexity low, it helps to take a look at the factors that contribute to management complexity, discussed earlier in this chapter.

One major factor is heterogeneity complexity—dealing with the complexity caused by heterogeneity in the network. Incidentally, this factor is, to a certain extent, in the control of the network provider himself. One way to reduce heterogeneity complexity lies in minimizing the number of device types and equipment vendors deployed in the network. Each additional vendor and equipment type means new interfaces, new operational procedures and deployment rules, and possibly additional element management systems.

Of course, a network provider might come to the conclusion that it is still necessary to deploy equipment of multiple vendors and types, sometimes for nontechnical reasons. For example, competition might be a good way to keep suppliers on the edge to offer the best possible deal. However, this advantage needs to be carefully weighed because it can easily be offset by a higher total cost of ownership (which factors into the cost of running the network) that is incurred because of the added heterogeneity complexity. If a network provider uses a variety of equipment models from multiple vendors that perform similar functions, one way to contain complexity is to separate them in the topology of the network as much as possible. For example, if network operations are divided into different regions or subnetworks, it is generally a good idea to deploy equipment of one type and vendor in region A and equipment of a different type and vendor in region B instead of mixing them. This way, there is significant homogeneity at the level of network regions, which might be run by different groups that are then each exposed to only low heterogeneity complexity. At the same time, this retains the advantages of multiple suppliers.

By the same token, it helps to introduce "cookie-cutter" topologies and configurations across the network wherever feasible. For example, the deployment structure of each remote site of an enterprise should follow exactly the same pattern, or at least one of a very small number of different patterns (small, medium, large). This way, when a new type of service is provisioned, the same proven configuration that was tested once and is known to work can be reused across the entire network. When faults occur in some part of the network and troubleshooting becomes necessary, chances are, the same symptoms have been observed and the same problem has been resolved somewhere else in the network before, so the same resolution can be applied. The use of cookie-cutter topologies and configuration enables network managers to build up and reuse a large knowledge base of operational experience across the network instead of needing to custom-interpret each scenario they are confronted with.

Of course, this requires significant deployment discipline. For this to work, a network provider needs to resist the temptation to introduce variations across the network unless they are absolutely required. The benefits from standardizing deployment are well known across industries, and networking is no exception. A well-known example from which important lessons for network management can be learned is Southwest Airlines, which managed to remain profitable in an industry at a time when it was difficult to do so. One of the reasons attributed to this is that the company has managed to keep heterogeneity complexity low—which, in this case, means operating only one kind of aircraft and, unlike most other air carriers, resisting the temptation to diversify the fleet. As a result, there is only one type of aircraft to maintain, flying personnel can be deployed on any flight, and economies of scale can be realized that otherwise would not be possible.

Chapter Summary

Having an operational support environment that consists of multiple disconnected islands can lead to many problems when running a network. For example, it introduces the need to maintain (and keep consistent) multiple sets of data that might overlap, it leads to the risk of management tasks that fall through the cracks and are not resolved properly, and to operational inefficiencies that might result from the friction that arises between separate groups and systems that are isolated from one another yet need to work together to run the network smoothly. Most of these issues can be avoided with management that is integrated. At the same time, achieving truly integrated management is a major challenge and hence constitutes in many ways network management's "holy grail."

Management integration represents not only the technical problem of integrating multiple functions, applications, and systems into one integrated operational support environment. It has an organizational dimension as well, involving how different groups can best work together and complement each other in running a network. Integration of operations support infrastructure can follow along the different management dimensions that were introduced in earlier chapters. It often follows the way in which operations are organized, integrating those functions first that are of importance to a particular group, while defining interfaces to the parts of the infrastructure that primarily serve other management functions and parts of the organization that the group needs to interact with.

Several factors influence the complexity of management integration, in addition to systems and software architecture challenges at the heart of the task of integration itself. These include scale (the sheer number of things that need to be managed), heterogeneity (the diversity of what needs to be managed), and the number of management functions that need to be integrated. To address management integration successfully, it can be helpful to devise ways in which the complexity of the management task itself can be reduced. For example, when possible, cookie-cutter configurations should be applied, and unnecessary variations in subtopologies (for example, across different sites of an enterprise) should be avoided.

Several approaches exist for integrating functionality of management applications. Management platforms represent one such approach, providing commonly used management functionality out of the box and complementing that with infrastructure that other applications can build on top of and plug into. The solution approach, on the other hand, is to combine best-of-breed applications and systems that can each also stand on their own. They are integrated into one common operations support environment, often requiring significant custom integration effort in the process. It is also possible to combine both integration approaches by integrating a management platform with additional systems and applications in an operations support environment. In that case, the result basically constitutes a special case of a solution in which one component simply happens to be a management platform.

The trade-offs between platform-based integration and solution-based integration commonly involve lower cost and a smaller footprint that favor of the platform approach vs. the capability to custom-tailor and optimize the operational support infrastructure to the very specific needs that favor the solution approach.

Chapter Review

1. Imagine that you are a service provider. An equipment vendor offers you an integrated management system. Which aspects of management integration will such a system be unlikely to address, even if the equipment vendor's claims are, from his perspective, entirely correct?
2. Which factor has perhaps the most significant impact on management complexity?
3. Provide an example of how management integration as a technical issue mirrors management integration as an organizational issue.
4. Give three examples of management application infrastructure that is commonly provided by management platforms.
5. Why can functional overlap between management applications in an operational support environment be a problem?
6. Provide two examples of technical obstacles that can increase a management solution's footprint.
7. Which industry consortium concerns itself with the standardization of interfaces between management systems in operations support environments?
8. Contrast shallow with deep integration of applications at the user interface level.
9. Name three ways in which complexity that is caused by heterogeneity can be reduced.
10. How do cookie-cutter deployments make network management easier? Are there any downsides?