realtimepublishers.com®

The Definitive Guide™ To Enterprise Network Configuration and Change Management

VOYENCE™

Don Jones

Copyright Statement

Chapter 2: Network Configuration and Change Management and Stability

What can change management do for the stability of your environment? Wonderful things, to be sure, but what does that mean in specific terms? What kind of return on investment (ROI) can you expect from a change-management implementation? How will it affect the total cost of ownership (TCO) for your network? How will a change-management implementation affect downtime? How much training will it require? The answers to these questions are important for making a business case for implementing change management. Interestingly, all of the numbers come down to *stability*—how change management will decrease downtime. After all, downtime costs money; reducing it saves money and creates a ROI. Reducing downtime can in many ways reduce the need for highly trained network administrators because the change-management process helps protect junior administrators from making mistakes. In addition, companies with solid change-management practices can implement changes quickly and reliably, improving their business agility and their ability to capitalize on new opportunities.

In this chapter, I'll focus on how change management can affect the stability of your environment. I'll discuss the impact of changes on business performance and start you down the path of creating a formal process to manage change. I'll concentrate on key areas such as reviewing and approving changes, prioritizing changes, working with risk, and archiving changes. No process is perfect, so we'll explore how the process needs to accommodate imperfection by providing a means of restoring stability in the event of an error.

The Impact of Change on Business Performance

How can change impact business performance? Successful changes will modify the underlying network to support new business tasks, helping the business remain more agile and responsive to changing business conditions. However, unsuccessful changes can create downtime, hinder network performance, and reduce the ability of the network to meet business needs. In fact, some companies have such a poor history with making changes that they try to refrain from doing so whenever possible. Although this practice helps prevent downtime, it also prevents the network from adapting to the business' needs, forcing the business to accommodate the network rather than the other way around. Either way, change impacts performance.

Case Study

I recently spent several weeks at a network integration firm helping them develop a business case for managing network changes more effectively. The firm had a fairly small internal network with perhaps a thousand users, but their line of business was building and managing other companies' networks. Their customers' networks often supported tens of thousands of users spread across dozens of different locations and required fairly complex network infrastructures.

The firm had a form of change management in place: any changes to customer networks must be approved by a senior administrator other than the one making the change. Manual archives of device configurations were stored on a file server and could be used to restore a device if a change didn't work out. A basic Help desk ticket-tracking system was used to manage change requests, although the system usually wasn't used to store history or an ongoing status of the change.

The firm’s problem was that too much time and money was being spent managing devices, and too little was being learned from past mistakes. We performed a task analysis and discovered that an incredible number of steps were required to make just a single change, even one as simple as updating a static route or changing a firewall configuration to permit a new port. The following list highlights the challenges that faced the firm’s change-management implementation:

- Incoming change requests could be generated in one of two ways. First, the customer might request a change, such as a new port to be opened in a firewall. Another network engineer might request a change, too, such as directing that devices be updated to support a new range of IP addresses on a particular segment.
- Changes weren’t prioritized formally. Generally, whoever was yelling the loudest got the most attention.
- If the engineer making the change wasn’t personally familiar with the customer’s network, the engineer would pull the configurations for the devices involved and spend several hours researching the potential impact of a change.
- The firm’s customers used a variety of equipment, so the firm’s engineers usually had to research exactly *how* to make a change. All of the engineers understood Cisco equipment well enough, but only a few were familiar with Bay/Nortel equipment, and only a few were familiar with software firewalls such as Raptor.
- The change review process consisted of one engineer explaining the change to another engineer, and the senior of the two reviewing the proposed new configuration and approving it. The senior engineer rarely performed any in-depth investigation and instead tended to look for simple syntax errors on the proposed change.
- Once approved, changes were often made immediately and not on a particular schedule. Once the change was made, any accompanying ticket in the Help desk tracking system was closed.
- Tickets requesting changes were assigned to engineers at random, so it was common for two or more engineers to be working on changes for the same devices at the same time without realizing it.

Obviously, this process, while seeming to involve some change-management elements, doesn’t create a more stable business environment. Customers were frequently subjected to excess downtime, mistakes, and more. The firm’s staff worked much harder than they needed to, simply because their entire process didn’t involve a timeline—there were no references to past efforts, no schedules for changes, and no thought to the future. We performed a more detailed analysis of time expended and realized that critical steps in their process—such as reviewing changes—required significantly less time than steps that added little value, such as researching how to make a change on a particular device. Figure 2.1 shows how time was spent making the average change.

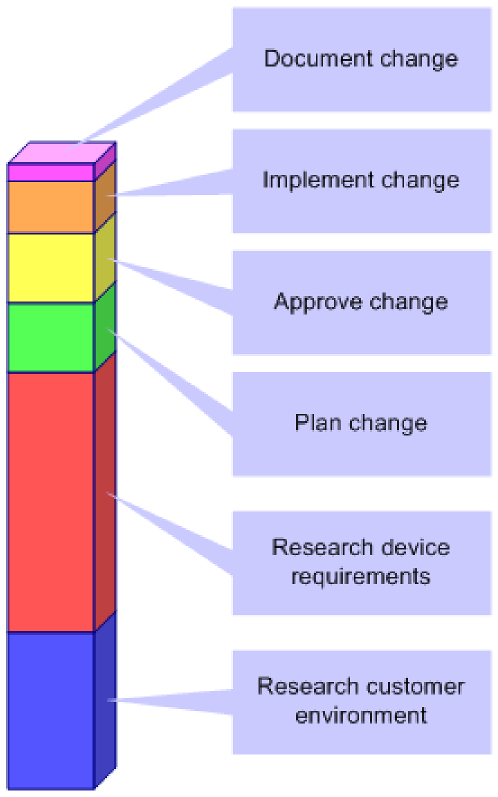

Figure 2.1: Breakdown of time spent making changes.

In my experience, this breakdown of time isn't unusual for change-management implementations. The key lesson to be learned from this case study is that the firm was wasting a lot of time—more than half of the time spent on every change—simply researching what needed to be done and how to do it. The initial solution was to simply assign engineers to a customer account. Doing so would ensure that those engineers were more familiar with the accounts' environment and equipment. Although a potentially workable solution, this method depends too heavily on a human factor. When engineers left the company, they were harder to replace because of all the specialized knowledge they carried. I made the argument that assigning engineers to an account would probably make the problem worse, because engineers would be less likely to document changes if they knew they were the only ones who would ever see the documentation.

Don't think for a moment that this situation is unique to a network integration firm. Network administrators working strictly on their own internal network spend their time in much the same way, building a body of specialized knowledge in their heads—knowledge that rarely, if ever, makes its way into documentation.

We decided that a formal change-management process would address several of the firm's issues while potentially reducing customer downtime. In this case, the ability to make personnel more efficient was a powerful motivator. Their process now incorporates the following:

- Incoming change requests are made through a ticket-tracking system, which helps to ensure that changes can be managed and that a place is available for follow-up notes and history related to the change. These notes have proved invaluable at reducing the amount of time spent researching similar changes later.
- Changes are prioritized. Emergency changes—ones addressing an active problem with a customer network—are assigned directly to a senior engineer. Other changes are assigned to junior engineers. Some customers pay for higher-priority service, and the Help desk system is now designed to recognize those priorities when assigning tickets.
- The engineer assigned to the ticket researches similar changes made in the past by using the Help desk system's search feature. This feature helps to quickly familiarize engineers with the best method for accomplishing a change as well as information about how the customer's environment is built.
- Customer information is now better organized on a file server and includes updated network diagrams, port lists, routes, and so forth. This information makes the process of researching a change much more effective and efficient.
- The change review process now uses formal change-management software. Once a change is developed, it is routed to a senior engineer who reviews it without needing to spend time contacting the junior engineer who prepared the change. The junior engineer documents the Help desk ticket with the information related to the change—which network diagrams were used, and so forth—to help the senior engineer locate necessary information more quickly.
- Once reviewed and approved, changes are scheduled for implementation based upon their priority. The change-management software provides this capability and allows the engineers to immediately see which other changes are scheduled to go to a particular device.
- To avoid multiple engineers working on the same device at the same time, engineers now work in small teams of two or three. Once a team has a ticket for a particular customer, the team handles all tickets for that customer until those tickets are closed. This setup ensures that a customer's issues are all on the table for a small group of individuals (but doesn't assign that group to the customer on a permanent basis).

Figure 2.2 illustrates how engineers at the firm now spend their time compared with the previous system.

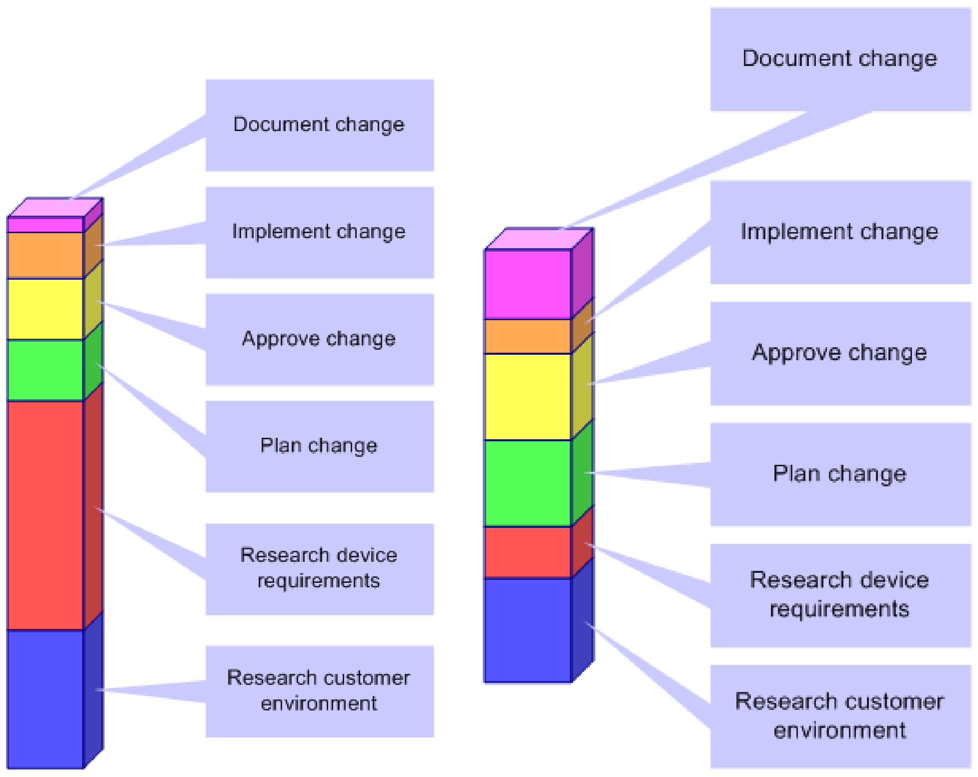

Figure 2.2: Revised time breakdown compared with previous time breakdown.

Notice in the revised breakdown (the right-hand graph) that more time is spent documenting changes, which contributes directly to the decreased research times. Engineers are no longer allowed to close tickets in their Help desk system. Instead, they transfer the ticket to a senior engineer or manager, who reviews the ticket to ensure that the change was made properly and documented properly. This final review also allows a customer follow-up to ensure that everything is working satisfactorily from the customer’s point of view. The ticket is then placed in a “pending” status for at least 48 hours to ensure that the customer doesn’t have any further issues. When customer “outage” calls are received at the Help desk, the first-tier technicians immediately review the “pending” tickets to determine whether a recent change is responsible.

Dramatic Impact

Most of the customers I've worked with in my consulting practice have realized significant changes in their business performance as a result of network change management. Unfortunately, not all of those customers were equipped to provide measurements of the improvements, but another recent example—this one of a financial firm's internal network—provides some promising figures:

- The time required to research and prepare a change was reduced by 62 percent after implementation of an effective change-management solution. This reduction is due primarily to the fact that changes are now well-documented, making future changes easier to research.
- The time required to implement a change was increased by 10 percent. This increase comes from the formal process that changes must now go through, and includes review and approval times.
- The number of hours of downtime attributed to incorrect change decreased by 89 percent in the first year. In actual numbers, that is a decrease of about 55 hours. Time is money in the financial world, and those 55 hours of downtime probably cost the company upwards of 70 million dollars the prior year.
- The number of changes that were completed by junior technicians increased by a factor of eight. Senior technicians completed just two changes on their own in the first year after implementing the change-management process; the senior technicians' primary responsibility is now future planning and change review, not day-to-day maintenance. The company was able to eliminate four unfilled senior technician positions and rely more on less-expensive junior resources.
- The impact of a senior administrator leaving the group (two of them transferred into other divisions during the first year) was minimal. The change-management process ensured that *systems*, rather than people, contained the information that had previously been locked in the administrators' heads.

The point is that network change management *does* have a significant impact on business performance and not just by reducing network downtime. Human resources can be deployed more efficiently, administrators can work more effectively, and the business can function more readily even in the event of senior administrative turnover.

☞ You can perform a quick efficiency check by simply asking your staff to solve a theoretical network issue. Choose a problem that you know one of them solved in the past, and see how long it takes each of them to individually research the problem and present a new solution. I've seen companies in which one individual spent 2 hours, on three separate occasions, researching the exact same problem—simply because the solution was never formally documented. Part of a good change-management process is maintaining a history that can act as a guide in the future.

Creating a Formal Process for Change

The argument for implementing a formal change-management process is persuasive and undeniable. I believe the reason companies must be persuaded to implement change management is because network management isn't a core competency for most businesses. Banks and credit unions don't ask for financial justification for cash-handling training for their employees because it is simple common sense that cash-handling employees must be taught proper techniques. Accounting firms don't wonder why they have to send their accountants to training every year; it is mandated by law in most places and simply makes sense in the industry. You will never hear an automotive manufacturer trying to justify a formal process that documents how cars are made—there is never a question that the process must be documented and controlled. Almost every company relies to some degree on a properly functioning network, so that network should be regarded as mission critical and managed and controlled just like any line of business process. But where do you start? What is the beginning point for a formal change-management process?

Where Are You Today?

What is your current process for managing network changes? No matter how simple or seemingly nonexistent, you should take a few minutes to document the current process. The way you do things now probably evolved out of simple need, which is a powerful business factor. Your new, formal change-management process must continue to acknowledge and accommodate you business needs, so it is wise to start with the process—however informal—you already follow. Using this methodology, your new process will be built on a foundation that at least partially serves the business' needs.

Begin by drawing a simple flowchart of your change-management process. Include all possible inputs into that process—the boss calls down and asks for a change, someone calls into a Help desk, or a problem occurs that needs to be fixed. Include all the documentation—again, however informal—that contributes to the process of making network changes. Figure 2.3 illustrates how simplistic this first flowchart might be, which is fine.

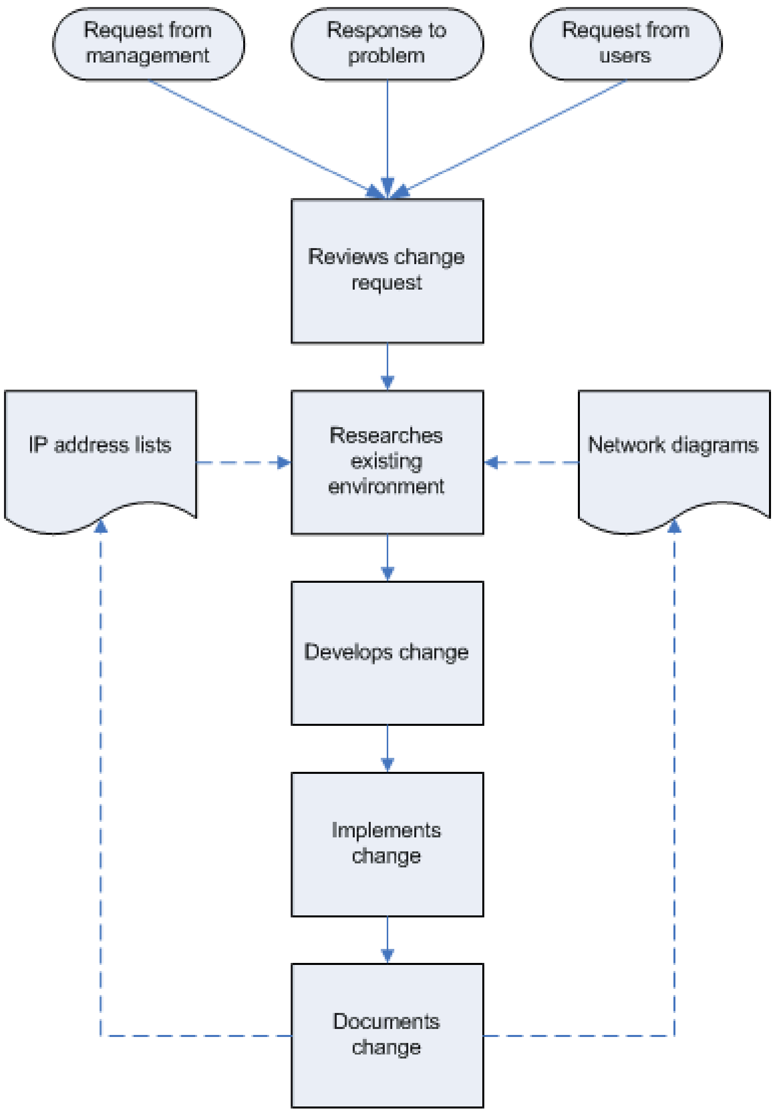

Figure 2.3: Existing change-management process.

Next, list what this current process isn't doing properly and the problems it seems to cause. You need to carefully evaluate what is broken before you attempt to fix it. Some possibilities include:

- All changes are handled start to finish by one person—No peer review or other checks and balances are in place
- Documentation is incomplete—No documentation of the existing device configurations
- No maintenance of old device configuration files to roll back to in the event of a problem
- No consistency check to ensure that changes meet corporate standards
- No automation for changes that must be made to multiple devices (such as changing the Simple Network Management Protocol—SNMP—community strings); these are all handled as individual changes
- No step to assess risk and advise management of possible outcomes from change
- No specialized handling for priority changes

- "Response to problem" input allows administrator to generate changes with no documentation
- Requests for change from managers is rarely documented, leaving no trail as to why a change was made
- No communication between staff members, meaning not all administrators will understand which changes are being made and why
- No coordination of multiple changes; administrators work independently

Additional problems that are not directly related to the process and are less obvious include:

- Senior administrator has most knowledge but doesn't document well—this problem creates a strong risk of the network being difficult to support if this person leaves
- No provisions for network changes to be retained, forming a historical trail
- No automatic documentation of changes to network devices
- Multiple interfaces required to manage each vendors' devices, which increases training and support costs
- Few tools (all vendor-specific) for automation

With a list of the problems in place, you can start modifying your process to create a formal change-management process. Your problems can probably be addressed by taking action in general categories:

- Reviewing and approving proposed changes
- Prioritizing changes
- Assigning and accounting for risk
- Monitoring pending changes
- Documenting and archiving changes
- Restoring stability after changes

Although the last step appears redundant—the point of the process is to ensure that changes don't introduce instability—every change won't work out every time. Thus, a successful change-management process must account for human error and provide a means of restoring the environment to a known-good configuration in the event that instability does occur.

In the next six sections, I'll discuss each of these categories and walk you through the steps of adding each one to your formal change-management process. Keep in mind that the process developed here will be necessarily generic and probably over-complex for many organizations. My goal is to simply show the possibilities

In Chapter 8, I'll provide examples of real-world change-management processes designed to fit a broad array of specific situations.

Reviewing and Approving Proposed Changes

A common step in many formal change-management processes is a peer review designed to help ensure that simple human error doesn't occur. By having another administrator or technician double-check proposed changes, you increase the likelihood that simple syntax errors and other common mistakes will be caught and not implemented. Figure 2.4 shows an easy change to the basic process that includes a peer review.

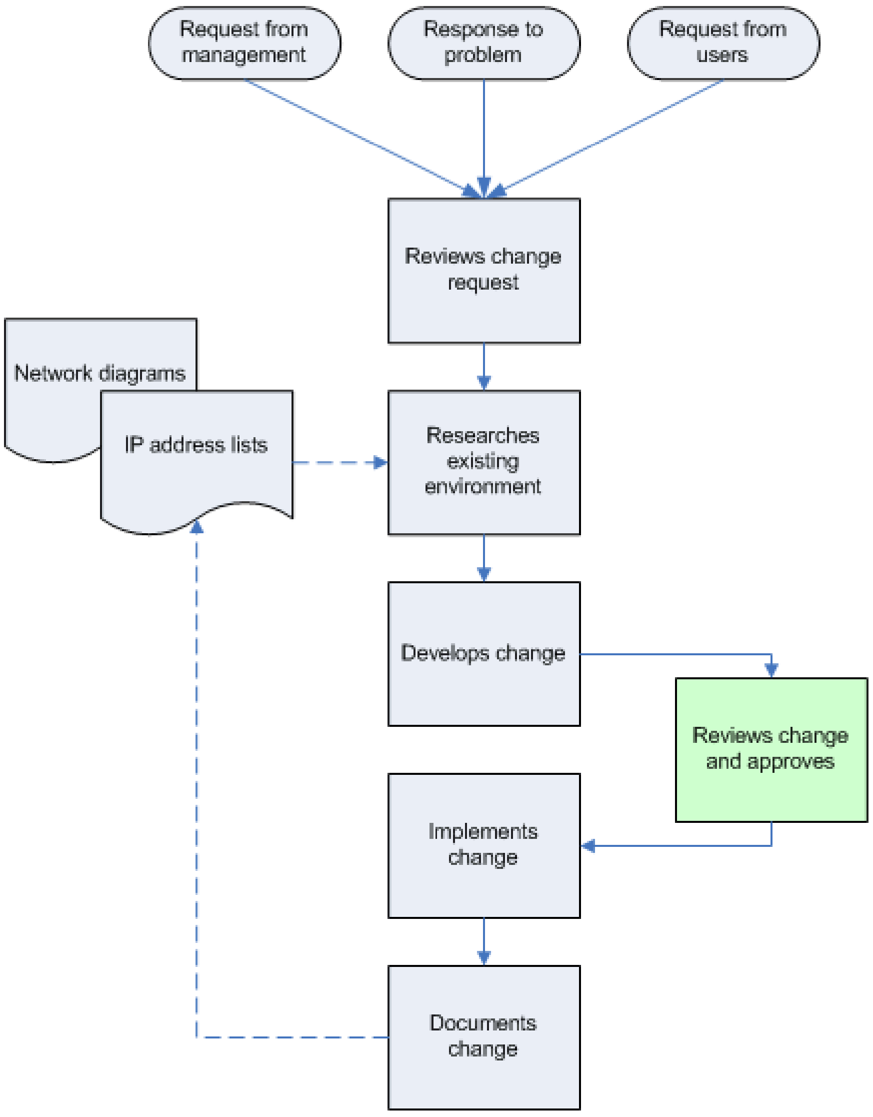

Figure 2.4: Incorporating peer review.

In this process, a separate administrator reviews the proposed change and gives it a thumbs-up. However, this simple addition might not meet all of your needs. For example, it is easier for the busy reviewing administrator to give the proposed change a brief read through, looking for syntax errors, then approve it. This type of review doesn't solve the problems presented by the original process, although it does help a bit. Figure 2.5 shows a potentially more effective change to the process.

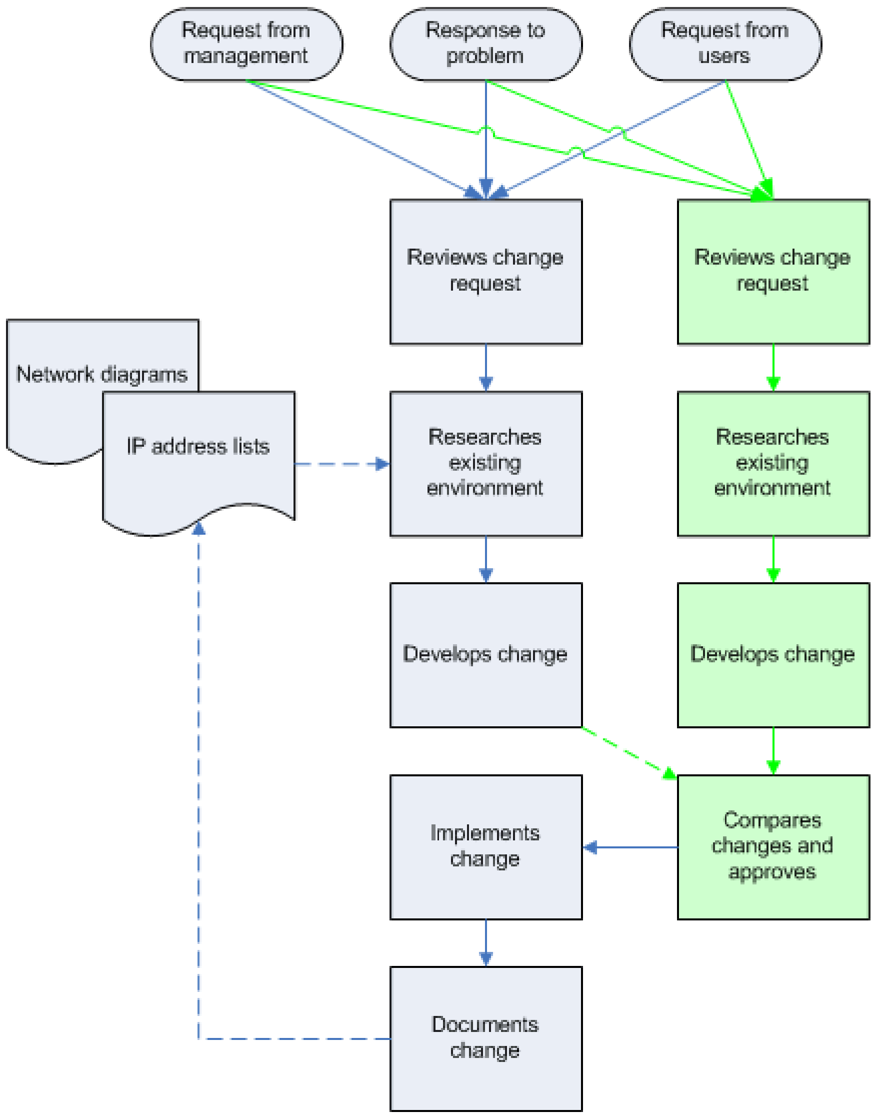

Figure 2.5: Parallel change development process.

In this revision to the process, two administrators independently develop the change. This method requires both administrators to independently research the change, which ensures that two sets of eyes are looking at the potential problems. At the end, if both come up with an identical solution, you have much better odds that the change won't introduce new problems. This revision *seems* to introduce additional work over a simple peer review, but it really doesn't. After all, when most managers think "peer review," they assume that the reviewer will be diligent, checking into all the details behind whatever they are reviewing. In effect, they're mentally working out the proposed change for themselves, then determining whether the change that they are reviewing matches the expectations. Thus, the process that Figure 2.5 illustrates simply better represents the most effective form of peer review, but doesn't involve more work than the process represented in Figure 2.4.

☞ Tools are available that enforce an effective workflow. Such tools often provide their own interface for creating network device configuration changes, and can incorporate workflow rules to require peer or manager signoffs. The tools can then deploy the approved change. If this workflow is important to you, add it as a consideration when shopping for network change-management solutions.

Prioritizing Change

You might want your process to involve prioritization for changes. After all, not every change is an immediate need, and acknowledging that within your process will help ensure that changes are treated appropriately. Prioritization is usually based upon business need. For example, you might define the following priority levels:

- Urgent—The change will correct a current problem that is negatively impacting business operations.
- Critical—Although not impacting business operations directly, the change might prevent a future negative impact. Many security updates to device software fall into this category.
- Required—The change will provide new capabilities or better functionality for business operations.
- Desired—The change will enhance operations or reduce TCO but will not necessarily provide a direct benefit to the business.
- Optional—The change will provide additional administrative features or improve administrative functionality, but does not provide a direct benefit to the overall business.

✎ You might choose to simply assign severity codes, with a "Severity 1" problem representing a production-impacting change, for example. Regardless of the categorization you develop, a prioritization system will help ensure that the most important changes occur first and receive the appropriate attention.

Deciding where you'll prioritize changes depends on where you want to control workload. For example, in Figure 2.6, change is prioritized *after* the change has been researched and created. In this case, the administrators creating the changes have sufficient bandwidth to do so; the reason to prioritize change is to control the introduction of changes into the environment. For example, "urgent" changes might be applied each evening, and "desired" changes are queued up and performed only once a month. The idea is to consolidate the risk of making changes (which can always cause a problem), while providing a means for important changes to be introduced more often.

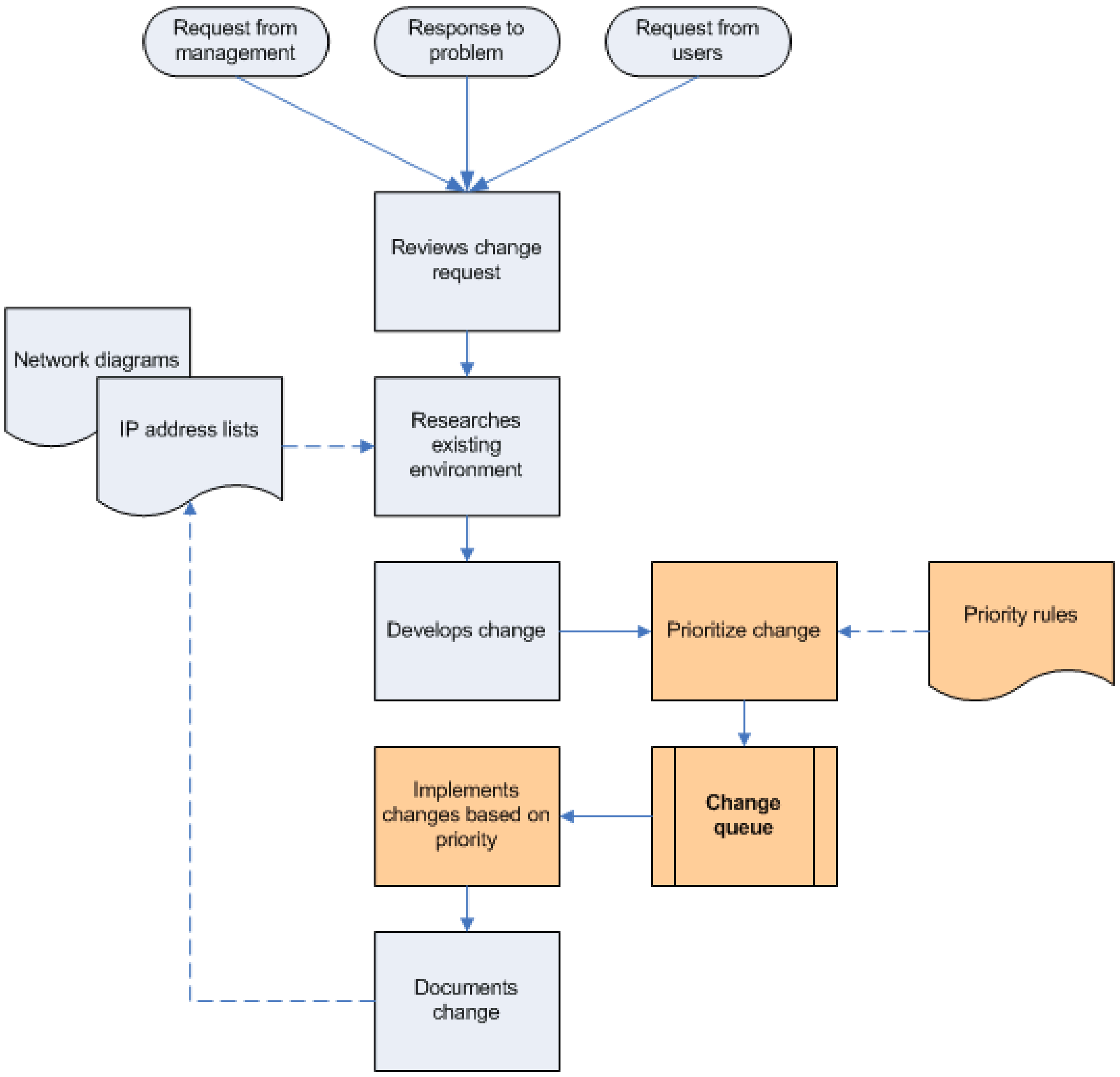

Figure 2.6: Prioritizing change to control introduction into the environment.

Another approach is to prioritize change requests as the first step in the process, and feed higher-priority changes to administrators first. This technique, illustrated in Figure 2.7, helps control the flow of changes into a limited administrative staff. Changes tend to be implemented immediately (once created and reviewed; for clarity, I've omitted the peer review that I added in the previous section), and the purpose of prioritization is to control the workload on the administrators.

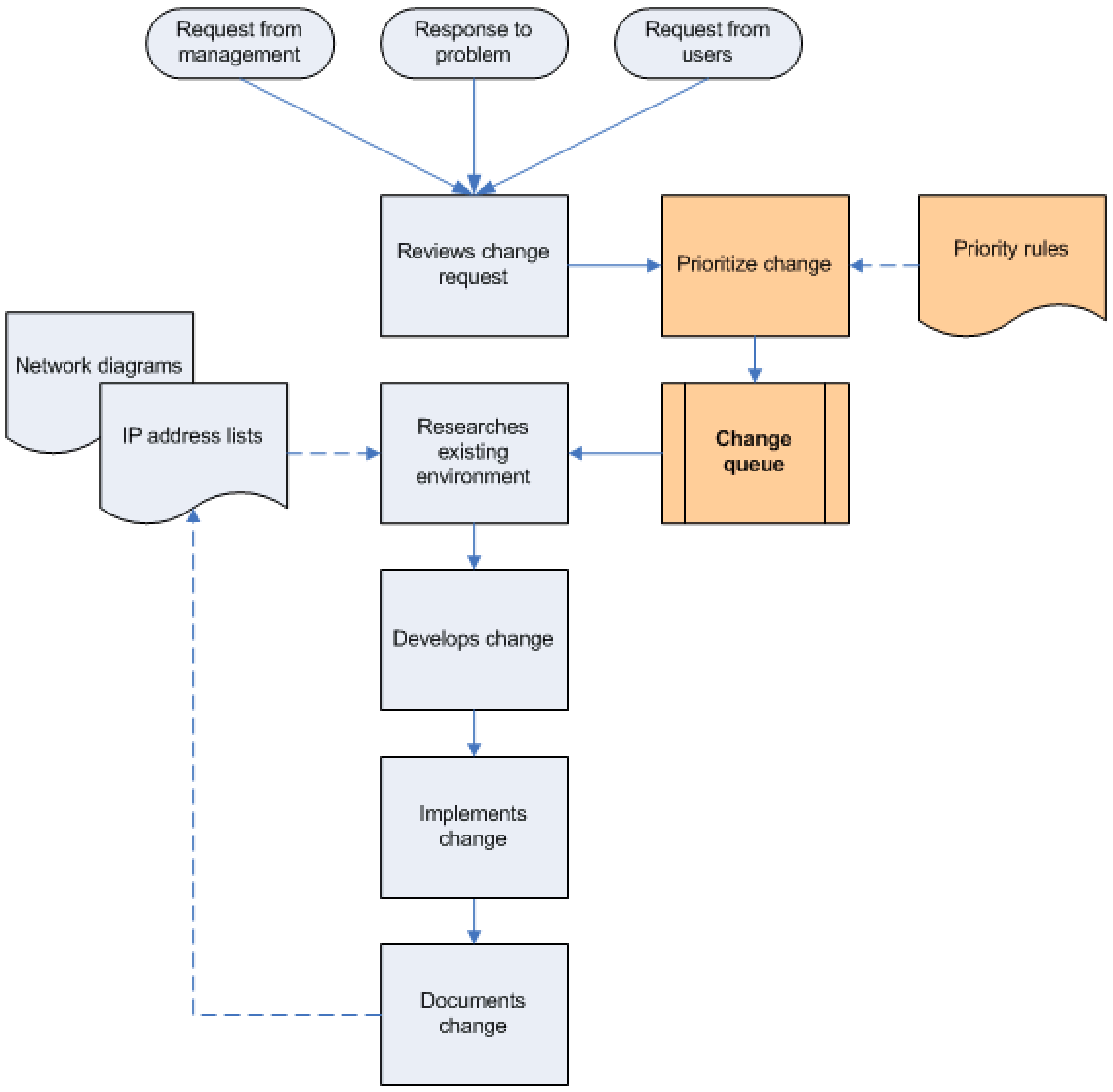

Figure 2.7: Prioritizing changes to control administrative workload.

Figure 2.7 shows the most common way to prioritize changes. A hybrid approach can also be useful. For example, you might decide to control priorities on both ends: administrators work on high-priority changes first, then queue them up for application. Only high-priority changes are applied each evening (or on the schedule you determine); any lower-priority changes that an administrator completes are queued up and applied on a less-frequent basis to consolidate the risk of making changes.

A downside to applying several queued-up changes at once is that it becomes more difficult to troubleshoot any problems that arise. It is at this point when priorities and risk—which I'll discuss in the next section—can have a delicate interplay. You might, for example, dictate that high-risk changes—such as ones that affect multiple devices—be applied individually regardless of their priority.

Assigning and Compensating for Risk

Your change-management process should also factor in the risk of making a change. For example, you might include a brief risk analysis in your change review process. Changes that have a higher risk of damaging the environment might be applied individually regardless of their priority. This setup might cause the change to be applied earlier or later than the normal schedule so that it can be applied by itself rather than with other queued-up changes. Higher-risk changes might also incorporate additional reviews to help reduce the risk of the change causing a problem in the environment. Very high-risk changes might require a formal lab test prior to production implementation. Your risk levels should be assigned on a dual basis: the likelihood of the risk being realized as well as the potential damage. Table 2.1 shows an example risk level assignment.

Likelihood	Impact
0—Very unlikely	0—No negative impact
1—Unlikely	1—Fixable within minutes, minor impact
2—Possible	2—Major impact, users or operations affected
3—Likely	3—Severe impact, operations affected for a significant period of time

Table 2.1: Example risk level evaluation.

To establish a complete risk level, simply multiply. For example, suppose you’re getting ready to deploy a new firmware update to your routers. The manufacturer has a reputation of problems with firmware updates. Such a change would be assigned a 3 for likelihood and a 3 for impact, for a total risk value of 9. Suppose you need to clean up the remark lines in your device configuration files. It is unlikely that a problem would occur, and because you can always restore the old configuration file, it is easily fixed if something does go wrong. Such a situation would have a 0 risk level.

Your process should have paths based on total risk:

- 0 to 1—No additional action required; ensure device configurations are backed up prior to making change
- 2 to 4—Additional peer review and approval required
- 5 to 6—Additional peer review required and a formal lab test required
- 7 to 9—Formal lab test and signoff by entire administration team required; change must be implemented by itself

Figure 2.8 shows how risk level can be incorporated into your change-management workflow.

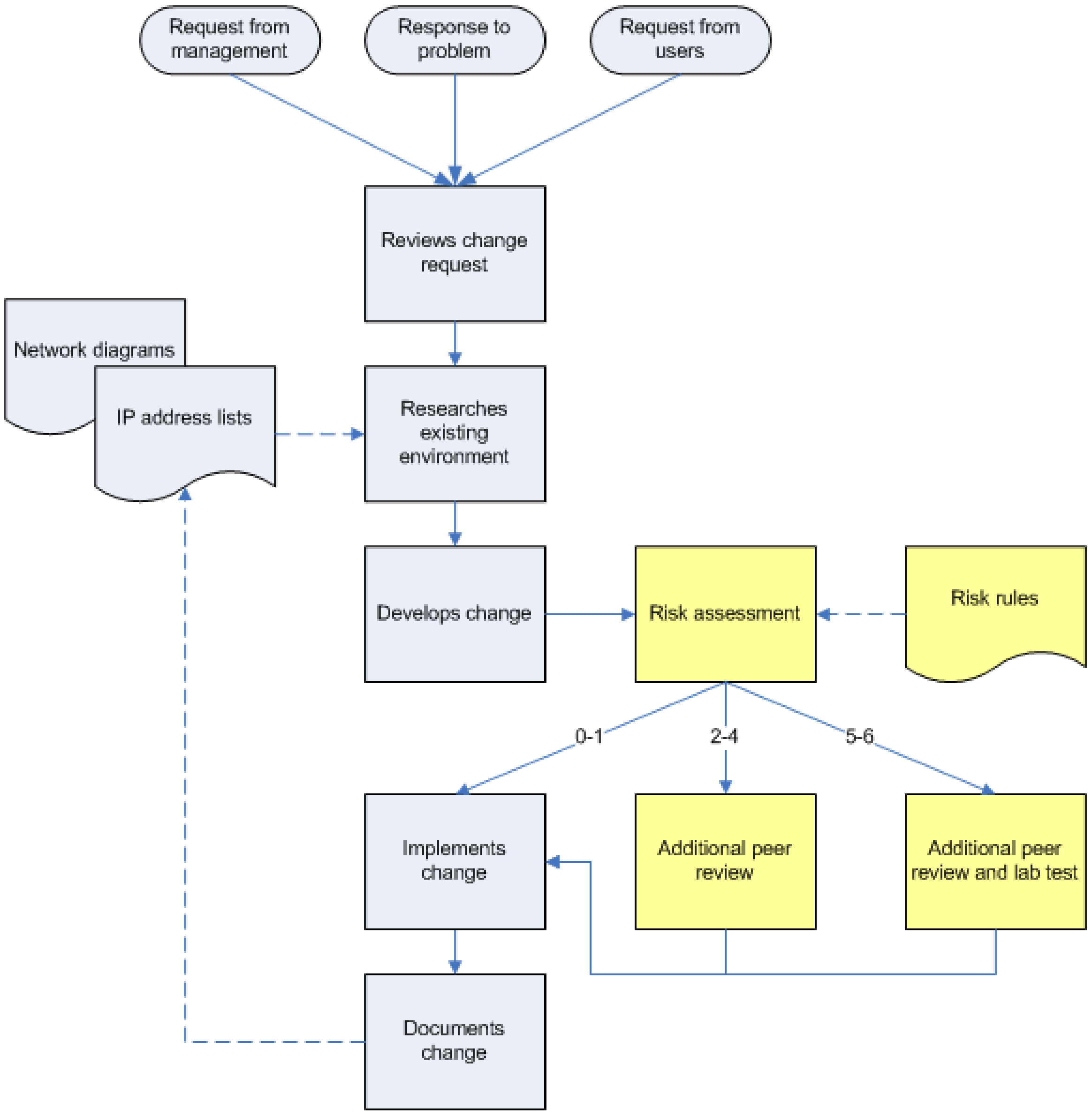

Figure 2.8: Adding risk assessment into the process.

The risk assessment should be performed by an experienced administrator with an understanding of the business cost of a network failure. In addition, the assessment should include a search of past changes. For example, the fact that all firmware updates from a particular manufacturer *always* cause problems should be apparent in your change-tracking system. It is important to track the reasons for changes (for example, through a Help desk ticket-tracking database) and to include follow-up notes about the problems that occurred as a result of the change. This "searchable past" will enable you to make better risk assessment decisions about future changes.

☞ Have a mitigation plan for every change. Assume on some level that every change will go wrong, as there are many factors outside your control (such as firmware bugs and data transmission errors). Consider requiring that all changes be accompanied by a rollback plan (such as restoring an earlier version of the device's configuration or installing a hot-spare device) that can be implemented to quickly undo the change and restore the previous working environment.

✎ Note that risk assessment is at the end of the process rather than the beginning. Oftentimes, risk can't be fully understood until you see exactly what is being changed. For example, implementing a new TACACS+ server might not seem like a big deal until you realize that it involves eight configuration settings on 30 devices; the sheer scope of this change might warrant a higher risk rating. The risk assessment can usually be combined with your usual peer review process, provided the peer reviewer is (as they should be) a senior administrator with a great deal of experience in your environment, who works with someone who has a business perspective on the change to provide the business-level input to the risk assessment.

Monitoring Pending Changes

Some of the most effective change-management processes incorporate software tools to monitor pending changes. Once administrators create a change, they submit them to a queue, where they are automatically applied on a scheduled basis by the software. The software knows to back up device configurations prior to making changes, and it can provide reports of upcoming and recently completed changes.

☞ Tools can help you avoid simple scheduling problems as well. For example, tools that allow you to schedule changes to be implemented at a future time make it easy to realize that you've scheduled a change to occur on a Monday (which is usually busy enough) or just before a holiday (when nobody will be around to handle any problems that occur the next day). Simply avoiding these silly errors can help prevent major support nightmares.

Every member of your IT team should be aware of upcoming changes, particularly those changes that have a high risk rating. This communication allows the team to be sensitive to incoming trouble calls that might have been caused by the change. These troubles might be as simple as users who can't connect to the network because they've disabled the Dynamic Host Configuration Protocol—DHCP—on their client computers and their segment was recently re-addressed. Making sure first-tier support understands the impact of the re-addressing will allow the support team to walk users through a solution without worrying that "something's wrong with the network today."

In addition, troubles might be more severe and occur simply because the complete impact of the change wasn't realized by the administrator making the change or by the reviewer. In these cases, having the Help desk understand that a change occurred will help ensure that related troubles are quickly escalated to the correct support level for corrective action. Figure 2.9 shows how scheduling changes and integrating with various support tiers can be added to your basic workflow.

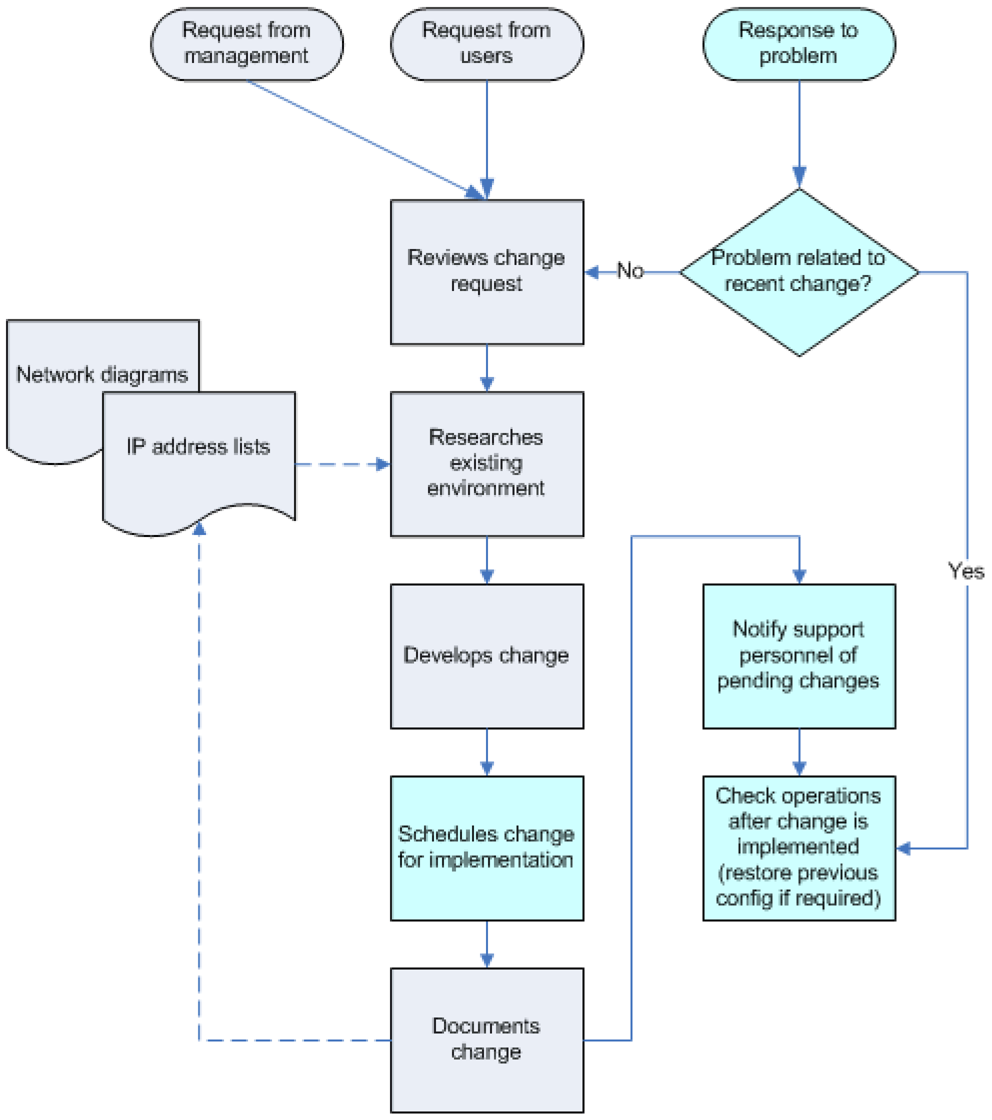
Request from management
Request from users
Response to problem
Reviews change request
No
Problem related to recent change?
Network diagrams
IP address lists
Researches existing environment
Yes
Develops change
Notify support personnel of pending changes
Schedules change for implementation
Check operations after change is implemented (restore previous config if required)
Documents change

Figure 2.9: Scheduling changes and notifying support personnel.

Ideally, support personnel are informed of changes the day before changes occur. Significant, high-risk changes might warrant earlier notification so that the first tiers of support can bring in additional staff to handle anticipated higher call volumes. Although the peer review, prioritization, and risk analysis is to ensure that changes *don't* cause problems, changes nonetheless *can* cause severe problems no matter how carefully they're reviewed or made. Keeping a schedule of upcoming changes and working to ensure that first-tier support personnel understand upcoming changes will allow your entire team to operate more efficiently.

Documenting and Archiving Change

You can't always trust people to update documentation. Opening Visio, Excel, or whatever program you use for documentation can be more time-consuming than simply making the change and forgetting about it. Unfortunately, a lack of updated documentation will result in a cascade of poor change decisions later.

> ☞ The beautiful network diagram someone spent 3 weeks making is the least likely to be updated when changes occur. Try to keep documentation as utilitarian as possible so that changes can be incorporated more easily.

The solution to lazy documentation is to employ a tool that can automatically provide documentation for you. The tool should be triggered to scan device configurations whenever a change is made and to scan on a regular basis—just in case some change occurs outside your regular change-management process. The tool should save change histories to a searchable database, allowing you to construct a history of what has happened to each device on your network. The tool should also help maintain documentation, such as device IP addresses, serial numbers, and other basic information. Additional tools might be able to automatically scan your network and construct network diagrams based on the devices and subnets that the tool finds. These tools provide an invaluable, efficient way to automatically create accurate documentation for your network configuration.

In short, take documentation *out* of your process as much as possible and make it an automated function that occurs in the background. Figure 2.10 shows how this provides useful, accurate input to the overall change-management process without requiring specific action on the part of administrators.

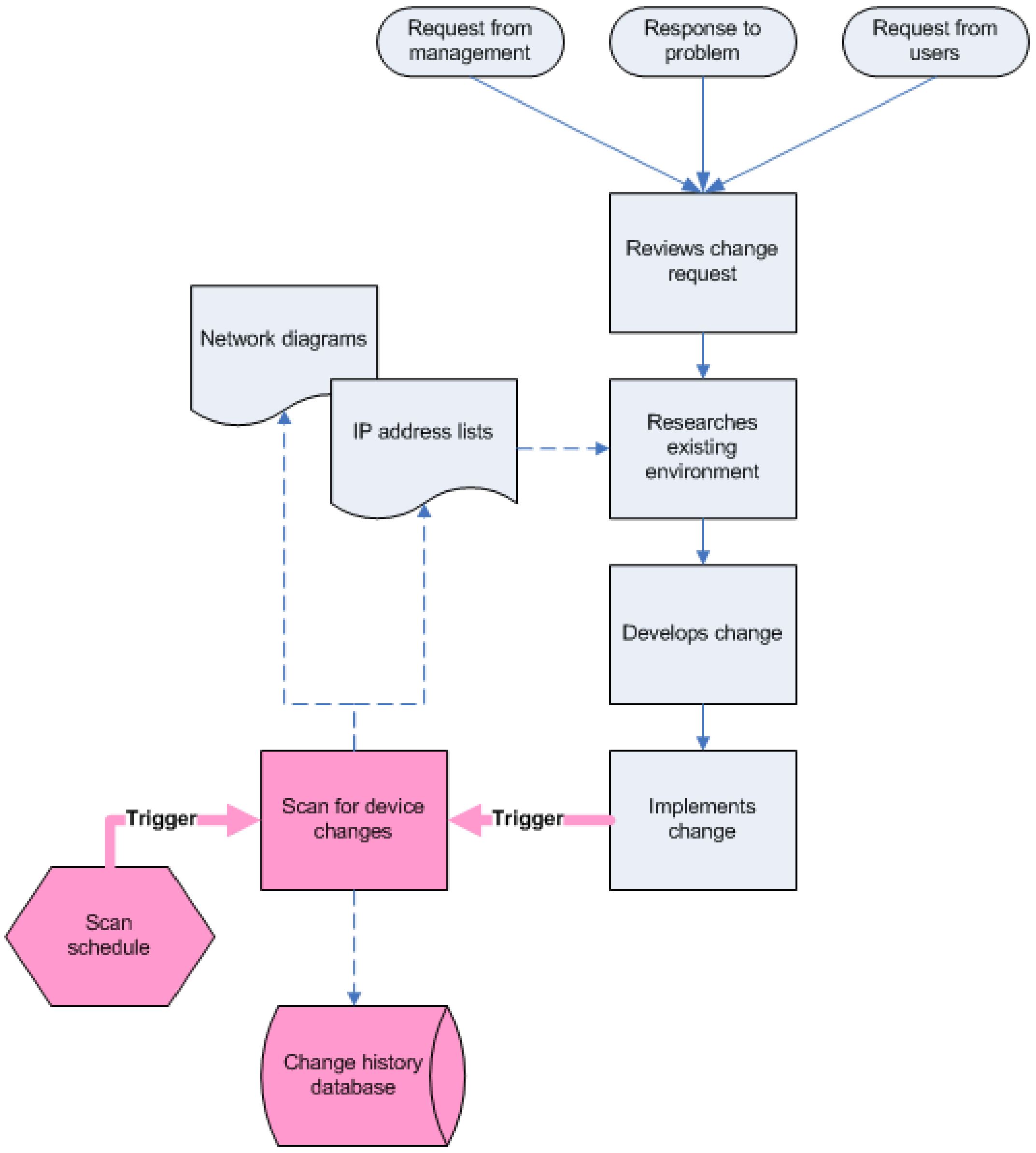

Figure 2.10: Automating documentation and change history collection.

☞ Tools for automatically collecting a change history can be as simple as scheduled scripts that use cron and tftp to periodically download device configurations and diff commands that highlight changes in that configuration history. There is also a growing market of dedicated network device management tools that provide a better user interface and more options than cobbled-together scripts.

Another technique, illustrated in Figure 2.11, doesn't require tools (although tools would still make the process easier to actually implement). In this workflow, changes sent for peer review must be accompanied by updated documentation. The change and documentation are reviewed as a set, and once the change is implemented, the new documentation becomes valid. You will want some kind of versioning or document-management system to control the documents so that only a central authority—perhaps the senior administrator who reviews and approves changes—can actually implement the updated documentation when a change is approved.

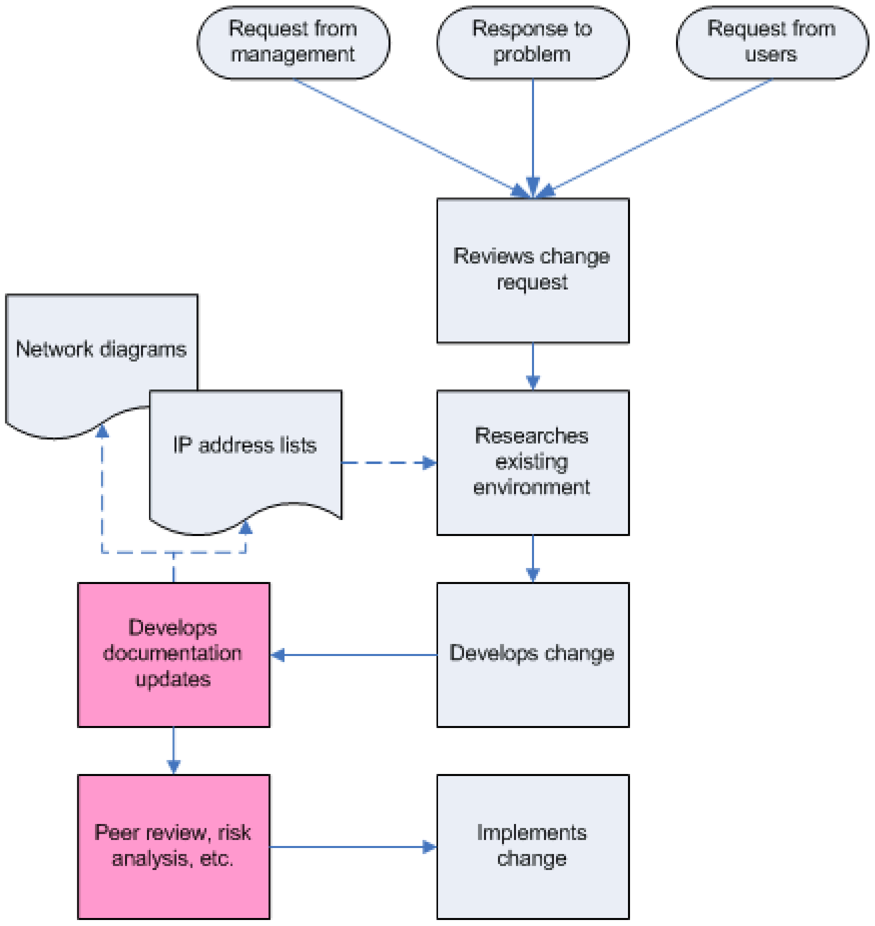

Figure 2.11: Requiring updated documentation along with changes.

☞ If you use Visio or other graphic documents, consider a versioning system such as Microsoft SourceSafe, Documentum, or some other version-control system. You will be able to check in updated documents and diagrams with a comment, such as "reflects change made in ticket #16475." This information will provide a link back to your ticket-tracking system, helping the documentation become better integrated with your change history.

Also, if your documents support it, use change tracking (such as Microsoft Word's "Track Changes" feature). Accept all outstanding changes when you begin working with a document, then track the changes that you make to it. Combined with a version-control system, you will be able to retrieve any prior version of a document and see exactly which changes were made to that version compared with prior versions.

Restoring Stability After Changes

The point of the change-management process is to prevent problems from occurring as the result of a change. However, no matter how perfect your process, problems will occur. Simply put, no process is completely perfect. Even if everyone on your team reviews every change and tests them in a lab, problems will still sneak through to production. Your process needs to recognize that problems will occur and incorporate the means to fix those problems as quickly as possible.

Generally, the fastest fix is to put things back the way they were—to a "known good" condition. In fact, most change-management tools provide this capability: They archive device configurations and can push an older, known-to-be-working configuration back to a device that is not working correctly. You should take advantage of this capability in your process (see Figure 2.12).

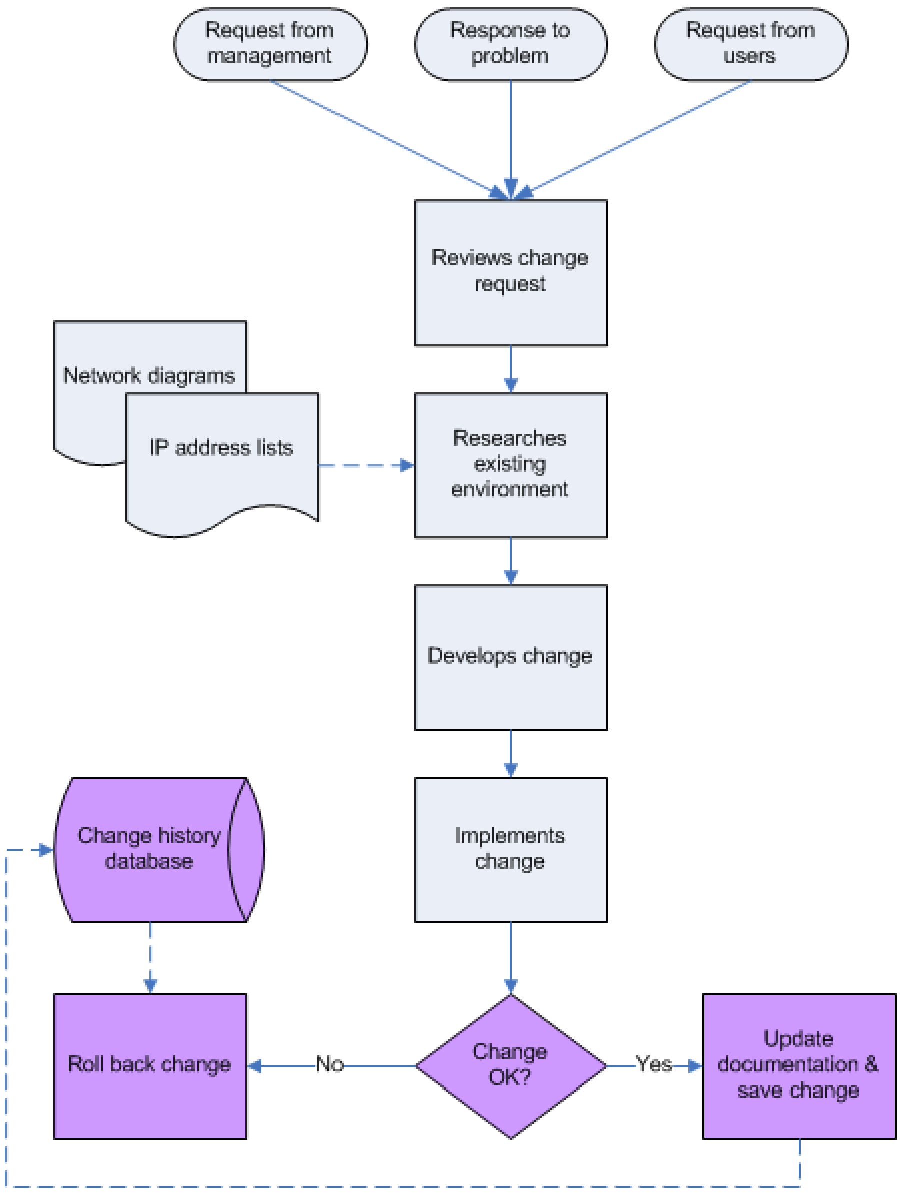

Figure 2.12: Rolling back changes that caused a problem.

Strictly speaking, you don't need a tool to add these steps to your process. You could manually back up your device configurations using tftp or other command-line tools, then restore an older configuration if necessary. However, having a comprehensive change-management tool makes the process easier, especially if multiple devices have been affected.

The ability to roll back, or undo, changes to network devices should be regarded as "Plan B." While this is an essential feature, you should need it for perhaps 5 percent or fewer changes made to your devices. Proactive change management means examining and reviewing changes *before* they are made to catch problems *before* they occur. Using a roll-back capability necessarily means that a problem has occurred and has potentially affected business operations; it is far better to catch problems in advance before they affect productivity.

The Whole Process

Combining these six categories can create a complicated-looking process. However, as Figure 2.13 shows, you can make the process understandable by rolling up some tasks—such as a comprehensive peer review—into a single step that you break down in another flowchart.

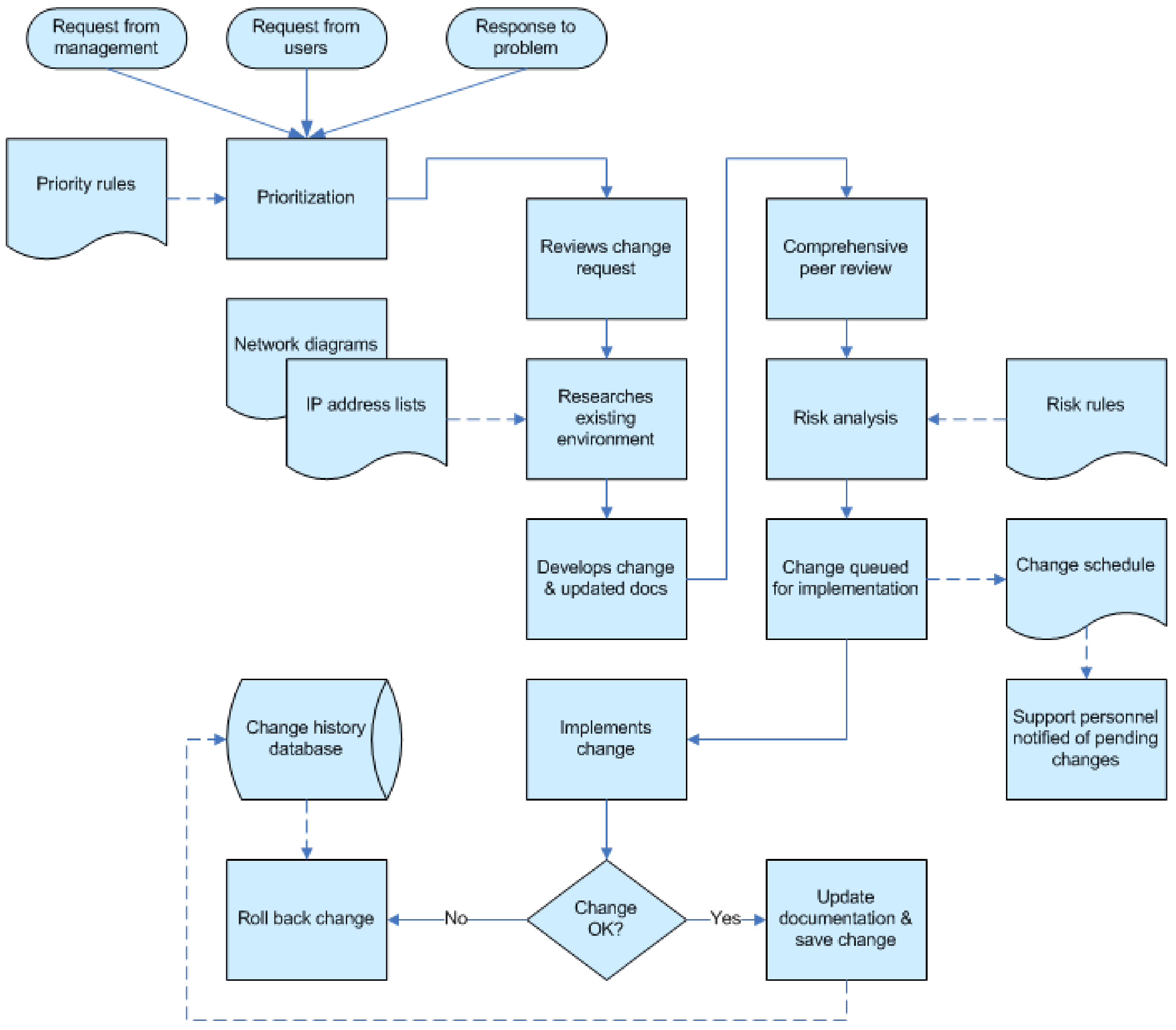

Figure 2.13: The whole change-management process.

More important, a comprehensive change-management tool can help automate this process for you. The change history database, rollback capability, interface for creating changes, workflow for reviewing and approving changes, and the means to schedule changes for implementation can be automated and enforced by a change-management solution.

I'll provide case-specific workflow suggestions for several environments in Chapter 8. The sample process in Figure 2.13 is a hybrid that illustrates many change-management best practices but isn't applicable to every environment.

Summary

Effective change management *always* has a positive benefit on business performance. It reduces downtime, increases administrator efficiency, reduces reliance on specific individuals within the organization, and helps create a consistent, repeatable business process for managing network change. The various elements of a change-management process—risk analysis, prioritization, and so forth—can be used to solve specific business problems and to make the environment better able to handle new changes to the network. These processes can also help make the network more responsive to business needs so that changes with the potential to positively impact the business environment receive increased attention.

Change management also has an impact on the security of your environment. It allows your environment to be more secure and provides hooks for change auditing, a process that is becoming more important to many companies, and even legally mandated for others. The impact of change management on network security will be the topic of the next chapter.