


Realtime
publishers

The Shortcut Guide[™] To



Untangling the
Differences Between
High Availability and
Disaster Recovery

sponsored by

MARATHON
Run to Infinity

Richard Siddaway

Chapter 4: People and Poor Processes: Eliminating the Biggest Causes of Application Downtime 59

- Identifying Downtime Causes 59
 - People 60
 - Mistakes 61
 - Ignorance..... 62
- Three Stages 62
 - Plan 62
 - Spend..... 64
 - Test..... 64
- Educating the Business 64
 - SLAs 65
 - The Art of the Possible..... 65
- People 67
 - Training..... 67
 - Skills and Knowledge 68
 - Management..... 69
 - Education..... 69
- Process 70
 - Documentation 70
 - Change Control..... 73
 - Maintenance..... 74
 - Automation..... 75
- Applying High-Availability Principles to Disaster Recovery 75
 - Document 75
 - Test..... 76
 - Automate 76
- Summary: Eliminate Downtime Before It Eliminates You 76

Copyright Statement

© 2010 Realtime Publishers. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtime Publishers (the “Materials”) and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtime Publishers its web site sponsors. In no event shall Realtime Publishers or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtime Publishers and the Realtime Publishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtime Publishers, please contact us via e-mail at info@realtimepublishers.com.

[Editor's Note: This book was downloaded from Realtime Nexus—The Digital Library for IT Professionals. All leading technology books from Realtime Publishers can be found at <http://nexus.realtimepublishers.com>.]

Chapter 4: People and Poor Processes: Eliminating the Biggest Causes of Application Downtime

In earlier chapters, we examined high availability and disaster recovery. We have looked at the differences, the similarities, and how, in some circumstances, the two are converging. The ability to deliver high availability and disaster recovery from the same technology can supply a cost saving to the organization as well as a greater depth of assurance that services can be maintained.

The configuration of high availability involves answering a number of questions, as we saw in the previous chapter. The high-availability solution can be designed and built once you have that information. Creating an environment that will supply high availability for your business processes is just the first step. You have to keep the systems running so that there isn't an interruption in service.

The continuation of service is a matter of people and processes. High availability can only be supplied, and maintained, when the people and processes are correct. In this chapter, we will examine how we can minimize, and preferably eliminate, the biggest causes of downtime:

- Poor processes
- Human error

The first step in eliminating downtime is identifying what causes it.

Identifying Downtime Causes

There are many possible causes of downtime. In Chapter 3, we looked at designing possible causes of downtime out of the environment. That discussion centered on removing the technological causes of downtime. Any electrical, or mechanical, system will fail eventually as components wear out or collapse. Good design will remove these points of failure by ensuring that redundancy and resiliency are built into the system.

There is one portion of the system to which we can't apply these design principles: the people who use and administer the system. Figure 4.1 illustrates this point. There are factors in any organization that act to keep a system running and another set of factors that could bring the system down. A high-availability system must be created to maximize those forces that preserve availability and minimize the potential of downtime. Keeping costs down is an additional consideration.

This discussion deliberately ignores the possibility of intentional damage being caused by someone inside or outside of the organization. In the latter case, your external security should be sufficiently layered to prevent a security breach. This responsibility is on the system designers. An internal attack is more difficult to guard against but careful, and correct, application of the principle of least privileges will aid in preventing such an attack. If the users are limited to just the privileges they need to do their jobs, they can't cause damage to the system.

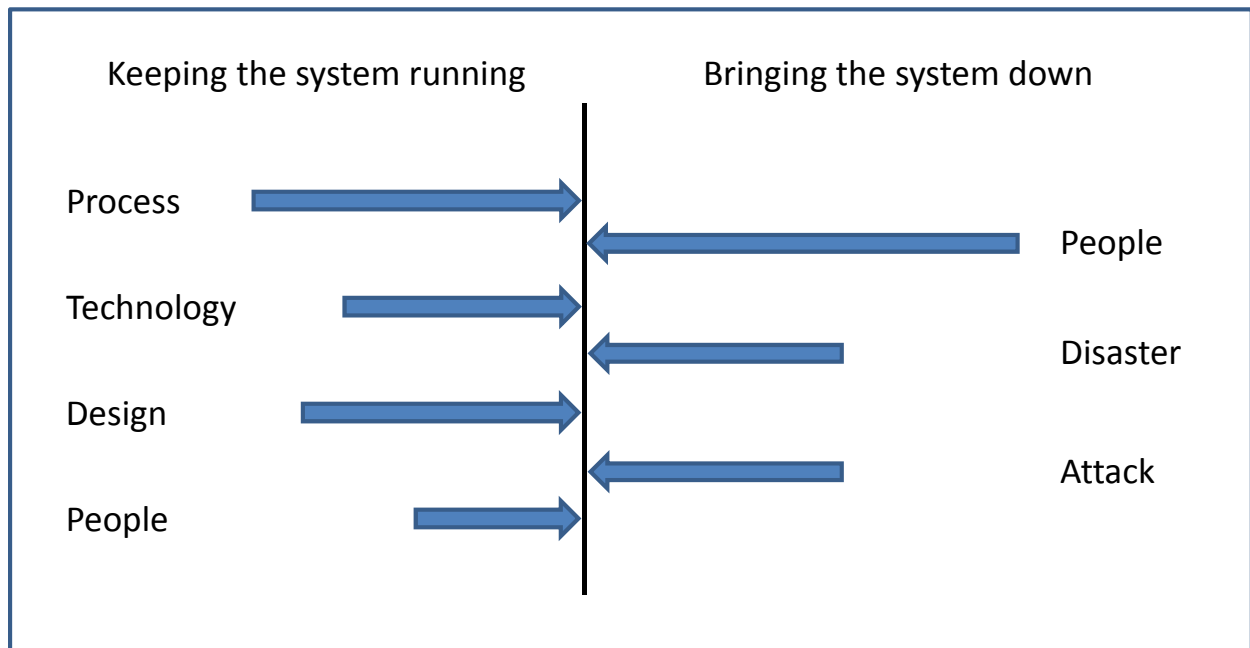


Figure 4.1 Forces acting to keep a system running and to bring it down.

There is still the possibility of a rogue administrator deliberately setting out to cause damage. This problem will not diminish. IT systems are getting more complicated and the number of people who really understand them is shrinking accordingly. If these people aren't trusted to administer your systems, why are they allowed access? How is vetting applied to new administrators? These questions have to be answered by each organization. Assuming that your administrators are deemed trust worthy, the possibility still exists that they can cause your systems to fail.

People

People make mistakes. Unfortunately, with high-availability systems, these mistakes can be very expensive and have a damaging effect on your career!

There are a lot of reasons for things going wrong because of people. Some of the reasons for making genuine mistakes can be found in the next section. There are other sources of error introduced by people:

- Tiredness—Especially if they are working “out of hours” after a full day of administration
- Distractions and being inattentive—Multi-tasking has its place but not when you are trying to reconfigure a business-critical system; one thing you do not want to hear from your junior administrator is “Oops. I didn’t mean to click on that!”
- A fear of being perceived ignorant—If you don’t know how to do something—ask! The other option is to research the technique before applying it in a production environment and double check with a colleague

Computers will always follow the instructions we give them. It’s called programming. People can deviate from the instructions they are given for a number of reasons:

- Attempting to take a short cut
- Lack of understanding
- Even simple misreading or misinterpreting the instructions

Anything involving people can involve error. Some of these mistakes may be genuine, as we will see next.

Mistakes

Mistakes happen. We can all probably recall an occasion when everything seemed to be going well and a mistake turned it all around to make a big mess. Mistakes can be made for a number of reasons:

- Lack of knowledge
- Poorly communicated instructions leading to misunderstanding and errors
- Complexity of systems again leading to a lack of knowledge and understanding
- Changes to the system that haven’t been properly communicated, leading to the wrong actions being taken
- Pressure to complete the work quickly because of other tasks that need to be performed

Some of these can be overcome with training and procedures ensure that the only people allowed to perform a critical task have been trained to do that task. This may not need an expensive external training course. It is possible to perform the training in-house by a more experienced administrator.

Procedures can involve the use of checklists or an observer to ensure that the tasks are performed correctly. Automation may be used to remove the possibility of human error, but this technique can introduce its own pitfalls as we will see later.

Ignorance

People acting through ignorance have the greatest capacity to damage your systems. A simple change can have profound implications; for instance, a change in the way a system behaves between versions. Imagine an administrator that learned their trade on Windows Server 2003. They are asked to install Windows 2000 one day. If they haven't done some research, or been told, they may well not realize that Windows 2000 installs IIS by default. This immediately extends the attack surface of a machine and makes it more vulnerable. A build process that documented this would prevent this mistake happening.

A similar situation exists when considering new technologies. Do you ever admit you don't know how something new works? No? You aren't alone. There are a lot of technical people who seem to think it is wrong to admit not knowing the answer to a technical question.

It is the organization's responsibility to ensure that their technical personnel are skilled in the technologies they will be asked to support. If they don't have those skills, you can say goodbye to your business process because a mistake will be made due to them not understanding the technologies. The mistake will lead to downtime. This is particularly true in the case of high-availability systems, not necessarily because the technologies are more complicated but because the systems are more critical and therefore more sensitive to mistakes. We need to determine how we can ensure that our people can support the high-availability systems our businesses need to thrive.

Three Stages

Putting the support and administration team in place for a major system is not a task that can be left to the end of the implementation project when it is handed over to the service delivery teams. They should be involved in the project for a number of reasons:

- Intimate knowledge of the system because they helped build it
- Ownership of the system because of their involvement
- Exposure to the technologies
- Training, possibly from the vendor

These activities don't just happen. We have to start by planning for them.

Plan

Planning is an essential part of IT activity. It may seem obvious when discussing a project, but it is equally important for a support organization as well. If the activities aren't planned, mistakes will happen and important tasks won't be completed.

Planning involves two major themes:

- What has to be done?
- Who will do it?

The first theme encompasses the tasks that have to be performed to keep your systems running. These tasks fall into a number of areas:

- Daily, weekly, monthly, quarterly, annual ,and less frequent tasks
- Monitoring—Do you know the state of your system?
- Capacity measurements—How fast is your disk space growing? Is there sufficient head room in processor power to accommodate the expected growth in the number of users?

There should be a defined set of tasks that are performed for each of your systems. The obvious daily task is ensuring the backup is performed. Other daily tasks that may be performed could include checking:

- The event logs
- Any application logs
- Whether scheduled or batch jobs have completed successfully
- Whether any data replication techniques are working correctly
- Cluster witness (quorum) is available

The list can be expanded both for the daily tasks and the tasks for the other timeframes based on the needs of your application. A good starting point for further tasks is vendor-supplied administration guides. In many cases, there will be a set of tasks that need performing. You won't necessarily get the scheduling information, but it will help.

As part of the latest version of Microsoft Operations Framework (MOF), Microsoft is releasing a set of reliability workbooks for major applications. These can be used directly or modified to suit your circumstances.

Whatever you put into your schedule, be sure to record that the task was performed and the outcome. The symptom you record may not have an obvious reason but may be a signal that could help with troubleshooting in the future.

The question regarding who will perform a task may have an obvious answer in many organizations—"that's Fred's job." What happens when Fred isn't available? Do other people know how to perform this task? The concept of "single point of failure" has been discussed in previous chapters with respect to technology. It is just as valid when applied to people. I have seen a number of organizations struggle when a particular individual wasn't available because he was the only one who could perform a task and *there wasn't any documentation*.

Once you have decided who is responsible for a particular task, documenting that task should be that person's responsibility. That includes maintaining the documentation. Other personnel can then learn to perform the task from the documentation and have the benefit of having the expert available. High availability can be an expensive proposition from a technology point of view. It is also expensive from a people viewpoint.

Spend

A number of studies have examined the costs within IT organizations. The results vary but at least 70% of your budget could be used to “keep the lights on.” This is especially true of high-availability and disaster recovery systems. The price of maintaining these systems will not diminish over time. If you are attempting to get high availability or disaster recovery on the cheap, it won’t work. This is definitely one area where you get what you pay for.

The budget for establishing the system should include the costs of maintaining the system until the next budget period is reached. Any attempts to reduce the budget for administering your high-availability systems should be resisted. They are highly available because they have been designed correctly and are looked after properly. Any change from that situation can cause the systems to be unavailable.

Cost is an ongoing fact of life. It’s not a one-off cost to establish and maintain a high-availability system. In addition to the set up and administration costs, you also have to commit budget and/or resources for testing.

Test

Any system needs to be tested. When we are talking about high-availability systems (and disaster recovery), we need to think about the sorts of testing we need. When the system was commissioned, we would have proven that the application worked and performed as expected. Have those tests ever been repeated?

- Do you know your application will fail over to another cluster node?
- How do you know your replication is working?
- Can your new administrator perform a manual failover in the event of a problem?
- Do your monitoring and scheduled tasks actually work?

You need to be able to prove that the system works and that what you are doing keeps it working. Tests on the system should be planned and scheduled to ensure that it is still behaving as expected. This has to be done without impacting performance or availability!

High availability is not just the responsibility of IT. The business needs to use these high-availability systems to perform their business processes. This means they have a responsibility in helping to maintain the high-availability systems.

Educating the Business

The business controls what IT can do and needs to do. Unfortunately, those decisions cannot be made without an understanding of what is possible and what is involved in delivering on those decisions. This is often portrayed as the fault of IT because they can’t talk in business terms; however, there can be an element of “I have a PC at home so I know all about IT systems” from the business side.

Discussions between IT and the business are essential to ensuring that systems meet the requirements of the business processes and that they are configured to provide the correct level of availability. One area that can lead to confusions is Service Level Agreements (SLAs).

SLAs

An “SLA” is an agreement around the level of service to be delivered. If we are talking about a critical application, it will normally be expressed as percentage availability over a particular timeframe. An important fact to establish is when the application should be available; for example:

- During core business hours Monday to Friday
- 8am to 8pm Monday to Friday, and 8am to 4pm Saturday
- 24×7 on a continual basis

The first option would seem to be easiest to deliver against, but do you have a written agreement as to what constitutes core business hours? No? Oops your SLA has just been breached. Rather than use a vague term, even if it is understood throughout the business, ensure that actual times are defined.

The other aspect of an SLA that needs to be agreed is the percentage availability. 100% means NO downtime in the given period. That means that a failover on a cluster, or database mirroring, will break your SLA!

Explain to the business what the implications of their choice of availability means. They will often start by expecting 100% availability on a 24×7 basis. It is essential that you explain to the business just exactly what that means and how much it will cost. If you agree to an SLA that cannot be met by the system as designed, and specified, then at the very least there will be a lot of friction between IT and the business process owners. At worst, it could have a huge impact on your business.

A bad SLA is a human error. It can be the result of a misunderstanding, a lack of willingness to speak up, or just plain old ignorance. In any case, it is an error and it will contribute to perceived downtime because the user’s expectations cannot be met. A sensible SLA involves explaining what is possible.

The Art of the Possible

These topics have been covered in detail in earlier chapter, but a quick refresh is in order. We need to control what is wanted by the business against what is actually needed and what can be afforded. A number of questions need to be answered:

- Will the systems support high availability?
- Is 24×7 really needed?
- Can we afford the cost of high availability and disaster recovery?
- Can we afford not to supply high availability and disaster recovery?

The dilemma is illustrated in Figure 4.2. The more wants, and needs, we pile on to one end of our balance, the bigger the pile of money we have to put on the other end to ensure everything remains in equilibrium (that is, to ensure we can afford to deliver what we are promising).

A system that attempts to deliver high availability without addressing these and the other questions raised in the previous chapters will fail expensively and usually at the most embarrassing moment. This comes back round to the question of design that was discussed earlier.

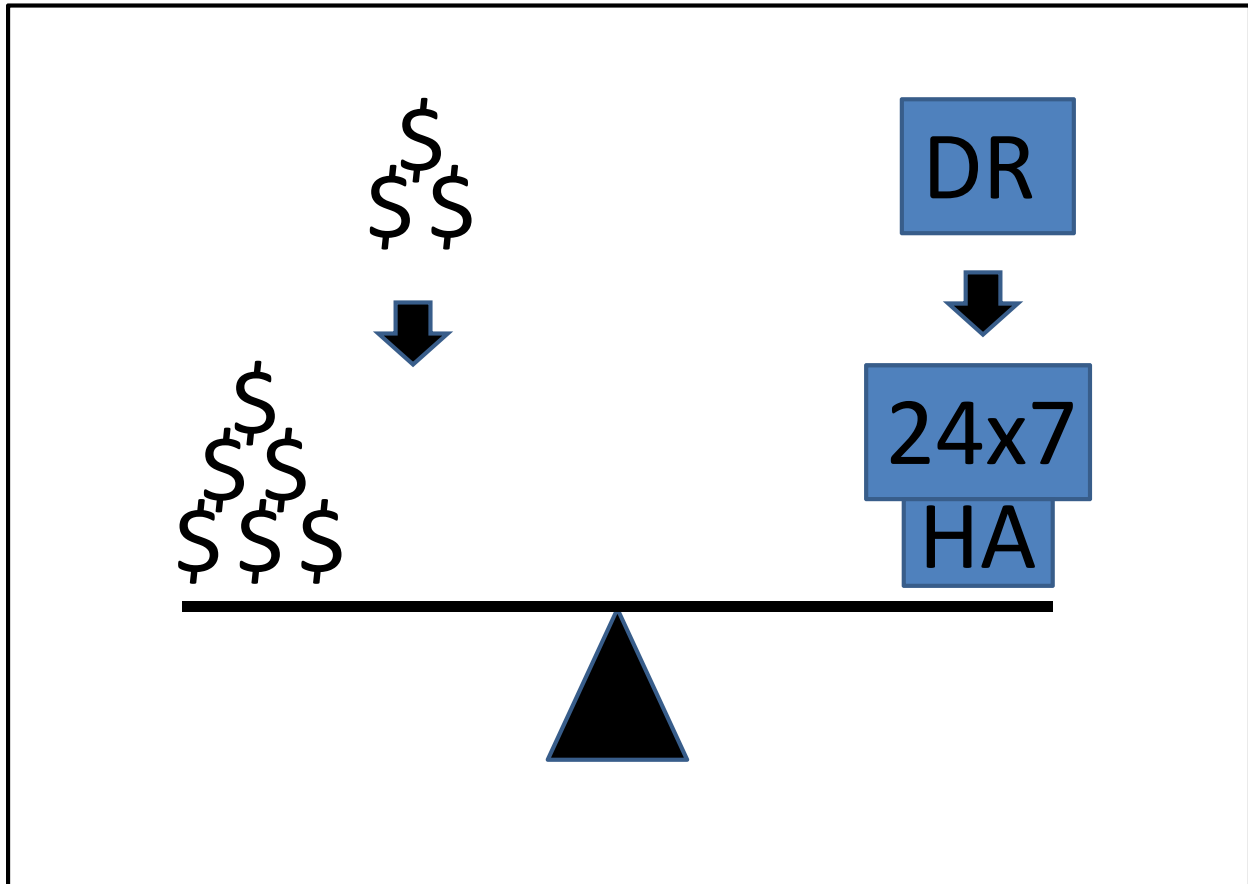


Figure 4.2: High availability desires and costs.

If the requirements aren't understood properly, the system won't be designed correctly and it will fail. A perceived failure because expectations weren't set properly or what was actually possible wasn't explained properly is still a failure. This is another case of a failure being due to human error because the system isn't doing what people think, or expect, it to be doing. Assuming we have the requirements, design, and build correct, we still have to ensure that the people involved in running our systems are capable of doing the job.

People

Administrators are often perceived as either the heroes who save the day when everything goes wrong or the greatest villains on earth for not keeping the systems running. A high-availability system depends on three things:

- Technology—Get the design right
- Process—Look after it properly
- People—The strongest and weakest link in the chain

People can still break the system even if we get the technology and processes right. We have seen how mistakes can be made by the best of people. If we have people who don't care, the probability of a mistake will rapidly rise.

The presence of good technical people in your admin teams is a good start, but they need the skills and knowledge to be able to do the job. Learning to administer a high-availability system with the “book on your knee” isn't going to create confidence in the end users. If they don't trust the admin team, it will be much harder to get things done. Trust can only be earned, in these situations, by doing a good job. The foundation to this is having the skills and knowledge, which means training.

Training

Training means different things to different people. There are numerous ways of receiving training including books, CBT, videos or Web casts, and instructor-led training. Many people will assume the last option is meant when they discuss training—“I can't do that because I haven't been on the course.” Make sure that a variety of methods are available and the administrators are matched with a training method that suits their learning method.

One big question is always “How can I measure the effectiveness of training?” In other words, how can we tell that the recipient of the training has learned anything? One way is to insist that the training lead to certification in the product. Are certifications worth anything? The answer should be yes if employees have worked for and obtained the certification by learning the material. Using brain dumps and other short cuts makes them meaningless. If you have been in IT for any length of time, you will undoubtedly meet a character who has a string of certifications but who can't perform the simplest of tasks or answer the most basic of questions.

In contrast, there are many very good people who don't have a single certificate. They have reached their level of expertise by using the technologies and researching answers to their problems as required.

Which is the best approach? It depends on the individual. Ideally, we want someone who can demonstrate their expertise by having the certificate and can actually use and understand the technologies. They also should have the necessary skills and background knowledge to fit into the administration team and contribute to solving general troubleshooting issues.

Skills and Knowledge

One of the first questions I ask as a consultant in a new situation is “Where is the documentation?” If I am really lucky, there is a nice big stack of documents for me to go through that will explain exactly how the systems work. Usually, the answer I get goes along these lines:

- We don’t have any—Honest but not helpful
- We have some but it hasn’t been kept up to date
- We didn’t create any because Joe installed the system and he knew everything about it—It then turns out that Joe left 3 years ago
- A variation on these, such as a third party that did the install

At this point, I know the first thing I have to do is document the system.

Let’s take this scenario and put it into the context of a high-availability system. The system is designed, built, and implemented. An administration team is set up and they do a good job of keeping the system running. One step that was missed in the implementation process was the production of documentation. It was decided that it wasn’t needed because the team that created the system would be administering it, and there wasn’t really the time to write the documents.

Over time, the members of the team move on. They may leave the organization or move on to other tasks. New people join the team and can only learn from the team members. The level of skills and knowledge regarding that system will be eroded over time until it cannot be supported. It will move from a well-designed high-availability system to a liability.

This leads to rule number one *Document your systems*, and rule number two *Keep your documentation up to date*. If these two rules are ignored, the consequences will be as bad as outlined earlier. We make things worse by extrapolating into a disaster recovery scenario where we don’t have any documentation. A business-critical system is down and we don’t understand how it works or how to get it back. It’s going to be a painful situation.

A situation to avoid is where people think “knowledge is power.” One, or more, individuals keep vital information, or tasks, to themselves. This is sometimes an attempt at self-preservation by making themselves indispensable.

Any situation in which knowledge is not freely shared will remove your chances of having a high-availability system. The individual with the knowledge will not be available, and a problem that should have been easily rectified becomes a major period of downtime.

A similar situation happens when individuals regard the system as their personal domain and no one else is allowed to participate. If the particular individual isn’t available, everything stops working and your systems aren’t available.

These particular situations are a management problem to ensure that they don’t happen. Management at various levels in the organization can have a significant impact, good or bad, on systems availability.

Management

Managers are under pressure to get things done. This includes changes to systems. A high-availability system can be helped or hindered by the way management decisions are implemented. This is not just an issue for the managers immediately responsible for the administration teams but extends throughout the management structure of the whole organization.

If decisions regarding high-availability systems are taken by management in isolation without consulting the correct people, there is a very good chance the decisions will not be good and will negatively impact the system, the users, and the business. Who are the correct people to be involved? At the very least, the following should be considered for any major decisions:

- Business process owner(s)
- End user representatives
- Technical support
- Technical design authorities

Decision making in a vacuum will cause problems. Decisions that override the controls and processes that are in place to protect the system can have just as big an effect.

It has been a constant theme throughout this book that high availability is dependent on technology, processes, and people. If decisions are made that override those processes, the model is broken and high availability will disappear. This could be as simple as performing a change without going through the change process or the business attempting to introduce a version upgrade with the vendor without consulting IT.

Another source of problems can be a change in management. The introduction of new people into the high-availability scenario can cause the processes to break down if they are not fully briefed on the processes and the importance of the system. The concept of “Just Do It” for instance does not belong in a high-availability environment but could be introduced by new people. This would override the processes and controls that have protected the system, eventually leading to a loss of availability.

Education

There is a requirement for a general education effort across the organization to explain the concepts of high availability in business terms and the impact to the business if the system was not available. Any business-critical system should be subject to a service review. This is a formal meeting between those responsible for administering the system and the business process owners. These meetings are a method of communicating what is happening with the system to ensure that there is a full understanding of the situation.

A valuable addition to this is to create a user community for the system. Service review meetings, and other formal occasions, tend to involve the management teams. There is a need to establish contact, and communication, at other levels. A method of communication between the people directly administering the system and those directly using the system can be beneficial.

Do your users understand how to use the business-critical systems? If they don't, there is a large potential for a situation developing in which the users break the system through ignorance. We have discussed the implications of administrators not having the skills and knowledge to perform their roles properly. The same statements are true for the user community.

A well-written application should guard against user errors, but what if the system was acquired from a third party? Are all the potential user errors trapped and dealt with properly? One of the steps that is needed when a new system is introduced is training for the users. They need to know how to use the system. This training must be kept up to date.

No system is static. There are changes to the system itself, the administration processes, and the people associated with the system. These people can be administrators or users. Ensuring that their knowledge of the system is up to date will prevent problems. How are new users trained on the system? If they are just given access to it and left to learn on their own, it is almost guaranteed that one of them will try something they are not supposed to and break the availability.

People are the biggest risk to the availability of our systems. Mistakes, inquisitiveness, and lack of knowledge and understanding will all contribute to the risk of the system becoming unavailable. This section has shown how we can work with the people to protect the system. Technology can add a further layer of protection. The processes we adopt, and how they are implemented, form the third layer of protection. None of these layers are more important than the others. We need all of them to be in place to fully protect our availability.

Process

Good processes are the unsung heroes of high availability. The technical community will praise the hardware and software; managers will praise the people, but no-one wants to champion the process! The heart of good processes, and many of the other concepts covered in this chapter, is good documentation.

Documentation

How many technical staff actually enjoy producing documentation? It is often produced at the end of a project when people are looking forward to their next challenge. "We've delivered this now what's next?" What's next is that we have to produce the documentation.

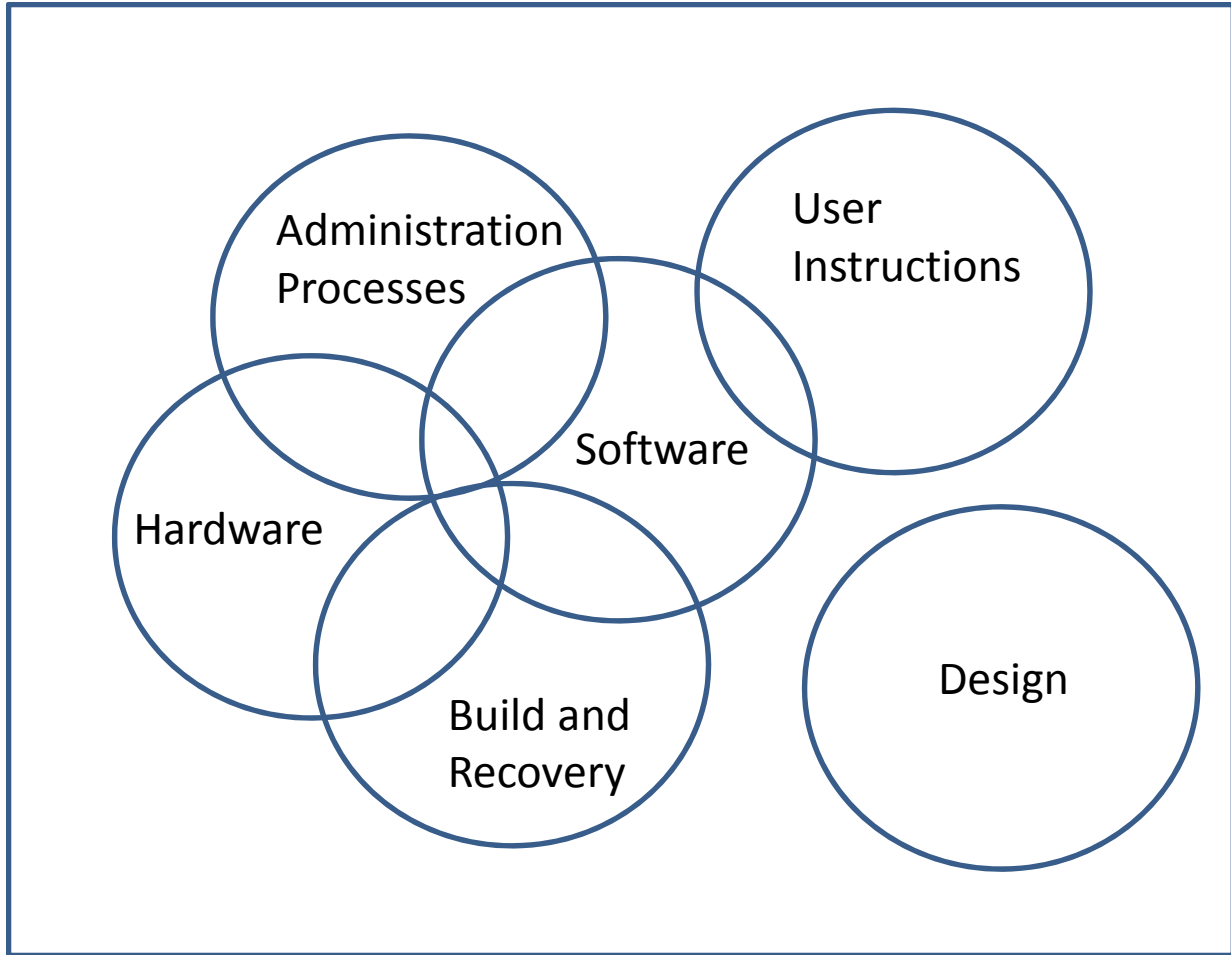


Figure 4.3: Documentation fields.

The documentation should be part of the project deliverables, but it can't be written by the project manager and "thrown over the wall." It needs to be written in conjunction with the people who will be using it to ensure that it delivers the information they need.

Note

The documentation areas we are discussing are in addition to the design documentation we thought about earlier (see Figure 4.3). Systems need documentation. It doesn't have to be stored on paper, but it needs to be available.

Before we discuss the documentation in detail, there are a few questions to consider:

- How will you publish the documentation?
- Should it have a restricted circulation? If so, who approves the readers and how is this managed?
- How are the documents kept up to date?
 - Periodic review?
 - As the system is changed?
 - Post audit?
- How are changes to the documentation communicated?
- What version control is applied to the documents?
- How is the existence of the documentation publicized?
- How do you ensure it is read, understood, and applied by the people who need it?

The last two points are vital. If no one knows the documentation exists, it won't be used. If it's not used, mistakes will be made and availability will suffer. This is another example of people inducing downtime. It may not be a direct cause, but it will be the root cause. Figure 4.3 shows the areas to consider. Remember that these are in addition to the extensive design documentation our project has already produced.

System and application documentation describe how the system works. This may start as an excerpt from the design documentation, but it will also include detailed configuration information. It should also include administrator-level passwords where necessary. I once had the enjoyable task of explaining to an IT Security Manager why we had to use a password-cracking tool on one of his servers. The machine had been taken out of the domain and we then discovered that the local administrator password had been changed and no one in the organization knew what it was. This is not the situation to be in when higher management is looking over your shoulder while you are attempting to restore the system.

We also need full instructions on how to build and recover the system. Ideally, these processes will have been tested so that we know they work.

The second major set of documentation we require is for the day-to-day processes. Full instructions on performing the administration processes will have to be produced. These need to include the processes to be followed in the event of a disaster and which, if any, of the day-to-day processes can be relaxed or eliminated in a disaster recovery situation. User instructions need to be included in the documentation pack. Any variations introduced by invoking disaster recovery should also be documented.

There is one last point to consider regarding documentation. Have you included it in your disaster recovery plans? Do you need any of this documentation to get systems back up and running in the event of a disaster? Is the documentation available off site? It's not going to be available if it's caught up in the disaster.

Don't rely on restoring from backup to access the documentation. There will be too much pressure to get the systems up. The documents need to be available electronically. In addition, they must be in a format that is readable. Servers shouldn't have personal productivity software such as word processors installed. Are there some workstations available for the administrators or will there be arguments with the users? An important point to remember for your disaster recovery plans: include workstations and associated software for the administrators who need to restore the systems.

Change Control

Change control is a subject that seems guaranteed to start arguments. The subject has a habit of polarizing opinion between the "we don't need it" and the "everything is under change control" ends of the spectrum. As with any process, there is the correct level of change control to apply to your environment. Would you want to upgrade your domain controllers without having formal agreement to the change?

A high-availability environment needs a greater degree of change control than less-critical systems. This can often be achieved by removing some, or all, of the standard changes that are allowed in the rest of the environment.

An organization that has embraced change control will have little difficulty in adapting their processes to deal with a business-critical high-availability system. It is the organization that hasn't utilized change control that will have problems. How do you introduce change control?

One method is to start by defining what standard changes are allowed (that is, those changes that are routine and get automatic approval). An example is creating a new user in Active Directory (AD). At the other end of the scale are those changes that definitely need to go through a formal change control process, such as applying a service pack to the operating system (OS) of the cluster nodes supporting the database of your business-critical application. Everything in between is open to debate depending on your circumstances and needs.

Change control should be viewed as providing protection for the people making changes. If a change has been approved and something goes wrong, the technician making the change is protected because he is following the agreed process. Imagine the scenario where you apply a patch to a Windows server. This is a fairly common occurrence in most organizations. The server reboots after the patch application, which again is common. Now suppose the unexpected happens and the server suffers a disk failure, or even two. If you are making an unexpected and unapproved change, expect a lot of questions and interest in your activities. If your change is approved, the event is an unexpected occurrence as you were following the agreed procedure.

Explaining these scenarios can go a long way towards overcoming the “change control is a blocker” view. It removes the ad hoc nature of administration and ensures that processes are followed. What happens when someone steps outside of the change control process and does something they shouldn’t? A lot will depend on circumstances, but there needs to be a procedure to enforce the fact that change control is not to be sidestepped. If you are not seen to mean that you will take disciplinary action, as appropriate, your efforts to introduce change control will fail.

Any process needs auditing. In the case of change control, the process itself needs auditing to ensure that it is being followed. The changes need to be audited to ensure that the correct changes are being performed and that other changes aren’t creeping into the environment. One area where we need to consider change control is when thinking about maintenance cycles for servers.

Maintenance

The primary maintenance activity we need to consider is patching. Many organizations have a policy of not patching on the principle that the systems are working and it takes a lot of effort to patch a server estate. This approach works until an exploit appears that would have been stopped by a patch that was available a number of months ago. A decision not to patch will definitely seem like a mistake at that point.

You need to have a patching policy that enables the application of at least critical patches in a sensible timeframe. This should include testing the patches. Applying untested patches may be worse than not applying any patches.

In many cases, applying patches requires a restart of the system. This is where the maintenance window you negotiated into your SLA comes to the rescue. You have the agreed and scheduled time to apply the patches and reboot the machine.

The other area that is often affected by maintenance is the data itself. The obvious requirement to maintain the data is to perform a backup. Don’t forget to test a restore on a periodic basis. Databases also need maintenance in the form of index rebuilding and re-organization.

Loading and pruning data can be a significant maintenance activity. If data is allowed to grow unchecked, the database could become too big for the disk and will certainly become bloated with un-needed data. These effects will have an adverse effect on availability. Making mistakes while performing these activities could also affect availability, possibly in a more drastic manner if a restore is required.

Data archiving is the ultimate data pruning activity and can be considered an activity that keeps your systems available by helping performance. It also contributes to disaster recovery by having historic data available and reducing the time to restore the system.

Automation

Many of the topics we have discussed in this chapter involve performing actions on the system. We have already seen that people make mistakes that can contribute to downtime. A method is required to automate these procedures. This will ensure that the activity is scheduled and not forgotten, it is repeatable, and it prevents some errors because typing and other manual intervention is not required.

Automation is not guaranteed to solve all your problems introduced by people. The procedures used to perform these automated tasks have to be created and tested. People will do this. It has been said that “automation is a way to make mistakes faster.” Ensure that any automation procedures are tested so that such isn’t the case.

Monitoring is another topic we touched on earlier. It is possible for an automated response to be generated from monitoring systems. This builds on the vendor’s knowledge of the system and applies that experience automatically. In some cases, this takes the system well down the path of being self-healing.

Applying High-Availability Principles to Disaster Recovery

High availability and disaster recovery are two very different concepts, as we have seen through the course of this book. They are, however, related in that the same techniques, and technologies, can be applied to both. It is time to come full circle and consider how the high-availability principles we have worked through can be applied to disaster recovery.

Mistakes are a big factor in causing downtime. Those mistakes may be due to someone doing the wrong thing or a larger mistake at the organizational level that didn’t put the correct processes in place. Disaster recovery situations tend not to be triggered by mistakes, but mistakes early in the recovery process can have a huge impact on the time taken to restore the organization to a working, and survivable, position.

The biggest mistake is to dive straight into the restore activity. Do you understand the scale of the disaster? Has anything survived that is usable? Do you have all the equipment, media, and people to perform the restores? Just as we plan to keep our high-availability environments running, we need to plan to perform a recovery.

Document

The central part of the recovery process is to plan, and document, every step. Circumstances may enable you to skip some steps because by luck a particular server survived. Make sure that the integration of that server into the recovery happens at the right point in the sequence. Don’t be tempted to jump ahead just because it’s there.

The documentation of your systems is just as essential for disaster recovery as for high availability. If you don’t know how your systems are built and configured, how are you going to restore them? Keeping the documentation up to date is just as important. If the live system has drifted from the documented state, it may not come back in a usable condition.

Where are you going to store your documentation? Multiple copies will be required that are positioned off site. Use replication techniques to automatically update the documents from the central repository.

Test

Once you have your documentation and recovery procedures, make sure they work. Test the procedures and the documentation. Get someone who hasn't worked on the system or produced the documentation to perform a restore based solely on what is documented. When that can occur, you know you are in a position to recover the system if you have to invoke the disaster recovery procedures.

Automate

Automation is a way to reduce mistakes by ensuring that the same steps are always performed. It makes sense to apply this to a high-availability environment as it reduces mistakes. There is even more sense to applying it to a recovery situation.

If you have automated some, or preferably most, of your recovery procedures, it becomes a simple matter to tell someone "run these scripts in this order." They don't need full understanding of what they are doing, and your recovery can proceed faster and more accurately as fewer mistakes are made. Introducing automation into the recovery process reduces the need for specific people with specific skills to be available. They can set up the procedures and whoever is available can run them.

The ultimate goal is for high availability and disaster recovery to be combined at the technology level so that the system recognizes there is a problem with your primary system and automatically switches to the "stand by" without anyone noticing. It would be nice to be notified afterwards so that the primary can be fixed.

Summary: Eliminate Downtime Before It Eliminates You

Downtime costs money that organizations cannot afford, especially in the current economic climate. The biggest cause of downtime has been identified as people—either through mistakes or bad processes. Identify these potential areas of downtime and eliminate them. Train the people, write and use the processes, and ensure that change control is viewed as having a vital place in protecting your environment.

Educate IT, users, and the business at large that high availability comes with a price. There is a monetary cost to implementing the system, but there is also the cost of investing the time and effort in ensuring the correct procedures are in place and followed.

Automate everything that you can. Don't put blind faith in your automation, but use people to do what they do best: analyze, think, and solve problems.

The final action is to make sure that everything is reviewed on a regular basis. Keep your documentation and procedures up to date and don't forget to include your disaster recovery planning in those reviews.

Download Additional Books from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this book to be informative, we encourage you to download more of our industry-leading technology books and video guides at Realtime Nexus. Please visit <http://nexus.realtimepublishers.com>.