

# THE MANAGEMENT OF SERVICE AVAILABILITY >>

## Keeping services running without breaking the bank

Written by Keith Inight

IT based services are intrinsic to every aspect of daily life and we take them for granted – so much so that when they don't work it is a big issue for the service provider. Service providers have responded to these expectations by trying to improve availability – the measure of how reliable a service is – typically by investing heavily in redundant hardware and network links. Experience has shown however that the actual causes of unavailability are frequently nothing to do with hardware and networking, and more to do with human factors and software issues. This should raise alarm bells in every CIO's mind – am I spending money wisely and will it actually deliver the availability that the business is expecting?

This paper starts by looking at business requirements and how to assess an appropriate level of availability for a service. Getting this right is important – the costs can triple for a relatively modest improvement in availability.

The paper then draws on original Atos Origin research into the root causes of unavailability and the things that could have prevented it. It builds upon the unsurprising but often ignored fact that defective processes and human error cause far more problems than hardware failure, and proposes pragmatic solutions.

It is our view that availability can be engineered and that simple models can be used to predict the availability of a service and ensure that investment is targeted appropriately. The paper shows how modelling can be used to analyse a service and present alternative scenarios.

Throughout the paper it is acknowledged that different services have different availability requirements and that a proportionate approach is needed. This is depicted graphically showing the levels of availability at which each approach becomes viable.

### Contents

» What does availability mean?	2
» What availability do you need?	4
» Availability in complex modern environments	4
» Availability in legacy environments	6
» The causes of unplanned downtime – and how to fix them	6
» Planned downtime – reduction by design	13
» What are the options?	14
» Summary	15
» Acknowledgements	15
» Appendix – modelling availability	16

## WHAT DOES AVAILABILITY MEAN?

Services take many forms – online hotel reservations, ATM transactions and the corporate finance system being just three examples. Service availability<sup>1</sup> is concerned with the proportion of time that an IT service is available for use by its users<sup>2</sup>.

IT management seeks to minimise day to day availability issues with each service provided from a data centre. By contrast Business Continuity and Disaster Recovery planning seek to address bigger problems that impact the entire organisation<sup>3</sup>.

### Atos Origin standard definition of availability

Service availability is a measure of the fraction of time during the Service Availability Window (SAW) when the service provided is deemed to be better than the Quality of Service (QoS) threshold. Service availability is expressed as a percentage to indicate the time during which the contracted service at the defined Service Access Point(s) is operational. Operational means that the customer has the ability to use the service as specified in the contract. Service availability is measured at the defined Service Access Point(s).

Availability is normally quantified as a percentage – typically the proportion of the year that the service is available (uptime – as opposed to downtime when it is not working). It is illustrative to calculate what this means in downtime per year. 99% sounds quite reasonable until it is realised that this could potentially imply three consecutive days of downtime.

Conversely so called '5 nines' (99.999%, equating to five minutes a year or less) is often quoted – whereas in reality many services can tolerate an hour or two downtime a year, and make very significant savings.

## Downtime per year<sup>4</sup>

Availability	24 hours a day 7 days a week service	16 hours a day 5 days a week service	10 hours a day 5 days a week service
90%	36.5 days	26 days	26 days
95%	18.25 days	13 days	13 days
98%	7.3 days	5.2 days	5.2 days
99%	3.65 days	2.6 days	2.6 days
99.5%	43.8 hours	20.8 hours	13 hours
99.9%	8.76 hours	4.16 hours	2.6 hours
99.95%	4.38 hours	2.08 hours	1.3 hours
99.99%	52.56 minutes	25 minutes	15.6 minutes
99.999%	5.26 minutes	2.5 minutes	1.56 minutes

Service providers distinguish between planned downtime (when they are allowed to bring services down for essential maintenance) and unplanned downtime (when something unexpected causes the service to stop working). This distinction is important for contractual reasons but of course users don't care – they just know that the service isn't working. The impact of planned downtime can of course be minimised by scheduling it to occur at quiet times. The following table gives an idea of typical levels of service availability that are actually achieved:

### Benchmarking availability – hours downtime/year/IT service

	Unplanned	Planned
Very good	Less than 61 hours (99.3%)	Less than 200 hours
Outstanding	Less than 26 hours (99.7%)	Less than 50 hours
Best-in-class	Less than 5 hours (99.95%)	Less than 12 hours

Source: Gartner Data Center Conference December 2008 – Donna Scott & Bill Malik – Best Practices for Continuous Application Availability

<sup>1</sup> It is acknowledged that some 'services' contain a human element – requiring some input. This paper focuses only on the IT-related aspects of availability including the network that delivers the service to the point of use.

<sup>2</sup> Mathematically the availability of a service is defined as:

$$\text{Availability} = 100 \times \frac{\text{Service Window} - \text{Planned Downtime} - \text{Unplanned Downtime}}{\text{Service Window}}$$

Downtime is the period of time during which the service does not perform according to the defined performance criteria.

<sup>3</sup> Business Continuity and Disaster Recovery are not discussed further in this paper.

<sup>4</sup> Note that downtime only impacts availability if it is during the service window. So for a 10\*5 system downtime during the weekend would not impact availability.

## 2 The management of service availability



# REDUCING DOWNTIME COSTS MONEY

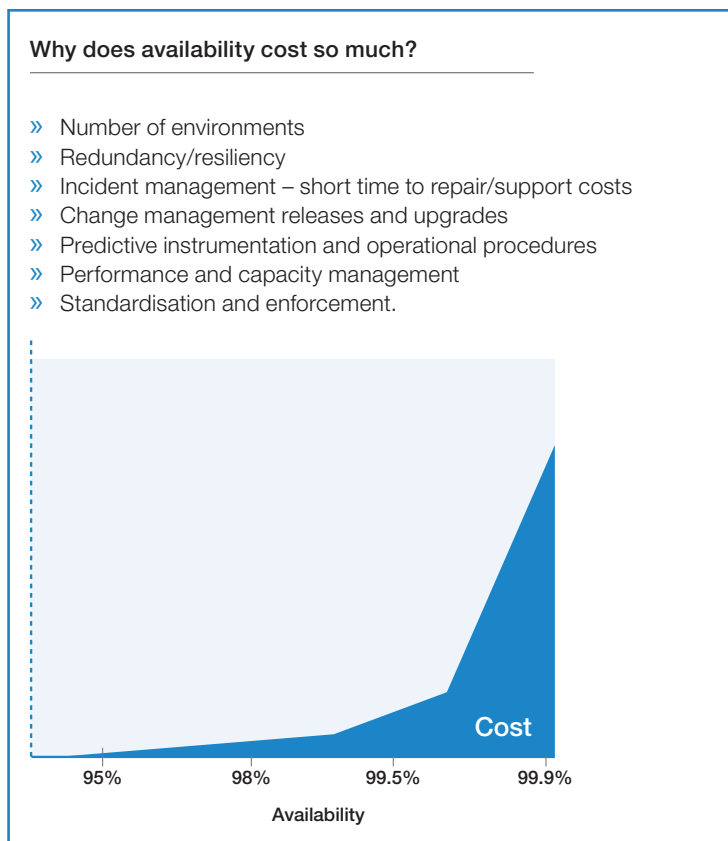
So why do service providers not seek to be best-in-class for each of the services they provide? Quite simply because reducing downtime costs money – lots of it. The costs rise exponentially the nearer that one gets to 100% which is in reality unattainable.

This extra cost is spread across all the components of delivering the service – hardware, software, support, data centre etc.

Typical areas of investment to achieve higher levels of availability are shown in the following graph:

It should be recognised that frequent short periods of downtime have a disproportionate impact – in one day a single period of an hour will have far less impact than 20 periods of three minutes. Frequent periods of downtime also create a perception of unreliability.

When discussing availability there is an implicit assumption that downtime is a rare occurrence – indeed service provider contracts often stipulate this. That is, indeed, generally the case; in contrast to the early days of IT contemporary systems rarely fail. But because of the increasing dependency on those systems, the exceptions do need to be managed carefully.



Source: Gartner Data Center Conference December 2008, Donna Scott & Bill Malik, Best Practices for Continuous Application Availability



**European Central Bank requirements for Systemically Important Payment Systems**

“...SIPS should aim to recover and resume critical functions or services... no later than two hours after the occurrence of a disruption.”

**WHAT AVAILABILITY DO YOU NEED?**

Having seen the profound effect that availability has on cost it is clearly important to establish the appropriate level of availability required for a service. To assess this a risk management process should be used. The extra costs that are accepted to increase availability can be seen as insurance against the risk of the service not being available.

The first step is to identify the major business risks that are driven by the IT service. The consequences, and financial impact, of each risk materialising then needs to be determined.

**Assessing the business consequences of unavailability**

Risk	Consequences	Impact	Likelihood		
			99% Availability	99.5% Availability	99.9% Availability
Service unavailable during December	Loss of sales	500,000/hour	1:1200	1:2400	1:12000
Service unavailable during two day year-end accounting period	Missing deadline for filing year-end accounts – reputation and share price impact	10,000,000	1:18250	1:36500	1:182500

It is then possible to balance the potential losses against the investment in IT needed to reduce the likelihood of it happening. It should be noted that it is impossible to completely mitigate these risks (the ‘residual risk’ remains) and that is why Business Continuity plans are needed. Risk Analysis and Business Continuity planning are not only best practice they are also a requirement in many industries such as finance.

Whilst the risks will be different in every case the typical availability requirements are likely to fall in the following ranges.

Type of service	Typical availability requirement
File and print	99.5 to 99.9%
Email	99.3 to 99.7%
Finance ERP (except at year end)	99.5 to 99.7%
Bank front office system	99.9 to 99.95%
Transportation real-time resource management	99.98 to 99.99%
High profile website	99.99% +

The service window will be determined by asking questions such as:

- » Does the business operate during the night?
- » Does the business have a global presence and need ‘follow the sun’ operations?
- » Does the business care about downtime if it is out of normal hours?
- » Does the business really need 24 x 7, or is this an aspiration because it’s ‘the best’?

**AVAILABILITY IN COMPLEX MODERN ENVIRONMENTS**

The fundamental architectures used for delivering systems have evolved radically from the centralised monolithic mainframe through the multi-tier client-server approach to modern Service Oriented Architectures (SOA). The key issue from an availability perspective is the proliferation of autonomous applications used to craft a business service and the links between them.

# IN AN SOA ENVIRONMENT, UNLIKE LEGACY ENVIRONMENTS, SOFTWARE 'APPLICATION FAILURE' HAS GROWN TO BE THE DOMINANT CAUSE OF DOWNTIME

There are a number of factors at work here:

## Process control and monitoring issues:

- » Applications are part of increasingly complex workflows involving multiple independent processes. Hence although the individual processes calculate data correctly the answers can still be wrong because of delays and timing issues with the messages between processes.
- » There are increasing pressures to implement software changes without comprehensive testing because of time to market issues. This can have disastrous consequences such as the Moody's incident, below, which resulted in over \$2 billion reduction in their market capitalisation.

## The increased complexity of service monitoring.

It is not enough to monitor the health of individual pieces of infrastructure. There is no longer any alternative but to continually measure the end-user experience (availability and response time). Furthermore it is important to perform basic checks to ensure that a valid response rather than an error message is returned. Modern tools<sup>5</sup> are now capable of monitoring the traffic passing across the network switch at the access point to the data centre and checking every transaction. Furthermore by analysing network traffic they can accurately assess the actual end-user experience by adding the network latency to the elapsed time within the data centre. Previously this could only be accomplished by remote monitoring devices submitting 'synthetic transactions' – an approach that was expensive to deploy and manage.

## It is increasingly difficult to schedule downtime

in order to upgrade software and hardware. Perversely with these modern complex and dynamic environments upgrades are needed more frequently than in the past due to business pressures such as globalisation.

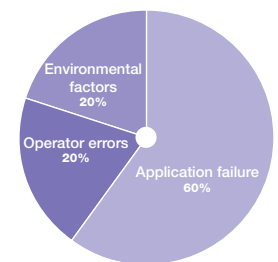
## An increasing tendency to use software

**components from third parties** (with their own objectives, release cycles and priorities) increases integration risks and also makes holistic testing difficult – there is no real overall architecture, design and governance. Gartner's Strategic Planning Assumption is that "Through 2010, 60% of unplanned downtime for SOA-based, loosely coupled applications will be due to application failure, up from 40% on non-SOA based applications."<sup>6</sup>

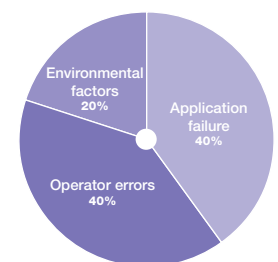
Why is this?

- » There are a lot more components. Even though each component is reasonably reliable the overall reliability drops because any one failing will cause the overall service to suffer downtime. Put simply if there are ten times as many things to go wrong then the service will go wrong ten times more often.
- » It is more complex to manage the development and operation of the service as many more people and organisations are involved.
- » Many of the components will inevitably use complex and relatively immature technology.
- » The focus of most modern environments is upon improved functionality and agility rather than high availability: the addition of ever more features to increase sales has led to larger and more complex products (bloatware), meaning there is more to fail.
- » Availability of skills. Writing software code in a highly available environment is a special skill that is rare. It also requires more development time and more testing. This means that project management for a highly available application also requires special skills, which are also rare. So business pressure on functional improvements aside, development teams often fail to understand what they need to do to maintain availability. That also means there are fewer people with the knowledge and competence to reinstate failing systems: although Mean Time Between Failures (MTBF) may be improved, when things do go wrong the Mean Time To Repair (MTTR) may be greatly elongated.

Unplanned downtime for SOA environments



Unplanned downtime for non-SOA environments



Source: Gartner Data Center Conference December 2008, Milind Govekar, Why Bother Managing SOA Applications?

## Software bug in Moody's ratings

The share price in Moody's (the well known ratings agency) dropped rapidly on 21 May 2008 and it never subsequently recovered its previous value. Exceptionally high trading in their shares was driven by widespread media coverage of the consequences of a software bug in one of their mathematical models. This error caused a financial product to be given a much higher rating than it merited. The error was apparently discovered in 2007 but did not become public knowledge until a year later.

<sup>5</sup> For example Compuware Agentless Vantage.

<sup>6</sup> Source: Gartner Data Center Conference December 2008, Milind Govekar, Why Bother Managing SOA Applications?

## AVAILABILITY IN LEGACY ENVIRONMENTS

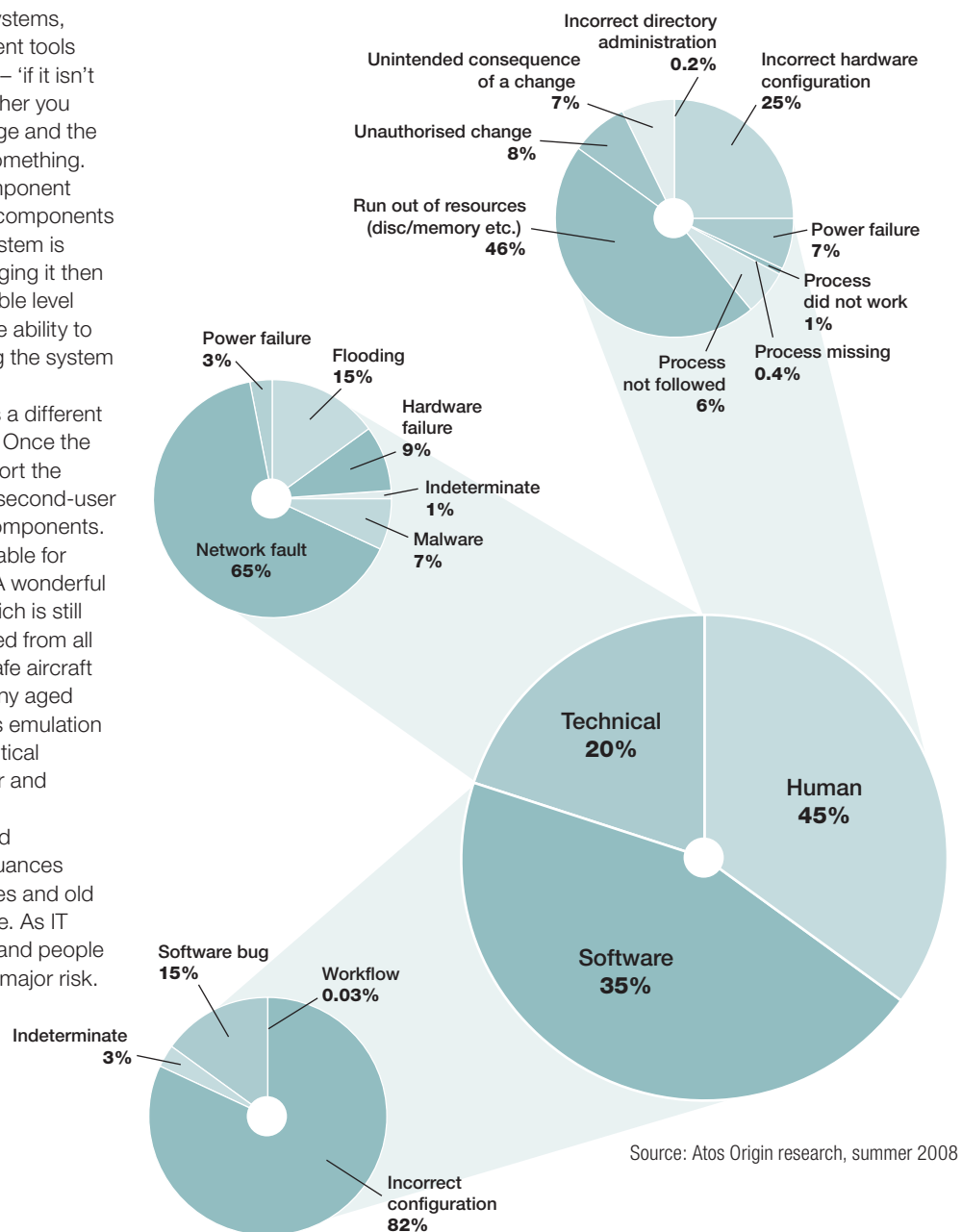
There is more than a little truth in the old joke that the definition of legacy software is stuff that has been fully debugged. The fact is that in many environments legacy systems are business critical and delivering high availability. The continued success of systems running on IBM mainframes exemplifies this fact.

The issues with some legacy environments tend to be:

- » **Unsupported software** – operating systems, databases, middleware and development tools etc. The temptation is not to touch this – ‘if it isn’t broken don’t fix it’, which is fine until either you get a fault or you need to make a change and the change requires the latest version of something. Moving to the latest version of one component usually precipitates upgrades to other components for compatibility reasons. Hence if a system is stable and there is no intention of changing it then it is possible to maintain it at a reasonable level of availability – the key is to maintain the ability to revert to a previous version by restoring the system from a backup.
- » **Unsupported hardware** – hardware is a different problem – unlike software it wears out. Once the main vendors have taken it out of support the options are to depend upon specialist second-user brokers or to maintain a set of spare components. In practice this approach is perfectly viable for many years – albeit not best practice. A wonderful analogy is the wartime DC3 aircraft which is still in service today – using parts scavenged from all over the world. A fundamentally very safe aircraft still delivers reliable service – as do many aged computer servers. Another approach is emulation – providing what appears to be an identical environment within a modern computer and operating system.
- » **Availability of skills** – knowledge of old application components, application nuances and configuration, old database releases and old operating systems can be lost over time. As IT operations focuses on newer versions and people move jobs or retire, this can become a major risk.

## THE CAUSES OF UNPLANNED DOWNTIME – AND HOW TO FIX THEM

We conducted an international study in 2008 to analyse the root cause of a large sample of major incidents that caused downtime – and also to establish what investments would have prevented each incident. We consider that our results are consistent with the Gartner findings in the pie charts on page 5, in particular that technical/environmental factors only account for 20% of incidents.

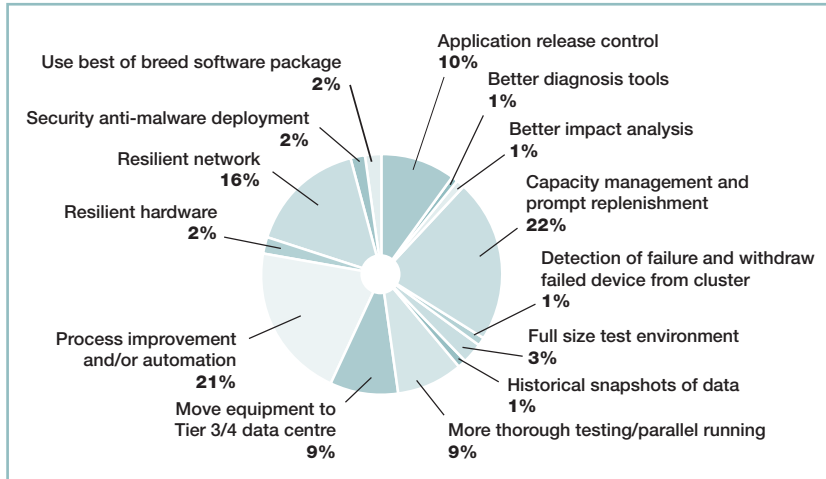


Source: Atos Origin research, summer 2008

# A HOLISTIC AND BALANCED VIEW NEEDS TO BE TAKEN TO DESIGNING FOR AVAILABILITY

## Preventing incidents

Our study identified for each incident what measure would have prevented it happening.



Source: Atos Origin research, summer 2008, Incident Prevention

It should be emphasised at this point that a holistic and balanced view needs to be taken to designing for availability – addressing all of the following points where appropriate. In particular when very high (over 99.95%) availability is needed this should result in specialised techniques being used at the application level to support load balanced active/active architectures. Taking the three groups in order:

### Technical

Technical is sometimes referred to as ‘other’ or ‘environmental factors’. The essence of any solution is to have an identical unit to the failed one available to take over (more or less) immediately. Increased availability comes from duplicating everything and leaving no ‘single points of failure’.

*Data centre infrastructure* provides many essential facilities such as power, cooling and networking. It can also suffer incidents such as localised fires and flooding within the machine room. Reliability is improved by duplicating facilities – either completely or by providing one spare for every ‘n’ active units (so called ‘n+1’). The resilience of a data centre follows from the Uptime Institute definitions.

### Tier requirements summary

	Tier I	Tier II	Tier III	Tier IV
Active capacity components to support IT load	N	N+1	N+1	N after any failure
Distribution paths	1	1	1 active and 1 alternate	2 simultaneously active
Concurrently maintainable	No	No	Yes	Yes
Fault tolerance (single event)	No	No	No	Yes
Compartmentalisation	No	No	No	Yes
Continuous Cooling*	Load density dependent	Load density dependent	Load density dependent	Yes (Class A)

\*For additional information on Continuous Cooling, refer to the Uptime Institute white paper, ‘Continuous Cooling Is Required For Continuous Availability’.

Source: © 2008 published by the Uptime Institute. Reprinted with permission.

Twinning is an approach that implements mirrored storage and load balanced clusters of servers where the equipment is split between two sites – albeit close enough to be on the same Local Area Network (LAN) and Storage Area Network (SAN). Hence in the event of a system failure at one site the service will carry on with reduced capacity. Note that this is not normally considered a Disaster Recovery solution; the separation needed to be on the same SAN is not enough to prevent the same disaster impacting both sites. If required a third site provides a Disaster Recovery capability.

Server resilience can be provided in four ways:

- » *Failover clustering* – where a normally inactive server takes over processing when the primary server fails. Whilst superficially attractive this can introduce problems. The issues are either false invocation (when the primary server is still functioning) or failure to invoke failover correctly. In practice a full set of equipment is needed in development in order to do proper testing of the various failure modes and prevent invocation failures. Even then it is difficult to simulate all of the various types of failure that could happen. Another key factor is rigorous change management, so that any change made to the primary systems is also applied, in a controlled manner, to the secondary. Standardisation of the clustering approach (buying as a pre-configured and pre-tested package) is a way of minimising these issues (e.g. SAP).
- » *Load balancing* – an inherently easier approach is to have multiple active servers and share work between them. When one device fails all the load just goes onto the remaining devices and does not require complex failure detection and repurposing.
- » *Specialised high availability servers* – these duplicate all hardware, and software is executed simultaneously on all processors. These servers can therefore survive an individual hardware fault with processing continuing uninterrupted on the surviving processor(s).
- » *Fast provisioning* – this is where an image of a running server is held on disk storage and, in the event that the server fails, it is automatically loaded and initiated on another server. In practice, this can be done quickly if there is a fast LAN.

*Network resilience.* The service is only any use if it reaches the end-user wherever they are. To do this the service has to transit one of the most unreliable pieces of infrastructure – the Wide Area Network (WAN). The WAN can fail for physical reasons or logical reasons. Physical risks can be offset by providing duplicate diversely routed cables that genuinely do not share any ducts. Logical risks are less easy to ameliorate and require careful design to limit the impact of an incident, mature software and configuration assurance to detect unauthorised configuration changes.

*Storage resilience.* This can normally survive a single disk failure as data is duplicated across other disks<sup>7</sup>. Enterprise disk devices also have complete duplication within the frame and are designed for all upgrades and maintenance to be completed without any downtime. This can be extended with virtualised storage across two data centres or by replicating data to another data centre.

*Database resilience* can be provided, for example, by using Oracle Real Application Clusters.

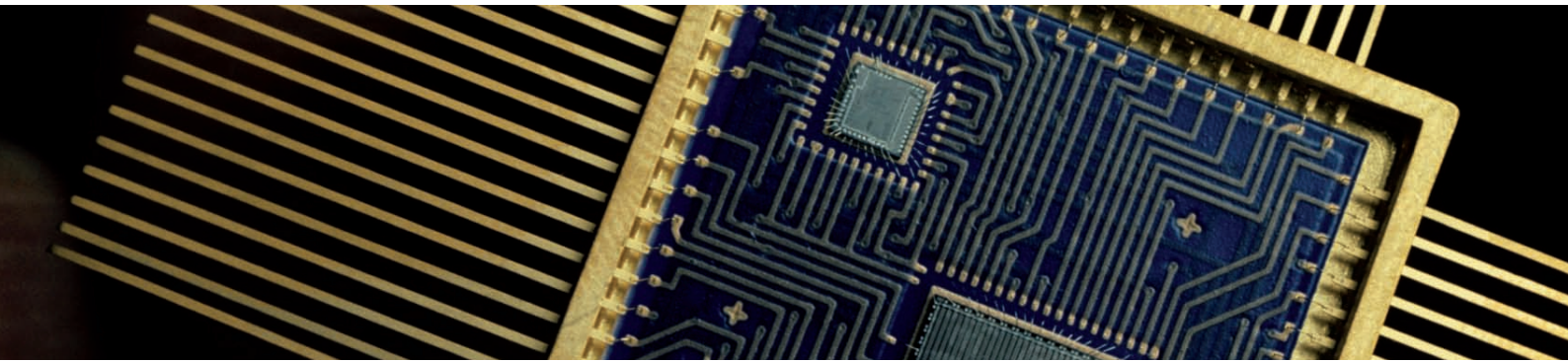
*Application server resilience.* For example JBoss has the ability to be clustered across multiple servers.

*Middleware resilience* can be provided. Taking BizTalk as an example it can either be clustered or alternatively redundant routing of messages across two BizTalk instances can be implemented.

<sup>7</sup> Typically using Redundant Array of Inexpensive Disks (RAID).



**HUMAN ERRORS OFTEN OCCUR AS A CONSEQUENCE OF MAKING SOME SORT OF CHANGE TO THE SYSTEM**



### Human factors

Human factors are sometimes referred to as 'operational' or 'operator errors'. Human errors often occur as a consequence of making some sort of change to the system – such as the notorious incident at the Tokyo Stock Exchange.

#### Tokyo Stock Exchange – system failure

The Tokyo Stock Exchange implemented an upgrade to their capacity on 1 November 2005 to cater for an increased volume of trading. Unfortunately this did not work out as planned – preventing trading for most of the day. The exchange reported that the cause was “part of a necessary step was missing in the instruction for the loading operations”. The exchange later announced that this, coupled with another unrelated incident on 8 December caused the resignation of the president of the exchange Takuo Tsurushima, along with Sadao Yoshino, who was in charge of the computer system.

- » Provisioning tools automate the whole process of deploying an application – potentially including software and patches to servers, storage and network connections.
- » Run Book Automation tools are an emerging class of tool that orchestrates an entire workflow. A typical workflow starts when a change request is created and includes provisioning the technology, updating the ticket and finally updating the Configuration Management Database (CMDB) at the end of the process.

**Information Technology Infrastructure Library – (ITIL)** identifies that the CMDB is a key element of maintaining a reliable infrastructure. Accurate information in a CMDB permits more effective impact analysis of a potential change and fewer incidents as a consequence.

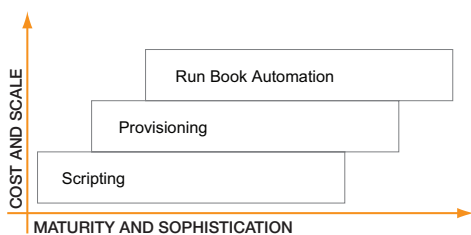
**Change control** – several principles have become established over time:

**Staff** – in some cases the solution is simply to recruit and retain enough suitably skilled and experienced staff.

**Automation** – often seen as the cure for problems caused by the operator, especially for regularly-performed tasks or those with complex inter-dependencies. It is not a panacea however – automation requires substantial investment and also regular testing – particularly when any environmental changes are made. The most common forms of automation are:

- » Scripting – this has been used for many years by system operators to mechanise tasks that they frequently perform. These often use free tools that are provided with operating systems.
- » Segregation of duties and in particular the person that authorises a change must not be the one that implements it. This ensures that there is an audit trail of changes that have been made and that quality processes have been followed, for example that a reversion plan exists.
- » Formal processes such as Change Advisory Boards can help provided that they are supported by good quality information so that decisions are rapid and effective. Unfortunately if the wrong people are appointed, or the process is used purely to syndicate the blame for something going wrong, the value is lost.
- » With any change a straightforward and bullet-proof reversion strategy is needed so that if things go wrong the service can be restored and the problem resolved later on.

### Automation maturity



**Capacity management** – or rather the lack of it – is a common cause of failure. Incidents occur because the supply of resources becomes exhausted – disk storage or memory for example. Good practice is to have a combination of formal capacity planning plus monitoring tools that detect when resources are close to their limits. Advanced approaches that automatically redeploy resources can help but only when the overall resource provision is adequate – which is an output of the formal planning process.

## Software

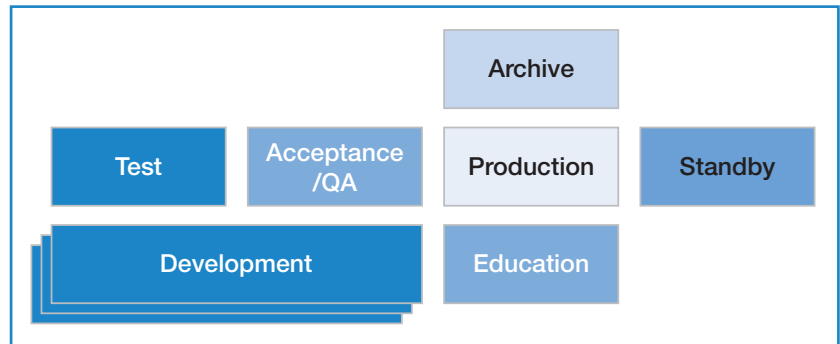
**Software testing** – expensive but a key requirement for delivering high availability:

- » Regression testing ensures that no new errors have been introduced (or old errors re-introduced) – typically by having an automated set of scripts to test all the functions of the system and validate that the results are the same as the previous version of software produced.
- » Volume testing reflects the fact that many problems only occur when the system is under stress. Some systems (notably databases) do not degrade linearly with load and so need to be tested at, or beyond, the maximum expected demand. Similarly Java engines can run out of memory or threads due to poor programming when a system is stretched.
- » Disaster Recovery plans are very complex and expensive to test. Whilst not directly related to availability, some applications need to be taken down whilst the Disaster Recovery test is performed – causing scheduled downtime.
- » Logical data corruption – a particularly difficult software issue to resolve is where a database gets corrupted for no obvious reason. There is downtime whilst the data is restored and inevitably some loss of production data. However if this is not to be repeated a full root cause analysis is needed. This is only possible when there is both a full size development system where the scenario can be replayed and also full capture of all system inputs – an expensive and hence relatively unusual situation.

**Packaged third party applications and components** – established and widely used software has the ability to deliver high levels of reliability – the large user base rapidly identifying problems and the supplier having the revenue stream to fix them quickly. Conversely niche software can cause issues as there is no direct control over either the code or the release process.

**Configuration management** – a common cause of problems is that an informal approach to release management results in an incorrect version of a piece of software being put into production. The only solution to this is tight control of software releases between development, quality assurance and production. We have evolved the Development, Education, Test, Acceptance and Production (DETAP) model to define the boundaries between environments and the process for migrating versions of software between them.

## The Atos Origin DETAP model



**Complexity** – is quite simply the enemy of reliability. Traditionally the number of bugs in a system was directly related to the number of lines of code and then reduced in proportion to the quantity and quality of testing. Contemporary systems have many more components and the number of bugs is a function of many factors. More importantly it is now far more difficult to test a system – the notion of an end-to-end system test makes no sense in an SOA environment where many services are shared and not formally ‘part of the system’.

**Standardisation** – the concern here is that a loosely-controlled development environment can proliferate both technologies and functionality. A key benefit claimed for SOA is to achieve the Holy Grail of re-use of code – however the realisation may be just as challenging as it has been for other development paradigms. The benefits of standards are clear however:

- » Problems only happen once
- » The number of discrete skills needed is reduced – particularly in the support organisation (who will have to retain these skills throughout the life of the system). Reducing the number of skills required not only cuts costs it also improves quality as staff will achieve greater depth of knowledge in the remaining technologies.

## SAP – a mature approach to configuration management

SAP, arguably the most successful ERP vendor in the world, has an enviable reputation for the reliability of its systems. Part of this is undoubtedly due to the rigid adherence to the model whereby software is moved in a highly controlled way from the development to the Quality Assurance (QA) environment where it is tested and then a further highly controlled process to take it into production.

### Problem fixing

Fixing problems more quickly is the other half of the availability discussion – downtime depends both on the number of incidents and the time taken to fix them. Our study identified factors that would have reduced the resolution time. There are three stages to resolving problems:

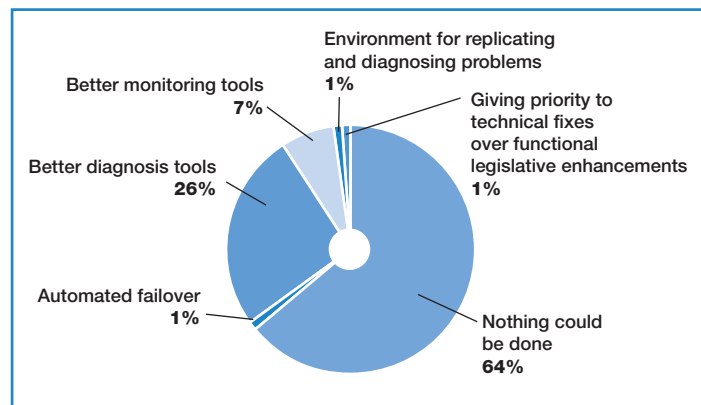
#### Identification that a problem has occurred

In the worst case this happens when a user raises a trouble ticket. However this is not acceptable in most modern environments and some sort of automatic detection is needed. There are a number of types of automation:

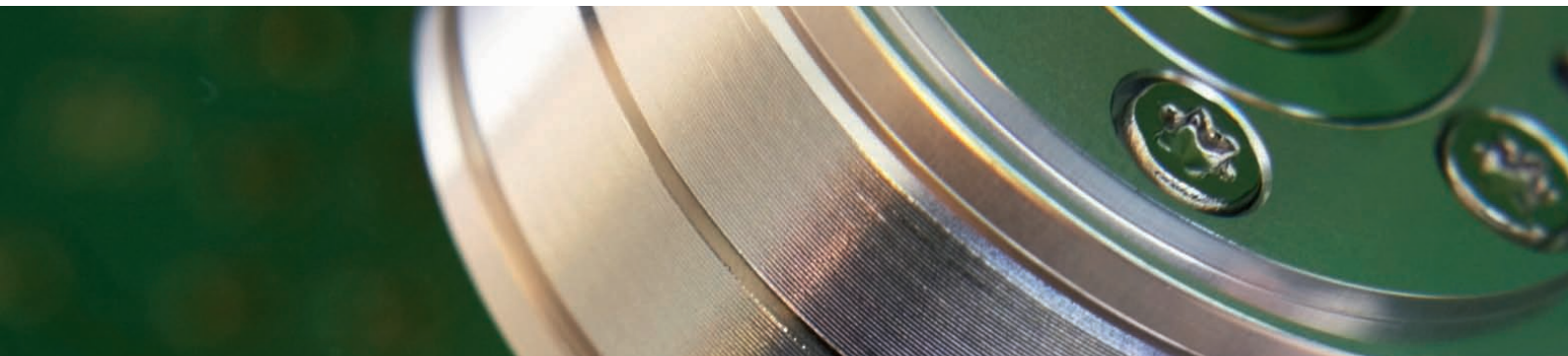
- » End-user experience tools are an innovative development in this area. Here the response time and availability seen by the end-user is directly and continuously measured from analysis of traffic travelling across the network. Advanced tools can identify not only the application in use but also the individual transaction enabling very granular monitoring of services – and to raise an alert when something goes wrong.
- » Alert based performance monitoring tools. These tools monitor all of the individual elements of the infrastructure and raise alerts when something malfunctions. They are good at detecting hardware and operating system failures but typically have no understanding of applications.

- » Application and middleware monitoring tools have a deep understanding of the application environment. These are becoming more important as the development technologies become more complex and the use of middleware grows.
- » Logical corruption is the most difficult problem to detect automatically. This will usually require the application to detect an anomaly and raise an exception. This may also occur within the database particularly if there are extensive validation rules within the database itself.

#### Factors that reduce the time to fix problems



**FIXING PROBLEMS MORE QUICKLY IS THE OTHER HALF OF THE AVAILABILITY DISCUSSION – DOWNTIME DEPENDS BOTH ON THE NUMBER OF INCIDENTS AND THE TIME TAKEN TO FIX THEM**



## Triage

Having identified that a problem has occurred the next step is to understand the root cause and hence how to resolve it:

- » Correlation is an essential element of triage. When a problem occurs there is often a flood of alerts; consider for example if a router fails which would in turn cause alerts from the entire infrastructure that is at the 'far' side of the router. Reducing this flood of alerts is a complex process to automate as it requires both knowledge of the topology of the infrastructure<sup>8</sup> (the devices and connections between them) and the ability to define policies. A well set-up correlation engine will reduce the number of alerts to perhaps 1.5 per incident.
- » Knowledge management is very valuable particularly for legacy applications where skills have been lost over time. Documenting common problems and their resolution saves time and improves quality.
- » Automation is an increasingly interesting approach. This can take several forms. Some standard tools will respond to an alert by automatically following a diagnostic path and polling individual devices in order to establish the root cause. In other cases the management tool can raise a trouble ticket but also append diagnostic information obtained from suspect devices.

## Resolution

Having established what the problem is the resolution is often relatively straightforward. However in fixing a current problem there is a great risk of introducing a new problem – particularly when staff are under pressure to restore service quickly. It is essential that formal processes are followed and documented so that there is an audit trail should any other problems occur.

Much has been written about autonomic approaches whereby the whole process of identification, triage and resolution is automated. Whilst simple examples have been implemented such as re-starting a 'hung' server, in general this is not worthwhile – the complexity of the diagnosis<sup>9</sup> and the need to have something that reliably works 100% of the time requires a lot of investment and testing<sup>10</sup>.

<sup>8</sup> As found in a CMDB – see white paper 'Advanced Configuration and Asset Management'.

<sup>9</sup> Additional problems can occur if an automated 'fix' is incorrectly initiated due to incorrect triage.

<sup>10</sup> It is interesting to note that, while in general technical failures are less common, they tend to take longer to fix. This is partly because better design has resulted in the 'easy' problems no longer occurring – leaving a rump of more complex problems. The other reason is that support staff have less practice in resolving problems.

**IN FIXING A CURRENT PROBLEM THERE IS A  
GREAT RISK OF INTRODUCING A NEW PROBLEM**

# 65% OF PLANNED DOWNTIME IS DUE TO APPLICATION AND DATABASE MAINTENANCE WHILST THE OTHER 35% IS DUE TO HARDWARE, OPERATING SYSTEM, DATA CENTRE AND OTHER WORK

## PLANNED DOWNTIME – REDUCTION BY DESIGN

The industry view is that 65% of planned downtime is due to application and database maintenance whilst the other 35% is due to hardware, operating system, data centre and other work. Opportunities to reduce planned downtime include the following:

### Application upgrades

The disruption that occurs whilst implementing a new version of software is largely in the hands of the developers. Packaging to enable fully automated deployment reduces downtime and also the risk of human error. Structuring the system so that a new version can be installed and configured whilst the existing version is running is desirable – the new version coming into operation when the system is restarted. If no database changes are required then this opens the possibility of installing the new version across a load-shared cluster of application servers and then restarting them one at a time – without any disruption to service.

### Operating system and database patching

These patches typically require a restart thereby causing downtime. By spreading applications and databases across multiple servers this can be done sequentially, thereby maintaining end-user availability.

### Reorganising data

This process was traditionally required to reduce disk fragmentation but now occurs as part of Information Lifecycle Management (ILM) where data is moved between different storage tiers to meet performance and cost objectives. The latest storage virtualisation technology can permit this move to occur without disrupting service.

### Backups

One of the main reasons for planned downtime is the need to take backup copies of data. Traditionally this required the application to be unavailable whilst the backup was taken (the so called 'backup window'). More recently it has been possible to create a 'snapshot' of the data during a brief (a few seconds) period of unavailability and the backup is then taken from the snapshot whilst the application is available. However the fundamental reasons for taking backups are being revisited:

- » As hardware becomes more reliable and resilient the need to fully restore a system following a hardware failure becomes increasingly uncommon.
- » The most common reason for restoring data is to recover a single file or mailbox – typically because a user has accidentally deleted some data or corrupted it.

- » The key remaining reason for backup is to be able to restore in case of operations error (e.g. accidental deletion of the wrong files or tables) or application error (e.g. the database got corrupted due to software errors and needs to be restored to a point in time when it was still valid).
- » Applications are becoming increasingly interdependent and hence restoring all related data to a common consistency point is often impractical. SAP, for example, specifically recommends that their systems should never be restored.

Best practice is therefore emerging that minimises the role of the traditional tape backup to being purely a recovery mechanism of last resort. Techniques such as replication provide the means to recover systems quickly in the unusual event of hardware failure. Another approach is to backup to disk – often using a Virtual Tape Library (VTL). Restoring individual files and mailboxes should be accomplished without system downtime.

### The need for hardware upgrades

This is a common cause of planned downtime. Hardware vendors have worked hard to design their enterprise-class equipment to be capable of non-disruptive changes. Hence additional equipment such as processor boards, disk storage and SAN ports can all be added without downtime. It should be noted that these features are often only available on the larger enterprise-class equipment. Another technique in a clustered environment is to upgrade the passive device, failover to the upgraded machine, upgrade the primary machine and then fail-back.

A related cause of downtime is the need for operating system and firmware upgrades. These are still necessary. Enterprise class operating system vendors are working to minimise the proportion of upgrades that necessitate downtime – for example 50-60% of upgrades to the kernel of the latest version of IBM's AIX operating system can be performed 'hot'.

Server virtualisation is a good way to obviate this type of downtime. Virtualised servers can be moved non-disruptively (using VMware VMotion for example) prior to work being undertaken on a physical server.

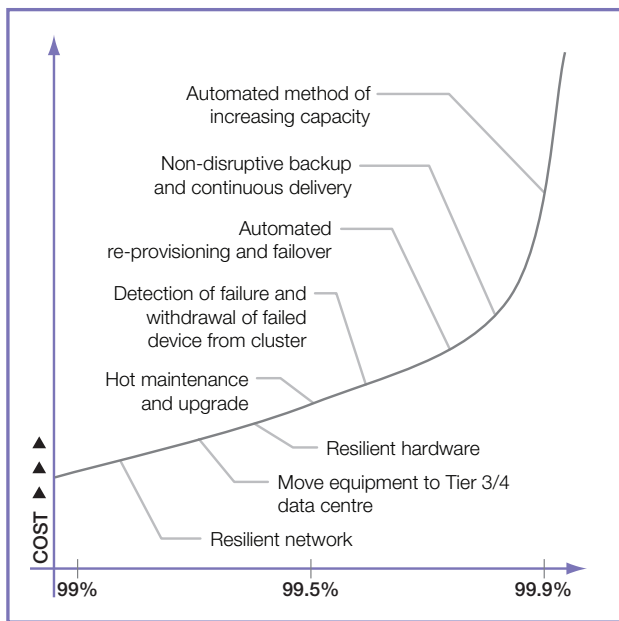
### Physical changes to the data centre

These are an infrequent but sometimes unavoidable cause of planned downtime. This normally only applies to the lower tiers of data centre and is most frequently associated with upgrades to the electrical supply, air conditioning and switchgear.

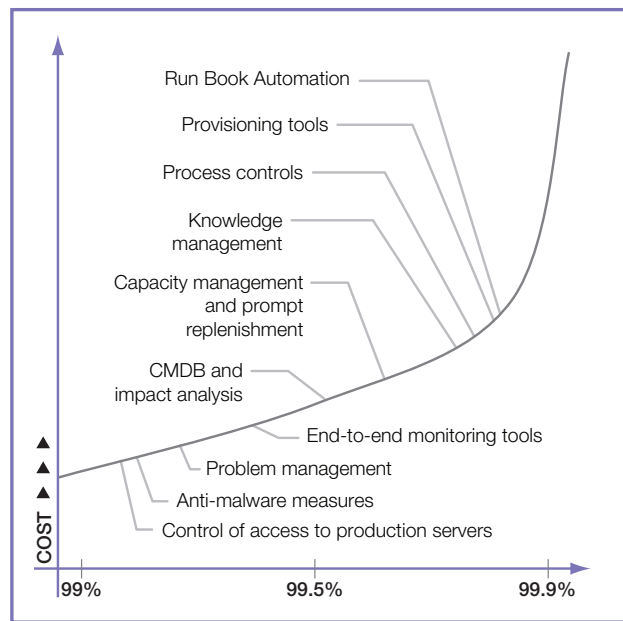
## WHAT ARE THE OPTIONS?

The investment in a service has to be proportionate to its business value – which will often determine the required availability. Decisions to invest in particular technologies, processes or software will therefore be based upon the impact that they will have on availability. There is no general level of availability at which each type of investment is worthwhile; however experience shows that they become viable at different levels as shown in the following diagrams.

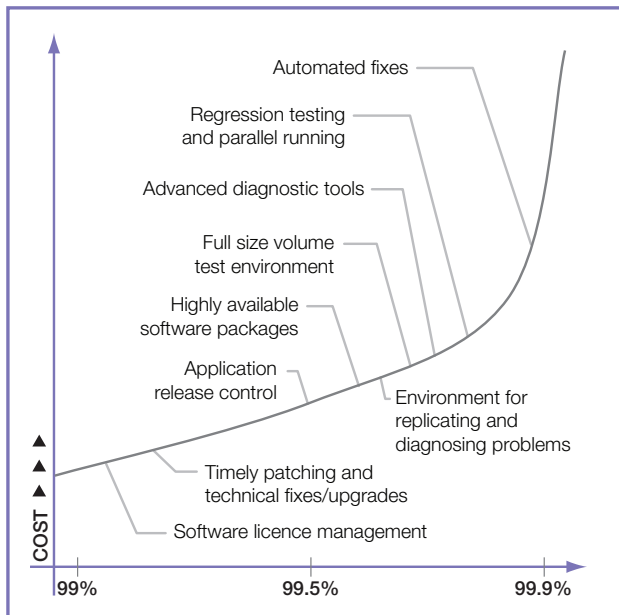
### Technical investments to improve availability

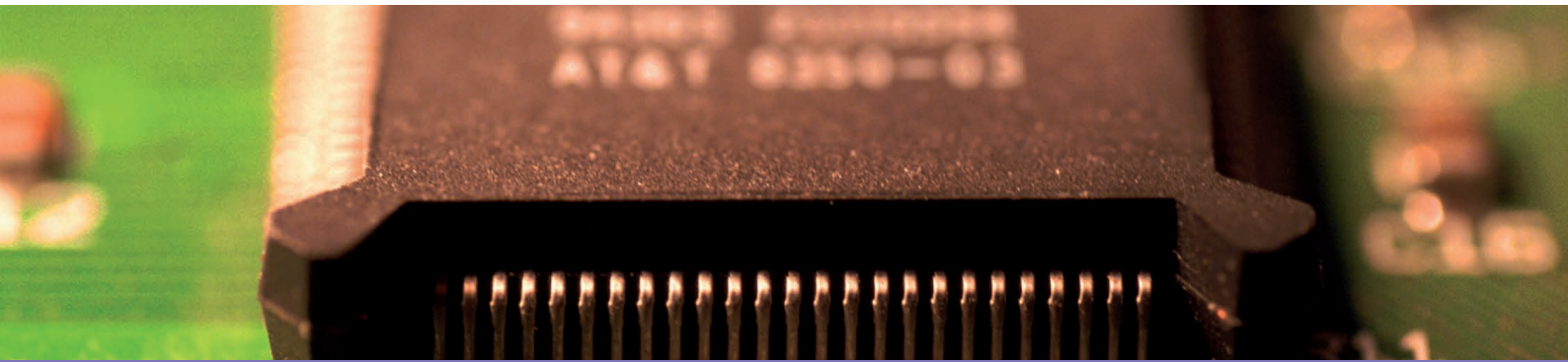


### Human investments to improve availability



### Software investments to improve availability





## THERE IS NO UNIVERSAL APPROACH – THE INVESTMENTS IN HIGH AVAILABILITY NEED TO MATCH THE BUSINESS REQUIREMENTS FOR A PARTICULAR SYSTEM

### SUMMARY

And so we have seen that:

- » Understanding the business requirements for service availability is essential – both to ensure mission-critical systems support the business and also to ensure value for money.
- » As technology has got more reliable the primary causes of downtime are related to software and human factors. Modern development techniques (such as SOA) have raised the proportion of incidents due to software.
- » Improving availability is a combination of three things:
  - Less scheduled downtime
  - Fewer incidents
  - Resolving incidents more quickly when they occur.
- » Improving service availability is a costly matter – particularly as permitted downtime approaches zero. Not all systems have the same business need for availability and therefore should have different SLAs. But highly-available systems cannot be allowed to be dependent on systems with lower availability.
- » There are many causes of downtime and attempts to improve availability have to take a holistic view of a system. Many of the improvements will require significant investment in areas such as automation and testing regimes. Naturally these are best implemented as part of the initial implementation rather than a remedial exercise.
- » There is no universal approach – the investments in high availability need to match the business requirements for a particular system.
- » Modelling is a valuable technique to evaluate the overall availability of systems prior to implementation.

### ACKNOWLEDGEMENTS

This paper would not have been possible without the help and support of many people. Special thanks are due to Mick Symonds. The author would also particularly like to thank:

- » Adil Tahiri
- » Mike Smith
- » Joep van Haastrecht
- » Mark Ellery
- » Michiel van der Waaij
- » Frans van Leuven
- » Richard Munro
- » Paul Hardy
- » Harjit Gill
- » Peter Genefaas
- » Steven Hird.

## APPENDIX – MODELLING AVAILABILITY

It is no accident that products in everyday use are as reliable as they are. Engineers have for many years used system modelling techniques to analyse product reliability and then focus investment on the areas that have the most overall impact. Although the underlying mathematics are complex the approach has been embodied in software packages that are straightforward to use. These approaches can be used to model IT services with similar effect.

A service is modelled by considering all of the things that can cause it to fail. Each of these is represented by a block on the diagram below. Many of these blocks are 'single points of failure' – if they fail then the service also fails. However resilience can remove single points of failure and can be employed in two ways:

### » Using load sharing

In this case there are two or more identical devices that share the workload equally. Availability is improved as a device can fail without the service necessarily being impaired – the remaining devices just share the extra workload. The devices must have spare capacity for this to work of course – typically running at a maximum of 50% or 66% under normal conditions.

### » Using failover clustering

In this case a duplicate device is provided that is normally idle. When the primary device fails this is identified and system management brings the duplicate device into action. It should be noted that the system management function is itself a single point of failure, although not part of the prime delivery path.

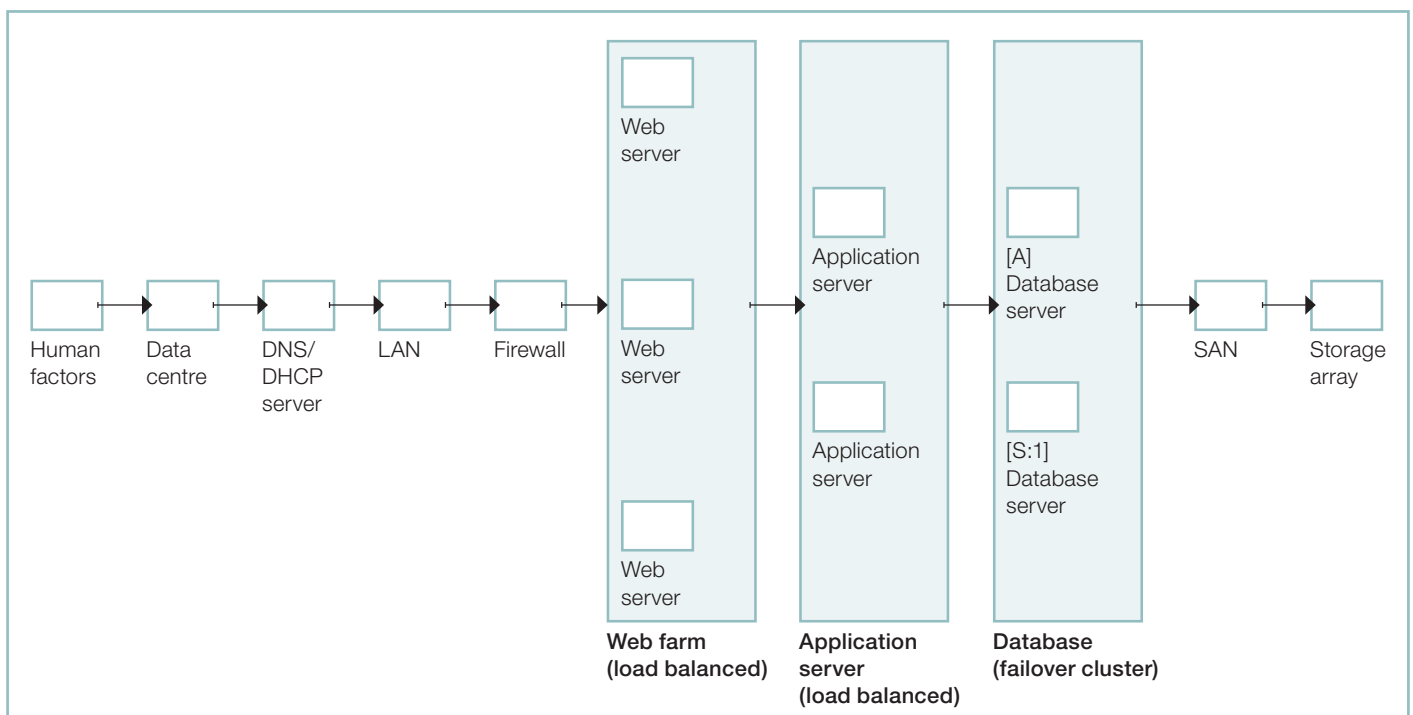
The parameters of each block need to be assessed – in simple terms how often it fails and how long it takes to fix it when it breaks:

- The MTBF of a device is a well established concept and is usually measured in months or years.
- The MTTR is again measured in terms of time.

It follows that the expected availability of a block (a percentage) is given by:

$$\text{Availability} = 100 \times \left( \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \right)$$

An availability model (drawn using ReliaSoft BlockSim 7)



The availability of individual blocks is used to calculate the availability of a service. This is a mathematical calculation based on probability theory<sup>11</sup>. It can be seen that the availability of a service deteriorates rapidly as it becomes dependent on more individual devices. So for example a service that depends upon 10 devices, which individually are 99.9% available, will only be 99% available. This clearly demonstrates an intuitive result – that the more complex something is the less reliable it will be.

At this point it is important to ask the question ‘what is a block’? For example is it:

- » A server?
- » The server plus the software running on it?
- » The motherboard of the server (with the other components treated as separate blocks)?

The answer is that it could be any of these – depending on the availability of data. Manufacturers can usually provide MTBF figures for hardware devices whilst data for software will need to be obtained from existing users. It is usually best to model using the highest level blocks for which data is available – treating the data centre as one block for example rather than separating it into power supply, cooling and all of the other constituent parts.

The data centre is an example of a block that is not a hardware component. This will have an availability figure of its own – which can be modelled like any other physical component. The effect of splitting infrastructure between two ‘twinned’ data centres can then be modelled – essentially a load balanced configuration as modelled earlier.

The modelling of software is a separate discussion and again depends upon the availability of data. A service will depend upon many individual pieces of software. However it is not helpful to model these individually as availability figures will not normally be available for them. In practice it is easiest to treat all the software on a server as one entity and derive an availability figure based on operational experience (this may or may not include the hardware availability). Hence a mature package will typically have a much higher availability than a recently developed piece of bespoke software. In some cases the only data will be for the complete service in which case a single block is used rather than modelling on individual server blocks.

It then remains to add the effects of human error. This is in inverse proportion to the degree of deployment of the various techniques referred to earlier. Having produced a model it can be used to analyse the relative impact that each block has upon the overall service availability. So in the following hypothetical example (using the blocks in the earlier diagram) investment should clearly be targeted at initiatives to reduce failures due to the firewall and storage array.

#### Availability of individual blocks following analysis by ReliaSoft BlockSim 7

Block	Availability
Data centre	99.99%
DNS/DHCP server	99.99%
LAN	99.96%
Firewall	99.71%
SAN	99.97%
Storage array	99.88%
Web farm (load balanced)	100%
Application server (load balanced)	100%
Database (failover cluster)	99.95%
Human factors	99.89%
<b>Overall availability</b>	<b>97.67%</b>

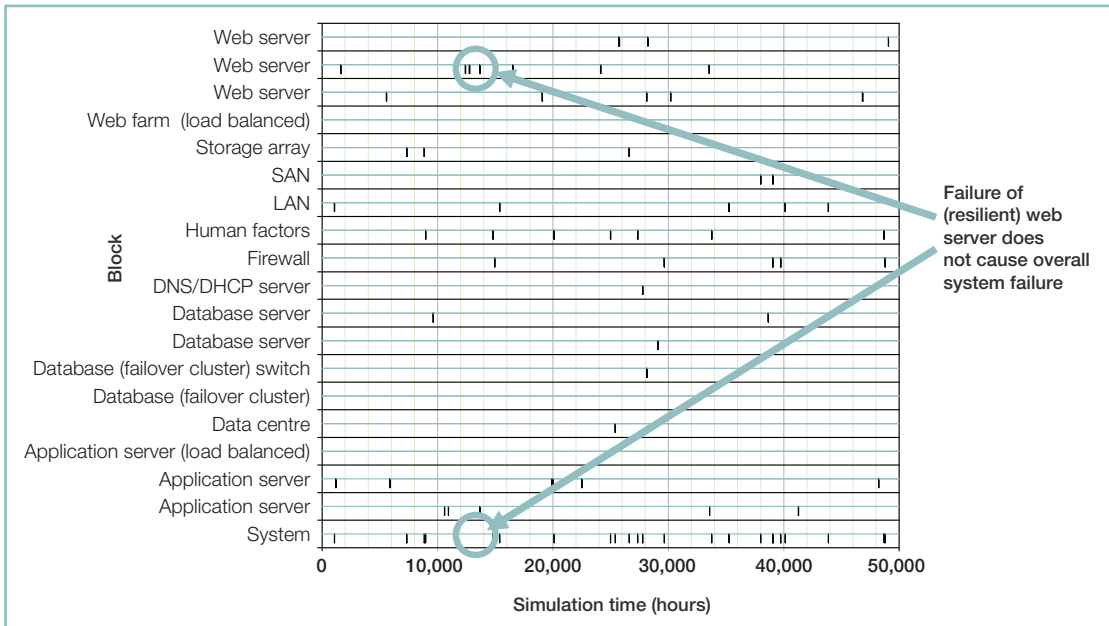
<sup>11</sup> So if there are ‘n’ devices that all have to be working for the overall service to be available then the availability of the service is:

$$\text{Availability} = 100 * \left( \frac{A_1}{100} \right) * \left( \frac{A_2}{100} \right) * \dots * \left( \frac{A_n}{100} \right)$$

Where  $A_r$  is the availability of device (r)

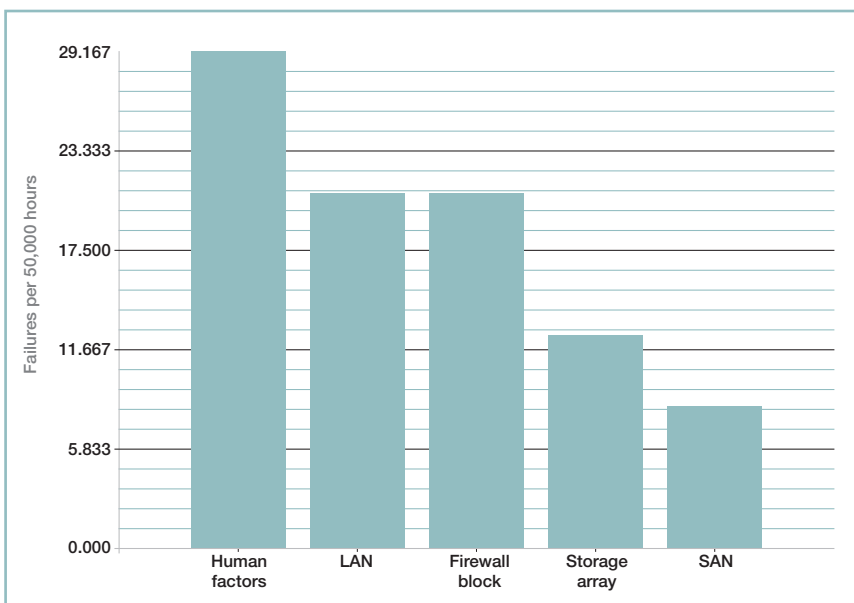
The model can also be analysed using Monte Carlo<sup>12</sup> methods to simulate the performance over time. It is interesting to notice that resilient blocks fail but that the overall service does not.

### Simulation of failures over time using ReliaSoft BlockSim 7



The notion of a failure criticality index is relevant as it highlights the blocks that have the greatest overall impact on service availability.

### Failure criticality analysis using ReliaSoft BlockSim 7



So it is possible to model the reliability of a service – and hence ensure that it meets the SLA at minimum cost.

<sup>12</sup> Monte Carlo methods use random numbers to model the time that each block fails (based on the specified MTBF) and the time to repair it (using the specified MTTR). These also make assumptions about the distribution of failures – the normal method is to assume that a failure is equally likely to happen at any time – an exponential distribution.

## About Atos Origin

Atos Origin is an international information technology services company. Its business is turning client vision into results through the application of Consulting, Systems Integration and Managed Operations. The Company's annual revenue is EUR 5.5 billion and it employs 50,000 professionals in 40 countries. Atos Origin is the Worldwide Information Technology Partner for the Olympic Games and has a client base of international blue-chip companies across all sectors. Atos Origin is quoted on the Paris Eurolist Market and trades as Atos Origin, Atos Worldline and Atos Consulting.

## Contact us

4 Triton Square  
Regent's Place  
London  
NW1 3HG  
United Kingdom  
Tel: +44 (0)20 7830 4444  
Email: [mo.marketing@atosorigin.com](mailto:mo.marketing@atosorigin.com)  
[www.atosorigin.co.uk](http://www.atosorigin.co.uk)

“Atos Origin is committed to delivering highly available systems that both meet the business needs of our clients and are affordable. This requires us to understand the real drivers of downtime and how they are best addressed when engineering solutions.”

Guy Lidbetter, Chief Technology Officer, Atos Origin UK