

## Introduction

For information technology (IT), the last decade has been a revolution. Our lives and our businesses depend on IT to a magnitude that has not been seen as possible before. And even though there is a lot of fuss about this change, many people have still not realized their dependencies on functioning IT systems. But this book will not tell you how one preaches the importance of IT. Instead, it is for the situations where the importance is acknowledged in principle, and where we have to map business objectives to IT availability plans and execute those plans.

In other words, the mission statement of this book is

**Show how it is ensured that IT services are available  
when they are needed, balancing benefit and costs.**

The central question that will be answered is “how,” not “why.” We use a holistic view of that topic. When a solution is planned and implemented, every aspect must fit together and it is not sensible to restrict ourselves to certain areas.

Therefore this book approaches solutions to IT service availability and continuity from planning, via implementation descriptions, through to operations. This covers the whole range of work areas that are needed to establish and maintain a solution.

In addition, technical areas are covered too. We will discuss problems that hardware, software, infrastructure services, human actions, and company locations (sites) cause and the solutions that they can provide. We do not want to rest with descriptions that explain how one can ensure IT service availability in the case of hardware and software failures, but that do not mention floods or human errors by system administrators. Instead, the whole problem and solution range is accommodated, just like all work areas.

The book’s goal is achieved with the experience that I acquired in planning and implementing successfully high-availability and disaster-recovery solutions in the last 15 years, for major international compa-

nies. The approaches and templates used have been successfully realized in many places and resulted in lots of practical experience about what works and where typical problems and stumbling blocks are – this book will make this experience available to you as well.

To achieve that goal, a solution-oriented presentation structure has been chosen. The first chapters will introduce the problem domain and will show the generic structure of possible solutions. The following chapters will present components of that structure, one after another.

But let us first describe the intended audience of this book in the next section. Then, Sect. 1.2 on p. 4 presents the book’s roadmap in more detail, and Sect. 1.3 finishes the introduction with real-life examples.

## 1.1 Audience

In today’s tight financial climate, business consequences of IT misbehavior or outages are often not realistically analyzed. Therefore the myth remains that high availability and disaster recovery is only something for large enterprises with an IT staff in the hundreds or thousands, big data centers, and established IT operations. But by now, solutions are mature and standardized enough to be able to be implemented in small work groups and with less formal processes. All that is needed is the intention to do good work and to care for quality.

We need to understand the link between business objectives and IT solutions. But we want to reuse existing design patterns; solutions that are created anew for each company’s situation are too expensive. For that, we need to understand also the possibilities and limitations of typical IT solutions to high availability and disaster recovery. Only with that understanding firmly in place are we able to plan and implement properly a solution that fits to our situation.

The information from this book is not only of interest if one wants to implement high availability and disaster recovery in one’s own company. Quite often, IT is outsourced, as it is not seen to be at the heart of one’s business and one hopes to get better services from professional companies that concentrate on IT. Then one still needs the information from this book to negotiate contracts – in particular, service level agreements (SLAs) – with suppliers. Do not forget that one can outsource implementation and operations, but one cannot outsource responsibility.

That said, the book’s main target audience is architects, system designers, and those who shall implement the systems. But it has also content for executive managers. For successful projects, one needs to look beyond one’s own nose and read up on topics that are in other spheres of activities. This book will provide information about such adjacent areas, for each audience group. So let us have a look at the information you can get from it.

► *Executives – CIO and CTO*

For chief technology officers and for chief information officers who are interested in technical matters, the book delivers:

- An overview of IT availability and continuity. This will provide everything that an executive needs to know about high availability and disaster recovery.
- Technical information that is needed to communicate with architects and system designers.
- A link between business requirements and IT solutions.

It serves as the guide for how to give good objectives and guidance to the project team that has to implement high availability and/or disaster recovery. It also has information that allows us to judge the quality of their work, and to understand the technical reports that they are producing.

► *Architects and System Designers*

This audience group is those who are responsible for the actual system design and planning. This is the main audience of this book.

- A roadmap is presented that will yield a full-fledged architecture that goes beyond mere system design, including objective collection and business process architecture.
- A structured approach for solution finding and implementation is presented. This is the process that delivers the system design, together with explanations on how one can adapt them to local requirements.
- Solution structures are worked out, to help to divide the big problem domain into chunks that can be approached one at a time. This is the template where we can draw our system design from.
- Technologies are presented that build a toolbox for realization of high availability and disaster recovery.
- Adjacent areas that are not strictly part of the architect's or system designer's task are mentioned as well, and serve as an introduction. In particular, operational aspects are covered.

We take a broad view of the topics covered that caters for generalists and not for specialists who want to know the ins and outs of each bit in every configuration file. The book's content will help by providing an end-to-end view and not losing itself in the deep trenches of particularities of specific products or specific versions.

► *System Implementors*

Solution structures and technologies are the meat that make up the later chapters of this book. These parts of book cater especially for those system

implementors who are interested in practical experience that they can use in their own environment.

- The central message for this audience group is how to implement a high-availability or disaster-recovery solution in such a way that it can be operated successfully and efficiently.
- To help with that, typical pitfalls and tricks are presented.
- Requirements are featured explicitly that help to trace back implementation plans to the system designer's work, and ultimately to the business objectives.
- The process of architectural planning is also of great interest as it enables better communication with one's colleagues.

In the end, as every implementor knows, failure will happen and cannot be avoided. It is necessary to plan for it in advance, both from a process and from a technical point of view, to take a broad view of the needs and the possibilities of implementation work. How this is done, both in principal and specifically for actual technology components, is an important aspect of this book.

## 1.2 Roadmap of This Book

We will start with the elementary concepts to get an introduction to the overall theme. Then two chapters will present the big picture: architecture and system design. After that, we will go into technical details and present categorized solution strategies. Each category will have its own chapter.

Within the technical chapters, we will present solutions that work for high availability and disaster recovery alike, but we will focus on high availability. A chapter on disaster recovery pulls together all parts from all solution categories and tells what is special about it.

The appendices feature a treaty on reliability and the math of it; we give an introduction to data centers and to service support processes. An index and the bibliography close the book.

That is the roadmap in general terms. Let us introduce each chapter with a short summary to give you more information.

### ► *Chapter 2: Elementary Concepts*

Business continuity is introduced as the overall goal of our activities. IT service continuity is our task to contribute to that goal. To achieve that, we need categorization of systems and outages, to leverage existing concepts. Minor outages are those that happen more often and do not do much damage; protection against or recovery from such minor outages is the task of high availability. Major outages are less probable and are covered by disaster recovery.

Quantification of availability is presented. We use them also in SLAs, which are part of (formal or informal) contracts between the IT department, IT contractors, and business owners.

The same basic approach is used to achieve high availability and disaster recovery: robustness and redundancy. With that tool in our arsenal, we are ready to design layered solutions that protect our services in depth – we can handle failures on several system levels.

► *Chapter 3: Architecture*

The architecture is described on those three abstraction levels: business objectives describe what shall be achieved. The conceptual model explains how the solution fits into business processes, and how it is connected to other IT processes. Furthermore, the architecture contains the system model that describes the technical solution strategy.

For each of these three abstraction levels, several questions are answered: What is the system concerned with? How does it work? Where is the data worked with, or where is functionality achieved? Who works with data and achieves functionality? And when is it done? Answering all these questions for each abstraction level gives us a structured presentation of the complete architecture.

► *Chapter 4: System Design*

The system design has the technical meat of our solution. It describes what systems are protected against which outages and how this is done. To achieve such a description in a structured way, we introduce the concept of the system stack, a categorization of those components that make up IT systems and services. The system stack is so important that we use it to structure the rest of this book when we present technological solutions for component categories.

We pick up the elementary concepts of robustness and redundancy and explain them in more detail. This cumulates in a solution roadmap that shows how we create a good system design that fulfills our requirements, starting from failure scenarios. As an illustration, we conclude that chapter with solution patterns and with the use case of high availability for an SAP server.

► *Chapter 5: Hardware*

Hardware is the component category where high availability was first realized; therefore, we find the most mature solutions in this area. First we look at all hardware components of a computer system and explain the technology that can be used for a high-availability solution.

Special focus is given to disk storage because it is very important – after all, all our business data is kept here. But we do not stop at the different technologies that make disks highly available, we also present stor-

age subsystems, appliances that provide storage area networks (SANs) or network-attached storage (NAS).

Since we do not build our systems ourselves, we need to pay special attention to vendor selection. When the purchasing decisions have been made, installation, maintenance, and operation are also important, and these topics are handled in this chapter.

► *Chapter 6: Operating Systems*

High availability on the operating system level is the most widely used technology today. It is named host clustering and comes in two forms, as failover clusters and as load-balancing clusters, both are presented in this chapter.

Failover clusters make IT services independent of the computer system they run on. When a hardware error or a software crash occurs, the service is migrated to another computer system in the cluster. This cluster technology relies on disk storage that can be accessed from all systems and that is redundant in itself.

Load-balancing clusters distribute server requests over a set of identically configured systems. The distribution handles outages of one of those systems. This kind of cluster can only be used if the application has no state, since there is no shared storage between the cluster systems.

The chapter concludes with an excursion on the future of host clustering in the face of current trends towards server consolidation. Host clustering is based on the usage of extra computer systems, to get redundancy for failure recovery. Server consolidation targets the reduction of the computer systems used and has therefore the exact opposite goals. Host virtualization may help to combine both objectives, though today this remains a trend and future development cannot be assured.

► *Chapter 7: Databases and Middleware*

Middleware components are application-independent software products that are integrated as part of an application to realize an IT service. The prime example of middleware components is database servers: they are utilized in many mission-critical applications. Other middleware components are Web servers, application servers, messaging servers, and transaction managers.

This chapter presents the high-availability and disaster-recovery solutions that are available as features of middleware components. Most prominently, database clusters are introduced.

► *Chapter 8: Applications*

After all base components have been covered, it remains for us to describe how high availability is achieved for applications. Seldom will we

implement full redundancy within an application – instead, we will utilize one of the high-availability options that we learned about in the previous chapters. But we have to decide which option to use, e.g., if we want to utilize a host clustering solution or if we want to use middleware clustering.

The solution approach chosen often has demands on the application, either on its realization or on its implementation. For example, one cannot use a failover cluster for any application, the application must fulfill some requirements. These requirements are spelled out in detail.

► *Chapter 9: Infrastructure*

Infrastructure is the set of application-independent IT services that are used by an application. Most important, this is the network and associated services like Domain Name Service (DNS), Dynamic Host Configuration Protocol (DHCP), or directory and authentication services.

This chapter will show how we create a highly available network infrastructure and will also present high-availability solutions for three important services: DHCP, DNS, and directory servers. It concludes with the influence of high-availability solutions on backup and monitoring.

► *Chapter 10: Disaster Recovery*

The previous chapters will have presented many solutions that apply both for high availability and for disaster recovery. But disaster recovery has specific important aspects that shall be presented in context; therefore, this chapter presents the overall approach, a conceptual model for disaster recovery, and the technology that is used to realize it.

A description of a prototypical disaster-recovery project rounds off the topic, as well as a step-by-step description of procedures that are used to activate backup systems in case of a disaster.

► *Appendices*

Reliability of components is an important property to achieve high availability. In particular for hardware components, reliability numbers are known and one can use them to assess reliability of combined systems. Appendix A introduces *reliability calculations and statistics*, the math that is necessary to do those assessments.

*Data centers* are the topic of Appendix B. We will not capture everything that is needed to create a top-notch data center – that goes beyond the book’s scope. But a good data center is an important asset in implementing high availability, and we will see what aspects contribute to that importance.

Last, but not least, *service support processes* are described in Appendix C. The presentation orients itself along the categories of the Information Technology Infrastructure Library (ITIL), an industry standard of

processes of how one operates IT systems properly. Good operation is very important for high availability and disaster recovery – all excellent implementation strategies and work will not help, if failures get introduced during maintenance operations. This appendix presents those parts of the ITIL processes that are especially relevant.

### 1.3 Real-World Examples

Before we start with business issues and elementary concepts in Chap. 2, let us entertain ourselves with some “war stories” of real-world IT problems and set the tone of issues we strive to avoid in our own installations.

**Example 1 (Deferred overdue upgrade of legacy system).** Over the 2004 Christmas holidays, Comair – a regional subsidiary of Delta Air Lines – needed to reschedule a lot of flights owing to severe winter storms. Reportedly, the aging crew management system software could not handle more than 32 000 changes per month. (Without knowing the cause exactly, let us assume that the real limit was  $32\,767 \pm 1 \dots$ ) When this limit was exceeded for the first time, the whole system crashed on December 24 and return to full operations took until December 29.

This IT crash caused cancellations or delays of roughly 3900 flights and stranded nearly 200 000 passengers. The costs for Comair and its parent company, Delta Air Lines, were estimated to be in the \$20 million ballpark, not counting the damaged reputation and subsequent investigation by the US Department of Transportation.

The reasons sound familiar to everybody who works in IT departments of large companies these days: Y2K, 9/11 fallout for aerospace companies, company acquisition by Delta, and a very tight lid on IT expenses brought enough workload for the IT department. The technical staff knew about the problem and management did not believe them. Replacing a supposedly working legacy system did not have high priority, and risk management was not prudent enough. Actually, the real issue was that they had no contingency plan for outages of that mission-critical system, and no means to create a quick workaround for the incident. ■

**Example 2 (Network outage for a whole company).** Flaky connections between network switches caused redundancy configurations to fail, turning the whole network unfunctional and thus almost all IT systems of a big manufacturing company unusable. Luckily, the actual manufacturing operation was not impaired, as the IT systems deployed there had their own separate network and were able to operate for several hours without connectivity to the company network.

The network used redundant lines between switches and relied on the Spanning Tree Protocol (STP) to create a loop-free topology that is needed



for network operations. The flaky lines resulted in continuous recalculations of the minimum spanning tree, overloading the switch CPUs at one point. Owing to problems in the switches' operating system, a loop was formed and this error situation propagated to all other network devices, rendering them all unfunctional.

Before that case happened, there were anecdotal reports about problems with STP, but no real hard data about its reliability was available. Therefore this redundant architecture was chosen, not knowing that it does not scale well enough for big installations. Even though the network staff were able to remedy the deficiency quickly, a complete reimplementation of the network architecture had to be done later on. This raised the incident costs even more, beyond the actual damages that were caused by the nonreachability of all servers.

This is an example case of a technology that was not robust enough for the creation of highly available infrastructure designs. Only experience would have prevented such a design; the importance of experience is discussed in more detail in Chap. 3. ■

**Example 3 (Propagation of human error during upgrade).** Even in situations where lots of effort was spent to create architectures that survive failures, there is always the last resort for error causes: our colleagues. A hierarchical storage management (HSM) environment implemented both the usual high availability and also intended disaster-recovery protection by placing backup tapes at a remote location, and running the HSM system as a metro cluster. This means that two HSM systems were placed at two different sites, with a high-speed network connection. Each system has a complete set of all data, on a tape library that is connected to it.

Upgrading a tape library and exchanging all backup tapes for new high-capacity tapes caused the complete erasure of all existing tapes. What happened is that cables got exchanged between two tape drives. Formatting of the new tapes was started in one drive. The exchanged cables only controlled the data transfer (write and read) commands; the load commands were transferred over another cable that was connected correctly. Since an HSM tape library is active all the time, new read or write requests led to loading of tapes into the wrong drive, where they were dutifully formatted. At the same time, all data access commands returned errors since they tried to access the new unformatted tapes; but some time was needed to detect this. In that time span, most of the tapes had already been reformatted.

By all accounts, this could be classified as human error and as software error. Of course, the cable should not have been exchanged. But such errors are so common that good software would take them into account and would not reformat tapes that are formatted already and which have data on them, without being told explicitly to do so.

Single point of failure analysis, as explained in Chap. 3, would have helped to prevent that failure. ■

**Example 4 (Failover cluster rendered useless).** A large business-important file server, with several terabytes of data, had a file system corruption that led to whole system aborts (colloquially called “system panic”). This file server was a highly available failover cluster system with two nodes; file system errors caused the file service to switch to another cluster node. The file system itself switches too, the error is not noticed during service initialization, and after a few minutes the new node panics again. The service switches continuously between the two cluster nodes.

Even though the switch ping-pong was noticed early, analysis of the problem needed a while. Then the decision had to be made to restore the whole file system. Such a decision could not be made by system administrators, as it leads to several hours of downtime. Instead the decision had to be escalated to senior IT management. After some time, it was decided to restore the system from backup tapes, and this was done.

In total, a downtime of 10 h occurred for that business-important service. Since 1000 office workers used that service and needed that service for their work, this led to roughly 10 000 lost working hours, which is quite a feat. As a follow-up activity, proper disaster recovery for that service was established, to prevent such long outages in the future.

What happened here is that the marketing hype that high-availability clusters provide round-the-clock availability was believed. Even though the architects knew that there were still single points of failure – not least the files that exist only once in the cluster – management did not allocate enough money to protect against those failures. Only after a disaster happened, fault protection against data corruption was established.

Proper single point of failure analysis, as explained in Chap. 3, would have detected that design flaw. Chapter 6 details more limitations of traditional failover clusters. ■

**Example 5 (System errors in SAP installation).** Many medium-sized and large companies use SAP as their enterprise resource planning (ERP) system. Their whole financial and human resources departments depend on the functionality of those systems: purchasing, invoicing, inventory management, and other areas do not work without them. In addition, all financial planning data is kept in those systems; storing such data is subject to many regulatory requirements.

For a medium-sized business, implementation of a high-availability infrastructure for the SAP server was deemed too complex and too expensive. The system vendor sold a “4-h service contract” and management thought that this is sufficient outsourcing to mitigate outage risks.

After a hardware outage and subsequent database crash, it needed three business days to get a new system in place and up again. One had

overlooked that the “4-h” time limit was about vendor reaction time, not about time to repair. Having all financial data not available for 3 days clearly did not satisfy business expectations. Luckily, postponing invoice creation for 3 days did not harm the company much. ■

**Example 6 (SAP installation that works).** Owing to their importance for business finance, SAP installations are an area where cautious business owners spend money; therefore, they make up not only good examples for outages, but also for well-established and thoroughly engineered solutions that deliver high availability.

Outsourcing companies like EDS operate hundreds of SAP installations in a standardized manner and deliver highly available services for many companies’ financial and human resources departments. They do so by using standardized templates for planning, implementation, installation, and operations. Up times of these services are measured in months or even years, and outages are in the small minute ranges.

Such design, implementation, and operation templates combine a multitude of methods that range from proper vendor selection, choosing the right hardware combination, using cluster technology to provide redundant solutions, and – almost most important of all – defining proper change and problem processes to keep the solution highly available over their lifetime. ■

**Example 7 (Disaster recovery for a snowstorm).** On March 13, 1993, a data center in New Jersey became a casualty of the blizzard that was later dubbed “the worst storm of the century.” A section of the roof collapsed under the weight of the snow and buckled the walls. Fortunately, nobody died and the operational staff were able to perform a controlled shutdown of all systems, and all employees were evacuated.

The outage of this site concerned 5200 automated teller machines (ATM), 6% of the ATMs nationwide. Execution of the disaster recovery plan was started immediately; that called for the creation of an alternative site and relocation of service to that place. First, a temporary data recovery facility with restricted functionality and performance was made operational; within 48 h relocation to a new permanent site had been done. In the meantime, prearrangements with dozens of regional ATM networks kicked in. These networks performed stand-in processing until IT systems and the network were up and running again. ■

### Examples Summary

Let us have a look at the examples with unnecessary outage problems, where we list the root causes for each one. One thing is clear from the outset: a recurring root cause is that there was a single point of failure, i.e., not enough redundancy in the implemented solution, or that redundancy was intended but was designed or implemented the wrong way.

Root cause	Example				
	1	2	3	4	5
Single point of failure			×	×	×
System used beyond design limits	×	×			
Software error		×		×	
Human error			×		
Wrong design assumptions					×

Two other failure causes were close runners-up: that a system is used beyond the limits that it was designed for, and that software errors occur that make all other fault protection methods moot. In fact, many other examples show that human errors also cause major outages quite often; again and again Murphy's Law is proved true. Analysis of more outage examples brings into focus that we find the same fault causes again and again – and that is no coincidence. While there are many details that can go wrong, the basic reasons do not vary so much. Experience shows that the five reasons from the table are representative for a good part of many problems that we know about.



<http://www.springer.com/978-3-540-24460-8>

High Availability and Disaster Recovery  
Concepts, Design, Implementation

Schmidt, K.

2006, XII, 410 p., Hardcover

ISBN: 978-3-540-24460-8