Availability of IS/IT

Ulrik Franke
Department of Information and Aeronautical Systems, Swedish Defense Research Agency (FOI), Stockholm, Sweden

Abstract
The entry introduces the concept of availability of information systems and information technology (IT). Starting from the definitions, it goes on to explain different ways to understand the concept, and how these have resulted in different scientific approaches over the years. Particular emphasis is given to the concept of service availability, and the corresponding notion of managing availability through Service Level Agreements. As IT service outages are very costly in modern society, the entry thoroughly introduces the economics of availability. The main formalisms for mathematical modeling of availability are also introduced.

INTRODUCTION
In modern society, information and communication technology (ICT) is becoming ubiquitous. Modern ICT is transforming our way of life; allowing us to work, buy things, educate ourselves, and meet our friends in ways not imagined two, or sometimes even one, decade ago. However, with these benefits comes an increased dependence on ICT. The consequences of outages – downtime – in ICT systems are becoming ever larger and more difficult to assess beforehand. ICT systems underpin our electricity distribution, the supply chains that bring food to our stores, the financial systems that let us use our money, the booking systems that let us travel, and so on. When these systems go down, modern society grinds to a halt.

The availability of information systems (IS) and information technology (IT) is a measure of the extent to which systems are able to perform their required functions at the time when they are needed. A typical measure is a percentage – a payment system, for example, might have 99.8% availability, roughly meaning that it fails to provide the required functionality 0.2% of the time. Clearly, maintaining high availability is a top priority in many different lines of business.

This entry aims to introduce and explain the concept of availability of IS and IT, including a few cases where this seemingly simple concept becomes more complicated. Furthermore, the reader should become acquainted with the basic statistics and modeling formalisms of availability, as well as some of the problems and questions that are open for research.

STEADY STATE AVAILABILITY ANALYSIS
A simple and widely used equation for availability is

\[
A = \frac{MTTF}{MTTF + MTTR}
\]

where \(A\) denotes availability, \(MTTF\) denotes “Mean Time To Failure” and \(MTTR\) denotes “Mean Time To Repair”. This equation is easy to interpret: the numerator is the time that the system is functioning (i.e., before it fails), and the denominator is that same time that the system is functioning plus the time it is not functioning (i.e., when it is being repaired). The quotient thus represents the functioning time as a fraction of all time. As \(MTTF\) approaches infinity or \(MTTR\) approaches zero, availability approaches 100%.

The concept of a system alternating between functioning and not functioning over time is illustrated in Fig. 1. In this case, \(MTTF\) can be calculated by summing the four TTFs and dividing by four, whereas \(MTTR\) can be calculated by summing the three TTRs and dividing by three.

It is important to note the use of the word “mean” in \(MTTF\) and \(MTTR\). It means that the times used are averages over many cycles of operation and repair. If the system has reached a “steady state,” that is, a state where the statistical distributions of time to failure and time to repair do not change with time, then Eq. 1 gives a good characterization of average system behavior over a period of time. The measure thus defined is referred to as steady state availability.

If, however, the system is not in a steady state, that is, if the statistical distributions of time to failure and time to repair do change with time, then Eq. 1 must be applied with caution. Old measurements of time to failure and time to repair may not be valid any more.
Availability studied under these conditions is called *instantaneous availability*, that is, the probability that the system provides the required functionality at a given instant, rather than over a longer period of time.

Technical standardization bodies such as ISO\(^1\) or ITU\(^2\) often mention both aspects of availability in their standards. For example, the ISO/IEC 20000 (Information technology – Service management) defines availability as the "ability of a component or service to perform its required function at a stated instant or over a stated period of time".\(^1\) In this entry, we will mostly concern ourselves with steady state availability.

### AVAILABILITY OF HARDWARE, SOFTWARE, SYSTEMS, AND SERVICES

The steady state availability concept introduced in the previous section applies to all technical systems – light bulbs and computers alike. However, IS and IT have some special characteristics that set them apart from other technology in ways that are relevant for availability. Whereas a light bulb has just a single function – to provide illumination – modern IS are more complex. In particular, they are complex both in that they consist of many parts (hard drives, service buses, billing software, etc.), and in that they offer many different functions or services (storage of data in a database, access to the internet over the HTTP protocol, perform credit ratings on customers, etc.). How should availability be assessed in such a complex setting?

Indeed, it is possible to analyze many different aspects of IS and IT from an availability point of view. Hardware failure was once the most important reason behind outages. However, over the decades, the share of outages attributable to hardware has steadily fallen,\(^3\) and in the 1980s, IT administration and software had become the main culprits.\(^4\) In modern IS and IT, software failures and human errors have become the main reasons for outages.\(^5\)

### Software Reliability

The traditional scientific approach to studying software failures is found in the discipline of software reliability. Over the decades, many statistical models of software failures and failure rates have been developed, most often trying to describe how error-corrections and debugging efforts have affected the reliability of the resulting (imperfectly) debugged software. This is a mature and important field, and there are many good introductory and more advanced sources, for example, Pham.\(^6\)

However, software reliability models are ill-posed to analyze some interesting and important aspects of IS and IT availability. To see this, we need to reconsider the “repair” word, familiar from MTTR, in the IS and IT context. Repairing a light bulb is straightforward – you spend a minute to exchange it for a new one. Repairing an information system, however, can mean different things. In software reliability, repairing a piece of software typically means fixing a bug. This is the work of programmers, and it is usually something that takes days, weeks, or months to complete, depending on complexity and priority. Once the software has been debugged, a new release is made. However, in the context of an enterprise information system such as an internet bank, repair means something else. If such a system experiences an outage, it has to come back up again, preferably in a matter of seconds, minutes, or hours, depending on complexity and priority. Bugs in the software engineering sense are rarely fixed at this stage – if software failure is to blame, the default strategy is most often a quick roll-back to the last working version. Bugs in the source code can then be fixed offline. This kind of repair, which might more appropriately be called a *recovery* or a *restore*, is often the more relevant kind in order to analyze IS and IT availability in business operations. (This is sometimes called Mean Time to Restore Service – MTRS.) Therefore, it is not necessarily very enlightening to apply Eq. 1 to software failure and repair rates as found in the software reliability literature, if the goal is to support business operations with high availability requirements on IS.

![Fig. 1](image-url) A system alternating between functioning and not functioning over time.
Service Availability

These observations have had important consequences for availability modeling and analysis. System availability can still be modeled as a function of underlying hardware and software availability, but another paradigm has become increasingly popular over the years: service availability. This perspective ties right into the importance of IT for modern society — availability is not first and foremost evaluated from the perspective of software (or hardware) failure, but from the consequences of IT service failures. From this point of view, it is not interesting per se to analyze whether a particular hard drive crashes or a fiber optical cable is severed. The interesting questions are whether the data storage or communications services are interrupted. One way to express this change of perspective is to say that focus has shifted from the internal technical aspects of hardware or software failure to the external consequences of such failures – and their corresponding downtime – for the business operations supported. In the context of service availability, topics such as quality of service-aware service composition, user perceived availability, and component or architecture based availability assessment have become popular in the academic community.

The service-centric availability concept is enlightening in its focus on the consequences of downtime. It also readily offers an operationalization for anyone who wants to measure availability – the relevant downtime is the unavailability noticed by the business. A failure in a payment system that occurs without anyone noticing until the log is checked does not count as service downtime, even though there certainly was system downtime. Similarly, if a single system provides many services, some of which fail (e.g., off-site backup of data) while others remain up and running (e.g., local backup), service availability remains straightforward to assess, even though system availability becomes harder to define. In this way, a mature service catalog with appropriate granularity is a very helpful tool for availability analysis. However, some difficulties remain. If a service (e.g., an e-commerce payment system) is available to most customers, but not to all, should the service be assessed as available or not in the relevant interval of time? Surely, it does not make sense to equate unavailability to 1% of customers to unavailability to 100% of customers – but simply labeling a service with such availability problems “available” might have dire consequences. In this case, the service-centric availability concept does not offer any universal solution. However, it does offer the general insight that definitions of what constitutes available services should be made from the point of view of business operations.

Root Cause Analysis

As part of mature business continuity planning, it is prudent to analyze the causes of recorded service downtime. Such root cause analysis enables learning from past experience and, as such, contributes to better business continuity in the future. For example, if root cause analysis reveals that a malformed message arriving through an integration solution caused a failure, then the external party that sent the message can be notified and/or better exception handling can be implemented in the integration platform. For nonrepairable systems, or for instantaneous availability analysis, this is conceptually straightforward. However, for steady-state analysis of repairable systems over a time interval, there need not be a unique root cause behind an outage. The malformed message can certainly be the immediate cause of failure – had it not occurred, neither would the failure. However, if service has not been restored after a few minutes of downtime, this might also be attributed to inadequate monitoring – had monitoring been in place, an automatic fail-over would have been completed by then. And indeed, if the requirements elicitation process had been sufficiently detailed about the message formatting requirements, the downtime would never have occurred.

The fact that there can be multiple root causes of IT service downtime is no reason not to conduct root cause analysis. On the contrary, this is a valuable practice. However, it does suggest that finding a (first) cause should not always be the end of the investigation. Indeed, while experts believe that lack of proper requirements and procurement practices are a top factor behind downtime, such abstract root causes are not likely to show up in incident management logs. Availability analysts performing root cause analysis should take this into consideration.

SERVICE LEVEL AGREEMENTS (SLAS) AND AVAILABILITY

In modern IS and IT environments, business requirements are increasingly governed and managed “by contract”, namely by so called SLAs. Typically, such agreements cover a wide range of issues, including all kinds of quality aspects that the service-provider undertakes to deliver to the service-procuer. SLAs are not only used in dealings with external service-providers, but also increasingly within organizations, to formalize the agreements between the business and the (in-house) IT department. However, with the advent and increasing popularity of cloud computing, SLA management has become more important than ever, as these contracts
increasingly become the only interface between the business and its IT support.

In terms of availability, a good SLA should reflect the needs of the business. Most often this is defined as a percentage, for example, 99.8%, reflecting steady-state availability over a time interval, as defined in Eq. 1. Sometimes, this is the only requirement on availability specified. This is not to be recommended. The reason is clear from Eq. 1: A given availability can be achieved by a lot of different combinations of MTTF and MTTR. For example, 99.8% availability 24 hours a day, 7 days a week means almost 18 hours of annual downtime. But clearly, there is a difference between a single 18 hour outage and some 200 separate 5 minute outages. Most companies would not be indifferent between these alternatives – shorter or fewer outages – even though the differences correspond to the same availability percentage. For example, the operator of an industrial plant with extensive supply chains, and a physical production process that is expensive to restart, probably prefers as few outages as possible even if they are longer, whereas a retailer for whom an 18 hour outage in the payment system the day before Christmas might mean the loss of the entire annual profit probably prefers as short outages as possible, even if they are many.9

To incorporate such considerations into an SLA, it is recommended to include constraints on the allowed time to service recovery (sometimes known as a Recovery Time Objective) as well as constraints on the number of service interruptions allowed. Together, two such constraints entail a worst-case availability in a time period, with the worst-case downtime being determined by the maximum number of outages allowed times the maximum outage duration allowed. Typically, the specified steady-state availability in the SLA should be better than this worst-case availability, in effect adding an additional constraint on the performance of the service-provider: every single outage cannot last for the longest acceptable duration.

Typically, other SLA objectives related to availability are also required. Recovery Point Objectives specifying the point to which data can be recovered are important for most lines of business, but the requirements can vary considerably. In a stock exchange trading system, each and every transaction made must be recoverable if the system crashes, whereas in a small business office environment, it might be enough to be able to recover yesterday’s spreadsheets and text documents. Appropriate availability requirements must be based on a thorough understanding of the supported business operations.

**PLANNED AND UNPLANNED DOWNTIME**

When assessing availability, it is important to make sure that it is assessed with regard to the relevant operating time. For IT systems or services running continuously 24 hours a day, 7 days a week, the operating time is simply the same as calendar time, but for many services, this is not the case. Even an extremely availability-critical system such as a trading system on a stock exchange shuts down at the end of the trading day, only to reopen in the morning again. For such systems, it is straightforward to plan maintenance such as hardware or software upgrades to occur outside operating hours. Such planned downtime is required, at one point or another, for all systems and services. For services running continuously, redundant instances are required for upgrades, etc. Thus, one redundant system can be upgraded while another runs, and vice versa, maintaining the service without interruption.

Unfortunately, planned downtime sometimes leads to unplanned downtime, for example, if an upgrade takes longer than expected, or if an upgrade has been insufficiently tested before it is taken into operation. These are not uncommon scenarios, and insufficient change control or configuration management has been identified by experts as a top cause of unplanned downtime.7 Therefore, it is prudent to add substantial safety margins to planned downtime, for example, when allocating service windows. Unfortunately, realistic testing of new configurations or upgrades to complex modern IS is not always possible in a test environment, making it virtually impossible to guarantee that a change will proceed without incident.

The risk of triggering unplanned downtime in relation to planned maintenance has important consequences for availability management. It is not uncommon to freeze the configuration of an IT system or an orchestrated set of IT services for the duration of particularly sensitive operations that must not suffer outages, such as during major product releases or transaction peaks such as Christmas in the retail sector. Here, it is interesting to observe a difference from traditional mechanical systems, where before sensitive operations old components can be changed for new ones earlier than they normally should. Such preventive maintenance before it is due is reasonable in systems where components wear out, and a new component is (almost always) better than an old one. However, software does not wear down as mechanical components do, leading to diametrically opposed practices in order to ensure high availability throughout a particular time interval.

**THE ECONOMICS OF AVAILABILITY**

So far, we have discussed availability from a *ceteris paribus* perspective: More is always better. However, in the real world all else is not equal, and trade-offs need
to be made between the costs of achieving higher availability and the resulting benefits.

The Costs of Increasing Availability

The first component of such a trade-off is an understanding of the costs involved in achieving increased availability. There is no simple way to calculate these costs in the general case, though rules of thumb are sometimes proposed. What is clear, however, is that availability exhibits diminishing returns, i.e., when investing to increase availability; every percentage point will come at a greater cost than the last one. This conceptual relationship between investment and availability is illustrated in Fig. 2. A concrete example of what such a relationship might look like in practice is the Gartner rule of thumb that 99.3% availability for a “standard IT service” costs 2.15 times the cost of the standard service, whereas 99.81% availability costs 6.45 times the cost of the standard service. While the exact figures should be taken with a large grain of salt, the fact that availability investments exhibit diminishing returns is indisputable.

Eq. 1 helps us understand the diminishing returns on availability investments. In order to increase $A$, Eq. 1 offers two possibilities: i) increase MTTF or ii) decrease MTTR. Any improvement in steady-state availability has to be the result of at least one of these. The standard (conceptual) way to increase MTTF is to build some kind of redundant solution with fail-over functionality – having two components rather than a single one. But for his method to double MTTF, a fail-over mechanism is needed for switching from one component to the other precisely when needed, and this comes at an additional cost, over and above the double component cost. (And indeed it only works if there is no common cause of failure for both components.) So in general, there will be diminishing returns on investments in the increase MTTF strategy. Similarly, the standard (conceptual) way to decrease MTTR is to hire more people to do the repairs – having two system administrators rather than a single one to make systems repairs or service recoveries when needed. Doubling cost in this way might decrease MTTR by half – if the two repairmen can always work in parallel on a problem, and if it is never the case that one of them is idle while the other one is working. So in general, there will be diminishing returns on investments in the decreased MTTR strategy. These particular examples are simplified, but the principles hold true, and help explain the diminishing returns on availability investments.

How should we go about increasing availability? By increasing MTTF, by decreasing MTTR, or by a combination? This question is relevant to any decision maker, whether there is a fixed budget that should be spent so as to maximize availability, or if there is a fixed availability goal that should be reached at a minimal cost. Of course, this question typically cannot be answered in a precise way, because there is no reliable way of exactly predicting the availability that will result from particular investments. Nevertheless, it is a conceptually enlightening question to ask. The answer is that it depends on the prices of the two strategies. If we simplify a bit and assume that increasing MTTF translates to buying more or better technology, and that decreasing MTTR translates to hiring more or better personnel, then increasing MTTF will be a good strategy whenever technology is cheap, whereas decreasing MTTR will be a good strategy whenever labor is cheap. In microeconomic terms, the technical rate of substitution between technology...
and labor should equal the economic rate of substitution between them.\[11\]

The Benefits of Increasing Availability

The second component of the trade-off is an understanding of the benefits of increased availability. The most important factor here is the outage costs: How much does an hour of downtime cost, or equivalently, how great is the benefit of preventing that outage? A straightforward but useful formula for calculating the average cost of an hour of downtime is the following:\[12\]

\[
\text{Employee costs/hour} \times \% \text{ Employees affected by outage} \quad + \quad \text{Average Revenue/hour} \times \% \text{ Revenue affected by outage}
\]

\[= \text{Estimated average cost of 1 hour of downtime} \quad (2)\]

The idea is that the costs of downtime either come from lost productivity (employees who get paid but cannot work) or from lost revenue (sales that never happen). These kinds of calculations are the basis of estimates such as IBM’s claim that IT systems downtime cost American businesses $4.54 billion in 1996.\[13\] (It is surprisingly hard to find more up-to-date estimates of downtime costs that are not very limited in scope. Nevertheless, it is clear that downtime costs are large, and that they have most probably grown considerably during the last 20 years, as society has become ever more dependent on IT.)

It is important to note the word “average” in Eq. 2. Whereas the hourly salaries of employees are known and fixed, the hourly revenues are more difficult to assess. This contributes a lot to the uncertainty inherent in downtime cost calculations. In particular, even if an average hourly revenue is known (e.g., calculated based on last week, month, or year) this average may be way of the mark if there is a lot of variance. For example, hourly revenues in the retail sector can vary by more than a factor of 10, depending both on hour of the day, day of the week, or proximity to major holidays. Taking the variance of outage costs into account, rather than just focusing on averages, is important for prudent decision making with regard to availability.\[9\]

There is also another simplification involved in Eq. 2, worth noting. It assumes that outage costs scale linearly with the duration of the outage: On average, a two hour outage costs twice as much as a one hour outage and a ten hour outage costs ten times as much as a one hour outage. This is a good first approximation. However, there are clearly instances when this is not the case. Some IT systems have large fixed restart costs. Perhaps the best examples are industrial control systems, where every outage entails the halt and restart of a physical industrial process. If the fixed costs of a restart are sufficiently large, the difference between a one hour and two hour outage might not be so great. Other IT systems have snowball effects, where a short outage may go entirely unnoticed by customers, but a longer outage has severe consequences on brand goodwill and customer loyalty. Perhaps the best examples are payment systems or automated teller machines (ATMs), where a short outage typically just results in the customer swiping the card again, but a longer outage may prompt other customers standing in line to leave the store, or even drive customers to other banks. Again, there is no universal solution that fits all organizations. Decisions about what kind of availability profile to strive for need to be made from the point of view of the business operations that are supported by the IT services.\[9\]

Only when both the costs and benefits of availability investments are understood is it possible to make an enlightened trade-off between them.

MODELING FORMALISMS

The most straightforward way to model availability in complex systems is to use Fault Trees, a technique adopted from reliability engineering. A fault tree depicts a system composed of constituent components, each with an availability level of their own. The total availability of the system can then be calculated, depending on whether the components are redundant (the OR
relation) or not (the AND relation). An example is given in Fig. 3.

Fault tree analysis is an old discipline with roots in reliability engineering, important challenges still remain. These include not least the economic aspects and trade-offs involved in decision making with regard to availability. There are efforts underway to turn this kind of "service level engineering" into a more mature discipline. This work will certainly evolve over the years to come. So will the search for modeling formalisms and practices that balance the needs for simplicity and usability with the needs of decision makers to get appropriate analytical insights. A third important strand is empirical studies, which provide important insights into the state of the practice.

ACKNOWLEDGMENT

The author wishes to thank Dr. Johan König, who gave a number of useful comments on this entry.

REFERENCES


