the Availability Digest

www.availabilitydigest.com
@availabilitydig

The High Availability Design Spectrum – Part 1

Dr. Terry Critchley
December 2016

[Editor's Note: In his book "High Availability IT Services," Dr. Terry Critchley lists twenty-three areas that can have an impact on the availability of IT business services. In this multipart series, and with his permission, we publish his observations. Part 1 of this series reviews his first four reflections in his Parts A through D.]

Dr. Terry Critchley: Most of the documentation on HA/DR I have come across majors on hardware, mainly redundant or fault tolerant, and, to some extent, software. My thesis is that the spectrum of activity needed to design, implement and maintain a high availability business IT system and recover from failures small and large (DR) is much, much greater. Below, I have listed 23 areas (A to W) which can have an impact on the availability of business services which are IT-based. I am sure it will be evident that these areas can have a significant impact on the availability and non-availability of any service or system.

Remember, focusing on availability and focusing on avoidance of non-availability are not the same thing, if you think about it.

The book and chapter references following refer to 'High Availability IT Services': https://www.crcpress.com/High-Availability-IT-Services/Critchley/9781482255904.

A. Availability by Systems Design/Modification

Highly available systems need to be designed or existing systems modified to meet the needs of crucial applications. The design stage with the tools and techniques which can help in this area are covered in Chapter 7. The design stage can be aided by judicious use of Reliability Block Diagrams (RBDs) and some of the techniques and methods outlined in Chapter 7. Where feasible, relevant calculations should be employed and documented.

B. Availability by Engineering Design

This area lies directly with the hardware and software engineering functions within the hardware vendor organization but will cover things like ECC, R.A.I.D., hot swap devices, automatic software recovery (shadow) and so on (essentially RAS). Nowadays, these features are offered by most hardware vendors who want to stay in business.

You can't really influence or use these features - it's just nice to know they are there. Software 'engineering' is the responsibility of the vendors of commercial off-the-shelf software and middleware; the responsibility for in-house software reliability lies with the organization itself.

Self Healing Hardware and Software

A useful (but marketing) article by IBM gives us the following information about self healing, generally talked about as part of 'autonomic computing'. In a nutshell, self healing hardware applied to 'chips' often means that built-in conducting material is built in and used to 'seal' gaps in broken circuitry.

> "Autonomic computing will help customers reduce the cost and complexity of their e-business infrastructures and overcome the challenges of systems management using systems that self-optimize, self-configure, self-heal and self-protect. System z plays a major role in the IBM autonomic computing initiative, since self-management capabilities available for System z will function as a model for other IBM servers. z/OS provides ... functions to address the goals of the IBM autonomic computing initiative".

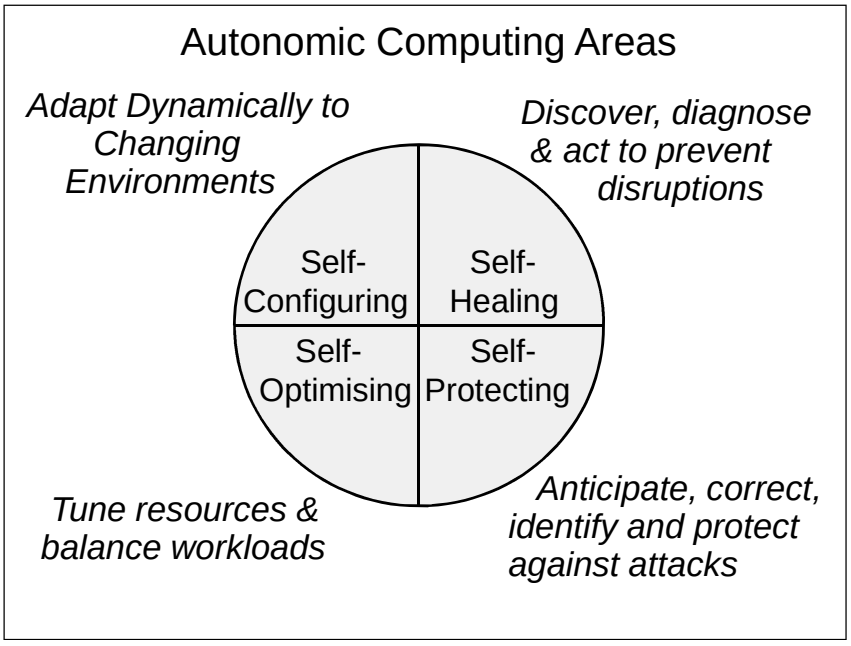

Figure 1: Autonomic Computing Areas (The Goal)

Self-Healing and Other Items

In an article[1] from nearly 20 years ago, Jim Gray stated *'Administration and maintenance people are doing a better job than we have reason to expect ....The only hope is to simplify and reduce human intervention in these aspects of the system*.' Was he an autonomic computing sage?

Although a level of operations automation is available today, full autonomic computing is still a goal with the following requirements, amongst others**:**

- Detection of various faults and even automatic switching to backups where available (Chipkill memory, ECC cache, Central Processor, Service Processor, system bus, Multipath I/O, etc.).
- Plug and Play and Hot swap I/O
- Capacity Upgrade on Demand
- Intrusion Detection Services
- System Automation It provides policy-based self-healing of applications, system and sysplex resources

[1] Why Do Computers Stop and What Can Be Done About It? TANDEM Technical Report 85.7 June 1984 PN87614 (Jim Gray)

- Security features such as LDAP, Kerberos, SSL, digital certificates, and encryption

In a self-healing software system[2], the system monitors itself for anomalous behavior (after the 'norm' has been established). When behavior is detected, the system enters a self-diagnosis mode that aims to identify the fault and extract as much information as possible as to its cause, symptoms and impact on the system (not the service, which it does not understand). Once these are identified, the system tries to adapt itself by generating candidate 'fixes', which are tested (assessed) to find the best candidate state.

The nearest we get to autonomic computing today is automation of various functions and some human intervention in others.

C. Availability by Application Design

It is possible to increase the availability of systems by judicious application design. Two examples are shown below - using message queuing and implementing local spooling (and possibly error checking) in a client/server environment. Although the server part of the application is not available, recovery can be effected faster when it does become available - no re-entry of transaction data and so on.

Poor Application Design

Conventional Programs: For internally developed systems, the design and coding obviously have a bearing on the correct functionality and availability of the applications supporting business services based on IT systems. This is in the province of software availability and QoS (quality of service), covered in Chapter 6, under 'Software Reliability'.

In general, experienced users will try to avoid poorly designed systems, which are sometimes developed without their input in terms of functions and usability. I have seen such users revert to manual methods where possible to avoid trying to use the unusable.

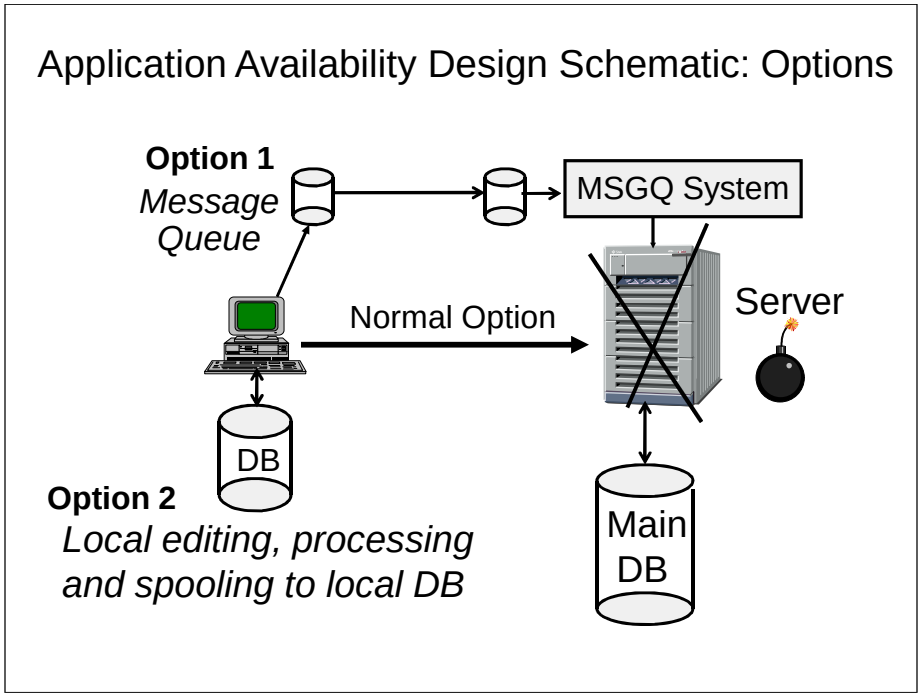

Figure 2: Availability by Design - Two Sample Options

[2] A.D. Keromytis, http://www.cs.columbia.edu/~angelos/Papers/2007/self-heal.*pdf*

Web Applications: An illuminating article[3] lists 10 faults designed 'to drive away your readers' from a customer-facing web site (Aug 2012)**:**

1. Use impossible navigation on your site (see next diagram)
2. Put up more advertisements than content
3. Do not maintain your site [*refer clients to other, non-existent sites*]
4. Require [*ask for*] too much information
5. Be too cutting edge [*require clients to be rocket scientists or Philadelphia lawyers*]
6. Crash your customers' browsers
7. Use sound on a business site *[except as a valid, requested webinar for example*]
8. Put up too many images
9. Embed too many tables
10. Pay no attention to your readers' [*needs/feedback*]

These 'failures' can cause a *logical outage* or, at the least, a partial outage by sending some or all potential customers (or internal staff) away from your site. One such site, guaranteed to lose a company business, and possibly the designer his job, is illustrated below in all its over-the-top glory.

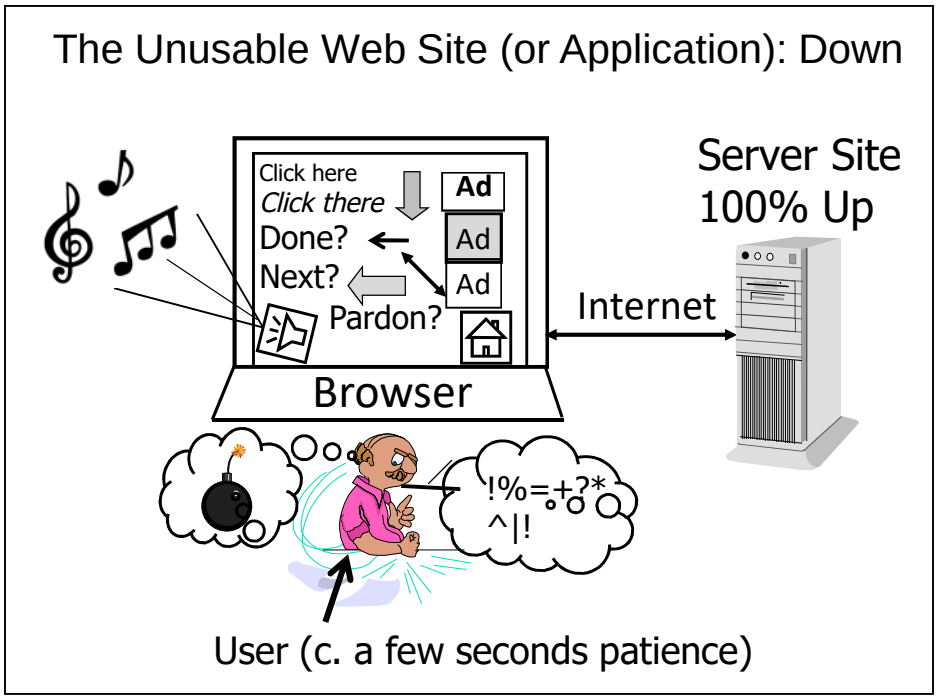

Figure 3: Non-availability: The Unusable Web Site or Application

I am a recognised authority on unusable and irritating web sites which are, for all intents and purposes, 'down' because they are not delivering the service they were designed for and possibly divert me to other sites to transact my business.

This 'poor design' thesis is supported by the 2012 report *'How to Stop Webpage Speed from Killing Your Marketing'* by Limelight Networks, Inc., which indicates that when web site page loading is slow, 62% of mobile device users abandon their sessions; and nearly one third of them never return. Not good for business. What does that suggest to us?

[3] webdesign.about.com/od/webdesignbasics/tp/aa122101a.htm

Later Research: The 7 second *'patience limit'* in the diagram above may be too optimistic. In 2009, a study by Forrester Research found that online shoppers expected pages to load in two seconds or fewer — and at three seconds, a large share abandon the site. Only three years earlier a similar Forrester study found the average expectations for page load times were four seconds or fewer. See the NY Times article below where it talks about 250 milliseconds being the *'patience limit'*:

http://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html?pagewanted=all&_r=1&

Monitoring performance and user experience at the sharp end of things should be mandatory for designers and operators of web sites. Marketing people, too, have a major part to play in this as well, although you may have to advise them since they are unlikely to have read this book.

D. Availability by Configuration

Hardware

Some of the ways that the availability of vendor servers, associated hardware and databases can be improved include:

- hardware redundancy, for example RAID, memory mirroring
- putting CPUs on different boards wherever possible
- putting memory on different boards wherever possible
- putting I/O cards on different system boards wherever possible
- connecting redundant resources to different power supplies
- hot swap/plug facilities for cards and peripherals
- redundant operating system boot disks
- duplicate consoles
- failover software (operating system, middleware, applications)
- clustering[4]
- employing fault-tolerant hardware and associated software
- use of monitoring software independent of the system being monitored
- other redundant 'ecosystem' devices (fans, sprinklers etc.)

The availability from basic hardware RAS to full fault tolerant features come at a cost as shown schematically in the next diagram. It may be that the level of availability aimed for in an enterprise depends on the final TCO. This diagram does not emphasize the other factors, both physical and logical, which can affect the users' view of the *usable* availability of a service which is governed by an SLA.

Data

- using R.A.I.D techniques, weighing safety against recovery times and operational update times (this affects response times of transactions)
- disk mirroring - local or remote (geographic)
- table layout, for example partitioning, as in Oracle, for example
- duplicated logs and recovery files on separate drives
- backup/archive software, part of DR
- data base 'image' copy software (local, remote)

[4] There are various forms of clustering used here: active/cold, active/warm, active/hot and active/active/standby as we discussed earlier.

- redundant backup hardware
- worst case scenario - key it in again from the beginning!

Networks

- redundancy
- ISDN backup
- alternate/virtual routing

Operating System

- recovery features (shadow OS, OLTP)
- fast reboot after failure
- automatic restart
- journal file system
- live OS maintenance facilities
- logging, backup, recovery utilities
- post dump analysis

Environment

- UPS (Uninterruptible Power Supplies, sometimes a mass of batteries)
- separate electricity supply from utility or own generators
- over-configured cooling, fire sprinklers, Halon gas, etc.
- machine environment cleanliness
- As outlined before, it is not good economics to configure a system and network for availability where the costs of doing so outweigh the benefits. In most situations, high availability solutions will be designed and managed on a financial 'need' basis, taking account of the criticality of the applications to the business and the consequences of their failing.

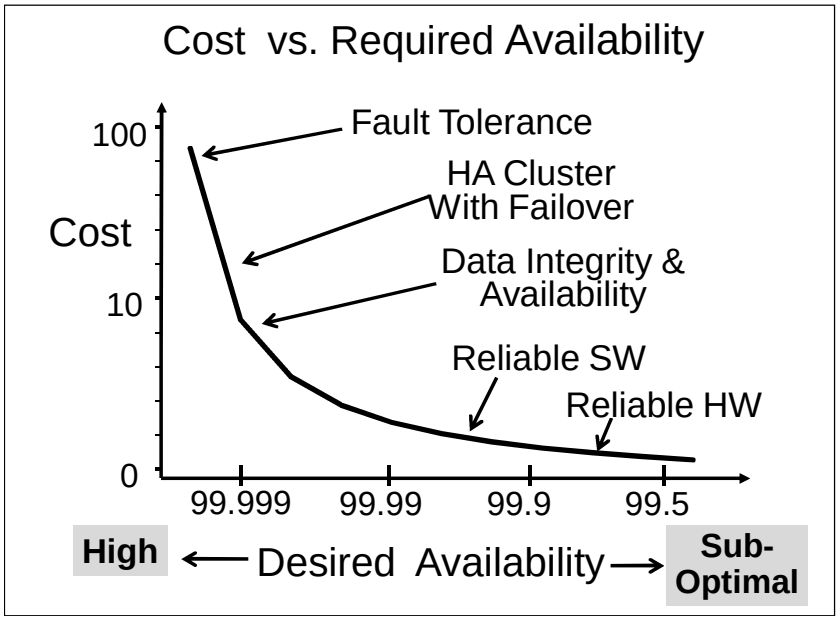

Figure 4: Costs of Reliable/Available Configurations

The schematic above (based on an IBM Redbook diagram) shows the cost effects of opting for configurations yielding higher and higher availability, but not to scale, just indicative of possible steep

increases in cost for little availability return. It is opportune at this stage to emphasize that spending vast amounts of money on hardware and software high availability will be to no avail if the service provided is unavailable or not performing according to requirements for other reasons.

By all means focus on the hardware/software area but not at the expense of other areas as outlined *ad nauseam* in this book.