Investing in Software Resiliency

Dr. C. Warren Axelrod
U.S. Cyber Consequences Unit

Software is inherently error-prone and such errors can lead to failure of those systems of which the software is part. On the other hand, with software being only one of many components of a system, there are many choices in regard to attaining a particular level of system resiliency, not all of which are software-related. It is important to consider software resiliency in relation to the resiliency of the entire system, including the human and operational components. The goal of this article is to help those who develop, implement, and operate computer networks and systems in determining the factors to include when investing in software resiliency.

What is software resiliency? How can it be achieved? What does it cost? What are its benefits and how can they be measured? Is an investment in software resiliency worthwhile? These questions might appear simple, but none of them have an easy answer.

The very concept of software resiliency is frequently ambiguous. For example, are we talking about the inherent or intrinsic resiliency of the software itself, the resiliency that the software imparts upon other components of the system, or both? There is a significant difference in requirements based upon how one defines software resiliency. There is even a question as to what software to include (e.g., end-user applications, system software, embedded firmware) and what to exclude. In this article, I will define software resiliency, examine how it fits into the overall resiliency agenda, and show how one might determine an appropriate level of investment in it.

Background

According to [1], 57 percent of about 1,200 responding organizations experience one or more application failures per month, resulting in user inconvenience or business disruptions. Interestingly, the survey shows that larger organizations tend to have more failures on average, which is thought to be due to the greater complexity in larger environments.

Application failures resulted in decreasing order from software component failure, failure or reduced performance of networks, and physical component failures through power outages and brownouts. Major reasons for application failures include inadequate configuration or change management, system sizing or capacity planning problems, IT staff errors, patch management issues, and security breaches. Another finding was that (for the most part) expenditures on resiliency are not made early enough in the application development life cycle. By delaying consideration of resiliency until late in the cycle, the costs are much higher and there are often insufficient funds to do the job.

For the purposes of this article, software includes any programs that are developed through a regular development life cycle, such as the software development life cycle. This applies whether the end product is a set of *soft* computer program code, firmware (which is program code etched into hardware), or even programmed hardware[1].

> *"The very concept of software resiliency is frequently ambiguous ... are we talking about the inherent or intrinsic resiliency of the software itself, the resiliency that the software imparts upon other components, or both?"*

Resiliency

In [2], the authors define a resilient system as one that can *take a hit* to a critical component and recover and come back for more in a known, bounded, and generally acceptable period of time.

This definition raises as many questions as it answers. Taking a hit can result from accidental activities or intentional attacks: It is when unauthorized damaging activities cause the system to fail noticeably and invoke some form of recovery-and-repair process. A hit can be as simple as a PC freezing and having to reboot it to a complex event that may take a long time and many resources to examine forensically and respond appropriately.

From a more general perspective (particularly when it comes to economic evaluations), one is interested in both how resistant the system is to events that threaten to cause it to fail, and how quickly the system can be brought back to an acceptable level of functioning.

In order to evaluate software resiliency sufficiently, one must always include the environment in which the software operates. The resilience of systems containing a particular piece of software will vary considerably within a particular context.

User View of Availability

There are a number of situations in which a system can be considered not to be available. *Unavailable time* is defined in [3] as the time during which any of the following takes place:

- The system fails to operate.
- The system fails to operate in accordance with formal specifications.
- The system operates inconsistently or erratically.
- The system is in the process of being maintained or repaired.
- A hardware or software component of the system is inoperative, which renders the entire system useless for user purposes.
- The system is not operated because there is a potential danger from operation of the system to employers or employees.
- There is a defect in software supplied by the manufacturer.

This is a more realistic view of availability since there are frequently arguments between the user population and the support technologists as to the real status and usability of a system. Therefore, it pays to be as specific as possible.

Software Resiliency

We now consider resiliency as it specifically pertains to software, as I have defined. First we look at those factors which

reduce resiliency. We then look at specific design and development attributes that affect software resiliency.

Factors Working Against Software Resiliency

The introduction of [4] provides a number of factors and trends that impact software trustworthiness. Many of these factors also affect software resiliency. What follows are some of the broader issues from [4] as well as some additional factors to consider:

Complexity

The size and complexity of software systems is increasing, thus the ways in which a system can fail also increases. It is fair to assume that the increase in failure possibilities does not bear a linear or additive relationship to system complexity. For example, combining two or more systems leads to a greater level of complexity than the combination of the complexities of the individual systems. Thus, if System A has a complexity of 5 and System B a complexity of 7, the combination of Systems A and B will be significantly greater than 12—perhaps in the 20 range.

This complexity attribute makes it increasingly difficult to incorporate resiliency routines that will respond effectively to failures in the individual systems and in their combined system. The cost of achieving an equivalent level of resiliency due to the complexity factor should be added to that of the individual systems.

Interdependency and Interconnectivity

Interdependency or interconnectivity via ever-larger networks adds to complexity in that as systems become increasingly interconnected and interdependent, achieving resiliency becomes a greater task. Another aspect of interconnectivity is the growth in infrastructures that contain systems belonging to different organizations. Thus, the resiliency of an entity's systems is increasingly dependent on the resiliency of systems over which the entity has no control. This means that a failure of another party's systems can have a ripple effect on your systems.

In order to protect against this situation, an entity must develop routines that preserve the integrity and operational continuity of its systems even if the systems of business partners, service providers, and customers were to fail.

Net-Centricity

This is somewhat similar to the interdependency case, except that it focuses on systems that include the Internet or other public/private network as part of its design. For example, service-oriented architecture and software as a service fall into this category, as do a whole range of so-called Web 2.0 applications and services. Again, the issue is whether the systems and networks not under the direct control of the customer organization can be trusted, and what evidence is available to verify such trust. In such situations, there is a need to ensure that software components can be trusted to interact securely without supervision [4]. It should also be noted that security assurance has to cover resiliency and integrity as well as confidentiality.

Globalization

With the growth in increasingly extended software development supply chains, the concern is that the focus will be more on functionality and low cost rather than resistance to attack and resiliency. The challenge is to spread the knowledge as to how to design and build more secure and resilient systems to the far reaches of the development universe and to enforce standards. It is essential to introduce mechanisms that reward such aspects as security, resiliency, and integrity rather than only functionality and speed to market.

"The time that it takes to recover depends mostly on the degree of preparation made through business continuity and disaster recovery plans."

Open Source Software

To some extent, open source products are the software equivalent of the interconnectivity and net-centricity aspects of networking in that there is not necessarily a specific group to go to in order to ensure trustworthiness and resiliency and resolve any failures. It is true that there are *communities* that are responsible for the evolving and fine-tuning of the software (and some of the open source software that is supported by commercial enterprises). However, as shown in a recent study by application security firm Fortify, these groups may not be responsive [5]. Another challenge raised in [4] is the funding of evaluations of such software. There has been some movement in regard to the latter, such as the Software Assurance Initiative being conducted for the banking and finance sector by the Financial Services Technology Consortium in collaboration with the Financial Services Roundtable[2].

Hybridization

Hybridization relates to the increasing trend of combining into single systems software of different origins, and subject to different development methodologies, time and cost constraints, and so on. Thus commercial and government off-the-shelf software, custom and proprietary software, and open-source software may be combined in various ways in the ultimate realization of a particular system. One could argue that such a system is, as a result, only as resilient or secure as its weakest component. This aspect of context is key when attempting to evaluate the combined resiliency or security of a complex system.

Rapid Change

The common belief that change is the only certainty is particularly true in the software arena, where new versions of existing software and frequent releases of new software make for a very dynamic and highly complex environment. Such rapid change creates innumerable problems with software security and resiliency. There is often not the time to test one version of a software product before a new one appears, making the tests on the original software obsolete. A frequently held criticism of Common Criteria testing is that, by the time the results are available, there is a good chance that the tested software has already been replaced.

The danger here is that the new software may contain new vulnerabilities that may not have existed in prior versions. Thus, determining that an obsolete piece of software is sufficiently resilient is not particularly indicative of the state of the newest version and, therefore, is not very useful.

Reuse in Different Contexts

As organizations are being driven by economic and speed-to-market considerations, there is a tendency to increase the use of off-the-shelf and open-source software. While such systems may have been designed to operate in a specific environment, they are being increasingly used in situations for which they were not designed. As a result, they may not meet

the security and resiliency requirements of the new environments.

While one might be cynical in interpreting the standard software use agreement (that protects the software vendor against virtually any liability if the software doesn't do as intended), there is a valid argument about it not working when used inappropriately. This is particularly true of lightweight software applied to critical large operations uses.

Specific Design and Development Issues

There are many situations in which systems fail because they do not even incorporate necessary resiliency routines, or the ones that are inserted do not perform as needed or have not been tested thoroughly enough.

Poor Design

Regarding the absence of resiliency routines, I recall a development manager expressing amazement at the general lack of understanding—both by the presenters of a new product and the audience—of the need to design-in the ability to restart a program from a prior status. The developers were relatively new to the profession.

Inadequate Testing

In another situation, I was about to implement a leading-edge digital telephone turret on a newly built trading floor. The only other installation to date was experiencing intermittent crashes. After weeks of research, the turret vendor determined that the reason for failure was an untested error routine. Apparently, in the pristine and carefully engineered test version at the vendor's testing laboratory, the system did not invoke this particular routine because of the high quality of the installation. Out in the *real world*, the less well-engineered cable runs began generating errors that forced the software into the untested error routines leading to the consequential crashes.

Inappropriate Use

Another resiliency issue arises when the software is used incorrectly or is inappropriate for a particular purpose. Software for the PC is generally not as reliable and does not have the same fail-safe design as software intended to be used in a demanding production environment—yet such unreliable software regularly becomes incorporated into critical production or financial systems. These systems are not held to the same standards for testing and documentation as are major production systems and, as a result, can be the *Achilles heel* of the overall system.

Ineffective Change Management

In order to maintain a high level of application security, integrity, and resiliency, it is necessary to carefully control the software change process. There are many instances where programming errors can result in major failures.

Table 1: *Protection, Costs, and Benefits for Different Types of Events*

Type of Event	Protection	Costs	Benefits
Component failure	• Hardening • Fault tolerance • Redundant components	• Additional components • Software overhead • Hardware overhead • Increased complexity • Maintenance and support	• Increased availability • Reduced downtime
System failure	• Fail-over • Redundant systems		

Site down	• Off-site backup
	  • Hot
	  • Warm
	  • Cold
	  • White-wall
		• Facilities
		• Systems
		• Networks
		• Staffing
		• Utilities
			Ability to restore operation when primary facility inoperable with minimal downtime

Regional disaster	• Out-of-region backup
	  • Hot
	  • Warm
	  • Cold
	  • White-wall		

National or global catastrophe	• Out-of-country facility • Catastrophe contingency planning and backup	As for regional disaster	Ability to recover from a disastrous event affecting large regions of the country or the world.
All off-site backup	• Backup in the cloud	As for some on-site and all off-site	• Ability to purchase amount of resources for backup as needed • Largely independent of location

As an example, on January 15, 1990, AT&T's long-distance network failed and was down for nine hours. The failure occurred when a system-wide software upgrade was installed on 4ESS digital circuit switches. It was reported that the failure began when a switch in New York City suffered a minor hardware glitch, which caused it to go offline [6].

While scheduled changes can clearly cause problems, unscheduled or emergency changes represent an even greater danger to the integrity and continuing operation of software.

Fault Tolerance and Failure Recovery

Anderson points out that "... failure recovery is often the most important aspect of security engineering, yet it is one of the most neglected" [7].

Fault tolerance is the ability of the software to resist damage or destruction from errors. Thus, if there is an error condition, the software has the capability of recognizing the error and correcting it according to some pre-specified set of rules. The tolerance level is only as good as the rules. Therefore, the software, on recognizing an error, will correct it with the most likely correct condition. There is, of course, a possibility that the correction is not appropriate, in which case either the integrity of the system is called into question or a subsequent test will reveal that the attempted correction was inappropriate.

In other cases, if the fault is thought by the system to be a component failure, the fault tolerance results in automatic switching to a backup component or software routine. The system continues processing in backup mode while the faulty component is being fixed. This latter situation is failure recovery within the primary system.

Fail-Over to Other Systems

Fail-over can also be to an on-site or off-site backup system. While fail-over within a system usually assumes operational continuity, fail-over to backup systems can be hot, warm, or cold.

If hot, the backup system is running in parallel with the primary system and automatically detects a failure in the primary system and switches to the backup, which may be on-site or off-site. If off-site, it can be in-region, out-of-region, or *in the cloud*. There are often technology restrictions on the allowable distance between sites for hot backup. One common limitation comes from the technical feasibility of maintaining data current at two or more sites via disk shadowing or similar technologies.

Recovery and Restoration

The ability to resist *attacks*—and to recover quickly to an acceptable level of performance after failure due to successful exploits, unintended damaging actions, or accidents—is crucial for the systems running in most organizations.

The time that it takes to recover depends mostly on the degree of preparation made through business continuity and disaster recovery plans. There are escalating levels of backup and recovery, each costing more but enabling improving recovery from increasingly destructive events. These levels are shown in Table 1. The table also shows the various forms of protection that can be instituted and their respective costs and benefits.

In the commercial world, unavailability costs might include loss of productivity for internal users and business partners, loss of business in the form of failure to attract new customers or retain existing customers, and so forth. In the government sector, lack of availability might result in military compromise or a reduction in safety. While difficult, it is necessary to come up with cost estimates related to unavailability of critical systems. These costs will typically not be easy to estimate. They will also typically not be linear, but more in the form of exponentially increasing costs.

In terms of recovery costs, these are usually minimal when recovery involves a *hot backup system* or *facility* where switching or fail-over to the backup system—whether on-site or at another facility—is virtually instantaneous and there is no loss of data or processing availability. Such a transition is effectively transparent to end-users and business partners. Of course, a hot backup is considerably more expensive to design, implement, and maintain than other forms of backup.

Warm backup is where the backup system or facility is up and running and on standby and can be brought into operation within a short time, typically minutes. The recovery time usually consists of a process for bringing the backup system up and synchronizing it to the point in processing at which the primary system failed. The activation of such a process usually takes from minutes to hours to accomplish and the time when the switchover takes place (i.e., whether the system is in use or idle at the time of failure) can have a significant impact on end-users and business partners.

It is interesting to note that hot backup is not always better than warm backup from operational and availability perspectives. I recall a situation in which two sister organizations had taken different approaches to achieving high availability for critical financial systems. The larger, wealthier organization had both hot backup and warm backup, and a process whereby the warm backup system was activated to hot status if the primary system failed over to the hot backup. The smaller, less affluent organization ran two separate systems in parallel and, in the event of a failure of the primary system, physically switched to the backup system. This resulted in having to reenter the few missed transactions that were lost in the switchover. It turned out that the highly automated larger systems were considerably more expensive and far less reliable than the simpler manually operated systems. This was because, in the highly automated case, an error occurred in common memory resources, which brought down all three systems for an extended period. The lesson learned was that one has to be aware of single points of failure and how they might impact the recovery process.

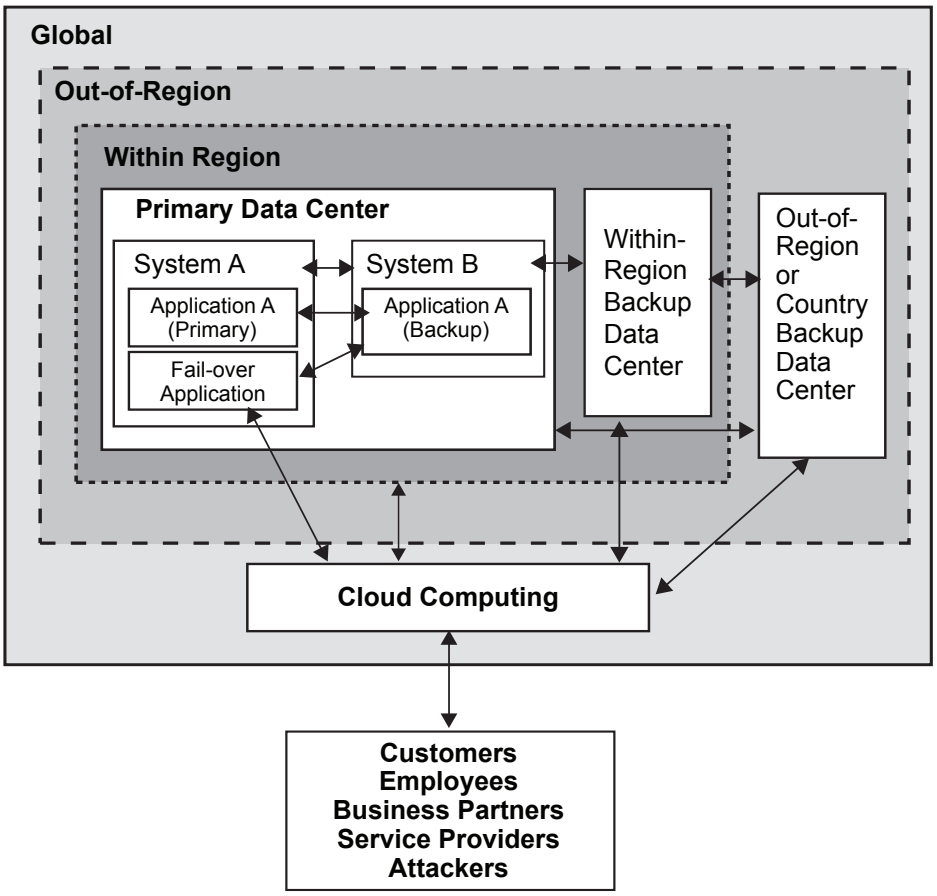

Figure 1: *Various Degrees of Backup at Application, System, and Facility Levels*

With regard to backup sites, there are a number of lower-cost options. One option is to have a *cold site*, which will generally have power, cabling, communications, and some systems installed. Either all necessary equipment will be on a site but not necessarily powered up or an arrangement with a vendor will be in place for the rapid shipment of standard equipment, such as PCs. The timeframe for activating a cold site can range from hours to days of elapsed time, depending on factors such as the time it takes for required staff to travel to the backup site, the time to deliver additional equipment and software, and the time to initiate and synchronize systems. I recall a personal experience where a Florida company declared a disaster due to a major storm soon after the Sept. 11 terrorist attacks, but was delayed several days in effecting its backup plan because the backup facility was in Chicago and there were no planes flying. The need to use land-based transportation to move people and resources (such as physical data media) significantly extended the recovery time.

A *white-wall facility*[3] is an extreme form of a cold site. It is essentially an empty space that the organization has previously obtained for its use in a disaster. It generally offers little more than the bare walls, with a minimum of power, heating, and cooling utilities, and perhaps some minimal telecommunications, installed. Such a site must be built out on demand. This generally means that the organization must order, receive, and install necessary

equipment, software, and other resources at the time of an incident.

Organizations are well advised to at least have previously negotiated arrangements and agreements with vendors and service providers for the rapid delivery and installation of required resources. Experience has shown that vendors and service providers are usually very responsive in the face of an emergency, giving the affected organization priority service in order to minimize downtime. Of course, it is also in the self-interest of vendors to do what they can to enable the customer organization to survive a disaster. While setting up a white-wall facility can take weeks, it is a big step ahead of having no facility at all[4].

If no backup system or facility has been provided, the choice (depending on the nature of the failure) is to fix and recover the primary system or facility, or to build a new system from available resources. Here, a good practice is to take snapshots of the system and data at predetermined intervals and go through a restart process. Depending on the level of recovery and reconstruction necessary, this type of recovery can be an extremely expensive endeavor in and of itself and the consequential financial losses due to lost productivity, damaged reputation, fleeing customers, and the like can be enormous.

Another option that should be mentioned is a one-way or two-way agreement with another party. One can subscribe to commercial disaster recovery services and pay a monthly fee to have the right to use their facilities and additional fees if a disaster is declared. Such facilities can be very effective as they are often staffed and run continuously. One issue to be aware of is that when disaster recovery facilities are shared among a number of customers within a given region, the amount of backup services might not be available to the degree expected if a disaster were to be regional in scope. The level of services provided can range from hot backup to white-wall, with corresponding charges.

Another option is to institute a reciprocal arrangement with another nearby company, often in the same or similar business. However, they can be difficult to implement since there is no guarantee that the reciprocating partner will be able to provide the facilities when needed. I recall a situation in which my company needed to invoke such an arrangement at 6 a.m. one day. However, my staff could not get into the other company's facility since the persons familiar with the arrangement were in transit and the building guards had not been informed about the arrangement and would not let the operators into the building. Ironically, when the other party needed to invoke the arrangement a few months after the unsuccessful attempt by my company to use their facilities, they needed to invoke the mutual backup arrangement. In contrast, however, my company was able to provide the resources on demand. As seen by this example, such arrangements may be very low cost, but they are also unreliable and difficult to enforce.

The use of *cloud computing* is similar to the disaster recovery services model except that cloud computing services might not require a monthly fee if the arrangement is only to pay for cloud services actually used.

Figure 1 (see previous page) illustrates the various backup relationships previously discussed. It also shows that other parties—such as customers, service providers, and business partners—need to be included. In particular, it is highly advisable to test connectivity and operability between backup facilities and third parties. More recently, there have been calls for backup-to-backup testing between organizations and third parties.

Table 2 shows, for various failure scenarios and types of backup, the relative costs of setting up and operating the backup capabilities, how much (on a relative basis) it might cost to recover if an incident occurs, as well as what the combined costs might be.

Please note that these are very rough ordinal assessments that do not allow for essential characteristics of systems (such as their criticality to the business and their technical complexity) nor do they account for the frequency and magnitude of events. The assessments are provided as guidance as to what one might find in a typical business or government situation.

The Economics of Resiliency

It is clear that there is a need to balance

Table 2: *Costs of Backup, Response, and Recovery by Scope of Event and Type of Backup*

Scope of Failure or Event	Type of Backup Put in Place (if any)	Cost of Setting Up Backup	Ongoing Costs of Maintenance and Support	Typical Time to Respond and Recover	Cost of Incident Response and Recovery	Probable Frequency of Event per Period*	Incident Cost per Period (Magnitude x Frequency)
Component	Hot fail-over	High	High	Seconds/Minutes	Low	Moderate	Low
Component	Warm fail-over	Moderate	Moderate	Hours	Moderate	Moderate	Low
Component	No fail-over	Low	Low	Days	Very high	Moderate	Extremely high
System	Hot backup	High	High	Seconds/Minutes	Low	Moderate/High	Moderate
System	Warm backup	Moderate	Moderate	Hours	Moderate	Moderate/High	High
System	No backup	Low	Low	Days	Very high	Moderate/High	Extremely high
Site (Facility)	Hot site	Very high	Very high	Seconds/Minutes	Low	Moderate	Moderate
Site (Facility)	Warm site	High	High	Hours	Moderate	Moderate	High
Site (Facility)	Cold site	Moderate	Moderate	Days	Very high	Moderate	Very high
Site (Facility)	White wall	Low	Low	Weeks/months	Extremely high	Moderate	Extremely high
Regional	Hot site	Extremely high	Extremely high	Seconds/Minutes	Low	Low/Moderate	Low
Regional	Warm site	High	High	Hours	Moderate	Low/Moderate	Moderate
Regional	Cold site	Moderate	Moderate	Days	Very high	Low/Moderate	Very high
Regional	White wall	Low	Low	Weeks/months	Extremely high	Low/Moderate	Extremely high
National/Global	Hot site	Extremely high	Extremely high	Seconds/Minutes	Low	Low	Low
National/Global	Warm site	High	High	Hours	Moderate	Low	Moderate
National/Global	Cold site	Moderate	Moderate	Days	Very high	Low	Very high
National/Global	White wall	Low	Low	Weeks/months	Extremely high	Low	Extremely high

* Note that the frequency of events, other than those outside the control of the organization, can be influenced by those responsible for designing and setting up systems, facilities, and infrastructures. The levels shown are for frequency are based on experience, but may not be applicable to a particular case.

the cost and effectiveness of backup and recovery capabilities against the expectations of damaging and destructive events. The mentioned scenarios and costs relate to recovery from successful attacks or damaging events. These costs may be reduced if the expectation of failure or compromise is lowered through preventative measures, deterrence, or avoidance.

There is a trade-off between protective measures and investments in survivability. The determination of the optimum level of backup is based on the expectations of damaging events, the impact of these events, and the ability to recover quickly and return to acceptable operation.

It should also be noted that the different levels of backup are not independent. Hence, if one has a hot backup system installed in a within-region backup facility, it may not be cost-effective to have an on-site backup system. Conversely, if one installs a highly resilient primary system with various degrees of internal redundancy, it is less likely that a backup system will be required and thus a warm off-site backup system may be adequate.

This suggests that a number of combinations need to be evaluated, depending on the resiliency of the primary systems, the criticality of the application, and the options as to backup systems and facilities. Thus, it is up to the analyst to determine which options and which combinations make the most sense for a particular environment and then to cost out the preferred options.

Summary

The topic of software resiliency is not addressed at a level appropriate to its impact on organizations. This article has examined the factors that affect software resiliency and the contexts in which applications might run, particularly in regard to the wide choice of backup options.

Further work is needed—particularly with respect to running some numbers for a variety of cases and reviewing the results. It may be that the realistic options are much more limited than expected. Also, the growing availability of cloud computing may completely change the results of disaster backup analyses in favor of backup in the cloud. At the same time, cloud computing introduces its own issues in regard to resiliency and recovery.◆

Software Defense Application

As software and its implementation become increasingly complex and dependent on diverse infrastructures, it has become essential for those designing and developing computer applications to be aware of, and allow for, the evermore challenging environments into which software is installed. This article provides those in the DoD responsible for software design and development, infrastructure support, data center operations, disaster recovery planning, and incident response with the necessary guidance, concepts, techniques, and methodologies to provide the overall level of resiliency required for specific systems. As cyber attacks grow in their capabilities and effectiveness, those developing and deploying DoD systems must enhance their understanding of the impact of failures from attacks, inadvertent actions, and natural events on the availability of computer systems and networks. They need to take steps so that systems can rapidly and accurately be recovered from failures and outages, whatever their cause.

References

1. The Register and Freeform Dynamics, Ltd. "Risk and Resilience: The Application Availability Gamble." *The Register*. July 2008 <http://white papers.theregister.co.uk/paper/view/485/c-documents-and-settings-regmar06-002-desktop-pdf-risk-resilience.pdf>.
2. Marcus, Evan, and Hal Stern. *Blueprints for High Availability: Designing Resilient Distributed Systems*. New York: John Wiley & Sons, 2000.
3. Brandon, D.H., and Sidney Segelstein. *Data Processing Contracts: Structure, Contents, and Negotiation*. New York: John Wiley & Sons, 1976.
4. Architecture-Driven Modernization Object Management Group. "A White Paper of Software Assurance." 28 Nov. 2005 <http://adm.omg.org/SoftwareAssurance.pdf>.
5. Fortify's Security Research Group, and Suto, Larry. "Open Source Security Study." July 2008 <www.fortify.com/landing/oss/oss_report.jsp>.
6. Gershenfeld, Neil. "Everything, the Universe, and Life." *IBM Systems Journal* 39.3/4 (2000): 932-934.
7. Anderson, Ross J. *Security Engineering*. 2nd ed. New York: John Wiley & Sons, 2008: 192.

Notes

1. Such a technology is described in the article, "Soft Hardware for a Flexible Chip," which is available at <http://cordis.europa.eu/ictresults/index.cfm?section=news&tpl=article&id=90572>.
2. Information regarding the Software Assurance Initiative and other projects of the Financial Services Technology Consortium is available at <www.fstc.org/projects/index.php?new=1>.
3. A white-wall facility is a term that I heard while developing disaster recovery plans for a major financial institution. The term does not appear to be in the literature and a search for its particular use in the context of disaster recovery did not produce any results.
4. It is necessary to begin looking for a building and then negotiating a lease or purchase, which can take weeks or months.

About the Author

C. Warren Axelrod, Ph.D., is the research director for financial services for the U.S. Cyber Consequences Unit and is executive adviser to the Financial Services Technology Consortium. Previously, he was the chief privacy officer and business information security officer for the U.S. Trust Division of Bank of America. He is also a founder of the Financial Services Information Sharing and Analysis Center. He received the Information Security Executive Luminary Leadership Award in 2007 and *Computerworld*'s Premier 100 Leadership and Best in Class awards in 2003. He has published three books, two of which are on computer management, and numerous articles on a variety of information technology and information security topics. Axelrod holds a doctorate in managerial economics from Cornell University, as well as a master's degree in economics and statistics, and a bachelor's degree in electrical engineering from Glasgow University. He is a Certified Information Systems Security Professional and Certified Information Security Manager.

U.S. Cyber Consequences Unit
P.O. Box 234030
Great Neck, NY 11023
Phone: (917) 670-1720
E-mail: warren.axelrod@usccu.us