Survey of the Survivability of IT Systems

Pentti Tarvainen
VTT Technical Research Centre of Finland
P.O. Box 1100, FIN-90571 Oulu, Finland
email: pentti.tarvainen@vtt.fi

Abstract— **Failure of IT systems often causes a major loss of service. Thus their dependability has become an important issue. Recent facets of the dependability of IT systems, such as reliability, availability, safety, security, confidentiality, integrity, and maintainability do not address the needs of IT systems because they do not include the notion of a degraded service as an explicit requirement. The concept termed survivability is a precise notion of the forms of services that are acceptable in a system, the circumstances under which each form is most useful, and the fraction of time that is acceptable in degraded services. In this paper survivability is discussed as a necessary new facet of dependability. The contribution of this paper is to give system architects the latest knowledge on survivability in order to help them develop survivable IT systems. Definitions of dependability and survivability are presented and discussed. In addition, the key properties and survivability requirements, and the role of fault tolerance in survivable systems are discussed. Furthermore, survivability implementation techniques and examples of survivability architectures are introduced and discussed. Finally, software architecture design and analyzing methods and frameworks relevant to survivability are discussed.**

Index Terms—**Dependability, reliability, security, survivability**

I. Introduction

MODERN society faces a substantial, and generally acknowledged, risk of IT systems failure or compromise with potentially serious consequences. Sectors such as energy, financial services, telecommunications, healthcare, and defense are potential application areas of such systems [4], [5], [6], [8], [12]. Despite the best efforts of security professionals, no amount of system hardening can assure that a system that is connected to an unbounded network, such as the Internet, will not be vulnerable to attack. From point of view of a system architect's practical work, it is important to know what survivability really means and how it can be applied to IT systems. Survivability, and the survivability requirements, of an IT system must be taken into account at the system architecture design phase.

Survivability in IT systems is a relatively new research area. The precise definition of survivability is still being debated, with a number of definitions proposed. Most commonly, survivability is defined as "the ability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures or accident" [8], [9], [10], [11], [18], [19]. The term system is used in the broadest possible sense, including networks and large-scale systems of systems. The term mission refers to a set of abstract requirements or goals.

Survivability is a necessary new branch of dependability [1], [2], [3], [5], [8], [9], [10], [11], [18]. It addresses explicit requirements for restricted modes of operation that preserve critical essential services in adverse operational environments. A survivable system is one that satisfies its survivability specification of essential services and adverse environments. Essential services are defined as the functions of the system that must be maintained when the environment is hostile or failures or accidents are detected that threaten the system. The discipline of survivability can help ensure that IT systems can deliver essential services and maintain such essential properties as performance, security, reliability, availability and modifiability despite the presence of intrusions. Unlike the traditional security measures that require central control or administration, survivability is intended to address unbounded network environments.

Survivability requirements can vary substantially, depending on the scope of the system, the criticality and the consequences of failure and interruption of service. The definition and analysis of survivability requirements is a critical first step in achieving system survivability [8]. Survivability must address not only software function requirements but also requirements for software usage, development, operation and evolution.

Survivability architectures offer an approach to tolerating faults in which the continued service element of fault tolerance differs from normal service. By introducing this type of fault tolerance, progress can be made towards meeting the survivability goals of IT systems.

The rest of the paper is organized as follows: definitions of dependability and survivability are presented and discussed in Section II,. In Section III key properties and survivability requirements, and the role of fault tolerance in survivable systems are discussed. Techniques for implementing survivability are discussed in Section IV and examples of survivability architectures are introduced. Software architecture design and analyzing methods and frameworks that take survivability into account are introduced in Section V, and, Section VI contains the concluding remarks.

Manuscript received October 4, 2004. This research work was conducted in the Moose project under the ITEA cluster projects of the EUREKA network, and financially supported by Tekes (the National Technology Agency of Finland).

P. Tarvainen is with VTT Electronics, VTT, the Technical Research Centre of Finland, P.O. Box 1100, FIN-90571 Oulu, Finland, pentti.tarvainen@vtt.fi.

II. Definitions Related to Survivability

A definition of dependability and different definitions of survivability based on recent research work are discussed in this section; unofficial and common definitions of survivability are also introduced and discussed.

A. Definition of Dependability

Dependability is the system property that integrates such attributes as reliability, availability, safety, confidentiality, integrity, maintainability and survivability [32]. The dependability of a computing system is its ability to deliver a service that can justifiably be trusted. The service delivered by a system is the behavior of a system as it is perceived by its user(s). A user is another system, physical or human, that interacts with the former at the service interface. The function of a system is what the system is intended to do, and is described by the functional specification. The correct service is delivered when the service implements the system function.

B. Definitions of Survivability

Software quality is depicted in IEEE 1061 [30] and represents the degree to which the software possesses a desired combination of quality attributes. Another standard, ISO/IEC 9126-1 [31], defines the software quality model for external and internal quality. Based on this model, there are six categories of characteristics - functionality, reliability, usability, efficiency, maintainability and portability - which are further divided into sub-characteristics. Either of these standards does not define survivability.

A preliminary scoping of the general survivability problem was suggested by a 1993 report, "Survivable Computer-Communication Systems: The Problem and Working Group Recommendations" [25], written for the U.S. Army Research Laboratory (ARL). The report outlines a comprehensive multifunctional set of realistic computer-communication survivability requirements and makes related recommendations applicable to U.S. Army and defense systems [22].

The precise definition of survivability is still being debated, with a number of definitions proposed as described in Table I. The definitions in the table are listed in chronological order, based on the year the definition was published. Also the respective references are also denoted in the table. Based on Table I, survivability in respect of IT systems is a relatively new research area and the content of the definition of survivability depends on the domain.

In the context of software engineering, Deutsch [2] has offered the first definition shown in the table. This definition is not sufficient for all needs [18]. If it were applied to IT systems in this form, the user of the system could not be sure which functions had been selected as "essential functions" nor under what circumstances (i.e., after what damage) these functions would be provided.

The Institute for Telecommunications Services, a part of the U.S. Department of Commerce, has created an extensive glossary of telecommunications terms in Federal Standard 1037C [53]. The glossary contains a definition of survivability for telecommunications systems (the second definition in Table I). This definition seeks a framework for defining a service after some form of damage and relates closely to the goal of defining survivability for IT systems [1], [18]. Furthermore, this definition includes the notion of a degraded or different service and requires that it be defined.

Specifically on IT systems survivability, Ellison et al. [3] introduce the third definition of survivability shown in Table I. While this definition is a good beginning, it does not have the precision needed to permit a clear determination of whether a given system should be considered to be survivable [1], [18]. The first problem is that much is implied by the phrases "essential services", "attacks and failures" and "timely manner". If nothing further is defined, it is not possible for the architect of a system to determine whether a specific design is adequate to meet the needs of the user community. More importantly, if a phrase such as "essential service" is not precisely defined, it might be the case for any specific system that the determination of what constitutes an essential service is left to the system's developers rather than being defined carefully by application experts. A second problem with a definition of this form is that it provides no testable criterion for the term being defined. Essential services are defined as the functions of the system that must be maintained when the environment is hostile, or failures or accidents that threaten the system are detected.

Most commonly [8], [9], [10], [11], [18], [19], survivability

TABLE I
Definitions of Survivability

	Definition	Year	Domain	Ref.
1.	Survivability is the degree to which essential functions are still available even though some part of the system is down.	1988	IT systems in general	[2]
2.	Survivability is a property of a system, subsystem, equipment, process or procedure that provides a defined degree of assurance that the named entity will continue to function during and after a natural or man-made disturbance. Note: Survivability must be qualified by specifying the range of conditions over which the entity will survive the minimum acceptable level or post-disturbance functionality and the maximum acceptable outage duration.	1996	Telecommunication Systems	[54]
3.	Survivability is the ability of a network computing system to provide essential services in the presence of attacks and failures and recover full services in a timely manner.	1997	Network Computing Systems	[3]
4.	Survivability is the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures or accidents.	1999	Critical and Defense Systems	[8],[9], [10],[11], [18],[19]
5.	Survivability is the ability [of a system] to continue to provide service, possibly degraded or different, in a given operating environment when various events cause major damage to the system or its operating environment.	2000	Critical and Defense Systems	[1], [20]

is defined as the fourth definition in Table I. The term system, consisting of software and hardware, is used in the broadest possible sense, including networks and large-scale systems of systems. The term mission refers to a set of very high-level (i.e., abstract) requirements or goals. Missions are not limited to military settings since any successful organization or project must have a vision of its objectives, whether expressed implicitly or as an official mission statement. The terms attack, failure, and accident are meant to include all potentially damaging events, but these terms do not partition these events into mutually exclusive or even distinguishable sets. Attacks are potentially damaging events orchestrated by an intelligent adversary. Attacks include intrusions, probes and denials of service. Failures are included with accidents as part of survivability. Failures are potentially damaging events caused by deficiencies in the system or in an external element on which the system depends. Failures may be due to software design errors, hardware degradation, human errors or corrupted data. Accidents describe a broad range of randomly occurring and potentially damaging events, such as natural disasters. It is important to recognize that it is the mission fulfillment that must survive, not any particular subsystem or system component.

The fifth definition of survivability [1], [20] in Table I suggests a number of key points regarding the notion of survivability:

Survivability is a system property, relating the level of service provided to the level of damage present in the system and operating environment [7],

A system must be capable of providing different levels of service. In a system free of damage, the level of service should equate with full functionality. Different levels of service will correspond to varying subsets of functionality, where some functions that a system performs are obviously more critical than others [1], [18], and

The events that cause major damage can range from failures to attacks to accidents. It is often difficult to immediately determine the cause of the damage, e.g. whether the damage is the result of an intentional security attack or a random failure [3]. More important is the effect of the event in terms of damage to the system and operating environment — the amount of damage is central to the level of service that a survivable system can and should provide [7].

III. Requirements and Key Properties of Survivable Systems

In this section the key properties and requirements, and the role of fault tolerance in survivable systems are defined and discussed. The key properties of survivable systems are as follows [8]:

Firstly, central to the delivery of essential services is the ability of a system to maintain essential properties, i.e. specified levels of integrity, confidentiality, performance and other quality attributes. Thus it is important to define the minimum levels of quality attributes that must be associated with essential services.

Secondly, quality attributes are so important that definitions of survivability are often expressed in terms of maintaining a balance among multiple quality attributes, such as performance, security, reliability, availability, modifiability and affordability. Quality attributes represent broad categories of related requirements, so a quality attribute may contain other quality attributes. For example, the security attribute traditionally includes the three attributes of confidentiality, integrity and availability.

Thirdly, the ability to deliver essential services and maintain the associated essential properties must be sustained, even if a significant portion of the system is incapacitated. This ability should not be dependent upon the survival of a specific information resource, computation or communication link.

Fourthly, key to the concept of survivability is identifying the essential services, and the essential properties that support them, within an operational system. There are typically many services that can be temporarily suspended when a system is dealing with an attack or other extraordinary environmental condition. Such a suspension can help isolate areas affected by an intrusion and free system resources to deal with the intrusion's effects. The overall function of a system should adapt to the situation to preserve essential services. If an essential service is lost, it can be replaced by another service that supports mission fulfillment in a different but equivalent way. However, the identification and protection of essential services is an important part of a practical approach to building and analyzing survivable systems.

The survivability requirements of survivable systems can vary substantially, depending on the scope of the system, and the criticality and consequences of failure and interruption of service [8]. The definition and analysis of survivability requirements is a critical first step in achieving system survivability. Survivability must address not only the requirements for software functionality but also the requirements for software usage, development, operation and evolution. Five types of requirements definitions are relevant to survivable systems [8]: (1) System/Survivability Requirements, (2) Usage/Intrusion Requirements, (3) Development Requirements, (4) Operations Requirements and (5) Evolution Requirements.

Fault tolerance enables systems to continue to provide service in spite of the presence of faults. Fault tolerance consists of four phases: (1) error detection, (2) damage assessment, (3) state restoration and (4) continued service. Survivability is a dependability property; it is not synonymous with fault tolerance [1]. Fault tolerance is a mechanism that can be used to achieve certain dependability properties. In terms of dependability, it makes sense to refer to a system as reliable, available, secure, safe and survivable, or some combination, using the appropriate official definition(s). Describing a system as fault tolerant is really a statement about the system's design, not its dependability. While fault tolerance is a mechanism by which some facets of dependability might be achieved, it is not the only mechanism. Other techniques, such as fault avoidance, can also be used. In similar ways, fault elimination and fault forecasting can be used as mechanisms to improve a system's dependability.

IV. Implementation Techniques for Survivability

This section discusses implementation techniques related to the survivability of IT systems. In addition, examples of survivability architectures of IT systems are introduced and discussed

A. Survivability and Security

It is important to recognize the relationship between survivability and security. An application may employ security mechanisms, such as passwords and encryption, and may still be very fragile [17]. For instance, it may fail when a server or a network link dies. On the other hand, a survivable application must be able to survive some malicious attacks. Therefore, survivability must involve security. There are two aspects of survivability [17]: (1) survival by protection and (2) survival by adaptation.

Security mechanisms like access control and encryption attempt to ensure survivability by protecting applications from harmful, accidental or malicious changes in the environment [17]. The application could also survive by adapting itself to the changing conditions. These two aspects may not be mutually exclusive; as part of survival by adaptation, an application may utilize security mechanisms. For example, it may start using access control or increase the key length when it perceives the threat of an intrusion. Most current applications fail rather than adapt when QoS assumptions turn out to be too optimistic. The problem is made worse by the fact that survivability mechanisms are complicated, have little to do with an application's functionality, and developers only have limited tool support for incorporating them.

Furthermore, based on [17], the general notion of survival by adaptation results from years of experience in designing, implementing and deploying wide-area distributed systems, and is based on the ARMD (Adaptive, Redundant, Monitored, and Diversified) principles. These principles are not independent, and they need to be organized in a meaningful way to lead to survivability. For instance, being adaptive generally means that the adaptation is driven by some kind of monitoring. However, monitoring could be used without any kind of adaptation at all. Similarly, redundancy and diversity could very well be used without any adaptation, but in the context of adaptation they often define or broaden the scope of adaptation. Not all adaptive behaviors lead to survivability. For instance, shutting an application down on an exception indicating a change in the environment does not add anything to the survivability of the application. In fact, such an adaptation facilitates a whole class of denial of service attacks, whereas survivability is about continuing to provide useful service despite environmental changes.

Survival by adaptation typically involves monitoring and changing the Quality of Service (QoS) available to applications [17]. An application's design always makes some assumptions about the QoS provided by the environment for bandwidth, reliability, security services, etc. When these assumptions are violated, the application should try to adapt rather than fail. Most current applications, however, fail rather than adapt when QoS assumptions turn out to be too optimistic. The problem is made worse by the fact that survivability mechanisms are complicated, have little to do with an application's functionality, and developers only have limited tool support for incorporating them.

One important technique for improving system dependability and survivability is to provide mechanisms for a system to adapt at run time in order to accommodate varying resources, system errors and changing requirements [52]. For such self-repairing systems, one of the difficult problems is determining when a change is needed, and knowing what kind of adaptation is required. Based on [52], the challenge is to engineer things so that the system adapts appropriately at run time. There are two problems with this [52]. First, information must be got out of the running system. This can be done by employing low-level monitoring mechanisms that cover various aspects of the executing system. The second problem is to translate architectural repairs into actual system changes. This can be solved by writing table-driven translators that can interpret architectural repair operators in terms of the lower level system modifications.

B. Survivability Architectures

Survivability architecture is a system architecture that is designed specifically to deal with certain non-local faults [50]. A significant difficulty arises when the various concepts involved in survivability architectures have to be evaluated because experimentation with real systems is precluded. One approach to dealing with this problem is to use operational models that can be built and studied in the laboratory using a developed experimentation system [50].

1) Survivability Architecture: Block, Evade, React (SABER). Paper [33] proposes a survivability architecture called SABER. SABER blocks, evades and reacts to a variety of attacks by using several security and survivability mechanisms in an automated and coordinated fashion. SABER integrates several different technologies in an attempt to provide a unified framework for responding to the wide range of attacks malicious insiders and outsiders can launch. Most commercial responses to the diverse array of vulnerabilities have been to apply several discrete solutions [33]: (1) utilization of network-based firewalls [34], [35], (2) deployment of network-based and host-based intrusion detection systems [36], [42] and (3) manual installation and deployment of patches [37], [38].

At present, SABER is in the prototyping stages, with several interesting open research topics. It currently makes use of the following reaction and protection mechanisms [33]: (1) a network Denial-of-Service (DoS) resistant and Secure Overlay Services (SOS) architecture [39], (2) Manuscript intrusion and anomaly detection tools, [43], [44], placed within service contexts to detect malicious activity as well as stealthy "scans and probes", (3) a process migration system [40] that can be used to move a service to a new location that is not (currently) targeted by an attacker, (4) an automated software-patching system [41] that dynamically fixes certain classes of software-based attacks, such as buffer overflows, and (5) a high-level coordination and control infrastructure to correlate and coordinate the information and control flow.

2) Intrusion Tolerant Distributed Object System (ITDOS): Intrusion Tolerant Distributed Object System (ITDOS)

TABLE II
FEATURES OF SURVIVABILITY ARCHITECTURES

Feature	SABER	ITDOS	C4I
Security Mechanism	Integrates several security and survivability mechanisms	Symmetric Encryption Session Keys	"Plan-Based Survivability", Mobile IP and Ad Hoc network protocols for military use
Reaction/Protection Mechanism	Process Migration System, Network Denial-of-Service (DoS), Secure Overlay Services (SOS), An automated software-patching system	Distributed Object Middleware, CORBA	"Plan-Based Survivability"
Intrusion Detection	Network- and host-based intrusion detection, Anomaly-, registry- and file-based detection, Surveillance detection	Fault Tolerant Multicast Protocol + CORBA	"Plan-Based Survivability"
Domain	Network Systems	Heterogeneous Distributed Middleware Systems	Mobile Military Tactical Systems
Maturity	Prototype	Prototype	Prototype
Publishing Year	2003	2002	1999
Reference	[33]	[47]	[51]

architecture [47] discusses some of the challenging technical issues related to intrusion tolerance in heterogeneous middleware systems. The intrusion tolerant systems provide integrity and availability services in the face of successful attacks from an adversary.

Middleware is one area in which a system can provide intrusion tolerance [47]. Middleware is a very useful category of software that removes much of the tedium of distributed systems programming and shields programmers from having to deal with the numerous kinds of heterogeneity inherent in a distributed system [48]. Distributed object middleware is considered the most general kind of middleware, and CORBA [49] is a widely adopted standard for distributed object middleware. Middleware provides an ideal platform for intrusion tolerance extensions because it allows a variety of applications to be built that can transparently take advantage of the intrusion tolerance properties of the middleware, eliminating the need for custom solutions for each application [47].

3) Middleware Architecture for Secure and Survivable Mobile C4I Systems: An overview of a middleware-based mobile C4I (Command, Control, Communications, Computers, & Intelligence) architecture is discussed in [51]. The architecture is an outgrowth of work on a mobile distributed operating system that attempts to deal with various shortcomings in the Mobile Internet Protocol (Mobile IP) for military use. The architecture provides a foundation for balanced treatment of the complex, and frequently conflicting, dependability requirements (i.e. security, survivability, etc.) of military tactical systems.

The survivability architecture presented in [51] is controversial in that the Session Layer performs the primary communications function of a mobile "hand-off", instead of relying exclusively on the Network Layer to perform this function. This approach is defended on the basis that where tactical survivability is paramount, a mobile "hand-off" must be carefully controlled by the Session Layer, even if not specifically performed by that layer. The popular private sector approaches (e.g. Mobile IP) attempt to provide a "virtually stationary" environment by use of address mappings, which, except for performance impact, completely hide motion effects in the Network Layer. Such mobile networking approaches are unsuitable for military mobile C4I use, unless they are modified or designed to carefully coordinate their resource-management facilities with the survivability mechanisms of the Session Layer [51]. The mobile C4I architecture is a part of an evolving paradigm for C4I survivability called the Plan-Based Survivability, which seems to be able to solve many open problems with current survivability technology, and which has already been partly demonstrated by a working prototype. In effect, Plan-Based Survivability is a "superstructure" for unifying a diverse hierarchy of C4I defenses, both physical and informational.

C. Summary

Table II summarizes the features of the survivable architectures mentioned above. As a conclusion, the maturity of the architectures is insufficient for practical utilization in a system architect's work, but they will help to understand and solve the problem of survivable systems. The technical approaches of the architectures heavily depend on the system domain.

V. DESIGN OF SURVIVABILITY

In this section the available design and analysis methods and models, as well as frameworks relevant for the survivability of IT systems, are introduced and discussed.

A. Architecture Tradeoff Analysis Method (ATAM)

Paper [8] outlines an approach that addresses how to evaluate the ability of a system to deliver essential functions in an environment that includes intrusion scenarios. The general approach to survivability and security is consistent with the ATAM [14]. ATAM is a method for evaluating architecture-level designs that consider such multiple quality attributes as modifiability, performance, reliability and security to gain insight as to whether the fully fleshed out incarnation of the architecture will meet its requirements [14]. The method identifies tradeoff points between these attributes, facilitates communication between stakeholders (such as user, developer, customer, maintainer) from the perspective of each attribute, clarifies and refines the requirements, and provides a framework for an ongoing, concurrent process of system design and analysis.

B. The Survivable Network Analysis Method (SNA)

A four-step SNA method [23] has been developed for analyzing survivability in distributed systems. SNA is a practical engineering process that enables systematic assessment of the survivability properties of proposed and existing systems, and modifications to existing systems. The analysis can be carried out at the lifecycle, requirements or architecture level. Based on [23], although the SNA method is developed for use with large-scale distributed-network systems, it is equally applicable to other architectures, including host-based and real-time systems. SNA's scenario-based approach is a generalization of the operation sequence and usage scenario methods.

C. The Willow Survivability Architecture

The Willow Architecture [13] is designed to enhance the survivability of IT systems and is a comprehensive approach to survivability in distributed applications. Based on [13], survivability is achieved in a deployed system using a unique combination of (1) fault avoidance by disabling vulnerable network elements intentionally when a threat is detected or predicted, (2) fault elimination by replacing system software elements when faults are discovered, and (3) fault tolerance by reconfiguring the system if non-maskable damage occurs.

D. Open Implementation Toolkit for Building Survivable Applications (QuO)

In [17] Pal, Loyall, Schertz and Zinky consider two aspects of survivability - namely, survival by adaptation and survival by protection. They show how the Quality Objects (QuO) distributed adaptive middleware framework enables the introduction of these aspects of survivability in a flexible and systematic manner. Furthermore, they also describe a toolkit for developing adaptive applications and demonstrate how more survivable applications can be built using the toolkit.

E. A Survivability Framework for Wireless Access Networks

Based on [21], a Survivability Framework for Wireless Access Networks consists of four layers, with survivability strategies possible in each layer. The four layers are termed access, access link level, transport and intelligent. The logical layers are independent of the physical implementation of the network. Each of the four layers is characterized by network functions, network components and communication links. This framework includes metrics for quantifying network survivability, possible survivability strategies, and restoration techniques for each layer.

F. An Architectural Framework and Algorithms for Engineering Dynamic Real-Time Distributed Systems

In [24] Ravindran presents a resource management architecture for engineering dynamic real-time, military, computer-based, Command and Control (C2) systems using commercial off-the-self technologies. In the proposed architecture a real-time system application is developed in a general-purpose programming language and system description language is used to specify the architectural-level description of the system. An abstract model that is constructed from the language specification is dynamically augmented by the System Description Language Runtime System to produce a dynamic Intermediate Representation (IR). The dynamic IR characterizes the state of the system and is used by a recourse management element to deliver the desired application QoS. The middleware techniques achieve the timeliness and survivability requirements through runtime monitoring and failure detection, diagnosis and dynamic recourse allocation.

G. Easel Modeling and Simulation Language

Easel [15] is a modeling and simulation programming language primarily intended for the research, analysis and depiction of unbounded systems, survivable architectures and emergent algorithms in applications, including Internet security, ad hoc communication networks, electric power and cooperation among autonomous vehicles. Easel is a notation

TABLE III
FEATURES OF THE SURVIVABILITY METHODS, MODELS, AND FRAMEWORKS

Method	Type of Method	Survivability Approach	Domain	Maturity Level	Ref.
ATAM	Design and Analysis Method	Intrusion Scenarios, Quality attributes	Not limited	High, widely used	[8], [14]
SNA	Design and Analysis Method	Scenarios	Large-Scale Distributed-Network Systems	High, based on ATAM	[23]
Willow	Modeling tool	Fault Avoidance, Fault Elimination, Fault Tolerance	Critical Distributed Applications	Medium	[13]
QuO	Modeling tool	Quality Objects, Adaptation, Protection	Middleware Applications	Medium	[17]
Survivability Framework for WANs	Modeling tool	Metrics, Restoration Techniques	Wireless Access Networks	Low	[21]
Architectural Framework	Modeling tool	System Description Language, Runtime Monitoring, Failure Detection, Dynamic Recourse Allocation	Military C2 Systems	High	[24]
Easel	Modeling and Simulation Language	Discrete Event Simulations Models	Unbounded Systems, Ad Hoc Networks	Medium	[15]

for describing abstract models of anything, a translator run-time system for running discrete event simulations from those models. An example of the use of Easel in network survivability analysis is presented in [16].

H. Summary

Table III summarizes the features of the survivability methods, models and frameworks described above. As a conclusion, the survivability approaches vary depending on the system domain. From the point of view of the system architect's practical work, there is a remarkable lack of suitable and mature methods. Only two (ATAM and SNA) design and analysis methods are available. The rest of the methods are modeling or simulation tools.

VI. CONCLUSION

Survivability is a new branch of dependability that addresses the explicit requirements for restricted modes of operation that preserve essential services in adverse operational environments. A survivable system is one that satisfies its survivability specification of essential services and adverse environments. In addition, survivability must address not only the requirements for software functionality but also the requirements for software usage, development, operation and evolution. Furthermore, survivability is a dependability property; it is not synonymous with fault tolerance. Fault tolerance is a mechanism that can be used to achieve certain dependability properties. In terms of dependability, it makes sense to refer to a system as reliable, available, secure, safe, and survivable, or some combination, using the appropriate definition(s). Describing a system as fault tolerant is really a statement about the system's design, not its dependability.

Survivability in respect of IT systems is a relatively new research area and the definition of survivability is still being debated. Two of the five definitions of survivability in Table I mention "essential services", and three of them mention the "degree of degraded or different" service to be provided by the survivable system, so these could represent points of agreement for a unified survivability definition. However, definition three mentions that full services will be recovered, whereas the other definitions only mention mission fulfillment, not full service recovery.

Security attacks are a major concern for IT systems. In some discussions survivability is viewed as synonymous with secure operation. A survivable application must be able to survive some malicious attacks, so survivability must involve security. There are at least two aspects of survivability: survival by protection and survival by adaptation. Security mechanisms like access control and encryption attempt to ensure survivability by protecting applications from harmful, accidental or malicious changes in the environment. Survival by adaptation typically involves monitoring and changing the QoS available to applications.

The maturity of the survivable architectures is insufficient for practical utilization in a system architect's work, but they will help to understand and solve the problem of survivable systems. Furthermore, there is a remarkable lack of suitable and mature methods, models and frameworks for practical use.

REFERENCES

[1] J. C. Knight and K. J. Sullivan, "On the Definition of Survivability", University of Virginia, Department of Computer Science, Technical Report CS-TR-33-00, 2000.

[2] M. S. Deutsch and R. R. Willis, "Software Quality Engineering: A Total Technical and Management Approach", Englewood Cliffs, NJ: Prentice-Hall, 1988.

[3] R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. Longstaff and N. R. Mead, "Survivable Network Systems: An Emerging Discipline", Technical Report CMU/SEI-97-TR-013, Software Engineering Institute, Carnegie Mellon University, 1997.

[4] The Information Survivability Workshops of CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University. Available at: http://www.cert.org/research/isw.html, 13.02.2004.

[5] R. C. Linger and A. P. Moore, "Foundations for Survivable System Development: Service Traces, Intrusion Traces and Evaluation Models", Technical Report, CMU/SEI-2001-TR-029, Software Engineering Institute, Carnegie Mellon University, 2001. Available at: http://www.cert.org/archive/pdf/01tr029.pdf, 13.02.2004.

[6] K. Sullivan, J. Knight, X. Du and S. Geist, "Information Survivability Control Systems", Proceedings of the 21st International Conference on Software Engineering, Los Angeles, California, pp. 184-192, 1999.

[7] M. C. Elder, "Fault Tolerance in Critical Information Systems", Dissertation, Faculty of the School of Engineering and Applied Science, University of Virginia, 2001.

[8] R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. A. Longstaff and N. R. Mead, "An Approach to Survivable Systems", Technical Report, CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, 1999.

[9] I. Byon, "Survivability of the U.S. Electric Power Industry", Master's Thesis, Carnegie Mellon University, Information Networking Institute, 2000.

[10] J. Caldera, "Survivability Requirements for the U.S. Health Care Industry", Master's Thesis, Carnegie Mellon University, Information Networking Institute, 2000.

[11] R. J. Ellison, D. A. Fisher, R., C. Linger, H. F. Lipson, T. A. Longstaff and N. R. Mead "Survivability: Protecting Your Critical Systems", CERT Coordination Center Software Engineering Institute, IEEE Internet Computing, pp. 55-63, 1999.

[12] R. J. Ellison, L.C. Linger, T. Longstaff, and N. R. Mead "Survivable Network System Analysis: A Case Study", IEEE Software, Vol. 16, Issue: 4, pp. 70-77, 1999.

[13] J. Knight, D. Heimbigner, A. L. Wolf, A. Carzaniga, J. Hill, P. Devanbu and M. Gertz, "The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications", Technical Report CU-CS-926-01, Department of Computer Science, University of Colorado, Boulder, Colorado, 2001.

[14] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson and S., J. Carriere, "The Architecture Tradeoff Analysis Method", Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, IEEE Computer Society, 1998.

[15] WWW-pages of CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University. Available at: http://www.cert.org/easel/, 13.02.2004.

[16] A. M. Christie, "Network Survivability Analysis Using Easel", Technical Report, CMU/SEI-2002-TR-039, ESC-TR-2002-039, Software Engineering Institute, Carnegie Mellon University, 2002.

[17] P. P. Pal, J. P. Loyall, R. E. Schantz, J. A. Zinky and F. Webber, "Open implementation toolkit for building survivable applications", DARPA Information Survivability Conference and Exposition, Proceedings, Volume: 2, pp. 197 – 210, 2000.

[18] J. C. Knight, E. A. Strunk and K. J. Sullivan, "Towards a Rigorous Definition of Information System Survivability", DARPA Information Survivability Conference and Exposition, Washington DC, 2003.

[19] M. A. Hiltunen, R. D. Schlichting, C.A. Ugarte and G. T. Wong, "Survivability through Customization and Adaptability: The Cactus

Approach", DARPA Information Survivability Conference and Exposition, pp. 294-307, 2000.

[20] J. C. Knight, K. J. Sullivan, M. C. Elder and C. Wang, "Survivability Architectures: Issues and Approaches", DARPA Information Survivability Conference and Exposition, January 2000.

[21] D. Tipper, T. Dahlberg and H. Shin, "Providing Fault Tolerance in Wireless Access Networks", IEEE Communications Magazine, Vol. 40 Issue: 1, pp. 58-64, 2002.

[22] P. G. Neumann, "Practical Architectures for Survivable Systems and Networks", Technical Report supported by the U.S. Army Research Laboratory (ARL), Computer Science Laboratory, SRI International, 2000.

[23] N. R. Mead, R. J. Ellison, R. C. Linger, T. Longstaff and J. McHugh, "Survivable Network Analysis Method", Technical Report, CMU/SEI-2000-TR-013, ESC-2000-TR-013, Software Engineering Institute, Carnegie Mellon University, 2000.

[24] B. Ravindran, "Engineering Dynamic Real-Time Distributed Systems: Architecture, System Description Language, and Middleware", IEEE Transactions on Software Engineering, Vol. 28 Issue: 1, pp. 30-56, 2002.

[25] A. Barnes, A. Hollway and P. G. Neumann, "Survivable Computer-Communication Systems: The Problem and Working Group Recommendations", Technical report VAL-CE-TR-92-22 (revision 1), U.S. Army Research Laboratory, AMSRL-SL-E, White Sands Missile Range, NM 88002-5513, 1993.

[26] R. Kazman, G. Abowd, L. Bass and P. Clements, "Scenario-Based Analysis of Software Architecture", IEEE Software, Vol. 13, Issue: 6, pp. 47-55, 1996.

[27] M. Klein, T. Ralya, B. Pollak, R. Obenza and H. M. Gonzales "A Practitioner's Handbook for Real-Time Analysis", Boston, MA, Kluwer Academic, 1993.

[28] B. Boehm, "A Spiral Model of Software Development and Enhancement", ACM Software Eng. Notes 11, 4, pp. 22-42, 1986.

[29] M. Matinlassi, E. Niemelä and L. Dobrica, "Quality-driven architecture design and quality analysis method. A revolutionary initiation approach to product line architecture", VTT Electronics, Espoo, VTT Publications: 456, 2002. ISBN 951-38-5967-3, 951-38-5968-1

[30] IEEE Standard 1061- 1998, "Standard for a Software Quality Metrics Methodology", New York: The Institute of Electrical and Electronics Engineers, 1998.

[31] ISO/IEC 2001 - International Organization for Standardization and International Electrotechnical Commission. "Software engineering - Product quality - Part 1: Quality model". ISO/IEC 9126-1:2001(E)

[32] A. Avizienis, J.-C. Laprie and B. Randell, "Fundamental Concepts of Dependability", Technical report CS-TR-739, at University of Newcastle, 2001.

[33] A. D. Keromytis, J. Parekh, P. N. Gross, G. Kaiser, V. Misra, J. Nieh, D. Rubenstein and S. Stolfo, "A Holistic Approach to Service Survivability", Technical Report CUCS-021-03, Department of Computer Science, Columbia University, 2003.

[34] P. Thompson, "Web services – beyond HTTP tunneling". In W3C Workshop on Web Services, 2001.

[35] D. Moore, G. M. Voelker and S. Savage, "Inferring internet Denial-of-Service activity". In Proceedings of the 10th Usenix Security Symposium, pp. 9-22, 2001.

[36] D. Newman, J. Snyder and R. Thayer, "Crying wolf: False alarms hide attacks", Network World, June 2002. Available at: http://www.nwfusion.com/techinsider/2002/0624security1.html. 13.02.2004

[37] "Microsoft Security Tool Kit: Installing and Securing a New Windows 2000 System". Microsoft TechNet. Available at: http://www.microsoft.com/technet/security/tools/tools/w2knew.asp, 13.02.2004

[38] "RedHat 9 Security Advisories". Available at: https://rhn.redhat.com/errata/rh9-errata-security.html, 13.02.2004

[39] A. D. Keromytis, V. Misra and D. Rubenstein, "SOS: Secure Overlay Services". In Proceedings of ACM SIGCOMM, Pp. 61-72, 2002.

[40] S. Osman, D. Subhraveti, G. Su and J. Nieh, "The design and implementation of Zap: A system for migrating computing environments". In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI), pp. 361–376, 2002.

[41] S. Sidiroglou and A. D. Keromytis, "A Network Worm Vaccine Architecture". In Proceedings of the IEEE Workshop on Enterprise Technologies: Infrastructure for Collaborative Enterprises (WET-ICE), Workshop on Enterprise Security, 2003.

[42] M. V. Mahoney and P. K. Chan, "Learning non-stationary models of normal network traffic for detecting novel attacks". In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 376–385, ACM Press, 2002.

[43] D. L. Cook, W. G. Morein, A. D. Keromytis, V. Misra and D. Rubenstein, "WebSOS: Protecting Web Servers from DDoS Attacks". In Proceedings of the IEEE International Conference on Networks (ICON), 2003.

[44] S. Hershkop, R. Ferster, L. H. Bui, K. Wang, and S. J. Stolfo., "Host-based anomaly detection by wrapping file system accesses", Technical report, Columbia University, Department of Computer Science, 2003.

[45] P. Pal, M. Atighetchi, F. Webber, R. Schantz and C. Jones, "Reflections on Evaluating survivability: The APOD Experiments", The 2nd IEEE International Symposium on Network Computing and Applications (NCA-03), 2003.

[46] P. Pal, F. Webber, J. Zinky, R. Shapiro and J. Megquire, "Using QDL to specify QoS aware distributed (QuO) application configuration", IEEE Int'l Symp. Object-Oriented Real-Time Distributed Comp., 2000.

[47] D. Sames, B. Matt, B. Niebuhr, G. Tally, B. Whitmore and D. Bakken, "Developing a Heterogeneous Intrusion Tolerant CORBA System", International Conference on Dependable Systems and Networks (DSN'02), 2002.

[48] D. Bakken, "Middleware", Chapter in Encyclopedia of Distributed Computing, Urban, J., Dasgupta, P., eds., Kluwer Academic Publishers, 2001.

[49] "The Common Object Request Broker: Architecture and specification", OMG, Revision 2.5, 2001.

[50] J. C. Knight, K. J. Sullivan, M. C. Elder and C. Wang, "Survivability architectures: issues and approaches", DARPA Information Survivability Conference and Exposition, Proceedings, Volume: 2, pp. 157 -171, 2000.

[51] R. Browne, J. Valente and S. Hariri, "An advanced middleware architecture for secure and survivable mobile C4I systems", Military Communications Conference Proceedings, MILCOM 1999, IEEE, Volume: 1, pp. 506 -513, 1999.

[52] R. de Lemos, C. Gacek and A. Romanovsky, (Eds.), "Architecting Dependable Systems", Series: Lecture Notes in Computer Science, Vol. 2677, XII, 309 p., 2003, ISBN: 3-540-40727-8.

[53] "Federal Standard 1037C", U.S. Department of Commerce, National Telecommunications and Information Administration, Institute for Telecommunications Services, 1996. Available at: http://www.its.bldrdoc.gov/fs-1037/dir-001/_0065.htm, 09.03.2004.