

Requirements Definition for Survivable Network Systems

R. C. Linger, N. R. Mead, and H. F. Lipson
Software Engineering Institute
Carnegie Mellon University

Abstract

Pervasive societal dependency on large-scale, unbounded network systems, the substantial risks of such dependency, and the growing sophistication of system intruders, have focused increased attention on how to ensure network system survivability. Survivability is the capacity of a system to provide essential services even after successful intrusion and compromise, and to recover full services in a timely manner. Requirements for survivable systems must include definitions of essential and non-essential services, plus definitions of new survivability services for intrusion resistance, recognition, and recovery. Survivable system requirements must also specify both legitimate and intruder usage scenarios, and survivability practices for system development, operation, and evolution. This paper defines a framework for survivable systems requirements definition and discusses requirements for several emerging survivability strategies. Survivability must be designed into network systems, beginning with effective survivability requirements analysis and definition.

1. The Idea of Survivable Network Systems

From their modest beginnings some 20 years ago, network systems have grown to become a critical enabling agent in business, industry, government, and defense. Major economic sectors, including energy, transportation, telecommunications, manufacturing, financial services, health care, and education all depend on a vast array of network systems operating on local, national, and global scales. This pervasive societal dependency on networks magnifies the consequences of failure and amplifies the vital importance of ensuring their survivability. As a result, methods for achieving system survivability are receiving increasing attention [1].

Survivability refers to the capability of a system to complete its mission in a timely manner, even if significant portions are compromised by attack or accident. In particular, survivability refers to the capability of a system to provide essential services in the presence of successful intrusion, and to recover compromised services in a timely manner after intrusion occurs. For example, a survivable financial network would maintain the integrity and availability of essential information, such as account and loan data, and services, such as transaction validation and processing. Integrity would be maintained even if particular nodes or communication links were incapacitated through intrusion or accident, and would recover compromised information and services in a timely manner. While survivability focuses on the preservation of mission capabilities, it includes issues of confidentiality and integrity as well. Because of the value of the CERT® intrusion knowledge base, this work has focused on attack and compromise by intelligent adversaries.

Experience with network systems has shown that no amount of hardening can guarantee invulnerability to attack. Despite best efforts, systems will continue to be breached. Thus, it is vital to expand the current view of information systems security to encompass system behavior that contributes to survivability in spite of intrusions or accidents. Network systems must be robust in the presence of attack and able to survive attacks that cannot be completely repelled. The growing societal dependency on networks and the risks associated with their failure require that survivability be designed into these systems, beginning with effective survivability requirements analysis and definition.

In today's network environment, system security is largely dependent upon the encryption of data and isolation through mechanisms such as firewalls. While

the firewall approach is currently practical in a limited fashion, it will become increasingly inadequate to protect systems from intrusion in the rapidly expanding world of unbounded network computing. Current systems are characterized by customer owned and controlled computing resources communicating over unbounded networks. In future systems, most computing resources will be resident within unbounded network infrastructures, and will be controlled by a multitude of computing and communications service providers. These environments will be so unbounded as to render ineffective current security approaches, such as firewalls, that are based solely on isolation. In such environments, firewalls will be ineffective in detecting attacks, recovering from attacks, or helping systems survive intrusions and complete their missions in spite of malicious activity. Future unbounded systems will also embody dynamic architectures, capable of automated, real-time reconfiguration and adaptation in response to changing requirements and environments.

In summary, survivable network systems embody two essential characteristics. First, they preserve essential services under intrusion and recover full services in a timely manner. Second, they ensure survivability in environments characterized by unbounded networks and dynamic architectures. It is often the case that insufficient emphasis is placed on these survivability issues. As a result, the processes and techniques for addressing survivability are generally inadequate to deal with the threat. Concepts of system survivability provide a framework for integrating established disciplines of system reliability [2], safety [3], security [4], and fault tolerance [5], as well as emerging disciplines such as dynamic system adaptation, diversification¹, and trust maintenance.

2. Survivability Requirements

In this paper, we focus on requirements definition for survivable software systems. Figure 1 depicts an iterative model for defining survivable system requirements. We recognize that survivability must address not only requirements for software functionality, but also requirements for software usage, development, operation, and evolution. Thus, five specific types of requirements definition are relevant to survivable systems in the model

¹ From the following document currently submitted for publication: "Systematic Generation of Stochastic Diversity in Survivable System Software," by R.C. Linger.

of Figure 1, as discussed below. Other areas of survivability requirements definition may emerge in the future.

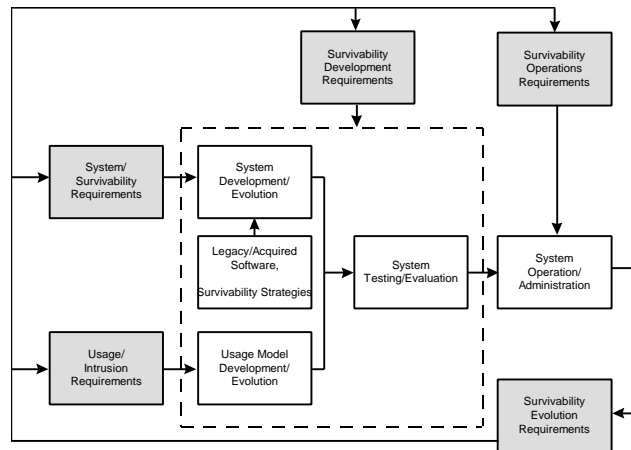


Figure 1. Requirements Definition for Survivable Systems

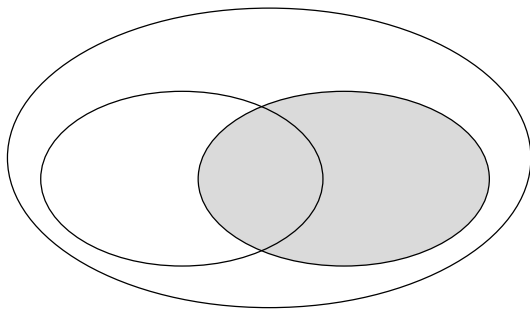
System/Survivability Requirements. In this exposition, *system requirements* refers to traditional user functions that a system must provide. For example, a network management system must provide user functions for monitoring network operations, adjusting performance parameters, and so forth. System requirements also include non-functional aspects, such as timing, performance, and reliability. *Survivability requirements* refer to system capabilities for the delivery of essential services in the presence of attacks and intrusions, and recovery of full services.

Figure 2 depicts the integration of survivability requirements with system requirements at node and network levels. First, survivability requires that system requirements be organized into *essential services* and *non-essential services*, perhaps organized in terms of user categories or business criticality. Essential services must be maintained even during successful intrusions; non-essential services are to be recovered after intrusions have been dealt with. Essential services may be further stratified into a number of levels, each embodying fewer and more vital services, as a function of increasing severity and duration of intrusion. It is also possible that the set of essential services may vary in a more dynamic manner, depending on a particular attack scenario and the resulting situation. In this dynamic case, services that are essential under one scenario may not be essential under another, resulting in different combinations of essential services that are scenario dependent. Thus, definitions of requirements for essential services must be augmented with appropriate survivability requirements. As shown in Figure 1, survivable systems may also

include legacy and COTS components not originally developed with survivability as an explicit objective. Such components may provide both essential and non-essential services and may engender special functional requirements for isolation and control through wrappers and filters to help permit safe use in a survivable system environment.

Second, Figure 2 shows that survivability itself imposes new types of requirements on systems for *resistance* to, *recognition* of, and in particular, *recovery* from intrusions and compromises [6]. These survivability requirements are supported by a variety of existing and emerging *survivability strategies*, as noted in Figure 1 and discussed in more detail below. Finally, Figure 2 depicts *emergent behavior requirements* at the network level. These requirements are characterized as “emergent” because they result from the collective behavior of node services communicating across the network, without benefit of centralized control or information. These requirements deal with the survivability of overall network capabilities, for example, capabilities to route messages between critical sets of nodes regardless of how intrusions may damage or compromise network topology.

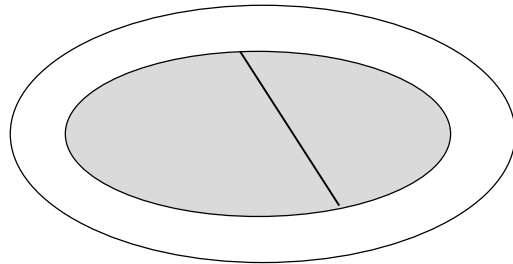
We envision survivable systems as being capable of adapting their behavior, function, and resource allocation in response to intrusions. When necessary, for example, functions and resources devoted to non-essential services could be reallocated to the delivery of essential services and intrusion resistance, recognition, and recovery. Requirements for such systems must specify the behavior for adaptation and reconfiguration in response to intrusions.



Systems can exhibit large variations in survivability requirements. Small local networks may have few or even no essential services, with acceptable manual recovery times measured in hours. Large-scale networks of networks may be required to maintain a core set of essential services, with automated intrusion detection and recovery times measured in minutes. Embedded command and control systems may require essential services to be maintained in real time, with recovery periods measured in milliseconds. The attainment and maintenance of survivability consumes resources in system development, operation, and evolution. Survivability requirements for a system should be based on the costs and risks to an organization associated with loss of essential services.

Usage/Intrusion Requirements. Survivable system testing must demonstrate the performance of essential and non-essential system services, as well as the survivability of essential services under intrusion. Because system performance in testing (and operation) depends totally on the usage to which it is subjected, an effective approach to survivable system testing is based on usage scenarios derived from usage models [7, 8].

Usage models are developed from *usage requirements*, which specify legitimate usage environments and all possible usage scenarios. Usage requirements for essential and non-essential services must be defined in parallel with system and survivability requirements. Furthermore, intrusion usage must be treated on a par with legitimate usage, and *intrusion requirements*, which specify intrusion usage environments and all possible scenarios of intrusion use, must be defined as well. In this approach, intrusion usage is modeled in conjunction with the legitimate use of system services. Figure 3 depicts the relationship between legitimate and intrusion usage. Intruders may engage in usage scenarios beyond legitimate scenarios, but may also employ legitimate usage for purposes of intrusion if they become privileged to do so.



Development Requirements. Survivability places stringent requirements on system development and testing practices. Software errors can have a devastating effect on system survivability and provide ready opportunities for intruder exploitation. Sound engineering practices are required to create survivable software. We assert the following five principles, four technical and one organizational, as example requirements for survivable system development and testing practices:

1. precise specification of required functions in all possible circumstances of use
2. correctness verification of implementations with respect to function specifications
3. specification of function usage in all possible circumstances of use, including intruder usage
4. testing and certification based on function usage and statistical methods
5. establishment of permanent readiness teams for system monitoring, adaptation, and evolution

Sound engineering practices are required to deal with legacy and COTS software components as well.

Operations Requirements. Survivability places demands on the requirements for system operation and administration to define and administer survivability policies, monitor system usage, respond to intrusions, and evolve system functions as necessary to ensure survivability as usage environments and intrusion patterns change over time.

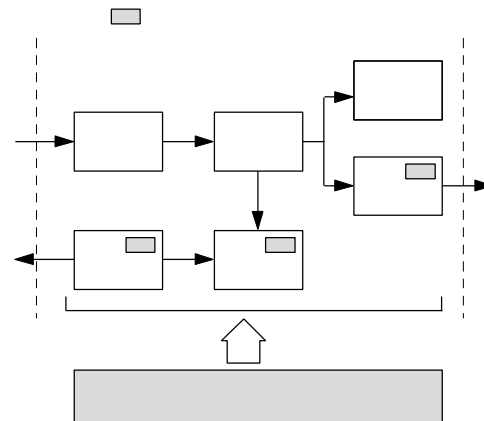
Evolution Requirements. System evolution is an inevitable necessity to respond to user requirements for new functions and increasing intruder knowledge of system behavior and structure. In particular, survivability requires that system capabilities evolve more rapidly than intruder knowledge, to prevent the accumulation of information about invariant system behavior and structure needed to achieve successful penetration and exploitation.

3. Requirements Definition for Essential Services

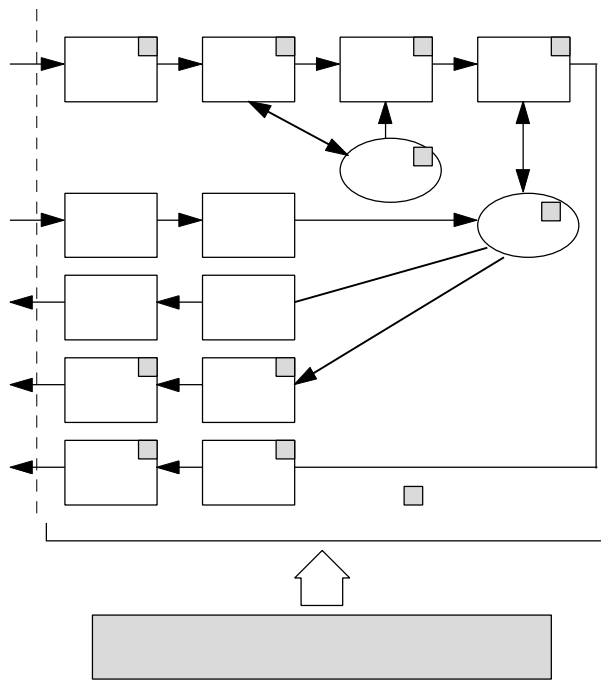
The preceding discussion distinguishes between essential and non-essential services. At the highest level, each system requirement needs to be examined to determine whether it corresponds to an essential service. The set of essential services must form a viable subsystem relative to the original system. In the event that levels of essential services are required, the set of services provided at each level must be examined for completeness and coherence. As noted above, the set of essential services could vary in a more dynamic way, depending upon particular scenarios or situations. In addition, requirements must be defined for transitioning to and from essential service modes.

In distinguishing essential and non-essential services, all the usual requirements definition processes and methods can be applied. Elicitation techniques such as those described in *Software Requirements Engineering* [9] can help to identify essential services, and a tradeoff and cost/benefit analysis can help to determine appropriate sets of services that sufficiently address business survivability risks and vulnerabilities. Provisions for the traceability of survivability requirements through design and code must be established, and special test cases would be required as well. As noted above, the simulation of intrusion through intruder usage scenarios would be included in the testing strategy.

Consider an example of the tradeoff between essential and non-essential services. Figure 4 depicts a block diagram that represents the highest level partitioning of functional requirements for a radar system, itself perhaps embedded in a large-scale command and control network. Although one could argue that the entire radar



system needs to survive, for purposes of illustration, we assume that a subset of the system requirements correspond to essential services. In this system, essential services might include sending default commands to the radar if the real commands are not available due to intrusion, and ensuring that critical communications messages, such as the notification of incoming missiles, are sent even if communication channels are compromised. Other, more routine communications would likely not be included in the survivability requirements. In this way, the system can continue to operate in a compromised state, even in the presence of intrusion. As shown in Figure 4, the Send and Form Radar Commands blocks would thus contain survivability requirements, as would the Send/Receive Messages block. In turn, these requirements assume the presence of supporting real-time services, such as communications and task initiation services, which would also need to survive.



As the architecture for the system is developed, we might arrive at a task structure similar to that shown in Figure 5. In this architecture, it is evident that the set of essential services is embedded within and distributed across tasks, and that the execution of survivable services would potentially involve many modules and interfaces. This may or may not be a viable strategy for ensuring

survivability, as it provides many points for penetration. An alternate architecture might combine and isolate essential services, to provide a separate “survivable environment” that could be managed and controlled more effectively when intrusion occurs. However, such an architecture could suffer reduced performance and increased maintenance costs. This discussion illustrates the close relationship and tradeoff between survivability requirements and system architectures.

4. Requirements Definition for Survivability Services

In addition to specifying requirements for essential and non-essential services, an additional set of requirements for survivability services must also be defined. As noted above, these services can be organized into three general categories, namely: resistance, recognition, and recovery. These survivability services must operate in an intruder environment that can be characterized by the following three distinct phases of intrusion:

Penetration Phase. In this phase, an intruder is attempting to gain access to a system through various attack scenarios, ranging from random inputs by hobbyist hackers to well-planned attacks by professional intruders designed to capitalize on known system vulnerabilities.

Exploration Phase. In this phase, the system has been penetrated, and the intruder is exploring internal system organization and capabilities to learn how to exploit the access to achieve intrusion objectives.

Exploitation Phase. In this phase, the intruder has gained access to desired system facilities and is now performing operations designed to compromise system capabilities.

Penetration, exploration, and exploitation create a spiral of increasing intruder authority and an ever-widening circle of compromise. For example, penetration at the user level is typically employed as a means to explore for root-level vulnerabilities. User-level authorization is then employed to exploit those vulnerabilities to achieve root-level penetration. Furthermore, a compromise of the weakest host in a networked system allows that host to be used as a stepping-stone to comprise other more protected hosts.

Requirements definitions for resistance, recognition, and recovery services embody selected survivability strategies

to deal with these phases of intrusion. Some strategies, such as firewalls, are the product of extensive research and development and are used extensively in current bounded networks. The following new strategies are emerging as necessary responses to the unique challenges of unbounded networks.

Resistance Service Requirements. Resistance refers to the capability of a system to deter attacks. Thus, resistance is important in the penetration and exploration phases of an attack, prior to the point where actual exploitation occurs. Current strategies for resistance include the use of firewalls, authentication, and encryption. Diversification is an example of a strategy that will likely become important in future unbounded networks.

Diversification requirements must define a planned variation in survivable system function, structure, and organization, together with a means for achieving it. Diversification is intended to create a “moving target” to intruders and to render ineffective the accumulation of system knowledge as an intrusion strategy. Diversification also eliminates intrusion opportunities associated with multiple nodes that execute identical software and thus exhibit identical vulnerabilities. Such systems offer tempting economies of scale to intruders, since all nodes can be penetrated once one node has. Diversification requirements can include a variation in programs, retained data, and network routing and communication. For example, systematic means can be defined to randomize software programs while preserving functionality².

Recognition Service Requirements. Recognition refers to the capability to recognize attacks, or to recognize the probing that may precede attacks. The ability to react or adapt in the face of intrusion is central to the capacity of a system to survive an attack that cannot be completely repelled. Reaction or adaptation is impossible without some form of recognition, and thus recognition is essential in all three phases of attack.

A substantial body of research and development exists in this area. Current strategies for attack recognition include not only state-of-the-art work in intrusion detection, but also more mundane but nevertheless effective techniques of logging and frequent auditing, as

² From the following document currently submitted for publication: “Systematic Generation of Stochastic Diversity in Survivable System Software,” by R.C. Linger.

well as follow-up investigations of reports generated by ordinary error-detection mechanisms. There are two types of advanced intrusion-detection techniques: anomaly detection and pattern recognition. Anomaly detection is based upon models of normal user behavior. These models are often established through the statistical analysis of usage patterns. Deviations from normal usage patterns are flagged as suspicious. Pattern recognition is based upon models of intruder behavior. User activity that matches a known pattern of intruder behavior raises an alarm.

The requirements for future survivable networks will likely employ additional strategies, such as self-awareness, trust maintenance, and black-box reporting. Self-awareness refers to the establishment of a high-level semantic model of the computations that a component or system is executing or has been asked to execute. A system or component that “understands” what it is being asked to do is in a position to refuse those actions that would be dangerous, compromise a security policy, or adversely impact the delivery of minimum essential services. By trust maintenance, we refer to a requirement for periodic queries among the components of a system (e.g., among the nodes in a network) to continually test and validate trust relationships. The detection of intrusion signs would trigger an immediate test of trust relationships. Black-box reporting refers to a dump of system information that could be retrieved from a crashed system or component for analysis by the rest of the system to determine the cause of the crash (e.g., design error or specific intrusion type), and thereby prevent other components from suffering the same fate.

In summary, a survivable system design must include explicit requirements for attack recognition. These requirements will ensure the use of one or more of the strategies described above, through the specification of architectural features, automated tools, and manual processes. Since intruder techniques are constantly advancing, it is essential that recognition requirements be subject to frequent review and continuous improvement.

Recovery Service Requirements. Recovery refers to a system’s ability to restore services after an intrusion has occurred and to improve its capability to resist or recognize future intrusion attempts. Recovery also contributes to a system’s ability to maintain essential services during intrusion.

The requirements for recoverability are what most clearly distinguish survivable systems from merely secure systems. Traditional computer security leads to the design of systems that rely almost entirely on hardening (i.e., resistance) for the protection of system resources and services. Once security is breached, damage may soon follow with little to stand in the way. As stated earlier, the ability of a survivable system to react or adapt in the face of an active intrusion is central to the capacity of a system to survive an attack that cannot be completely repelled. Thus, recovery is crucial during the exploration and exploitation phases of intrusion.

Recovery strategies in use today include the replication of critical information and services, the use of fault-tolerant designs, and a variety of backup systems for hardware and software, including maintaining master copies of critical software in isolation from the network. Future recovery strategies will most certainly include dynamic system adaptation, which will not only help a system recover from a current attack, but also permanently improve a system's ability to resist, recognize, and recover from future intrusion attempts. For example, a recoverability requirement for a survivable system may include infrastructure support for the capacity to inoculate the entire system against newly discovered security vulnerabilities, through the automated distribution and application of security fixes to all network elements. Similarly, recoverability requirements may specify that intrusion-detection rule sets are to be updated in a timely manner, in response to reports of known intruder activity from an authoritative source of security information, such as the CERT Coordination Center.

In summary, explicit requirements for recovery are crucial for the design of a survivable system. Recovery requirements make adaptability an integral part of a system's design. As was the case for resistance and recognition requirements, the constant evolution of intruder techniques makes it essential that recovery requirements be subject to frequent review and continuous improvement.

5. Techniques for Representing Survivability Goals and Requirements

5.1 Expressing and Analyzing Survivability Goals

In some areas of software engineering, there is a good understanding of how to express required system goals

and relate them to demonstrated capabilities of various software engineering techniques [10]. For example, suppose that a quality goal is expressed in terms of defects per KSLOC. A sample goal might be to achieve less than 0.1 defects per KSLOC of released code. Defects might in turn be broken down into major and minor defects, with appropriate definitions and subgoals.

To achieve such a goal, a number of techniques could be employed to prevent or detect code defects. These techniques could include, for example, code inspections [11], correctness verification [12], and cleanroom statistical testing [13]. In each case, published evidence is available to support projections of the number of errors that could be avoided or removed.

In illustration, assume that X defects per KSLOC are injected, and that published studies show that Y defects can be removed by inspections, Z defects by correctness verification, and W defects by statistical testing. These projections are easily represented in tabular format, as shown in Table 1:

Injected Defects	Removed by Code Inspection	Removed by Correctness Verification	Removed by Statistical Testing	Defect Quality Goal
X	Y	Z	W	less than 0.1 per KSLOC

Table 1. Sample Defect Projections

In order to achieve the goal, the following relation must hold:

$$X - (Y+Z+W) < 0.1/KSLOC$$

Projections such as this can be extremely useful in managing development. They can be used to help select development processes, and as a benchmark for measuring intermediate project results and improving the processes as necessary.

Such a tabular format can also be used to analyze survivability goals and the techniques used to achieve them. Survivability goals constitute a major part of the requirements definition for survivability development methods as depicted in Figure 1. For example, a survivability goal for a network system might be to ensure no more than one successful intrusion in a 24-hour period, with no loss of essential service. To achieve such a goal, essential services must be identified, and techniques must be selected to prevent intrusions, such as firewalls, encryption, diversification, and trust

maintenance. This analysis would also become part of the survivability specification. Intrusions could be characterized in terms of attempts, with each prevention technique projected to produce a measurable result in reducing the likelihood of intrusion success, based on published evidence in similar systems. This is shown in Table 2 below.

Intrusion Attempts	Avoided by Firewalls	Avoided by Encryption	Avoided by Diversification	Avoided by Trust Maint.	Intrusion Goal
X	Z	Y	W	U	no more than 1 per 24 hours

Table 2. Sample Intrusion Projections

To achieve the intrusion goal, the following relation must hold:

$$X - (Y+Z+W+U) \leq 1 \text{ per 24 hours}$$

Such an analysis is useful in managing development, selecting intrusion avoidance strategies, and comparing projected and actual performance achieved. In complex applications, survivability goals may require partitioning into localized subgoals to permit the analysis of specific forms of intrusion and corresponding avoidance techniques. The cumulative effect of multiple intrusion strategies over time must also be considered. Full analysis may lead to elaborate spreadsheet models that encompass the requirements for a variety of intrusion strategies and defensive measures. Because new methods of intrusion detection and response are continually being discovered, these models constitute reusable project assets that can be expected to evolve over time.

5.2 Expressing and Analyzing Survivability Requirements

Consider the usual textual method of representing requirements, for example, *Commands shall be sent to the radar every n milliseconds.* A refinement of this statement might be: *Computed commands shall be sent to the radar communication subsystem every n milliseconds. In the event that the computed commands are not available, a default set of commands shall be sent to the radar communication subsystem.* In the refinement, we can identify the second sentence as an essential service and annotate it as such, for example, *In the event that the computed commands are not available, a default set of commands shall be sent to the radar communication subsystem (ES-02).*

In some cases, an essential service may be different from ordinary requirements, in that it occurs in a degraded mode. For example, the ordinary requirement might be *X number of clutter returns will be discarded in Y milliseconds by the Radar Returns Processing function,* whereas the essential service requirement might be *X number of clutter returns will be discarded in 2Y milliseconds by the Radar Returns Processing function (ES-05).*

Resistance requirements could be standalone requirements that do not correspond to normal function. For example, a requirement might be *The Send/Receive Messages function shall use encryption techniques on all messages that are sent (ES-07).* Recognition and recovery requirements could be identified similarly in a special way. Essential service requirements could then be tracked through the design and code using the set of essential services to be extracted and analyzed as a standalone survivability specification.

As noted earlier, levels of essential service requirements may be necessary, for example, sending a communications message with notification of an incoming missile might be an essential service requirement at a different priority level. If the primary mission is to preserve the radar, sending out default commands might be the highest priority level of essential service requirements. If the primary mission is to prevent the impact of an incoming missile, sending the communications message could be the highest priority level of essential service requirements. Both might be considered equally important and included at the highest level of essential service requirements. It is easy to see that essential service requirements might include not only behavioral requirements, but also non-behavioral requirements, as well as data and interface specifications.

5.3 Expressing Survivability Requirements in Box Structure Form

The level of precision and completeness necessary for specifying survivable system requirements suggests the application of engineering methods such as box structures [14, 15]. In particular, the black-box form of box structures permits the precise definition of required behavior of systems and system components in all possible circumstances of use. The black-box transition function is as follows:

$$(\text{stimulus, stimulus history} \rightarrow \text{response})$$

In this model, the current stimuli and history of stimuli determine the response to be produced. Black-box requirements specifications are often expressed in tabular form, with columns for stimuli, conditions on stimulus histories, and corresponding responses. By making all possible behaviors visible and explicit, such specifications can be checked effectively for completeness and correctness, and essential services can be readily identified, annotated, and if desired, grouped together. For example, a fragment of the black-box requirements specification for the Radar Returns Processing function discussed above could be expressed as follows, with designation of normal and degraded mode (essential) transition types:

Type	normal mode
Stimulus	clutter return
Condition on Stimulus History	X clutter returns(history) = true and Intrusion(history) = false
Response	discard X clutter returns in Y milliseconds

Type	degraded mode – Essential
Stimulus	clutter return
Condition on Stimulus History	X clutter returns(history) = true and Intrusion(history) = true and Recovery = false
Response	discard X clutter returns in 2Y milliseconds

Table 3. Sample Black-Box Requirements

6. Conclusions

Survivable system requirements definition represents a new and challenging area in software engineering. The proliferation and risks of large-scale unbounded network systems in vital areas such as finance, defense, and health care, makes it essential to create effective methods for survivability requirements analysis and specification. In particular, essential services need to be identified in the process of developing survivable systems requirements. This set of services needs to form a viable system in its own right. Additional tools, architectural strategies, and development methods will be needed to support such systems.

7. Research Directions

There are a number of promising research areas that address requirements definition for survivable systems. The plans for the Survivable Network Technology team at the Software Engineering Institute (SEI) include the following:

- 1) adaptation and development of architectural description techniques to adequately describe large-scale distributed systems with survivability attributes
- 2) representation of the intruder environment through intruder usage models
- 3) creation of an analysis method to evaluate survivability as a global emergent property from architectural specification
- 4) refinement of the analysis technology and instruments through pilot tests of real distributed systems

Additional areas of future work could include the following:

- 1) definition of suitable measures for survivability strategies
- 2) development of new strategies in addition to resistance, recognition, and recovery for survivability
- 3) strengthening of the link to requirements engineering as well as other traditional software engineering processes and techniques

8. Acknowledgments

This paper describes some of the work of the Survivable Network Technology Project at the SEI. We would like to acknowledge the fruitful discussions with and comments from Bob Ellison, David Fisher, and Tom Longstaff.

9. References

- [1] Lipson, H. & Longstaff, T.(eds). *Proceedings of the SEI 1997 Information Survivability Workshop*. San Diego, CA, Feb. 12-13, 1997. Los Alamitos, CA: IEEE Computer Society Press, 1997.
- [2] Musa, J.D.; Iannino, A.; & Okumoto, K. *Software Reliability: Measurement, Prediction, and Application*. New York, NY: McGraw-Hill, 1987.
- [3] Leveson, N. G. *Safeware: System Safety and Computers*. Reading, MA: Addison-Wesley, 1995.
- [4] Clark, R.K.; Greenberg, I.B.; Boucher, P.K.; Lund, T.F.; Neumann P.G.; Wells, D.M.; & Jenson, E.D. "Effects of Multilevel Security on

- Real-time Applications,” 120-129. *Proceedings of the 9th Annual Computer Security Applications Conference*. Orlando, FL, December 6-10, 1993. Los Alamitos, CA: IEEE Computer Society Press, 1993.
- [5] Mendiratta, V.B. “Assessing the Reliability Impacts of Software Fault-Tolerance Mechanisms,” 99-103. *Proceedings of the 7th International Symposium on Software Reliability Engineering*. White Plains, NY, Oct 30-Nov 2, 1996. New York: IEEE Computer Society Press, 1996. IEEE Catalogue Number: 96TB100090.
- [6] Ellison, R.J.; Fisher, D.; Linger, R.C.; Lipson, H.F.; Longstaff, T.; & Mead, N.R. *Survivable Network Systems: An Emerging Discipline* (CMU/SEI-97-TR-013) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997.
- [7] Mills, H.D. “Certifying the Correctness of Software,” 373-381, Volume 2. *Proceedings of the 25th Hawaii International Conference on System Sciences*. Kauai, Hawaii, January 7-10, 1992. Los Alamitos, CA: IEEE Computer Society Press, 1992.
- [8] Trammell, C.J. “Quantifying the Reliability of Software: Statistical Testing Based on a Usage Model,” 208-218. *Proceedings of the Second IEEE International Symposium on Software Engineering Standards*. Montreal, Quebec, Canada, August 21-25, 1995. Los Alamitos, CA: IEEE Computer Society Press, 1995.
- [9] Thayer, R.H. & Dorfman, M. (Eds). *Software Requirements Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1997.
- [10] Ebert, C. “Dealing with Nonfunctional Requirements in Large Software Systems.” *Annals of Software Engineering 3*: (September, 1997): 367-395. Baltzer Science Publishers, Amsterdam, Netherlands, 1997.
- [11] Ebenau, R.G. & Strauss, S.H. *Software Inspection Process*. New York, N.Y.: McGraw-Hill, 1994.
- [12] Linger, R.C. Ch. 6, “Cleanroom Process Model,” 111-131. *Cleanroom Software Engineering: A Reader*. Oxford, England: Blackwell Publishers, 1996.
- [13] Linger, R.C. & Trammell, C.J. *Cleanroom Software Engineering Reference Model* (CMU/SEI-96-TR-022, ADA319071). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996.
- [14] Mills, H.D.; Linger R.C.; & Hevner, A.R. *Principles of Information Systems Analysis and Design*. Orlando, FL: Academic Press, 1986.
- [15] Mills, H.D.; Linger, R.C.; & Hevner, A.R. Ch. 7, “Box-Structured Information Systems,” 139-167. *Cleanroom Software Engineering: A Reader*. Oxford, England: Blackwell Publishers, 1996.