7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 27 – 29, 2004

DEVELOPMENT OF SURVIVABLE SYSTEMS

Valentin MEȚGHER
Compudava S.R.L.
Str. 31 August 67B,
Chisinau, Moldova
vmetgher@compudava.com

Abstract. *The vast majority of modern systems serving organization needs are closely linked into the computer networks which have evolved over recent years into unbound networks with no central authority and distributed administrative control providing a low visibility and control on the entire network. At the same time, organizational dependencies on networks are increasing, and the risks and consequences of intrusions and compromises are amplified. Incorporating survivability capabilities into organisation's systems can mitigate these risks. It is important to mention that survivability doesn't guarantee that the system will not be attacked and penetrated, but importantly it shall preserve the essential services during its operation. The development of survivable systems shall be assured by sound engineering practices throughout the system life-cycle.*
Keywords: *survivability, security, life-cycle, mission, tolerance, system, process, risks.*

Introduction

Survivability is defined as the capability of a system to fulfil its mission, in a timely manner, in the presence of attacks, failures, or accidents. The term mission refers to a set of very high-level requirements or goals.

It is important to recognize that it is the mission fulfilment that must survive, not any particular subsystem or system component. Central to the notion of survivability is the capability of a system to fulfil its mission, even if significant portions of the system are damaged or destroyed.

A key characteristic of survivable systems is their capability to deliver essential services in the face of attack, failure, or accident. Central to the delivery of essential services is the capability of a system to maintain essential properties (i.e., specified levels of integrity, confidentiality, performance, and other quality attributes) in adverse environments.

To maintain their capabilities to deliver essential services, survivable systems must exhibit the four key properties, and namely: resistance, recognition, recovery (the three R's), and adaptation [1].

As the computer systems become more complex and sophisticated the attacks from the potential adversaries become more ingenious. The attack targets range from common services of organizations such as mail, DNS servers to web services, which are not becoming secure only by protecting the perimeter but require proper planning and for applications proper development process in place in the early stages of developing a system.

Security is routinely addressed late in the development lifecycle. A common scenario in the current commercial development practices is that non-functional requirements, such as security, are simply ignored considering the threats present. Systems are vulnerable to attacks without the client knowing about this, waiting for a potential intruder to discover the vulnerability and attack the system, only after which the security is addressed for that system.

The lack of formal methods rigorously applied for the development process is a critical factor when addressing the survivability of the systems. Current software development life-cycle models are not focused on creating survivable systems and exhibit shortcomings when the goal is to develop systems with a high degree of assurance of survivability.

Table 1. Survivability elements during life-cycle activities.

Life-Cycle Activities	Survivability elements
Initiation	- Mission definition and analysis of consequences of failure; - Planning integration of survivability into the life-cycle process.
Requirements	- Survivability requirements definition; - Establishing the essential services required to survive. - Specifying the usage/intrusion scenarios.
Design	- Integrating the survivability requirements into the architecture; - Design is analysed if it corresponds with the mission of the system.
Development	- Four principles of coding; - Integrated evaluation approach.
Testing	Testing against the threat patterns and usage scenarios.
Operation	Maintaining the survivability of the system via evolving design.

This isolation of survivability considerations from primary system-development tasks results in an unfortunate separation of concerns. Survivability should be integrated and treated on a par with other system properties, to develop systems with required functionality and performance that can also withstand failures and compromises.

Important design decisions and tradeoffs become more difficult when survivability is not integrated into the primary development life-cycle. In addition, tools for supporting survivability engineering are often not integrated into the software development environment. For each life-cycle activity, survivability goals should be addressed, and methods to ensure survivability incorporated [2].

In this paper, a software development life-cycle model for survivability will be presented and techniques that can be applied during new development activities to support survivability goals will be illustrated. In Table 1 the main survivability elements during life-cycle activities are reflected.

Development of the systems with critical requirements such as survivability shall incorporate the best software engineering practices:

- Sound software engineering process: *requirements, design, development, testing, project management.*
- Robust and secure evolvable interoperable architectures that avoid excessive dependence on untrustworthy components.
- The use of formal methods is recommended in particularly critical applications, and can help move the current highly unpredictable ad-hoc development process into a much more predictable formal development process [3].
- Also, no methods will help if people participating in the process will not be properly trained and possess the necessary skills.

Iterative system development

Two life-cycle models, the waterfall and the spiral models, are probably the most well known.

The waterfall model represents a linear process, where clear inputs and outputs are defined for each phase of development of a product and one phase can start only after the previous one has finished.

In the spiral model (more widely used recently) the development is iterative with essential activities such as prototyping, architecture first approach, re-use and early assessment of risks. The spiral model can be adapted/modified to

satisfy specific requirements such as for survivable systems.

The iterative development approach is spanned throughout the entire life-cycle of a system ensuring the *mission fulfilment* and *survivability strategy*, being the driving factor, are omnipresent and can be traced through all of the project phases as illustrated in Fig.1.

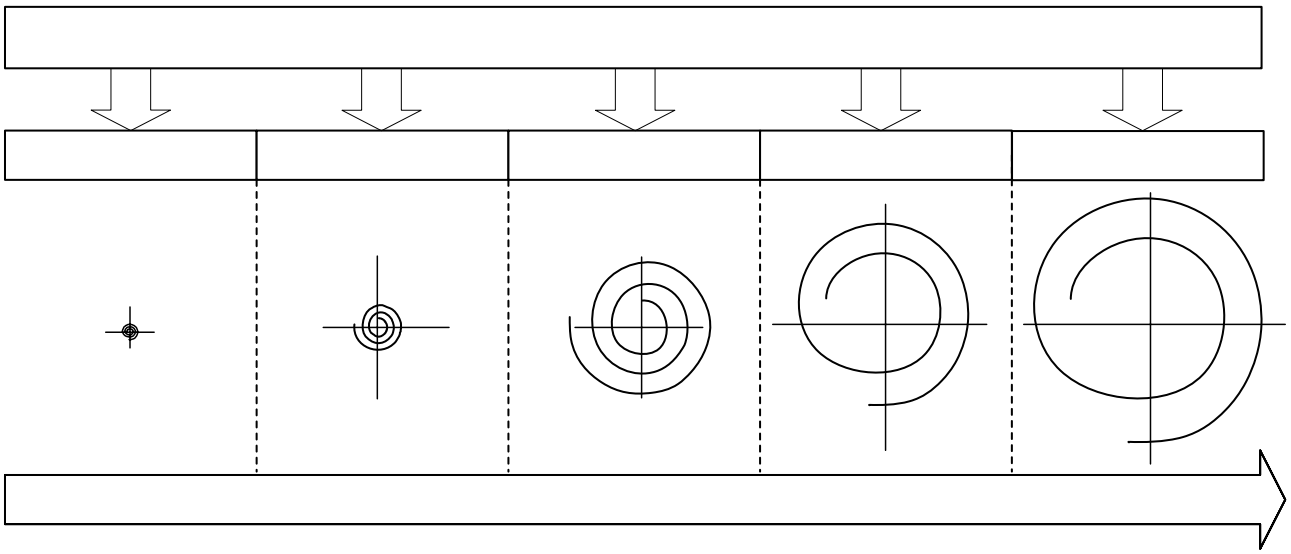

Fig 1. The Development process of survivable systems.

Requirements

The key activities of the requirements process are: elicitation, analysis, specification and validation [4]. The different activities of the requirement process are repeated until an acceptable level is achieved. Modern requirements processes use extensively UML tools to automate, ensure accuracy and traceability of information. Two common formats for requirements are Formal Requirements and Use Cases.

The proper definition of survivability requirements is crucial for achieving the survivability of a system. The survivability requirements may vary for different systems according to the scale, to the cost and risks for the organization of losing the essential services.

There shall be well defined the model of a mission (specified at the inception phase) that a specific system is intended to fulfil in adverse environments and conditions.

Functional/Survivability. The requirements of a survivable system are based on functional/system requirements defining how the system will operate under normal conditions and survivability requirements specifying the capabilities of a system to deliver essential services under attack.

There shall be established a set of essential services the system must deliver and each system requirement shall be checked if it is related to an essential service.

The survivability requirements are represented by these categories: Resistance, Recovery, Recognition and Adaptation [2]:

- Resistance Service Requirements. Resistance is the capability of a system to deter attacks. Resistance is thus important in the penetration and exploration phases of an attack, before actual exploitation. Current strategies for resistance include the use of firewalls, authentication, and encryption.

- Recognition Service Requirements. Recognition is the capability of a system to recognize attacks or the probing that precedes attacks. The ability to react or adapt during an

intrusion is central to a system's capacity to survive an attack that cannot be completely repelled. To react or adapt, the system must first recognize it is being attacked. In fact, recognition is essential in all three phases of attack.

- Recovery Service Requirements. Recovery is a system's ability to restore services after an intrusion has occurred. Recovery also contributes to a system's ability to maintain essential services during intrusion. Recovery strategies in use today include replication of critical information and services, use of fault-tolerant designs, and incorporation of backup systems for hardware and software.

- Adaptation and Evolution Service Requirements. Dynamic adaptation permanently improves a system's ability to resist, recognize, and recover from intrusion attempts. For example, an adaptation requirement may be an infrastructure that enables the system to inoculate itself against newly discovered security vulnerabilities by automatically distributing and applying security fixes to all network elements.

Usage/Intrusion Requirements. Survivable-system testing must demonstrate the correct 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 system's use, an effective approach to survivable-system testing is based on usage scenarios derived from usage models [2].

There are other system survivability requirements throughout the life-cycle related to *development, operation, and evolution.*

In the iterative approach of development the requirements are adjusted/modified throughout the entire life-cycle managed via a configuration management process. When the requirements are enough detailed the design/architecture phase can start.

Design

The software design must describe the architecture of the system, how the system is decomposed and organized into components, and must describe the interfaces between these components. It must also describe these components into a level of details suitable for allowing their construction [4].

Software design consists normally of two main activities: *architectural design* and *component design.*

The design along with requirements process are the most critical and effort consuming activities in the life-cycle, especially for survivable systems where much of analysis has to be done while developing the conceptual architecture.

The design passes through a number of iterative activities: Definition; Analysis; Evaluation.

Software Design Definition. The conceptual architecture is developed based on the requirements for the system including survivability; the architecture styles and design patterns are selected; the respective framework is applied for object-oriented design. The architectural design describes how the software is decomposed and organized into components, including operating systems, network protocols, encryption key management, authentication systems. The main components shall be carefully considered/selected taking into account the survivability element.

Software Design Analysis. The design is analysed if it corresponds to the mission of the system *and against the* main survivability and quality attributes as security, reliability, performance, fault-tolerance. The essential services defined earlier in the life-cycle have to be reflected in the architecture to ensure the system (essential services) will operate in adverse environments. The main attributes of the survivability (recognition, resistance, recovery and availability) shall be considered

for each component of the architecture, especially in the context of supporting the main services.

Design Evaluation. The design shall be evaluated by using iterative design reviews, but also by using dynamic techniques, such as simulation and prototyping. The architecture shall be evaluated against the common threats (grouped in a regularly updated library) to the systems to ensure its resistance to such attacks.

Each system architecture will have its own number of iterations, but finally a trade-off shall be found between various attributes (such as security vs. performance for example) to produce an architecture which is in line with the system's mission, and has the necessary survivability attributes incorporated.

Development

The development is the act of constructing the system through a combination of coding, validation and testing activities. Proper coding practices shall be applied to avoid the existence of vulnerabilities out of poor coding practices, which can lead to backdoors to the system intrusion.

The four principles can be applied for building the system, which significantly affects the way the software is constructed: *reduction of complexity*; *anticipation of diversity*; *structuring for validation*; *use of external standards*.

Integrated evaluation is a powerful approach which includes periodic internal checks for intermediate versions of code to ensure in an iterative manner that it works correctly.

- Unit testing. Used by developers to test well-defined software units after their completion.
- Daily builds and smoke tests. A methodology ensuring the code keeps stable daily.
- Code inspections. Manual way of checking for abstract classes of errors and adherence to coding standards adopted by the organization.

Important is for the implemented system that it will be as bug free as possible [5] and behave in all circumstances in the way it was specified and intended to work.

Testing

In the modern software engineering practice, the testing is no longer seen as an activity which starts after the coding phase, but encompasses the whole system development process. Three big stages in testing of complex software systems are distinguished:

i) Unit (testing of finalized components),
ii) Integration (testing of assembled components),
iii) System testing (concerned with the behaviour of the system). System testing deals with testing the non-functional specifications such as security, performance, tolerance, all parts of system's survivability behaviour.

Penetration testing, usage-based testing, and boundary-value analysis are useful approaches for evaluating system survivability.

Penetration testing. The system is tested using a wide range of techniques in order to break the system and interrupt its mission: probing and scanning, brute force attacks, audit at the security, network and application level. The evidence that the system has the survivability attributes is if it performs its essential services even after successful attacks on the system.

Usage-Based Testing. The approach constructs usage models considering the possible users (legitimate and intruders) and their behaviour according to probabilities. The results are used to predict survivability properties.

Boundary-value analysis. Test cases are chosen near the boundaries of the input where the defects tend to concentrate.

"Enough" testing shall be provided and best practices testing techniques [5] applied to provide assurance for survivability properties of the system.

Operation

The evolutionary design shall permit the system to adapt to the changing operating environment along with the evolution of intrusion techniques. Monitoring tools shall be in place to detect anomalies in system behaviour, to detect if survivability related attributes are affected. The survivability shall be sustained by continual incorporation of new survivable solutions through an evolutionary design process.

Conclusions

Developing systems with critical survivability requirements relying on open-bounded uncontrolled networks is a complex and difficult task to achieve.

Some of the challenges are in areas of designing evolvable architectures, developing new robust, secure protocols, develop more secure operating systems, explore the process of dynamic adaptability of the systems, etc.

Although much research is yet to be done in the area of survivability, the path to development of truly survivable systems is by combining the latest advances in computer and network technologies (especially concerning the security, reliability, fault-tolerance), with applying the software engineering best practices and serious discipline and QA throughout the development cycle.

References

[1] Ellison, Robert; Fisher, David (1997). *Survivable Network Systems: An Emerging Discipline*. SEI. Carnegie Mellon University.

[2] Nancy R. Mead, Robert Ellison, Richard C. Linger, Howard F. Lipson, John McHugh (2000). *Life-Cycle Models for Survivable Systems*. CERT. Carnegie Mellon University.

[3] Peter Neuman (2000). *Practical Architectures for Survivable Systems and Networks.* Computer Science Laboratory. SRI international.

[4] *Guide to the Software Engineering Body of Knowledge.* IEEE Computer Society Press. 2001.

[5] Steve Maguire (1993). *Writing Solid Code.* Microsoft press.

[6] M. Haug, E.W. Olsen, L. Consolini (2001). *Software Quality Approaches: Testing, Verification, and Validation*. Springer-Verlag Berlin Heidelberg.