

Design Diversity and the Immune System Paradigm: Cornerstones for Information System Survivability

Algirdas Avižienis

UCLA Computer Science Department
University of California
Los Angeles, CA 90095-1596
aviz@cs.ucla.edu

Faculty of Informatics
Vytautas Magnus University
Kaunas, Lithuania

1. The Contribution: Useful Concepts

The Call for Participation of ISW-2000 defines information survivability as “the ability of a system to continue to fulfill its mission in the presence of attacks, accidents, or failures”. This position paper presents two concepts: the *design diversity* technique [1, 2] and the *immune system* paradigm [3] that already have been found to be useful in assuring the dependability of mission-critical information systems. The above concepts as well as the concept of *fault tolerance* [3, 4] have originated in the research that the author and his associates have conducted since 1960 [5]. The research has shown that these concepts have the potential to enhance information survivability in the presence of attacks as well [6–9]. Our goal at ISW-2000 is to bring the potential usefulness of the concepts to the attention of the survivability community.

Fault tolerance, design diversity and the immune system paradigm become especially important for information survivability when mission-critical systems are built using COTS hardware and software components. Most COTS components have very few, if any at all, built-in features that support survivability, therefore the defenses have to be introduced at the system design level.

The remainder of this position paper summarizes: (1) the *threats* to information survivability; (2) the *defenses*, i.e., the technique of design diversity and the use of the immune system paradigm; and (3) the first *example* of their joint application: the recently described “fault tolerance infrastructure for dependable computing with COTS components” [10].

2. The Universe of Threats

Information systems fail because of two fundamental causes: natural phenomena and human actions. The *natural*

causes are:

- (1) *Random failures*, i.e, permanent physical changes (faults) of hardware components.
- (2) *External environmental phenomena* that interfere with the operation of the system by causing:
 - (2a) *physical damage* to system hardware due to excessive temperatures, water or fire in the facility, earthquakes, etc.;
 - (2b) *corrupted information* (without damage to hardware) due to electromagnetic interference, cosmic rays, solar flares, alpha particles, etc. (transient faults).

The failures due to *human actions* fall into two categories, depending on whether the actions are unintentional or intentional. The *unintentional* human causes are:

- (3) *Design faults*: software “bugs” and hardware “errata” that remain undetected during program or hardware development and manifest themselves during system operation.
- (4) *Interaction faults*: mistakes by system operators, maintenance personnel, and others with access to the system that lead to incorrect operation, system shut-down, or accidental physical damage, such as accidentally cut cables, disconnected cooling, etc.

Finally, there are attacks, i.e., human actions with *malicious intent* [11]:

- (5) Introduction of *malicious logic*: viruses, worms, Trojan horses, logic bombs, etc. into the system.
- (6) *Intrusions* that exploit weaknesses of the system to gain unauthorized access with intent to steal, alter or destroy information.

(7) *Physical attacks* that deliberately cause damage to system hardware or information content. The damage is similar to that of causes (2) and (4) above.

To assure the survivability of a system, effective defenses against all potential causes of failure summarized above need to be incorporated into its design. The problem of providing defenses is especially difficult for systems built of COTS hardware and software components, as discussed below.

On the good news side, we are witnessing a very significant and continuing reduction of failure rates for complex COTS semiconductor devices. For example, recent Intel literature [12] quotes the failure rate of 100 FITs (failures per 10^9 hours) for their best products, with 10 FITs being the goal for the near future. The 100 FIT rate translates into a device MTTF of 10^7 hours, given that unknown “wearout” effects do not appear. It may be reasonably concluded that device failures have become the least critical threat for contemporary information systems.

However, at the same time we observe an increasing severity of the other threats to the dependability and survivability of information systems. Some illustrations are given next.

Susceptibility to environmental interference. The continuing reductions in the size and power level of logic elements raise device susceptibility to *interference* by various kinds of radiation and other environmental factors that cause transient faults, i.e., information is corrupted without device failure.

Design faults. While software “bugs” have been a serious threat for a long time, in the past few years it has become common that complex devices, especially high-performance processors, contain hardware design faults (called “*errata*”) that are discovered after the design is completed. For example, the seven processors of the Intel P6 family in April 1999 had from 45 to 101 reported design faults, of which from 30 to 60 have remained uncorrected to the most recent versions (“steppings”) of the processors, and new “*errata*” are announced at the rate of about one per month [13].

Maintenance mistakes and other unintentional interaction faults have remained a serious concern, while *attacks* of various types have been gaining in numbers and in severity, especially by exploiting the weakness of complex COTS operating systems and other COTS software.

One more consequence of the relative decrease of the random failure threat needs to be noted. Distributed systems were considered to have a “natural” form of fault tolerance, since the loss of one *randomly failed* node was compensable by the redistribution of the workload to the remaining nodes. However, external interference may affect several nodes at once, and the “natural” fault tolerance only exists when adequate spatial separation is provided for the nodes.

Furthermore, design faults are generic, i.e., present in all copies of the same design, and distribution does not offer an advantage. Finally, attacks are not deterred by distribution alone, as long as the nodes are alike. For design faults and attacks we need design diversity in order to prevent node crashes or intrusions of epidemic proportions.

3. The Application of Design Diversity

Design diversity [1, 2] is a fundamental approach to the tolerance of design faults. It is applicable to all elements of an information system: hardware, software, communication links, man/machine interfaces, design tools, etc.

Design diversity is implemented by performing a function in two, three, or more independently designed elements (channels) and then executing a decision algorithm (a comparison or a majority vote) on the results. Frequently the comparison or vote has to be *inexact* because different algorithms are used by the diverse elements. A well-protected implementation of the decision algorithms is the key requirement for successful application of design diversity. Since design diversity is a relatively costly technique, only the mission-critical parts should employ it in large and complex systems that are subject to hardware and software design faults.

Examples of *software* diversity techniques are N-version programming [14] and recovery blocks [15], while *hardware and software* diversity is employed in the flight control computer of the Boeing 777 [16] and Airbus [17] commercial airliners. *Communication link* diversity is exemplified by the use of a satellite link to back up an optic cable or a microwave link on the ground. Furthermore, diversity of *location* is essential to assure information survivability during natural disasters and terrorist attacks.

Early exploratory research of our UCLA group led to the conclusions that N-version programming was a potentially effective means to detect and neutralize malicious logic and to limit the effectiveness of intrusions [6, 8, 9].

4. The Immune System Paradigm

The conventional approach to improving the dependability of systems built from COTS components is to build a software monitor subsystem (such as the Pentium II’s Machine Check Exception Handler) that resides and executes on the COTS hardware elements. A software monitor tries to check all subsystems for indications of failure and records abnormal symptoms. When necessary, it initiates shutdowns, BIST, and restarts. This approach has two major weaknesses: the monitor software itself is unprotected because it resides and executes on a COTS processor, and it limits recovery handling to restarts only.

At the level of a complete server platform [18] additional COTS hardware (buses and controller chips) and system management software is added to support fault detection and recovery. Again, neither the extra hardware nor the software is protected itself. The proliferation of such features adds to system complexity, and the improvement in dependability remains uncertain.

We have proposed a fundamentally different approach: to build a generic *fault tolerance infrastructure* that is analogous in its attributes to the human immune system [3]. In this model, the body is analogous to hardware, and the cognitive processes supported by the body are analogous to software. Upon execution, software delivers the required high-confidence services for which the system is programmed. Four fundamental attributes of the immune system are particularly relevant [19]:

- (1) It functions (i.e., detects and reacts to threats) continuously and autonomously, independently of cognition (the analog of software).
- (2) Its elements (lymph nodes, other lymphoid organs, lymphocytes) are distributed throughout the body, serving all its organs.
- (3) It has its own communication links - the network of lymphatic vessels.
- (4) Its elements (cells, organs, and vessels) themselves are self-defended, redundant and in several cases diverse.

The following section summarizes a system design in which these attributes, the “Immune System Paradigm”, have been implemented as the “fault tolerance infrastructure” that supports design diversity and possesses the immune system attributes summarized above. It is important to note that the infrastructure relieves system software from the handling of hardware problems, but it remains transparent to any software - implemented error detection and recovery procedures.

A different and independently devised analog of the immune system is the “Artificial Immune System” (AIS) of S. Forrest and S. A. Hofmeyr [20]. Its origins are in computer security research, where the motivating objective was protection against illegal intrusions. The analog of the body is a local-area broadcast network, and the AIS protects it by detecting connections that are not normally observed on the LAN. Immune responses are not currently included in the model.

5. The Fault Tolerance Infrastructure for COTS Systems

The “Fault Tolerance Infrastructure” (FTI) is a system composed of four types of special-purpose controllers

called “nodes”. The nodes are ASICs (Application-Specific Integrated Circuits) that are controlled by hard-wired sequencers or by microcode. They do not employ any software. The names of the four kinds of nodes are: (1) A-nodes (Adapter nodes); (2) M-nodes (Monitor nodes); (3) D-nodes (Decision nodes); (4) S^3 -nodes (Startup, Shutdown, Survival nodes).

The purpose of the FTI is to provide protection against all causes of system failure for a computing system composed of COTS components, called C-nodes (Computing nodes). The C-nodes are connected to the A-nodes and D-nodes of the FTI. The C-nodes are COTS microprocessors, memories, and components of the supporting chipset in a high-performance COTS computer system.

The M- and S^3 -nodes form a fault-tolerant controller (the M-cluster) for recovery management of the C- and D-nodes of the system. The A-nodes connect the M-cluster to the C- and D-nodes. The D-nodes serve as the decision elements for diverse C-nodes and also provide the means for the C-nodes to communicate with the M-cluster. A detailed discussion is presented in [10].

The following protection for the COTS system is provided when it is connected to the FTI:

- (1) The FTI provides error detection and recovery support when the COTS system is affected by physical failures of its components and by external interference. The FTI provides power switching for unpowered spare COTS components to replace failed C-nodes in long-duration missions.
- (2) The FTI provides a “shutdown-hold-restart” recovery sequence for catastrophic events that affect either the COTS system or both the COTS and FTI systems. Such events are: a “crash” of the COTS system software, an intensive burst of radiation, temporary outage of COTS system power, etc.
- (3) The FTI provides (by means of the D-nodes) the essential mechanisms to detect and to recover from the manifestations of software and hardware design faults. It is accomplished by the implementation of design diversity, where each channel (i.e., C-node) employs independently designed hardware and software, while the D-node serves as the decision element. Design diversity also supports detection and neutralization of malicious software and intrusions, as well as of mistakes by operators or maintenance personnel.

Finally, it is essential that the nodes and interconnections of the FTI should be designed to provide protection for the FTI system itself as follows:

- (1) Error detection and recovery algorithms are incorporated to protect against random component failures and transient faults due to external interference.

- (2) The absence of software in the FTI provides isolation against software “bugs”, malicious logic, and intrusion.
- (3) The overall FTI design allows the introduction of diverse hardware designs for the A-, M-, S^3 -, and D-nodes in order to provide protection against hardware design faults. Such protection may prove to be not necessary, since low complexity of the node structure could allow complete verification of the node designs.

When interconnected, the FTI and the high-performance COTS computing system form a high-performance computing system that is protected against most system failure causes. This system is called DiSTARS: *Diversifiable Self Testing And Repairing System* and is discussed in detail in [10]. DiSTARS is the first example of a joint implementation of design diversity and the immune system paradigm.

References

- [1] A. Avižienis. Design diversity - the challenge of the eighties. In *Digest of FTCS-12*, pages 44–45, June 1982.
- [2] A. Avižienis and J. P. J. Kelly. Fault tolerance by design diversity: concepts and experiments. *Computer*, 17(8):67–80, August 1984.
- [3] A. Avižienis. Toward systematic design of fault-tolerant systems. *Computer*, 30(4):51–58, April 1997.
- [4] A. Avižienis. Design of fault-tolerant computers. In *Proc. 1967 Fall Joint Computer Conf. AFIPS Conf. Proc. Vol. 31.*, pages 733–743, 1967.
- [5] A. Avižienis and D. A. Rennels. The evolution of fault tolerant computing at the Jet Propulsion Laboratory and at UCLA: 1960-1986. In *The Evolution of Fault-Tolerant Computing*. Springer-Verlag, 1987.
- [6] M. K. Joseph and A. Avižienis. Software fault tolerance and computer security: A shared problem. In *Proc. of the Annual National Joint Conference And Tutorial On Software Quality and Reliability*, pages 428–436, March 1989.
- [7] M. K. Joseph and A. Avižienis. A fault tolerance approach to computer viruses. In *Proc. of the 1988 IEEE Symposium on Security and Privacy*, pages 52–58, April 1988.
- [8] M. K. Joseph. Towards the elimination of the effects of malicious logic: Fault tolerance approaches. In *Proc. 10th National Computer Security Conf.*, pages 238–244, September 1987.
- [9] M. K. Joseph. *Architectural Issues in Fault-Tolerant, Secure Computing Systems*. PhD thesis, Computer Science Department, University of California, Los Angeles, June 1988.
- [10] A. Avižienis. A fault tolerance infrastructure for dependable computing with high-performance COTS components. In *Proc. of the Int. Conference on Dependable Systems and Networks (DSN 2000)*, pages 492–500, June 2000.
- [11] C. E. Landwehr et al. A taxonomy of computer program security flaws. *ACM Computer Surveys*, 26(3):211–254, September 1994.
- [12] Intel Corp. *Intel’s Quality System Databook*. January 1998. Order No. 210997-007.
- [13] A. Avižienis and Y. He. Microprocessor entomology: A taxonomy of design faults in COTS microprocessors. In J. Rushby and C. B. Weinstock, editors, *Dependable Computing for Critical Applications 7*, pages 3–23. IEEE Computer Society Press, 1999.
- [14] A. Avižienis. The methodology of n-version programming. In M. R. Lyu, editor, *Software Fault Tolerance*, pages 23–46. John Wiley & Sons, 1995.
- [15] B. Randell and J. Xu. The evolution of the recovery block concept. In M. R. Lyu, editor, *Software Fault Tolerance*, pages 1–21. John Wiley & Sons, 1995.
- [16] Y. C. Yeh. Dependability of the 777 primary flight control system. In R. K. Iyer, M. Morganti, W. K. Fuchs, and W. Gligor, editors, *Dependable Computing for Critical Applications 5*, pages 3–17. IEEE Computer Society Press, 1998.
- [17] D. Briere and P. Traverse. AIRBUS A320/A330/A340 electrical flight controls: a family of fault-tolerant systems. In *Digest of FTCS-23*, pages 616–623, June 1993.
- [18] Intel Corp. *The Pentium II Processor Server Platform System Management Guide*, June 1998. Order No. 243835-001.
- [19] G. J. V. Nossal. Life, death and the immune system. *Scientific American*, 269(3):52–62, September 1993.
- [20] S. A. Hofmeyr and S. Forrest. Immunity by design: An artificial immune system. In *Proc. 1999 Genetic and Evolutionary Computation Conference*, pages 1289–1296. Morgan-Kaufmann, 1999.