Chapter II
Fault–Tolerant Software: Basic Concepts and Terminology

INTRODUCTION AND OBJECTIVES

After having described the main characteristics of dependability and fault-tolerance, it is analyzed here in more detail what it means that a program is fault-tolerant and what are the properties expected from a fault-tolerant program. The main objective of this chapter is introducing two sets of design assumptions that shape the way our fault-tolerant software is structured—the system and the fault models. Often misunderstood or underestimated, those models describe

- what is expected from the execution environment in order to let our software system function correctly, and
- what are the faults that our system is going to consider. Note that a fault-tolerant program shall (try to) tolerate only those faults stated in the fault model, and will be as defenseless against all other faults as any non fault-tolerant program.

Together with the system specification, the fault and system models represent the foundation on top of which our computer services are built. It is not surprising that weak foundations often result in failing constructions. What is really surpris-
ing is that in so many cases, little or no attention had been given to those important factors in fault-tolerant software engineering. To give an idea of this, three well-known accidents are described—the Ariane 5 flight 501, Mariner-1 disasters, and the Therac-25 accidents. In each case it is stressed what went wrong, what were the biggest mistakes, and how a careful understanding of fault models and system models would have helped highlighting the path to avoid catastrophic failures that cost considerable amounts of money and even the lives of innocent people.

The other important objective of this chapter is introducing the core subject of this book: Software fault-tolerance situated at the level of the application layer. First of all, it is explained why targeting (also) the application layer is not an open option but a mandatory design choice for effective fault-tolerant software engineering. Secondly, given the peculiarities of the application layer, three properties to measure the quality of the methods to achieve fault-tolerant application software are introduced:

1. Separation of design concerns, that is, how good the method is in keeping the functional aspects and the fault-tolerance aspects separated from each other.
2. Syntactical adequacy, namely how versatile the employed method is in including the wider spectrum of fault-tolerance strategies.
3. Adaptability: How good the employed fault-tolerance method is in dealing with the inevitable changes characterizing the system and its run-time environment, including the dynamics of faults that manifest themselves at service time.

Finally, this chapter also defines a few fundamental fault-tolerance services, namely watchdog timers, exception handling, transactions, and checkpointing-and-rollback.

WHAT IS A FAULT-TOLERANT PROGRAM?

So what makes a program fault-tolerant? In order to answer this key question, let us further detail what a service is: In the following a service is considered as a set of manifestations of external events that, if compliant to what agreed upon in a formal specification, can be considered by a watcher as being “correct”. This said, a program can be defined as a physical entity, stored for instance as voltage values in a set of memory cells, which is supposed to drive the production of a service. One of the main goals of software engineering is being able to set up of a robust link (in mathematical terms, a homomorphism) between a service’s high-level specification and a low-level computer design (the program).
Related Content

The Efficiency of Interactive Differential Evolution in Creation of Sound Contents: In Comparison with Interactive Genetic Algorithm

Developing Local Association Network Based IoT Solutions for Body Parts Tagging and Tracking
ZongWei Luo, Martin Lai, Mary Cheung, ShuiHua Han, Tianle Zhang, Zhongjun Luo, James Ting, Patrick Wong, Sam Chan, Kwok So and George Tipoe (2010). *International Journal of Systems and Service-Oriented Engineering* (pp. 42-64).
[www.igi-global.com/article/developing-local-association-network-based/48518?camid=4v1a](www.igi-global.com/article/developing-local-association-network-based/48518?camid=4v1a)
Planning and Managing the Human Factors for the Adoption and Diffusion of Object-Oriented Software Development Processes
www.igi-global.com/chapter/planning-managing-human-factors-adoption/29500?camid=4v1a

Combining Indexing Units for Arabic Information Retrieval
www.igi-global.com/article/combining-indexing-units-for-arabic-information-retrieval/166540?camid=4v1a