A UML Framework for Safety Mechanisms
Based on IEC 61508

Shourong Lu and Wolfgang A. Halang

shourong.lu@fernuni-hagen.de

Abstract:
A safety mechanisms based on safety standards IEC 61508 for developing safety-critical systems is presented. The safety mechanisms are constructed by a fault-tolerant structure associated with SIL and an safety architecture based on concepts taken from the real-time programming language PEARL, which can be incorporated into models of safety-related embedded real-time control systems.

1 Introduction

Architecture is a crucial element in the life-cycle of safety-critical systems, as indicated by the possibility to provide well-known safety mechanisms such as forms of design diversity (e.g., N-version programming) [6]. Good design of software architectures can help to meet SIL-specific demands, but also requirements with respect to performance, reliability, portability. For these reasons, and to cope with the complexity and safety-related requirements of embedded real-time systems, it may be beneficial to combine in their development process well-established fault-tolerant techniques and SIL-related programming language structures with modern techniques of object-orientation, which could enable designers to both design abstract solutions based on the systems requirements, and implementation solutions based on the semantics of the SIL-related programming language. A good match appears to be possible on the basis of the semi-formal modeling notation Unified Modeling Language (UML) [2].

2 Fundamentals of IEC 61508 for Safety-Related Applications

A safety-related system should fulfill the requirements of a certain SIL. For each SIL, the Appendices A and B of the IEC 61508[5] standard’s Part 3 state a number of highly recommended development practices, techniques and measures. In Fig. 1, we take a SIL requirement as starting point. The supports will include safety techniques used in the analysis and design phases. Some practices can be translated into safety patterns. The supports will be implemented within an executive pattern[3]. We can indicate patterns by
using UML notations.

Figure 1: SIL with executive patterns

3 A UML Framework for Safety Mechanisms

Here, we are aiming to build a set of UML extensions that support architectures to be dependable and to meet relevant safety criteria. The leading idea is to design systems in such a way that component faults do not result in system failures. This UML framework is shown in Fig. 2. It includes standard UML defined by the Object Management Group (OMG), a UML profile for fault-tolerance techniques and a safety architecture oriented at PEARL concepts.

Figure 2: Framework for safety mechanisms; Basic structures of NVP and RB

3.1 Defining UML profile for a Fault-Tolerance Techniques

Typical approaches to fault-tolerant programming are N-version programming and Recovery blocks as shown in Fig. ?? The fault-tolerance techniques differ in their respective architectures, but commonly deal with Fault-detection, Groups and Replication styles concepts. Conforming to the structure of fault-tolerance techniques, a mechanism for fault-tolerance structure is designed as shown in Fig. 3.

In UML, the metamodel of ReplicationManager, RecoveryManager and FaultManager are built as shown in Fig. 3. It presents the core-concept, and describes basic functions of fault-tolerance mechanism. The functions of the mechanism are:
Replication management responsible for the creation or removal of objects, and for modifying fault-tolerance properties. Fault management concerning the detection of object failures, the creation and notification of fault reports, as well as the analysis of the latter. Recovery management performing logging and invoking recovery mechanisms in order to find out where a failure happened and recover to a correct and consistent state.

3.2 A Safety Architecture based on PEARL Concepts

“Process and Experiment Automation Realtime Language” (PEARL) [1] is one of the very few genuine high-level real-time languages, which has been enhanced towards distributed systems and object-oriented design of applications having to meet severe safety requirements [4]. In the PEARL methodology, an executable program is a collection of modules, each of which is composed of a set of tasks responding to events. Tasks represent the (computing) processes of a running system. A safety-oriented architecture oriented at PEARL concepts is defined in Fig. 4, which illustrates the structure of an application architecture enhanced to design dependable systems. The element EventDetection in Fig. 4 is defined to provide fail-safe behaviour. The various architectural elements have the following semantics.

Configuration Manager (CM) controls the execution path of a system. The class Event defines the functions that dispatch an event to its destination task. Configuration classes are described in terms of modules and of input and output ports. Module consists of a collection of tasks, that together provide a desired functionality. Task is the execution of a procedure within a given time-frame, i.e., it has a trigger condition and a response time. Transitions are directed relationships between a source and a target state. Class EventDetection is used to detect fault occurrences. With the help of the CM it checks for
such fault events as depicted in the sequence diagram, Fig. 4.

4 Conclusion

The UML framework for safety mechanisms can provide facilities to capture safety requirements of safety-critical applications, as well as dependability structures which deal with safe elements and desired measures. The benefit of using UML in modeling safety mechanisms is that UML provides numerous diagrammatic techniques to comfortably describe process and system models of any kind. Thus, for each aspect to be modeled the most expressive technique can be selected by the user.

References


