$2 + 10 \succ 1 + 50$!

Hans Hansson, Christer Norström, and Sasikumar Punnekkat

Mälardalen Real-Time Research Centre
Department of Computer Engineering
Mälardalen University, Västerås, SWEDEN
han@idt.mdh.se, cen@mdh.se, spt@idt.mdh.se
WWW home page: http://www.mrtc.mdh.se

Abstract. In traditional design of computer based systems some effort, say 1, is spent on the early modeling phases, and some very high effort, say 50, is spent on the later implementation and testing phases. It is the conjecture of this paper that the total effort can be substantially reduced if an increased effort, say 2, is spent on the early modeling phases. Such a shift in focus of efforts will also greatly improve the overall effects (both quality and cost-wise) of the systems developed, thereby leading to a better (denoted by "$\succ$") design process. In this paper, we specifically consider the design of safety-critical distributed real-time systems.

1 Introduction

Designing safety-critical real-time systems involves assessment of functionality, timing and reliability of the designed system. Though several design methods have been proposed in literature (such as HRT-HOOD, DARTS, UPPAAL, UML-RT), none of them have been able to gain widespread acceptance due to the range and magnitude of the issues involved and probably due the restricted focus of these methods.

In Figure 1 we present a generic design model for the development of safety critical distributed real-time systems.

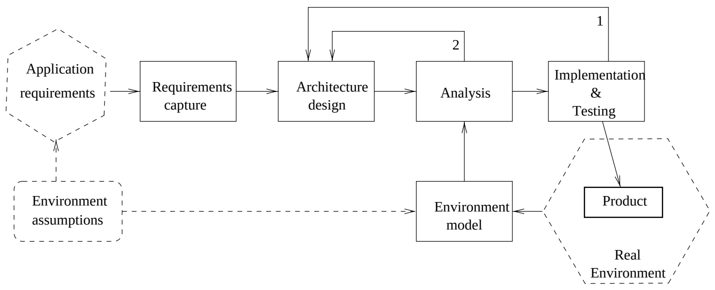

Fig. 1. A generic real-time design model

The architecture design is the highest abstraction level for the design and construction of the system. Here the system is partitioned into components, processes for realisation of them are identified, and boundaries for desired quality levels are set. For real-time systems, timing budgets are typically allocated to individual components at this stage. The analysis part in the design process contains both functional analysis (such as, temporal behaviour, reliability modelling, safety, performance) as well as non-functional analysis (such as, testability, maintainability, portability, cost, extensibility). To be able to make these analyses, the architecture has to be described by a language that provides a precise syntax and semantics. Such a language should define the computational model with possible extensions for hierarchical and functional decompositions.

Experiences from an industrial co-operation [1] have further convinced us of the benefits of performing architecture analysis on temporal requirements, communication and synchronisation. Based on these insights we will be focusing on architecture analysis rather than analysis of the implementation. It is apparent that such a shift in focus from the implementation & testing phase to the architecture design and analysis phases, by adding more resources and efforts in these earlier phases, is absolutely necessary to detect many critical issues before they manifest in the product and necessitate a costly product re-design. In terms of figure 1, this amounts to iterating more on the inner loop (marked 2) rather than on the outer loop (marked 1) to improve the quality at a lower cost.

Using such an approach, one of the major issues, i.e., timing compliance of the system was achieved in our project at Volvo [1] by applying a time-budgeting and negotiations strategy for individual tasks. We now present briefly two other major issues, viz. fault modelling and testability, representing a functional and non-functional issue, respectively. An accurate fault modelling and analysis will assist the designer in incorporating sufficient fault-tolerance capabilities into the system, whereas testability analysis can greatly reduce the final testing efforts. It should be noted that, both these issues are addressed in conjunction with their effects on the temporal requirements and properties.

2 Fault Modelling and Analysis

Though there has been sizable amount of research efforts in both the fault tolerance and the real-time realms, these two fields have been more or less treading along parallel paths. These two research domains are all the more relevant in the case of safety-critical systems and their mutual dependencies and interactions need to be analysed for achieving predictable performance. There are very few studies in literature, aimed at bridging the gap between these two areas and many issues remain open that need to be further investigated. One such important issue is the effect of faults on schedulability analysis and on the timing guarantees provided.

The major stumbling block in having an intergrated approach is the orthogonal nature of the two factors, viz., the stochastic nature of faults and the deterministic requirements on schedulability analysis. This calls for development of more realistic fault models which capture the nuances of the environment as well as methods for incorporating such models into the timing analysis with ease. In applications such as automobiles, the systems are often subjected to high degrees of Electro Magnetic Interference (EMI). The common causes for such interferences include cellular phones and other radio equipments inside the vehicle and electrical devices like switches and relays as well as as radars and radio transmissions from external sources and lightning in the environment. These interferences may cause errors in the transmitted data.

In this context we have recently [3] developed a model for calculating worst-case latencies of messages on the Controller Area Network (CAN) under errors. CAN is a predictable communication network widely used in the automotive and automation industries. The basic CAN analysis assumes an error free communication bus, which is not always true. To reduce the risk due to errors, CAN designers have provided elaborate error checking and confinement features, which identify the errors and retransmit the affected messages, thus increasing the message latencies and potentially leading to timing violations.

Tindell and Burns [2] have proposed a model for calculating worst-case latencies of CAN messages under errors. They define an error overhead function $E(t)$, as the maximum time required for error signaling and recovery in any time interval of length t. Their model is relatively simplistic and assumes an initial error-burst followed by sporadic error occurrences (i.e., errors separated by a known minimum time). Our new fault model [3] is more general, in that it

– models intervals of interference as periods in which the bus is not available
– allows more general patterns of interferences to be specified and from that description derive the effect on message transmissions
– allows the combined effects of multiple sources of interference to be modeled.
– considers the potential delay induced by the interference durations

With this fault model it is possible to build parameterised models of different types of interferences originating from different sources. Using these models, realistic worst-case scenarios can be characterised and analysed. We believe that this kind of analysis will be a step towards future design of adaptive scheduling strategies which takes in to account the error occurrences and decides on-line issues such as graceful degradation and choosing different policies for different classes of messages.

3 Testability Analysis

A large part of the effort, time and cost in developing safety-critical real-time (and most other) systems is related to testing. Consequently, one of the most important non-functional quality attributes of a design is its testability, i.e. the effort required to obtain a specific coverage in the testing process. High testability means that relatively few tests have to be exercised. The design with highest testability may however not be the preferred one, since testability typically is in conflict with other desired qualities, such as performance and maintainability.

Using testability measures in choosing between alternative designs that are similar in other respects is however highly desirable, and sacrificing other qualities for increased testability may be a good compromise in many situations.

An intuitive metric for the testability of a system is its number of distinguishable computations. For a sequential program this is proportional to the number of program paths. For concurrent and distributed systems we must additionally consider the possible interleavings of the program executions (the tasks). Clearly, by limiting the freedom in scheduling and by making synchronization between distributed nodes tighter, we can substantially reduce the number of interleavings, thus increasing testability. Testability is further increased if the variations (jitter) in release and execution times of individual tasks can be reduced.

In [4], we introduce a method for identifying the set of task interleavings of a distributed real-time system with a task set having recurring release patterns. We propose a testing strategy which essentially amounts to regarding each of the identified interleavings as a sequential program, and then use sequential techniques for testing it. Due to the large number of interleavings, this in general is a formidable task. We are however convinced that for a sufficiently large class of safety-critical real-time systems this approach is both feasible and desirable.

4 Conclusion and future challenges

In this paper, we have described some important issues in the design of safety-critical distributed real-time systems. We emphasize the potential gain of shifting the focus from implementation & testing phase to the architectural design phase, by obtaining a high effects-efforts ratio. In this context, we also highlighted two of our latest research contributions.

The vision and objective of current research in the Systems Design Laboratory at Mälardalen Real-Time research Centre is to provide engineers with scientific methods and tools for designing safety-critical real-time systems, such that the state-of-art and practice for developing such systems is advanced to a mature engineering discipline. This amounts to developing, adopting and applying theory with industrial applications in mind, as well as designing appropriate engineering tools and methods.

References

1. Christer Norström, Kristian Sandström, and Jukka Mäki-Turja: Experiences and findings from the usage of real-time technology in an industrial project, *MRTC-Technical report*, January 2000.
2. Ken W. Tindell, Alan Burns, and Andy J. Wellings: Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, 3(8), 1995.
3. Sasikumar Punnekkat, Hans Hansson, and Christer Norström: Response time analysis of CAN message sets under errors, *MRTC-Technical report*, December 1999.
4. Henrik Thane and Hans Hansson: Towards Systematic Testing of Distributed Real-Time Systems, *20th IEEE Real-Time Systems Symposium*, Phoenix, December 1999.