

# Some Unresolved Problems in Real-Time Design

Bran Selic

*IBM Rational Software*

*bselic@ca.ibm.com*

## Abstract

*Real-time systems are among the most complex software systems to design and construct. This short position paper outlines some of the most pressing research problems that need to be addressed. They are all related to the concept of "temporal cost" of design choices and the ability to easily and directly track those costs during design.*

## 1. Temporal Cost and Its Tracking

In modern automobile design the cost of each part is a fundamental design concern, on par with technological concerns such as durability, strength, or ease of manufacture. Quite literally, the price of every part is known down to the individual cent level and is a primary criterion in devising and in selecting design alternatives. This is because the cost of each part is multiplied by the total number of vehicles manufactured. A similar situation can be found in the design of transport aircraft, where each ounce of weight adds to the cumulative operating costs over the lifetime of the aircraft.

Of course, it is not at all surprising that in the process of design engineers maintain such tight and direct scrutiny of factors to which their products are most sensitive. In fact, it would be remarkable if it were otherwise. Unfortunately, when it comes to real-time software, which is fundamentally sensitive to computation time, the *temporal cost* of individual design decisions is only vaguely understood at best when the design choices are made. Preliminary "guesstimates", made even by the most experienced designers, are quite often off by orders of magnitude. Typically, the actual cost is known only after the system is fully constructed and tested – that is, when the cost of changing the design is greatest.

Ideally, the precise temporal cost of every design choice would be known in advance so that its impact on the overall time budget is well understood. Needless to say, we are nowhere near this ideal. One of the primary reasons behind this is that we generally have a very poor understanding of the relationship between software and its underlying *platform*. By "platform" I mean the full complement of support software and hardware that underlies application software. Clearly, the speed of the processor, busses, and memory can play significant roles in this as well as the execution overhead and nature of

such operating system services as multi-tasking, inter-process communication, scheduling, etc. A second key factor that is equally poorly understood is the effects that compilers have on performance. How well and how predictably a particular compiler translates a source program into executable code, the kinds of optimizations it uses, its ability to exploit caching in the hardware, and so on, can all have fundamental impact on temporal cost. Last but not least, there is the complex effect due to resource contention, that is, the consequences of multiple software applications competing for and sharing the same platform resources.

Traditionally, application software was designed separately from the underlying platforms or compilers. This yields important advantages since it enables us to separate complex concerns and allows a degree of technology independence such that the same software can be ported to different platforms with little or no change. However, taking this too far means separating software design from any consideration of its raw material. After all, it is the software in conjunction with hardware that actually gets things done. Engineers in more traditional disciplines are very much aware of the crucial influence that the materials they are using can have on design [1] [2]. The case of real-time software is quite similar; designing real-time software with complete disregard of platform and technology capabilities is not only bad engineering but also fundamentally irresponsible.

## 2. Related Research Issues

There is, therefore, a set of fundamental design issues that still remain to be solved in the domain of real-time software construction. In particular, I would like to point to the following:

- *Characterization of (real-time) platforms.* At present, there is no standardized conceptual framework for talking about the key real-time relevant characteristics of either software or hardware platforms. While some common characteristics have emerged, such as context switching time, instruction execution times, etc. none of them are precisely defined and are generally not measured in any standard way. If a standardized conceptual framework could be codified it could then be incorporated into tools enabling designers to get relatively quick

assessments of the temporal costs of key design choices.

- *Understanding the relationship between software and its platform.* This refers to the way in which software is deployed on its underlying platform and the impact that this has on the temporal characteristics of software. For example, what are the ways in which application-level concurrency maps to the underlying concurrency support mechanisms of the platform (schedulers, threads, etc.)?
- *Identifying compiler requirements for real-time software.* Numerous commercial real-time oriented compilers are available in the market, but most of them are based on ad hoc techniques. There is no systematic or standardized theoretical framework that would allow designers to calculate in direct and obvious ways the impact that a compiler might have on the temporal characteristics of their software. Such a framework could and should be based in part on the previously mentioned framework for characterizing platforms.
- *Further evolution of contention analysis techniques.* At present, such techniques have been developed primarily in two domains: schedulability analysis and performance analysis based on queueing

theory. These are important and useful developments. However, we need to develop new even more precise techniques in addition to evolving the current ones.

### 3. Conclusion

While these are by no means the only problems facing real-time software, I feel that these are not only fundamental issues that need to be addressed immediately but also that they are well within our grasp. None of them require quantum leaps in the state of technology but can be based on a consolidation and systematic organization of current knowledge.

### 4. References

- [1] Levy, M. and Salvadori, M., *Why Buildings Fall Down*, W. W. Norton & Co., New York, 2002.
- [2] Petkoski, H., *To Engineer is Human – The Role of Failure in Successful Design*, Vintage Books, New York, 1992.