From Bold Idea to Product—a Case Study

Wolfgang Pree
Department of Computer Science, University of Salzburg, Austria
firstname dot lastname at cs.uni-salzburg.at

ABSTRACT
In state-of-the-art methodologies the critical timing properties of embedded software systems are accidental consequences of an implementation, rather than specified parts of the design. The Giotto project, lead by Thomas Henzinger at the University of California, Berkeley, delivered the formally sound concept of Logical Execution Time (LET) as abstraction for explicitly specifying the timing behavior of embedded software. The presentation outlines the various hurdles along the way from the initial idea in 2000 until the shipment of products that incorporate model-based design relying on LET in 2005. For example, it was tempting to overcome certain limitations of the programming model at the price of loosing crucial embedded software properties such as determinism.

1. LOGICAL EXECUTION TIME (LET)
Traditional development of software for embedded systems is highly platform specific. The hardware costs are reduced to a minimum whereas high development costs are considered acceptable in case of large quantities of devices being sold. However, with more powerful processors even in the low cost range, we observe a shift of functionality from hardware to software and in general more ambitious requirements. A luxury car, for example, comprises about 80 electronic control units interconnected by multiple buses. In order to cope with the increased complexity of the resulting software, a more platform independent “high-level” programming style becomes mandatory. In case of real-time software, this applies not only to functional aspects but also to the temporal behavior of the software. Dealing with time, however, is not covered appropriately by any of the existing component models for high-level languages.

A particularly promising approach towards a high-level component model for real time systems has been laid out in the Giotto project [1] by introduction of logical execution time (LET), which abstracts from the physical execution time on a particular platform and thereby abstracts from both the underlying execution platform and the communication topology. Thus, it becomes possible to change the underlying platform and even to distribute components between different nodes without affecting the overall system behavior. Giotto, however, is primarily an abstract mathematical concept and there exist only simple prototype implementations, which show some of the potential of LET. LET means that the observable temporal behavior of a task is independent from its physical execution. It is only assumed that physical task execution is fast enough to fit somewhere within the logical start and end points. Figure 1 illustrates the relationship between logical and physical task execution.

LET introduces a delay for observable outputs, which might be considered a disadvantage. On the other hand, however, LET provides the cornerstone to deterministic behavior, platform abstraction and well-defined interaction semantics between parallel activities. It is always defined which value is in use at which time instant and there are no race conditions or priority inversions involved.

2. PRODUCT REQUIREMENTS
The inputs of a task are read at the release event and the newly calculated outputs are available at the terminate event. Between these, the outputs have the value of the previous execution. Thus, the LET abstraction maximizes the delay between sensor read and actuator update. At the same time Giotto minimizes the delay between the actuator update and the next sensor read because both are logical zero time activities.

Both aspects are in conflict with control theory: The software implementation of digital controllers should come close to the ideal situation of having indefinitely fast computing resources. The desired 10 to 1 relation between period and execution time cannot be achieved with LET. In addition, in the Giotto programming model there are no provisions for so-called advance calculation. This technique can be used in the common case of solving first order difference equations in a digital controller.

The presentation outlines the way from the initial LET concept to its extension and refinement until LET-based products could be shipped. In addition to solving the control theory issues, a syntax had to be defined and a component model had to be introduced. All that was accomplished in the MoDECS2 project.


The MoDECS project (www.MoDECS.cc) is supported by the FIT-IT Embedded Systems grant 807144 (www.fit-it.at).