A combination of system software techniques aimed at raising
the run-time safety of complex mechatronic applications

Author(s):
Brega, Roberto

Publication Date:
2002

Permanent Link:
https://doi.org/10.3929/ethz-a-004370826

Rights / License:
In Copyright - Non-Commercial Use Permitted
A Combination of System Software Techniques Aimed at Raising the Run-Time Safety of Complex Mechatronic Applications

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZÜRICH

for the degree of
Doctor of Technical Sciences

presented by
Roberto Brega
Dipl. Informatik-Ing. ETH
born April 15, 1971
citizen of Italy
roberto.brega@acm.org

accepted on the recommendation of
Prof. Dr. G. Schweitzer, examiner
Prof. Dr. M. Norrie, co-examiner
Dr. S. J. Vestli, co-examiner

Zürich, 2002
The increasing presence of embedded software controlling critical systems—i.e., systems that are critical to missions, to the environment, to human lives or society—makes their reliability a prime concern. This Thesis presents system software techniques aimed at providing a higher degree of safety in complex embedded systems by raising the reliability of their software. These techniques address issues that range from safe composition of software modules to the choice of the programming language, while focusing on three crucial themes: The management of time as a system-wide resource; The handling of memory; The estimation of the worst-case execution time for real-time tasks.

The management of time-constrained processes is performed by the task scheduler: A system primitive that assigns processing time to the concurrent tasks according to some priority mechanism. We describe a scheduling algorithm—called Earliest Deadline First (EDF)—that has not yet seen widespread adoption despite its unique features, which make it very suitable for safety-critical systems. We illustrate the benefits it provides and suggest a possible implementation that has been aggressively optimized for the PowerPC processor architecture.

The handling of the available memory is a further critical component of a computing system, as the incorrect use of memory can undermine its run-time safety. Moreover, memory errors represent common programming problems, which are far too frequent, awkward to track and to debug. In this case, we demonstrate how the joint effort of a safe programming language, the harnessing of the underlying processor architecture and a run-time garbage collector can help in minimizing—if not completely eliminating—the occurrence of memory-related errors, without compromising on run-time performance.

A safe real-time system requires, for each task, the Worst-Case Execution Time (WCET), which is imperative for the scheduler's admission tests and subsequently limits a task's execution time during operation. While only modern microprocessors can provide the necessary computation cycles, the combination of pre-emption, dynamic loading of compilation units and state-of-
the-art microprocessors creates unique challenges when estimating the WCET. To obtain a realistic estimate of a task’s execution time we performed static analysis of the source code coupled with information concerning the task’s run-time behaviour. The implemented predictor is able to compute a good estimation of the WCET even for tasks that contain a lot of dynamic cache usage.

These software mechanisms have been implemented in a prototypical operating system called XO/2. The XO/2 system has demonstrated its advantages as it has been used as the primary software building block for many research projects and commercial products. Their high degree of dependability, reliability and safety let us argue that a careful choice of the computer science paradigms not only eases the development of the application software, while reducing the time needed to write it, but also fosters innovation, since the implementor, relieved from the computer science issues, can better focus on the problem to be solved.