Real-Time Software Development Requires Rigid Constraints

Development of real-time software differs from the development of non-real-time software because execution time must be considered. This brings the operating system and the underlying hardware processor architecture (memory, processor speed, bandwidth, etc.) into play. Also, in addition to new classes of defects related to stringent timing requirements, defect management in general becomes more critical because real-time applications are often life- or mission-critical.

Most real-time systems require a proprietary real-time operating system to run on, and therefore the software is typically not transferable to other platforms. Because embedded systems frequently are airborne (or spaceborne) platforms, the memory used for the software has weight and power constraints. Bandwidth issues include the amount of memory available or the number of memory cycles available. Defects can cost more than an inconvenience or money; depending on the system, defects can cost lives. For example, if the real-time software is used in a military aircraft, all data on an approaching missile is critical and time-sensitive. Only the most current data will suffice – and older data (even if only a few seconds old) is useless. However, if the real-time software is used in weather radar, and the current screen update is not available, then the last update will probably be sufficient if it is recent enough.

With these additional real-time software requirements come additional testing and maintenance requirements. The software is difficult to sustain since in many cases, changes to the software require that all of the code be retested.

Our first article is an overview that discusses many topics that must be considered while developing real-time software. Dennis Ludwig's article, *An Introduction to Real-Time Programming*, uses terms that most software developers can understand as they try to learn about the different world of real-time system development.

In *The Ravenscar Profile for Real-Time and High Integrity Systems*, Brian Dobbing and Alan Burns discuss this model for building safe and reliable real-time systems. The discussion includes the motivation behind the creation of the Ravenscar Profile, a definition of its specification, the ways in which it may be used with verification tools to produce evidence of dependability, and even a short example. Although the Ravenscar Profile is specified in Ada terms, it is based on a language-independent set of building blocks that are suitable for constructing typical real-time systems.

As discussed earlier, real-time software is often also safety-critical. Andy German shares his experiences from developing safety-critical, real-time systems in *Software Static Code Analysis Lessons Learned*. This article also presents a unique perspective for many readers since it comes from the United Kingdom (UK) and shares UK standards.

Defense Order (DO)-178B certification is a standard for the development of aviation software; however, much is expected under this type of grueling development and verification process. In *Decision Point: Will Using a COTS Component Help or Hinder Your DO-178B Certification Effort?* author Timothy J. Budden describes how the demands of DO-178B certification can be achieved with commercial off-the-shelf (COTS) modules if the vendor is a willing partner. I am hoping this perspective on the ability to use COTS software will help many readers.

In this month's supporting articles, Dr. John A. Hamilton Jr., Col. Kevin J. Greaney, and Gordon Evans discuss a process to evaluate potential vulnerabilities in shared simulation software in *Defining a Process for Simulation Software Vulnerability Assessments*. In addition, Dan W. Christenson and Lynn Silver discuss issues for tying software and hardware together in *Developing a Stable Architecture to Interface Aircraft to Commercial PCs*. Finally in our online article, Walt Lipke writes about evaluating whether or not a project will be on time and within budget in *The Probability of Success*.

Development of real-time software is often more tricky than development of other software, and the resulting software's performance is often more critical than other software. If you are currently developing real-time software, I hope you find new and useful information in this month's issue of CrossTalk. If you are in a position where you need to learn about real-time software development, I hope these articles will start you down the right path.

Elizabeth Starrett

Elizabeth Starrett
Associate Publisher