High-Assurance Distributed, Adaptive Software for Dynamic Systems

Kurt Rohloff, Joseph Loyall, Partha Pal, Richard Schantz
BBN Technologies
10 Moulton St., Cambridge, MA 02138, USA
{krohloff, jloyall, ppal, schantz}@bbn.com

Exhaustive testing, documentation, code review, and formal methods have been the main approaches for software certification in high confidence cyber-physical systems. Although these methods have been sufficient in the past, continued reliance on these methods is no longer economically feasible for increasingly complex modern, distributed, dynamic systems due to inherent problems of state-explosions. Examples of such distributed high-assurance systems include DoD systems (including secure, timely command, control and information sharing systems and for military logistics), systems for manufacturing and process control (for industries whose safety is of critical national importance such as transportation, chemical, oil and natural gas), and medical systems among others. We need to develop innovative, economically feasible means to certify distributed dynamic control software for cyber-physical systems so that when these systems are deployed, harmful unpredictable emergent behavior does not manifest itself.

1 Ongoing work

We have been developing distributed, real-time embedded (DRE) systems for the DARPA Adaptive Reflective Middleware System (ARMS) program that require predictable, controlled real-time behavior. As part of our efforts in this program, we have developed methods to measure and evaluate the quality, or utility, of DRE system performance in the context of the dynamic, unpredictable environments in which they operate. These developments run hand-in-hand with the challenge of providing adequate control of the system to effect behavior that will allow the controlled system to be certified as exhibiting correct behavior. As part of the ARMS program, we have been developing utility-based measures of system quality for assessing correct behavior of a system and for driving feedback control at runtime [3]. We have been using utility measures as feedback signals to a control system that manages the dynamic allocation of resources in the system and as part of an evidence-based certification process for dynamic real-time systems [2].

Related to our utility measurement work is our efforts to develop scalable, adaptive and time-bounded general approaches to assure reliability and safety for real-time Node-Failure Detection (NFD) for large-scale, high load networks comprised of Commercial Off-The-Shelf (COTS) hardware and software [1]. We have been developing generalizable, multi-layer, dynamically adaptive monitoring approaches to NFD where a small, designated subset of the nodes are communicated information about node failures. We have been able to show through experimentation that in real-world deployments the variations in

1530-2059/07 $25.00 © 2007 IEEE
DOI 10.1109/HASE.2007.17

the scheduling of messages in NFD systems can cause large variations in the false-positive notification behavior of our NFD subsystem. We have been developing a per-node adaptive enhancement of NFD subsystems that adapts to provide run-time assurance of low false-alarm rates while providing finite node-failure detection delays.

2 Paths Forward

Based on new, experimental approaches to distributed systems utility functions and behavior evaluation, provisioning the resource management functionality as middleware infrastructure enables us to better and more certifiably control the resources provided through these interfaces. As a first step along this avenue for development, we can use clearly partitioning resource allocation mechanisms, such as CPU reservations and network reservations. These enable the dynamic resource manager to provide a clear set of resources to each component (and component interaction), simplifying the reasoning and certification.

A more general problem involves removing these constraint mechanism interfaces in favor of policy-driven application control which would automatically regulate component interaction. A promising idea in this space is to use specialized, domain-specific, mutually composable sequential process languages such as a duration calculus to express the certified behaviors of individual components of the system. The union of shared expressions in these domain-specific sequential process languages would need to be expressive enough to specify the behavior of the overall system (with respect to both functional behavior and resource usage), but should be sufficiently easy to perform computable operations on to certify the behavior of the overall system. The languages could be used as control specifications to regulate the behaviors of the local components with respect to these desired behaviors.

An additional avenue to "on-the-fly" certification of highly reconfigurable systems would be an incremental online approach to system regulation that permits "certifiable" behavior to occur and facilitates system recovery if partial system failures cause the system to enter an uncertified operating mode. For this approach, based on our previously mentioned approach to experimentally evaluating behavior, a regulator in the system could be continually run to identify "acceptable" and "unacceptable" variations of the system's current configuration (based on some measure of utility). The regulator would attempt to prevent the system from entering to an "unacceptable" configuration and could be used as a kind of "fail-safe" governor for system behavior. If the system would ever exhibit "unacceptable" behavior due to partial system failure or the use of the system in an unproscribed manner, the governor would push the system to return into an acceptable configuration, or at least prevent the system from entering a less acceptable configuration. As this on-the-fly certification procedure progresses, we would get directed coverage of large numbers of static certified configurations over the areas of highest interest/utility for overall high-assurance operation. This certification process could run in the background when the software is deployed, and the software's configuration controller would only reconfigure to use configurations that have been designated as certified.

References

[1] M. Gillen, K. Rohloff, P. Mangwhani, and R. Schantz. Scalable, adaptive, time-bounded node failure detection. In *IEEE High Assurance Systems Engineering (HASE) Symposium*, 2007.

[2] K. Rohloff, Y. Gabay, J. Ye, and R. Schantz. Scalable, distributed, dynamic resource management for the ARMS distributed real-time embedded system. In *Proceedings of the International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS-2007)*, 2007.

[3] K. Rohloff, J. Loyall, and R. Schantz. Embedded systems and their application to control and certification. *ACM SIGBED Review*, 3(4), 2006.