The Grand Challenge of Embedded System Dependability

Philip Koopman
ECE Department
Carnegie Mellon University
Pittsburgh, PA, United States
koopman@cmu.edu

Abstract: Four significant challenges in embedded system dependability are: embedded-specific security approaches, unifying security with safety, dealing with composable emergent properties, and enabling domain experts to use advanced dependability techniques.

Embedded systems permeate our everyday lives, including applications as diverse as cars, consumer electronics, thermostats, and industrial process controls. We have a surprising amount of reliance upon these systems, and we take their dependability almost for granted. Given extreme cost constraints, tremendous deployment scales, and the wide range of application domains, it is amazing that things more or less work well today. But, as application complexity increases, more applications become safety critical, and more embedded systems are attached to the Internet, we cannot expect business as usual with design approaches to maintain the level of dependability we want and need from such systems.

In my opinion the biggest challenges facing embedded systems lie in the areas of creating more suitable security techniques, finding a more unified approach to safety+security, dealing with composable emergent properties, and deploying dependability techniques to small product development teams.

Deeply embedded system security has significantly different constraints and requirements than enterprise and personal computing security. Embedded control systems often have severe resource constraints, limited development budgets, and stringent real time performance requirements. But an even more pressing security problem in many embedded systems is that the effects of a malicious fault can cause physical damage to people and the environment. It is less difficult to reverse or adequately insure against most malicious financial transactions than it is to reverse the release of toxins into the environment or "roll back" a multi-vehicle collision. Additionally, most embedded systems to date have been designed with near-zero security once an attacker has access to the internal control network. IT-based techniques for addressing that situation are unlikely to suffice due to matters of cost, real time dynamics, and lack of complete physical isolation from attackers.

Inevitably, embedded system safety and security will have to merge into a unified discipline, or at least a tightly-coordinated set of sub-disciplines. It is questionable to build safety cases for most everyday systems upon a faulty presumption of perfect security. At the same time, security techniques will need to take into account the safety implications of vulnerabilities and system outages. One element of a safety and security unification strategy might be to look at security faults as an attack on the assumptions of the safety case (e.g., an attacker negates the random independence assumption of fault arrival rates).

Due to the limitations and realities of embedded system development, workable dependability approaches will likely include some notion of cost-effective resilience in the face of inevitable faults, as well as a way to balance the tension among the often conflicting goals of safety, security, performance, and reliability. The good news will be that there are opportunities to exploit domain characteristics such as physical process inertia in ways not practical in desktop and enterprise computing.

A long-standing problem has been increasing the composability of emergent system properties. It is desirable to have building blocks that can be composed arbitrarily without surprises, and by the same token have an ability to decompose a system architecture so that predictable building blocks can be identified in a way that minimizes cross-coupled quality attributes. Much progress has been made on this in the area of real time systems, but much remains to be done in other areas such as safety and security. An additional challenge will be ensuring the composability of massively deployed distributed systems so that, for example, a city full of smart thermostats doesn't display emergent aggregate behavior that takes down the power grid.

Finally, the serious challenges posed by creating a dependable system are made more difficult when the development teams are typically composed of a handful of domain experts who may have no formal computer training beyond an introductory programming course. The traditional way to deploy advanced knowledge is via synthesis and analysis tools, and this has been done with astonishing success in IC design. More recently, model-based design has been helping embedded system designers in some domains perform code synthesis from relatively high level system behavioral descriptions. But will be a long time before tools can provide us with push-button automation that addresses the myriad aspects of embedded system dependability.

Extended Version, June 30, 2011. Original shorter version appeared at a panel session for DSN 2011.