Embedded systems have been changing rapidly these days as the application areas for the systems get proliferated from industrial controls to home automation, communication equipment, consumer gadgets, medical devices, defense systems etc. The emerging features demanded in the applications are often implemented by software programs rather than hardware as a cost-effective approach to realize sophisticated functionality. To improve software development process and system dependability, this software centric development needs a system platform with robust execution environment and programming language support.

It is imperative to note that an effective software development and execution platform can be crucial to software productivity which has fallen behind that of process technology and hardware development. Following Moore’s law, the number of transistors in a chip gets doubled every 18 months, i.e. a productivity increase of 58% per year in process technology. With effective CAD tools and mature IP blocks, the design productivity for hardware systems increases at a pace of 21%. However, only an 8% increase of software development has been observed. The gap is certainly contributed by many factors, including the software complexity of large scale project and the robustness of execution environment.

An effective software development and execution platform for embedded systems should support software portability, predictability, and dependability. It should facilitate timely and concurrent operations, and have a bounded resource usage in application systems. In this talk, we will first present an approach to enable deterministic and schedulable operations in VMs (Virtual Machines). Given the popularity of JVM (Java Virtual Machine) and CLI (Common Language Infrastructure), VMs have become an attractive
environment for real-time embedded applications to lower production cost, enable fast time-to-market, assure high integrity etc. To demonstrate the approach, we will show that extended and unpredictable blocking delays can be eliminated in two VM components: garbage collection for automatic memory management; and persistence service for high availability and failure recovery.

The second subject to be discussed is application isolation and protection for safe operations. It is envisioned that seamless integration of embedded applications is required while keeping individual applications away from any interference within a shared architecture. The principles of temporal and spatial partitioning adopted in ARINC 653 for Integrated Modular Avionics will be illustrated and compared with the language safety approaches of JSR-121 and CLI.

The talk will also include an introduction of the runtime environment for model-driven development, i.e. a virtual machine supporting the model of computation defined by various modeling languages. Examples of the message-driven run-to-completion semantics of UML-RT (Rational Rose’s TargetRTS), the synchronous reactive model of computation in Simulink Real-time Workshop, and the actor model of nesC and galsC, will be illustrated along with several optimization techniques.