Model-driven Development
From Object-Oriented Design to Actor-Oriented Design

Edward A. Lee, *UC Berkeley*

Abstract:

Most current software engineering is deeply rooted in procedural abstractions. Objects in object-oriented design present interfaces consisting principally of methods with type signatures. A method represents a transfer of the locus of control. Much of the talk of "models" in software engineering is about the static structure of object-oriented designs. However, essential properties of real-time systems, embedded systems, and distributed systems-of-systems are poorly defined by such interfaces and by static structure. These say little about concurrency, temporal properties, and assumptions and guarantees in the face of dynamic invocation.

Actor-oriented design contrasts with (and complements) object-oriented design by emphasizing concurrency and communication between components. Components called actors execute and communicate with other actors. While interfaces in object-oriented design (methods, principally) mediate transfer of the locus of control, interfaces in actor-oriented design (which we call ports) mediate communication. But the communication is not assumed to involve a transfer of control.

By focusing on the actor-oriented architecture of systems, we can leverage structure that is poorly described and expressed in procedural abstractions. Managing concurrency, for instance, is notoriously difficult using threads, mutexes and semaphores, and yet even these are extensions of procedural abstractions. In actor-oriented abstractions, these low-level mechanisms do not even rise to consciousness, forming instead the "assembly-level" mechanisms used to deliver much more sophisticated models of computation. In this talk, I will outline the models of computation for actor-oriented design that look the most promising for embedded systems.