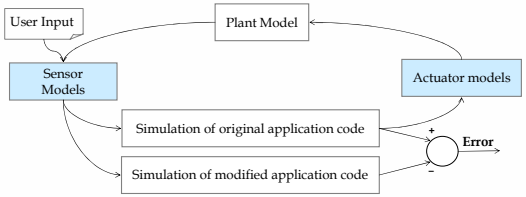
APES-LESS: Access Point Event Simulator for Legacy Embedded Software Systems

CHESS
Center for Hybrid & Embedded Software Systems
College of Engineering, UC Berkeley
http://chess.eecs.berkeley.edu/
Stefan Resmerita
Patricia Derler

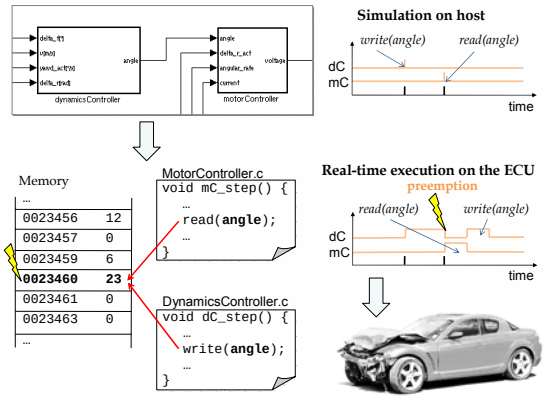
Objectives

- Fast simulation of software models mapped to platform models
- Achieve Software-In-the-Loop testing of closed-loop control systems, capturing the influences of execution times and preemptive scheduling
- Provide a validation environment for obtaining high-level models of legacy code



Scope

- Automotive applications with legacy C code executed on an OSEK operating system
- Multitasking, with fixed-priority preemptive scheduling
- Intertask communication by means of shared memory



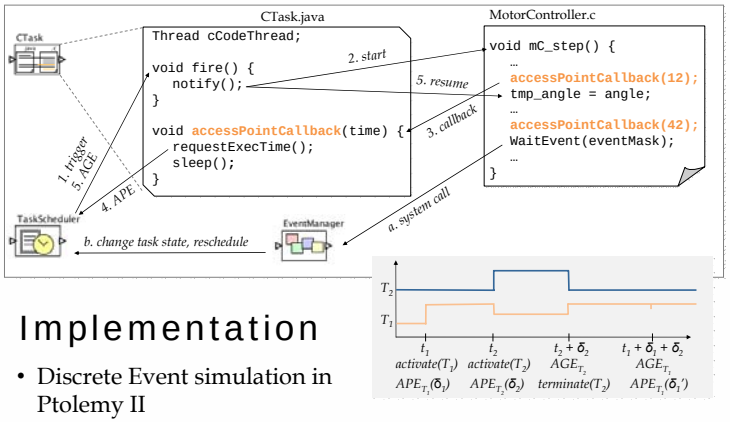
Approach

The proposed simulation framework is based on:

- The ability to catch all APEs during an execution
  - ✓ *Instrument the source code: insert a callback to the simulation engine at every access point*
  - ✓ *Generate an APE in the callback*
- The ability to stop/resume the execution of a task at any access point
  - ✓ *Execute each application task in a dedicated thread*
- The ability to determine the (platform-specific) execution time of the portion of code between any two consecutive access points of the same task.
  - ✓ *Leverage existing methods for estimation of execution times, e.g. WCET, BCET, statistically computed ET, measured ET*
  - ✓ *The execution time is passed as a parameter of the callback and becomes the APE's timestamp*
- Considering an APE as a request for CPU time
  - ✓ *Each APE is sent to the task scheduler, which generates an answer to it when the requested time is consumed. This is called an Access Granted Event (AGE)*
  - ✓ *Each application thread is paused in its callback, after issuing an APE, and is resumed upon receiving the corresponding AGE*

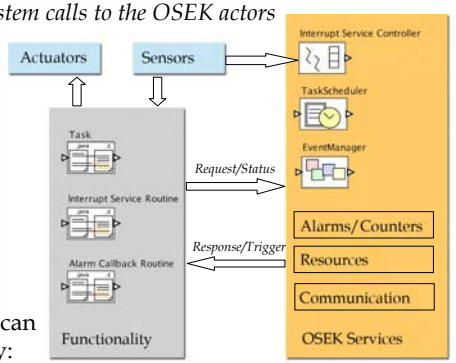
Access Point Event (APE)

- An *access point* is a line of source code with an I/O access or a system call
- In a run of the software, an *access point event* occurs whenever the code of an access point starts executing

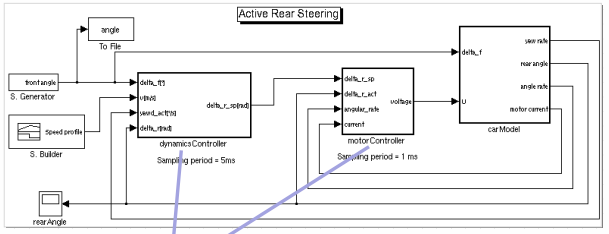


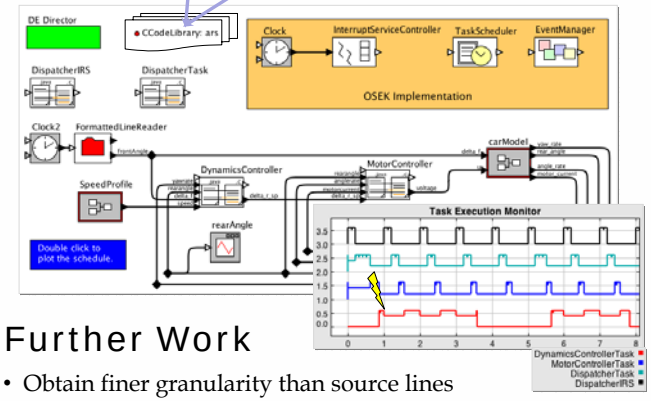
Implementation

- Discrete Event simulation in Ptolemy II
- OS services are provided by an OSEK implementation in Ptolemy
- Each C task has a corresponding actor in Ptolemy II
- The instrumented C code is augmented with wrappers to
  - ✓ *Direct each callback to the right task actor*
  - ✓ *Route the system calls to the OSEK actors*
- All the C application code is compiled for the host platform into one library
- Closed-loop simulation models in Ptolemy II can be achieved by:
  - ✓ *Using a software plant model regarded as a C task with zero execution time*
  - ✓ *Using a Ptolemy II plant model*



Example: Active Rear Steering





Further Work

- Obtain finer granularity than source lines
- Develop tools for automatic instrumentation of legacy code and automatic generation of the Ptolemy simulation model
- Adapt/use various methods of execution time estimation

Sponsors: TOYOTA, UNIVERSITÄT SALZBURG