Simulation as a Software Engineering Tool

Přemek Brada, Petr Grillinger
DCSE, University of West Bohemia
Pilsen, Czech Republic

(draft slides for ECBS 2004 Brno)

Motivation

- Software Engineering Best Practices
  - Spiral model: determine alternatives – *prototype/simulate* – select architecture – implement
  - Agile methods: textual specification – design – write test cases – *implement and test*
- FIT Project Experiences
  - Simulation model as precise specification
  - Parallel hw-sw development (chip in preparation)
  - Cost (access to) hw prototype

FIT Project: TTP/C Controller Verification

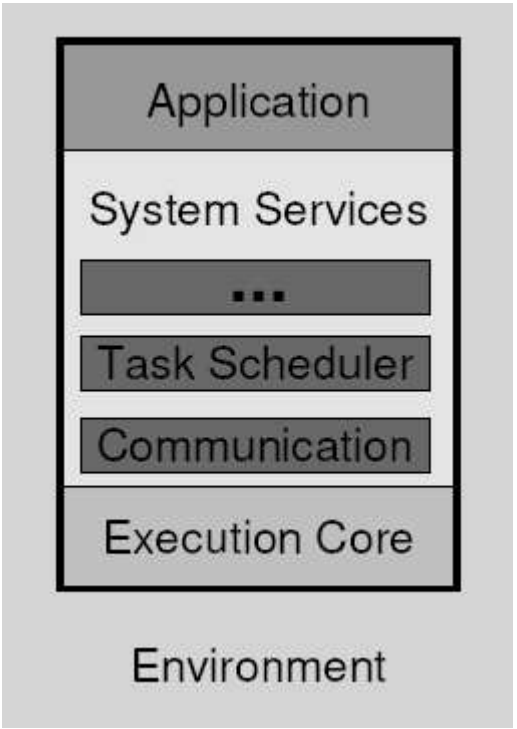
- Goal: implement specification, verify FT properties, cross-check other methods

- Sine wave application
  - basic functionality
  - method trial
- Brake by wire (BBW4)
  - realistic workload
  - car model by Volvo

- Tools: C-Sim, ANSI C

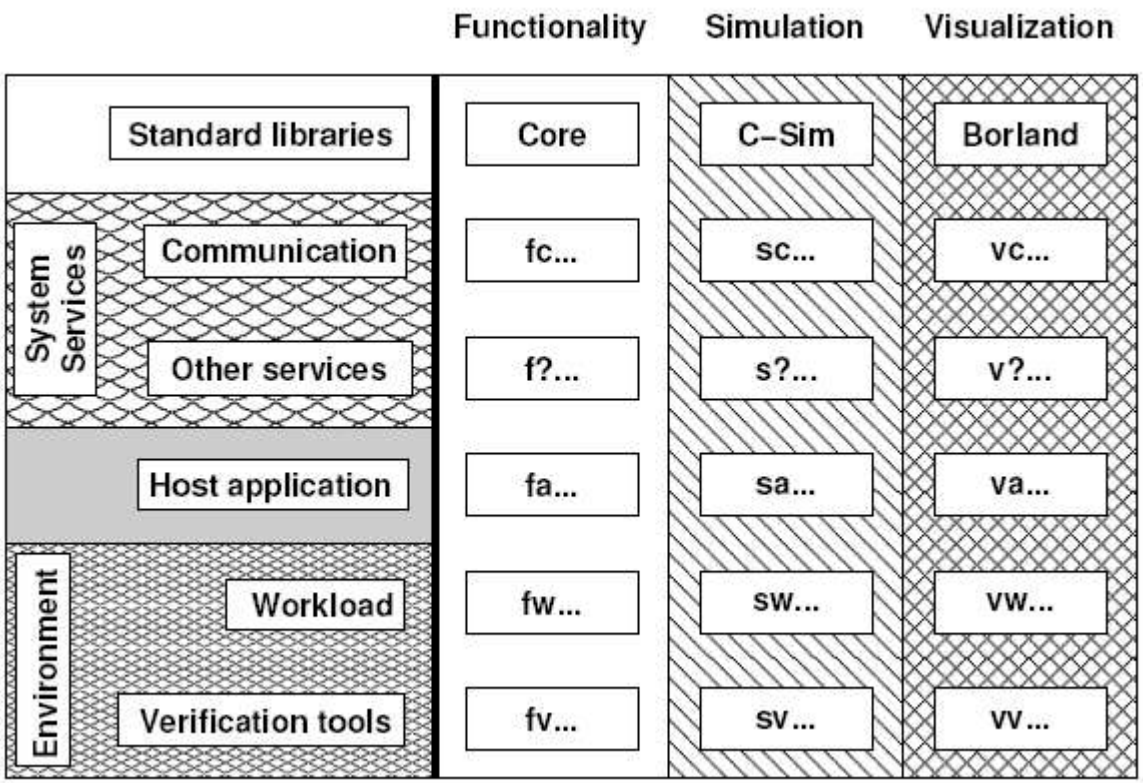
Embedded Software Model

- Reality: (A,S,E)
  - sw applications, APIs, hw controllers
- Simulation: (A', S', E')
  - Simulation models
  - Or in combination with reality
- Key issue: simulation-reality distance
  - Premise: C as hw app language, process-based app, cooperative tasks OS
  - Consequence: C-Sim model adequate



Model Implementation: 2D Modular Structure

- Separate functionality and simulation aspects



Development Process

- Ensure correct result
  - But thorough testing difficult & costly
  - Fault injection as one option for FT
- Enable fast development
  - In parallel with platform hardware
  - Cannot sacrifice correctness

- Approach
  - Take advantage of close-to-real code simulation

Process Phases

- Preparation
  - Create support modules for app development
  - Simulation mode
- Development
  - Results in working, verified application
- Transition
  - Final application version from the simulation model

Phase 1 – Preparation

- Steps
  - Create E' – simulation model of E
  - Obtain/Create S' – simulation model of S
  - Verify their correctness
- E', S' will serve as A' scaffolding

- Corectness: testing apps
  - Empty (A'_0,E'_0) – integration, S' operational
  - Exhaustive synthetic app, driver environment – exercise services, S' correct

Phase 2 – Development

- Steps
  - Design A' for E',S'
  - Test, debug A'
  - Evaluate with alternative S' (communication)
- Benefits
  - Comfort of development platform
  - Repeatable tests
  - Bug localisation possible
  - No need for hw/devices

Phase 3 – Transition

- Steps
  - Remove simulation code
  - Transfer to real platform (S,E')
  - Integrate into real environment
  - Perform hw verification
- Enabling factors
  - Close-to-code simulation
  - Virtual machine platforms (real-time Java)

Issues

- Model-reality distance
  - Application language (C-Sim vs. Python)
  - OS model (C-Sim vs. Java)
- Verification results validity with transition
  - Depends on model-to-reality distance
  - Unknown test coverage
- Collaboration of simulated and real parts
  - Useful for Phase 3
  - Language (Java) or platform (HLA) interfaces

Approach Summary

- Develop, then test? Simulate, debug, transfer!
- Benefits: time, costs
  - Parallel hw and sw development
  - Better debugging facilities
  - Deployed application free from design errors
  - Less need for expensive prototypes
- Details
  - www.kiv.zcu.cz/groups/dss/
  - csim.zcu.cz, j-sim.zcu.cz