Model Based Development of Embedded Control Software

Part 1: Introduction

Claudiu Farcas

Credits: MoDECS Project Team, Giotto

Department of Computer Science
cs.uni-salzburg.at

UNIVERSITÄT
SALZBURG

Contents

- Motivation
- What is an Embedded Control System?
- Traditional programming for control systems
- Model based development

Motivation - Cost

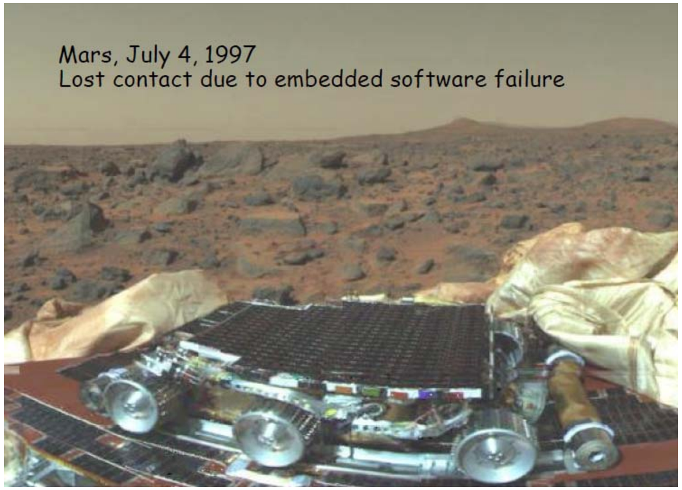
- Development
- Testing
- Integration
- Testing
- Validation
- Certification

$4 billion development effort
> 50% system integration & validation cost

Motivation – Risk analysis

- Mean time between failures
- Failure cost in human lives/money
- Warranty/Insurance
- Fixing/Repairing possibility and costs

French Guyana, June 4, 1996
$800 million embedded software failure



Mars, July 4, 1997
Lost contact due to embedded software failure

Embedded Control System

- Based on software that runs on computers (low powered)
- Interacts with physical world
- Software
  - derived from mathematical functions
  - execution takes non negligible time
  - Increasing complexity

- Consumes power that may be insufficient
- Reliability standards are very high

Embedded Software

- Timeliness
  - Requirements for real-time operations
  - Faster hardware does not solve all problems
- Concurrency
  - Software must react to multiple external stimuli
  - Threads/processes, semaphores, monitors, etc
  - Synchronous reactive languages (Esterel, Lustre)

Embedded Software

- Liveness
  - Software must not lock/crash/terminate
  - Predictable response
- Component technology
  - Interfaces/APIs
  - Libraries
  - OOP
  - Processes/Threads

Embedded Software

- Heterogeneity
  - Mix of hardware and software designs
  - Handling of irregular or periodic events
  - Generalization and particularization of software, implementation language, programming techniques
- Reactivity
  - Respond to the environment at the speed of the environment
  - Real-time constraints, generally safety-critical
  - Adaptation to new requirements – robustness
  - Concurrency analysis and smart compilers

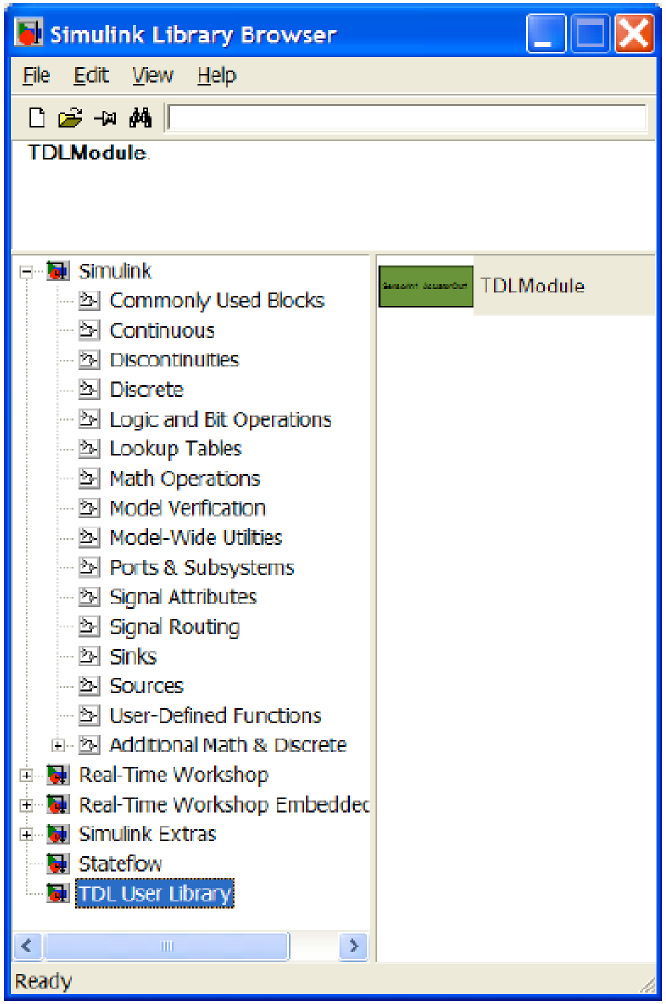
Traditional programming

- Manual coding for more than 90% of application code
- Highly platform dependent (HW + OS)
- Functionality code mixed with timing code
- Hardly reusable code
- High testing and integration costs
- Loss of “overall picture” after several development cycles

Model Based Development

- Application development aided by visual tools (e.g., Matlab)
- Behavior specified via a model (i.e, pure mathematical, descriptive, etc)
- Simulation possible prior to full implementation (e.g., Simulink, Stateflow, etc)
- Shift of development resources from hard-core implementation to better design

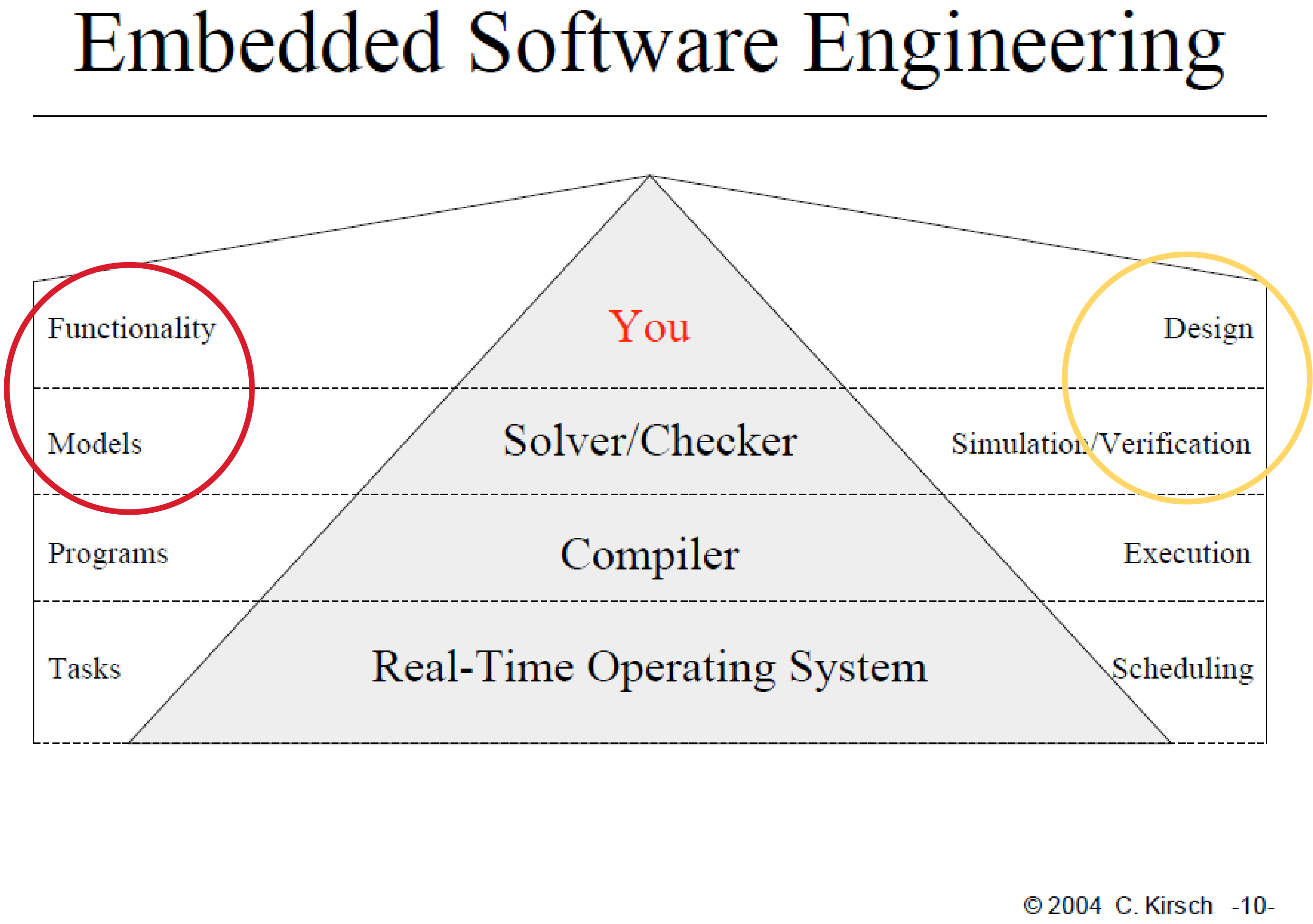
Model Based Development



Model Based Development

- Modular/component oriented design – component frameworks, libraries
- Higher reusability factor
- Automatic code generation (e.g., Real-Time Workshop)
- Increased portability
- Shift from platform oriented to platform independent design

Model Based Development



Embedded Software Engineering

Functionality	You	Design
Models	Solver/Checker	Simulation/Verification
Programs	Compiler	Execution
Tasks	Real-Time Operating System	Scheduling

© 2004 C. Kirsch -10-