Automated model-based powertrain software production

January 10-12, 2004

Tunc Simsek
Mark Wilcutts
Pravin Varaiya
Mobies program

- 4 stage process from control design to embedded s/w
  - 3 levels of control models
  - s/w implementation on target
- Mobies: technology development + B’kley automotive testbeds for
  - Power train
  - CACC/CW
- Focus on ETC
Integrated electronic systems

Legend:
- sensors
- control platforms
- actuators

Outline

✓ Modeling
Design
Analysis
Implementation
Conclusions
Multi-view modeling

• Construct model in 3 levels.
  Level 1: Physical, continuous-time models
  Level 2: Stylized signal characterization (eg. naming convention). Discrete-time and event-triggered controllers only. Restrict use of blocks to those that admit analysis.
  Level 3: Augment model with software guidance.

• Don’t have a single model providing multiple level-specific views of its subsystems.

Level-1 ETC model

Adapted from Jason Souder, Paul Griffiths, Mark Wilcutts, *ETC Simulink Model*, UC Berkeley, November 1, 2001
Level-1 ETC controller

Adapted from Tunc Simsek, *ETC Ptolemy Model*, UC Berkeley, 2002
Level-1 ETC controller
Level-2 ETC controller

Adapted from Tunc Simsek, *ETC Ptolemy Model*, UC Berkeley, 2002
Level-2 ETC controller
Level-3 ETC controller

Summary of example

- Going from models to software is not easy
  - Manual intervention is expensive and error prone
  - Control engineers and software engineers don’t talk the same language and interpret models very differently
- Stylized use of modeling tools helps
  - Capture information for control and software engineers in the models early in the process
  - Capture the information in a presentable fashion (using subsystems, nested state machines, …)
  - Do this in the semantics of the original model (no obvious solution)
  - Do this by negotiating different views of the original model (multi-view modeling)
- Can we automate part or all of the process?
  - Will look at design, analysis, and software implementation techniques developed for Mobies?
Outline

Modeling

✓ Design

Analysis

Implementation

Conclusions
Design

• Anything goes at Level-1
  – Control theory, hybrid systems, …, modeled in Simulink, Shift, Ptolemy, etc.
• A lot of literature for Level-2
  – Discrete-time control and discretization (sample-and-hold)
  – Discrete event and supervisory control theory
• Some work done for Level-3
  – Control view
    • Elimination of jitter attributed to software delays\textsuperscript{1}
    • LQG optimal control in the presence of software induced delays\textsuperscript{2}
    • Control scheduling for networked control systems\textsuperscript{3}
  – Software view
    • Allocation (of control tasks) to (computational) resources (\textsc{Aires, TimeWiz})
    • Pipelining (Synchronous data flow) (\textsc{Ptolemy})
    • Scheduling with deterministic guarantees (\textsc{Giotto})

\textsuperscript{1} Arzen, K.-E.; Cervin, A.; Eker, J.; Sha, L. \textit{An introduction to control and scheduling co-design}, 2000.
Schedule design (AIRES)

Rate Information

Task: manager , Period: 1000, WCET: 500.00
Task: monitor , Period: 1000, WCET: 200.00
Task: servo_control, Period: 1000, WCET: 800.00

Task: manager , Cluster: 1
Resp time: 926.00, Res consumption: 0.50
Allocated to Processor 1

Task: monitor , Cluster: 2
Resp time: 300.00, Res consumption: 0.20
Allocated to Processor 1

Task: servo_control, Cluster: 0
Resp time: 900.00, Res consumption: 0.80
Allocated to Processor 0

Processor 0 has Utilization 0.824000
Processor 1 has Utilization 0.850000

Example generated using AIRES: Automatic Integration of Reusable Embedded Software, Toolkit version 2.0, University of Michigan.
Outline

Modeling

Design

✓ Analysis

Implementation

Conclusions
Analysis

• A lot of literature for Level-1
  – Hybrid system simulation tools (Charon, Shift, Lambda-Shift, Ptolemy)
  – Hybrid system verification tools (Charon, SAL, Checkmate)

• Some work done for Level-2
  – Control, communication and signal processing toolboxes (Simulink)

• Some work done for Level-3
  – Control view
    • controller performance for software-induced delay and jitter
    • quantization-induced errors
  – Software view
    • schedulability of (control) tasks (AIRES, TimeWIZ, Taxys¹)

Outline

Modeling
Design
Analysis
✓ Implementation
Conclusions
Implementation

• Start with a Level-2 model
  – Typically 50-60 subsystems
  – Each subsystem has 10-100 input/outputs

• Enforce style of Level-2 model

• Produce a Level-2 software model

• Produce a Level-3 software model
Ford style guide (top-level)

- Software timing
- Hardware timing
- Software I/O
Ford style guide (context diagram)

Feature I/O and interconnections

Explicit data exchange
Ford style guide (execution context)

Execution context I/O, and interconnections
Level-2 software model

Target def’n  Data model  Execution functions

Model Info:
Software oriented diagram for
Berkeley Powertrain CEP.

Tunc Simsek, Dec 20, 2002
simsek@eecs.berkeley.edu

Target:
Ocles 2.0
Acalo: CME-0555 Rev. B.
Level-2 to Level-3

- Targetlink is used to produce target code for execution contexts
- Timing analysis to allocate schedule
- Target support is used to assign device drivers to memory maps
Level-3 (execution contexts)

Unique function ‘id’

Encapsulated execution context – no implicit interconnections
Encapsulated scheduler – calculates the ‘id’ and ‘time-stamp’ of the next feature to execute.
Level-3 (scheduler, memory map)

Schedule calculation

Device-driver memory maps
Implementation of Level-3 models

• What is your trigger mechanism
  – time, events, periodic?
• What is your schedule?
• What are the component interaction semantics?
  – data flow, queues, buffers, stacks?
• What is your concurrency model?
• How do you map Level-3 components into RTOS tasks?
• How do you use RTOS features: memory, scheduling, priorities, threads, IPC, drivers?
Conclusions

- Large gaps between levels 1-2 and level 3
- Larger gap between level 3 and implementation
- Little feedback from implementation to design
- Software model helps to use standard software engineering tools