Execution modeling

The missing leg in model-based development of performance-oriented embedded systems

Jukka Mäki-Turja & Mikael Sjödin @ MRTC and Arcticus Systems
Affiliations

• Mälardalen Real-time research Centre
  – Research centre at Mälardalen University
  – Internationally competitive research in component-based software engineering and real-time
  – 13 Professors, 20 Senior Researchers, 45 PhD students

• Arcticus Systems AB
  – State of the art development tools for dependable real-time system
    • Developed in close cooperation with MRTC for over 15 years
  – Rubus Integrated Component development Environment
    • Design, Analysis and Synthesis tools and OS
Model based development (MBD)

- Emerging approach for embedded real-time systems
  - Handling increased complexity of products
  - Shortening development cycles
  - Addresses quality concerns

- This is, we believe, a healthy trend
  - Reason about a system
    - On various levels of abstraction
    - Early in the development process
  - Generating code directly from models
Traditional MBD

- Structural modeling
  - UML – class diagram, sequence diagram, use cases

- Functional modeling
  - Statemate
  - Matlab/simulink
  - UML statemachines

- Push the button => Runnable system

**HOWEVER!**

No Idea of run-time properties!
Detrimental for performance critical systems
The missing leg: Execution modeling

- Solution
  - Promote execution to the modeling level

- Execution modeling
  - What executes?
  - When does it execute?
  - Who interacts with it?
  - How long does it take?
  - How much resources does it need?

- Information and control over these properties means that performance can be predicted and tuned
Aim and focus on execution modeling

- Concerned with the run-time properties and requirements of software functions
- Closes the semantic gap from functional behavior and execution behavior
- The developer have a direct control of the actual run-time behavior
- From a formal model that includes run-time properties an optimized run-time framework can be generated
- Control of execution details is necessary for real-time analysis as well as the overall system performance.
The development context
The system architecture

• Expresses the properties and requirements of SW Components
  – A control- and dataflow graph model of the execution environment of SWC’s
    • Triggering and data dependencies
  – How SWC are triggered by clocks or events

• Real-time properties and requirements.
  – End-to-end deadlines of trigger chains
  – Precedence constraints SWCs
  – Resource usage (space and time) of each SWC

• Expressed in the Rubus Component Model (RCM)
Developers view

• Express properties and requirements for SW functions
• Control over execution infrastructure
  – Separate code and data-, control-flow
• Simple yet expressive
• Different levels of abstraction
  – SWCs, Assemblies, Composites, Modes, ...
• Different views
  – Control flow: triggering
  – Data flow
  – Real-time temporal view
    • Properties and requirements
  – Per node, system, communication
Software Circuits

Data ports
- Input (n) and Output (n) ports

Trigger ports
- Input (1)
- Output
  - Unconditional (1) and Conditional (n)
Connectors
- 1 producer – n consumers
- Output to Input
- Output to “sink”
- Constant to Input
• **Assemblies & Composites**

• **Input and output ports**
  – Data (n) and trigger (n)

• **Collection of**
  – SWCs+ASMs
  – Their interconnections
Triggering

• Chains of trigger ports define precedence

• Started by
  – Clocks
  – Interrupts
  – Events (=>Unconditional Trigger Ports)

• 1-to-many

• AND to support many-to-1
View of the analysis framework

- RCM automagically transformed to an analysable model
  - Task model with offsets
    - Temporal dependencies
    - Precedences
    - Triggering events: Period, MINT
      - Future complex triggering conditions?
    - Stack usage
- Analyse
  - Response times (TT and ET tasks)
  - Overall system stack usage required
  - Specific for Embedded control systems
View of the run-time system

- Predictable behavior
  - Adheres to:
    - Component requirements
    - Analysis assumptions

- Low footprint
Industrially proven concept

• VCE has had 10+ years of execution modeling
  – Not a single timing fault during that time

• BAE Hägglunds
  – In the process of implementing stear-by-wire and break-by-wire

• Industrial partners over 15 years:
  – Arcticus, BAE, CC Systems, Hoerbiger, SICS, Volvo-Technology, Volvo-CE
Theory need to be encapsulated in tools

Rubus ICE

Design

Analysis

Off-Line

Synthesis

On-Line

Test

Simulator

Target
Conclusion

• Execution modeling concerns with describing run-time properties and requirements
  – Control and understanding of the execution is in focus

• Execution modeling
  – Simplified design and verification
  – Control of execution gives predictability, reproducability and control over performance
  – Analysis and formal guarantees

• Good experience from the vehicle domain
  – Can influence the telecom domain?
  – Might need other analysis and synthesis frameworks
Thank you!
Questions, Comments?

Execution-modeling
The missing leg in model-based development of performance-oriented embedded systems