ProCom – a Model-Intensive Component Model for Embedded Systems

Jan Carlson, Sevrine Sentilles
Aneta Vulgarakis, Tomáš Bureš, Ivica Crnković

Charles University 2010-01-12
Mälardalen University
The PROGRESS Centre for Predictable Embedded Software Systems

- 5-year project (2006-2010)
- ~45 people involved
  (8 prof., 15 researchers, 20 PhD stud.)
- Overall research vision:
  
  *Predictable component-based embedded software*

- Targeted domains: Vehicular and automation
  - Complex!
  - Distributed, control-intensive, embedded systems.
  - Safety-critical, real-time demands.
PROGRESS background

- The PROGRESS approach focuses on:
  - Reusability
    Component-based development
  - Predictability (timing, resource usage and dependability)
    Early model-based analysis
  - Run-time efficiency
    Lightweight runtime component framework
ProCom – Key aspects

- Design-time components.
- Rich component concept.
  - Including source code, models of timing and resources, analysis results, documentation, etc.
  - Reuse all this information.
  - Components of different maturity should be allowed to co-exist.
ProCom – Key aspects

• Components abstracted from physical deployment.

• Different concerns depending on granularity.
  – Distribution, communication, analysis, etc.
ProCom – a two-layered component model

- **ProSys (upper layer)**
  - Subsystem components
  - Active, typically distributed
  - Asynchronous message passing

- **ProSave (lower layer)**
  - "Function block" components
  - Passive, non-distributed
  - Explicit transfer of data and control

- **Connection between the layers**
  - A subsystem can internally be modelled by ProSave.
ProSys – the upper layer

- Components (subsystems):
  - Active, possibly distributed.
  - Interact through message ports.

- Communication:
  - Asynchronous messages.
  - Explicit message channels.
ProSave – the lower level

- Passive components (similar to task or function block)
- Interact through input- and output ports.
  - Data ports
  - Trigger ports
- Component semantics:
  1. Initially passive, receiving input data.
  2. When triggered, read input data and turn active.
  3. Write output once.
  4. Return to the passive state.
ProSave – the lower level

- More complex components can have:
  - Multiple output groups:
    - Output can be produced at different points in time.
    - Each group written once per activation.
  - Multiple input groups (services):
    - Services can share state.
    - Individual control flows
ProSave – the lower level

• Separated data- and control flow

• Hierarchical nesting
  • Primitive components (implemented in C)
  • Composite components
ProSave – the lower level

- Connectors for more elaborate control:
  - Control fork
  - Control join
  - Control selection
  - Control or
  - Data fork
  - Data or
ProSave – the lower level

- Connectors for more elaborate control:
  - Control fork
  - Control join
  - Control selection
  - Control or
  - Data fork
  - Data or
Modelling a ProSys subsystem in ProSave

- Message ports ↔ trigger and data
- Clocks and events
Truck example
Truck example II

ProComTruck

Movement → Light mode → Lights

Lights

Light actuators

Movement

Line sensors

Speed sensor

Movement decision

Steering

Speed
Truck example III

Movement decision

Mode changer

Control select

Follow

Find

Turn

Light mode

Data or

Control or
ProCom – Summary

• A component model for “distributed control-intensive embedded systems”.

• Reuse, predictability, early model-based analysis.

• Rich design-time component notion including models, attributes, realization, analysis results, etc.

• Two layers:
  – ProSys: Active subsystems, message passing.
  – ProSave: Passive components, trigger/data flow.
Thank you!