PROGRESS - Predictable Component-based Development of Embedded Systems

Paul Pettersson

PROGRESS
A national Swedish Strategic Research Centre
Mälardalen University

- Campuses in Eskilstuna & Västerås
- 110 km west of Stockholm
- Top ranking in quality work evaluation by Swedish National Agency for Higher Education
- Six prioritized research areas
  - Embedded systems
  - Innovation and Product Development
  - Environment, Energy and Resource Optimization
  - Sustainable Development, Working Life, and Management
  - Welfare and Health
  - Didactics and Intercultural Communication

- In numbers:
  - Staff 1000
  - Professors 60
  - Students 13000
  - 60 programs and 750 courses
  - Turnover 75MEuro
MRTC & Progress

Mälardalen Real-Time Research Centre:
- Hosted by Mälardalen univ.
- Real-Time Embedded Systems
- Industrial Software Engineering

Research groups:

Progress:
- National Strategic Centre for Embedded Software Development hosted by MRTC
- Lead by Hans Hansson
- 2006-2011

Research groups:
- Real-Time Modeling & Analysis
  Paul Pettersson
- Program Analysis
  Björn Lisper
- Dependable Software Engineering
  Sasi P./Kristina L.
- Industrial Software Engineering
  Ivica Crnkovic
- Communication Performance & Analysis
  Mats Björkman
- Real-Time Systems Design
  Hans H./Mikael S.
- Embedded Systems SW Engineering
  Christer Norström
Focus on component-based development of real-time embedded systems

Hypothesis:
building embedded software (and systems) from reusable components
  – complexity,
  – integration, and
  – quality assurance

can be handled in a more cost efficient and scalable way
Overview

- MRTC, and Progress
- Procom – Progress component model
  - ProSave, ProSys, attribute framework
- PrIDE – Progress IDE
  - editing
  - simulation
  - verification

- UPPAAL PORT
  - analysis model for real-time components
  - verification by partial-order reduction

- REMES
  - modelling of embedded resources
  - analysis: feasibility, optimality, trade-off

- Conclusion
PROCOM
A component model for real-time embedded systems
ProCom – Key aspects

- **Design-time components.**

- **Rich component concept.**
  - Including models of timing and resources, analysis results, documentation, source code, etc.
  - Support for reuse components.
  - 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 multi-layered component model

- ProSys model (upper layers)
  - Systems and subsystem components
  - Active, typically distributed
  - Asynchronous message passing

- ProSave model (lower layers)
  - Function block, primitive
  - Passive, non-distributed
  - Explicit transfer of data and control

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

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

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

- Passive components
  - Similar to task or function block

- Interact through input- and output ports.
  - Data ports
  - Trigger ports

- Read-execute-write semantics:
  1. Initially **passive**, receiving input data.
  2. When triggered, **read** input data and turn active
  3. **Executing** internal component behaviour
  4. **Write** output; goto 1.
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
  • Composite components
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
Progress IDE - PrIDE

Procom editor

REMES editor

Timed Automata editor

Verification using UPPAAL Port and Cora

Simulator
ATTRIBUTE FRAMEWORK
Integration of Extra-Functional Properties in PROCOM
Attribute Framework

- Integration of extra-functional properties in the component model
- Reuse and composability of EFP
  - often poor support for this in CBD
- Attribute Framework:
  - Manage EFP in a systematic way
  - Store various analysis results (for reuse)
  - Generic for may different type of EFPs
Huge List of Properties...

Nb of reuse

Throughput

Security

Cost

Safety
Reliability
Availability
Recoverability
Maintainability
Accessibility

Integrity

Precision

Credibility

Evolvability

Confidentiality

Execution time
Priorities
Deadline
Schedule policy
End-to-end deadline
Response time
Computation time
WCET/BCET

LoC

Value range

Confidentiality

Compliance to standard

Static memory usage
Dynamic memory usage
CPU usage
Power consumption
Memory footprint
Disk access
Network access

Nb of tests

Huge List of Properties...

...
Origin differs...
Relation to different component model entities differs...
Development phase differs...

- Requirements Specification
- Early estimate
- Static Analysis
- Probabilistic Analysis
- Model Checking
- Measurement(s)
- Expert estimate
- Schedulability Analysis
- Simulation
- Value refinement
Attribute Structure

Attribute = (Type, Identifier, Value)

Value = (Data, Metadata, Validity Condition)
PROGRESS IDE
Worst-case Execution Time

General information

<table>
<thead>
<tr>
<th>Id</th>
<th>se.mdh.progressside.attribute.ports.WorstExecTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Worst-case Execution Time</td>
</tr>
<tr>
<td>Category</td>
<td>Category B</td>
</tr>
<tr>
<td>Applicability</td>
<td>ProComMetamodel ProSave InputTriggerPort</td>
</tr>
<tr>
<td>Brief description</td>
<td>Worst-case execution time between an input and an output trigger ports within the same service</td>
</tr>
<tr>
<td>Contributing bundle</td>
<td>se.mdh.progressside.attributes</td>
</tr>
</tbody>
</table>

Detailed description

Definition

The worst-case execution time (WCET) between two port groups of a service (see Section 4.3.2 and Figure 9) or, more precisely, between an input trigger port of an input port group and an output trigger port of one of the output port groups within the same service is measured as an integer number of milliseconds, always greater than zero. More specifically, it is an attribute of a input trigger port and the output trigger port to which the WCET is measured is referred by the attribute value.

Remarks

This serves for demonstration purposes only.
UPPAAL PORT
Modelling and verification of Real-time components
Modelling and Verification

- Establish correctness at design time of models
- Functional and timing properties of components
  - Model of functional and real-time behaviour
  - Verification of safety liveness properties
- Difficulties/complexity:
  - hierarchical model, communication structure, functional behaviour, timing

- Common approach:
  - Perform analysis of component model on equivalent "flat" model
  - Performance problems

- Our approach:
  - Perform analysis directly on hierarchical component model
  - Apply reduction techniques to exploit component model structure
  - Partial-order reduction
ProSave + Timed Automata

- **ProSave**
  - Ports, read-execute-write
  - Function and timing
    - timed automaton with start/final location
    - ports mapped to data variables
    - analysis model
  - Horizontal composition
    - Connections
    - Components
      - Vertical composition
      - Composite components
PORT for Timed Systems

• Attempted before:
  – Bengtsson et. al. (’98), Minea (’99)
    • Local time semantics
    • Allows time to progress independently in parallel automata
  – Niebert et al (event zones, ’04-), Maler et al (interleavings ‘06)

• Our approach
  – Based on local time semantics
  – Structured model → more information
  – Components execute independently
    • read → execute → write semantics
Partial Order Reduction

- Interleaving semantics:
  - A and B parallel actions: explore AB and BA

- Reduce interleaving
  - A and B are independent
  - Commutativity, enabledness, ...

- Explore representative traces
  - Maintains correctness
  - BA representative of AB

- Timing adds problems
Independence in Components

• **active** component (in a transition):
  – delay A = \{ A \}
  – internal A = \{ A \}
  – write A = \{ A \} \cup \{ K \mid \text{connection from } A \text{ to } K \}

• **independent**: transitions \( \alpha \) and \( \beta \) if
  – no component active in both:
    \[
    \text{active}(\alpha) \cap \text{active}(\beta) = \emptyset
    \]

• example:

  ![Diagram of components A, B, C, and D with connections and arrows indicating active and internal states]

  **Dependent:**
  - internal A – write A
  - internal C – write A
  - internal D – write A
  - write A – write B
  - internal B – write B
  - internal C – write B
Local Time Semantics

- **local clocks** $c^A$, $c^B$, $c^C$
- **local transitions**: delay $K$, internal $K$,
  - delay $K$ advances clock $c^K$
- **write transition of A**
  - synchronize reference clocks $c^A$ and $c^K$
    if A is connected to K

![Diagram of Local Time Semantics]

- Time:
  - Idle
  - Internal
  - Final
• internal K and delay K as before
• write transition of A
  – if A triggers K then $c^A = c^K$
  – If dependent($write^A$, $write^K$) then

Further Relaxed Synchronization

Preserve order of dependent transitions

Preserves start of component C

A  B  C

Internal

Idle

Final

Time

write (trig)

A

internal

write (data)

B

internal

delay

C

internal

write (data)
Implementation

- UPPAAL PORT: Extension of model-checking tool UPPAAL with
  - partial order reduction
  - native support for component model ProSave
- IDE support for
  - editing and simulation of ProSave
  - PORT model-checking of local reachability
  - model-checking of safety and liveness
Benchmark

• Each component:
  - initial read
  - delay $\in [0,4]$ write

Based on example of [Salah, Bozga and Maler’06]

<table>
<thead>
<tr>
<th>Method</th>
<th>3x2</th>
<th>3x3</th>
<th>3x4</th>
<th>4x3</th>
<th>4x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>global</td>
<td>264/0.15s</td>
<td>499/0.19s</td>
<td>838/0.25s</td>
<td>2553/0.45s</td>
<td>4778/0.88s</td>
</tr>
<tr>
<td>local</td>
<td>52/0.14s</td>
<td>78/0.17s</td>
<td>104/0.22s</td>
<td>145/0.25s</td>
<td>195/0.43s</td>
</tr>
<tr>
<td></td>
<td>4x5</td>
<td>5x4</td>
<td>5x5</td>
<td>6x5</td>
<td>6x6</td>
</tr>
<tr>
<td>global</td>
<td>8146/1.7s</td>
<td>26108/5.2s</td>
<td>48096/11.7s</td>
<td>481318/2m29s</td>
<td>1087216/7m30s</td>
</tr>
<tr>
<td>local</td>
<td>245/0.76s</td>
<td>326/0.94s</td>
<td>411/2.0s</td>
<td>637/4.8s</td>
<td>770/9.8s</td>
</tr>
<tr>
<td></td>
<td>6x7</td>
<td>7x4</td>
<td>7x5</td>
<td>7x6</td>
<td>7x7</td>
</tr>
<tr>
<td>global</td>
<td>2350413/19m40s</td>
<td>5166020/26m50s</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>local</td>
<td>903/17.7s</td>
<td>736/4.6s</td>
<td>932/10.7s</td>
<td>1128/21.5s</td>
<td>1324/39.0s</td>
</tr>
</tbody>
</table>

Table 1. Benchmark results. We use $\bot$ to denote that the experiment did not terminate in 30 minutes. Based on example of [Salah, Bozga and Maler’06]

global = no partial order reduction
local = relaxed partial order semantics
REMES
Modelling and verification of Embedded Resources
Embedded Resources

• Embedded systems typically designed w.r.t. resource constraints
  – computational power (CPU), memory, energy, bus bandwidth, ports, etc.

• Challenge
  – provide early design stage modelling and prediction methods
  – model resource usage and provide analysis techniques
  – resource-wise feasibility, optimal resource usage, and trade-off analysis
Embedded Resources

\[ \{R_B\} \leq \{R_{C1}\} \]

Repository

\[ 
\begin{align*}
C_1 & \quad \{R_{C1}\} \\
C_2 & \quad \{R_{C2}\} \\
C_n & \quad \{R_{Cn}\} \\
C_3 & \quad \{R_{C3}\}
\end{align*} 
\]
Resource Analysis Problems

- **Feasibility**
  - Accumulated resource usage within provided resource bounds
- **Optimal and worst-case consumption**
  - Min/max accumulated resource usage
- **Trade-off analysis**:
  - Decide the best trade-off between multiple resources, possibly dependent, e.g., memory and cpu
REMES

- **REsource Model for Embedded Systems**
  - To model resource-wise behaviour of interacting embedded components
  - Charon-based - atomic or composite - models behaviour and timing of component
  - control points - init, entry, exit
  - variables - integer, clocks (arrays) - global, local
  - actions
  - constraints (invariants) - conditional connectors

![Diagram of REMES model with nodes, edges, and labels representing modes, control points, variables, and actions.](image-url)
Embedded Resources

- \( r \) – accumulated resource consumption
- \( r' \) – rate of consumption over time
  - discrete or continuous
  - referable or non-referable

Examples:
- \( r' = 0 \) and referable: discrete, e.g. memory
- \( r' = 0 \) and non-referable: discrete, e.g., CPU
- \( r' = n, n \in \mathbb{Z} \), and non-referable: energy or bandwidth
Analysing REMES based ES

- REMES modes have access to resources $R_1, \ldots, R_n$
- Goal to analyse scenarios of resource usage
- Analysis model:

$$r_{tot} = (w_1 * r_1) + \ldots + (w_n * r_n)$$

- $r_{tot}$: accumulated resource consumption for $R_1, \ldots, R_n$
- $r_1, \ldots, r_n$: accumulated consumption of $R_1, \ldots, R_n$
- $w_1, \ldots, w_n$: weights, relative importance of $R_1, \ldots, R_n$
REMES to PTA

- Translation to Priced Timed Automata [Alur et al’01, Behrman et al’01]
  - Timed automata extended with linear cost variable
  - Minimum reachable, etc decidable
- Mapping of REMES modes, edges, variable, etc to PTA locations, edges, etc.
- Multiple resources $r_1, \ldots, r_n$:

  \[
  \text{cost} = (w_1 \times c_1) + \ldots + (w_n \times c_n)
  \]

  - $c_1, \ldots, c_n$ : cost of resource $r_1, \ldots, r_n$
  - for each location: $\text{cost}' = (w_1 \times c'_1) + \ldots + (w_n \times c'_n)$
    where $w_1, \ldots, w_n$ constants and $c'_1, \ldots, c'_n$ static
Analyzing REMES models

Model Checker
(UPPAAL Cora)

PTA (MPTA)

resource-aware property

$EF_{\text{cost} \leq n} \psi$

Assumptions from hardware abstraction:
Memory budget, Bandwidth, Cost model

error trace

yes
Resource Analysis

• Feasibility analysis:
  – are the accumulated values of consumed resources within the provided resource amounts?
  – one cost variable encoding all accumulated resources
    • Strong feasibility: \( AF_{cost \leq n} \)
    \[ AG(q \Rightarrow AF_{cost \leq n}) \]
    • Weak feasibility: \( EF_{cost \leq n} \)
    • Live feasibility: \( AG(q \Rightarrow EF_{cost \leq n}) \)
Resource Analysis (2)

- Optimal and Worst-Case Resource Consumption
  - minimum/maximum cost for reaching given location or predicate
  - minimizing/ maximizing the one-cost function

\[
\text{cost} = (w_1 \times c_1) + \ldots + (w_n \times c_n)
\]

- decidable also if cost is not a monotonically increasing function [Bouyer et al. “On the optimal reachability problem of weighted timed automata” 2007]
Resource Analysis (3)

- Trade-off analysis:
  - more than one property to satisfy
  - Pareto analysis
  - adjust weights and use weighted sum
    - weights can be set by AHP analysis
  - minimize a primary cost, while imposing an upper bound on secondary cost (two costs)
    - Optimal conditional reachability of MPTA [Larsen & Rasmussen 2005]. E.g. energy minimized, cpu bounded, location v:

\[
EF_{(w_{\text{energy}} \times c_{\text{energy}}) \leq n} (v \land (w_{\text{cpu}} \times c_{\text{cpu}}) \leq m)
\]
Integrated in PrlDE

- REMES and PTA editor/visualizer
- Transformation to TA or PTA
- Simulation (using UPPAAL)
- Verification by model-checking in UPPAAL, UPPAAL Port (partial-order) and UPPAAL Cora (PTA)
Conclusion

This talk:
• Procom:
  – Progress component model
  – ProSys and ProSave
  – attribute framework
• PORT + REMES:
  – partial order reduction
  – modelling and analysis of embedded resources
  – feasibility, optimality and trade-off analysis
• PrIDE: Progress IDE
  – modelling, simulation, model-checking (functional, timing, resources)

Not in this talk:
• Modelling:
  – UML state machines + MARTE
  – connection to EAST-ADL2
  – modelling patterns
  – error modelling
• Predictability analysis
  – static analysis
  – WCET
  – dependability
• Platform
  – scheduling
• Case studies, …
Thanks!

Acknowledgments: Tomáš Bureš, Jan Carlson, Ivica Crnkovic, John Håkansson, Marin Orlić, Cristina Seceleanu, Séverine Sentilles, Petr Stepan, Jagadish Suryadevara, Aneta Vulgarakis