Early Embedded Software Design Space Exploration Using UML-based Estimation

17th IEEE International Workshop on Rapid System Prototyping (2006)

Marcio F. da S. Oliveria, Lisane B. de Brisolara, Luigi Carro, Flavio R. Wagner

2009. 5. 13
Presented by Jinho Choi

© KAIST SE LAB 2009
Contents

- Introduction
- Background
- UML-based Estimation Methodology
- Case study
- Conclusion
- Discussion
“Embedded Everywhere”
- Realization with embedded systems

Characteristic of embedded systems
- Limited resources
- Cost constraints
- Real time operations

Therefore, embedded system design is so difficult
Change of development focus from code to model
  - Provide a chance for a fast design space exploration in the early design steps

However, SW engineers do NOT know the impact of design decisions for target embedded system
  - Performance, energy, and memory footprint
UML-based estimation methodology is proposed

Objective

- Finding the adequate design solution of embedded software in the early design step

Implementation tool

- SPEU(System Properties Estimation with UML)
Platform-Based Design*

Objective

- Maximizing the reuse of pre-designed components
- Solving the problems by the ever-increasing complexity and time-to-market pressure

Platform
- A library of pre-designed components

* Sangiovanni-Vincentelli A. et al, Benefits and Challenges for Platform Based Design, Design Automation Conference, June 2004
Background (2/2)

- **UML-SPT**
  - Real-Time UML Profile
  - Specify and Design embedded systems
    - UML extensions such as stereotypes, tagged values, and constraints
  - General Resource Model
    - Parameters for modeling resource, time, concurrency and schedulability

UML-based Estimation Methodology

- Estimation of system properties
  - Mapping the application model into a target platform

- Definition of three aspects
  - The UML modeling rules
  - The platform repository model
  - The estimation process
Modeling Rules

- Specifying the target software
  - UML 2.0 and UML-SPT notation

- Describing scenarios
  - Use cases
  - Class diagrams
    - Specifying the structural model
  - Sequence diagrams
    - Specifying conditional execution, interaction between objects
  - Deployment diagrams
    - Mapping the software component to processing unit
The Platform Repository Model

- Based on the General Resource Model of UML-SPT

- Storing information about reused components

<table>
<thead>
<tr>
<th>Category</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>- Timer services</td>
</tr>
<tr>
<td></td>
<td>- Real-time API</td>
</tr>
<tr>
<td></td>
<td>- Instruction set</td>
</tr>
<tr>
<td></td>
<td>- Size for each instruction</td>
</tr>
<tr>
<td></td>
<td>- Number of execution cycles for each instruction</td>
</tr>
<tr>
<td>Energy</td>
<td>- Energy consumption for each instruction</td>
</tr>
<tr>
<td>Memory</td>
<td>- Data types</td>
</tr>
<tr>
<td></td>
<td>- Dynamic allocation</td>
</tr>
<tr>
<td></td>
<td>- Reserved Memory</td>
</tr>
<tr>
<td></td>
<td>- Method allocation rules</td>
</tr>
</tbody>
</table>
Estimation Process

Select use cases → Select scenarios → Add information? → Structural Analysis → Behavioral Analysis → Sequence Model → Map to platform → Estimation

Estimation results → Application signature → Platform Model 

© KAIST SE LAB 2009
1. Select Use Cases and Scenarios

Objective
- Identifying critical use cases and scenarios to need estimation
2. Increase Model Information

- Adding more information on scenarios
  - The number of executions for a scenario
  - The best and worst case scenarios
  - Loop bounds
3. Application Analysis (1/2)

- Structural analysis
  - Analyzing with class diagrams

- Behavioral analysis
  - Analyzing with sequence diagrams
    - Dependences between scenarios, loop and conditional executions
3. Application Analysis (2/2)

- **Application signature**
  - Output of application analysis step
  - Information about pseudo-trace
    - Symbolic Instruction flow graph
4. Application Mapping

- **Objective**
  - Mapping application signature to platform components
  - Generating the symbolic instructions costs for memory, performance, and energy
5. Estimation

- Known the best and worst case execution path
  - Adding symbolic instruction costs

- Unknown best and worst case execution path
  - Analyzing the pseudo-trace with Integer Linear Programming formulation
Case study: Early SW Design Exploration (1)

- **Real-time embedded system for wheelchair**
  - **Objective**
    - Showing process of choosing adequate modeling solution

Wheelchair movement control use case
- Actuating & sensoring scenarios

4 candidate modeling solutions

- A processing unit - FemtoJava processor

The result is compared to the result of cycle-accurate simulation

© KAIST SE LAB 2009
Case study: Early SW Design Exploration(2)

- The first modeling solution
  - Actuating & sensing scenarios
    - 2 threads implement the movement actuating function
    - 2 threads implement the movement sensoring function
    - Use of joystick movement in software

<table>
<thead>
<tr>
<th>Property</th>
<th>Estimated</th>
<th>Exact</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program memory (Kbytes)</td>
<td>5,460</td>
<td>6,248</td>
<td>-12.61</td>
</tr>
<tr>
<td>BC Data memory (Kbytes)</td>
<td>510</td>
<td>582</td>
<td>-12.37</td>
</tr>
<tr>
<td>WC Data memory (Kbytes)</td>
<td>510</td>
<td>582</td>
<td>-12.37</td>
</tr>
<tr>
<td>BC Performance (Cycles)</td>
<td>27,383</td>
<td>28,588</td>
<td>-4.21</td>
</tr>
<tr>
<td>WC Performance (Cycles)</td>
<td>40,106</td>
<td>41,591</td>
<td>-3.57</td>
</tr>
<tr>
<td>BC Energy (mJ)</td>
<td>38,723,290</td>
<td>40,569,570</td>
<td>-4.55</td>
</tr>
<tr>
<td>WC Energy (mJ)</td>
<td>56,477,333</td>
<td>58,863,463</td>
<td>-4.05</td>
</tr>
</tbody>
</table>

Results for the first solution
The second modeling solution

- Actuating & sensing scenarios
  - No thread design
  - Use of joystick movement in software

<table>
<thead>
<tr>
<th>Property</th>
<th>Estimated</th>
<th>Exact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program memory (Kbytes)</td>
<td>-69.83%</td>
<td>-66.98%</td>
</tr>
<tr>
<td>BC Data memory (Kbytes)</td>
<td>-36.47%</td>
<td>-36.08%</td>
</tr>
<tr>
<td>WC Data memory (Kbytes)</td>
<td>-36.47%</td>
<td>-36.08%</td>
</tr>
<tr>
<td>BC Performance (Cycles)</td>
<td>-95.42%</td>
<td>-93.36%</td>
</tr>
<tr>
<td>WC Performance (Cycles)</td>
<td>-65.82%</td>
<td>-65.32%</td>
</tr>
<tr>
<td>BC Energy (mJ)</td>
<td>-99.61%</td>
<td>-93.31%</td>
</tr>
<tr>
<td>WC Energy (mJ)</td>
<td>-66.57%</td>
<td>-65.51%</td>
</tr>
</tbody>
</table>

Results for the second solution
Case study: Early SW Design Exploration(4)

- The third modeling solution
  - Actuating & sensing scenarios
    - Only One thread executes both the actuating and sensing functions
    - Use of joystick movement in software

<table>
<thead>
<tr>
<th>Property</th>
<th>Third x First</th>
<th>Third x Second</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Estimated</td>
<td>Exact</td>
</tr>
<tr>
<td>Program memory</td>
<td>-16.76%</td>
<td>-16.65%</td>
</tr>
<tr>
<td>BC Data memory</td>
<td>-22.16%</td>
<td>-25.95%</td>
</tr>
<tr>
<td>WC Data memory</td>
<td>-22.16%</td>
<td>-25.95%</td>
</tr>
<tr>
<td>BC Performance</td>
<td>-71.50%</td>
<td>-68.15%</td>
</tr>
<tr>
<td>WC Performance</td>
<td>-49.56%</td>
<td>-47.89%</td>
</tr>
<tr>
<td>BC Energy</td>
<td>-72.01%</td>
<td>-68.16%</td>
</tr>
<tr>
<td>WC Energy</td>
<td>-49.93%</td>
<td>-48.12%</td>
</tr>
</tbody>
</table>

Results for the third solution
Case study: Early SW Design Exploration (5)

The fourth modeling solution

- Actuating & sensing scenarios
  - Similar third solution
  - No use of joystick movement in software

<table>
<thead>
<tr>
<th>Property</th>
<th>Fourth / Second * 100</th>
<th>Fourth / Third * 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program memory</td>
<td>166.36%</td>
<td>-3.48%</td>
</tr>
<tr>
<td>BC Data memory</td>
<td>24.07%</td>
<td>-1.95%</td>
</tr>
<tr>
<td>WC Data memory</td>
<td>24.07%</td>
<td>-1.95%</td>
</tr>
<tr>
<td>BC Performance</td>
<td>448.92%</td>
<td>-11.85%</td>
</tr>
<tr>
<td>WC Performance</td>
<td>-1.87%</td>
<td>-33.51%</td>
</tr>
<tr>
<td>BC Energy</td>
<td>552.14%</td>
<td>-10.66%</td>
</tr>
<tr>
<td>WC Energy</td>
<td>-0.14%</td>
<td>-33.34%</td>
</tr>
</tbody>
</table>

Results for the fourth solution
Conclusion

- UML-based estimation methodology
  - SPEU tool provides system properties estimation from UML model
  - SW designers can select the most adequate modeling solution in early design stage
Discussion

- Contributions
  - UML-based estimation methodology
  - A feasibility study of early SW design exploration
  - Comparison to alternative design solutions

- Limitations
  - Lack of information about SPEU tool
  - Preparation of platform repository model
  - Hard to estimate system properties
Thank You.