"Following a Model-Based Systems Engineering approach to bridge the gap from SysML modeling to HW/SW co-design"

mark.elpers@medtronic.com

Telelogic User Group Conference 2008
This presentation shows how we propose to enhance the Telelogic Harmony process by adding an architectural development and analysis process.

We then present a vision for a tool suite, which includes Rhapsody, to support this process.

Finally, we present our work to date with IBM/Telelogic & Cadence Design Systems to implement this vision and to generate cycle-accurate SystemC for portions of the SysML model that have been apportioned to hardware.

With the existing systems-software path provided by Telelogic Harmony we have thus bridged the gap from system design using SysML modeling to both HW & SW implementation.
• IBM/Telelogic Harmony
• Extending Harmony
• Architecture Process Overview
• System Design Process Manager
• SystemC Code Generation For HW
• Questions
• IBM/Telelogic Harmony
• Extending Harmony
• Architecture Process Overview
• System Design Process Manager
• SystemC Code Generation For HW
• Questions
2.2 Harmony-SE Overview

- **Requirements Analysis**: Capturing of requirements and grouping of requirements into use cases.
- **System Functional Analysis**: Use case-based identification and verification/validation of system-level operations (= set of functional requirements).
- **System Architectural Design**: Optional decomposition of system-level operations and allocation of operations to (functional or physical) subsystems. Definition of subsystem interfaces.
- **Subsystem Architectural Design**: Partitioning of operations between subsystem components (HW and/or SW). Definition of subsystem component interfaces.
CASE STUDY: SECURITY SYSTEM 24
4.1 Setting up a SysML Project 25
4.2 Security System Requirements 27
4.3 Requirements Analysis 29
4.4 System Functional Analysis 31
4.4.1 Uc1_Controlingless Analysis 31
4.4.1.1 Defining the Scope of the Use Case Model 31
4.4.1.2 Describing Use Case Behavior 34
4.4.1.2.1 Use Case Functional Flow 34
4.4.1.2.2 Use Case Scenarios 36
4.4.1.3 Formalizing Messages and Operations 39
4.4.1.4 Checking OpCon Consistency 40
4.4.1.5 Creating Port and Interfaces 42
4.4.1.6 Adding State-Based Behavior to the Use Case Model 44
4.4.1.7 Model Verification and Validation 46
4.4.2 Uc2_ControlEgress Analysis 48
4.4.3 Use Case Consistency Analysis 56
4.4.4 Merging Use Case Model Information 59
4.5 System Architectural Design 61
4.5.1 Defining the System Architecture 62
4.5.2 Allocating Operational Contracts to Subsystems 65
4.5.3 Defining Subsystem Interfaces 71
4.5.4 Creating Ports and Interfaces 77
4.5.5 Adding State-Based Behavior to the System Architecture Model 80
4.5.5.1 Extending the State-Based Behavior of the Actors 80
4.5.5.2 State-Based Behavior of Subsystem Blocks 81
4.5.6 System Architecture Model Verification and Validation 84
4.6 Subsystem Architectural Design 86
4.6.1 Expanding the Subsystem Specification 86
4.6.1.1 Subsystem Related System Use Cases 86
4.6.1.2 Subsystem Black-Box Use Case Scenarios 87

HANDOFF TO SUBSYSTEM DEVELOPMENT 90
The same system design steps exist in each systems engineering textbook:

<table>
<thead>
<tr>
<th>Systems Design Task</th>
<th>INCOSE Handbook (version 2a)</th>
<th>Blanchard, Fabrycky</th>
<th>Hatley/Pirbhai</th>
<th>Buede</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Analysis</td>
<td>Requirements Definition (Chapter 8)</td>
<td>Mission Requirements Analysis (“Systems Engineering Process in Life Cycle” Figure, Session 3)</td>
<td>Initial Problem Statement (Chapter 8)</td>
<td>Requirements (Chapter 6)</td>
</tr>
<tr>
<td>Functional Analysis or Decomposition</td>
<td>Functional Analysis &amp; Allocation (Chapter 9)</td>
<td>Functional Analysis (“Systems Engineering Process in Life Cycle” Figure, Session 3)</td>
<td>Modeling &amp; Building Upon The Known Pieces (Chapters 9,10)</td>
<td>Functional Architecture Development (Chapter 7)</td>
</tr>
<tr>
<td>Physical Architecture Development</td>
<td>Architecture Synthesis (Chapter 10)</td>
<td>Design Synthesis (“Systems Engineering Process in Life Cycle” Figure, Session 3)</td>
<td>Modeling &amp; Building Upon The Known Pieces (Chapters 9,10)</td>
<td>Physical Architecture Development (Chapter 8)</td>
</tr>
<tr>
<td>Mapping Functions To Physical Components</td>
<td>Functional Analysis &amp; Allocation (Chapter 9)</td>
<td></td>
<td>Modeling &amp; Building Upon The Known Pieces (Chapters 9,10)</td>
<td>Operational Architecture Development (Chapter 9)</td>
</tr>
<tr>
<td>Requirements Allocation</td>
<td>Functional Analysis &amp; Allocation (Chapter 9)</td>
<td>Requirements Allocation (“Systems Engineering Process in Life Cycle” Figure, Session 3)</td>
<td>Filling In The Blanks, Completion (Chapters 11, 12)</td>
<td>Interface Design (Chapter 10)</td>
</tr>
</tbody>
</table>
The same system design steps exist in each systems engineering standard:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Analysis</td>
<td>Requirements Analysis &amp; Validation (Section 4.2.1)</td>
<td>Requirements Analysis Section 5.5.2-5.5.4</td>
<td>Requirements Analysis &amp; Validation (Section 1 &amp; 2)</td>
<td>Requirements Definition Process Requirements (14-16)</td>
</tr>
<tr>
<td>Functional Analysis or Decomposition</td>
<td>Functional Analysis, Allocations, Validation or Logical Solution Representation Definition (Section 4.2.2)</td>
<td>Functional Analysis &amp; Verification Sections 5.5.4, 5.5.7</td>
<td>Functional Analysis &amp; Verification Sections 3 &amp; 4</td>
<td>Solution Definition Process Requirements (17)</td>
</tr>
<tr>
<td>Physical Architecture Development</td>
<td>Physical Solution Representation (Section 4.2.4)</td>
<td>Synthesis Section 5.5.4, 5.5.7</td>
<td>Physical Architecture Synthesis &amp; Verification (Section 5)</td>
<td>Solution Definition Process Requirements (18)</td>
</tr>
<tr>
<td>Mapping Functions To Physical Components</td>
<td>Physical Solution Representation (Section 4.2.4)</td>
<td>Synthesis Section 5.5.4, 5.5.7</td>
<td>Physical Architecture Synthesis &amp; Verification (Section 5)</td>
<td>Solution Definition Process Requirements (18)</td>
</tr>
<tr>
<td>Requirements Allocation</td>
<td>System Product Technical Requirements Analysis &amp; Validation (Section 4.2.3)</td>
<td>Synthesis Section 5.5.4, 5.5.7</td>
<td>Physical Architecture Synthesis &amp; Verification (Section 5)</td>
<td>Solution Definition Process Requirements (19)</td>
</tr>
</tbody>
</table>
Focus On
This Step…

4 CASE STUDY: SECURITY SYSTEM 24
4.1 Setting up a SysML Project 25
4.2 Security System Requirements 27
4.3 Requirements Analysis 29
    4.3.1 Defining System Use Cases 29
4.4 System Functional Analysis 31
    4.4.1 Uc1_Control ingress Analysis 31
        4.4.1.1 Defining the Scope of the Use Case Model 31
        4.4.1.2 Describing Use Case Behavior 34
            4.4.1.2.1 Use Case Functional Flow 34
            4.4.1.2.2 Use Case Scenarios 36
        4.4.1.3 Formalizing Messages and Operations 39
        4.4.1.4 Checking OpCon Consistency 40
        4.4.1.5 Creating Port and Interfaces 42
        4.4.1.6 Adding State-Based Behavior to the Use Case Model 44
        4.4.1.7 Model Verification and Validation 46
    4.4.2 Uc2_ControlEgress Analysis 48
    4.4.3 Use Case Consistency Analysis 56
    4.4.4 Merging Use Case Model Information 59
4.5 System Architectural Design 60
    4.5.1 Defining the System Architecture 62
        4.5.2 Allocating Operational Contracts to Subsystems 65
        4.5.3 Defining Subsystem Interfaces 71
        4.5.4 Creating Ports and Interfaces 77
        4.5.5 Adding State-Based Behavior to the System Architecture Model 80
            4.5.5.1 Extending the State-Based Behavior of the Actors 80
            4.5.5.2 State-Based Behavior of Subsystem Blocks 81
        4.5.6 System Architecture Model Verification and Validation 84
4.6 Subsystem Architectural Design 86
    4.6.1 Expanding the Subsystem Specification 86
        4.6.1.1 Subsystem Related System Use Cases 86
        4.6.1.2 Subsystem Black-Box Use Case Scenarios 87
5 HANDOFF TO SUBSYSTEM DEVELOPMENT 90
4.5 System Architectural Design

4.5.1 Defining the System Architecture

The first step in the system architectural design is the definition of the (functional or physical) subsystems. This is done in a block definition diagram.

In the case study, the security system is partitioned into the two subsystems: IO_Devices and Control.

- Open the block definition diagram BDD_SecuritySystemArchitecture in the SystemArchitecturePkg
- Drag&drop the actors from the ActorsPkg into the block definition diagram

-->

From Where Did This Come???

Someone Must **Conceive** Of A Sub-system Structure As An Input To The Harmony Process…

⚠️ Deselect code generation of the actor associations (ref. section 4.4.1)
• IBM/Telelogic Harmony

• Extending Harmony

• Architecture Process Overview

• System Design Process Manager

• SystemC Code Generation For HW

• Questions
We’ve added to Harmony an architecture process:

- Provides physical architectures for Harmony to allocate to
- Architectures created from corporate IP libraries
- Aggregates SysML performance annotations from models
- Analyzes architecture performance
- Iteration with Harmony & trade studies selects & optimizes
- Provides clear division of labor between teams

Extends Harmony to a “classic” system design methodology
**Feature Teams**
- Work down system physical hierarchy
- Execute Harmony process
- Define functional structure
- Define performance requirements
- Collectively own functional architecture

**Architecture Teams**
- Work across system features
- Execute architecture process
- Define physical structure
- Define performance capability
- Own physical architecture

**Orthogonal Feature Teams & Architecture Teams**
• IBM/Telelogic Harmony

• Extending Harmony

• Architecture Process Overview

• System Design Process Manager

• SystemC Code Generation For HW

• Questions
Architecture Process Overview

Architecture Process

Objectives, Priorities & Requirements Analysis
Functional Architecture Aggregation (from feature teams executing Harmony)
Physical Architecture Development
Operational Architecture Aggregation (from feature teams executing Harmony)
Performance Analyses
Trade Studies & Iteration
Requirements Allocation

Suggestion Box:
For tool enhancements that would be useful
Architecture Process Overview

Objectives, Priorities & Requirements Analysis:

• Define TPMs (Technical Performance Measures) (e.g. reliability >= .99999)
  • “Measuring stick” to gauge merit of architectures (see system design process manager)
  • Define business objectives (e.g. use DSP)

Functional Architecture Aggregation:

• Collect FunctionalAnalysisPkgs from feature teams (w/performance annotations)
• Meld into composite SysML model of functional architecture
• Annotate with the most stringent feature performance requirements

Suggestion Box:

Enhance Gateway to allow transfer of numerical quantities from attributes in DOORs
Physical Architecture Development:

- **Select physical components from corporate IP library to:**
  - Implement functions and functional performance requirements in functional architecture
  - Satisfy external interface requirements
  - Achieve architectural *performance* requirements (see system design process manager)

- Assemble components into set of physical architecture candidates

- Capture as internal & external block diagrams in SystemArchitecturePkgs
  - See Harmony desk book version 2.0 step 4.5.1

**Suggestion Box:**

Rhapsody allows population of block “requirement” attributes with requirements values from Gateway

This would enable automatic comparison of calculated architecture performance to requirements
Operational Architecture Aggregation:

- Collect SystemArchitecturePkgs from feature teams (w/performance annotations)
- Meld into set of composite SysML models of operational architectures
- Annotate with most stringent feature performance requirements
Performance Analyses:

- Designs for performance already included in physical architecture
- Analyze performance aspects of operational architectures
  - System Initialization
  - Safety & Reliability
  - Real-Time
  - Maintainability/Serviceability
  - Power Management
  - Synchronization
  - Life-Cycle Cost
  - Manufacturability
  - Environmental
  - Security
  - Scalability
  - Testability
  - Technical Risk

- Requires sharing of models by suite of tools (see system design process manager)
- Results drive trade studies

Suggestion Box:

Expand performance analyses via 3rd party tool partnerships (like you have with Tri-Pac & RapidRMA)

Put math execution behind parametric diagrams
Trade Studies & Iteration:

- **Formal/Informal decisions** (e.g. components, performance designs, final architecture)

- **Quantitative (objective) or qualitative (subjective):**
  - Quantitative preferred, qualitative sometimes necessary
  - Quantitative combines TPMs, priorities & performance analyses from earlier steps

- **TPM charts “close the loop” on entire process** (drives architectural convergence)

**Suggestion Box:**

Capture analysis results from other modeling tools

Compare analyses results to requirements via TPM charts

Feed analysis results & priorities into trade study equation, calculate result by parametric diagram execution
Requirements Allocation:

- Performance requirements & *tolerances* allocated to each component
- Performance model hand-offs (when possible, just as with Harmony)

**Suggestion:**
Enhance Gateway & Rhapsody to auto-populate sub-system block requirement attributes via allocation equations in parametric diagram.
• IBM/Telelogic Harmony

• Extending Harmony

• Architecture Process Overview

• **System Design Process Manager**

• SystemC Code Generation For HW

• Questions
System Design Process Management Includes:

- **Capture and revision control of requirements, priorities & verification suites**
  - What do we want & how will we know we have it?

- **Capture and revision control of design content in multiple domains**
  - What is it? What does it do? How is it structured?

- **Capture and revision control of performance analyses results**
  - Is it what we ask for? How does it perform?

- **Optimization engine to assist with architectural convergence**
  - If it’s not what we ask for, how do we modify it?

- **Automated flow-down of requirements & model hand-offs in multiple domains**
  - How do we insure continuity of purpose?

- **Synthesis wherever possible**
  - Build it for us

- **Report generation**
  - Give us documentation
System Design Process Manager

Conceptual Drawing:

Physical, Functional & Stimulus Model Exchange Bus

System Requirements, Verification Tests/Results Vault

Optimization Engine (Inputs: TPM Requirements & TPM equations, Outputs: TPM equation coefficients)

Decision Engine (Inputs: Verification Results & Optimized TPM values, Outputs: Best Design)
Usage Scenario:

1. Capture requirements, TPMs and priorities in control group vault, transfer to design group tools
2. Develop verification tests in design group tools, transfer to control group vault
3. Capture, analyze and verify architectures in design group tools, exchange physical, functional or stimulus elements as necessary (XMI, AP-233, etc.)
4. TPM equations & results from design group tools transferred to control group optimization engine
5. Optimization engine provide local optimization of design candidates, TPM equations transferred to design group tools to back-annotate design files
6. Optimized TPM measurements are transferred to control group decision engine
7. Trade studies executed using control group decision engine to select best design
8. Requirements, model and verification suites allocated and handed to subsystem teams
9. Subsystems synthesized
10. Repeat process for subsystems
IBM/Telelogic, Cadence, & Medtronic are realizing this vision!

System Design Process Manager

More Information: mark.a.stockburger@medtronic.com
Agenda

• IBM/Telelogic Harmony
• Extending Harmony
• Architecture Process Overview
• System Design Process Manager
• SystemC Code Generation For HW
• Questions
SysML, SystemC, RTL & Tool (Revision 1)

Current SystemC & RTL
Current SysML & Tool-generated SystemC
UML Threads Not Yet Pursued
Legacy System Test Benches

Abstraction
Physical
Language
SysML, SystemC, RTL & Tool (Revision 2)

Current SystemC & RTL
Current SysML & Tool-generated SystemC
UML Threads Not Yet Pursued
Legacy System Test Benches
Future SysML & Tool-generated SystemC
SysML, SystemC, & FW (Current)

Current SystemC
Current SysML & Tool-generated SystemC
UML Threads Not Yet Pursued
Legacy System Test Benches
Future SysML & Tool-generated SystemC
Current FW C
SysML, SystemC, & FW (Future)

Current SystemC & RTL
Current SysML & Tool-generated SystemC
UML Threads Not Yet Pursued
Legacy System Actors (Test Benches)
Future SysML & Generated SystemC
Current FW Code
Future FW UML & C
• IBM/Telelogic Harmony

• Extending Harmony

• Architecture Process Overview

• Architecture Process Tool Suite Manager

• SystemC Code Generation For HW

• Questions