Streamlining the Development of Complex Systems through Model-based Systems Engineering

What is Systems Engineering?

- **Systems Engineering** is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem.
Agenda

- Model-based systems engineering in a model-driven development lifecycle
  - Fundamentals of model-based systems engineering
    - Essential SysML artifacts
    - Service request-driven modeling approach
  - Task flow and work products in Rational Harmony™ for Systems Engineering
  - Deploying MbSE with Rational Rhapsody®
  - Documentation of Rational Harmony™ for Systems Engineering

Document driven development of embedded systems
The "Throw-it-over-the-Fence" approach
Rational Workbench for Systems and Software Engineering
Built on a core solution set

Use modeling to validate requirements, architecture and design throughout the development process

Rational Rhapsody
Rational Quality Manager
Rational Team Concert

Manage all system requirements with full traceability across the lifecycle
Achieve "quality by design" with an integrated, automated testing process

COLLABORATE AUTOMATE REPORT

Rational Workbench for Systems and Software Engineering

Integrated system / software development process
Domains involved in the different phases of the model-driven development

Systems Engineering Software Engineering Electrical/Mechanical Engineering Test Engineering

Change Request

Requirements Analysis System Function Analysis Design Synthesis System Architecture

System Validation Plan Test Strategy

Embedded IT Development

Service Acceptance Sub-Systems Integration & Test HW Implementation & Test HW Build

© 2011 IBM Corporation
Agenda

- Model-based systems engineering in a model-driven development lifecycle
- **Fundamentals of model-based systems engineering**
  - Essential SysML artifacts
  - Service request-driven modeling approach
- Task flow and work products in Rational Harmony™ for Systems Engineering
- Deploying MLESE with Rational Rhapsody®
- Documentation of Rational Harmony™ for Systems Engineering
SysML artifacts in Rational Harmony™ for Systems Engineering
Capturing the static view

Defines structural elements (Blocks) and their relationship

Defines the realization of system structure

Defines the parametric relationship between system properties

SysML artifacts in Rational Harmony™ for Systems Engineering
Capturing system behavior (example use case model)

• Defines system scope
• Groups requirements into Use Cases ("Table of Contents")

Defines functional flow ("Storyboard") of the Use Case

Defines interactions with the environment

Aggregates AD- and SD-info and adds state-based behavior
Agenda

- Model-based systems engineering in a model-driven development lifecycle
- Fundamentals of model-based systems engineering
  - Essential SysML artifacts
  - Service request-driven modeling approach
- Task flow and work products in Rational Harmony™ for Systems Engineering
- Deploying MbSE with Rational Rhapsody®
- Documentation of Rational Harmony™ for Systems Engineering

Service request-driven modeling approach

Communication described through operational contracts (OpCon), i.e.
- asynchronous service requests via SysML Standard Ports followed by
  - provided services at the receiving part (state/mode changes or operations)
Agenda

- Model-based systems engineering in a model-driven development lifecycle
- Fundamentals of model-based systems engineering
  - Essential SysML artifacts
  - Service request-driven modeling approach
- Task flow and work products in Rational Harmony™ for Systems Engineering
- Deploying MbSE with Rational Rhapsody®
- Documentation of Rational Harmony™ for Systems Engineering

Key objectives of the Rational Harmony™ for Systems Engineering workflow

- Identify / derive required system functionality
- Identify associated system states and operational modes
- Allocate required system functionality to a system architecture taking into account non-functional aspects of the requirements
Rational Harmony™ for Systems Engineering

Requirements analysis

In the requirements analysis phase, the focus is on the analysis of the process inputs. Stakeholder requirements are translated into system requirements that define:
- what the system must do (functional requirements) and
- how well it must perform (quality of service requirements).

Once the requirements are sufficiently understood they are grouped into Use Cases.

System functional analysis

In the system functional analysis phase, the focus is on the translation of the functional requirements into a coherent description of system operations.

Each use case of an iteration is translated into a model and the underlying requirements verified and validated through model execution.
The System-Level Interface Control Document (ICD) defines the logical (= functional) interfaces between the (black-box) system and its actors and is the aggregate of all use case blocks interfaces. This ICD is the basis for the later system-level (black-box) test definition.

In the design synthesis phase, the focus is on the development of a system architecture capable of performing the required operations within the limits of the prescribed performance constraints.
The objective of the architectural analysis phase is to elaborate the optimum design concept based upon a set of criteria (e.g., Measures of Effectiveness, MoEs) that are weighted according to their relative importance.

The focus of the architectural design phase is on the allocation of the functional requirements (system operations) and non-functional requirements to an architectural structure. The architectural design is performed incrementally for each use case of an iteration by transitioning from the black-box view to the white-box view—also referred to as use case realization.
Systems engineering handoff to the subsequent system development

In a model-driven development the key artifact of the handoff from systems engineering to the subsequent system development is the baseline executable model.

This model is the repository from which specification documents (e.g. HW/SW requirements specifications, ICDs, ...) are generated.

Scope and content of the hand-off is dependent on the characteristics of the project and the organizational structure systems engineering is embedded.
Systems engineering handoff to the subsequent system development

The hand-off packages typically are composed of baselined executable CI model(s) which contain

- The definition of allocated operations including their links to the associated system functional and performance requirements
- The definition of the associated state-based behavior, captured in a statechart diagram
- The allocated and linked non-functional requirements
- The definition of ports and logical interfaces
- Test scenarios, derived from system-level use case scenarios

Characteristics of Rational Harmony™ for Systems Engineering

- Tool-independent best practices for model-based systems engineering, covering
  - Requirements analysis
  - System functional analysis
  - Design synthesis
    (architectural analysis and architectural design)
  - Incorporation of design iterations into the overall project flow.
- Use case driven requirements capture and analysis
- Early and frequent requirements verification and validation via model execution.
- Requirements tracing throughout the entire process
- Test aspects supported through scenario-driven workflow
- Seamless transition from systems engineering to software engineering by using the UML™/SysML™ as paradigm independent modeling language
### Agenda

- Model-based systems engineering in a model-driven development lifecycle
- Fundamentals of model-based systems engineering
  - Essential SysML artifacts
  - Service request-driven modeling approach
- Task flow and work products in Rational Harmony™ for Systems Engineering
- Deploying MbSE with Rational Rhapsody®
- Documentation of Rational Harmony™ for Systems Engineering

### Approach for enterprise process adoption

*Top-down (e.g., Capability Maturity Model Integration [CMMI])*

- **Process engineers**
  - Standard processes
  - Organization-wide standard process and tools
  - Coaches
  - Pilot projects:
  - Live projects:
    - Project-specific process and tools

- **Tool specialists**
  - Supporting tools

- **System admins**
  - Infrastructure

*Bottom-up (e.g., “grassroots”)*
Deploying model-based systems engineering
Project specific model-based Systems Engineering Handbook

 MbSE Handbook
Standardizing the MbSE workflow and Rhapsody tool usage
The Alternative: MbSE Handbook in RMC

Deploying model-based systems engineering

The need for modeling guidelines
Software and Systems Engineering | Rational

MbSE Handbook
Standardizing the usage of the modeling language

Support of the MbSE workflow through the Rhapsody SE-Toolkit
1.10 Create New Scenario from Activity Diagram

Creates a sequence diagram from selected actions in an activity diagram.
If the source is a single action then the user will be asked to choose a path each time a condition connector is encountered.

Support of the MbSE workflow through the Rhapsody SE-Toolkit

Deploying model-based systems engineering
Managing the modeling activities: Project plan

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Analysis</td>
<td></td>
</tr>
<tr>
<td>Define System Architecture</td>
<td>1.2 days</td>
</tr>
<tr>
<td>Define Non-functional Analysis</td>
<td>1.4 days</td>
</tr>
<tr>
<td>Define Use Cases</td>
<td>1.6 days</td>
</tr>
<tr>
<td>Create System Architecture</td>
<td>2.1 days</td>
</tr>
<tr>
<td>Define System Non-functional</td>
<td>3.2 days</td>
</tr>
<tr>
<td>Define System Functional</td>
<td>3.4 days</td>
</tr>
<tr>
<td>Define System Properties</td>
<td>3.6 days</td>
</tr>
<tr>
<td>Define System Test Plan</td>
<td>3.8 days</td>
</tr>
<tr>
<td>Define System Pre-Release Plan</td>
<td>4.0 days</td>
</tr>
<tr>
<td>Define System Post-Release Plan</td>
<td>4.2 days</td>
</tr>
<tr>
<td>Define System Release</td>
<td>4.4 days</td>
</tr>
<tr>
<td>Define System Quality</td>
<td>4.6 days</td>
</tr>
<tr>
<td>Define System Deployment</td>
<td>4.8 days</td>
</tr>
<tr>
<td>Define System Operation</td>
<td>5.0 days</td>
</tr>
<tr>
<td>Define System Security</td>
<td>5.2 days</td>
</tr>
<tr>
<td>Define System Maintenance</td>
<td>5.4 days</td>
</tr>
<tr>
<td>Define System Operations</td>
<td>5.6 days</td>
</tr>
</tbody>
</table>

© 2011 IBM Corporation
Agenda

- Model-based systems engineering in a model-driven development lifecycle
- Fundamentals of model-based systems engineering
  - Essential SysML artifacts
  - Service request-driven modeling approach
- Task flow and work products in Rational Harmony™ for Systems Engineering
- Deploying MbSE with Rational Rhapsody®
- Documentation of Rational Harmony™ for Systems Engineering
Appendix

Case study: Security System

Use Case Diagram: System Use Cases

Requirements Diagram: Visualization of Links between Stakeholder Requirements and System Requirements
Case study
Definition of the use case functional flow (“storyboard”)

Use Case Black-Box Activity Diagram of Uc1ControlEntry

Case study
Derivation of a use case scenario from a use case black-box activity diagram
Case study
Internal block diagram IBD_Uc1_ControlEntry with ports and interfaces

© 2011 IBM Corporation

Case study
State-based behavior of use case Uc1_ControlEntry

© 2011 IBM Corporation
Case study
Model verification and validation through model execution (Rational Rhapsody tool)

Animated Statechart Diagram (Uc1ControlEntry)

Animated Sequence Diagram (Uc1SecSys)

Case study – Uc1 (Control Entry) realization
Allocation of system-level operations to parts (subsystems)

Selected system architecture captured in a Block Definition Diagram

Uc1ControlEntry White-Box Activity Diagram
Case study – Uc1 realization

Derivation of white-box scenarios from the use case white-box activity diagram

Uc1ControlEntry White-Box Activity Diagram

Case study – Uc1 realization

Allocation of system-level operations to parts and definition of ports and interfaces

© 2011 IBM Corporation
Case study – Uc1 realization
Derivation of state-based behavior

Case study
Uc1 realization
Model verification through model execution

Animated Sequence Diagram (Nominal Flow)
Case study – Uc2 (Control Exit) realization
Allocation of system-level operations to parts, definition of ports and interfaces
derivation of state-based behavior, model verification through model execution

Case study – Integrated use case realization
Case study
Integrated System Architecture verification through model execution

Animated Sequence Diagram (Uc1, Uc2 Nominal Flows)