Deploying Model-Based Systems Engineering with IBM® Rational® Rhapsody®

Sudhir Rao

Practice Manager,
INDIA Lab Services – Rational Brand
SudhirRao@in.ibm.com

IBM Software
Innovate2011
The Premier Event for Software and Systems Innovation

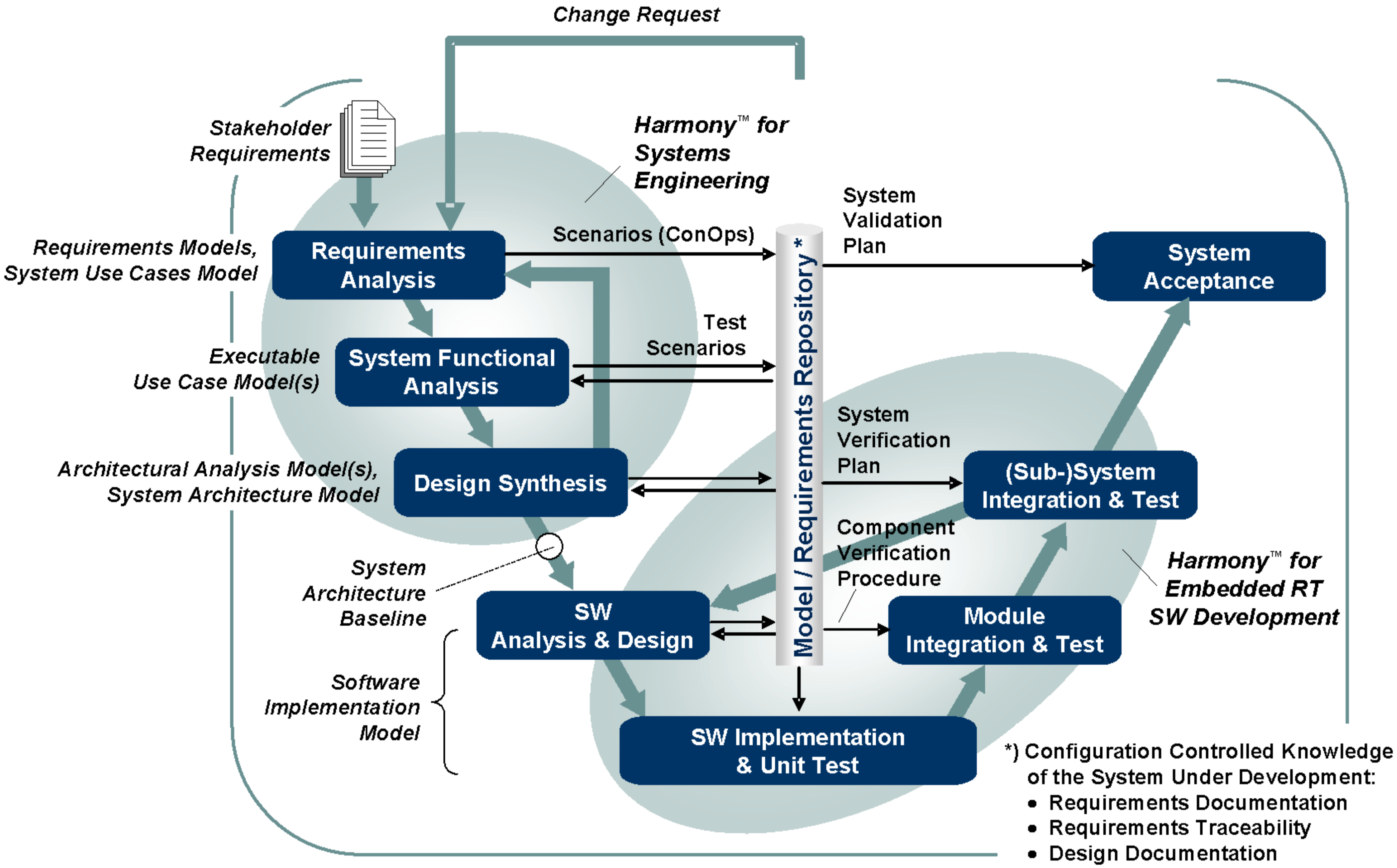
Software. Everyware.

Aug 11th 2011, Delhi

Agenda

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

Integrated system / embedded software development
Model-driven development


Change Request
Stakeholder Requirements
Harmony™ for Systems Engineering
Requirements Models, System Use Cases Model
Requirements Analysis
Scenarios (ConOps)
System Validation Plan
System Acceptance
Executable Use Case Model(s)
System Functional Analysis
Test Scenarios
Model / Requirements Repository*
Architectural Analysis Model(s), System Architecture Model
Design Synthesis
System Verification Plan
(Sub-)System Integration & Test
Component Verification Procedure
System Architecture Baseline
SW Analysis & Design
Module Integration & Test
Harmony™ for Embedded RT SW Development
Software Implementation Model
SW Implementation & Unit Test

*) Configuration Controlled Knowledge of the System Under Development:

- Requirements Documentation
- Requirements Traceability
- Design Documentation
- Test Definitions

Agenda

§ Model-based systems engineering in a model-driven development lifecycle

- **Fundamentals of the Rational model-based systems engineering approach**
  - Essential SysML artifacts
  - Service request driven modeling approach

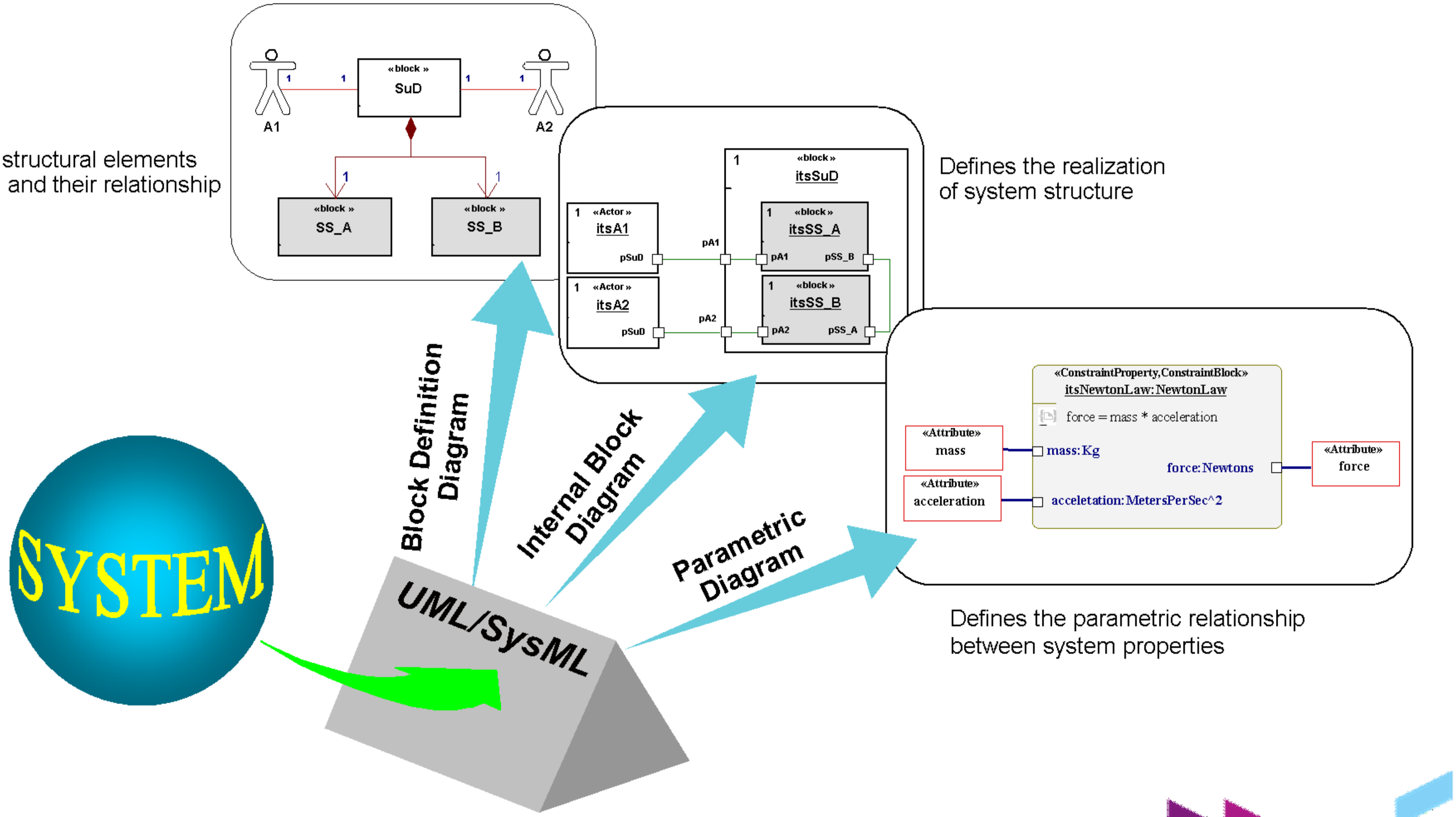
§ Task flow in Rational *Harmony™ for Systems Engineering*

§ Deploying MbSE 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*

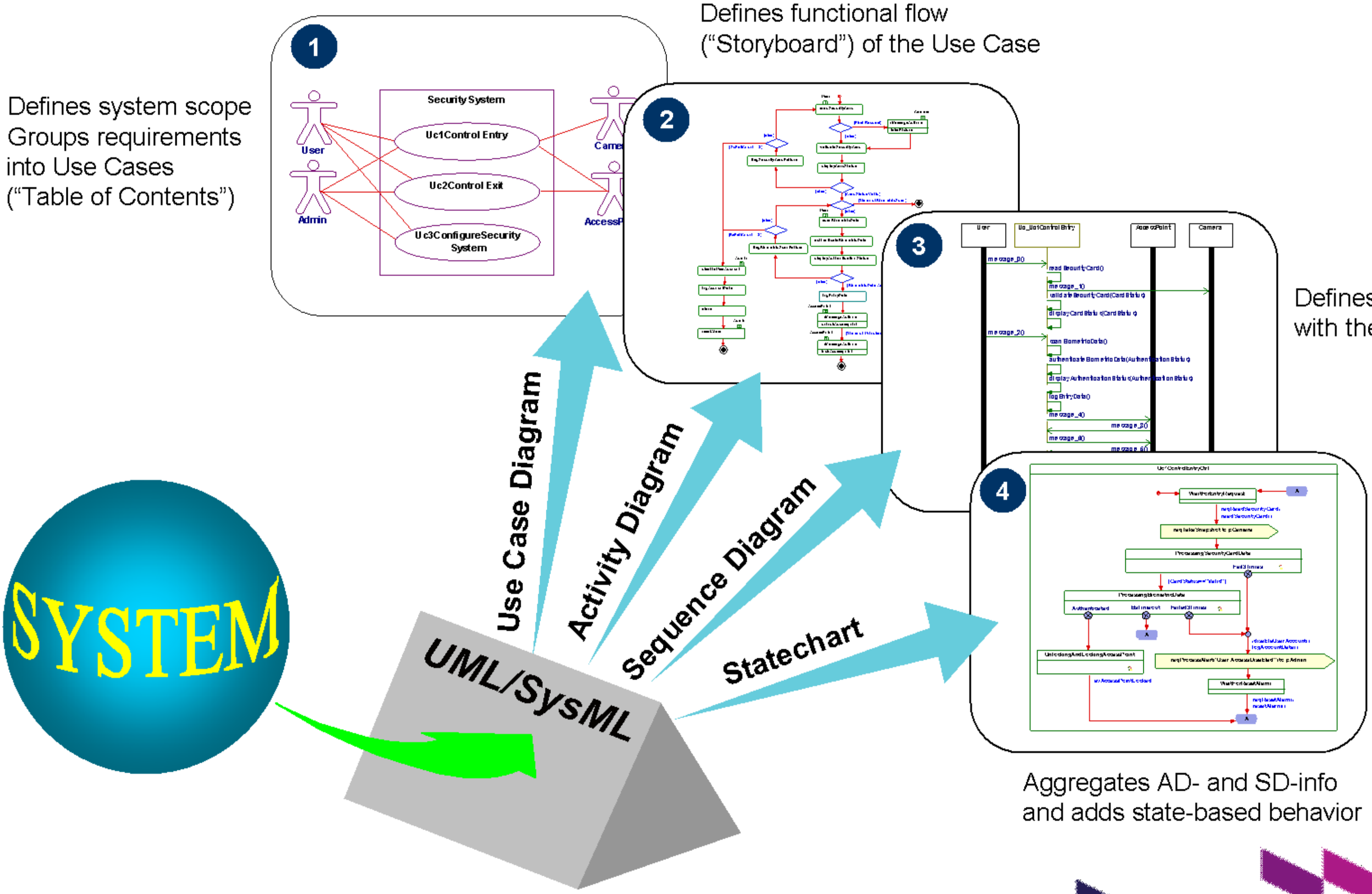
Capturing system behavior (example use case model)

- 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

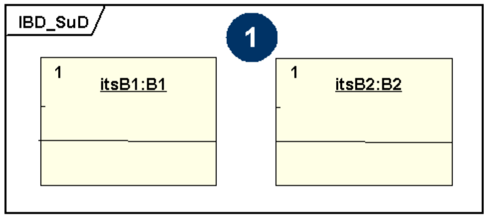


Service request driven modeling approach

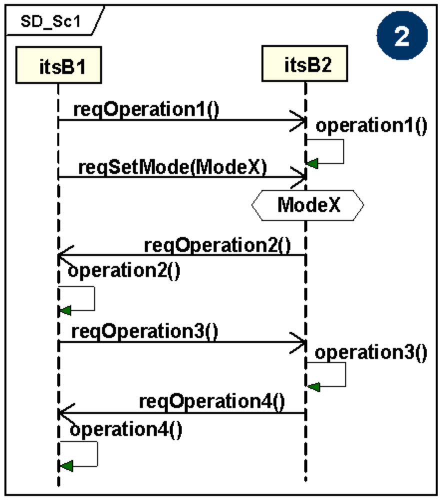
Communication described through *operational contracts* (*OpCon*), i.e.

- asynchronous *service requests* via *SysML Standard Ports* followed by
- provided services at the receiving part (state/mode changes or operations)

Define Communication Nodes

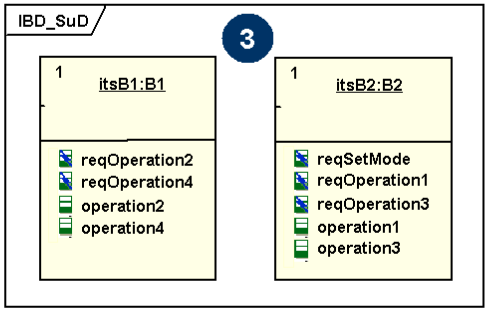


Describe Inter-Nodal Communication *)

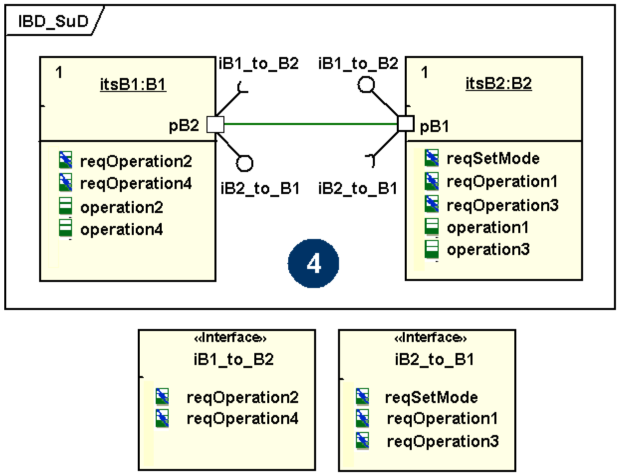


*) e.g. derived from Activity Diagram

Allocate Service Requests and Operations



Create Ports and Interfaces



Agenda

§ Model-based systems engineering in a model-driven development lifecycle

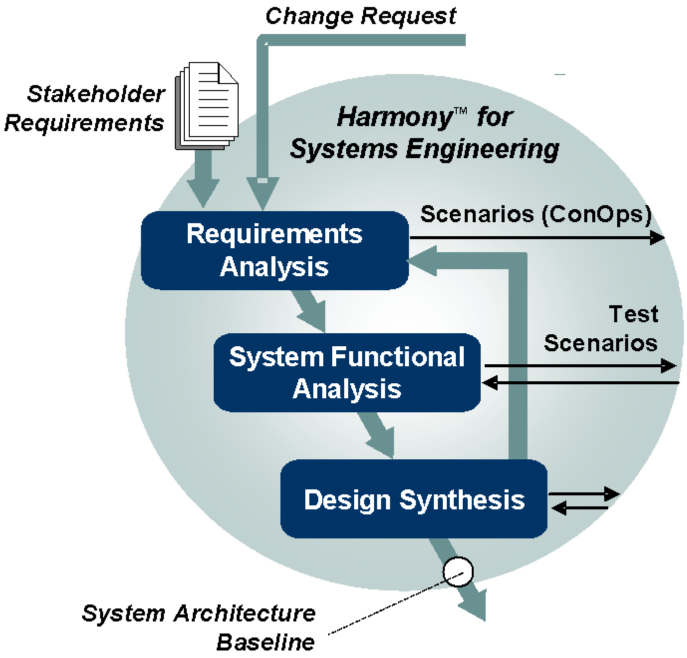
- Fundamentals of the Rational model-based systems engineering approach
  - Essential SysML artifacts
  - Service request driven modeling approach

§ **Task flow in Rational *Harmony*™ *for Systems Engineering***

§ Deploying MbSE with Rational® *Rhapsody*®

§ Documentation of Rational *Harmony™ for Systems Engineering*

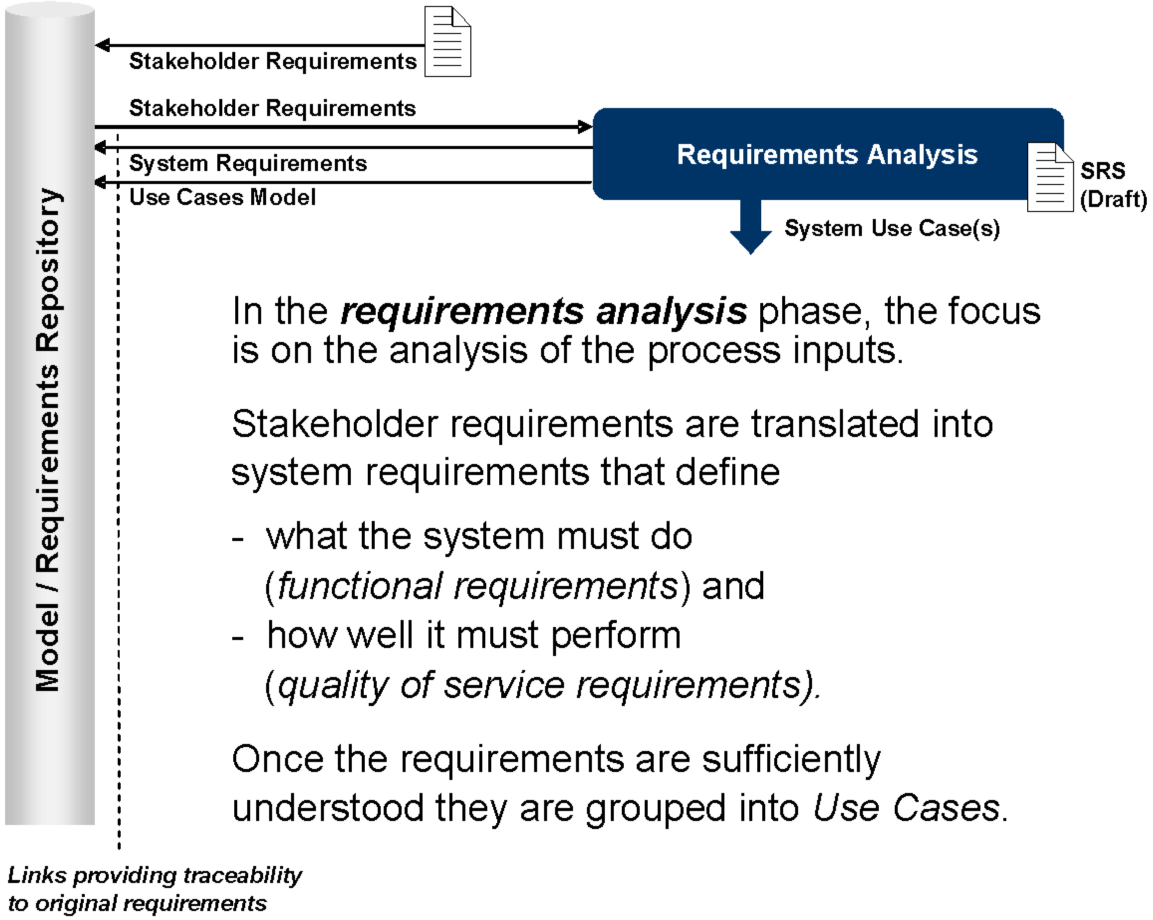
Key objectives of the Rational *Harmony™ for Systems Engineering* workflow



- 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.

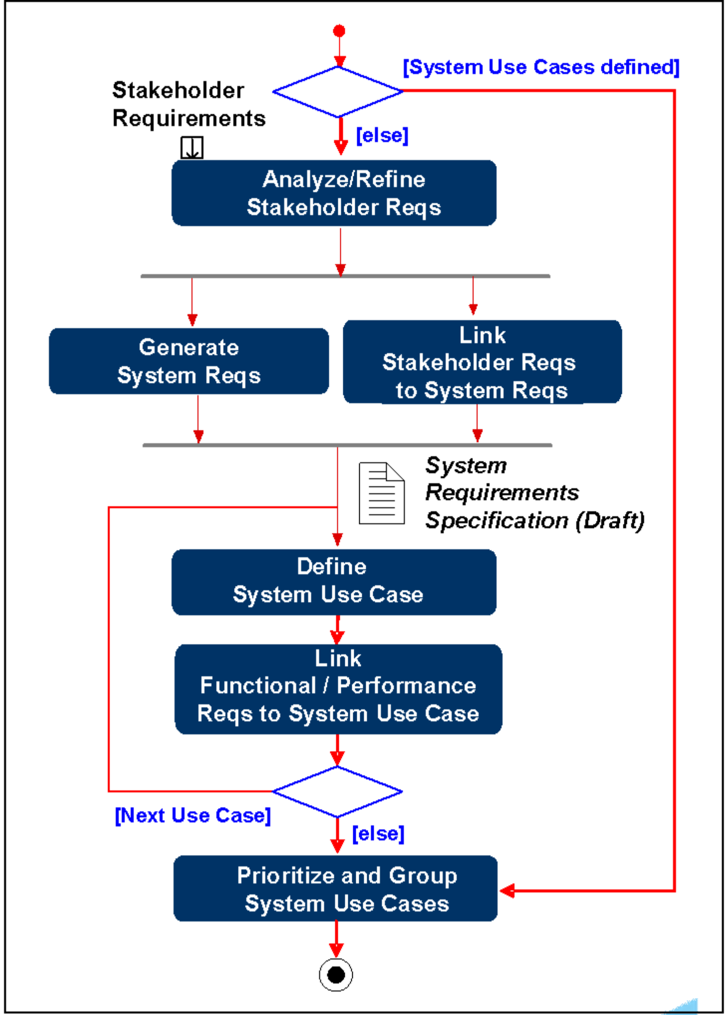
Stakeholder requirements are translated into system requirements that define

- 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*.

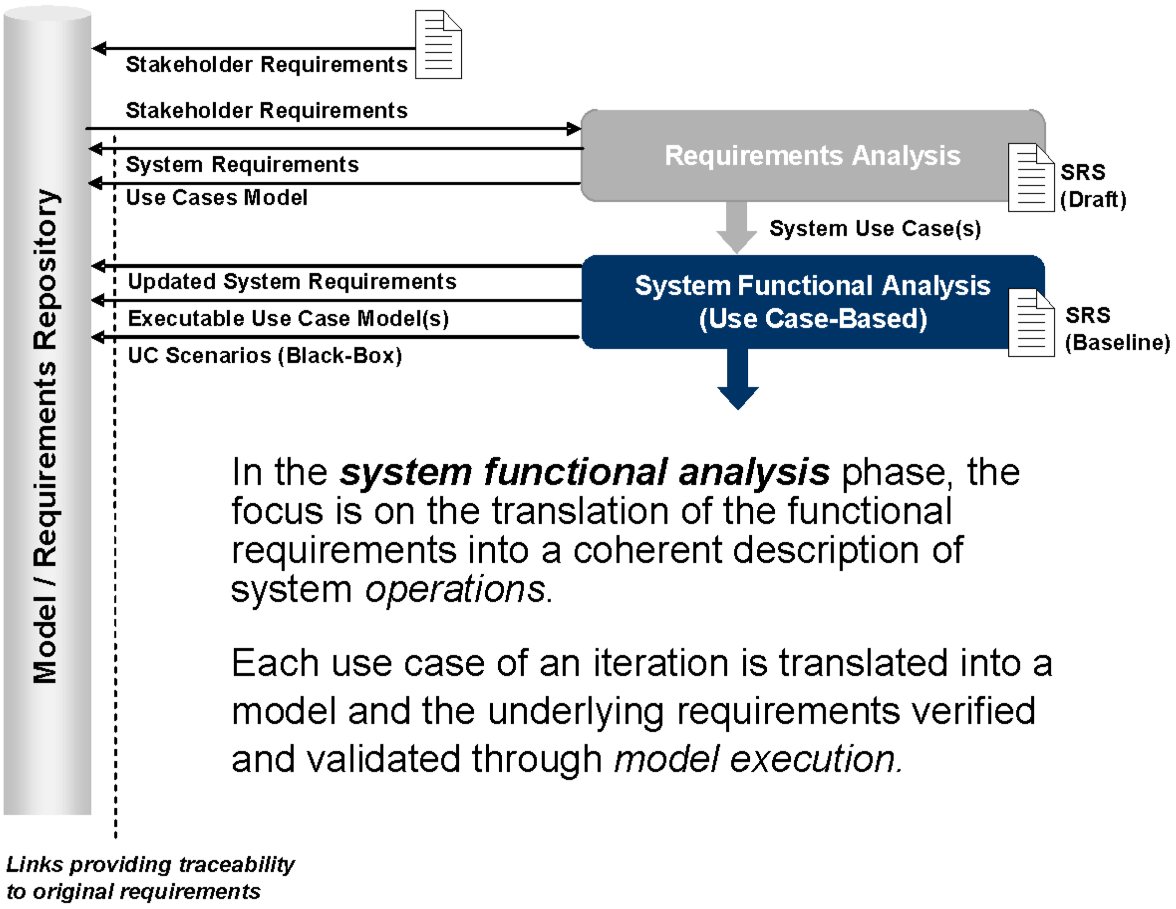
Links providing traceability to original requirements

Requirements Analysis Task Flow


[System Use Cases defined]
Stakeholder Requirements
[else]
Analyze/Refine Stakeholder Reqs
Generate System Reqs
Link Stakeholder Reqs to System Reqs
System Requirements Specification (Draft)
Define System Use Case
Link Functional / Performance Reqs to System Use Case
[Next Use Case]
[else]
Prioritize and Group System Use Cases

Rational *Harmony™ for Systems Engineering*

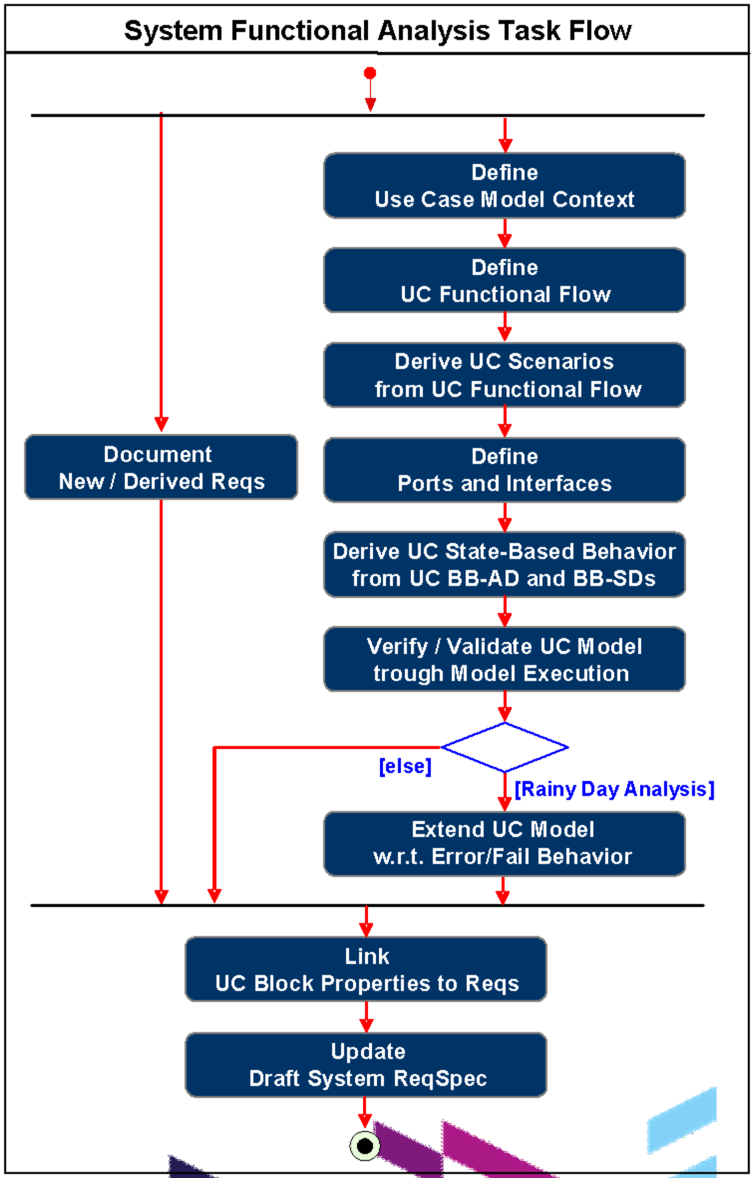
System functional analysis



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

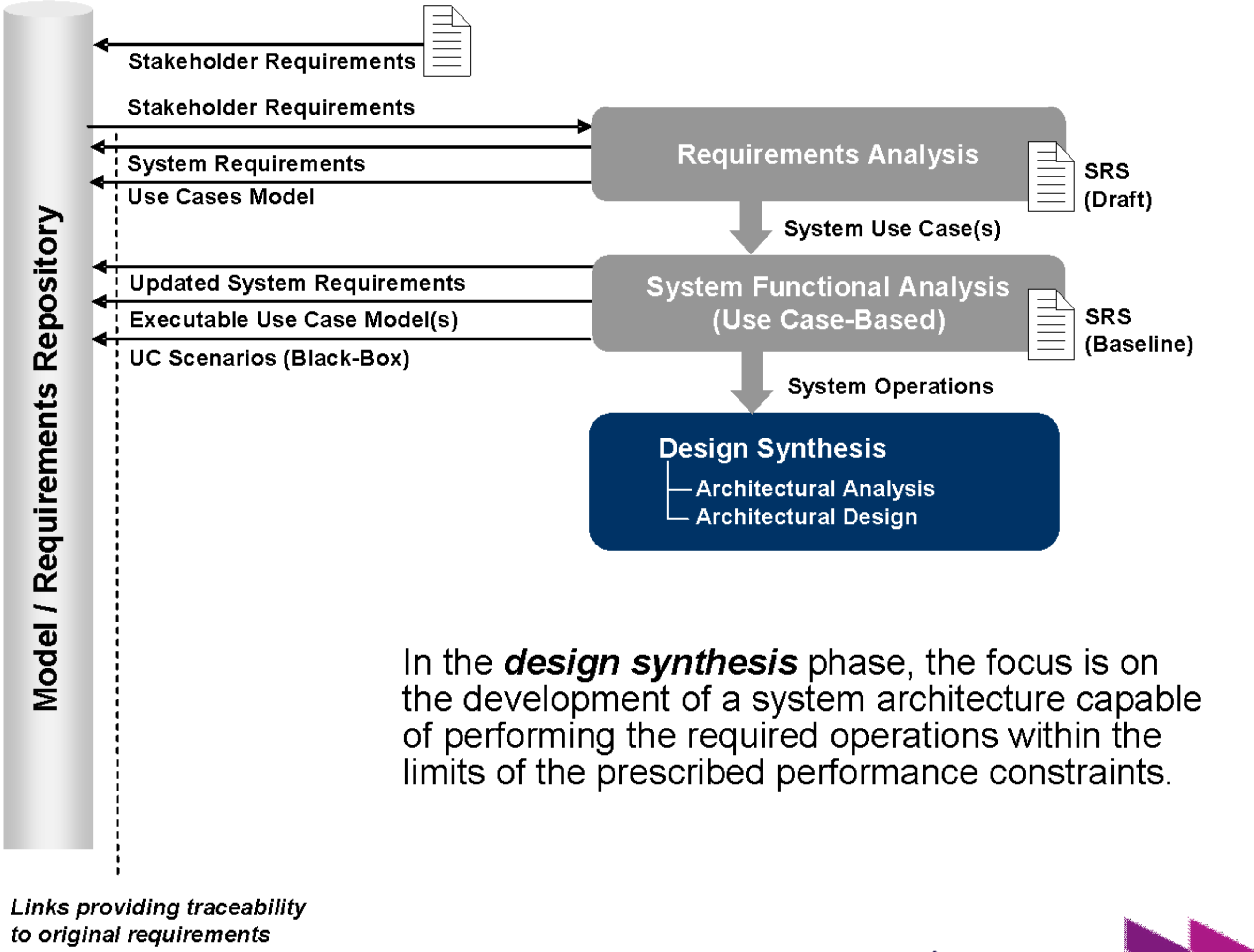
Each use case of an iteration is translated into a model and the underlying requirements verified and validated through *model execution.*

Links providing traceability to original requirements



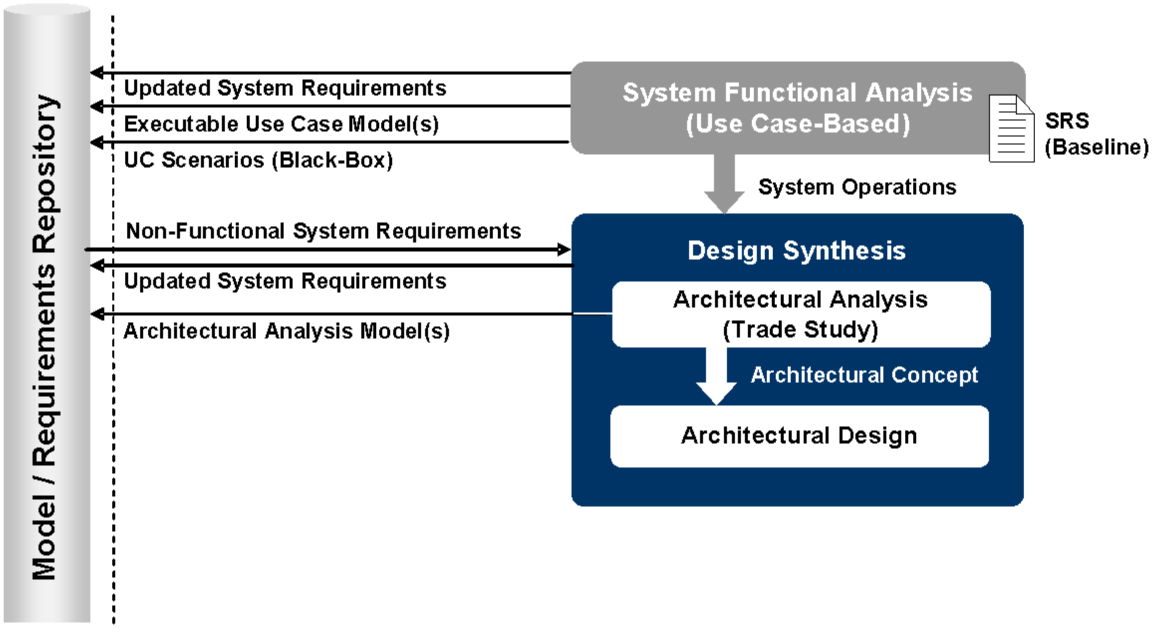
Rational *Harmony*™ *for Systems Engineering*

Design synthesis



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.

Design synthesis
Architectural analysis



Model / Requirements Repository

Updated System Requirements

Executable Use Case Model(s)

UC Scenarios (Black-Box)

System Functional Analysis (Use Case-Based)

SRS (Baseline)

System Operations

Non-Functional System Requirements

Updated System Requirements

Architectural Analysis Model(s)

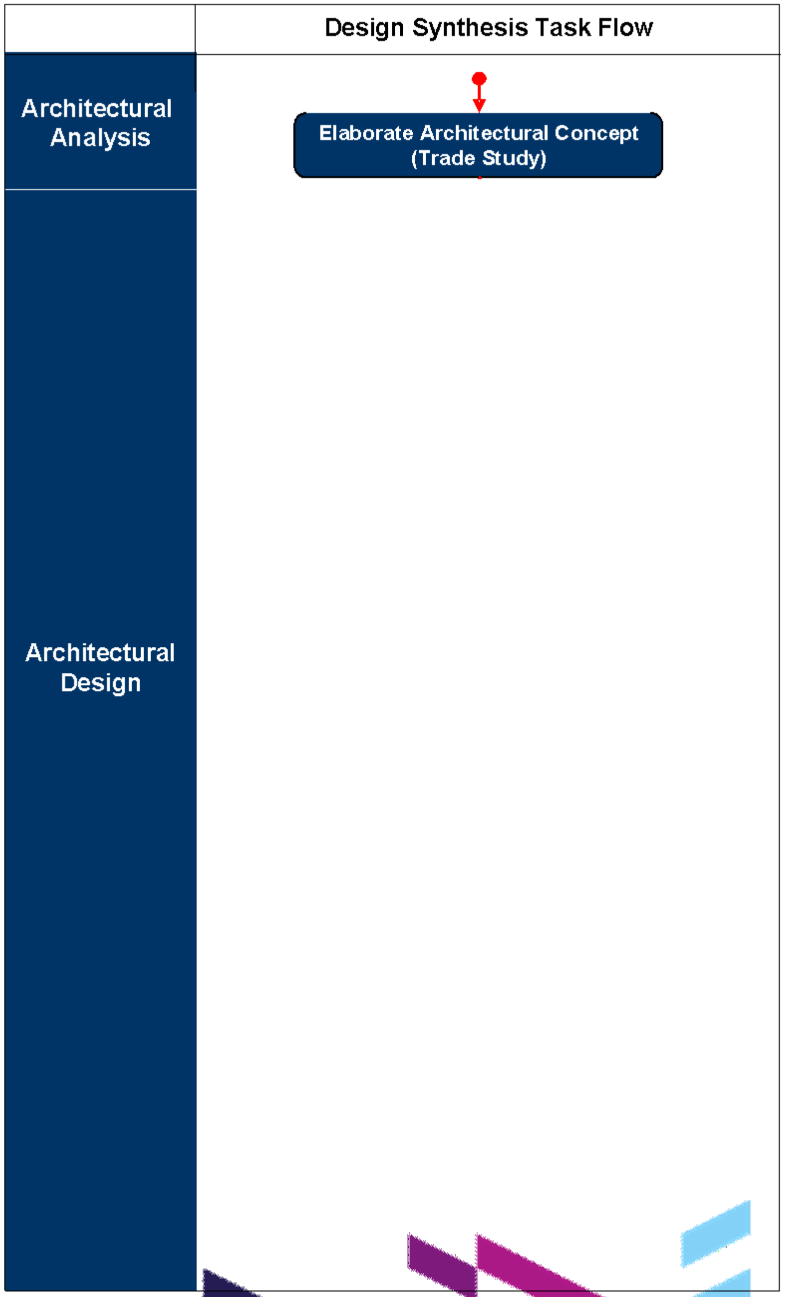
Design Synthesis

Architectural Analysis (Trade Study)

Architectural Concept

Architectural Design

Links providing traceability to original requirements



Design Synthesis Task Flow

Architectural Analysis

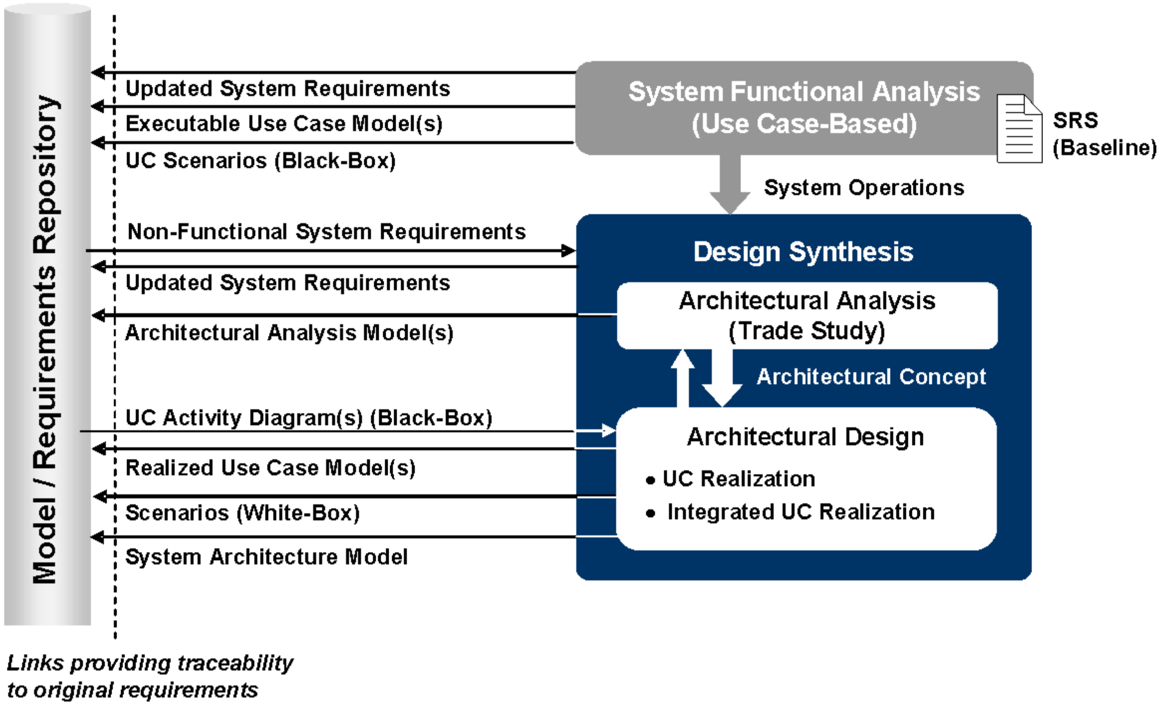
Elaborate Architectural Concept (Trade Study)

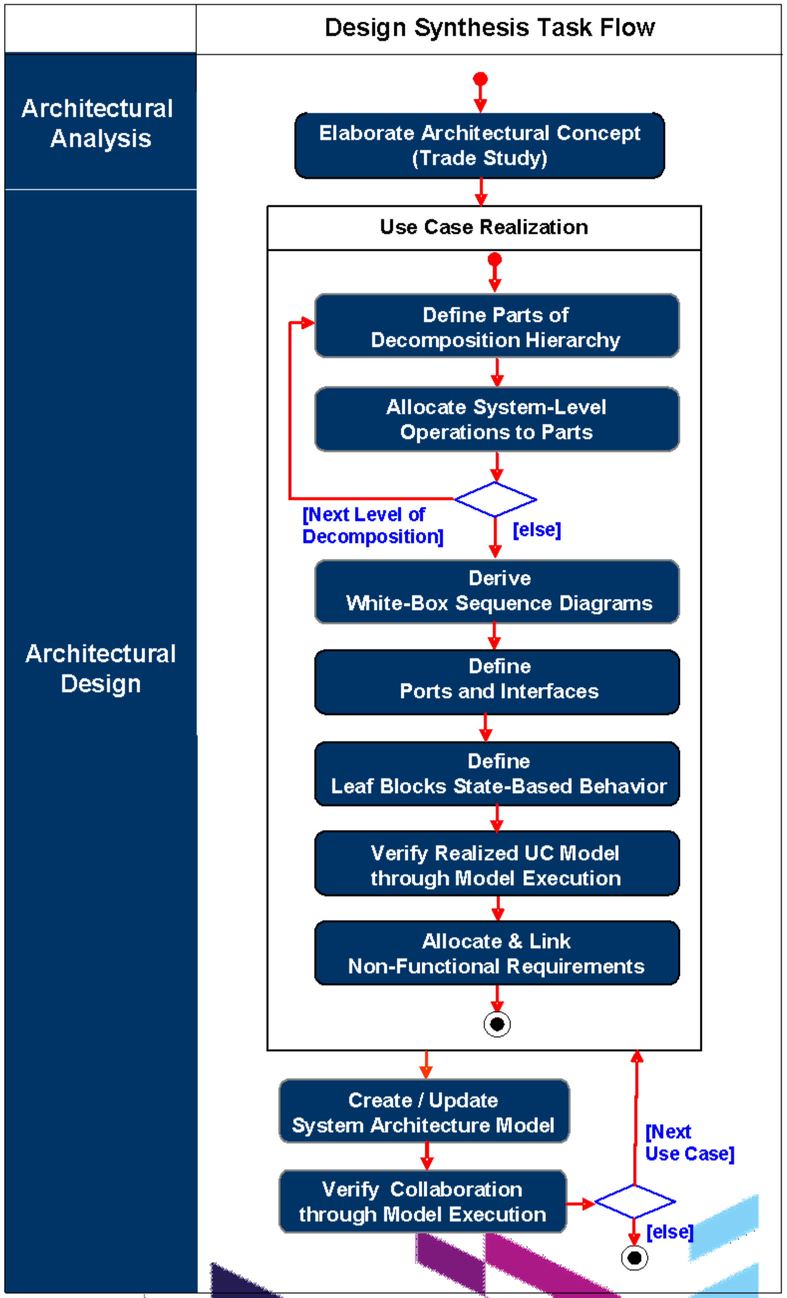
Architectural Design

The objective 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.

Design synthesis

Architectural design


Model / Requirements Repository
Updated System Requirements
Executable Use Case Model(s)
UC Scenarios (Black-Box)
System Functional Analysis
(Use Case-Based)
SRS
(Baseline)
System Operations
Non-Functional System Requirements
Updated System Requirements
Architectural Analysis Model(s)
Design Synthesis
Architectural Analysis
(Trade Study)
Architectural Concept
UC Activity Diagram(s) (Black-Box)
Realized Use Case Model(s)
Scenarios (White-Box)
System Architecture Model
Architectural Design
• UC Realization
• Integrated UC Realization
Links providing traceability
to original requirements



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***.

Agenda

§ Model-based systems engineering in a model-driven development lifecycle

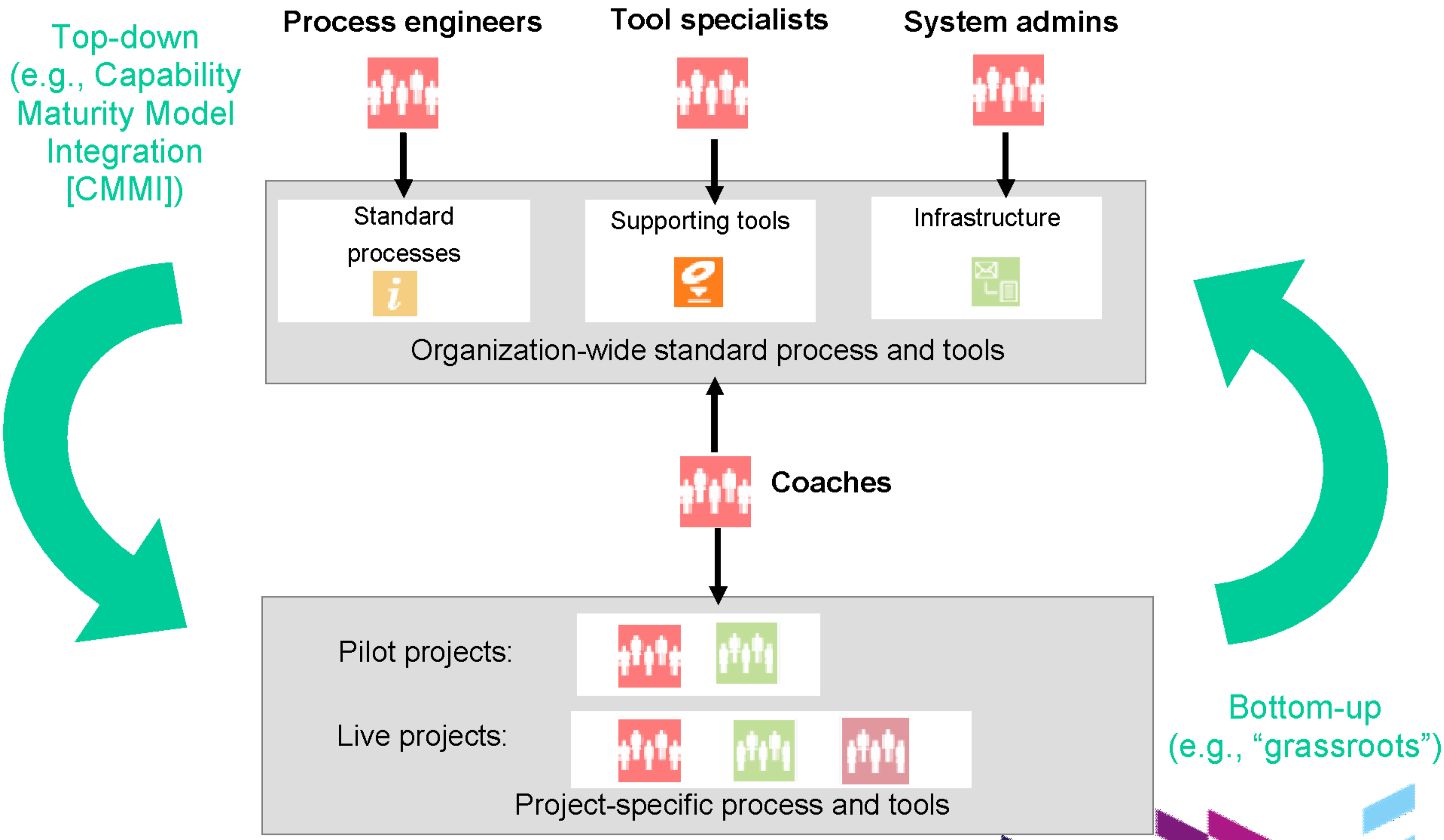
- Fundamentals of the Rational model-based systems engineering approach
  - Essential SysML artifacts
  - Service request driven modeling approach

§ Task flow in Rational *Harmony™ for Systems Engineering*

§ **Deploying MbSE with Rational® *Rhapsody*®**

§ Documentation of Rational *Harmony™ for Systems Engineering*

Approach for enterprise process adoption

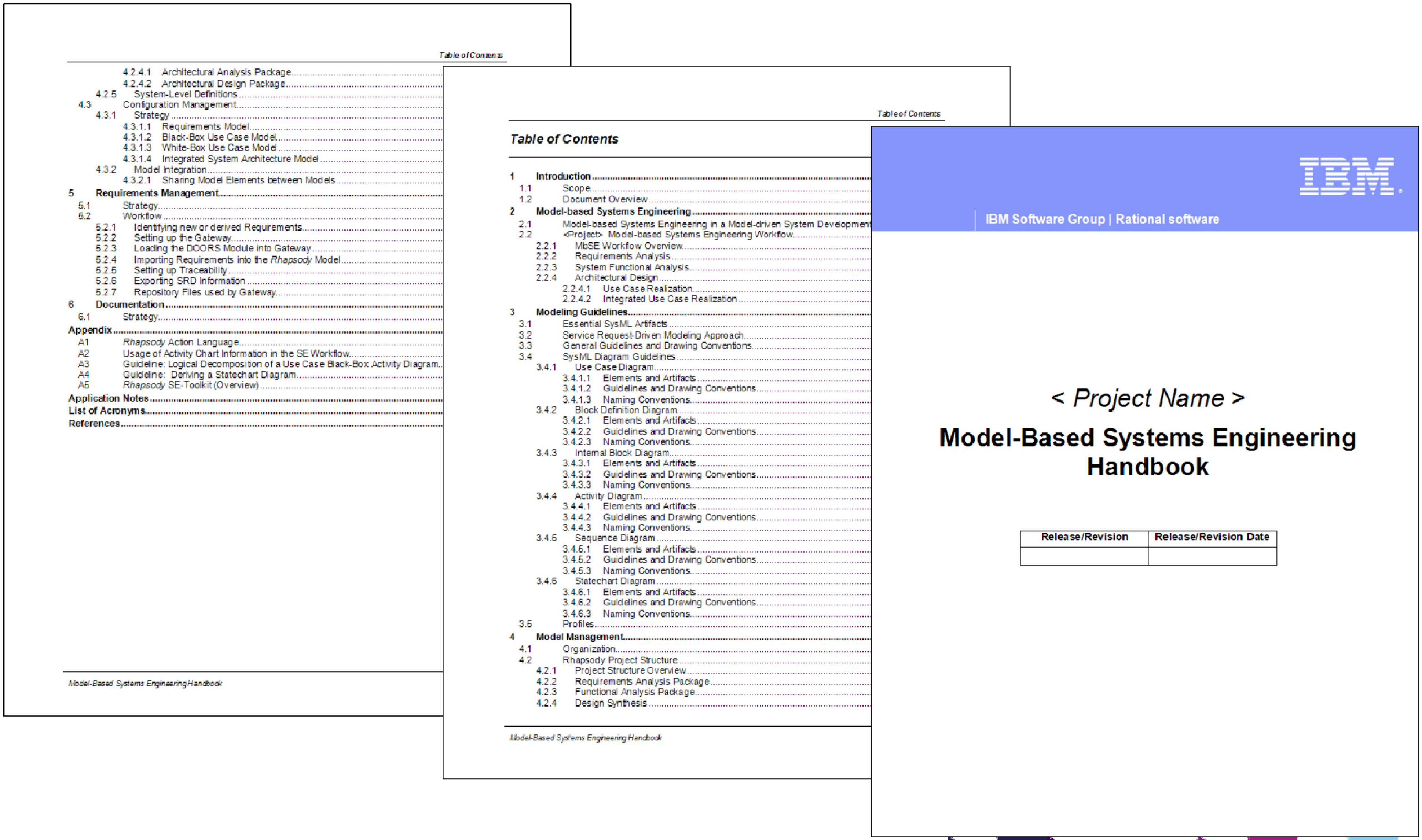


Top-down (e.g., Capability Maturity Model Integration [CMMI])

Bottom-up (e.g., “grassroots”)

Deploying model-based systems engineering

Project specific model-based Systems Engineering Handbook



IBM

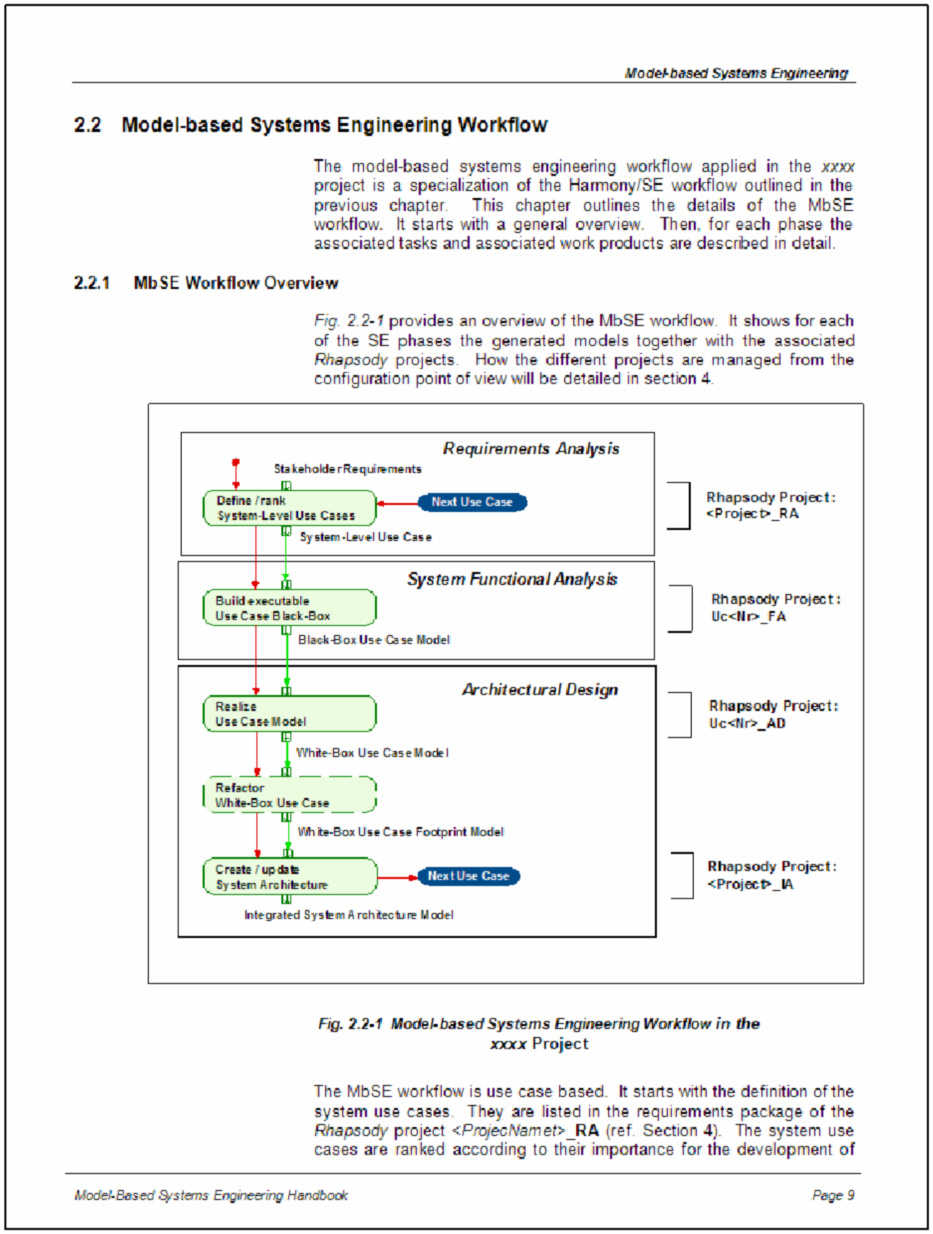
IBM Software Group | Rational software

< Project Name >

Model-Based Systems Engineering Handbook

Release/Revision	Release/Revision Date

Table of Contents



Model-based Systems Engineering

2.2 Model-based Systems Engineering Workflow

The model-based systems engineering workflow applied in the *xxxx* project is a specialization of the Harmony/SE workflow outlined in the previous chapter. This chapter outlines the details of the MbSE workflow. It starts with a general overview. Then, for each phase the associated tasks and associated work products are described in detail.

2.2.1 MbSE Workflow Overview

Fig. 2.2-1 provides an overview of the MbSE workflow. It shows for each of the SE phases the generated models together with the associated *Rhapsody* projects. How the different projects are managed from the configuration point of view will be detailed in section 4.

Requirements Analysis
Stakeholder Requirements
Define /rank System-Level Use Cases
Next Use Case
System-Level Use Case
Rhapsody Project: <Project>_RA
System Functional Analysis
Build executable Use Case Black-Box
Black-Box Use Case Model
Rhapsody Project: Uc<Nr>_FA
Architectural Design
Realize Use Case Model
White-Box Use Case Model
Refactor White-Box Use Case
White-Box Use Case Footprint Model
Create / update System Architecture
Next Use Case
Integrated System Architecture Model
Rhapsody Project: Uc<Nr>_AD
Rhapsody Project: <Project>_IA

***Fig. 2.2-1 Model-based Systems Engineering Workflow in the xxxx* Project**

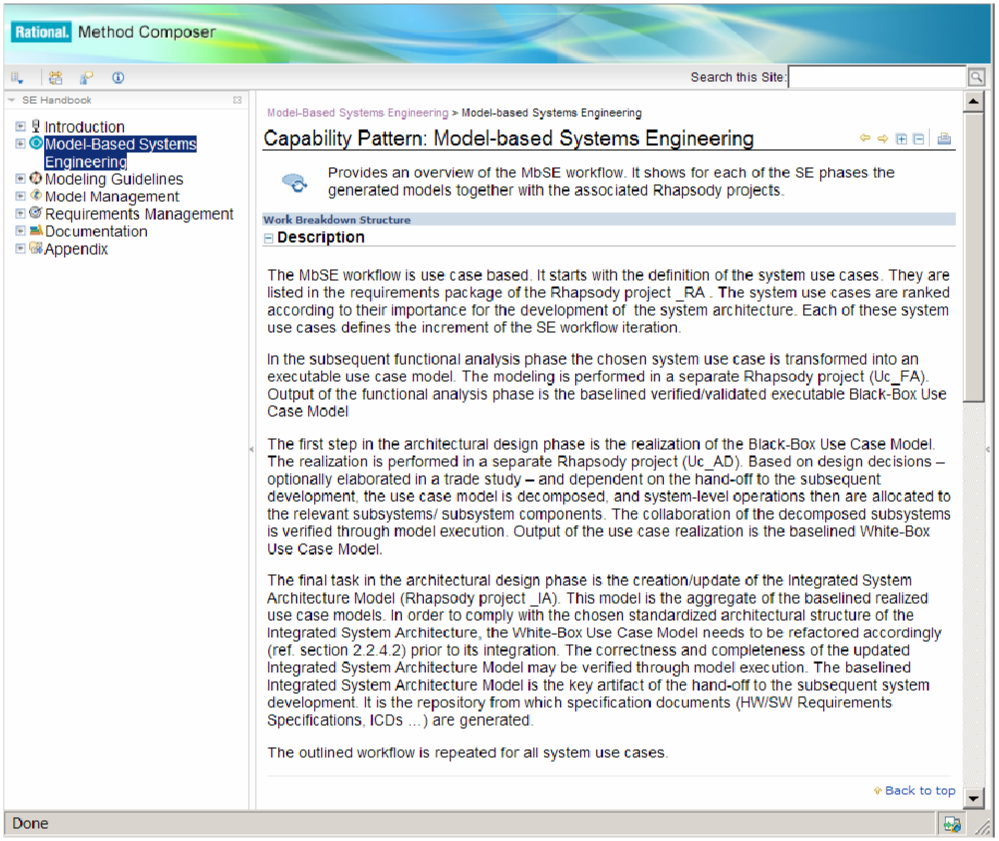
The MbSE workflow is use case based. It starts with the definition of the system use cases. They are listed in the requirements package of the *Rhapsody* project *<ProjecNamet>*_**RA** (ref. Section 4). The system use cases are ranked according to their importance for the development of

Model-Based Systems Engineering Handbook Page 9

MbSE Handbook

Standardizing the MbSE workflow and Rhapsody tool usage

The Alternative: MbSE Handbook in RMC



Model-Based Systems Engineering > Model-based Systems Engineering

Capability Pattern: Model-based Systems Engineering

Provides an overview of the MbSE workflow. It shows for each of the SE phases the generated models together with the associated Rhapsody projects.

Work Breakdown Structure

Description

The MbSE workflow is use case based. It starts with the definition of the system use cases. They are listed in the requirements package of the Rhapsody project _RA . The system use cases are ranked according to their importance for the development of the system architecture. Each of these system use cases defines the increment of the SE workflow iteration.

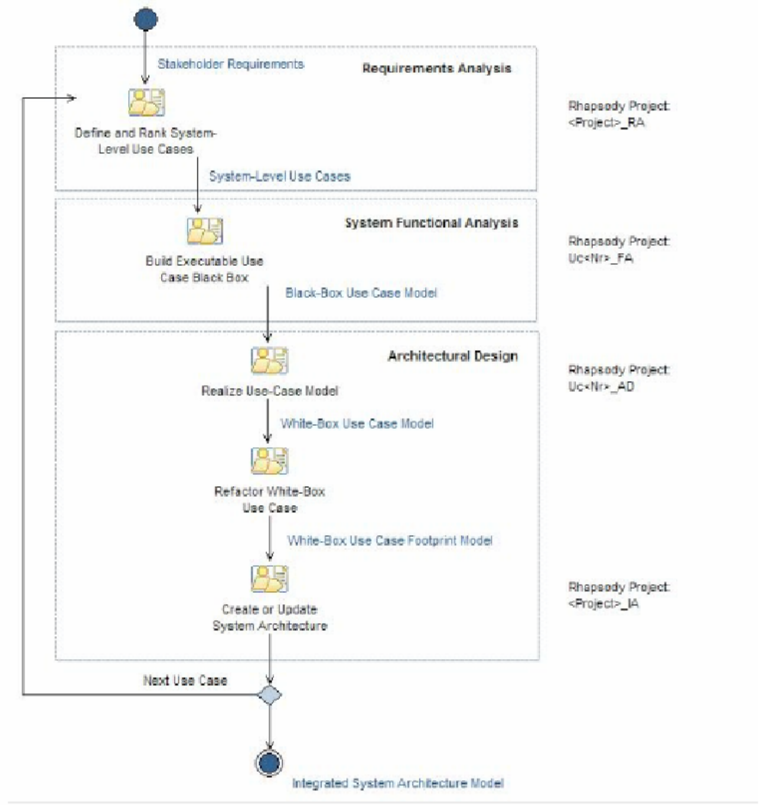
In the subsequent functional analysis phase the chosen system use case is transformed into an executable use case model. The modeling is performed in a separate Rhapsody project (Uc_FA). Output of the functional analysis phase is the baselined verified/validated executable Black-Box Use Case Model

The first step in the architectural design phase is the realization of the Black-Box Use Case Model. The realization is performed in a separate Rhapsody project (Uc_AD). Based on design decisions – optionally elaborated in a trade study – and dependent on the hand-off to the subsequent development, the use case model is decomposed, and system-level operations then are allocated to the relevant subsystems/ subsystem components. The collaboration of the decomposed subsystems is verified through model execution. Output of the use case realization is the baselined White-Box Use Case Model.

The final task in the architectural design phase is the creation/update of the Integrated System Architecture Model (Rhapsody project _IA). This model is the aggregate of the baselined realized use case models. In order to comply with the chosen standardized architectural structure of the Integrated System Architecture, the White-Box Use Case Model needs to be refactored accordingly (ref. section 2.2.4.2) prior to its integration. The correctness and completeness of the updated Integrated System Architecture Model may be verified through model execution. The baselined Integrated System Architecture Model is the key artifact of the hand-off to the subsequent system development. It is the repository from which specification documents (HW/SW Requirements Specifications, ICDs ...) are generated.

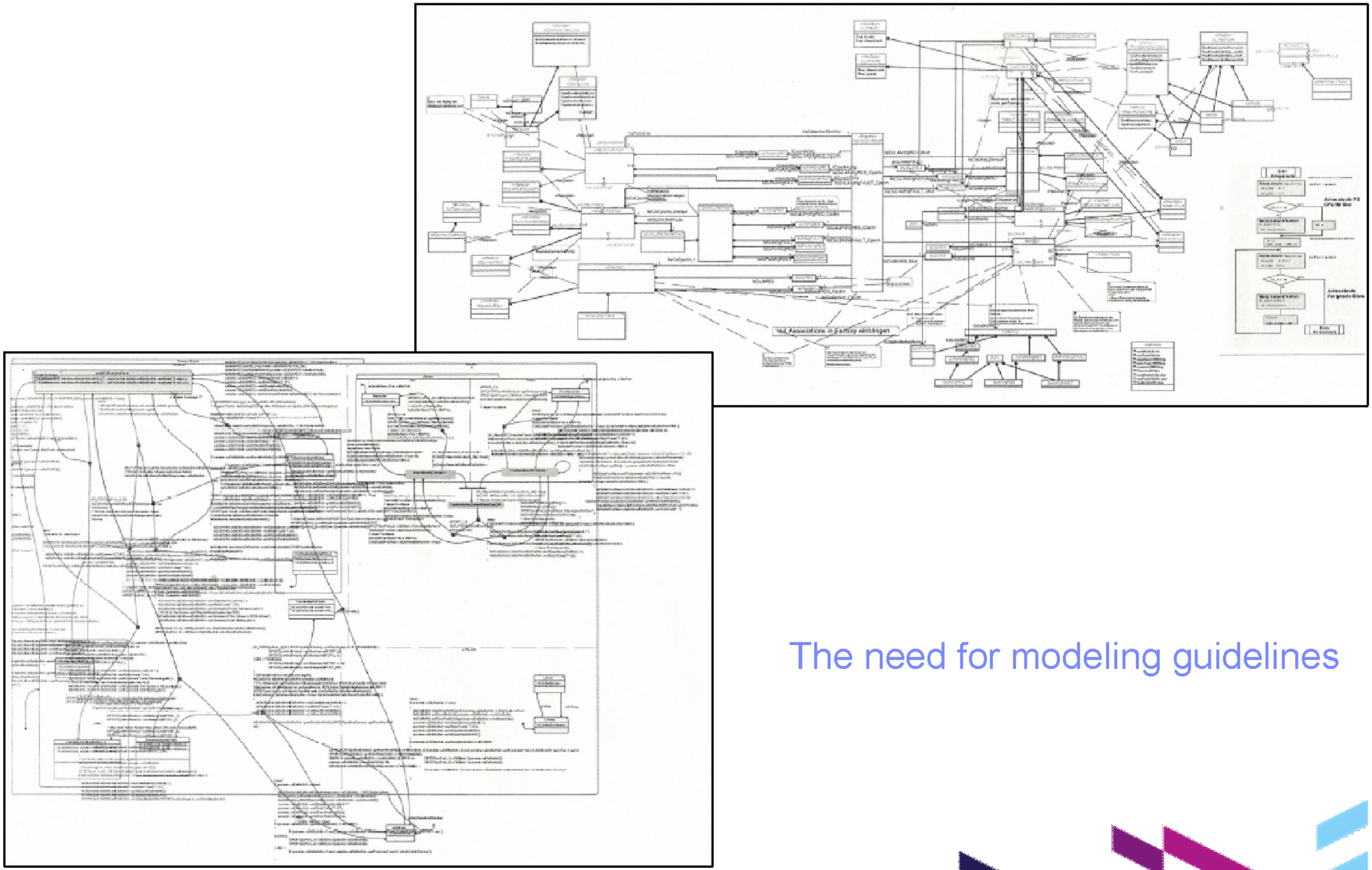
The outlined workflow is repeated for all system use cases.

Workflow



Stakeholder Requirements
Requirements Analysis
Define and Rank System-Level Use Cases
System-Level Use Cases
Rhapsody Project: <Project>_RA
System Functional Analysis
Build Executable Use Case Black Box
Black-Box Use Case Model
Rhapsody Project: Uc<Nr>_FA
Architectural Design
Realize Use-Case Model
White-Box Use Case Model
Rhapsody Project: Uc<Nr>_AD
Refactor White-Box Use Case
White-Box Use Case Footprint Model
Create or Update System Architecture
Rhapsody Project: <Project>_IA
Next Use Case
Integrated System Architecture Model

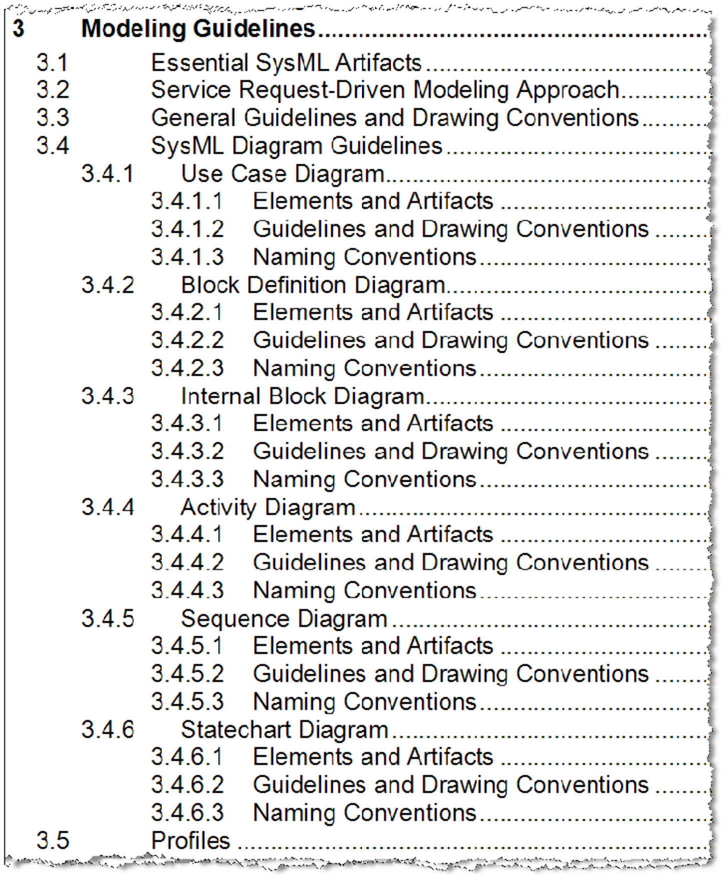
Deploying model-based systems engineering

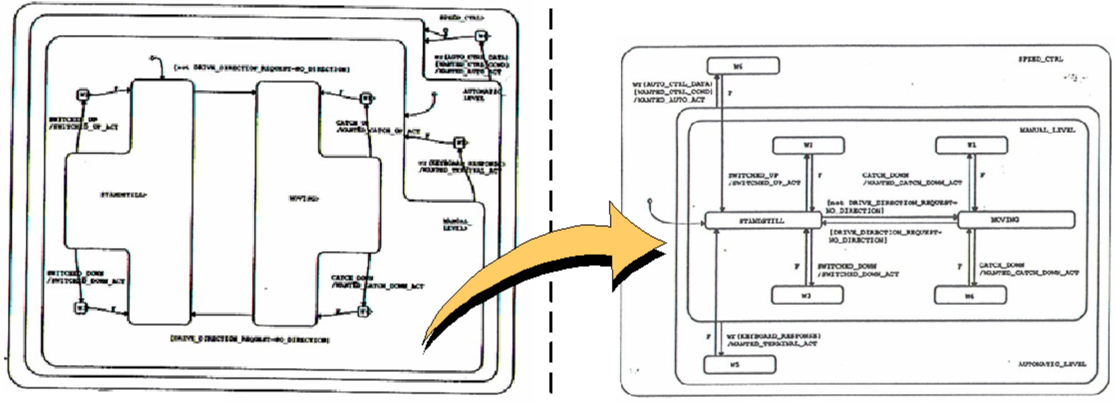


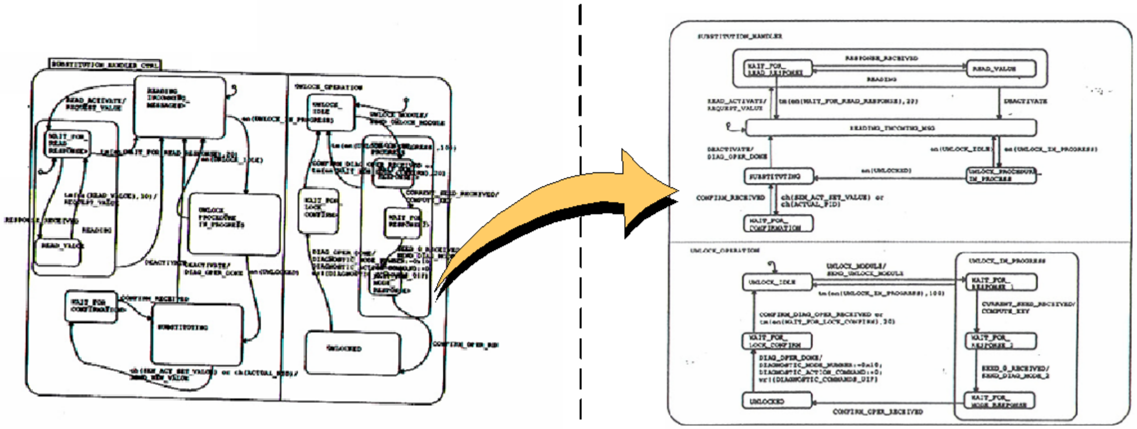
The need for modeling guidelines

MbSE Handbook

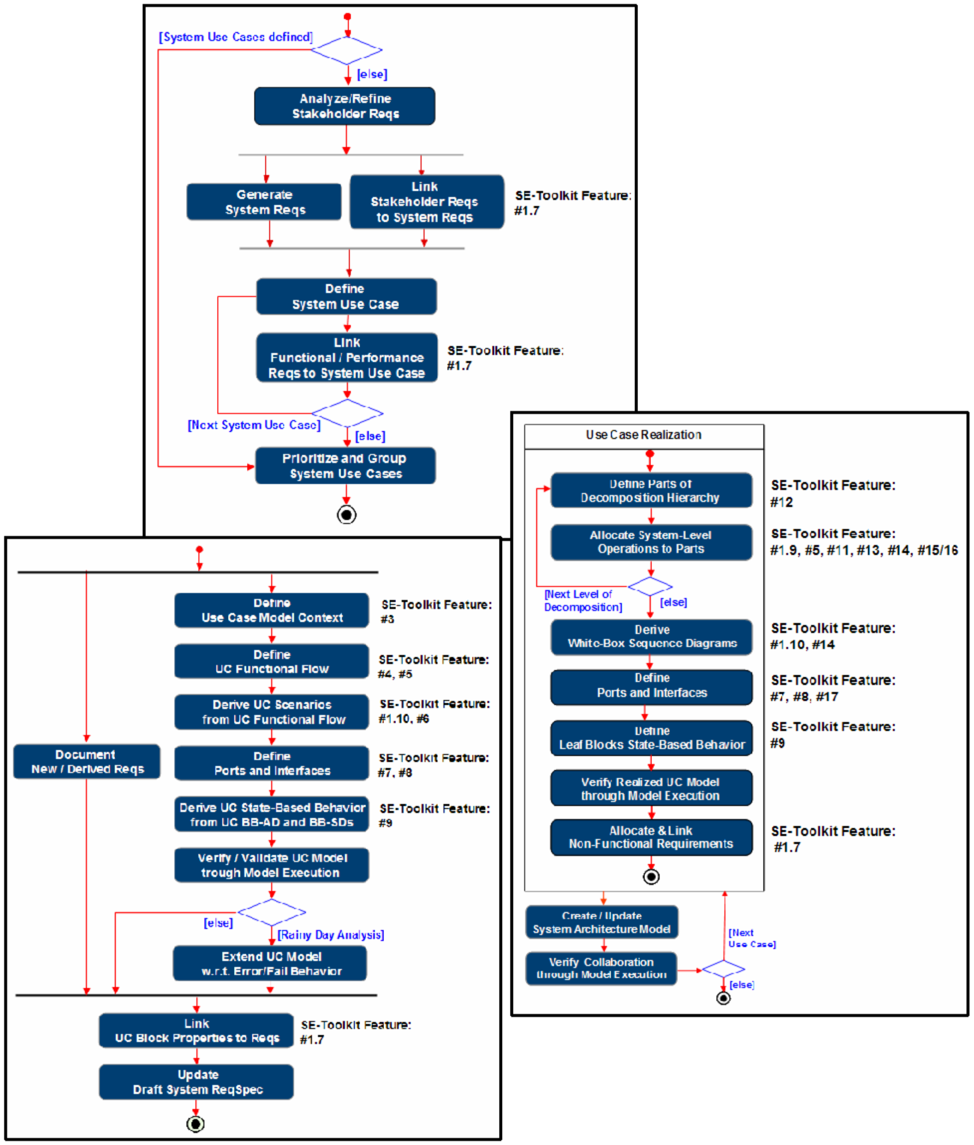
Standardizing the usage of the modeling language







Support of the MbSE workflow through the *Rhapsody SE-Toolkit*



[System Use Cases defined]

[else]

Analyze/Refine Stakeholder Reqs

Generate System Reqs

Link Stakeholder Reqs to System Reqs

SE-Toolkit Feature: #1.7

Define System Use Case

Link Functional / Performance Reqs to System Use Case

SE-Toolkit Feature: #1.7

[Next System Use Case]

[else]

Prioritize and Group System Use Cases

Define Use Case Model Context

SE-Toolkit Feature: #3

Define UC Functional Flow

SE-Toolkit Feature: #4, #5

Derive UC Scenarios from UC Functional Flow

SE-Toolkit Feature: #1.10, #6

Document New / Derived Reqs

Define Ports and Interfaces

SE-Toolkit Feature: #7, #8

Derive UC State-Based Behavior from UC BB-AD and BB-SDs

SE-Toolkit Feature: #9

Verify / Validate UC Model trough Model Execution

[else]

[Rainy Day Analysis]

Extend UC Model w.r.t. Error/Fail Behavior

Link UC Block Properties to Reqs

SE-Toolkit Feature: #1.7

Update Draft System ReqSpec

Use Case Realization

Define Parts of Decomposition Hierarchy

SE-Toolkit Feature: #12

Allocate System-Level Operations to Parts

SE-Toolkit Feature: #1.9, #5, #11, #13, #14, #15/16

[Next Level of Decomposition]

[else]

Derive White-Box Sequence Diagrams

SE-Toolkit Feature: #1.10, #14

Define Ports and Interfaces

SE-Toolkit Feature: #7, #8, #17

Define Leaf Blocks State-Based Behavior

SE-Toolkit Feature: #9

Verify Realized UC Model through Model Execution

Allocate & Link Non-Functional Requirements

SE-Toolkit Feature: #1.7

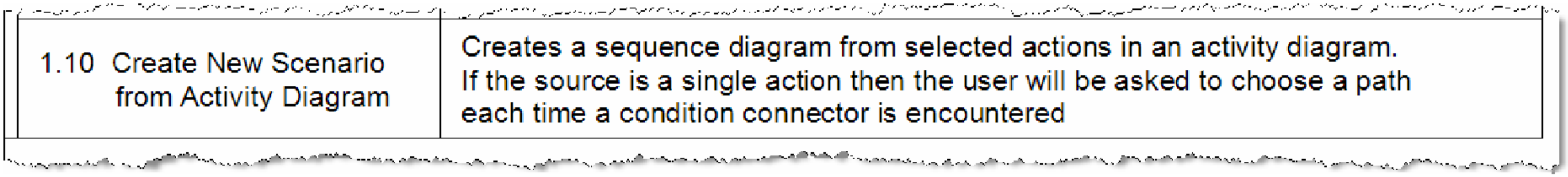
Create / Update System Architecture Model

Verify Collaboration through Model Execution

[Next Use Case]

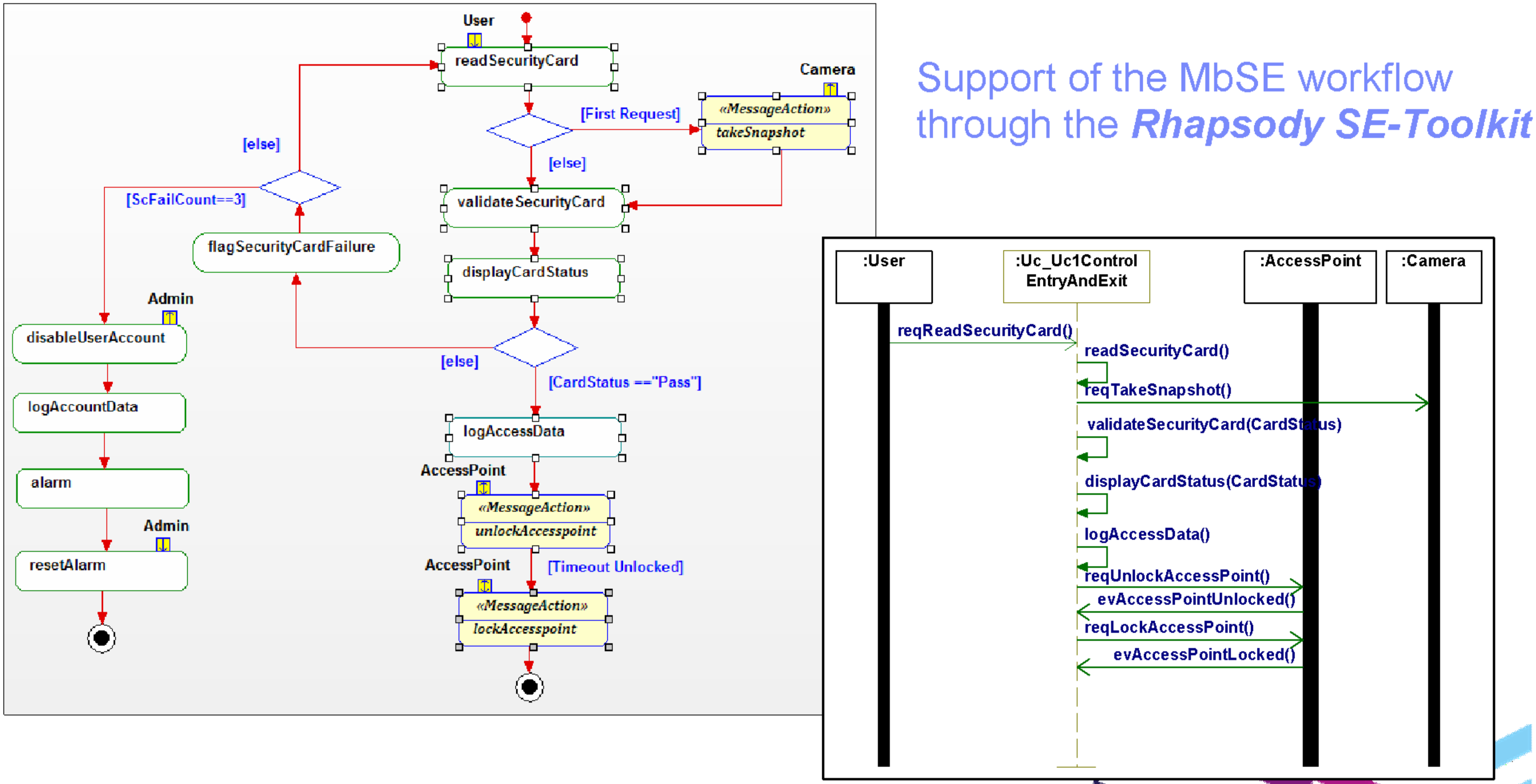
[else]

	SE-Toolkit Feature		Description
1	Modeling Toolbox	1.1 Add Hyperlink(s)	Adds a hyperlink from the source(s) to the destination(s).
		1.2 Add Anchor(s)	Adds an anchor from the source(s) to the destination(s)
		1.3 Add SD Ref(s)	Adds selected sequence diagram(s) as *Referenced Sequences* to the use case.
		1.4 Add Event Reception(s)	Adds receptions of the chosen events to the target interface.
		4.5 Add Value Typr	Maps the seletec value type to the selected unit. Tags of the value type are populated from the unit.
		1.6 Merge Blocks	Copies any operations, receptions, and attributes from the source blocks to a single destination block.
		1.7 Create Dependency	Creates dependencies between model elements.
		1.8 Populate Activity Diagram	For each reflexive message on the selected sequence(s) an action is created on the selected activity diagram
		1.9 Allocate Operations from Swimlanes	Copies operations allocated to a swimlane in a White-Box Activity Diagram into the relevant sub-system block.
		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
2	Create Harmony Project		Creates a *Harmony for Systems Engineering* compliant project structure
3	Create System Model from Use Case		Creates a *Harmony for Systems Engineering* compliant package structure for the use case model
4	Auto-Rename Actions		Harmonizes the action statement and action name in an activity diagram.
5	Add Actor Pins		Adds SysML *action pins* stereotyped *<<ActorPin>>* to the selected action on an activity diagram. User selects the direction and the actor from a drop down list.
6	Perform Activity View Consistency Check		Checks the consistency between actions of the black-box activity diagram and the operations in the derived use case scenarios.
7	Create Ports and Interfaces		Creates behavioral ports and associated interfaces based on scenarios captured in sequence diagrams
8	Connect Ports		Creates links between ports on an internal block diagram
9	Create Initial Statechart		Creates wait state(s) and action states based on the information captured in an Activity Diagram .
10	Merge Functional Analysis		Copies all operations, event receptions and attributes from all use case blocks into the selected block
11	Duplicate Activity View		Makes a copy of an activity view and strips away any referenced scenarios
12	Create Sub Packages		Creates a package per subsystem and moves subsystem blocks into those packages.
13	Architectural Design Wizard		Copies operations from one architectural layer to another and tracks when operations have been allocated.
14	Perform Swimlane Consistency Check		Checks consistency between the allocated actions in swimlanes against the allocated operations in subsystem blocks.
15	Create Allocation Table		Summarizes the allocation of operations of a white-box activity diagram in an Excel spreadsheet.
16	Create Allocation CSV File		As 'Create Allocation Table' – except in a CSV form. Added to the model as a *controlled file*.
17	Generate N2 Matrix		Creates an Excel spreadsheet of the provided and required interface matrix from an internal block diagram
18	Copy MoEs to Children		Copies the MoE attributes of the key function block into the solution blocks.
19	Copy MoEs from Base		Copies the MoE attributes of the key function block into a selected solution block.
20	Perform Trade Analysis		Calculates for a set of solutions a *Weighted Objectives Table* and displays the results in an Excel spreadsheet.



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

"6 Weeks Rhapsody MbSE Engagement" (1)

Prerequisites – to be provided by Customer

§ Provide overview of present workflow/process.

§ Provide requirements associated to the project (e.g. project objective and definition, functional requirements) that will be the basis of the 6 weeks MbSE engagement.

§ Identify "core team" that will be assigned to the project for the next 6 weeks

Set-up Engagement

§ **Step 1 (2 days)**
Analysis of customer's present SE workflow and definition of the MbSE workflow that will be applied in the engagement. This workflow will be documented in the ***MBSE Handbook.***

§ **Step 2 (1-2 days)**
IBM in collaboration with the customer:
- Definition of use cases to be elaborated in the engagement
- Identification of a small use case that will be the basis of the workflow-oriented training

§ **Step 3 (3 days)**
Workflow-oriented Rhapsody tool training. The focus of this training is on the usage of the SE-Toolkit. The attendees should be given additional 2 days to familiarise themselves with the Rhapsody tool.

Deploying model-based systems engineering

“6 Weeks Rhapsody MbSE Engagement” (2)

Week 1

Project Kick-off Workshop. Each team member will be assigned one use case.

Exemplary one or two use cases will be modeled down to the hand-off level (use case realization). At the end of the week IBM will provide a project plan for the next 6 weeks.

Week 2-6

IBM will perform reviews/support of the progress. Especially, when 2 realized use case models are baselined, the merger of the models in the Integrated System Architecture Model will be supported by IBM.

During the engagement, IBM will update/extend the SE Handbook. At the end of the engagement period there will be a review and the MbSE Handbook will be baselined (Release 1.0).

Provided Documents

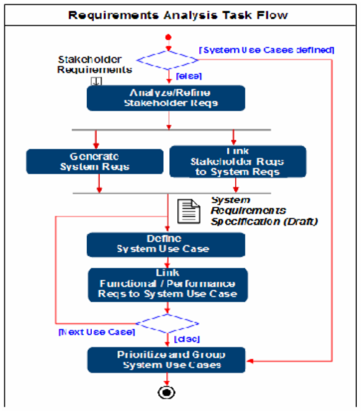
§ ***Deskbook Release 3.1.2***

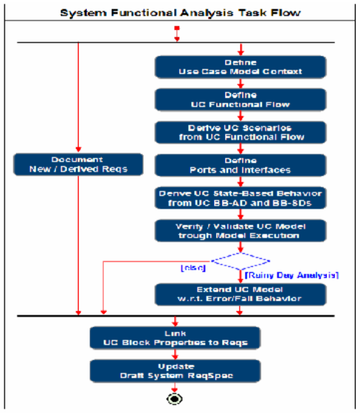
§ Customer project tailored ***MbSE Handbook***.
In order to standardize the usage of the modeling activities (approach, modeling language, …) a draft SE Handbook will be provided at the start of the engagement.

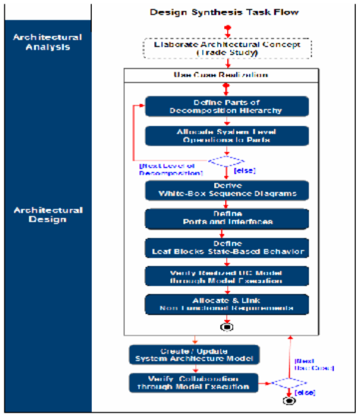
§ **Project Plan**
At the end of the Kick-off Workshop a project plan will be provided.

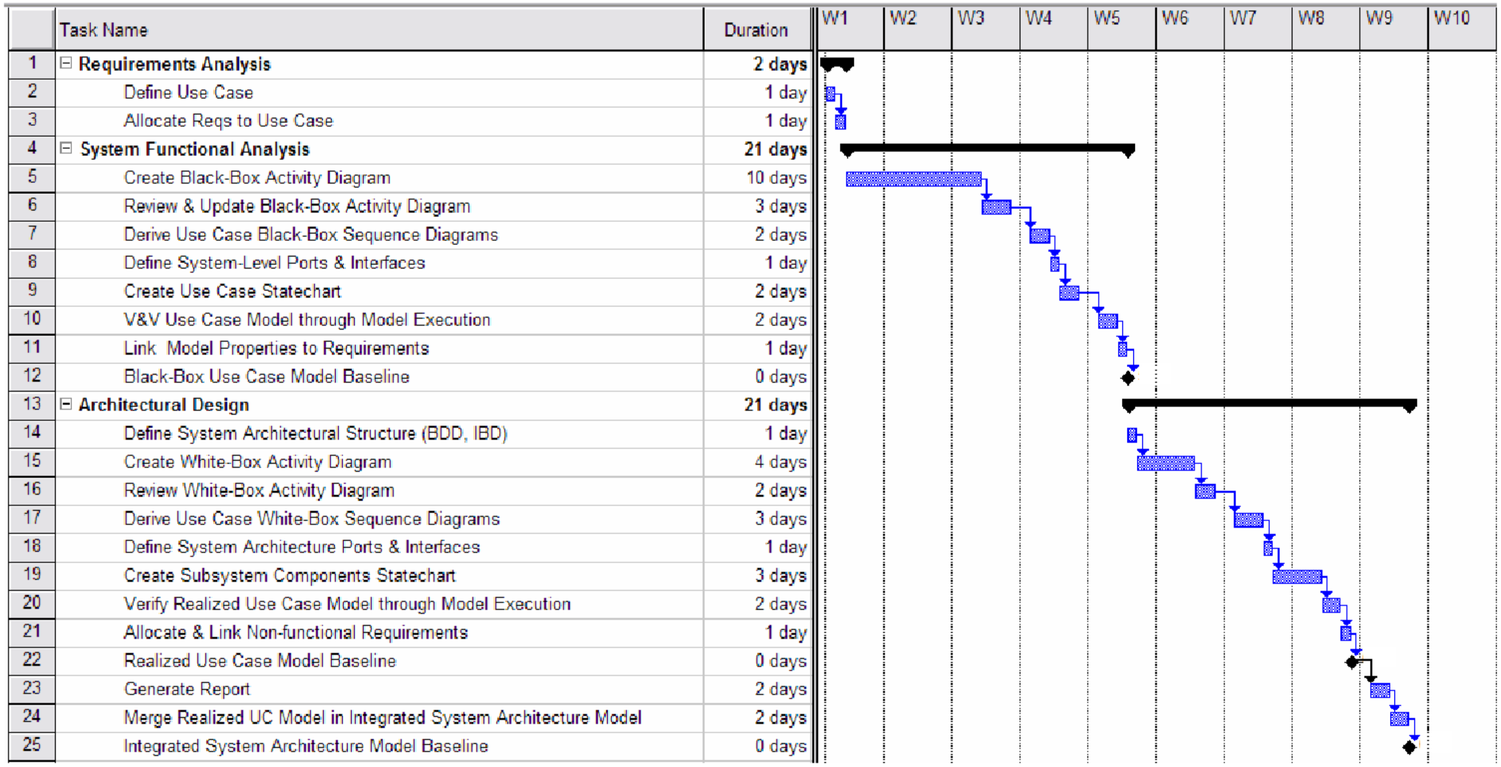
Deploying model-based systems engineering

Managing the modeling activities: Project plan


Requirements Analysis Task Flow
Stakeholder Requirements
[System Use Cases defined]
[else]
Analyze/Refine Stakeholder Reqs
Generate System Reqs
Link Stakeholder Reqs to System Reqs
System Requirements Specification (Draft)
Define System Use Case
Link Functional / Performance Reqs to System Use Case
[Next Use Case]
[else]
Prioritize and Group System Use Cases


System Functional Analysis Task Flow
Define Use Case Model Context
Define UC Functional Flow
Derive UC Scenarios from UC Functional Flow
Document New / Derived Reqs
Define Ports and Interfaces
Derive UC State-Based Behavior from UC BB-AD and BB-SDs
Verify / Validate UC Model trough Model Execution
[else]
[Rainy Day Analysis]
Extend UC Model w.r.t. Error/Fail Behavior
Link UC Block Properties to Reqs
Update Draft System ReqSpec


Design Synthesis Task Flow
Architectural Analysis
Architectural Design
Elaborate Architectural Concept (Trade Study)
Use Case Realization
Define Parts of Decomposition Hierarchy
Allocate System Level Operations to Parts
[Next Level of Decomposition]
[else]
Derive White-Box Sequence Diagrams
Define Ports and Interfaces
Define Leaf Blocks State-Based Behavior
Verify Realized UC Model through Model Execution
Allocate & Link Non Functional Requirements
Create / Update System Architecture Model
Verify Collaboration through Model Execution
[Next Use Case]
[else]



	Task Name	Duration
1	⊟ **Requirements Analysis**	**2 days**
2	Define Use Case	1 day
3	Allocate Reqs to Use Case	1 day
4	⊟ **System Functional Analysis**	**21 days**
5	Create Black-Box Activity Diagram	10 days
6	Review & Update Black-Box Activity Diagram	3 days
7	Derive Use Case Black-Box Sequence Diagrams	2 days
8	Define System-Level Ports & Interfaces	1 day
9	Create Use Case Statechart	2 days
10	V&V Use Case Model through Model Execution	2 days
11	Link Model Properties to Requirements	1 day
12	Black-Box Use Case Model Baseline	0 days
13	⊟ **Architectural Design**	**21 days**
14	Define System Architectural Structure (BDD, IBD)	1 day
15	Create White-Box Activity Diagram	4 days
16	Review White-Box Activity Diagram	2 days
17	Derive Use Case White-Box Sequence Diagrams	3 days
18	Define System Architecture Ports & Interfaces	1 day
19	Create Subsystem Components Statechart	3 days
20	Verify Realized Use Case Model through Model Execution	2 days
21	Allocate & Link Non-functional Requirements	1 day
22	Realized Use Case Model Baseline	0 days
23	Generate Report	2 days
24	Merge Realized UC Model in Integrated System Architecture Model	2 days
25	Integrated System Architecture Model Baseline	0 days

Agenda

§ Model-based systems engineering in a model-driven development lifecycle

- Fundamentals of the Rational model-based systems engineering approach
  - Essential SysML artifacts
  - Service request driven modeling approach

§ Task flow in Rational *Harmony™ for Systems Engineering*

§ Deploying MbSE with Rational® *Rhapsody®*

§ **Documentation of Rational *Harmony™ for Systems Engineering***

Rational *Rhapsody*® tool focused documentation:

Rational *Harmony*™ *for Systems Engineering Deskbook.*

Case Study: Hand-Off to Subsystem Development

Case Study: Design Synthesis

Case Study: System Functional Analysis

Case Study: Requirements Analysis

Rhapsody Project Structure

Fundamentals of Harmony for Systems Engineering

IBM Software Group

February 2011

Systems Engineering Best Practices with the Rational Solution for Systems and Software Engineering

Deskbook Release 3.1.2

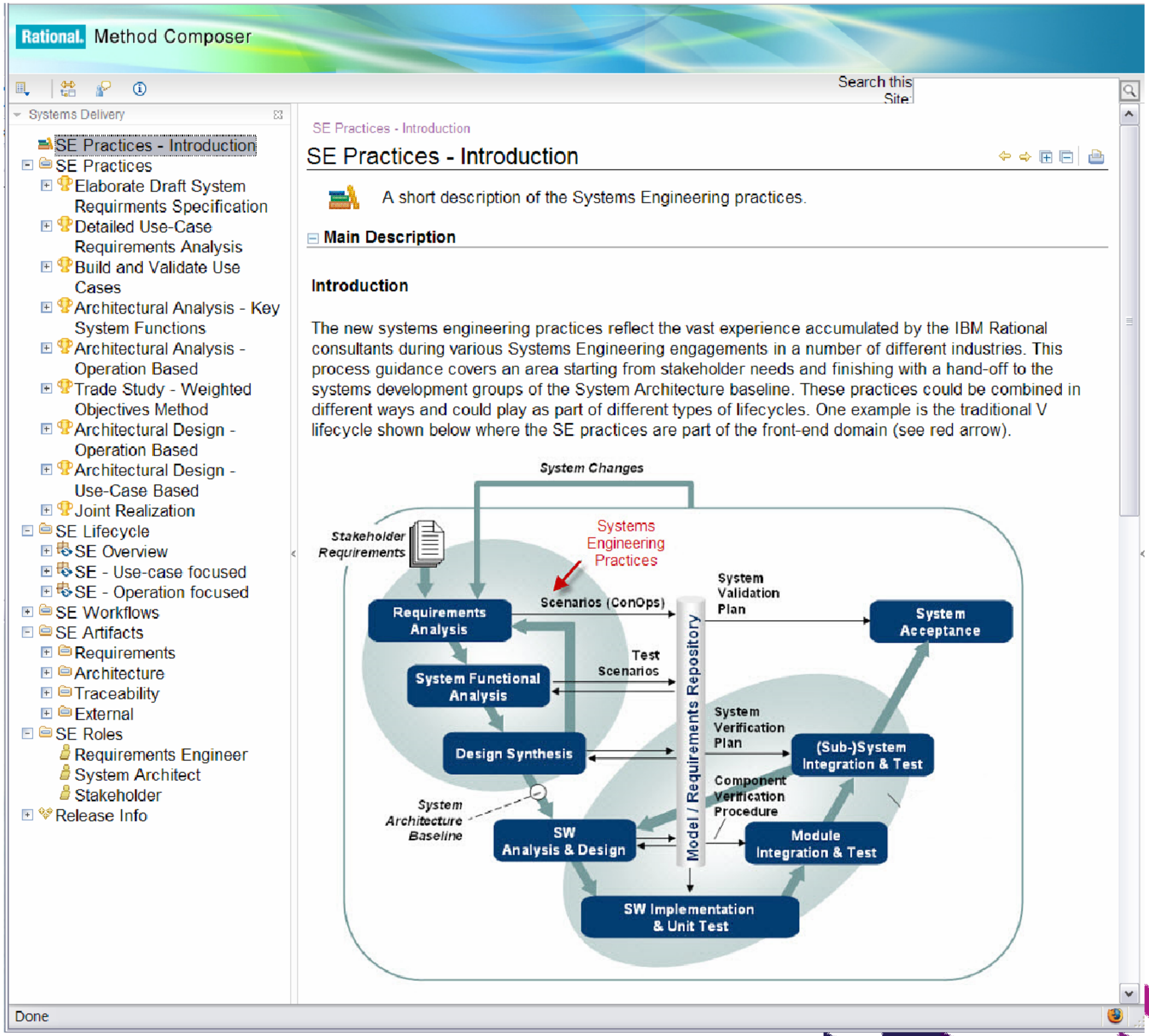
Model-Based Systems Engineering with *Rational Rhapsody* and *Rational Harmony for Systems Engineering*

Hans-Peter Hoffmann, Ph.D.
Chief Systems Methodologist

hoffmape@us.ibm.com

© 2011 IBM Corporation

Rational Systems Engineering Practices captured in RMC



- SE Practices - Introduction
- SE Practices
  - Elaborate Draft System Requirments Specification
  - Detailed Use-Case Requirements Analysis
  - Build and Validate Use Cases
  - Architectural Analysis - Key System Functions
  - Architectural Analysis - Operation Based
  - Trade Study - Weighted Objectives Method
  - Architectural Design - Operation Based
  - Architectural Design - Use-Case Based
  - Joint Realization
- SE Lifecycle
  - SE Overview
  - SE - Use-case focused
  - SE - Operation focused
- SE Workflows
- SE Artifacts
  - Requirements
  - Architecture
  - Traceability
  - External
- SE Roles
  - Requirements Engineer
  - System Architect
  - Stakeholder
- Release Info

SE Practices - Introduction

SE Practices - Introduction

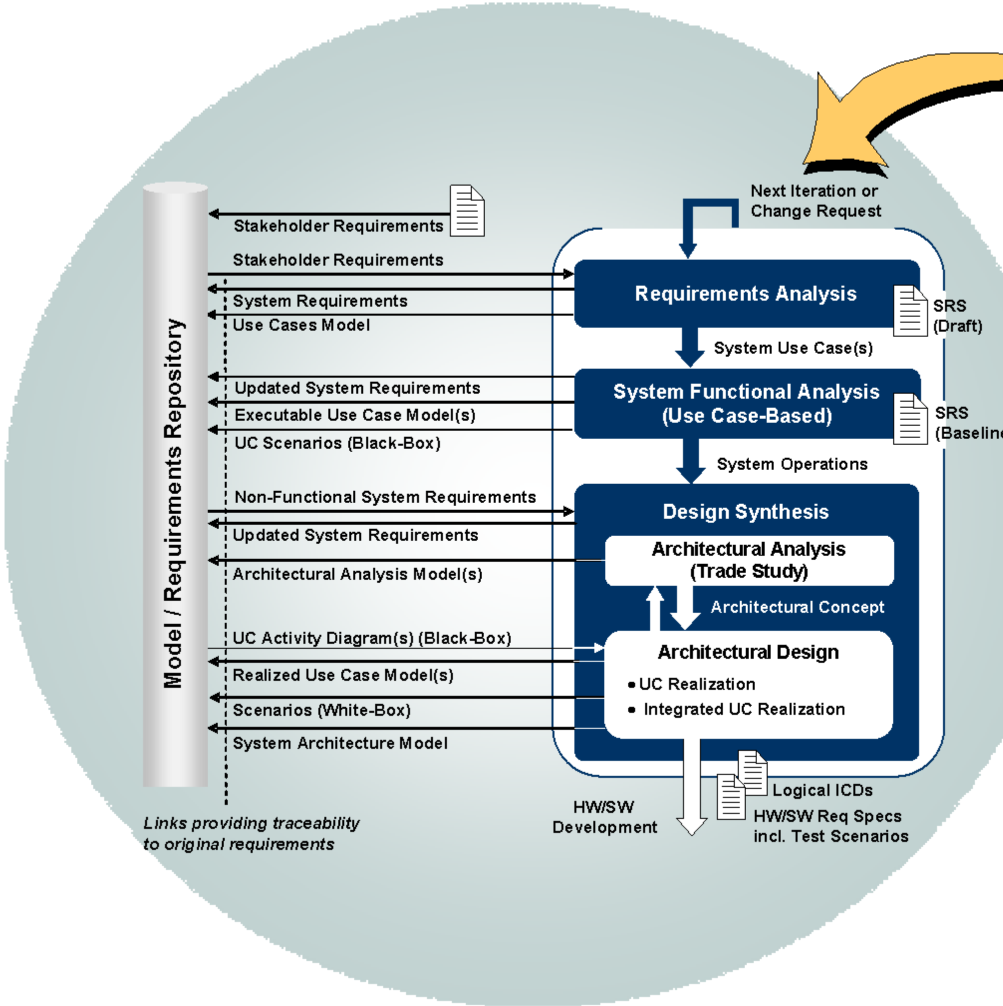
A short description of the Systems Engineering practices.

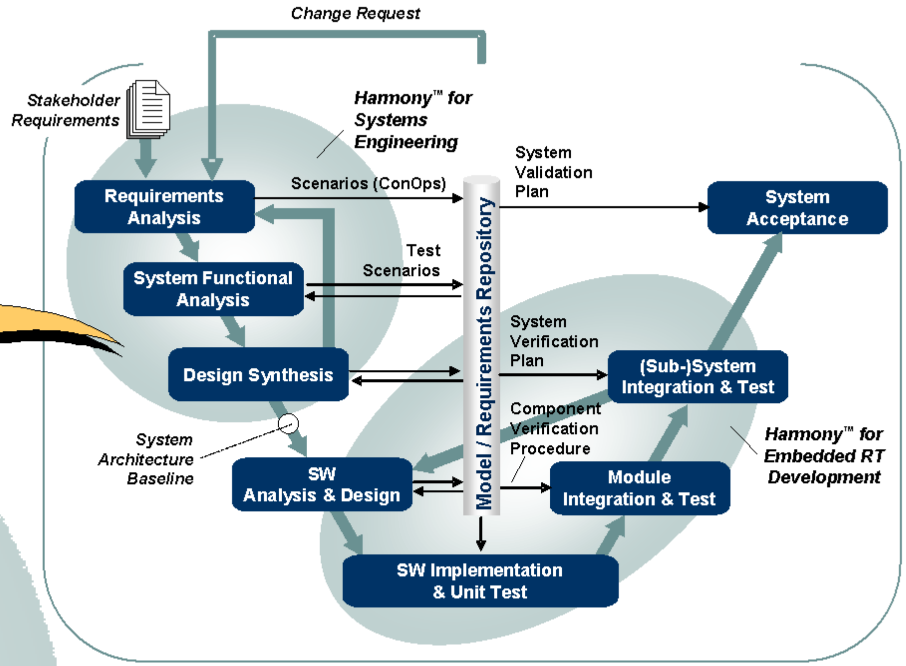
Main Description

Introduction

The new systems engineering practices reflect the vast experience accumulated by the IBM Rational consultants during various Systems Engineering engagements in a number of different industries. This process guidance covers an area starting from stakeholder needs and finishing with a hand-off to the systems development groups of the System Architecture baseline. These practices could be combined in different ways and could play as part of different types of lifecycles. One example is the traditional V lifecycle shown below where the SE practices are part of the front-end domain (see red arrow).

System Changes
Stakeholder Requirements
Systems Engineering Practices
Requirements Analysis
Scenarios (ConOps)
System Validation Plan
System Acceptance
System Functional Analysis
Test Scenarios
Model / Requirements Repository
System Verification Plan
Design Synthesis
(Sub-)System Integration & Test
Component Verification Procedure
System Architecture Baseline
SW Analysis & Design
Module Integration & Test
SW Implementation & Unit Test


Model / Requirements Repository
Stakeholder Requirements
Stakeholder Requirements
System Requirements
Use Cases Model
Updated System Requirements
Executable Use Case Model(s)
UC Scenarios (Black-Box)
Non-Functional System Requirements
Updated System Requirements
Architectural Analysis Model(s)
UC Activity Diagram(s) (Black-Box)
Realized Use Case Model(s)
Scenarios (White-Box)
System Architecture Model
Links providing traceability to original requirements
Next Iteration or Change Request
Requirements Analysis
SRS (Draft)
System Use Case(s)
System Functional Analysis (Use Case-Based)
SRS (Baseline)
System Operations
Design Synthesis
Architectural Analysis (Trade Study)
Architectural Concept
Architectural Design
• UC Realization
• Integrated UC Realization
Logical ICDs
HW/SW Development
HW/SW Req Specs incl. Test Scenarios


Change Request
Stakeholder Requirements
Harmony™ for Systems Engineering
Requirements Analysis
Scenarios (ConOps)
System Validation Plan
System Acceptance
System Functional Analysis
Test Scenarios
Model / Requirements Repository
System Verification Plan
Design Synthesis
(Sub-)System Integration & Test
Component Verification Procedure
System Architecture Baseline
SW Analysis & Design
Module Integration & Test
Harmony™ for Embedded RT Development
SW Implementation & Unit Test

Rational *Harmony™ for Systems Engineering*

www.ibm.com/software/rational