



**WP 8.1.3 – DEFINE  
METHODOLOGY FOR VALIDATION  
WITHIN OATA**

**Architecture Non-Functional  
Assessment Process**

Document Identifier:	OATA-P2-D8.1.3-01
Edition:	2.0
Edition Date:	24-04-2006

**DOCUMENT CHARACTERISTICS**

TITLE			
<b>Architecture Non-Functional Assessment Process</b>			
<b>EATM Infocentre Reference:</b>			
<b>Document Identifier:</b>	OATA-P2-D8.1.3-01	<b>Edition:</b>	2.0
<b>Contractual Ref:</b>	D5b	<b>Version Date:</b>	24-04-2006
<b>Contractual ID:</b>			
<b>Abstract</b>			
This document presents a methodology that permits the assessment of the capability of the OATA logical architecture to meet a set of non-functional requirements identified by Eurocontrol Domains.			
<b>Keywords</b>			
Validation ISO 13236	Logical Architecture Quality Framework	Method NFR	
<b>Contact Person(s)</b>		<b>Tel</b>	<b>Unit</b>
Prepared by:	SWEDAVIA, COMBITECH, ISDEFE		
Issued by:	Hans Wagemans, WP 8.1 Manager	+32 2 729 3334	DAS/ESC

STATUS, AUDIENCE AND ACCESSIBILITY					
Status		Intended for		Accessible via	
In progress	<input type="checkbox"/>	General Public	<input type="checkbox"/>	Intranet	<input checked="" type="checkbox"/>
Internal Draft	<input type="checkbox"/>	EATM Stakeholders	<input checked="" type="checkbox"/>	Extranet	<input type="checkbox"/>
Working Draft	<input type="checkbox"/>	Restricted Audience	<input type="checkbox"/>	Configuration Manager	<input type="checkbox"/>
Proposed Issue	<input checked="" type="checkbox"/>	<i>Printed &amp; electronic copies of the document can be obtained from the EATM Infocentre or from the OATA PSO</i>			
Released Issue	<input type="checkbox"/>				

ELECTRONIC SOURCE		
<b>Path:</b>	http://pollux.mis.eurocontrol.be/iw/cci/meta/no-injection/iw-mount/default/main/template_web/eatm/valfor/WORKAREA/work/web/gallery/content/public/zz_tst_borkenhu_1_OATA-P2-D8.1.3-01 DMVO Architecture NFRs Assessment Process.doc	
<b>File Name:</b>	zz_tst_borkenhu_1_OATA-P2-D8.1.3-01 DMVO Architecture NFRs Assessment Process	
<b>Host System:</b>	<b>Software Application</b>	<b>Size:</b>
Windows XP:	Microsoft Word 10.0	1033 Kb

<b>EATM Infocentre</b> EUROCONTROL Headquarters 96 Rue de la Fusée, B-1130 BRUSSELS Tel: +32 (0)2 729 51 51 Fax: +32 (0)2 729 99 84 E-mail: eatm.infocentre@eurocontrol.int	<b>OATA Project Support Office (PSO)</b> EUROCONTROL Headquarters 96 Rue de la Fusée, B-1130 BRUSSELS Tel: +32 (0)2 729 50 40 E-mail: oata.pso@eurocontrol.int
--	--

### DOCUMENT APPROVAL

The following table identifies all management authorities who have successively approved the present issue of this document.

AUTHORITY	NAME AND SIGNATURE	DATE
Contractor	SWEDAVIA	18-04-2006
Work Package Manager	Hans Wagemans	03-05-2006
Internal Review Board		
Technical Review Group		
Project Manager		

### DOCUMENT CONTROL

#### Copyright notice

© 2007 European Organisation for the Safety of Air Navigation (EUROCONTROL).  
 All rights reserved.  
 "Member States of the Organisation are entitled to use and reproduce this document for internal and non-commercial purpose under their vested tasks. Any disclosure to third parties shall be subject to prior written permission of EUROCONTROL".

### DOCUMENT CHANGE RECORD

The following table records the complete history of the successive editions of the present document.

Edition Number	Edition Date	Reason for change	Pages affected
0.A	15 Feb 04	Creation.	
0.B	15 Mar 04	Internal comments	
0.C	01 April 04	Comments from Review meeting. Internal comments	
0.D	12 May 04	Comments from Review meeting	
0.E	21 June 04	Comments from E. Isambert, OATA WP5 Manager	
0.F	21 July 04	New characterizations of Quality attributes following comments of July meeting	
0.G	07 Dec 04	Reconstruction	
0.H	17 Jan 05	Update after review (9/12/04) (Pre delivery)	

## Architecture Non-Functional Assessment Process

0.I	14 Feb. 05	Update after review (14/02/05) Meeting+review	
0.J	2 Mar 05	Update during meeting	
0.K	8 Mar 05	Executive summary	
1.0	5 April 2006	<p>Updates after Pilot Validation and internal review.</p> <p>The major changes are that all validation that is dependant on an engineering model has been moved to the appendices. The methodology presents the quality matrix that shall be used during the definition of non-functional validation objectives. Two new roles are introduced; Documentation Expert and Repository Expert. Added information and recommendations gained in the Pilot Validation.</p>	All
2.0	24 April 2006	Update after IRB 12-04-2006	

**TABLE OF CONTENTS**

**DOCUMENT CHARACTERISTICS ..... II**

**DOCUMENT APPROVAL..... III**

**DOCUMENT CONTROL..... III**

**DOCUMENT CHANGE RECORD ..... III**

**TABLE OF CONTENTS..... V**

**LIST OF FIGURES..... VI**

**EXECUTIVE SUMMARY..... 1**

**1 INTRODUCTION..... 2**

1.1 DOCUMENT BACKGROUND..... 2

1.2 DOCUMENT STRUCTURE ..... 2

1.3 READING GUIDELINES..... 3

1.4 DEFINITIONS AND TERMS ..... 4

1.5 BIBLIOGRAPHY..... 5

**2 METHODOLOGY OVERVIEW..... 7**

2.1 INTRODUCTION ..... 7

2.2 VALIDATION OF NON-FUNCTIONAL REQUIREMENTS ..... 7

2.3 VALIDATION ASPECTS..... 7

2.4 OVERVIEW OF SELECTED ASPECTS ..... 8

2.4.1 NFR mapping..... 8

2.4.2 NFR dependencies..... 9

2.4.3 NFR system ability..... 10

2.5 GENERAL RECOMMENDATIONS..... 10

**3 PROCESSES ..... 11**

3.1 INTRODUCTION ..... 11

3.1.1 Required roles ..... 15

3.2 PREPARATION..... 16

3.2.1 Introduction ..... 16

3.2.2 Required resources ..... 18

3.2.3 Activity: Select the OATA Architecture ..... 18

3.2.4 Activity: Plan the validation cycle..... 19

3.2.5 Activity: Prepare the repository..... 20

3.2.6 Activity: Define Validation Aims and objectives..... 21

3.2.7 Activity: Identify Operational Threads..... 23

3.3 EXECUTION ..... 25

3.3.1 Introduction ..... 25

3.3.2 Required resources ..... 27

3.3.3 Activity: Assess NFR mapping ..... 27

3.3.4 Activity: Assess NFR dependencies..... 28

3.4 ANALYSIS AND REPORTING..... 30

3.4.1 Introduction ..... 30

3.4.2 Required resources ..... 32

3.4.3 Activity: Validate NFR Mapping ..... 32

3.4.4 Activity: Validate NFR Dependencies..... 33

3.4.5 Activity: Assess Validation Aims..... 34

3.4.6 Activity: Write the validation report(s)..... 35

<b>APPENDIX A QUANTITATIVE VALIDATION PROCESS .....</b>	<b>37</b>
3.5 NFR SYSTEM ABILITY .....	37
3.6 PREPARATION.....	37
3.6.1 Activity: Define and prioritise Validation Scenarios (System ability) .....	37
3.6.2 Activity: Select validation technique (System ability).....	39
3.6.3 Activity: Write guidelines for Validation Models (System ability) .....	40
3.7 EXECUTION .....	41
3.7.1 Activity: Run Validation Models (System ability) .....	41
3.8 ANALYSIS AND REPORTING.....	43
3.8.1 Activity: Validate system ability.....	43
<b>4 APPENDIX B: VALIDATION SCENARIOS.....</b>	<b>45</b>
4.1 CONCEPT AND STRUCTURE OF QUALITY ATTRIBUTE VALIDATION SCENARIOS .....	45
4.2 TYPES OF QUALITY ATTRIBUTE VALIDATION SCENARIOS .....	45
4.3 HOW TO BUILD QUALITY ATTRIBUTE VALIDATION SCENARIOS .....	46
<b>5 APPENDIX C: EFFICIENCY ANALYSIS. PROPOSED TECHNIQUES .....</b>	<b>48</b>
5.1 EFFICIENCY ANALYSIS IN THE CONTEXT OF THE OBJECT ORIENTED LIFECYCLE.....	48
5.2 EFFICIENCY MEASUREMENT .....	49
5.2.1 Resource Utilization.....	49
5.2.2 Resource Throughput.....	49
5.2.3 Mean Service Time.....	50
5.2.4 Residence Time.....	50
5.2.5 Queue Length.....	50
5.3 PERFORMANCE ENGINEERING TECHNIQUES.....	51
5.4 SCHEDULABILITY ANALYSIS TECHNIQUES. RATE MONOTONIC ANALYSIS.....	55

### LIST OF FIGURES

Figure 1: Summary of the OATA Validation Process .....	2
Figure 2: Qualitative and Quantitative Validation .....	8
Figure 3: Process Overview .....	12
Figure 4: Resource Overview .....	13
Figure 5: Information Flow Overview .....	14
Figure 6: Preparation.....	17
Figure 7: Execution .....	26
Figure 8: Analysis and Reporting.....	31
Figure 9: Execution Profile .....	50
Figure 10: Notation used in the Execution Model.....	52
Figure 11: Execution Model Example .....	53
Figure 12: Queuing Network Model Example .....	54
Figure 13: Scenarios Validated using Performance Engineering Techniques.....	55
Figure 14: Scenarios validated using schedulability analysis techniques (RMA).....	60

## EXECUTIVE SUMMARY

The OATA logical architecture addresses the definition of high-level services to be provided by future systems in order to achieve the user requirements. The non-functional requirements (NFRs) describe the required quality of services, for instance timeliness, capacity or other quality characteristics.

A validation of non-functional requirements is important in order to increase the confidence in the OATA logical architecture compliance with the defined non-functional requirements in [OATA WP 5.1]. The main objectives for WP 8.1.3 are defined as follows:

- *Qualitative validation* on the OATA logical architecture regarding mapping and correctness of non-functional requirements.
- *(OPTIONAL) Quantitative validation* on the OATA logical architecture regarding the ability to fulfil the non-functional requirements.

*Qualitative validation* assesses the quality of validation objects. *Qualitative validation* in WP 8.1.3 is done in two ways:

- Validation of the correctness of the NFR mapping to the OATA logical architecture.
- Validation of the correctness of the NFRs in the context of Operational Threads.

*Quantitative validation* assesses the validation objects expressed in numbers. Quantitative validation requires a Validation Model for a specific scenario in order to assess an NFR regarding e.g. performance or capacity. The Validation Model is based on OATA logical architecture added with technical assumptions concerning e.g. pay-loads and storage capacity in order to simulate a specific scenario. The simulation result is used to assess the assumed ability of OATA architecture to fulfil the defined non-functional requirement. Quantitative validation is not further described in this methodology since it is not possible on a logical model.

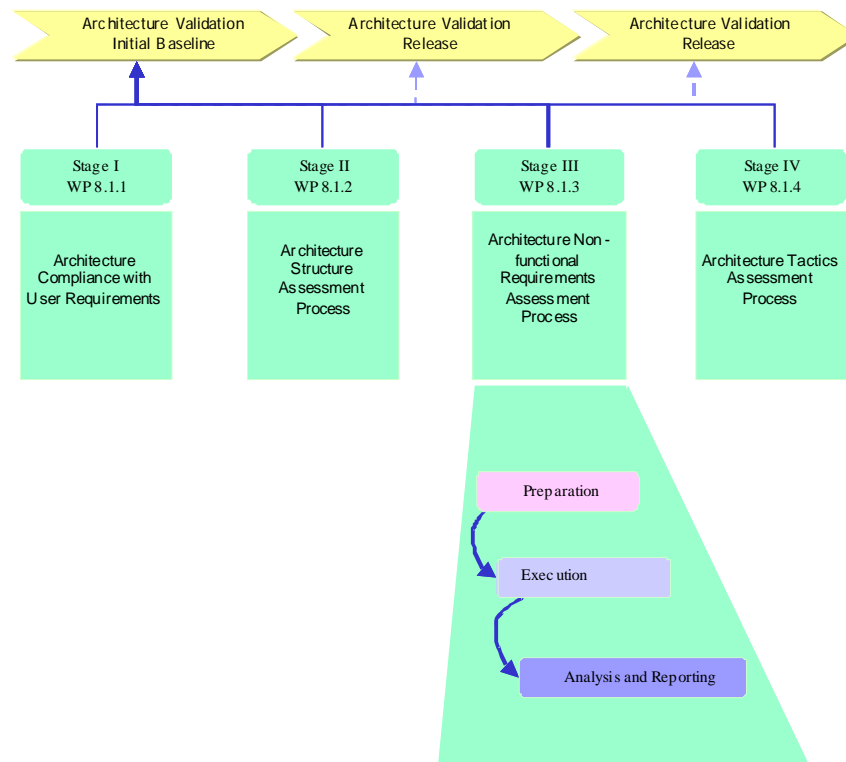
A validation of the OATA logical architecture and its compliance with the defined non-functional requirements will result in an assessment of the consistency and completeness of the specification of required services and quality of services, an analysis of performance issues and finally recommendations of improvements of the non-functional requirements.

# 1 INTRODUCTION

## 1.1 Document Background

The objective of this document is to present a methodology that permits the assessment of the capability of the OATA logical architecture to meet a set of non-functional requirements identified by Eurocontrol Domains.

The assessment of the non-functional requirements with respect to the logical architecture is depicted as part of the OATA validation framework represented in Figure 1.



**Figure 1: Summary of the OATA Validation Process**

The quality characteristics of a complex system are determined by its architecture. Functional completeness is not the unique motivation for architecture design. Architectural decisions have a deep impact on the achievement of quality characteristics and therefore they are the focus of architecture assessment.

A prerequisite of an assessment is to have specified the non-functional requirements that are motivated by the user needs. The architecture documentation should include a clear articulation of the architecture design decisions based on specified non-functional requirements.

Several sources have been used as references in this document. A complete list of these external references can be found in section “Bibliography”.

## 1.2 Document structure

This document has the following main chapters:

- **Introduction.**

The introduction contains description of the background of this document, definitions and terms and a list of references.

- **Methodology Overview.**

The methodology overview contains a high level description of the validation process and the proposed quality model to be used in the validation.

- **Processes.**

The processes chapter details each part of the validation process and describes the activities that must be undertaken during a validation cycle.

There are also several appendices included in this document giving a more detailed description of quantitative validation.

- Quantitative validation process
- Validation Scenarios.
- Validation Techniques.

**Quantitative validation is only applicable when an engineering model exists. Quantitative validation is included in this document for information and future use.**

### 1.3 Reading guidelines

The processes in this methodology are visualised as process diagrams using standard UML Business Modelling syntax, [Eriksson, Penker 00].

The process diagram contains information about:

- **Processes and Activities.**

A process is a set of related activities and they are drawn in the centre/middle of the process diagrams with solid arrows linking processes/activities that follow one another.

- **Resources.**

Resources are the objects within the business, such as people, material, information, and products, which are used or produced in the business. Resources are manipulated (used, consumed, refined, or produced) through processes.

*Controlling resources* control or run a process. These resources are drawn above a process with a dashed line from the resource to the process.

*Supporting resources* participate in a process and are not refined or consumed. These resources are drawn below a process with a dashed line from the resource to the process.

*Input information resources* are placed to the left of the process and connected to the process with a dashed line.

*Output information resources* are placed to the right of the process and connected to the process with a dashed line.

- **Activity synchronization.**

Activities can be performed in parallel. A need for synchronization is shown in the process diagram by using a vertical synchronization bar. The activities after the

synchronization bar must wait for all the activities before the bar to complete before they can start.

A detailed process diagram contains activities and their relationships.

For more detailed information see [Eriksson, Penker 00].

## 1.4 Definitions and terms

This section presents some of the basic terms that are needed to understand this document. More acronyms can be found on the web.

[http://www.eurocontrol.int/oca/public/site\\_preferences/display\\_glossary\\_list.html](http://www.eurocontrol.int/oca/public/site_preferences/display_glossary_list.html)

### Engineering Model

A description of a system derived from a Logical Architecture that includes a view on the physical distribution of the different parts of the Logical Architecture as well as the expected performances of such distributed system (complete or partial).

### Legacy architecture

A description of a system that is in operation.

### Logical architecture

A description of a system in term of services offered and its decomposition into modules, which is technology and implementation independent.

### NFR

Non-functional requirement.

### Operational Thread

An Operational Thread is an architecture execution path that uses several modules of the architecture to validate that a User Need (Validation Aim) is met.

### QoS characteristic

Is a measurable aspect of the Quality of Service, abstracted from the means employed for measuring/monitoring it.

[Astor Final Report]

### Qualitative validation

Qualitative validation is an assessment regarding quality of a validation object.

(i.e acceptable, not acceptable)

### Quantitative validation

Quantitative validation is an assessment of a measured quantity and expressed in numbers or quantities.

### Role

A required resource with a specific responsibility and knowledge suitable for validation. A role may be assigned to one or several resources (Validation Team Members).

### UML

Unified Modelling Language, an OMG standard for visual object-oriented modelling.

### Use Case

A description of how a system can be used from an external actor's point of view.

### Validation Aim

A Validation Aim is a sentence that represents what the Stakeholders that participate in the validation process would expect to see reflected in OATA.

### Validation Aspect

In this methodology it is referring to NFR mapping assessment, NFR dependencies assessment and assessment of system ability to fulfil non-functional requirements on a Validation Model.

### Validation Model

A model designed to predict the effect on system behaviour of a certain requirement.

### Validation Objective

A statement of what the validation team will work towards to achieve the aim. A formulation of the validation aims in measurable factors.

### Validation Scenario

A validation scenario is a short statement describing a set of interactions between users and the system to be validated as well as interactions between modules of the system. The validation scenario anticipates a situation where users can face the system before its deployment.

### Validation Team

A set of roles required for the conduct of the validation methodology. Each role may be assigned to one or several resources.

### Validation Team Member

A resource which has been assigned one or more specific roles.

### Validation Technique

In this methodology it is referring to techniques like performance engineering (Queuing modelling), schedulability analysis techniques etc.

## 1.5 Bibliography

Buschmann 96	Buschmann F.;Meunier R.;Rohnert, H.; Sommerlad, P.; and Stal M. Pattern Oriented Software Architecture, Volume 1: A System of Patterns. New York, John Wiley & Sons,1996.
Bass 98	Bass L.; Clements P. and Kazman R. Software Architecture in Practice.Reading, MA.Addisson Wesley 1998.
Clements 02	Clements P.; Kazman, R.; and Klein, M. Evaluating Software Architectures. Methods and Case Studies. Addison Wesley Pearson, Indianapolis. 2002.
Eriksson, Penker 00	H-E Eriksson, M Penker, "Business Modeling with UML", 2000
Fernandez 98	Fernandez J.L., Alvarez B., García F., Perez A. and de la Puente J.A. A Case Study in Quantitative Evaluation of Real-Time Software Architectures. Ada Europe'98 Conference. Uppsala (Sweden). Springer Verlag LNCS 1411.
Fernandez 00	Fernandez J.L. and A. Monzon. A Metamodel and a Tool for Software Requirements Management. Short presentation. 5 <sup>th</sup> International Conference on Reliable Software Technologies. Postdam (Berlin). June 2000.
IEEE 88	IEEE Standards Board. Guide for the Use of the IEEE Standard Dictionary of Measures to produce Reliable Software. Institute of Electrical and Electronic

## Architecture Non-Functional Assessment Process

	Engineers. September 1988.
IEEE 92	IEEE Standard 1061-1992. Standard for a Software Quality Metrics Methodology. New York. Institute of Electrical and Electronic Engineers.1992.
ISO 91	International Standards Organization. Information Technology. Software Product Evaluation. Quality Characteristics and Guidelines for their Use. ISO/IEC 9126. 1991
ISO 98	International Standards Organization. Information Technology. Information technology-Quality of Service: Framework. ISO/IEC 13236. 1998
Klein 93	Klein, M.H., Ralya,T., Pollak, B. Obenza, R. and Gonzalez Harbour, M. A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.
Kruchten 95	Kruchten P. The 4+1 View Model Of Architecture. <i>IEEE Software</i> . November 1995.
Lee 99	Lee, J. and N.L. Sue. Analyzing User Requirements by Use Cases: A Goal-Driven Approach, <i>IEEE Software</i> , July/August 1999, 92-101.
Locke 99	Locke C.D. An Architectural Perspective of Real-Time Ada Applications. Reliable Software Technologies- Ada Europe 99. Lecture Notes in Computer Science 1622. Springer Verlag, 1999.
OATA WP 5.1	Eurocontrol, E.Isambert OATA-P2-D5.1-1 WP 5.1 "Report -- Review & Categorise Existing Non-Functional Requirements "
OATA WP 5.2	Eurocontrol, E.Isambert OATA-P2-D5.2-2 WP 5.2 "Guidelines for NFR mapping to the logical architecture",
OATA Method Overview	Eurocontrol, H. Wagemans OATA-P2-D8.1-01 Validation Methodology Overview
OATA WP 8.1.1	Eurocontrol, H. Wagemans OATA-P2-D8.1.1-01, WP 8.1 "Define Methodology for Validation within OATA - Architecture Compliance Assessment Process"
OATA WP 8.1.2	Eurocontrol, H. Wagemans OATA-P2-D8.1.2-01, WP 8.1 "Define Methodology for Validation within OATA - Architecture Structure Assessment Process"
OATA WP 8.1.4	Eurocontrol, H. Wagemans OATA-P2-D8.1.4-01, WP 8.1 "Define Methodology for Validation within OATA - Architecture Tactics Assessment Process"
Smith 02	Smith C.U. and LI. G. Williams. Performance Solutions. A Practical Guide to Creating Responsive, Scalable Software. Indianapolis, IN. Pearson Education 2002.
Sommerville 92	Sommerville I. Software Engineering. Addison Wesley. Reading, MA. 1992.

## 2 METHODOLOGY OVERVIEW

### 2.1 Introduction

This methodology is concerned with the validation of the OATA logical architecture compliance with non-functional requirements.

The logical architecture is addressing the definition of high-level services to be provided by the system in order to achieve the user requirements (e.g. use cases). As a principle, the logical architecture does not address the potential design solutions to achieve such requirements.

The early assessment of the logical architecture and the non-functional requirements, i.e. before the development phase, is therefore to achieve the following objectives:

- To assess the consistency and completeness of the specification of required services and quality of service.
- To perform on specific cases a detailed analysis of performance issues and propose recommendations.

### 2.2 Validation of non-functional requirements

User or System requirements are requirements within system scope. User or System requirements are either functional or non-functional.

Functional requirements are mainly described in OATA through actor's description, use cases and module interfaces.

Non-functional requirements (NFR) are those describing the required quality of services, for instance timeliness, capacity or other quality characteristics.

The actor description may contain an interface description whenever the actor represents an external system. The interface describes those required characteristics of individual data elements that the actor must provide, send, access, receive, etc. Priority, timing, frequency, volume and other data element constraints are also described.

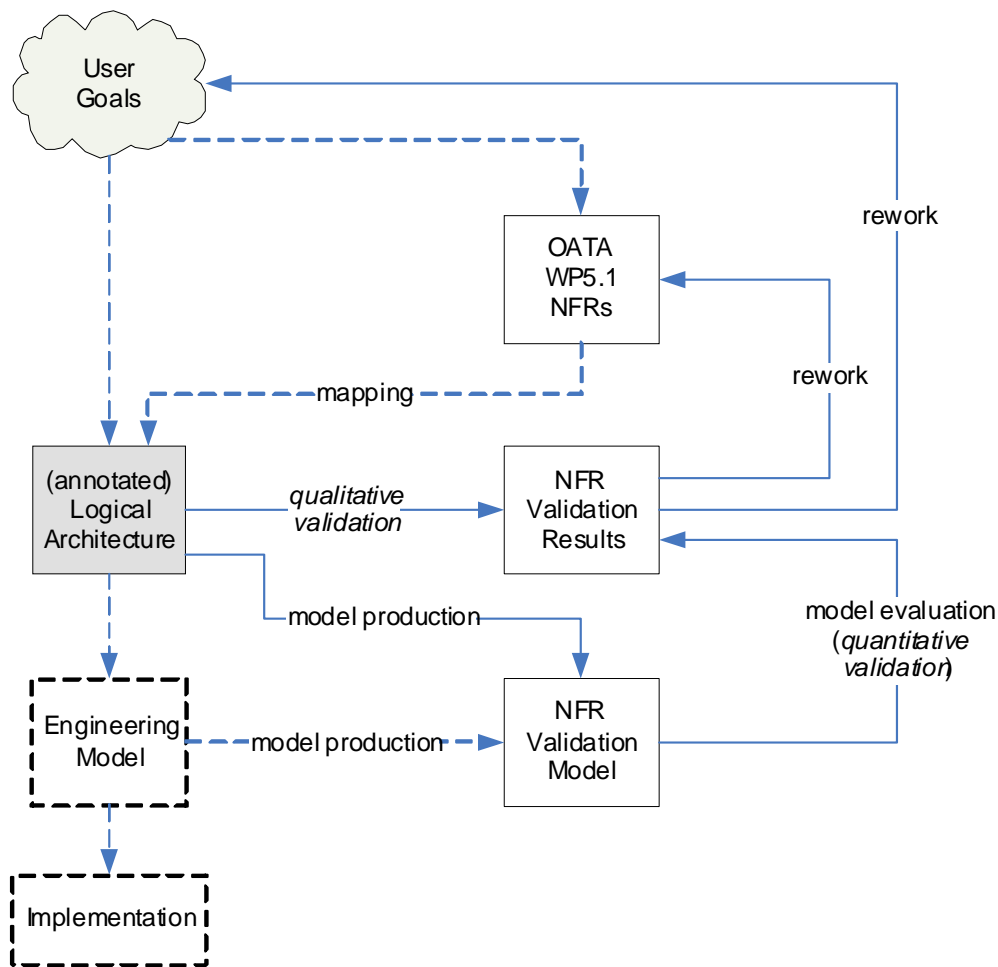
### 2.3 Validation aspects

At the level of the logical architecture it is possible to perform qualitative validation, which may include assessment of requirements documentation, dependencies and consequences.

This methodology looks at two validation aspects on qualitative validation in a logical architecture, NFR mapping and NFR dependencies.

Quantitative validation can be achieved through the use of a combination of so called operational and exploratory validation scenarios, complemented with analytical or simulation methods mainly for the efficiency assessment part of the validation process. A validation scenario is defined in such way that it is possible to assess the ability of the system to fulfil the requirement. Quantitative validation is performed on a Validation Model, which are built from an Engineering model. The Engineering model is derived from the logical architecture and includes technical information about the physical distribution and implementation.

Figure 2 gives an overview on qualitative and quantitative validation.



**Figure 2: Qualitative and Quantitative Validation**

Next chapter gives an introduction to the three validation aspects used in this methodology.

## 2.4 Overview of selected aspects

### 2.4.1 NFR mapping

The objective of the NFR mapping validation aspect is to assess the mapping of NFRs in the OATA Logical Architecture.

The mapping of NFRs in the OATA logical model is described in the document [OATA WP 5.2] and is a top down approach where each NFR is separately mapped in the architecture. There are also references in the document [OATA WP 5.1 Report] for each NFR to specific parts of the logical model where the NFR should be applied.

The validation of the NFR mapping has two objectives:

- Assessment of the mapped NFRs.
- Assessment of whether there are missing NFRs.

The first objective is to assess whether the NFRs have been correctly mapped and the second objective is to assess whether there should be additional NFRs specified, possibly uncovered NFRs in Eurocontrol specifications, which are necessary to complete the specification of the services and modules of the OATA logical architecture.

The assessment of NFR mapping is done by examination of a subset of the logical architecture. Each examination is concentrated to one or several quality characteristics (i.e. Response Time). The task is then to examine the subset and decide whether the derived quality characteristic is applicable or not in the subset and assess the NFR mapping.

According to [OATA WP 5.2], the mapping of NFRs to modules and services is established using the sequence diagrams prepared for the Use Case realisations or using the Logical Architecture Model. Therefore an activity for the selection of sequence diagrams for each Operational Thread is necessary (Identify Operational Threads, see section 3.2.7).

The quality characteristics selected for the examination/validation shall be as detailed as possible. The Validation Aims and objectives control the selection of quality characteristics. The addressed quality characteristic should be further detailed when possible.

For example if the addressed quality characteristic is Time delay. According to [OATA WP 5.1], Time delay has been further specialised into Message Transfer Delay and Response Time. The characteristics to use while examine a subset should therefore be Message Transfer Delay and Response Time.

The assessment of the NFR mapping is performed as follows:

- Identify the logical architecture subset and sequence diagrams for each Operational Thread associated to each Validation Aim (or detailed validation objective).
- Identify the quality characteristics from each Validation Aim (or detailed validation objective).
- Evaluate if specialised/derived characteristic is applicable or not to the sequence diagram.
- Evaluate whether NFR mapping is correct.
- Assess if the quality characteristics that do not have any mapped NFR, needs any additional NFR to be defined.

#### **2.4.2 NFR dependencies**

NFRs dealt by OATA have been specified separately, although they may be individually realistic and correct, by putting them together in the context of an Operational Thread they may contradict or constrain each other.

A trace of the dependencies between NFRs is used to identify the impact that one non-functional requirement has on one or more other non-functional requirements. “Depends on” traces are important in finding a logical hierarchy of the non-functional requirements. This hierarchy is used to validate the impact and consequences of dependency for each NFR.

Another trace form is “constrained by”, it is used to define how non-functional requirements may constrain, or limit, the functionality described in one or several other non-functional requirements. Constraints usually include operational and/or performance criteria which circumscribe the conditions under which the function is performed. These constraints are then used to calculate the minimum performance criteria for a functional requirement.

This methodology considers the dependency trace on Operational Threads to find impacts between non-functional requirements.

The assessment of NFR dependencies is performed as follows:

- Trace the NFR dependencies in the Operational Thread to find out the logical hierarchy of non-functional requirements (if applicable).
- Assess the NFR dependencies in the Operational Thread by using the logical hierarchy. This assessment can lead to the discovery of contradictory, restricting, missed or frequent NFRs.

- Assess the correctness and realism of the NFR value.

### 2.4.3 NFR system ability

*Validation of the system ability is not part of the main process description due to the absence of a proper OATA engineering model and is therefore described in the appendices.*

The objective of the ability of the system validation aspect is a quantitative assessment of the ability of the system to fulfil the non-functional requirements. Quantitative validation requires an engineering model to base the validation simulations on.

Depending on the NFR characteristic and the validation scenario, the most suitable validation technique shall be selected. The selected validation technique impacts the construction of the validation model. The construction of Validation Model is an external activity. To be able to run the validation model, assumptions has to be made on loads, frequencies etc. These assumptions have to be documented in the final validation report.

The assessment of system ability is performed as follows:

- Define scenarios for quality characteristics in the Validation Aims. Scenarios shall be defined as stimuli – environment – response terms.
- Select only scenarios with associated NFRs from [OATA WP 5.1].
- Prioritise the scenarios.
- Select validation technique for each scenario (i.e manual, performance engineering techniques, schedulability analysis techniques (RMA))
- Prepare guidelines for the validation modeller.
- Run Validation Model using the selected validation technique.
- Validate result.

## 2.5 General recommendations

The recommendation is to first assess NFR mapping before continuing with NFR dependency assessment and the quantitative assessment. The reason is that the NFR dependency and the quantitative assessment rely on the mapping of NFRs in the OATA logical architecture.

## 3 PROCESSES

### 3.1 Introduction

The methodology consists of three separate processes that have to be executed in a sequence since each process produces information that the following process requires.

The first process, "Preparation", is responsible for the planning of the validation cycle and that the Validation Team is well prepared and each team member is given assigned responsibilities. The purpose of the current validation and OATA architecture version that shall be validated must also be identified. The results and other information about the validation cycle shall be documented and stored in a repository. Therefore a repository must be identified or created in the beginning of the validation cycle.

The second process, "Execution", handles the actual assessment of the different aspects per Operational Thread. The results of these assessments must be well documented and stored in the repository for later analysis.

And finally, the last process to be executed, "Analysis and Reporting", validates the overall mapping of NFRs and their dependencies. The validation result shall be published in one or several Validation Reports that can be distributed to the Stakeholders and the results can be used to refine the OATA architecture and the NFRs.

An overview of the processes and their activities are shown in Figure 3.

The required information for each process is described in the process diagrams, and these diagrams also include which information that is refined or created by the processes. An overview of the information flow is shown in Figure 4.

The processes are controlled and supported by roles, information and physical resources. An overview of these resources is shown in Figure 5.

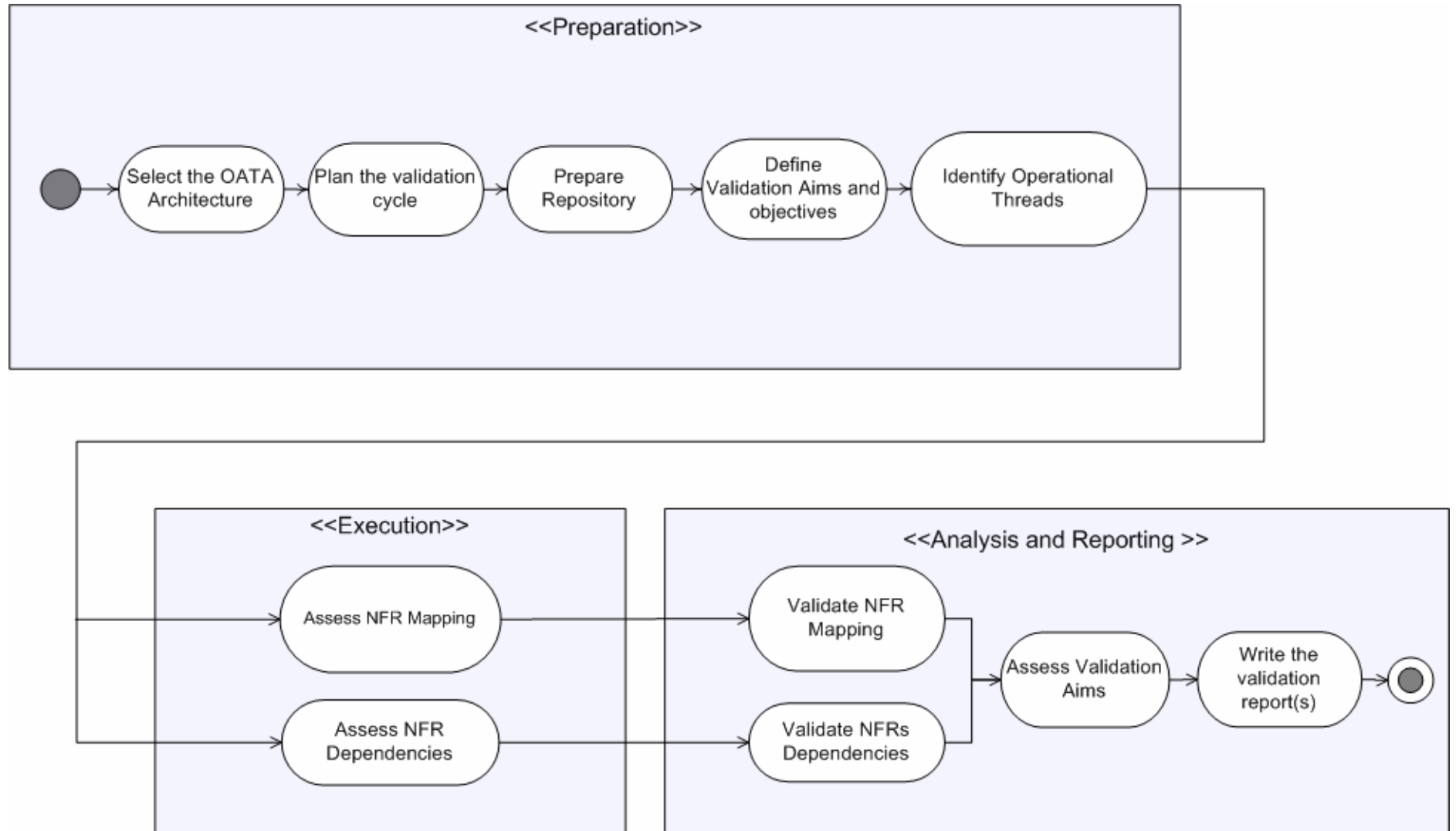


Figure 3: Process Overview

# Architecture Non-Functional Assessment Process

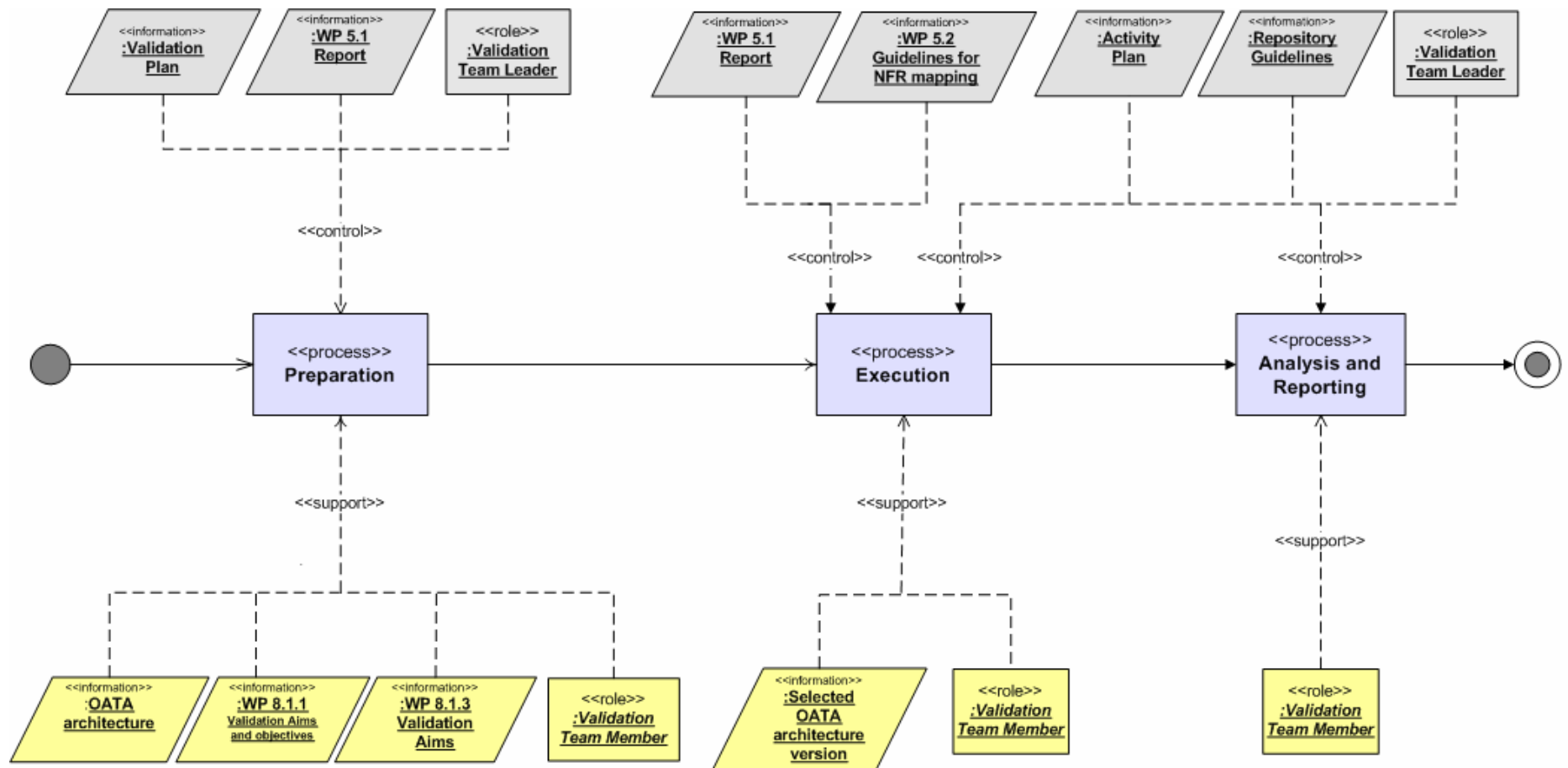


Figure 4: Resource Overview

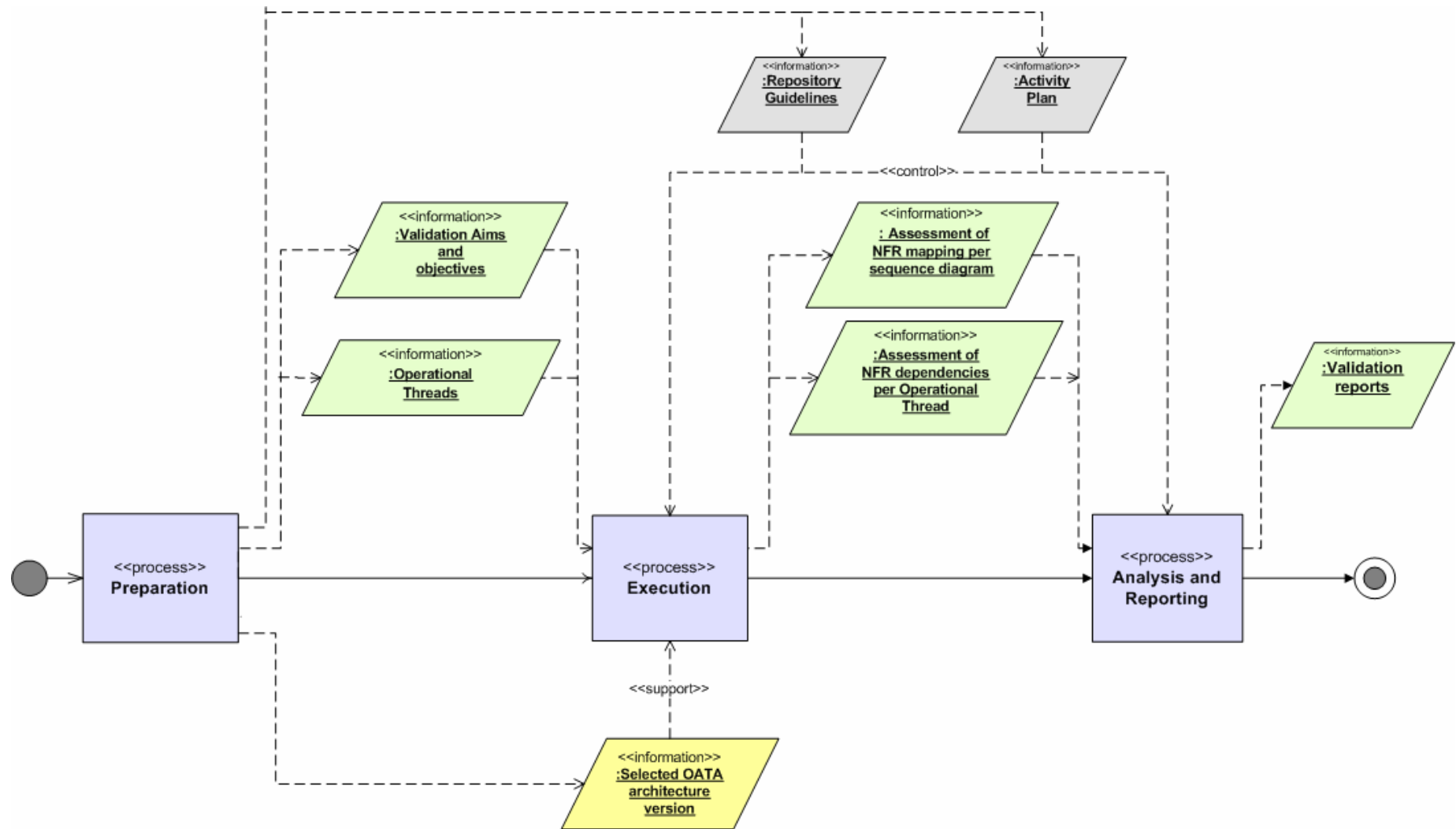


Figure 5: Information Flow Overview

### 3.1.1 Required roles

The table shows the different roles and their responsibility and requirements for the validation methodology. The described roles represent the Validation Team. Each role may be assigned to one or several resources and are referred to as the Validation Team Members.

Validation Team		
Role	Responsibilities	Requirements
Validation Team Leader	<p>Is responsible for the control of the validation process and team management.</p> <p>Is responsible of the information flow.</p> <p>Is responsible for the Validation Aims and objectives.</p> <p>Is responsible for the Activity Plan.</p> <p>Is responsible for the validation reports.</p>	<p>Good knowledge of the OATA project and organisation.</p> <p>Good knowledge of the validation methodology described in this document.</p>
Validation Expert	<p>Is responsible for quality characteristics.</p> <p>Is responsible for the performance of the workshops</p>	<p>Good knowledge of quality characteristics.</p> <p>Expert knowledge about validation and the OATA Validation Methodologies.</p>
OATA Architecture Expert	<p>Is responsible for the OATA architecture.</p> <p>Is responsible for the defined NFRs.</p>	<p>Expert knowledge about the OATA architecture.</p> <p>Full access to the OATA architecture.</p> <p>Mandate to select version to validate.</p> <p>Good knowledge of NFRs.</p>
Stakeholder	<p>Is involved in the definition of Validation Aims.</p> <p>Is involved in the validation of the NFRs with a defined value.</p> <p>Represent the Stakeholders interests.</p>	<p>Expert knowledge about user needs and expectations.</p>
Documentation Expert	<p>Is responsible for the documentation of the workshops.</p>	<p>Expert knowledge about documentation and knowledge about the methodology and the repository.</p>
Repository Expert	<p>Is responsible for the Repository and the Repository guidelines</p>	<p>Expert knowledge about the OATA Repository</p>

## 3.2 Preparation

### 3.2.1 Introduction

The objective of the “Preparation” process is to prepare the necessary information required in the “Execution” and the “Analysis and Reporting” processes. The most important activity is to define Validation Aims.

The Preparation process consists of the following activities:

- Select the OATA Architecture.  
Select a specific OATA architecture version and the subset to validate.
- Plan the validation cycle.  
Write an activity plan, assign responsibilities and prepare the team members.
- Prepare the repository  
A suitable repository to store the validation results shall be created (or identified) and prepared.
- Define Validation Aims and objectives.  
Identify Validation Aims and define non functional validation objectives.
- Identify Operational Threads  
Identify Operational Thread for each Validation Aim or objective.

The ordering of the activities, the information flow and the required resources are shown in Figure “Preparation”.

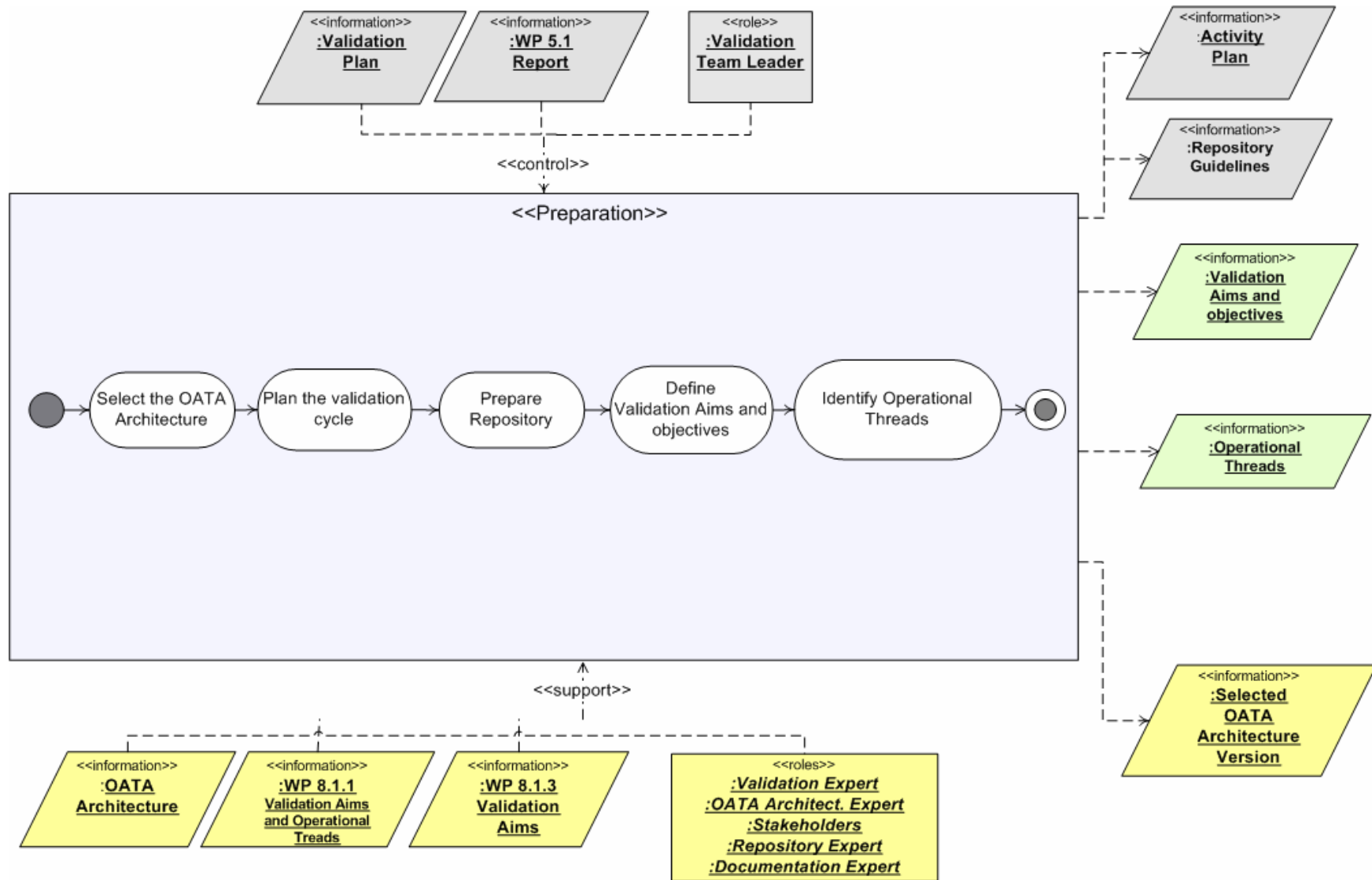


Figure 6: Preparation

### 3.2.2 Required resources

The table shows the planned resources for the activities in “Preparation”.  
Responsibility is highlighted in the table.

Activities	Roles					
	Validation Team Leader	Validation Expert	OATA Architecture Expert	Stakeholders	Repository Expert	Documentation Expert
Select the OATA Architecture			X			
Plan the validation cycle.	X	X				
Prepare the repository		X			X	
Define Validation Aims and objectives.	X	X	X	X		X
Identify Operational Threads.		X	X			X

### 3.2.3 Activity: Select the OATA Architecture

#### 3.2.3.1 Objective

The objective of this activity is to select the OATA architecture version and subset for the current validation cycle.

#### 3.2.3.2 Rationale and Context

The development of the OATA architecture is an iterative work process with regular version releases. To ensure that the OATA architecture is not changed during the validation cycle, one specific release version must be selected. This is also necessary for the traceability to the validated version when future analysis and comparisons are done. A subset of the OATA Architecture can also be selected when a validation will not be performed of the whole model.

#### 3.2.3.3 Description

The OATA Architecture Expert shall select and document the OATA architecture version that shall be validated. The selected version must be clearly identified and accessible for the Validation Team. When the OATA architecture shall be validated partly, the OATA Architecture Expert shall select and document a relevant subset of the OATA architecture. The OATA Architecture Expert shall also document the reason why certain parts have been selected for the current validation cycle.

#### 3.2.3.4 Examples and recommendations

The selected architecture version will refer to a specific OATA baseline, for example: OATA Architecture iteration 7.

The subset can be either all clusters or a set of clusters in the OATA Architecture.

### 3.2.3.5 Product

The result of this activity is a selected version of the OATA architecture for the current validation cycle and the subset to validate.

### 3.2.3.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
OATA Architecture Expert	X	X	OATA architecture.	Selected OATA architecture version. The selected OATA architecture subset

### 3.2.4 Activity: Plan the validation cycle

#### 3.2.4.1 Objective

The objective of this activity is to ensure that required resources are available and properly trained for the assignment.

#### 3.2.4.2 Rationale and Context

The Validation Team Leader should have a detailed Activity Plan, which makes it possible to achieve an effective participation of the Validation Team Members.

Before the validation process can be initiated it is important that the Validation Team Leader ensures that the Validation Team Members are available and well prepared for the validation.

#### 3.2.4.3 Description

The Validation Team Leader shall perform the following tasks:

1. Plan the activities in the current validation cycle.
2. Introduce the validation methodology to the team members. Each Validation Team Member shall understand the processes and their activities.
3. Assign responsibilities to each Validation Team Member.

#### 3.2.4.4 Examples and recommendations

It may be necessary to update the Activity Plan later during the validation cycle, i.d. when the amount of Operational Threads is known. Depending on the available time and the amount of Operational Threads the validation team may have to select a set of Operational Threads that shall be investigated further during the current validation cycle.

One major important planning issue is the detailed planning of the workshops, especially those that concerns stakeholders. The stakeholders must be noticed well in advance and also receive proper documentation. They should also be properly trained and prepared for the workshop that they shall participate in.

The defined NFRs in the [OATA WP 5.1] should be compiled in a more user friendly way that makes it possible to sort and search easily, like an Excel file or in a database.

### 3.2.4.5 Product

The expected result of this activity is:

- Well prepared Validation Team Members with assigned responsibilities.
- An activity plan for the current validation cycle.
- A set of relevant documentation including a summary

### 3.2.4.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Team Leader	X	X	Validation Plan	Activity Plan
				Set of relevant documentation
Validation Expert		X		Summary of the documentation

### 3.2.5 Activity: Prepare the repository

#### 3.2.5.1 Objective

The objective of this activity is to prepare a suitable repository for information storage. All information gathered during the validation shall be stored in the repository.

#### 3.2.5.2 Rationale and Context

A repository shall be selected or created to be used for documenting of the validation cycle. Guidelines for its usages should also be prepared and made available for the Validation Team Members.

#### 3.2.5.3 Description

The validation repository can be either a set of Word or Excel templates or a database. It is important that the repository is coordinated with the other validation methodologies in order to be able to establish common search criteria.

The **first task** is to identify existing repositories from other validation cycles (e.g. Pilot Validation). If none exists the repository has to be created. Identify suitable type of repository and collect type of information that shall be stored in the repository by examining the results from the validation.

The **second task** is to create (or identify) repository guidelines to show how it shall be used.

#### 3.2.5.4 Examples and Recommendations

A repository should be a well-structured place to store the documentation of identified architectural tactics and other important information produced during a validation cycle. It can be for example an Excel-sheet, database or a folder in a file system (or a combination).

Following criteria should be considered:

- The results must be easy to identify.
- Common information like Validation Aims shall be stored in the same way as Validation Aims from other validation aspects, e.g. functional and non-functional Validation Aims.
- The documentation of identified architectural tactics should be stored in the repository in order to facilitate analysis.

If a repository is already available (for example from previous validation iterations) for storing validation data, then this should be used. It is essential that the validation data is fully identifiable and traceable.

It is more important that it is possible to search for information from several different validations (like Validation Aims, Conclusions, Recommendations from Functional, Non Functional and Tactics) than that the repository is effective to use during the validation.

### 3.2.5.5 Product

The expected result from this activity are:

- A prepared repository for storage of the validation data results
- Repository guidelines

### 3.2.5.6 Resources

The required resources for this activity are shown in the table below.

The highlighted markings are the role responsible for the activity.

Role	Responsible	Executor	Input Information	Output Information	Physical
Repository Expert	<b>X</b>	X	None	Repository guidelines	Repository
Validation Expert		X			

## 3.2.6 Activity: Define Validation Aims and objectives

### 3.2.6.1 Objective

The objective of this activity is to select and define Validation Aims for the current validation cycle. The Validation Aims shall be decomposed in detailed objectives.

### 3.2.6.2 Rationale and Context

To ensure high quality and confidence of the validation results it is important that the selection of Validation Aims adequately reflects Stakeholders needs and expectations regarding non-functional requirements.

In order to structure the assessment of a Validation Aim, it is recommended to decompose the Validation Aim into more detailed objectives.

The Validation Aims and the objectives shall be used to guide the identification of Operational Threads and characteristics. They will be used during the analysis to make conclusions.

### 3.2.6.3 Description

The definition of non-functional validation aims is achieved by re-using functional validation aims as a basis. For each functional validation aim, the validation team shall identify the applicable quality characteristics. For each quality characteristic a validation objective is identified and used during the validation of non-functional requirements. The definition of validation aims is recommended to be performed in a workshop together with stakeholders.

The **first task** is to present functional Validation Aims from the [OATA WP 8.1.1] to the validation team. When the set of Validation Aims have been presented, the validation team shall select or prioritise the functional validation aims for the current validation cycle. This can be done by voting (as explained in the WP 8.1.1) or by picking the highest prioritised validation aim and continue with the validation. If new validation aims are defined it is important that they are selected using the same selection criteria as for the selection of functional validation aims (e.g. in scope of the OCD, 2011, the OATA Architecture).

The **second task** is to use the list of quality characteristics below and identify all characteristics that are applicable to the selected functional validation aim. For each identified characteristics a non-functional validation objective shall be defined. These non-functional validation objectives will then form the basis for the analysis and conclusions. The list below contains the quality characteristic and examples of specialised/derived characteristics.

- Time (e.g. Message transfer delay, Response time, Freshness)
- Coherence (e.g. Spatial consistency, Temporal coherence)
- Capacity (e.g. Processing capacity, Loading, Throughput)
- Integrity (e.g. Criticality, Timeliness, Situation assessment)
- Accuracy (e.g. Error rate, Position precision, Timing accuracy, Sensor accuracy)
- Priority (e.g. Absolute priority, Relative priority)
- Security (e.g. Protection, Confidentiality, Access control, Authenticity)
- Reliability (e.g. MTBF, MTTR, Continuity, Availability)

For more details see Annex 1 in [OATA WP 5.1].

In the **third task** the high-level non-functional validation aims should also be defined. This can be done by re-defining the validation aim (the expectation) with a high-level prioritised characteristic.

### 3.2.6.4 Examples and recommendations

Examples of Validation Aims addressing operational aspects:

- Example 1: Efficiency of flight:

*The system shall support the ATC Controllers to maintain the optimum sector/runway throughput.*

Technical aspects addressed by Validation Aims are for instance:

- The refinement and correctness of the OATA logical architecture specification with respect to NFR mapping.
- The correctness of the dependencies between NFRs.

The validation aims should be identified in such way that the corresponding operational threads cover the intended architecture as much as possible.

At the stage of a logical architecture, it is not expected to achieve a full-proof assessment of OATA but rather to increase the confidence in the OATA architecture from its stakeholder's perspective and to identify potential refinements to it.

### 3.2.6.5 Product

The result of this activity is a list of Validation Aims and detailed objectives for the current validation cycle.

### 3.2.6.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Team Leader	X	X	Validation Aims from [OATA WP 8.1.1] validation and/or previous iterations using this methodology.	List of Validation Aims and objectives.
Validation Expert		X		
OATA Architecture Expert		X		
Stakeholder		X		
Documentation Expert		X		

## 3.2.7 Activity: Identify Operational Threads

### 3.2.7.1 Objective

Identify Operational Threads (e.g. sequence of interactions between ATM /CNS systems) that shall be used to find corresponding sequence of interactions between modules of OATA logical architecture (modelled as UML sequence diagrams) in order to assess non-functional requirements.

### 3.2.7.2 Rationale and Context

Operational Threads are identified for each defined Validation Aim in order to validate NFRs, which are important for the Stakeholders.

### 3.2.7.3 Description

The **first task** is to identify Operational Threads for each Validation Aim. Depending on the detail level of the Validation Aim, the Operational Thread can be identified from either the Validation Aim or from detailed validation objectives.

If there are suitable Operational Threads already identified during [OATA WP 8.1.1] validation (or previous iterations), then these should be selected and re-used.

If no suitable Operational Thread exists then new threads have to be identified. New Operational Threads are identified using the guidelines in [OATA WP 8.1.1] "Identification of Operational Threads".

All Validation Aims that do not have an associated Operational Thread shall be documented.

When the sequence diagrams (described in [OATA WP 5.2]) have been used for mapping NFR it is necessary to identify the sequence diagrams associated with each Operational Thread.

The **second task** is therefore, to identify the sequence diagrams to assess when the NFR mapping is expected to be found in the UML model. First the Use Cases for each Operational Threads have to be found and then the sequence diagrams to be examined can be selected.

#### 3.2.7.4 Examples and recommendations

The validation regarding functional compliance [OATA WP 8.1.1] should be planned and performed before the validation of non functional requirements. This will make the validation of non-functional requirements more effective. This will ensure that existing Validation Aims and Operational Threads can be re-used for this validation. The overall validation will also be improved because the same Operational Thread will be validated against both functional and non functional requirements and expectations.

Validation Aims having no associated Operational Thread may occur when:

- The Validation Aim does not specify a specific interaction with the system.
  - Rephrase or remove the Validation Aim because it is impossible to make a conclusion.
- The OATA logical architecture do not cover the intended interaction with the system
  - Add to the validation report. Assess whether the interaction reflected in the Validation Aim should be covered in the OATA logical architecture.

The identification of sequence diagrams are more important when NFRs are expected to be mapped in the logical model according to [OATA WP 5.2]

When NFRs have not been mapped in the logical model (Rose model) then the mapping done in the [OATA WP 5.1] shall be validated.

#### 3.2.7.5 Product

The results of this activity are;

- A set of Operational Threads and their description associated to Validation Aims. If there are NFRs mapped in the logical model there will also be a set of sequence diagrams for each operational thread.
- A list of Validation Aims that do not have an associated Operational Thread. If the model is complete their will be no validation aims that can not be traced to the model.

The Operational Threads shall be documented as described in the [OATA WP 8.1.1]. It is important that the Operational Threads covers a complete interaction with the system. The Operational Threads can be simplified to increase the understanding and still detailed enough to identify UML elements to map the requirements on. The Operational Threads shall be uniquely identifiable and stable enough to map requirements on. Normally the non functional requirements will be mapped on the Use Case that corresponds to the Operational Thread.

#### 3.2.7.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	X	X	Validation Aims and objectives.	Operational Threads (with associated sequence diagrams if needed.)
OATA Architecture Expert		X	Operational Threads from [OATA WP 8.1.1] validation.	Validation Aims that do not have an associated Operational Thread.
Documentation Expert		X		

### 3.3 Execution

#### 3.3.1 Introduction

The objective of the “Execution” process is to collect results from the different validation aspects in order to assess the Validation Aims and objectives.

The “Execution” process consists of the following activities:

- Assess NFR mapping.
  - Assess the NFR mapping per Operational Thread.
- Assess NFR dependencies.
  - Assess the NFR dependency per Operational Thread.

Each assessment activity is based on the Operational Threads and the corresponding Validation Aims and Objectives. The Validation Team shall iterate the activities and assesses as many Operational Threads that the planned time allows. Both activities are recommended to be performed as workshops.

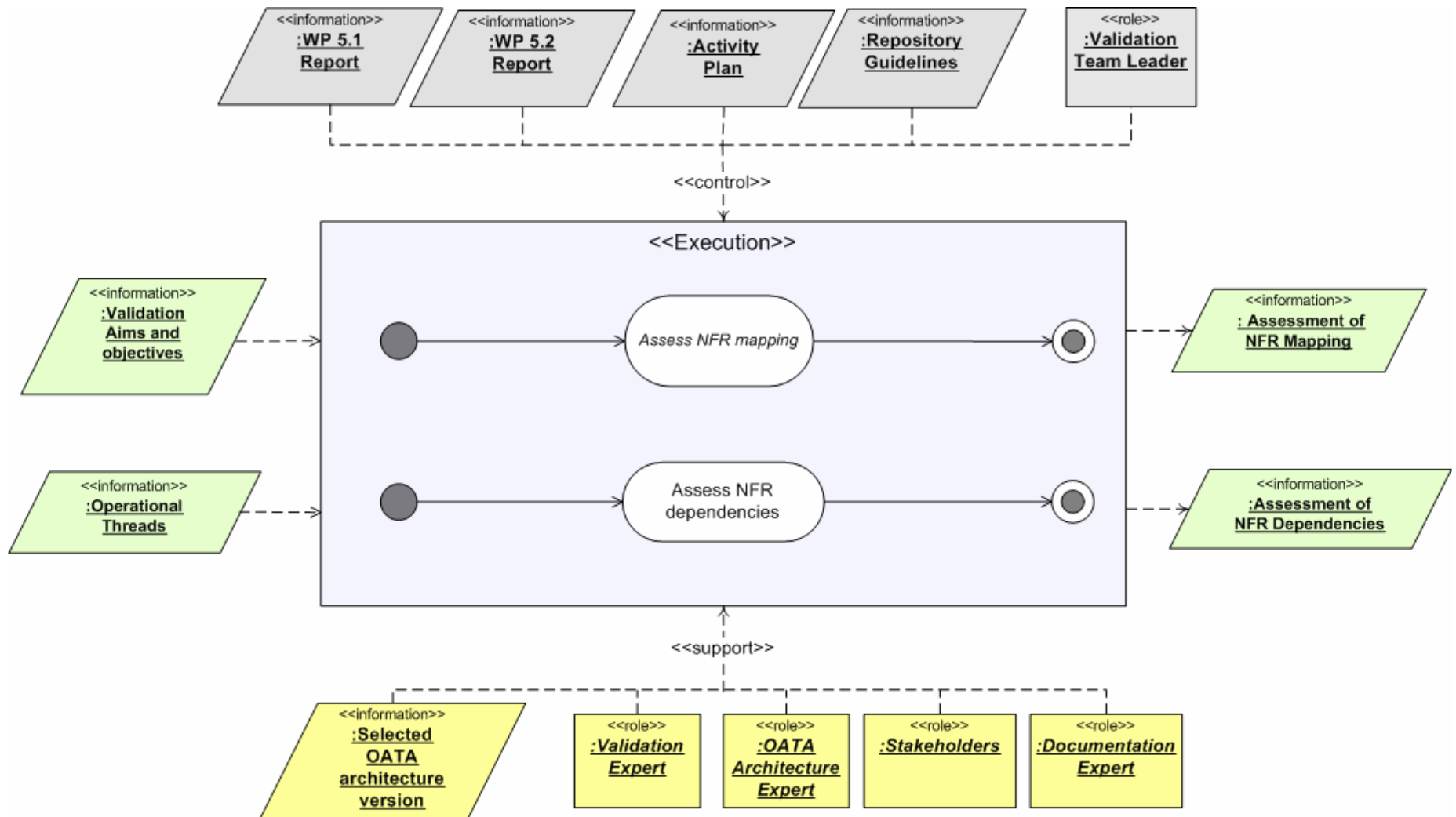


Figure 7: Execution

### 3.3.2 Required resources

The table shows the planned resources for the activities in “Execution” and the main responsible is highlighted in the table.

Activities	Roles			
	Validation Expert	OATA Architecture Expert	Documentation Expert	Stakeholder
Assess NFR Mapping.	X	X	X	
Assess NFR Dependencies.	X	X	X	X

### 3.3.3 Activity: Assess NFR mapping

#### 3.3.3.1 Objective

The objective of this activity is to identify all defined and mapped NFRs from the [OATA WP 5.1] for each Operational Thread.

#### 3.3.3.2 Rationale and Context

The identified quality characteristics for each Operational Thread shall be used to assess the mapping of NFRs in the [OATA WP 5.1]. This will make it possible to assess the correctness of the NFR mapping and to find possibly missing NFRs. The results of the assessment may lead to the discovery of missing NFRs or not mapped NFRs from the [OATA WP 5.1] and the possibility to assess the validation objectives.

#### 3.3.3.3 Description

Input to this activity is the Operational Threads, the quality characteristics from the validation aims and objectives as well as the non functional validation objectives. This activity is recommended to be performed in a workshop.

The **first task** is to list all identified quality characteristics applicable in the Operational Thread found in the activity “Define Validation Aims and objectives”.

The **second task** is for each of the specialised/derived quality characteristics evaluate if it is applicable to a single module or the whole operational thread. The Time and Capacity quality characteristics are often more suitable to map on a whole operational thread. It is important to understand the Operational Thread and what kind of services, interfaces, actions and responses that are involved for the evaluation of the applicability,

When all the Operational Threads have been examined, the result will be a set of quality characteristics per Operational Thread and an overall mapping to the OATA Architecture. Each quality characteristic and operational thread shall be checked against [OATA WP 5.1] and the NFR mapping.

The **third task** is to identify NFRs in the [OATA WP 5.1] that corresponds to each quality characteristic in the Operational Thread and connect them to corresponding non functional validation objective.

The **fourth task** is related to the quality characteristics that do not have any associated NFR. For each of these characteristics it shall be assessed if there is any NFR defined in [OATA WP 5.1] that should be associated with this characteristic. If this is the case, the conclusion is that there might be a non-mapped NFR. If there isn't any NFR regarding this quality characteristic in [OATA WP 5.1], the conclusion is that there might be a missing NFR. However, the Validation Team must evaluate if an NFR needs to be defined or if the quality characteristic is not prioritised. The evaluation result shall be documented.

### 3.3.3.4 Examples and recommendation

The defined NFRs in the [OATA WP 5.1] should be compiled in a more user friendly way that is searchable and possible to sort, like an Excel file or in a database. This should be prepared in the planning activity.

The list of missing NFRs should also include a proposal of a NFR to define regarding the characteristics and the mapping.

The assessment should not include NFRs related to the human factor since the OATA Logical model do not contain modelling of the HMI (presentation components to the user).

### 3.3.3.5 Product

The result from this activity is assessment of NFR mapping in each selected Operational Thread and a list of possibly missed NFRs.

### 3.3.3.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	X	X	Operational Threads	Assessment of NFR Mapping
OATA Architecture Expert		X	Validation Aims and objectives.	List of possibly missing NFRs
Documentation Expert		X		

## 3.3.4 Activity: Assess NFR dependencies

### 3.3.4.1 Objective

The objective of this activity is to identify all NFRs in each Operational Thread and assess the dependencies between them.

### 3.3.4.2 Rationale and Context

To assess whether NFRs are contradictory, restricting, redundant, frequent or missing, the dependencies between them must be identified in each Operational Thread.

### 3.3.4.3 Description

Input to this activity is the Operational Threads and the non functional validation objectives.

The **first task** is to identify all mapped NFRs that have an impact on the Operational Thread. This is performed by examination of the sequence diagrams that correspond to the Operational Thread and identification of all NFRs.

When all NFRs per Operational Thread are identified, the dependencies between them shall be assessed.

The **second task** is therefore to assess the dependencies taking following issues into consideration:

- Are there any NFRs that restrict any other NFR?
- Are there any frequent NFRs?
- Are there any missing NFRs?
- Are there any redundant NFRs?

The **third task** is to assess those defined NFRs that have a defined value for 2011. This task should include stakeholders. Following evaluations shall be done.

- Are the NFRs realistic and correct?
- Are there any contradictory NFRs?

### 3.3.4.4 Examples and recommendations

This activity is recommended to be performed in a workshop.

When identifying dependencies between mapped NFRs, following issues shall be considered:

- Contradictory NFRs occurs when an NFR is not possible to be fulfilled because of another NFR.
- Restricting of a NFR occurs when an NFR is restricted by another NFR due to lower requirements.
- Frequent NFRs are found by an examination of all Operational Threads. An NFR that is included in several Operational Threads is considered to be frequent.
- Missing NFRs are found by an examination of the Operational Thread and correlation with the identified characteristics and the defined validation objectives. This should be correlated with the activity of NFR mapping assessment. The list of missing NFRs should also include a proposal of a NFR to define regarding the characteristics and the mapping.
- Redundant NFRs is two NFRs with the same characteristics but two different values. The NFRs can also have different metrics/units but are in practice redundant.

Example of dependencies:

- Detailed NFR within the scope of another NFR for the whole process.  
E.g. Response Time NFR for an interaction within the scope for a process Response Time NFR.
- Modules with Capacity NFRs interacting with a cross-domain module with one or more Capacity NFR.

To make it practical and more tangible the dependency analysis should be commended on only specialised/derived quality characteristics within the same group of quality characteristics. I.e. analyse the dependencies between NFRs with characteristics Time separate from the analysis of NFRs with characteristic Capacity.

Even if there are only a few identified NFRs in an Operational Thread it is still worth the effort to make a dependency analysis to identify possible missing NFRs.

The assessment should not include NFRs related to the human factor since the OATA Logical model do not contain modelling of the HMI (presentation components to the user).

### 3.3.4.5 Product

The result from this activity is assessment of NFR dependencies in each Operational Thread. The results will be introduced in the NFR matrix maintained by [OATA WP5.1].

### 3.3.4.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	<b>X</b>	X	Operational Threads	Assessment of NFR Dependencies per Operational Thread.  List of possibly missing NFRs
OATA Architecture Expert		X		
Stakeholders		X		
Documentation Expert		X		

## 3.4 Analysis and Reporting

### 3.4.1 Introduction

The objective of the “Analysis and reporting” process is to make the final conclusions and present the results.

The “Analysis and reporting” process consists of the following activities:

- Validate NFR Mapping.
  - Assess the characteristics applicability to the Operational Threads in order to verify the mapping, assess missing NFRs or not mapped NFRs.
- Validate NFR Dependencies.
  - Assess the result from NFR dependency chains in Operational Threads.
- Assess Validation Aims.
  - Assess detailed validation objectives and make conclusions on Validation Aims.
- Write the validation report(s).

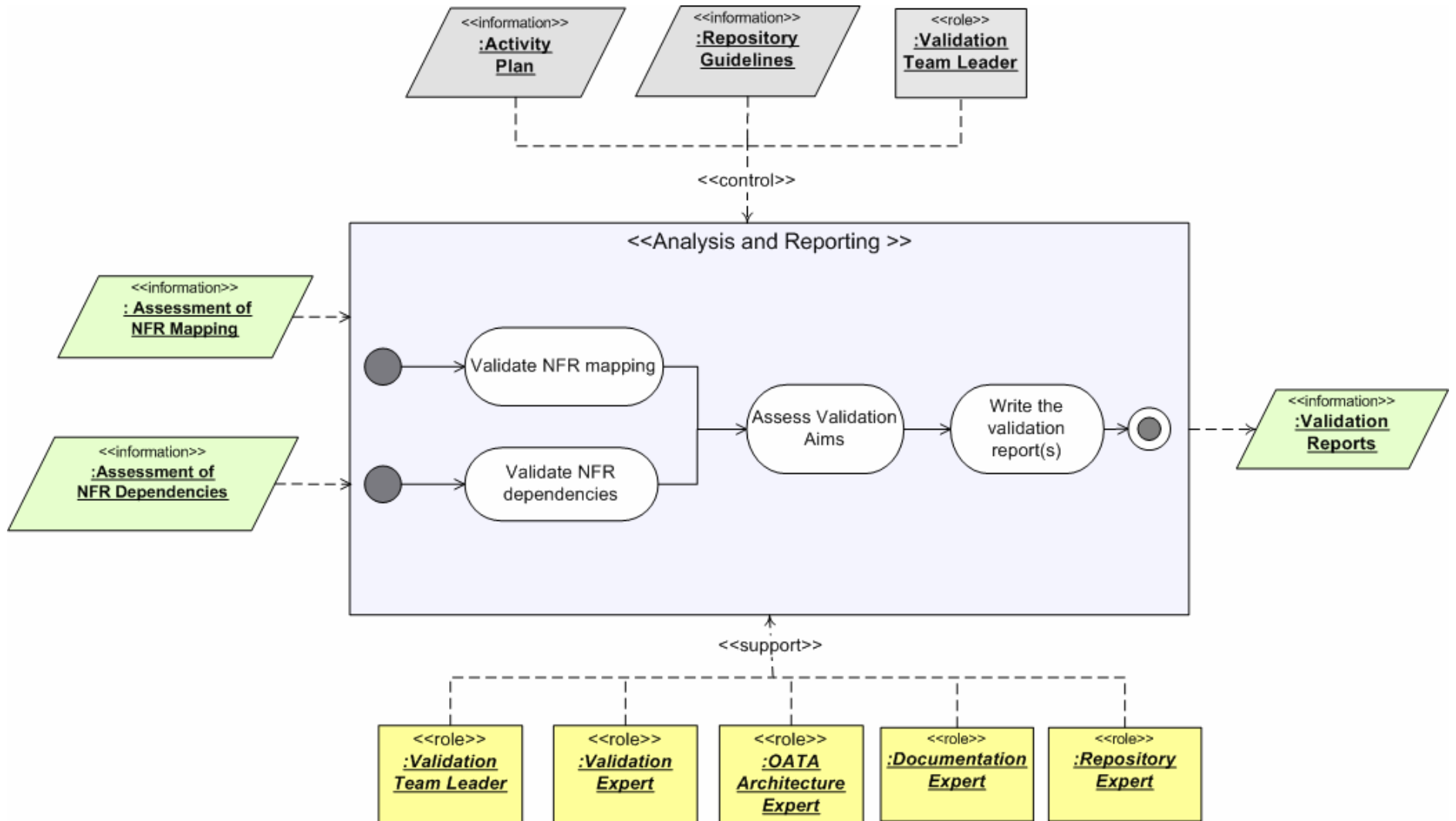


Figure 8: Analysis and Reporting

### 3.4.2 Required resources

The table shows the planned resources for the activities in “Analysis and reporting” and the main responsible is highlighted in the table.

	Roles				
	Validation Team Leader	Validation Expert	OATA Architecture Expert	Documentation Expert	Repository Expert
Validate NFR mapping.		X	X		
Validate NFR dependencies.		X	X		
Assess Validation Aims.	X	X	X	X	X
Write the validation report(s).	X	X	X	X	

### 3.4.3 Activity: Validate NFR Mapping

#### 3.4.3.1 Objective

The objective of this activity is to analyse the mapping of NFRs in the OATA Logical Architecture and compile a summary assessment.

#### 3.4.3.2 Rationale and Context

The result from NFR mapping assessment will be used to assess and refine the OATA Logical Architecture.

#### 3.4.3.3 Description

The **task** is to compile a summary assessment for NFR mapping.

The results from the activity “Assess NFR Mapping” shall be analysed in order to validate NFR Mapping in the OATA Logical Architecture.

The validation result shall be documented in the validation report as well as used during the assessment and conclusion on Validation Aims.

#### 3.4.3.4 Examples and recommendation

No information.

#### 3.4.3.5 Product

The result from this activity is a compiled assessment of the NFR mapping in OATA Logical Architecture.

#### 3.4.3.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	X	X	Assessment of NFR mapping per operational thread.	Summary assessment of NFR mapping.
OATA Architecture Expert		X		

**3.4.4 Activity: Validate NFR Dependencies**

**3.4.4.1 Objective**

The objective of this activity is to analyse the dependencies between NFRs in the OATA Logical Architecture and compile a summary assessment.

**3.4.4.2 Rationale and Context**

The result from NFR dependencies assessment will be used to refine the NFRs in [OATA WP 5.1] and to give an indication of missing, redundant or frequent NFRs.

**3.4.4.3 Description**

The **task** is to compile a summary assessment for NFR dependencies.

The results from the activity “Assess NFR Dependencies” shall be analysed in order to validate NFR Dependencies in the OATA Logical Architecture.

The validation result shall be documented in the validation report as well as used during the assessment and conclusion on Validation Aims.

**3.4.4.4 Examples and recommendations**

No information.

**3.4.4.5 Product**

The result from this activity is a compiled assessment of the NFR dependencies in OATA Logical Architecture.

**3.4.4.6 Resources**

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	X	X	Assessment of NFR dependencies per Operational Thread.	Summary assessment of NFR dependencies.
OATA Architecture Expert		X		

**3.4.5 Activity: Assess Validation Aims**

**3.4.5.1 Objective**

The objective of this activity is to assess the Validation Aims using the results for each detailed Validation Aim.

**3.4.5.2 Rationale and Context**

During preparation, all Validation Aims were defined and decomposed into more detailed objectives. These objectives have been validated from different aspects (NFR mapping, NFR dependencies) in order to make a conclusion on the Validation Aims.

**3.4.5.3 Description**

The **first task** in this activity is to collect all validation results for each detailed Validation Aim. By assessing all the collected results, the Validation Team shall make a conclusion for each Validation Aim.

The **second task** is to store the validation results in the Repository and ensure consistency in the results.

**3.4.5.4 Product**

The result from this activity is assessed Validation Aims with conclusions.

**3.4.5.5 Resources**

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Team Leader		X	Validation results.	Assessed Validation Aims and Validation Objectives with conclusions and recommendations
Validation Expert	X	X		
OATA Architecture Expert		X		
Repository Expert		X		
Documentation Expert		X		Updated Repository

### 3.4.6 Activity: Write the validation report(s)

#### 3.4.6.1 Objective

The collected information and recommendations from the performed OATA validation cycle needs to be summarized and prepared for presentation to the OATA Stakeholders.

#### 3.4.6.2 Rationale and Context

The information produced during the previous steps and activities must be structured in such a way that it is easily accessible and interpreted. Documenting and distributing the validation results precipitates two effects. First, it results in a closure of the validation by relating the final results to the initial presentation. Second, it elevates the risks that were uncovered to the attention of the management. What might otherwise have seemed to a manager like a complicated technical issue is now unambiguously identified as a threat to system qualities.

#### 3.4.6.3 Description

The validation report should be prepared during the planning phase and continuously updated to make this activity easier and more efficient. When the validation report is ready it should be sent to the validation team including Stakeholders for review. The validation report shall be updated and the reviewers should be reflected in the document history.

The following paragraphs present and describe the different areas that should be addressed by a validation report. Each one of the following items corresponds to a section of the Validation Report template used in the Pilot Validation.

Report Section	Comment	Validation Process
1. Introduction and Document Background.	Establish the information needed to understand the report. Project background, glossary, definition of terms, references, etc.	Preparation
2. Summary of validation strategy and planning	Describe the overall planning and strategy of the validation	
3. Conduct of validation exercise	Describe deviations, the overall schedule and validation team	
4. Validation results	Describe the results: <ul style="list-style-type: none"> <li>• Validation Aims</li> <li>• Validation Objectives</li> <li>• Operational Threads</li> <li>• Specific validation results</li> </ul>	Execution
5. Analysis of the validation result	Document the analysis based on the validation aims and objectives	Analysis & Reporting
6. Conclusions and Recommendations	Document conclusions and recommendations from the validation team.	
7. APPENDIX A: Detailed Validation Results and summary matrices	Synchronized with the Repository	Execution  Analysis & Reporting

### 3.4.6.4 Examples and Recommendations

It is also recommended to generate a lessons-learned report, to record the main conclusions on the validation methodology itself, regardless of the concrete data obtained in the assessment process.

### 3.4.6.5 Product

The output from this activity and the validation cycle is the Validation Report and optionally a Lessons Learned Report.

### 3.4.6.6 Resources

The required resources for this activity are shown in the table below.

The validation team leader is responsible for that a final report is produced. A template of the final report is prepared by the validation team leader and sent to the team members for its complete elaboration.

Role	Responsible	Executor	Input Information	Output Information
Validation Team Leader	<b>X</b>	X	Validation results	Validation Report
Validation Expert		X		Optional: Lessons Learned Report
OATA Architecture Expert		X		
Documentation Expert		X		
Stakeholders (review)		X		

## APPENDIX A QUANTITATIVE VALIDATION PROCESS

### 3.5 NFR system ability

The objective of the ability of the system validation aspect is a quantitative assessment of the ability of the system to fulfil the non-functional requirements. Quantitative validation requires an engineering model to base the validation simulations on.

Depending on the NFR characteristic and the validation scenario, the most suitable validation technique shall be selected. The selected validation technique impacts the construction of the validation model. The construction of Validation Model is an external activity. To be able to run the validation model, assumptions has to be made on loads, frequencies etc. These assumptions have to be documented in the final validation report.

The assessment of system ability is performed as follows:

1. Define scenarios for quality characteristics in the Validation Aims. Scenarios shall be defined as stimuli – environment – response terms.
2. Select only scenarios with associated NFRs from [OATA WP 5.1].
3. Prioritise the scenarios.
4. Select validation technique for each scenario (i.e. manual, performance engineering techniques, schedulability analysis techniques (RMA))
5. Prepare guidelines for the validation modeller.
6. Run Validation Model using the selected validation technique.
7. Validate result.

The following sections describe the recommended activities for each process to be performed.

### 3.6 Preparation

#### 3.6.1 Activity: Define and prioritise Validation Scenarios (System ability)

##### 3.6.1.1 Objective

The objective is to define validation scenarios for prioritised NFRs in order to prepare the execution of a validation model.

##### 3.6.1.2 Rationale and Context

Defining a set of validation scenarios has proven to be a great facilitator when stakeholders are gathered to participate in the validation activity.

Scenarios are used to

- Represent stakeholders' interest;
- Focus on the user requirements (functional or non-functional) that need to be fulfilled by the system in a restricted context, e.g. the phase of taxi movements from runway to stand.

##### 3.6.1.3 Description

The input to this activity is the quality characteristics associated with the Validation Aims. The quality characteristics in specified service will be the subject of validation scenarios.

The **first task** is that the stakeholders shall propose new scenarios. These scenarios should be written following the guidelines given in Appendix A.

The **second task** is that the stakeholders shall merge the scenarios that they believe represent the same behaviour or quality characteristic.

The **third task** for the stakeholders is to prioritise the scenarios. The stakeholders shall vote for the scenarios they believe are most important. Each stakeholder is allocated a number of votes and casts the votes publicly. Architecture evaluation experience tells that this exercise builds an important unity among the participants. The Validation Expert orders the scenarios by vote total and looks for a sharp drop-off in the number of votes. Validation scenarios above the line are adopted and carried out to subsequent validation activities. For example a team might consider only the top five scenarios.

The **fourth task** the Validation Team shall perform is to identify the non-functional requirements from [OATA WP 5.1] that are associated with the scenarios. This identification can also lead to the conclusion that NFRs are missing or not mapped. Only validation scenarios with associated NFR shall be selected for further validation.

### 3.6.1.4 Examples and recommendation

In this activity, the validation team asks the stakeholders to define and prioritise two types of validation scenarios: use case based scenarios and exploratory scenarios.

Use case based scenarios are extracted from the documented OATA Use Cases. They describe a significant user's intended interaction with the future running system.

Exploratory validation scenarios push the envelope and stress the system. The goal of these scenarios is to expose the limits or boundary conditions of the current OATA architecture, exposing possibly implicit assumptions. Systems are seldom conceived and architected to handle these kinds of modifications, but at some point in the future these might be realistic requirements for change, so the stakeholders might like to understand the ramifications of such changes.

Examples of validation scenarios;

- Example 1 (see 3.2.5): the scenario for the interactions between a Tower Controller and the system to replan the taxi movements of a flight in case of a delay at stand (passengers late).
- Example 2 (see 3.2.5): the scenario for the support to the opening of a military airspace to civil aviation (Flexible Use of Airspace, FUA).

### 3.6.1.5 Product

The result from this activity is prioritised validation scenarios with associated NFRs. All scenarios shall be documented and described.

### 3.6.1.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	X	X	Validation Aims and objectives.	Validation Scenarios with associated NFR(s)

Role	Responsible	Executor	Input Information	Output Information
OATA Architecture Expert		X		and descriptions.
Stakeholder		X		

**3.6.2 Activity: Select validation technique (System ability)**

**3.6.2.1 Objective**

The objective is to select the most suitable validation technique for each scenario selected in activity “Define and prioritise validation scenarios”.

**3.6.2.2 Rationale and Context**

Since quality goals against which architecture will be judged are implicitly built in the architecture, and frequently not explicitly described, mechanisms are needed to extract them. The mechanism used in this validation methodology is the quality attribute validation scenario. A scenario, as described previously, is a short statement describing an interaction of one stakeholder with the system.

Validation scenarios can be used to validate diverse quality attributes: efficiency, availability, security, etc.

The validation team members may not validate all these scenarios manually. For characteristics related to efficiency, validation scenarios are executed using mathematical techniques or modelling techniques based on queues.

Diverse techniques are presented in this architecture validation approach to execute validation scenarios. Some are performed manually, others require the building of queuing models and others require the use of mathematical formulas.

Each technique requires different inputs, resources and expertise. Based on these constraints and in the quality attribute to be evaluated, the validation team proposes the most suitable technique to test the validation scenarios.

**3.6.2.3 Description**

The validation team should classify the high priority validation scenarios taking into account some concerns that are enumerated here:

- Scenarios related to efficiency. Typically these scenarios have an associated response time that should be evaluated analytically (RMA) or by performance engineering techniques.
- Scenarios with hard deadlines should be evaluated using analytical technique such as RMA. In such a case the availability of the architecture “process view” is highly recommended.
- Scenarios dealing with physical or logical resources contention should be evaluated using either performance engineering techniques or RMA, both are compatible and may be applied independently.

In some cases more than one technique may be recommended. Sometimes the limitation of the usage of several techniques is due to the lack of budget, human experts, tools or project time constraints.

The **task** is to select the most suitable validation technique for each prioritised scenario considering the classification.

**3.6.2.4 Examples and recommendation**

No information.

**3.6.2.5 Product**

The result from this activity is the most adequate validation technique(s) for each scenario.

**3.6.2.6 Resources**

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	X	X	Validation scenarios with associated NFR and descriptions.	Validation technique per validation scenarios.

**3.6.3 Activity: Write guidelines for Validation Models (System ability)**

**3.6.3.1 Objective**

The objective is to collect information required to build a validation model and create a guideline report.

**3.6.3.2 Rationale and Context**

In order to assess whether the system will be able to fulfil the non-functional requirements, it is necessary to build a validation model of some kind. There are different kinds of validation techniques that can be used and the validation model must represent the chosen technique. The selection of validation technique has been done in activity "Select validation technique".

**3.6.3.3 Description**

To build a validation model, at least following information is required:

- Validation scenario with description.
- A selected subset of the OATA logical architecture.
- Most suitable validation technique for the validation scenario.
- Assumptions on implementation, for example distribution of modules into different system, data sharing.
- Assumptions on performance and loading, for example message load and frequency, processor capacity.

The **first** task is to collect information from previous activities and Appendix B in order to compile guidelines for each validation scenario.

The **second** task is to analyse and document the identified assumptions that have to be considered/included when running a simulation of the validation scenario and the associated NFR.

The compiled guidelines will be an important guidance when building the Validation Models.

### 3.6.3.4 Examples and recommendation

Some examples of assumptions that have to be considered:

- Traffic load/frequency.
- Processor capacity.
- Communication capacity.
- Distribution of modules over several legacy systems.
- Data sharing.

### 3.6.3.5 Product

The product from this activity is a document with guidelines for each scenario required to build the necessary validation models.

### 3.6.3.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	<b>X</b>	X	Validation scenarios.	Guidelines for building Validation Models.
			Validation technique per validation scenario.	

## 3.7 Execution

### 3.7.1 Activity: Run Validation Models (System ability)

#### 3.7.1.1 Objective

The objective of this activity is to validate the ability of the system to fulfil non-functional requirements by running Validation Models and assessing the result.

#### 3.7.1.2 Rationale and Context

Validation scenarios will be evaluated by a combination of manual (validation meetings), mathematical (RMA) and automatic techniques (Queuing modelling) selected for each validation scenario in activity “Select validation techniques”, see 3.2.8.

#### 3.7.1.3 Description

The task is to run each scenario (Validation Model) with the selected validation technique in order to assess the ability of the system to fulfil each scenario. Each scenario represents a quality characteristic and a subset of OATA architecture referred to in the Validation Aims. The result from this activity will be part of the conclusion for the associated Validation Aim.

Three different validation techniques are distinguished:

*Scenarios validated manually.*

Begin by making sure the scenario's stimulus, environment, and response are clearly stated; if not take time to put it in the proper form.

Then with the architect's help, identify the architectural approaches that are relevant to carrying out the scenario. Using architectural decisions-based analysis questions begin probing for known risks with the approach.

Try to see how the approaches affect each quality attribute of interest, including those beyond the one on which the scenario is associated. Judge the answers received as problematic or not with respect to the quality attribute goals.

When a risk, sensitivity, trade-off, or non-risk is identified, make sure the validation scribe captures it publicly. The validation document is updated consequently.

*Scenarios validated using performance engineering techniques.*

See Appendix B.

*Scenarios validated using schedulability analysis techniques (RMA).*

See Appendix B.

### **3.7.1.4 Examples and recommendations**

See Appendix B.

### **3.7.1.5 Product**

The products from this activity are:

- The list of assumptions made on the implementation of the system and covered by the validation model.
- A report describing the risks, sensitivity points, tradeoffs, and non-risks that are related to the part of the OATA architecture evaluated.

Risks are potentially problematic architectural decisions. Non-risks are good decisions that rely on assumptions that are frequently implicit in the architecture. Both should be understood and explicitly recorded.

Documenting of risks and non-risks consist of:

- An architectural decision or a decision that has not been made.
- A specific quality attribute that is being addressed by that decision along with the consequences of the predicted level of the response.
- A rationale for the positive or negative effect that the decision has on meeting the quality attribute required.

A sensitivity point is a property of one or more architecture components (and/or a component relationship) that is critical for achieving a particular quality attribute response.

A trade-off point is a property that affects more than one quality attribute and is a sensitivity point for more than one quality attribute.

### **3.7.1.6 Resources**

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	X	X	Validation Models.	Validation Model execution results.

### 3.8 Analysis and Reporting

#### 3.8.1 Activity: Validate system ability

##### 3.8.1.1 Objective

The objective of this activity is to analyse the result from the execution of validation models and compile a summary assessment.

##### 3.8.1.2 Rationale and Context

The result from system ability assessment will be used to refine the NFRs in [OATA WP 5.1] and to give an indication of critical NFRs.

##### 3.8.1.3 Description

The task is to compile a summary assessment for NFR dependencies.

The results from the activity “Run validation models” shall be analysed in order to validate the ability of OATA architecture to fulfil the non-functional requirements.

The validation result shall be documented in the validation report as well as used during the assessment and conclusion on Validation Aims.

##### 3.8.1.4 Examples and recommendations

No information.

##### 3.8.1.5 Product

The result from this activity is a compiled assessment of the ability for OATA architecture to fulfil the non-functional requirements.

##### 3.8.1.6 Resources

The required resources for this activity are shown in the table below.

Role	Responsible	Executor	Input Information	Output Information
Validation Expert	X	X	Assessment of system ability per validation scenario.	Summary assessment of system ability.
OATA Architecture Expert		X		



## 4 APPENDIX B: VALIDATION SCENARIOS

### 4.1 Concept and structure of Quality attribute validation scenarios

An architecture validation shall elicit specific quality goals, which are used to assess the quality of the architecture. This methodology proposes to use quality attribute validation scenarios to elicit the quality goals.

A quality attribute validation scenario is a short statement describing a stakeholder's interaction with the system. The quality attribute validation scenario anticipates a situation that a stakeholder can face when the system is deployed. An end-user would describe using the system to perform some task; his or her quality attribute validation scenarios would resemble the OATA use cases in object-oriented terminology. A maintainer would describe making a change to the system, such as upgrading the operating system in a particular way or adding a specific new function. A developer's quality attribute validation scenarios might focus on using the architecture to build the system or predict its efficiency.

Quality attribute validation scenarios provide a vehicle for taking vague development-time qualities such as Maintainability, and turning them into something concrete: specific examples of future uses of OATA system. Quality attribute validation scenarios are also useful in understanding run-time qualities such as efficiency or availability. This is because Quality attribute validation scenarios specify the kind of operations over which efficiency needs to be measured or the kinds of failures the system will have to withstand.

### 4.2 Types of quality attribute validation scenarios

Architecture evaluation methods, such as ATAM [Clements 02], propose the usage of three types of quality attribute validation scenarios:

- **Use Case Scenarios, here called “operational threads”.**

Operational threads are quality attribute validation scenarios extracted from the documented OATA use cases. They describe a significant user's intended interaction with the running system.

- **Growth Scenarios.**

Growth scenarios represent typical anticipated changes to the system. Each growth quality attribute validation scenarios has quality attribute-related ramifications, many of which are attributes other than Maintainability. For example, some growth quality attribute validation scenarios will have Efficiency consequences. Other may have Efficiency, security and reliability implications.

Some examples of growth quality attribute validation scenarios are:

- Double the maximum number of tracks to be handled by the system and keep the maximum latency of track data to the ATC screen to 200 ms.
- Add a new data server to reduce latency in use case scenario 5 to 2.5 seconds within one person-week of effort.
- Double the size of existing database tables while maintaining a one second average retrieval time.

- **Exploratory Scenarios**

Exploratory quality attribute validation scenarios push the envelope and stress the system. The goal of these scenarios is to expose the limits or boundary conditions of the current OATA architecture, exposing possibly implicit assumptions. Systems are seldom conceived and architected to handle these kinds of modifications, but at some point in the future these might be realistic requirements

for change, so the stakeholders might like to understand the ramifications of such changes. For example:

- Improve the system's availability from 99% to 99.999%.
- Increase the number of slot requests processed hourly by a factor of ten while keeping the worst-case response time below 5 seconds.

Each type of quality attribute validation scenarios, operational thread, growth, and exploratory; helps illuminate a different aspect of the architecture, and together provide a three pronged strategy for architecture evaluation.

### 4.3 How to build quality attribute validation scenarios

Quality attribute validation scenarios tell a very brief story about an interaction with the system from the point of view of an OATA stakeholder. It is recommended to phrase Quality attribute validation scenarios using a three-part format that helps keep the descriptions precise and to make sure that the scenario provides enough information to serve as the basis for architecture validation. The three parts are:

- **Stimulus.**

The stimulus is the part of the scenario that explains or describes what the OATA stakeholder does to initiate the interaction with the system. An end-user may invoke a function; a maintainer may make a change; a system administrator may reconfigure the system in some way; and so on.

- **Environment.**

The environment describes what is going on at the time of the stimulus arrival. What is the system state? What unusual conditions are in effect? Is the system overloaded? Is one of the processors down? Is one of the communication channels flooded? Any ambient condition that is relevant to understanding the scenario should be described. By convention, if the environment is simply "under normal conditions" is omitted.

- **Response.**

The response tells us how the system - through its architecture - should respond to the stimulus. Is the function carried out? Does the reconfiguration happen? How much effort did the maintenance change require?

The response is often the key to understanding what quality attribute the stakeholder who proposed the quality attribute validation scenarios is concerned about. If the response to an end-user invokes a function stimulus is simply that the function happens, and then the stakeholder is probably interested in the system functionality. If the stakeholder appends "with no error" or "within two seconds" to the end, that indicates an interest in reliability and efficiency, respectively.

By noticing the quality attribute of interest, the architecture evaluation leader can stimulate the stakeholder to clarify or refine the scenario if necessary. Perhaps the stakeholder is interested in efficiency but left out a statement of the scenario's environment. The architecture validation leader could ask about the ambient workload on the system to see if that played a part in stakeholder's interest. Or he could ask whether the stakeholder was interested in worst-case, average, or best-case responsiveness.

Ideally, all scenarios are expressed in this stimulus-environment-response form. It is recommended to use templates where a place for stimulus, environment and response is located. Evaluators can spend time structuring only those scenarios that the validation group selected for architecture validation.



## 5 APPENDIX C: EFFICIENCY ANALYSIS. PROPOSED TECHNIQUES

### 5.1 Efficiency analysis in the context of the Object Oriented Lifecycle

The efficiency assessment models used in software intensive systems are similar to those used for conventional performance evaluation studies. In conventional studies of already existing systems, capacity planners model the system to predict the effect of workload or hardware configuration changes. The conventional modelling procedure is as follows:

- Study the system.
- Construct a system execution model (usually a queuing network model).
- Measure current execution paths.
- Characterize workloads.
- Develop model input parameters.
- Validate the model by solving it and comparing the model results to observed and measured data for the system.
- Calibrate the model until the results match the measurement data.

Efficiency analysis for software intensive systems as OATA, can be performed in the described way. However, because the OATA software does not yet exist is the software first modelled explicitly. The software execution model represents key facets of the envisioned software execution behaviour. Its solution yields workload parameters for the system execution model that closely resembled the conventional models.

System efficiency is largely a function of the frequency and nature of intercomponent communication, in addition to the performance characteristics of the components themselves, and hence can be predicted studying the architecture of a system.

Object-Oriented systems present particular problem for efficiency analysis. The functionality of object-oriented systems is decentralized. Performing a given function is likely to require collaboration among many different objects from several classes. These interactions can be numerous and complex and are often obscured by polymorphism and dynamic binding, making them difficult to trace. Distributed systems architectures increase the difficulty of the problem.

One aspect of distributed systems that can have a significant impact on efficiency is the overhead required for communication and synchronization among distributed objects.

The object-oriented modelling notations, such as the UML used in OATA, and the associated development methods, help to reduce the impact of the above mentioned problems. Much of the information needed to perform Efficiency assessment can be captured as part of the OATA object oriented analysis and design process with a minimum of disruption.

Use cases, which are identified as part of the functional requirements definition, are a natural link between software development activities and the OATA efficiency assessment. The scenarios that describe the use cases provide a starting point for constructing the assessment models.

The techniques proposed are language independent. The models are constructed from architectural and design-level information. The execution behaviour of the software will be different with different programming languages. Nevertheless, this is reflected in the resource requirements specifications, not in the assessment model structure.

## 5.2 Efficiency Measurement

Quality Attribute	Quality Factor	Related QoS Metric
Efficiency	Time Behaviour	Resource Throughput
		Mean Service Time
		Residence Time
	Resource Utilization	Resource Utilization
		Queue Length

**Table 1: Utility Tree: Efficiency Branch**

The Efficiency Metrics proposed in this document are related to the Queuing Network Model of the system, where all system resources are characterized as queues and servers. Stage III presents a more detailed description of how Queuing Network Models can be used to assess the efficiency of the architecture. Additionally Stage III presents some performance engineering techniques as those proposed by Smith [Smith 2002],

According to the approach proposed each individual resource is considered as a service provider (server), with an incoming requests buffer (queue). Resource utilization is not performed at the same time the request is received by the server. The execution is delayed according to a resource scheduling polling.

These efficiency metrics depend on the following parameters:

- Measurement Period (T): Observation period under consideration.
- Number of Arrivals (A): Number of jobs requesting the service.
- Number of Completions (C): Number of jobs finishing service execution.
- Busy Time (B): Time required by the server to process the jobs.
- Number of Jobs vs. Time (W): Number of jobs in a time interval.

In addition to these parameters, the resource scheduling policy is also required to perform the calculations. This policy defines the way the next job in the queue is selected (e.g. FIFO, First-Input-First-Output).

### 5.2.1 Resource Utilization

This is defined as the average percentage of time that the server is busy providing service to all the jobs requested in a period of time.

$$U = B/T \times 100$$

B: Busy Time

T: Measurement Period

### 5.2.2 Resource Throughput

This is defined as the average rate at which jobs complete service.

$$X = C/T$$

C: Number of Completions

T: Measurement Period

### 5.2.3 Mean Service Time

This is defined as the average amount of time that a job spends receiving the service from the resource.

$$S=B/C$$

B: Busy Time

C: Number of Completions

### 5.2.4 Residence Time

This is defined as the average amount of time that jobs spend at the server. This time includes the waiting time (in the queue) and the service execution time (in server).

$$RT=W/C$$

W: Represents the area under the graph, Number of Jobs vs. Time (see Figure 6-1). This is calculated by adding the number of jobs at the server (resource) for each time interval analyzed.

$$W= \Sigma(\text{number of jobs})$$

C: Number of Completions

### 5.2.5 Queue Length

This is defined as the average number of jobs at the server. This includes both the waiting jobs (in the queue) and in execution jobs (in server).

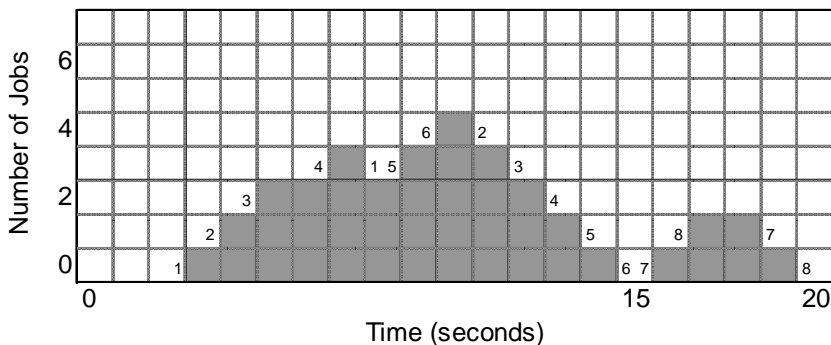
$$RT=W/T$$

W: Represents the area under the graph, Number of Jobs vs. Time (See Figure 9). This is calculated by adding the number of jobs at the server (resource) for each time interval analyzed.

$$W=\Sigma(\text{number of jobs})$$

T: Measurement Period

In order to illustrate the metrics defined above a simple example is provided below:



**Figure 9: Execution Profile**

Figure 9 represents a hypothetical execution profile for the resource modelled. Each step up in the graph represents the arrival of a job (see 1, 2, 3... on the left side of grey squares). Each step down represents the completion of a job (see 1, 2, 3... on the right side of grey squares).

The execution policy is first-come-first-served and simultaneous arrivals and completions are not allowed.

Calculated parameter *W* represents the area in grey, as grey squares represent individual job waitings.

According to these rules and the formula definitions given, the concrete values for parameters and metrics are:

Parameter	Value	Metric	Value
<b>T</b>	20 sec	<b>U = B/T x 100</b>	80%
<b>A</b>	8 jobs	<b>X = C/T</b>	0.4 jobs/sec
<b>C</b>	8 jobs	<b>S = B/C</b>	2 sec
<b>B</b>	16 sec	<b>RT = W/C</b>	5.125 sec
<b>W</b>	41 jobs-sec	<b>N = W/T</b>	2.05 jobs

**Table 2: Efficiency Metrics Values**

### 5.3 Performance Engineering Techniques

For object oriented-systems, are the general performance engineering techniques adapted to the objective of efficiency validation and the development process typically followed for object-oriented systems and to the artefacts that the development process produces.

The performance engineering process focuses on system’s use cases and the scenarios that describe them. Use cases are defined as part of the requirements definition and are refined throughout the architecture and design process. From a development perspective, use cases and their scenarios provide a means of understanding and documenting the system’s requirements, architecture and design. From a responsiveness evaluation perspective, use cases allow the identification of the significant workloads from a performance point of view, that is, the collections of system requests made by the users.

Smith proposes a performance engineering process for software intensive systems, and widely adopted by industry, in the Performance Solutions book [Smith 2002]. The process includes the following steps:

Assess efficiency risk.

Assessing the system efficiency risk at the outset of the project allows estimating how much effort to put in validation activities.

Identify critical use cases.

The critical use cases are those that are important to the operation of the system or that are important for responsiveness as seen by the end-user. The validation team considering the risks performs the selection of critical use cases. The validation team looks for use cases where there is a risk that, if efficiency objectives are not met, the system will fail or be less than successful.

Select key scenarios.

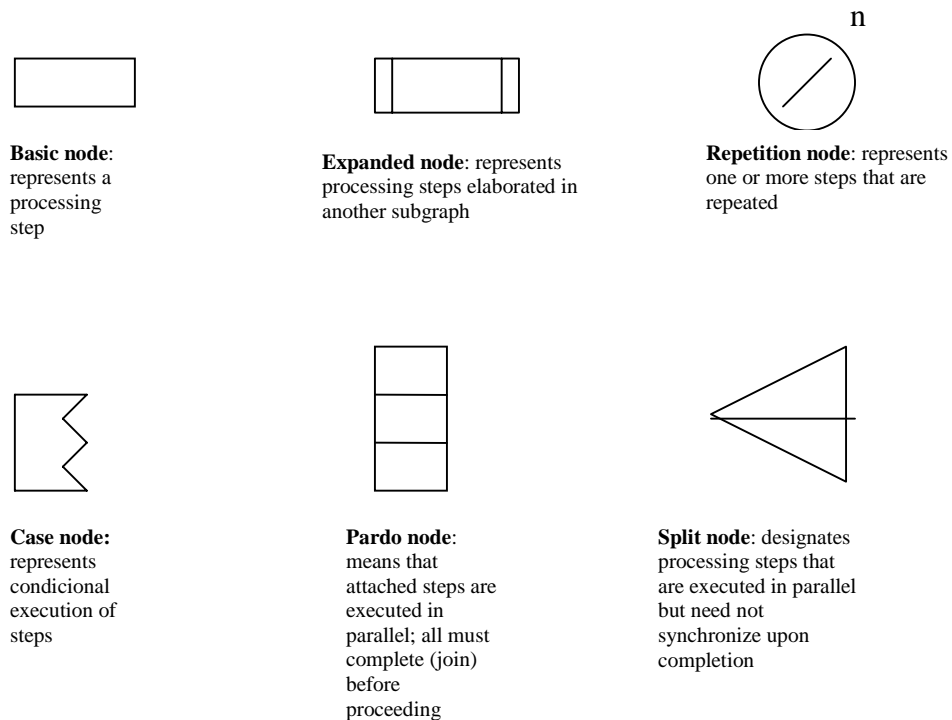
It is unlikely that all scenarios for each critical use case will be important from an efficiency perspective. For each critical use case, the key scenarios are those that are executed frequently, or those that are critical to the perceived responsiveness of the system. Each selected scenario corresponds to a workload. Scenarios may be represented by using UML sequence diagrams annotated with some timing information.

Establish efficiency objectives.

Perceived efficiency objectives and workload intensities for each selected scenario should be identified and defined. These objectives may be expressed in three primary ways by response time, throughput, or constraints on resource usage. For real-time systems, response time is the amount of time required to response to a given external event. Throughput figures are given as the number of transactions or events to be processed per unit of time. Workload intensities specify the level of usage for the scenario.

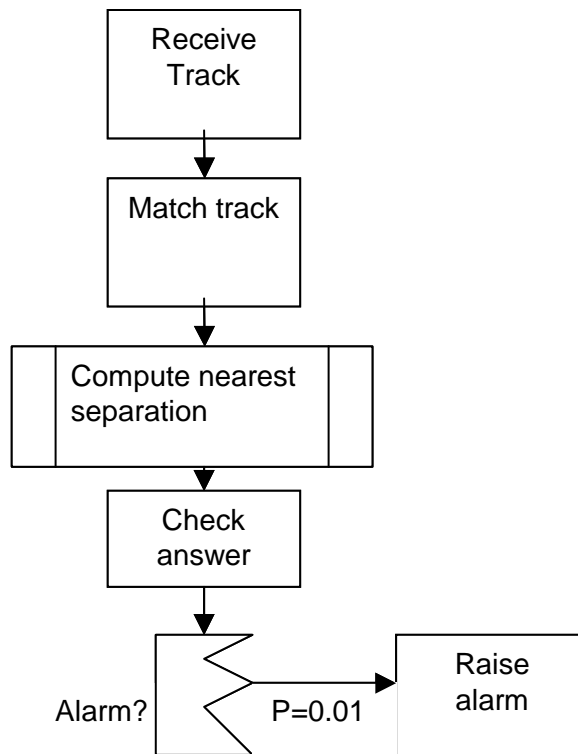
Construct evaluation models.

Execution models represented as execution graphs are used to represent software progressing steps in the efficiency evaluation model. The UML sequence diagrams representations of the Key system scenarios are translated to execution graphs (Performance Solutions Book Chapter 4 [Smith 2002]). The notation used is represented in Figure 10.



**Figure 10: Notation used in the Execution Model**

Figure 11 represents an example where an aircraft conflict detection scenario is represented as an execution graph.



**Figure 11: Execution Model Example**

Determine software resource requirements.

The processing steps in an execution graph are typically described in terms of the software resources that they use. Software resources requirements capture computational needs that are meaningful from a software perspective. For example a number of messages sent might be specified or the number of database accesses required in a processing step.

Add computer resource requirements.

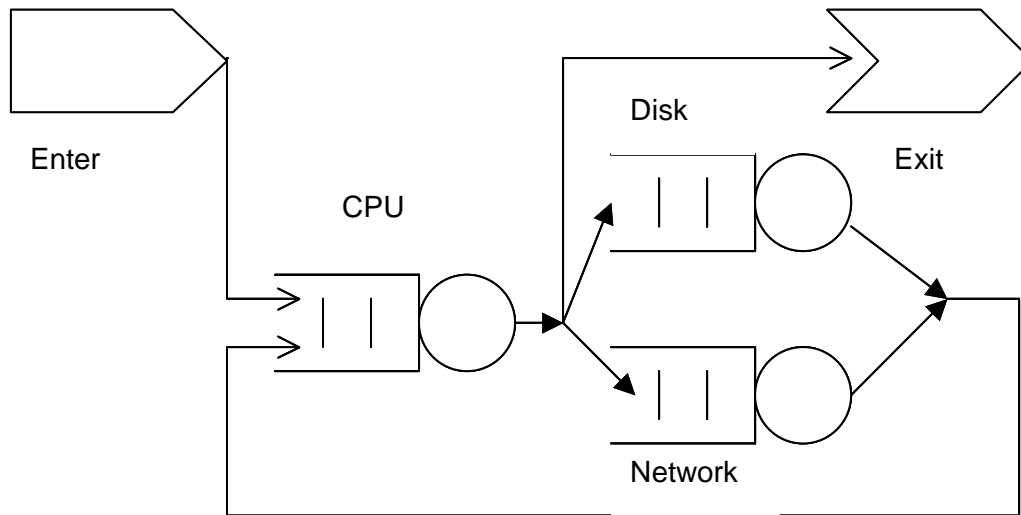
Computer resource requirements map the software resource requirements from step 6 onto the amount of service they require from the key devices in execution. Computer resources requirements depend on the environment in which the software executes. An example of computer resource requirement would be the number of CPU instructions and the disk I/Os required for database access (Performance Solutions Book, Chapter 6 [Smith 2002]).

Evaluate the models.

Solving the execution graph characterizes the resource requirements of the proposed software alone. If this solution indicates that there are no problems, the evaluation team can proceed to solve the systems execution model that is constructed based on queuing network models (figure 6-4). This characterizes the system efficiency in the presence of factors that could cause contention for resources, such as other workloads or multiple users (Performance Solutions Book, Chapter 6 [Smith 2002]).

Verify the models.

Model verification and validation are ongoing activities that proceed in parallel with the construction and evaluation of the models. Model verification is aimed at determining whether the model predictions are an accurate reflection of the system efficiency. It answers the question, "Are we building the model right?" For example, are the resource requirements that we have estimated reasonable?



**Figure 12: Queuing Network Model Example**

Validate the models.

Model validation is concerned with determining whether the model accurately reflects the execution characteristics of the system. It answers the question, “Are we building the right model?” We want to ensure that the model faithfully represents the evolving system. It is particularly important to detect any model omissions as soon as possible.

Both verification and validation require measures. In cases where performance is critical, it may be necessary to identify critical components, implement or prototype them early in the development process, and measure their performance characteristics. The model solutions help identify which OATA components are critical for efficiency perspective.

At each development phase, validators and software architects refine the models based on the increase knowledge of the details in the design.

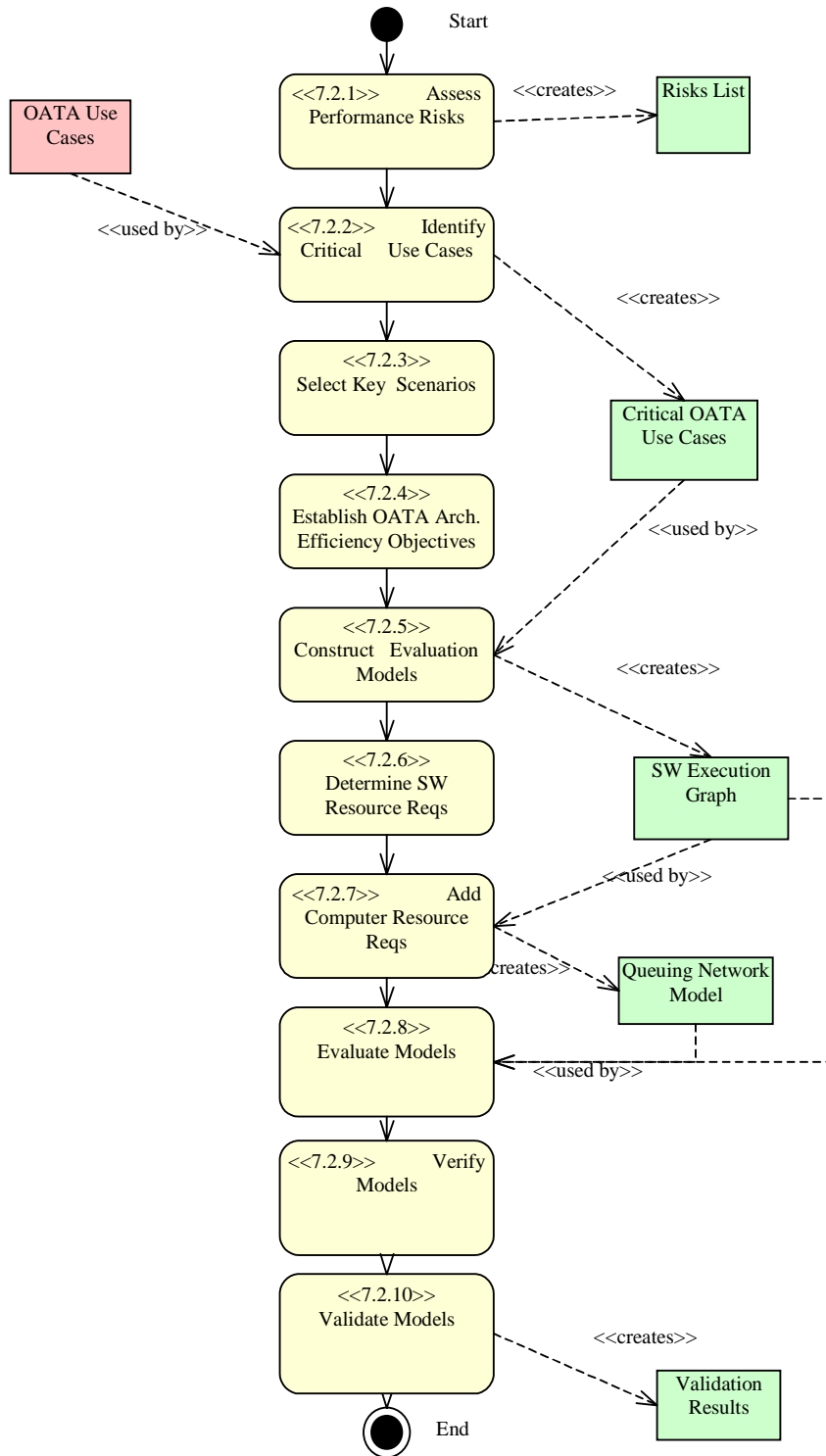


Figure 13: Scenarios Validated using Performance Engineering Techniques

#### 5.4 Schedulability Analysis Techniques. Rate Monotonic Analysis.

Schedulability is concerned with determining whether certain critical tasks in a multitasking system will meet their deadlines, ever under conditions where the software is temporarily overloaded.

There are diverse categories of timing expectations. Each category provides a different degree of rigor regarding expected system Efficiency. More important each category utilizes different infrastructure and communication approaches. The architectural models and

techniques appropriate for use in one of these categories are usually inappropriate for the others.

Previous section describes those models and techniques categorized as statistically predicted architectures, where statistical characterizations indicate average response time and related standard deviation. The techniques employed include queuing and synchronous and asynchronous messaging and the associated delays estimation.

Architectural approaches resulting in systems that exhibit guaranteed bounded latency, see section 4.1, and include shared resources. Shared resources are entities that are locked by client processes. As such, they exhibit protected regions of use. In conjunction with fixed priority scheduling, real time operating systems kernels permit the use of analytical techniques such as Rate Monotonic Analysis (RMA). Such systems may be analyzed and may therefore be guaranteed to possess upper bounded latency response times to specified stimuli.

Rate Monotonic Analysis (RMA) consists of a set of techniques for analyzing and guaranteeing that the executable threads within a system including periodic and a periodic activities – will be completed before their required deadlines. RMA is an analysis rather than a simulation or modelling technique. For complex systems, obtaining the scenario that embodies the worst-case performance stress upon a system is extremely difficult. Simulated execution of such a simulation case usually involves significant computing resources.

In 1982, a joint research initiative between IBM and Carnegie Mellon University (CMU) began, using Liu and Layland's theory. In 1988, the CMU Software Engineering Institute (SEI) began work on the theory and extended the methods to address a broader range of systems. Today RMA is useful on all real-time systems and has been applied by several organizations in development efforts: Boeing, GE, General Dynamics, Honeywell, IBM, Magnavox, Mitre, NASA, Naval Air Warfare Centre at China Lake, and Paramax. RMA principles have also been adopted in the standards of Ada 95, Futurebus+, and POSIX. RMA has been also applied for real-time systems developed in Europe [Fernandez 98].

RMA gives designers tools to mathematically guarantee that, when the system is deployed and running, critical deadlines will always be met, even in worst-case situations. RMA can also help ensure that requirements on soft deadlines will be met. In addition to helping troubleshoot performance problems in existing systems, RMA gives designers a scientific approach to identifying potential timing problems even before the system is built. With the use of RMA, timing problems can be identified more easily and with greater certainty than has previously been possible.

RMA can be compared to the use of static force equations to ensure the ability of a bridge to hold necessary loads. In particular, RMA will identify sources of time blocking.

Essentially every real-time system has time blocking caused by priority inversion. It is insidious priority inversion, like the often-cited case of a background task running while high-priority tasks are waiting, which can break a system's performance. An example of priority inversion is where a high-priority and low-priority task shares a data structure, such as a look-up table.

Suppose an air traffic control system has three tasks. One checks airplane paths and predicts collisions. This task has a very high priority. Another refreshes the controller's screen periodically and has a medium priority. A third low-priority task updates the data on each airplane's location. Clearly the collision checker cannot be accessing location information while the location updater is changing the information.

To prevent this, the collision checker and the location updater share the look-up table through a flag or semaphore that locks out the other task when the data is being accessed or changed.

During operation, a time can come when the low-priority task, the location updater, is running and reaches its critical section of writing to the look-up table. At that point, it sets the

semaphore and prevents any other task from using the shared resource. It is possible that, while the low-priority task has the look-up table locked, the collision checker task will need to run. The collision checker takes precedence, of course, and runs until it needs to read the table, which has been locked out by the low-priority task, the location updater. At this point, the location updater starts up again because higher priority tasks are either suspended or not running. But now the medium-priority task, the screen refresher, is ready to run. Since it has a higher priority than the location updater, and the collision checker is suspended, the screen refresher takes precedence and is allowed to run. It has no need for the shared resource, the look-up table, so it runs carefree to completion. There can be any number of such medium-priority tasks running concurrently. They can all prevent the lower priority task from finishing its critical section in the look-up table and in turn prevent the high priority task from completing its execution. In the meantime the collision checker is unable to do its work and meet its deadline. Applying RMA exposes this potentially fatal problem.

The basic priority inheritance protocol, which was developed as part of the work on RMA ensures that this sort of unbounded priority inversion cannot occur.

The concept of priority inversion is one that makes RMA so important in real-time systems. Hardware that is geared toward high throughput can cause unnecessary priority inversion. Because they are efficient, first-in-first-out (FIFO) queues are commonly used; however, they also cause priority inversion. RMA helps to identify these sources of priority inversion, also known as “blocking”. The analysis focuses on evaluating the pre-emption, execution, and blocking that affect each task’s ability to meet its deadlines in worst-case conditions. This information then provides designers with guidance for improving and ensuring the Efficiency of the system.

This section reviews briefly the fundamental principles and techniques of RMA. It’s contents come mainly from the SEI’s RMA Handbook [Klein 93]. For the sake of brevity, RMA mathematical formulas are not described here. The pace of the overview is somewhat rapid.

The following represents a distillation of the key principles of rate monotonic analysis.

These principles are provided here to give some quick rules of thumb for applying RMA. When possible, the entries are included with a reference to sections in the “RMA Handbook “[Klein 93] that cover the topics in greater depth.

### **Aperiodic events**

Aperiodic events should be cast into a periodic framework. This can be done either by exploiting natural limits on the aperiodic arrival pattern (such as a known minimum interarrival interval) or by using aperiodic server algorithms such as the sporadic server algorithm. (RMA Handbook Chapter 5, Group 3).

### **Deadlines**

Deadlines should be used as a guide for assigning priorities to responses: in some cases, the shorter the deadline, the higher the priority. (RMA Handbook Chapter 5, Group 1).

### **Delays**

Determine all factors that can delay the response to an event. Common factors include: tasks that can execute at higher priority, high-priority interrupts, non-preemptible sections, masked interrupts, waiting to lock semaphores, and waiting to use server tasks.

### **Distributed systems**

Analyze a distributed system by decomposing responses into a sequence of events in which each event uses a single resource. The sum of the response times on each resource is equal to the end-to-end distributed response. (RMA Handbook Chapter 6, Group 6).

### **FIFO queues**

Carefully manage or avoid FIFO queues. FIFO queues can exacerbate priority inversion by forcing a high-priority response to wait for lower priority responses that are ahead in the queue. Furthermore FIFO queues often cause unbounded priority inversion. (RMA Handbook Chapter 6, Group 6).

### **Interrupts**

Be aware that interrupts occur as a part of most responses: for example, to initiate a low-priority task, or to signal the occurrence of a background event. (RMA Handbook Chapter 5, Group 1 and Group 3).

### **Operating systems**

Account for the effects of operating systems by including operating system events and actions in the analysis. Also, account for limitations in representations of critical information such as time. (RMA Handbook Chapter 7).

### **Performance tracking**

Manage the performance of a system as you would manage any precious resource. Start tracking system performance early in system design. Establish performance budgets and use execution estimates from previous experience. Use performance budgets in conjunction with schedulability analysis to obtain performance estimates. Refine estimates and budgets as the design process progresses.

### **Priority inheritance**

Use priority inheritance to help in minimizing priority inversion. Priority inheritance can be invoked conditionally (when priority inversion occurs) as in the basic priority inheritance protocol. It can also be invoked unconditionally, as in the highest locker protocol. Another option is the priority ceiling protocol. For more information on synchronizations that invoke priority inheritance, see [Klein 93]. (RMA Handbook Chapter 5, Group 2).

### **Priority inversion**

Be aware of priority inversion; some priority inversion is unavoidable. When a fixed priority scheduling policy is being used, priority inversion occurs when a lower priority response is using a non-preemptible resource (such as a data object) while a higher priority response is waiting to use the resource. Delays caused by priority inversion must be included in the schedulability analysis of your system. (RMA Handbook Chapter 5, Group 2).

### **System resource**

Analyze the use of all system resources such as CPUs, local area networks (LAN), backplane buses, input/output controllers, etc. All resources that are used within the response to an event can affect the timeliness of the response. Scheduling policies, arbitration mechanisms, communication protocols, and queuing disciplines must be considered. (RMA Handbook Chapter 6, Group 6).

### **Unbounded priority inversion**

Unbounded priority inversion is intolerable and can compromise the integrity of an entire system. It can occur when a high-priority and low-priority responses share a non-preemptible resource and there are medium-priority response(s) that do not use the resource. The medium-priority responses can prolong the amount of time that the resource is locked by the low-priority response. Several synchronization protocols are available to prevent unbounded priority inversion. (RMA Handbook Chapter 5, Group 2).

### **Utilization and schedulability**

CPU utilization by itself is not a reliable measure of schedulability. Low levels of CPU utilization do not necessarily guarantee schedulability. High levels of CPU utilization do not necessarily prohibit schedulability. (RMA Handbook Chapter 4, Group 1).

### **Utilization and spare capacity**

CPU utilization by itself is not a reliable measure of spare capacity. Low levels of CPU utilization do not necessarily indicate a high level of spare capacity. Several techniques are available in [Klein 93] to determine spare capacity. (RMA Handbook Chapter 4, Group 3).

Rate Monotonic Analysis is based on an event-response framework, and is carried out through several phases.

Describe real-time situations that apply.

The first and most difficult part of the process is to identify all of the events in the system. It is very important to start at the system boundary and define external events and then add only the necessary internal events. A response is the processing that results from an event sequence, i.e., the set of all possible paths.

Estimate the execution time of actions.

An action is a computation within which no resource allocation decisions are initiated by the action itself. Resource allocation decisions are initiated by such things as changes to priority, invoking another task, starting I/O, masking interrupts, or suspending a task.

Build the Implementation Table.

The table is broken into 4 areas, events, responses, actions and resources. The process of refinement of requirements into architecture should provide all the information required to fill the implementation table. The table can represent a higher level cut at the implementation. For example, execution times might be the estimates of step2.

Build the Techniques Table.

The techniques table is a huge simplification of the implementation table. The idea is to generate a set of parameters that still describes the architecture but restricts the assumptions to conditions where proven mathematical reasoning can be brought to bear.

Analyze the situation to determine if timing requirements are met.

Results of schedulability analysis of the system architecture are presented.

In order to apply the RMA to the system to be evaluated, are the events first identified which drive the system behaviour, and the responses to them. The responses are decomposed into actions which use different resources. The worst case execution times of the actions are estimated. From this information, summarized in the so-called 'Implementation table', a set of analysis techniques can be applied which give information on the timing properties of the system [Fernandez 98].

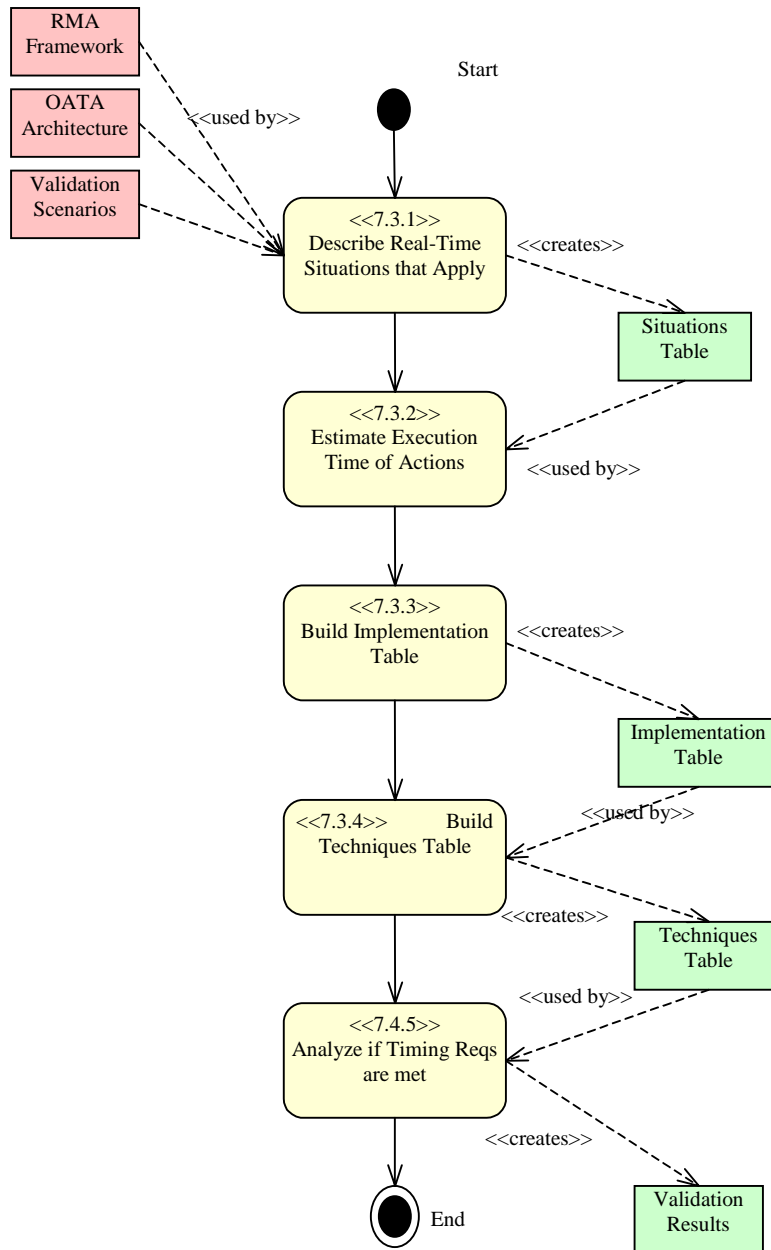


Figure 14: Scenarios validated using schedulability analysis techniques (RMA)