**for**

**Company XX**

Document history

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 27.11.2002 | 0.92 | Updated template version | FOH/IHA |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Template User guide:**

* **To be deleted in the final document**

This Template is a highly modified version of a RUP template (Rational Unified Process) for a software architecture document. The modifications includes elements from the Danish Structured Development Handbooks (SPU) SW documentation guide. The template is based on the famous “4+1” View Model described by Philippe Kruchten, “The 4+1 View Model of Architecture”, IEEE Software Nov.1995.

The Template is written as a **Microsoft Word .dot** file. A double click with the mouse on the xx.dot file in the file manager will open a new word document with the default name: Document 1, including the information in the template.

Start with modifying the document title and subject. Document title and subject (e.g. System/product <name>) is modified in **File/Properties** – **Summary**. Title and subject will automatically be inserted on the front page and in the page header. Save the document with a new name.

If the **.dot** file is installed in the directory where word looks for template files, it will be possible to activate a new document based on this template by selecting **File/New** from the Word program.

For helping the user with writing the document, help text is included for each section in the document. This help text is written as “Hidden Text” in Word.

To show (enable) the help text on the screen:

Menu choice: **Tools/Options – View**

and check the check box: Hidden text (alternatively remove the mark – for removing the help text).

To print the template file with help text:

Menu choice: **Tools/Options – Print**

and check the check box Hidden text.

To modify the template:

Start Word and open the template by selecting: **File/Open**

Table of Contents

[1. INTRODUCTION 3](#_Toc26171444)

[1.1 Purpose and Scope 3](#_Toc26171445)

[1.2 References 3](#_Toc26171446)

[1.3 Definitions and acronyms 3](#_Toc26171447)

[1.4 Document structure and reading guide 3](#_Toc26171448)

[1.5 Document role in an iterative development process 3](#_Toc26171449)

[2. SYSTEM OVEVIEW 3](#_Toc26171450)

[2.1 System context 3](#_Toc26171451)

[2.2 System introduction 3](#_Toc26171452)

[3. SYSTEMET INTERFACES 4](#_Toc26171453)

[3.1 Interface to human actors 4](#_Toc26171454)

[3.2 Interface to external system actors 4](#_Toc26171455)

[3.3 Interface to hardware actors 4](#_Toc26171456)

[3.4 Interface to external software actors 4](#_Toc26171457)

[4. Use Case View 4](#_Toc26171458)

[4.1 Overview of architecture significant Use cases 4](#_Toc26171459)

[4.2 Use case 1. scenarios 5](#_Toc26171460)

[4.2.1 Use case goal 5](#_Toc26171461)

[4.2.2 Use case scenarios 5](#_Toc26171462)

[4.3 Use case 2. scenarios 5](#_Toc26171463)

[5. Logical View 5](#_Toc26171464)

[5.1 Overview 5](#_Toc26171465)

[5.2 Architecturally significant design packages 6](#_Toc26171466)

[5.2.1 Package 1 6](#_Toc26171467)

[5.2.2 Package 2 6](#_Toc26171468)

[5.3 Use case realizations 6](#_Toc26171469)

[5.3.1 Use case 1. realization 6](#_Toc26171470)

[5.3.2 Use case 2. realization 6](#_Toc26171471)

[6. Process/task View 7](#_Toc26171472)

[6.1 Process/task overview 7](#_Toc26171473)

[6.2 Process/task implementation 7](#_Toc26171474)

[6.3 Process/task communication and synchronization 7](#_Toc26171475)

[6.4 Process group 1. 7](#_Toc26171476)

[6.4.1 Process communication in group 1 7](#_Toc26171477)

[6.4.2 Process 1. description 7](#_Toc26171478)

[6.4.3 Process 2. description 8](#_Toc26171479)

[6.5 Process group 2. 8](#_Toc26171480)

[7. Deployment View 8](#_Toc26171481)

[7.1 System configurations overview 8](#_Toc26171482)

[7.2 System configurations 8](#_Toc26171483)

[7.2.1 Configuration 1. 8](#_Toc26171484)

[7.2.2 Configuration 2. 8](#_Toc26171485)

[7.3 Node descriptions 8](#_Toc26171486)

[7.3.1 Node 1. description 9](#_Toc26171487)

[7.3.2 Node 2. description 9](#_Toc26171488)

[8. Implementation View 9](#_Toc26171489)

[8.1 Overview 9](#_Toc26171490)

[8.2 Component descriptions 9](#_Toc26171491)

[8.2.1 Component 1 9](#_Toc26171492)

[8.2.2 Component 2 9](#_Toc26171493)

[9. Data View 9](#_Toc26171494)

[9.1 Data model 10](#_Toc26171495)

[9.2 Implementation of persistence 10](#_Toc26171496)

[10. Generel design decisions 10](#_Toc26171497)

[10.1 Architectural goals and constraints 10](#_Toc26171498)

[10.2 Architectural patterns 10](#_Toc26171499)

[10.3 General user interface design rules 10](#_Toc26171500)

[10.4 Exception and error handling 11](#_Toc26171501)

[10.5 Implementation languages and tools 11](#_Toc26171502)

[10.6 Implementation libraries 11](#_Toc26171503)

[11. SIZE and performance 11](#_Toc26171504)

[12. Quality 11](#_Toc26171505)

[13. Compilation AND LINKING 12](#_Toc26171506)

[13.1 Compilation-hardware 12](#_Toc26171507)

[13.2 Compilation-software 12](#_Toc26171508)

[13.3 Compilation and linking process 12](#_Toc26171509)

[14. INSTALLATION AND Executing 12](#_Toc26171510)

[14.1 Installation 12](#_Toc26171511)

[14.2 Executing-hardware 12](#_Toc26171512)

[14.3 Executing-software 12](#_Toc26171513)

[14.4 Execution-control (start, stop and restart) 12](#_Toc26171514)

[14.5 Error messages 13](#_Toc26171515)

[15. Appendices 13](#_Toc26171516)

# INTRODUCTION

## Purpose and Scope

Describes the purpose of the **System Architecture Document** in the overall project documentation. Describes the target group for the document with an indication of how they are expected to use the document. Include a brief description of the scope for the System Architecture Document e.g. is it a system or a stand alone product.

## References

A list of relevant reference documents, this could be as well external as internal documents.

## Definitions and acronyms

A list of definitions and acronyms that are relevant for reading and understanding the document.

## Document structure and reading guide

A brief description of the document structure and introduction of the “4+1” view organization consisting of: **Use-Case, Logical, Process, Deployment,** and **Implementation Views.**

**It can be recommended to keep a standard structure for this type of document and insert a notice, if a certain section isn’t relevant for the actual system or product.**

The reading guide can be organized based on the needs from the different target groups.

## Document role in an iterative development process

A description of how the document is developed in an iterative process i.e. when and how to fill out the different document sections.

# SYSTEM OVEVIEW

This section shall give a short overview of the system and its environment.

## System context

The context of the system can be illustrated by informal diagrams and/or with one or more actor-context diagrams showing the system and it external actors.

**Diagrams: Informal overview diagrams and UML Actor-context diagram (from requirement document).**

## System introduction

A short description of the purpose and main functionality of the system described with reference to the system context.

# SYSTEMET INTERFACES

This section describes each of the interfaces to the external actors.

## Interface to human actors

Describes the Man Machine Interface (MMI) for the human actors interacting with the system.

The MMI is described for each human actor role or alternatively insert a reference to a MMI specification document.

## Interface to external system actors

Describes the interface for actors representing external systems communicating with the System under development.

Describe the used communication protocol for each external system actor or alternatively insert a reference to a protocol specification document e.g. an international protocol standard.

## Interface to hardware actors

Describes the interface for actors representing the hardware devices connected to the system.

Describe the specific hardware interface for each hardware actor or alternatively insert a reference to a hardware interface specification document. This interface specification will often include timing requirements etc.

## Interface to external software actors

Describes the interface for actors representing another software subsystem, developed either as a separate project or it could be reusing a complete subsystem from earlier systems.

Describe the specific interface for each external software component or alternatively insert a reference to a software interface specification document for the actual software subsystem.

# Use Case View

**Mandatory view.**

This section lists selected Use cases and Use Case scenarios from the Use case model.

## Overview of architecture significant Use cases

**For large systems:** select the part of the Use cases, which have an impact on the architecture. Large systems can have several Use cases of the same type, with the same impact on the system architecture. Select one of these candidates to represent the group.

Significant Use case are selected and shown on the Use case diagram:

1. if they represent some significant, central functionality of the final system

2. if they have a large architectural coverage and exercise many architectural elements

3. if they stress or illustrate a specific, delicate point of the architecture.

**For smaller systems:** include all Use cases

**UML Use case diagram:** showing the selected Use cases and associated actors

## Use case 1. scenarios

### Use case goal

A short description of the goal for the actual Use case (as stated in the requirement specification document)

### Use case scenarios

This section describes one or more significant Use case scenarios for the selected Use case no. 1. It will always include the normal scenario (the sun shine scenario) and one or more scenarios with exceptions.

For Use cases with significant nonfunctional requirement: include these nonfunctional requirements (a copy from the requirement specification).

For Use cases with human actors: include significant Pictures of the User Interface.

For Use cases with include, extends or inherit relations: include a **UML Use-case diagram** for the Use case.

**References:** The realizations of the Use case scenarios should be found in the logical view, in subsection Use case realizations 5.3.

## Use case 2. scenarios

As previous section.

# Logical View

**Mandatory view.**

A description of the logical view of the architecture.

This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. and their organization in a layered structure. Describe the most important classes, their organization in packages and subsystems, and the organization of these packages and subsystems into layers.

## Overview

This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers and documented on UML class diagrams (showing packages).

If the system has several levels of packages, you should first describe those that are significant at the top level. Include any class diagrams showing significant top-level packages, as well as their interdependencies and layering. Next present any significant packages within these, and so on all the way down to the significant packages at the bottom of the hierarchy.

**UML class diagrams:** with packages, dependencies and interfaces.

## Architecturally significant design packages

A description of the design packages which have an impact on the architecture.

### Package 1

**Mandatory:** A brief description of the responsibility for package 1.

**Alternative 1:** A UML class diagram with all significant classes and packages contained within the package. The class diagram may include classes from other packages named as package\_name::class\_name – this often eases the understanding of the class diagram.

For each class in the package:  
 include its name, a brief description of its responsibility, and, optionally a description of some of its   
 important operations and attributes. Also describe important associations that are necessary for   
 understanding the class diagram.

Describe the package interfaces.

**UML Class diagram**   
 with its decomposition into classes and optionally packages contained within the package.  
**UML State Chart:**   
 for classes with a state machine include a UML state chart describing the state machine.

**Alternative 2:** describe the complete package in a separate design document and insert a reference here.

### Package 2

As previous section.

## Use case realizations

This section illustrates how the software actually works by describing the realization for the selected Use-case (and Use case scenarios), and explains how the various design model elements contribute to their functionality.

**Reference:** The corresponding Use cases and scenarios of these realizations should be found in the use-case view in section 4.

### Use case 1. realization

A brief description of how Use case 1. is realized in the design with references to the OO/UML model elements. The Use case logic is mapped onto packages and classes in the logical model.

**UML class diagram** with theclasses used in therealization of the Use-case 1 scenarios.

**UML interaction diagrams** (sequence- or collaboration diagrams) showing how the objects interacts to fulfilling one or more of the Use-case 1. scenarios.

### Use case 2. realization

As previous section.

# Process/task View

**Optional view.**

A description of the process view of the architecture. Use this view if the system has more than one thread of execution.

This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes) and the communication mechanism between these processes.

## Process/task overview

**For large systems:** organized this view by groups of communicating processes.

**UML Class diagram** as a process/task diagram with active classes and passive classes of type monitor.

## Process/task implementation

This section describes the operating system interface used in the implementation of processes/tasks and process communication. This section can specify the interface for a process/task framework (or wrapper façade classes).

An alternative is to insert a reference to the framework interface specification document.

If an operating system API is used, without wrapping, a reference to this API documentation is inserted here.

## Process/task communication and synchronization

Describe the main modes of communication between processes, such as message passing, interrupts, monitors and rendezvous.

## Process group 1.

This section gives a detailed description of the interactions in process group 1.

### Process communication in group 1

This section describes the processes in group 1. The interactions between the processes are shown on collaboration- or sequence diagrams, in which the objects are active objects with their own threads of control.   
**UML class diagram** as a process/task diagram with active objects (processes/tasks) and passive classes of type monitor.  
**UML collaboration diagram** with active objects and monitors (if used)   
or alternatively an   
**UML sequence diagram** with active objects and monitors (if used).

### Process 1. description

A brief description of behavior, lifetime and communication characteristics.

### Process 2. description

As previous section.

## Process group 2.

As previous section.

# Deployment View

**Optional view.**

A description of the deployment view of the architecture showing the physical structure of the system with one or more interconnected computers and connected hardware units. Use this view if the system is distributed across more than one node and in situations where the connected hardware is important to clarify.

This section describes one or more physical network (hardware) configurations on which the software is deployed and run.

Indicate as a minimum for each configuration the physical nodes (computers, CPUs) that execute the software, and their interconnections (bus, LAN, point-to-point, and so on.)

Also include a mapping of the processes of the Process View onto the physical nodes.

## System configurations overview

If there are many possible physical configurations, just describe a typical one and then explain the general mapping rules to follow in defining others.

## System configurations

Describe the different possible or typical system configurations. One of these could be a description of system configurations for performing software tests and simulations**.**

### Configuration 1.

Show the configuration with a UML deployment diagram illustrating the nodes in the configuration and their communication, followed by a mapping of processes to each processor node.

**UML deployment diagram** showing nodes and communication stereotypes in configuration 1.

**Optional: UML interaction diagram** showing the interactions between the nodes in the actual configuration.

### Configuration 2.

As previous section.

## Node descriptions

A short description of each node represented on the deployment diagrams for the different configurations.

### Node 1. description

A short description of node 1 – a processor or a hardware unit.

A processor is described by key data e.g. CPU type, clock frequency, memory (RAM, ROM) etc.

For HW nodes insert a reference to the documentation of the Hardware unit.

### Node 2. description

As previous section.

# Implementation View

**Optional.**

This section describes the decomposition of the software into layers and subsystems in the implementation model. Use this view if there is a different distribution of responsibilities between corresponding packages in the logical and implementation model.

## Overview

This subsection names and defines the various subsystems components and show their dependencies in a layered structure.

**A UML component:** can be a source file, a configuration file, an executable file (.exe) or a dynamic load library (DLL).

**UML component diagram** showing the systems components in a layered component structure with their dependencies and optionally interfaces.

## Component descriptions

This section describes each of the components shown on the overview diagram.

### Component 1

**Mandatory:** insert a short description of the responsibility for component 1.

**Alternative 1:** Describe the component and show its part components on a component diagram.

**UML component diagram** with part components and their dependencies (e.g. an executable components and its source files).

**Alternative 2:** Insert a reference to an external component documentation.

### Component 2

As previous section.

# Data View

**Optional view.**

A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.

## Data model

This section describes the architecturally significant parts of the data model:

* The mapping from key persistent design classes, especially where the mapping is non-trivial.
* The architecturally significant parts of the system which have been implemented in the database, in the form of stored procedures and triggers.
* Important decisions in other views which have data implications, such as choice of transaction strategy, distribution, concurrency, fault tolerance. For example, the choice to use database-based transaction management (relying on the database to commit or abort transactions) requires that the error handling mechanism used in the architecture include a strategy for recovering from a failed transaction by refreshing the state of persistence objects cached in memory in the application.

You should present architecturally significant data model elements, describe their responsibilities, as well as a few very important relationships and behaviors (triggers, stored procedures, etc.).

## Implementation of persistence

This section describes the interface to the persistency implementation. This could be the interfaces to a specific framework for handling this layer.   
**Alternatively:** insert a reference to existing persistency framework documentation.   
If a framework is developed as part of the current system – it will typically be documented in a stand alone document.

# Generel design decisions

This section describes the general design decision taken in the architecture design phase and mandatory design constraints given in the requirement specification or as a part of a company standard.

## Architectural goals and constraints

This section describes the non-functional software requirements and objectives that have some significant impact on the architecture e.g. use of an off-the-shelf product, portability, distribution, and reuse.

It also captures the special constraints that may apply e.g. design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.

## Architectural patterns

This section list the standard architectural patterns used in the project and a reference to the original source.

## General user interface design rules

This section describes the general rules and principles for designing the user interface part of the system. This could also be a reference to an external document.

## Exception and error handling

This section describes general rules for exception and error handling. Alternatively insert a reference to a stand alone document.

## Implementation languages and tools

This section list the chosen implementation language and tools with version numbers.

## Implementation libraries

This section list the chosen implementation libraries with references and version numbers.

# SIZE and performance

A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.

In this section, list the critical size and performance dimensions of the project that shape the architecture.

The information presented may include:

* The number of key elements the system will have to handle (such as the number of concurrent flights for an air traffic control system, the number of concurrent phone calls for a telecom switch, the number of concurrent online users for an airline reservation system, etc.).
* The key performance measures of the system, such as average response time for key events; average, maximum and minimum throughput rates, etc.
* The footprint (in terms of disk and memory) of the executables - essential if the system is an embedded system which must live within extremely confining constraints.

Most of these qualities are captured as requirements; they are presented here because they shape the architecture in significant ways and warrant special focus. For each requirement, discuss how the architecture supports this requirement.

# Quality

A description of how the software architecture contributes to other non-functional capabilities of the system: extensibility, reliability, portability, and so on.

In this section, list the key quality dimensions of the system that shape the architecture. The information presented may include:

* Operating performance requirements, such as mean-time between failure(MTBF).
* Quality targets, such as "no unscheduled down-time"
* Extensibility targets, such as "the software will be upgradeable while the system is running".
* Portability targets, such as hardware platforms, operating systems, languages.

For each dimension, discuss how the architecture supports this requirement.

You can organize the section by the different views (logical, implementation, and so on), or by quality.

# Compilation AND LINKING

This section describes the process of compiling and linking a program.

## Compilation-hardware

Describe the compilation machine.

## Compilation-software

Describe the software used to compile and assemble the program.   
For example: compiler, assembler, linker, include files, files with system parameters, make files etc.   
Include vendor and version number for standard products.

## Compilation and linking process

Describe the process to follow for compiling and linking a given program. Include a specification of special compiler and link switches.   
Describe the directory structure for the program.

# INSTALLATION AND Executing

Describe how to install and run the program.

## Installation

This section describes the process to bring an executable file to run in the right environment. It could be to download the program to another machine or to program it on EPROMs and insert these in the target machine.   
Describe the necessary configuration files and start up parameters.

## Executing-hardware

Describe the necessary hardware for running the program.

It can be on the development platform or on the final target platform.

## Executing-software

Describe the necessary SW requirements for running the program on a given machine. It could be operating systems or DLL files.

## Execution-control (start, stop and restart)

Describes how the system is started, stopped and restarted.

## Error messages

A complete description of the software error messages. For each error type give an explanation of the error and describe how to proceed to get the system working again.  
Alternative: a reference to a stand alone document.

# Appendices

Necessary supplementary documentation can be inserted here as appendices.