**Software Architecture Document**

Version X.X

for

<Project name>

Prepared by

|  |  |  |
| --- | --- | --- |
| <name> | <student-id> | <email> |
|  |  |  |

|  |  |
| --- | --- |
| Instructor: | <*name>* |
| Course: | < name> |
| Date: | <date of submission> |

**Document history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of contents

[1. Introduction 5](#_Toc437439092)

[Purpose 5](#_Toc437439093)

[Scope 5](#_Toc437439094)

[Definitions, acronyms, and abbreviations 5](#_Toc437439095)

[2. Architectural representation 6](#_Toc437439096)

[3. Architectural requirements: goals and constrains 8](#_Toc437439097)

[Functional requirements (Use case view) 8](#_Toc437439098)

[Non-functional requirements 8](#_Toc437439099)

[4. Use case view (Scenarios) 11](#_Toc437439100)

[5. Logical view 12](#_Toc437439101)

[Layers, tiers etc. 12](#_Toc437439102)

[Subsystems 12](#_Toc437439103)

[Use case realizations 12](#_Toc437439104)

[10. Development (Implementation) view 13](#_Toc437439105)

[Reuse of components and frameworks 13](#_Toc437439106)

[11. Process view 14](#_Toc437439107)

[12. Deployment (Physical) view 15](#_Toc437439108)

[13. Data view (optional) 16](#_Toc437439109)

[6. Size and performance 17](#_Toc437439110)

[7. Quality 17](#_Toc437439111)

# Introduction

The introduction of the Software Architecture Document provides an overview of the entire document.

## Purpose

This section defines the role or purpose of the Software Architecture Document and briefly describes the structure of the document. Identify the intended audience for the document is identified, with an indication of how they are expected to use the document.

## Scope

A brief description of what the Software Architecture Document applies to; what is affected or influenced by this document.

## Definitions, acronyms, and abbreviations

Provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the Software Architecture Document. This information may be provided by reference to the project’s Glossary. For example:

**RUP**: Rational Unified Process

**UML**: Unified Modeling Language

**SAD**: Software Architecture Document

# Architectural representation

1. **Logical view** : Audience: Designers. The logical view is concerned with the functionality that the system provides to end-users. UML Diagrams used to represent the logical view include **Class diagram**, and **interaction diagrams** (**communication diagrams**, or **sequence diagrams**).
2. **Use case view** (also known as Scenarios) : Audience: all the stakeholders of the system, including the end-users. The description of an architecture is illustrated using a small set of use cases, or scenarios which become a fifth view. The scenarios describe sequences of interactions between objects, and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype. Related Artifacts : **Use-Case Model**.
3. **Architectural requirements: goals and constrains**

Requirements are already described in SRS. In this section describe *key* requirements and constraints that have a significant impact on the architecture.

## Functional requirements (Use case view)

Refer to Use Cases or Use Case scenarios which are relevant with respect to the software architecture. The Use Cases referred to should contain central functionality, many architectural elements or specific delicate parts of the architecture.

The overview below refers to architecturally relevant Use Cases from the Use Case Model (see references).

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Name** | **Architectural relevance** | **Addressed in:** |
| Use case(s) or scenario(s). | Name of case(s) or scenario(s). | Description on why this use case or scenario is relevant to the architecture. | Section number where this use case or scenario is addressed in this document. |

## Non-functional requirements

Describe the architecturally relevant non-functional requirements, i.e. those which are important for developing the software architecture. Think of security, privacy, third-party products, system dependencies, distribution and reuse. Also environmental factors such as context, design, implementation strategy, team composition, development tools, time to market, use of legacy code may be addressed.

Usually, the non-functional requirements are already in place and can be referenced here. This document is not meant to be the source of non-functional requirements, but to address them. Provide a reference per requirement, and where the requirement is addressed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Source** | **Name** | **Architectural relevance** | **Addressed n:** |
| e.g. Vision, SRS. | Name of requirement. | Description on why this requirement is relevant to the software architecture. | Section number where this requirement is addressed in this document. |

1. **Use case view (Scenarios)**

The scenarios (or functional view) represent the behavior of the system as seen by its actors. Use case scenarios describe sequences of interactions between actorsd and the system (seen as a black box) as well as between the system and external systemsTthe *UML use case diagram* is used to capture this view.

1. **Logical view**

The logical view captures the functionality provided by the system; it illustrates the collaborations between system components in order to realize the system's use cases. Describe the architecturally significant logical structure of the system. Think of decomposition in tiers and subsystem. Also describe the way in which, in view of the decomposition, Use Cases are technically translated into Use Case Realizations.

## Layers, tiers etc.

Describe the top-level architecture style. Deploy a *UML class diagram*.

## Subsystems

Describe the decomposition of the system in subsystems and show their relation.

**Architecturally significant design packages**

Desribe packages of individual subsystems that are architecturally significant. For each package nclude a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

## Use case realizations

In this section you have to illustrate how use cases are translated into *UML interaction diagrams*. Give examples of the way in which the Use Case Specifications are technically translated into Use Case Realizations, for example, by providing a sequence-diagram. Explain how the tiers communicate and clarify how the components or objects used realize the functionality.