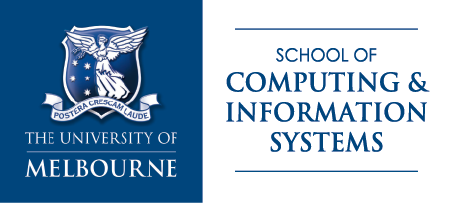
**Architecture Document**

**SWEN90007**

**<Project’s Name>**

Team: <Team’s Name>

In charge of: <Name of the person, or the department, in charge of the document >



*\*This page refers to the document template. It must be discarded and replaced by the real one (next page).*

Revision History (this template’s)

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 13/10/2017 | 01.00-D01 | Initial draft | Eduardo Araujo Oliveira |
| 2/2/2018 | 01.00-D02 | Added Section 3 details to the document | Eduardo Araujo Oliveira |
| 12/2/2018 | 01.00-D03 | Added Section 4 and 5 details to the document | Eduardo Araujo Oliveira |
| 1/3/2019 | 01.00 | First version of the document | Eduardo Araujo Oliveira |
| 28/7/2020 | 02.00-D01 | Adjust document to SWEN90007 subject | Eduardo Araujo Oliveira |
| 31/7/2020 | 02.00-D02 | Simplify the document to SWEN90007 project | Eduardo Araujo Oliveira |
| 2/8/2020 | 02.00 | Final review and improvements on the document | Eduardo Araujo Oliveira |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Revision History

*<students must use this table to track individuals’ contributions to this document.*

*Every time you change this document, add the date you changed it, a description of your performed task and your name. For the version, please adopt the following format:*

*01.00-D<number> for draft versions related to Part 1 (any version before the final submission is considered draft). When your document is reviewed and finally ready to be submitted, change it to 01.00. For Part2, start with 02.00-D<number> and so on. This document should always be kept on GitHub>*

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

**Contents**

[1.1 Proposal 5](#_Toc47094586)

[1.2 Target Users 5](#_Toc47094587)

[1.3 Conventions, terms and abbreviations 5](#_Toc47094588)

[3.1 Requirements of Architectural Relevance 7](#_Toc47094589)

[4.1 Modules 8](#_Toc47094590)

[4.1.1 <Module X> 8](#_Toc47094591)

[4.2 Components 9](#_Toc47094592)

[4.2.1 <Component X> 9](#_Toc47094593)

[5.1 Processes 11](#_Toc47094594)

[5.1.1 <Process XYZ> 11](#_Toc47094595)

[6.1 Architectural Patterns 12](#_Toc47094596)

[6.1.1 <Pattern XYZ> 12](#_Toc47094597)

[6.2 Source Code Directories Structure 13](#_Toc47094598)

[6.3 Libraries and *Frameworks* 13](#_Toc47094599)

[6.4 Development Environment 14](#_Toc47094600)

[7.1 Production Environment 14](#_Toc47094601)

[7.1.1 Hardware 14](#_Toc47094602)

[7.1.2 Software 15](#_Toc47094603)

[7.2 Development Environment 15](#_Toc47094604)

# Introduction

<This space must be used to describe the objectives of this document and for whom the document is intended. Complete and/or adapt the following text to give this information.>

This document specifies the system’s architecture <System’s name>, describing its main standards, module, components, *frameworks* and integrations.

## Proposal

The purpose of this document is to give, in high level overview, a technical solution to be followed, emphasizing the components and *frameworks* that will be reused and researched, as well as the interfaces and integration of them.

## Target Users

This document is aimed at the project team, with a consolidated reference to the research and evolution of the system with the main focus on technical solutions to be followed.

## Conventions, terms and abbreviations

<This sub-section should describe the conventions, terms and abbreviations necessary to appropriately interpret this document. The necessary explanations can be given directly in this section or by references to other documents or appendices of this document. Complete and/or adapt the text below to show this information.>

This section explains the concept of some important terms that will be used throughout this document. These terms are detailed alphabetically in the following table.

|  |  |
| --- | --- |
| Term | Description |
| Component | Reusable and independent software element with well defined public interface, which encapsulates numerous functionalities and which can be easily integrated with other components. |
| Module | Logical grouping of functionalities to facilitate the division and understanding of software. |
| *<students to add more>* |  |

# Architectural representation

*<Describe, in this section, the adopted model for architectural representation of the system. The framework “4+1” is used as a base, however, some of the views can be omitted in accordance with the project’s content.*

*The specific section of each vision will indicate which ones can be omitted and in which situation this can happen. The following text must be adjusted to reflect the followed options of each project, indication which visions are being represented which are being omitted. >*

The specification of the system’s architecture <System’s name> follows the *framework* “4+1” [1], which defines a set of views, as shown in Figure 1. Each of these views approaches aspects of architectural relevance under different perspectives:

* The **logical view** shows the significant elements of the project for the adopted architecture and the relationship between them. Between the main elements are modules, components, packages and the application main classes;
* The **process view** shows the concurrency and synchronization aspects of the system, mapping the elements of the logical view to processes, *threads* and execution tasks;
* The **development view** focuses on aspects relating to the organization of the system’s source code, architectural patterns used and orientations and the norms for the system’s development;
* The **physical view** shows the hardware involved and the mapping of the software elements to the hardware elements in the system’s environment.
* The **scenarios** show a subset of the architecturally significant use cases of the system.

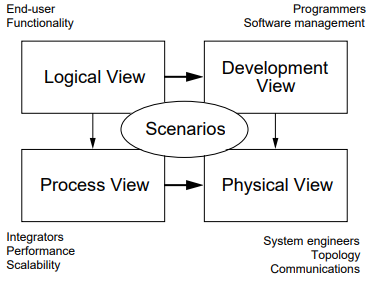


Figure 1. Views of *framework* “4+1”

source: Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *IEEE software*, *12*(6), 42-50.

# Architectural Objectives and Restrictions

*<In this section, the objectives that are intended to be attained with the defined architecture and the environmental restrictions that have impact on the definition of this architecture must be described.*

*Common objectives: portability, performance, scalability, tolerance to imperfections etc. Normally such objectives derive from the non-functional requirements of the system.*

*Common restrictions: memory limitation, band limitation, profile of the development team, production environment, integration with systems legacies etc. Normally such restrictions derive from restrictions and permission described in the Client Requirements Document and in the context in which the system is to be developed.*

*Start describing the objectives that will be attained and the main restrictions of textual form. After this, list the requirements (most commonly the non-functional) that have impact on the definition of the architecture.*

*Alter the following text.>*

The defined architecture’s main objective is to make the system...

*<students to edit this>*

## Requirements of Architectural Relevance

*<The requirements that have impact on the architecture and the impact of each one must be listed here. Additionally, the treatment given to each of them in the architecture must be indicated*

*Usually, the non-functional requirements, the restrictions and permissions described in the Client Requirement Document directly impact on the architecture of the system.*

*Alter the following text, if necessary.>*

This section lists the requirements that have impact on the system’s architecture and the treatment given to each of them.

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Impact** | **Treatment** |
| <requirement ID> | <Impact on the architecture > | <Treatment given to the architecture> |
| <RNF 005> | <number of accesses to service XYZ must be minimized> | <A *caching* mechanism will be implemented based on the parameters given to the service XYZ> |
| <REST 001> | <Memory consumption must be minimized> | <The number of system layers will be reduced to diminish the need of intermediary objects > |
| *<students to edit this>* |  |  |

# Logical View

*<Describe, here, the significant project elements for the adopted architecture, including modules, components, packages and the* application main classes *(in increasing level of granularity). The relationships between them must also be described.*

*Start with a high-level diagram of the system’s organization and afterwards supply more details about the principal elements and the interfaces between them.*

*Aspects which will be remembered in the definition of the architecture, in accordance with the context of the project are: security, presentation, business logic, persistence, integration, distribution etc.*

*Alter the following text, if necessary.>*

This section shows the system’s organization from a functional point of view. The main elements, like modules and main components are specified. The interface between these elements is also specified. Figure 2 illustrates the logical architecture of the system.

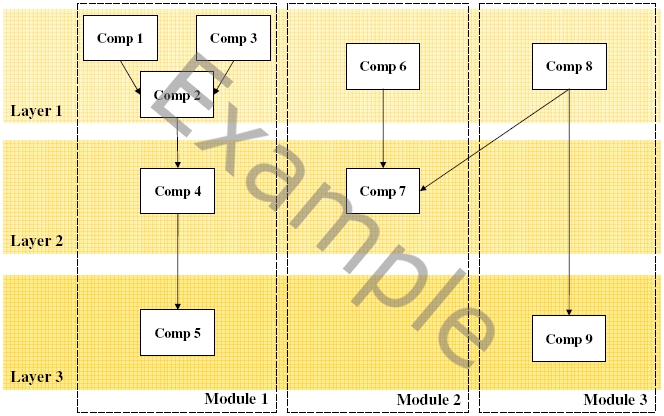


Figure 2. General view of the architecture

## Modules

*<Optional, if the system is divided in modules, list the modules and the requirements they handle (in general, functional requirements described in the Client Requirement Document) for each one.*

*This level of division is of smallest granularity and at the same time the biggest abstraction. In this case, it is more difficult for reuse to be obtained owing to the level of the modules specialization.*

*Alter the following text, if necessary.>*

This section describes the main application modules, emphasizing their responsibilities and functionalities, as well as the requirements they handle.

### <Module X>

*<Briefly describe the module, indicating the requirements handled.*

*Repeat this sub-section for each of the system modules.*

*Alter the following text.>*

The module <Module X> will put the functionalities into the relative groups …

|  |  |
| --- | --- |
| **Responsibilities** | <Module responsibilities>. |
| **Handled requirements** | <Handled requirements by the module >. |

## Components

*<The main components of the system must be specified in this section It is a high-level specification and its objective is to act as a base for the specification of the components in the phase of the project.*

*For each component, there must be a justification if the component will be reused, if it will be retrieved from third part products (COTS), or if it will be developed by the project. If an acquisition is required, the project manager shall be notified.*

*If the decision is to use COTS it is necessary to perform a feasibility study considering the following requirements:*

1. *Functionality, performance, quality, and reliability*
2. *Terms and conditions of warranties for the products*
3. *Risk*
4. *Suppliers' responsibilities for ongoing maintenance and support of the products*

*The decision criteria must be based on the architectural purposes and constraints, described in section 3. If the components are from third party products (COTS), the following aspects shall be analyzed: license, cost, support and maturity level.*

*It is still necessary to identify if the component will be developed with the object of it being reusable – that which have a higher cost and greater concern in relation to its s design. If reuse is not the objective, a justification for this is required.*

*If you wish, one can opt to list the components grouping them in the modules described in the previous sub-section (Modules). In this case, this section must be removed and copied for each application module.*

*Alter the following text.>*

This section describes the main components of the system <System’s name>.

### <Component X>

*< Briefly describe the component, indicating its source and version, if it is being reused.*

*Repeat this sub-section for each component of the system.*

*Alter the following text.>*

The <X Component> will be responsible for…

|  |  |
| --- | --- |
| **Responsibilities** | <Component Responsibilities>. |
| **Handled Requirements** | <Handled requirements by the component>. |
| **Justification** | *<It is obligatory to justify the choice of this component above others. This field must be completed with one of the following options:*   * ***The only option****:**The component will be used as it has been identified as the only option able to handle the requirement.;* * ***CIS Standard*** *: The component will be used because it is already part of the CIS standard;* * ***Decision Making****: The choice was made using the Decision Making Sub-Process. In this case a link to the decision table must be added.>* |
| **Will it be reused?** | <Yes/No>.  <Justification for reusing the component or not >. |
| **Source** | < Source and component version to be reused. If the component is not to be reused, insert NA.>. |
| **Will it be reusable?** | <Yes/No>.  <Justification for developing a component which can be reused by other projects >. |

# Process View

*< Optionally, if the system has a complex execution, involving different processes and threads, and concurrency and synchronization questions are relevant, specify the mapping of the logical elements to the execution elements. In case of the system doesn’t have these characteristics, replace the content with NA.*

*Start with a diagram that shows the main elements of the logical architecture being mapped to the processes and threads of the system. If it is necessary, detail the processes and threads with text. The creation of sequential or additional state diagrams are suggested to better clarify the interaction between the processes and/or threads.>*

The section shows the mapping of the logical architecture elements to the processes and threads of the system execution. Figure 3 illustrates this mapping.

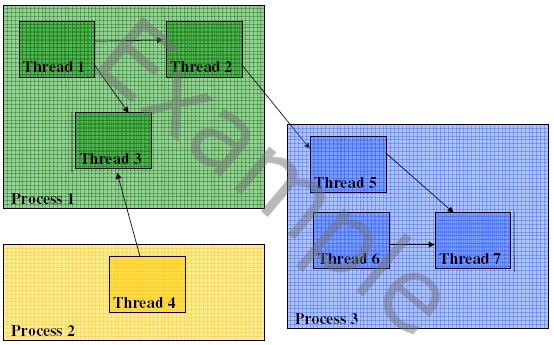


Figure 2. System execution diagram

## Processes

*< Optionally, list the system processes, noticing redundancy and synchronization aspects.>*

This section lists the system’s processes, together with their redundancy and synchronization requirements with other processes.

|  |  |  |
| --- | --- | --- |
| **Process** | **Instances** | **Synchronization** |
| <Process’s Name> | <Number of processes (1, if there is no redundancy) > | <List of processes that must be synchronized with this process > |

### <Process XYZ>

*< Optionally, list the system processes, noticing redundancy and synchronization aspects. Repeat this sub-section for other processes, if necessary.>*

|  |  |  |
| --- | --- | --- |
| ***Thread*** | **Instances** | **Synchronization** |
| <*Thread’s Name*> | <Number of *threads* (1, if it is not pooled)> | <List of *threads* that must be synchronized with this *thread* > |

# Development View

*< In this section the orientations to the developers must be given for the project and system implementation. Code patterns and norms, architectural patterns, best practice, developing environment, libraries and frameworks used must be described.*

*Optionally, it can be defined in the Development Guide including some of these orientations, leaving only a reference to them here.*

*It is highly recommended to give reference to existing guides in the Analysis and Project and Implementation sub-processes of you development process.*

*Optionally, insert a diagram of the implementation view, adding details about the technologies and patterns to handle non-functional aspects to the logical view of the architecture*

*Alter the following text.>*

This section provides orientations to the project and system implementation in accordance with the established architecture. Figure 3 illustrates the implementation view of system architecture.

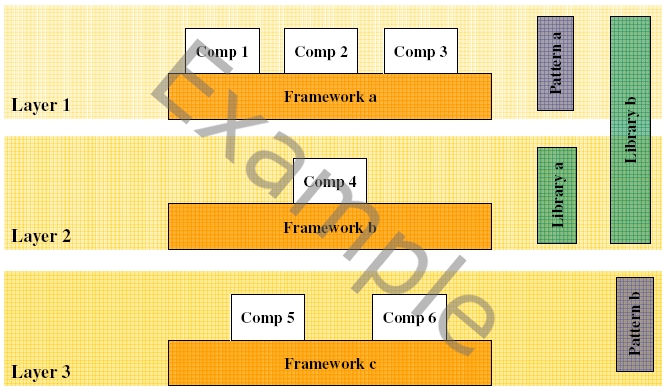


Figure 3. Development view of system architecture

## Architectural Patterns

*< In this section describe all architectural patterns which will be used in the system, justifying their use. Optionally, list the pattern adaptations and examples of use in the system.*

*If there are neither adaptations nor examples of use, the following table can be used. Otherwise, create a sub-section for each pattern.*

*All used patterns must be catalogued in the CIS’s Pattern Catalog. If some new pattern needs to be catalogued, it must be submitted to SEPG.*

*The following aspects must be considered when defining architectural patterns, according to the project context: security, presentation, business rules, persistence, error handling, logging, session management, transaction management, concurrency, integration, distribution, etc.>*

|  |  |
| --- | --- |
| **Pattern** | **Reason** |
| <Pattern Name> | < Justification for its use. If applicable, also justify the reason why another pattern that deals with the same problem has not been used*.*> |

### <Pattern XYZ>

*<Briefly describe how and where the pattern will be used in the project’s architecture.*

*Repeat this section for each architectural pattern of the system.>*

#### Examples of Use

*< Optionally, this section may provide some scenarios of use for this pattern in the system.>*

## Source Code Directories Structure

*< Describe the system directory structure, mapping the directories of the main elements of the architecture. This directory system relates only to source code structure and typically starts from the directory “src” Descriptions of the other structures of the directories can be found in the Management Configuration Plan.*

*For Java applications, how the packaging of components will be undertaken must also be described. Typically the packaging of the components at CIS have the prefix* ***org.unimelb.cis.<applicationname>.<componentname>****. It is further suggested that two more packages are created after the previous structure, using the names* ***api*** *and* ***impl,*** *to respectively group the classes “public” and the classes “private” of the component.>*

## Libraries and *Frameworks*

*<Libraries and frameworks form the infrastructure, providing basic functionalities, on which the system will be based. The main system frameworks and libraries must be specified in this section. It is a high-level specification and has the purpose to support the design phase.>*

This section describes libraries and *frameworks* used by the system <System’s Name>.

|  |  |  |  |
| --- | --- | --- | --- |
| **Library / *Framework*** | **Reason** | **Version** | **Environment** |
| *< Library/framework name>* | *< This field must be completed with one of these options:*   * ***The only option****:**The library will be used as it is the only option identified to deal with the requirement;* * ***CIS Standard****: The library will be used because it is already part of the CIS standard;* * ***Decision Making****: The choice was made using the Decision Making Sub-Process. In this case a link to the table with the decision must be added.>* | <Version to be used in the application> | <*Used Library/framework environment. E.g.: All, Production, or Development.>* |

## Development Environment

*< Describe the tools required to the design and implementation phases. Each tool must be listed in the Project Plan, detailing here its characteristics and the way in which they the project will use them.*

*Alter the following text.>*

The development environment will be formed by...

# Physical View

*< Optionally, if there is physical distribution in system, the mapping between the software elements and the hardware of the system is described in this section. At least, the production environment must be handled. Optionally describe the homologation and development environments. If the system doesn’t fit into these characteristics, replace the context of this section with NA>*

This section describes the hardware elements of the system and the mapping between them and the software elements.

## Production Environment

*<Describe the production environment and the mapping between the software elements and the hardware of the system, focusing on the processes that will be executed in each system hardware.>*

This section describes the production environment of the system and the mapping between the software elements and the available hardware. Figure 4 illustrates the physical view of the production environment.

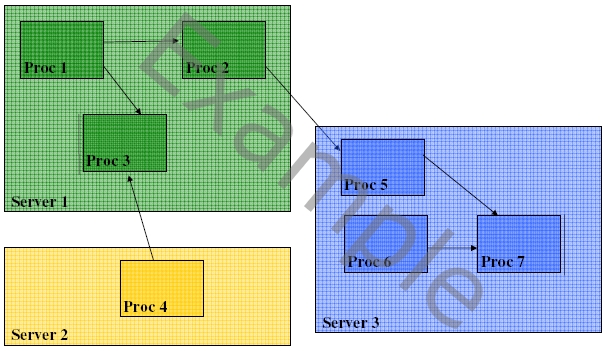


Figure 4. Production environment of the system

### Hardware

*< Describe the hardware of the production environment.>*

### Software

*< Describe the software of the production environment.>*

## Development Environment

*< Describe the development environment and the mapping between software and hardware elements of the system.>*

# Scenarios

*<Optionally, if it is necessary to describe the use cases set with most architectural relevancy, it could be done in this section.*

*The use cases with architectural relevancy are the ones that represent a core functionality of the application, or have a great architectural coverage, or illustrate an specific complex point of the architecture.>*

# References

< This section must provide a list of all related documents.>