C++

T-CPP-710

AbstractVM

Request for proposal

Damien Tohin Doumer Kake

Linda Binessi

Table of contents

[Introduction 3](file:///C:\Users\dtohi\Downloads\Template-Software-Architecture-Specifications-V2.docx#_Toc500948735)

[1. Project context 3](file:///C:\Users\dtohi\Downloads\Template-Software-Architecture-Specifications-V2.docx#_Toc500948736)

[2. Global architecture 3](file:///C:\Users\dtohi\Downloads\Template-Software-Architecture-Specifications-V2.docx#_Toc500948737)

[3. Component description 3](file:///C:\Users\dtohi\Downloads\Template-Software-Architecture-Specifications-V2.docx#_Toc500948738)

[a. Component 1 3](file:///C:\Users\dtohi\Downloads\Template-Software-Architecture-Specifications-V2.docx#_Toc500948739)

[b. Component 2 3](file:///C:\Users\dtohi\Downloads\Template-Software-Architecture-Specifications-V2.docx#_Toc500948740)

[c. Component 3 3](file:///C:\Users\dtohi\Downloads\Template-Software-Architecture-Specifications-V2.docx#_Toc500948741)

[4. Traceability matrix 3](file:///C:\Users\dtohi\Downloads\Template-Software-Architecture-Specifications-V2.docx#_Toc500948742)

# Introduction

The aim of this software architecture specification (SAS) is to present the technical elements necessary for the realisation of the AbstractVM project. This document outlines a very clear and concise overview and description of functionalities of each component which the AbstractVM project will be made of once completed.

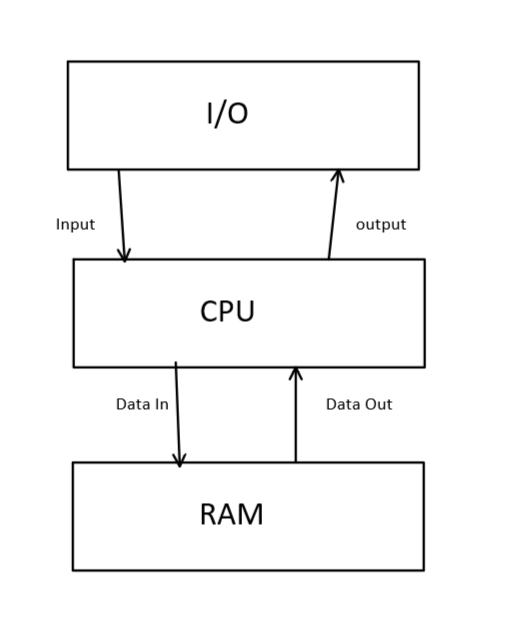
# Project context

The project consists of implementing a generic CLI calculator, which is able to interpret specific commands input by the user via its CLI interface, accept numeric data from the user, perform specified operations on the data input, and handle gracefully exceptional events which result from user’s interaction with the system.

# Global architecture

Taking into consideration the functionalities of the system, implementing this system, several components will be necessary to implement this application. Each component will have a clear and concise relationship with another for the system to function properly.

The Global architectural design pattern used for the implementation of this project will be the **Layered Pattern.** This pattern will consist of separating components of the application into sublayers, each having a distinct functionality. And communication between these layers will be made by using simple function calls. Nevertheless, each layer its self will be made up of its own architecture and design pattern implementation. As required by the design requirement specification, 3 distinct components are identified, which are named as follows.



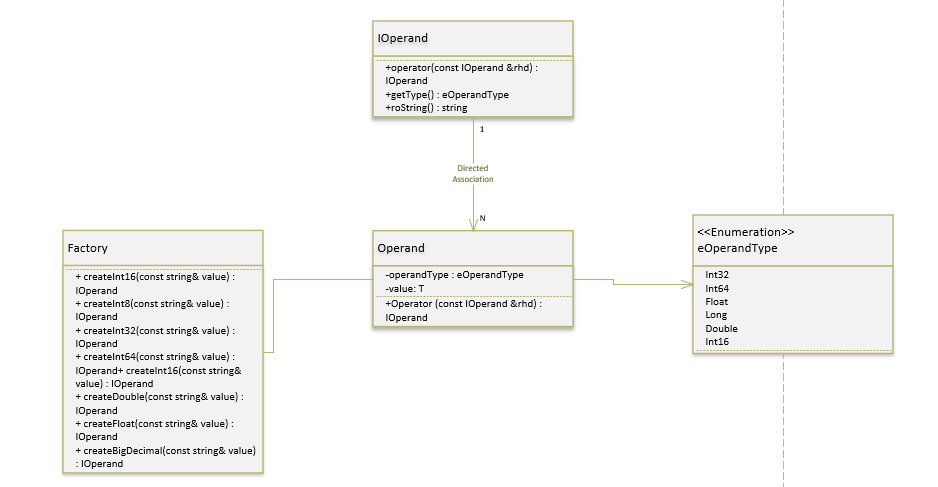
# Component description

## CPU

This component is responsible for:

* Performing the mathematical operations required by the system. Operations required by the user on the data input are handled by this layer.
* Handling any form of numeric data input by the user gracefully and perform arithmetic operations requested on this data seamlessly.
* Coordinating the interactions with other layers of the system.

It uses the **Factory** pattern to create new operands based on the data and operations given to it by the user.



It leverages the power of Templates, Overloading and Inheritance to have a **Polymorphic** behaviour, permitting it to handle various kind of numeric data input and multiple operations seamlessly.

This layer will be in direct communication with the **RAM** and the **I/O** component.

* It will access the stack to get operands and perform arithmetic operations needed.
* It will communicate with the I/O component to display results of operations to the user.

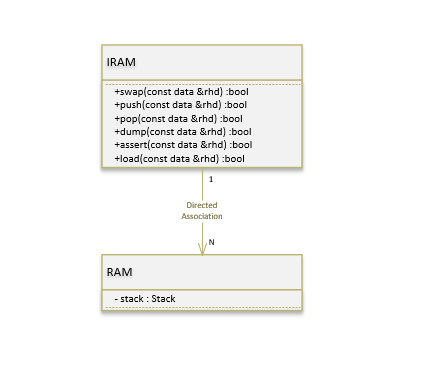
## RAM

The RAM will handle data storage. This will be done in a way. Data will be stacked and made available to the user only through stack operations such as push, pop… These are its functionalities.

* Store data on demand.
* Perform basic operations on the data such as push, pop…
* Handling exceptional situations which may occur during data manipulation.

This component will be available as a unique entity for all other components and to achieve this, it will be built with the **Singleton** Design Pattern.

Its main relationship with other components will be to provide data access for each operation required by these components.



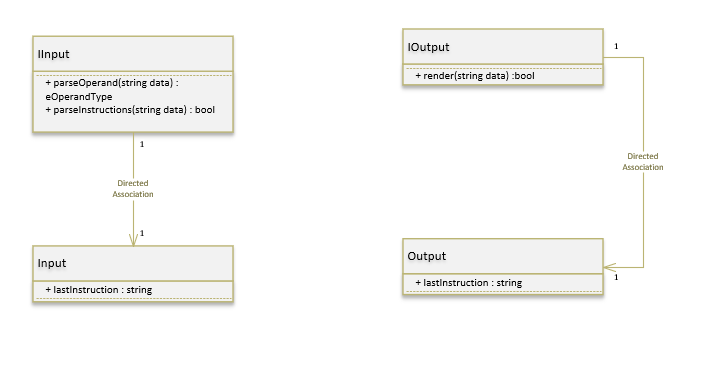
## I/O Input Output

This layer will handle input output operations and communication with the user. Here are its functionalities.

* Parsing user input
* Rendering system messages to the user
* Gracefully handling exceptional cases due to I/O operations such as parsing input…

This layer is essential since it is how the user will interact with the system. It will communicate with the other layers of the system as follows.

* Parse and interpret instructions sent to it by the user and communicates it to the appropriate layer.
* Handle messages from other layers and renders it to the user.



# Traceability matrix

This matrix makes the correspondence between components, classes, functions and requirements developed in the request for proposal.

|  |  |  |  |
| --- | --- | --- | --- |
| Id requirement | ***Requirement description*** | ***Component*** | ***Function / action*** |
| REQ\_XXX\_XXX | Description | Component X | Prototype function and/or action |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |