****

**HUSACCT**

**Architecture Notebook**

***Eclipse Plugin***

Team 1:

Erik Verhoef 1572690  
René van Aerle 1556647  
Tim Linschoten 1548807  
Niels Nijveldt 1562854  
Stefan Collette 1575795

Teachers: Leo Pruijt, Christian Köppe, Michiel Borkent.

Date: 25-05-2012

Inhoudsopgave

[Architectural significant requirements 2](#_Toc325740916)

[Use cases 2](#_Toc325740917)

[Use case model 2](#_Toc325740918)

[Use case Create Workspace 3](#_Toc325740919)

[Use case Save Workspace 4](#_Toc325740920)

[Use case Open Workspace 5](#_Toc325740921)

[Use case Close Workspace 6](#_Toc325740922)

[Use case Define Architecture 7](#_Toc325740923)

[Use case Show Logical Architecture 8](#_Toc325740924)

[Use case Import Architecture 9](#_Toc325740925)

[Use case Export Architecture 10](#_Toc325740926)

[Use case Analyse Project 11](#_Toc325740927)

[Use case Show Analyse frame 12](#_Toc325740928)

[Use case Show Analysed Architecture Graphics 13](#_Toc325740929)

[Use case Validate architecture 14](#_Toc325740930)

[Use case View violation 15](#_Toc325740931)

[Functional Requirements 16](#_Toc325740932)

[Non-functional requirements 16](#_Toc325740933)

[Decisions and justification 17](#_Toc325740934)

[Physical models 18](#_Toc325740935)

[Physical class diagram 18](#_Toc325740936)

[Physical software partitioning model 19](#_Toc325740937)

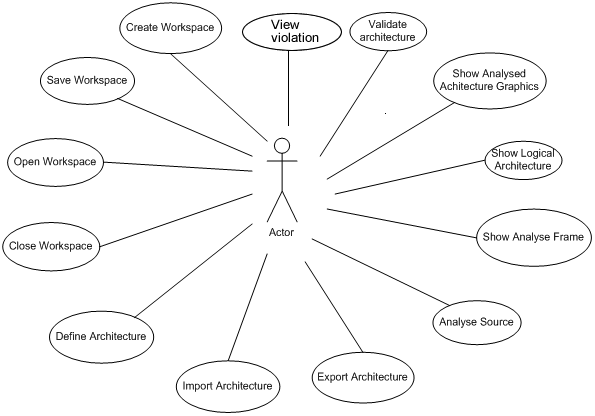
[Classmodel in TopCased 20](#_Toc325740938)

# Architectural significant requirements

## Use cases

* Create Workspace
* Save Workspace
* Open Workspace
* Close Workspace
* Define Architecture
* Import Architecture
* Export Architecture
* Analyse Source
* Show Analyse Frame
* Show Logical Architecture
* Show Analysed Architecture Graphics
* Validate architecture
* View violation

### Use case model



### Use case Create Workspace

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version: | 1.0 |
| Priority | Must have |
| Use Case | Create Workspace |
| Actors | User |
| Summary | Eclipse is opened with a java project with classes |
| Precondition | The HUSSAC-tool is opened |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor analyses a project within eclipse |  | |  |  | |  | 1. System creates new workspace with the eclipse project information. | |
| Postcondition | A new workspace is created within the memory. |

### Use case Save Workspace

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version: | 1.0 |
| Priority | Must have |
| Use Case | Save Workspace |
| Actors | User |
| Summary | The actor wants to save the workspace which is opened. |
| Precondition | There is a analysed project in eclipse. |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor pushes the button “Save project” when right-clicking on a project |  | |  | 1. System saves the workspace. | |  | 1. A hussact.hu file is created | |
| Postcondition | The workspace is saved and a hussact.hu is added to the root of the project. |

### Use case Open Workspace

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version: | 1.0 |
| Priority | Must have |
| Use Case | Open Workspace |
| Actors | User |
| Summary | The actor wants to open an earlier saved workspace. |
| Precondition | Eclipse is open with an earlier analysed project in it. |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor pushes the button “analyse project” |  | |  | 1. System checks if the project has a hussact.hu file in the root | |  | 1. The system loads the workspace with use of the hussact.hu from the root of the project | |  |  | |
| Postcondition | The workspace is opened with the available information loaded. |

### Use case Close Workspace

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version: | 1.0 |
| Priority | Could have |
| Use Case | Close Workspace |
| Actors | User |
| Summary | This use case is only triggered when the actor analyses a different project. |
| Precondition | There is a project opened. |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor pushes the button “analyse project” on a different project |  | |  | 1. System checks if a different project is selected 2. System returns a massage with a warning |  1. Actor chooses to continue  |  |  | | --- | --- | |  | 1. System saves workspace 2. System closes current workspace 3. System opens selected workspace. | |
| Postcondition | The current workspace is closed and the new selected workspace is open. |

### Use case Define Architecture

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version | 1.0 |
| Priority | Must have |
| Use Case | Define Architecture |
| Actors | User |
| Summary |  |
| Precondition | A workspace is opened. |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor opens the define-view |  | |  | 1. System shows define service gui. | |
| Postcondition | Define architecture GUI is shown. |

### Use case Show Logical Architecture

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version | 1.0 |
| Priority | Must have |
| Use Case | Show Logical Architecture |
| Actors | User |
| Summary |  |
| Precondition | A logical architecture is defined |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor opens graphics – defined architecture view |  | |  | 1. System shows logical architecture gui. | |
| Postcondition | Logical architecture GUI is shown. |

### Use case Import Architecture

**Use case description**

|  |  |
| --- | --- |
| Use case |  |
| Version | 1.0 |
| Priority | Must have |
| Use Case | Import architecture |
| Actors | User |
| Summary | If the user has made an architecture in the tool before and exported this with the export architecture use case then the user could import that architecture with this use case |
| Precondition | There is an architecture export file of an existing architecture |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor pushes ‘Import Architecture’ button in the menu or define view. | 1. System shows pop-up-screen for selecting a file. | | 1. Actor selects the architecture file. 2. Actor pushes “Import” button. | 1. The system triggers the import function of the Hussact | |  |  | |  |  | |
| Postcondition | An existing architecture is loaded. |

### Use case Export Architecture

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version | 1.0 |
| Priority | Must have |
| Use Case | Export architecture |
| Actors | User |
| Summary | After the architecture is made, the architecture should be able to be exported. |
| Precondition | Architecture rules need to be defined. |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor pushes ‘export Architecture’ button in the menu after right-clicking on the project or in the define gui. | 1. System shows pop-up-screen for selecting the export path. | | 1. Actor selects the architecture file location. 2. Actor pushes “Export” button. | 1. The system triggers the export function of Husacct | |  |  | |
| Postcondition | The architecture is saved to a .hu file. |

### Use case Analyse Project

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version: | 1.0 |
| Priority | Must have |
| Use Case | Analyse project |
| Actors | User |
| Summary | The user is able analyse a project, resulting in a graphical representation of the analysis. |
| Precondition | Eclipse is opened with a java project in it. |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor pushes the “analyse project”- button after right-clicking the project. |  | |  | 1. System analyses the project and triggers open or create workspace. | |
| Postcondition | The project is analysed and can be mapped to the logical architecture. |

### Use case Show Analyse frame

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version | 1.0 |
| Priority | Must have |
| Use Case | Show analyse |
| Actors | User |
| Summary | Show the analysed application in a tree view and dependency’s. |
| Precondition | An application is analysed. |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor opens the analyse view. | 1. System shows analyse frame. | |
| Postcondition | Application tree GUI is shown |

### Use case Show Analysed Architecture Graphics

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version | 1.0 |
| Priority | Must have |
| Use Case | Show Analysed Architecture Graphics |
| Actors | User |
| Summary |  |
| Precondition | A logical architecture is defined |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor opens the graphics – analysed architecture view. | 1. System shows the analyzed architecture | |
| Postcondition | The analysed architecture is shown on the screen |

### Use case Validate architecture

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version: | 1.0 |
| Priority | Must have |
| Use Case | Validate architecture |
| Actors | User |
| Summary | Checks if dependencies violate the rules |
| Precondition | A logical architecture is made and the source is mapped to it. |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor pushes ‘validate project’ when right-clicking the project or pushes validate in the validate view. |  | |  |  | |  | 1. System triggers the validate function of Hussact 2. A table will be filled with violationinformation. | |
| Postcondition | In the validate view a table is shown with all the violations within the project. |

### Use case View violation

**Use case description**

|  |  |
| --- | --- |
| Use Case |  |
| Version: | 1.0 |
| Priority | Must have |
| Use Case | View violation |
| Actors | User |
| Summary | Jump in to the source based on a violation. |
| Precondition | A table is created and has at least 1 violation. |
| Main scenario | |  |  | | --- | --- | | **Actor actions** | **System actions** | | 1. Actor selects violation in the violation table. |  | |  |  | |  | 1. System reads the file and linenumber of the violation. 2. Eclipse opens an editor with the right file and on the right linenumber. | |
| Postcondition | An editor is opened with the right file and is selected on the linenumber where the violation is occurred. |

## Functional Requirements

|  |  |  |
| --- | --- | --- |
| **CFR** | **Description** | **Priority** |
| 1 | Use Husacct functionality in eclipse with the use of a plugin. | Must have |
| 2 | Ability to call the define service. This service is responsible to handle several operations which has to do with defining architecture rules. | Must have |
| 3 | Ability to call the analyse service. This service is responsible to handle several operations which has to do with analyzing. | Must have |
| 4 | Ability to call the validate service. This service is responsible to handle several operations which has to do with validating. | Must have |
| 5 | Ability to save workspace. | Must have |
| 6 | Ability to open workspace. | Must have |
| 7 | Ability to create a new workspace. | Must have |
| 8 | Ability to show the architectural graphics. | Must have |

## Non-functional requirements

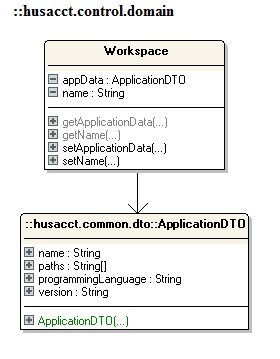
|  |  |  |
| --- | --- | --- |
| **NFR** | **ISO 9123 attr.** | **Requirement** |
| **1.** | **Functionality** |  |
| 1.1 | Suitability | The functionality’s of the original Husacct has to be re-used. |
| 1.2 | Suitability | The plugin has to add useful functionality’s for the user within eclipse. |
|  | **Reliability** |  |
|  | Maturity | When the original Husacct goes down, eclipse must still be able to be used. |
|  | **Usability** |  |
|  | Understandability | The division in required steps (manual and automated) must be clear to the user of the tool. |
|  | Operability | The plugin must be easy to operate and is a good addition for the user in eclipse. |
|  | **Maintainability** |  |
|  | Analyseability  Testability | Taking over the development of the tool by other development teams must be unproblematic. |
|  | Testability | Tests must test curtain packages to make sure if something changes it works. |
|  | **Portability** |  |
|  | Installability | The plugin must be easy to install. |
|  | Installability | The plugin must provide a complete installation package (no extra downloads needed). |

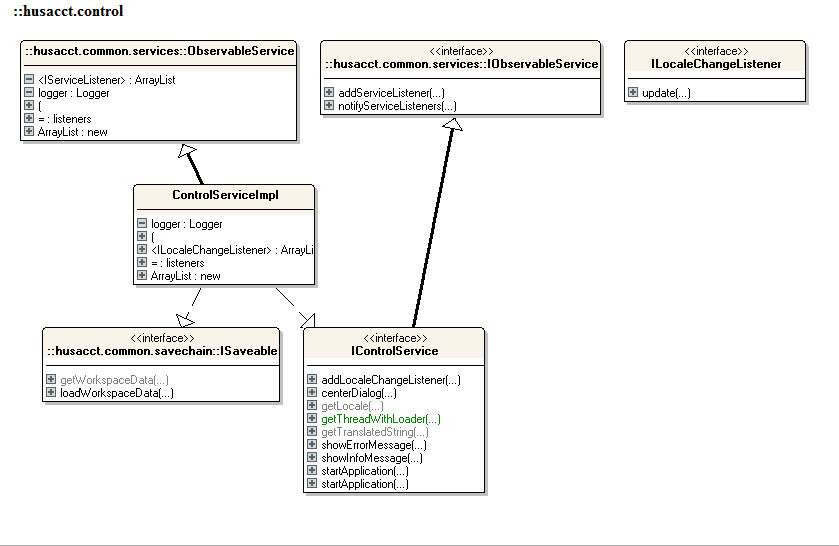
## Decisions and justification

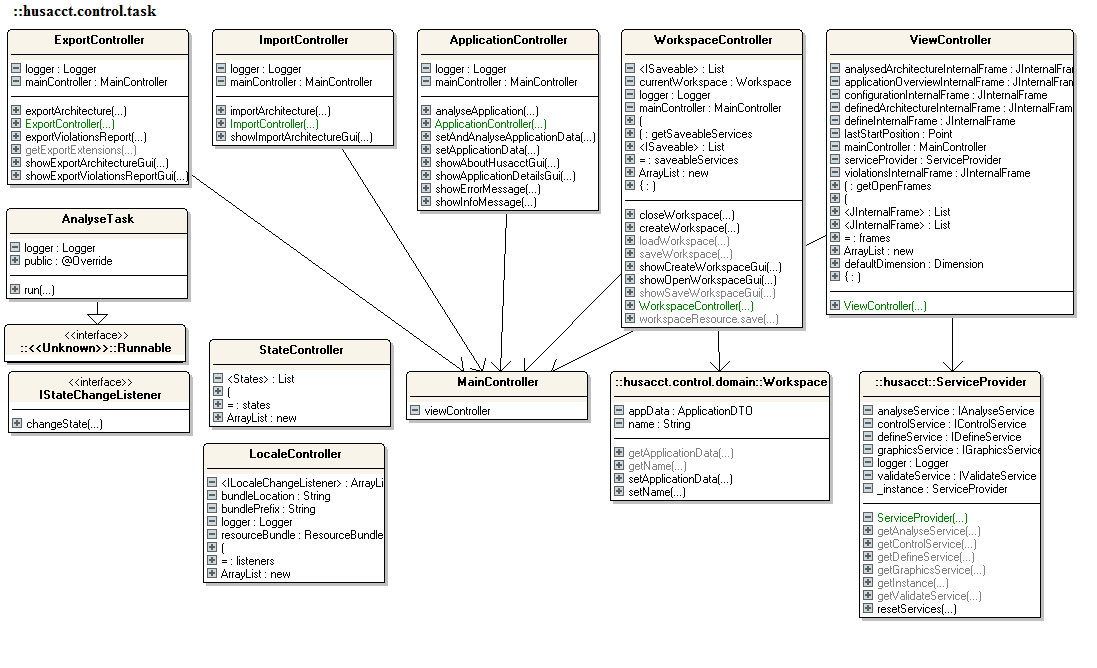
|  |  |
| --- | --- |
| **Decision** | **Justification** |
| In order to ensure other development teams can easily maintain the source code of the SACC-tool, we make use of GIT combined with GITHUB. | NF4.1 |
| A library which is an export of the original Husacct is used in the plugin | NF 1.1 |
| Eclipse specific functionality’s are used in the plugin. | NF 1.2 |
| A startview is added to help the user with using the plugin in eclipse. | NF 3.1  NF 3.2 |
| A statevew is added to show which steps are made or have to be made. | NF 3.1 |
| A set of unittest are set up to test the control part of the plugin. | NF 4.2 |
| Export can be made of the plugin and there is an installation guide available. | NF 5.1  NF 5.2 |

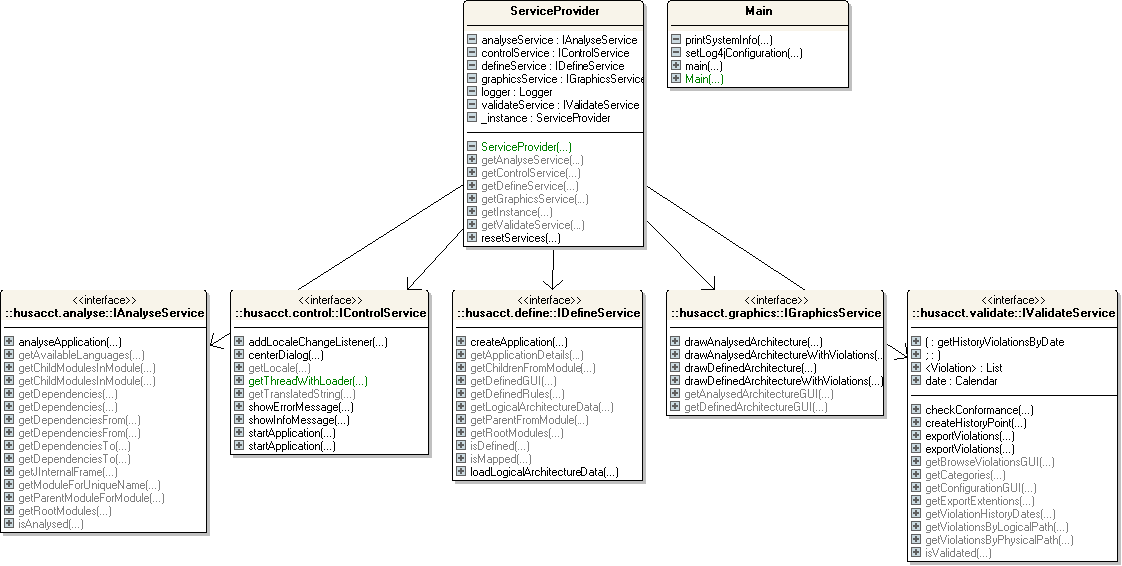
# Physical models

## Physical class diagram









## Physical software partitioning model

