****

**HUSACCT**

**Architecture Notebook**

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: 29-05-2012

Inhoudsopgave

[Architectural significant requirements 2](#_Toc326010062)

[Use cases 2](#_Toc326010063)

[Use case model 2](#_Toc326010064)

[Use case New Workspace 3](#_Toc326010065)

[Use case Open Workspace 5](#_Toc326010066)

[Use case Save Workspace 7](#_Toc326010067)

[Use case Close Workspace 9](#_Toc326010068)

[Use case Define Architecture 10](#_Toc326010069)

[Use case Defined architecture diagram 11](#_Toc326010070)

[Use case Import Architecture 12](#_Toc326010071)

[Use case Export Architecture 13](#_Toc326010072)

[Use case Application properties 14](#_Toc326010073)

[Use case Analysed application overview 16](#_Toc326010074)

[Use case Analysed application diagram 17](#_Toc326010075)

[Use case Validate 18](#_Toc326010076)

[Use case Validate configuration 19](#_Toc326010077)

[Use case Export violation report 20](#_Toc326010078)

[Use case Change Language 21](#_Toc326010079)

[Toolbar 22](#_Toc326010080)

[Taskbar 22](#_Toc326010081)

[Functional Requirements 23](#_Toc326010082)

[Non-functional requirements 23](#_Toc326010083)

[Decisions and justification 24](#_Toc326010084)

[Physical models 25](#_Toc326010085)

[Physical class diagram 25](#_Toc326010086)

[Physical software partitioning model 26](#_Toc326010087)

[Architectural Rules 27](#_Toc326010088)

# Architectural significant requirements

## Use cases

* New Workspace
* Open Workspace
* Save Workspace
* Close Workspace
* Define architecture
* Defined architecture Diagram
* Import architecture
* Export architecture
* Application properties
* Analysed application overview
* Analysed architecture diagram
* Validate
* Validate configuration
* Export violation report
* Change Language

### Use case model



### Use case New Workspace

|  |  |  |
| --- | --- | --- |
| New workspace |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user is able to create a new workspace with new definitions. This workspace can be saved and reopened for later usage. | |
| Precondition | The HUSSAC-tool is opened. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks File -> New workspace or types Ctrl + N |  |
|  |  | 1. System shows the ‘New Workspace’-dialog |
|  | 1. Actor fills in the workspace-name |  |
|  | 1. Actor clicks ‘OK’ |  |
|  |  | 1. System creates the workspace |
|  |  | 1. System opens the Define internalframe (see use case Define Architecture) |
| Postcondition | The workspace is created within memory and the Define internalframe is opened. | |
| Alternative scenario | At step 3 |  |
|  | * 1. Actor clicks the ‘set application’ checkbox |  |
|  |  | * 1. System collapses the dialog to show the ‘set application’-form |
|  | * 1. Actor fills in the application properties (see use case Application properties) |  |
|  |  | * 1. System sends the application properties to the Define-service |
|  |  | * 1. System requests the Analyse-service to analyse the application |
| Postcondition AS | The workspace is created within memory and the application is analysed. | |
| Constraints | Both the workspace name and the application name must be alphanumeric, the only special symbols allowed are the dash (‘-‘) and the underscore (‘\_’). Workspace and application name may not be empty. | |

#### User interface

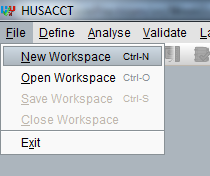


Figure 1. New workspace, step 1

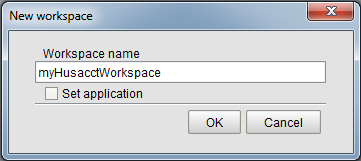


Figure 2. New workspace, step 2

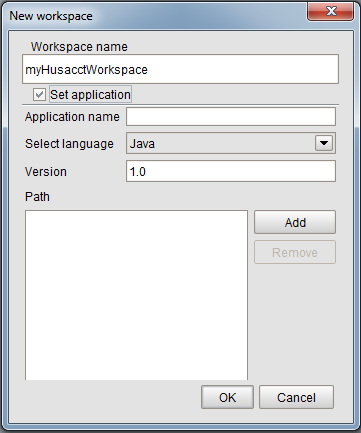


Figure 3. New workspace; alternative scenario step 3.2

### Use case Open Workspace

|  |  |  |
| --- | --- | --- |
| Open workspace |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | A workspace can be opened from a saved workspace-file. | |
| Precondition | The Husacct-tool is opened and a workspace file exists. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks File -> Open workspace or types Ctrl + O |  |
|  |  | 1. System shows the ‘Open Workspace’-dialog |
|  | 1. Actor selects open-method |  |
|  | 1. Actor fills in method-specific information |  |
|  | 1. Actor clicks ‘Open’ |  |
|  |  | 1. System sends workspace-data from file to the appropriate services |
|  |  |  |
| Postcondition | The workspace is opened with the available information loaded. | |
| Alternative scenario 1 | At step 3 |  |
|  | * 1. Actor selects ‘Husacct’ |  |
|  |  | * 1. System shows the ‘open as Husacct-file’-form |
|  | * 1. Actor specifies the location of the Husacct-file |  |
|  |  |  |
| Postcondition AS1 | The workspace is opened with the available information loaded. | |
| Alternative scenario 2 | At step 3 |  |
|  | * 1. Actor selects ‘XML’ |  |
|  |  | * 1. System shows the ‘open as xml-file’-form |
|  | * 1. Actor specifies the location of the xml-file |  |
| Postcondition AS2 | The workspace is opened with the available information loaded. | |
| Transformation rules | The data from the selected open-method is transformed to a xml-document | |

#### User Interface

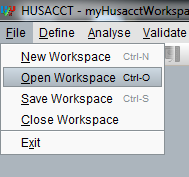
****

Figure 4. Open workspace; step 1

##### **Alternative scenario 1, step 3.2**

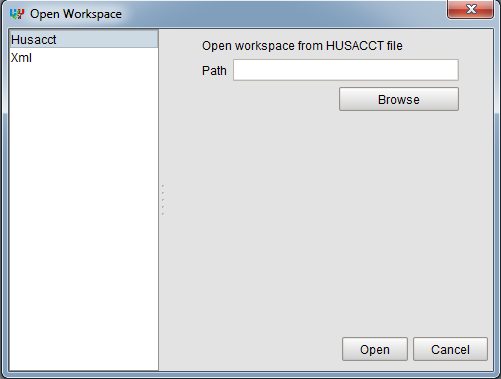
****

Figure 5. Open workspace; alternative scenario 1, step 3.2

##### **Alternative scenario 2, step 3.2**

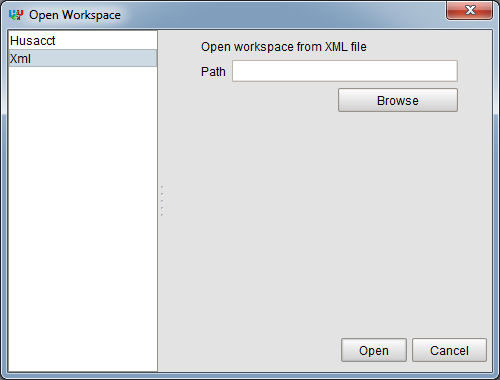


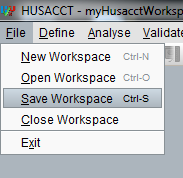
Figure 6. Open workspace; Alternative scenario 2, step 3.2

### Use case Save Workspace

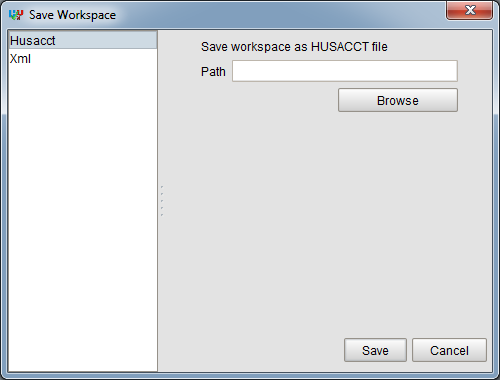
|  |  |  |
| --- | --- | --- |
| Save workspace |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The workspace can be saved with the save-method selected by the user. | |
| Precondition | A workspace is created. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks File -> Save workspace or types Ctrl + S |  |
|  |  | 1. System shows the ‘Save Workspace’-dialog |
|  | 1. Actor selects save-method |  |
|  | 1. Actor fills in method-specific information |  |
|  | 1. Actor clicks ‘Save’ |  |
|  |  | 1. System collects workspace-data from all services |
|  |  | 1. System saves workspace to the given location |
| Postcondition | The workspace is saved to the location specified by the user with the selected save-method. | |
| Alternative scenario 1 | At step 3 |  |
|  | * 1. Actor selects ‘Husacct’ |  |
|  |  | * 1. System shows the ‘save as Husacct-file’-form |
|  | * 1. Actor specifies the save-location and filename |  |
|  |  |  |
| Postcondition AS1 | The workspace is saved as a Husacct-file (.hu) to the specified location. | |
| Alternative scenario 2 | At step 3 |  |
|  | * 1. Actor selects ‘XML’ |  |
|  |  | * 1. System shows the ‘save as xml-file’-form |
|  | * 1. Actor specifies the save-location and filename |  |
| Postcondition AS2 | The workspace is saved as an XML-file (.xml) to the specified location. | |
| Transformation rules | Internal xml-document is transformed to the format that the selected save-method uses. | |
| Constraints | The filename must be alphanumeric, the only special symbols allowed are the dash (‘-‘) and the underscore (‘\_’). Only one dot (‘.’) is allowed in the filename (to prevent filenames like: myWorkspace.xml.xml.xml). Filename may not be empty. | |

#### User Interface

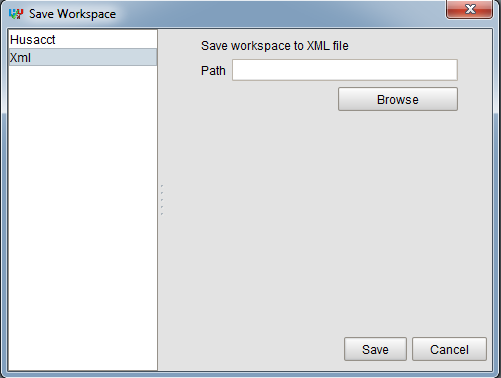
##### **Step 1**



##### **Alternative scenario 1, step 3.2**



##### **Alternative scenario 2, step 3.2**

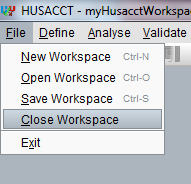
****

### Use case Close Workspace

|  |  |  |
| --- | --- | --- |
| Close workspace |  |  |
| Version | 1.0 | |
| Priority | Could have | |
| Actors | User | |
| Summary | The workspace can close the workspace. | |
| Precondition | A workspace is open. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks File -> Close workspace |  |
|  |  | 1. System closes all opened frames |
|  |  | 1. System resets all services |
| Postcondition | The workspace is closed. | |

#### User interface

##### **Step 1**

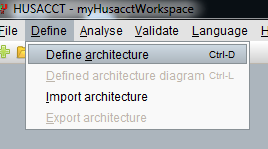
****

### Use case Define Architecture

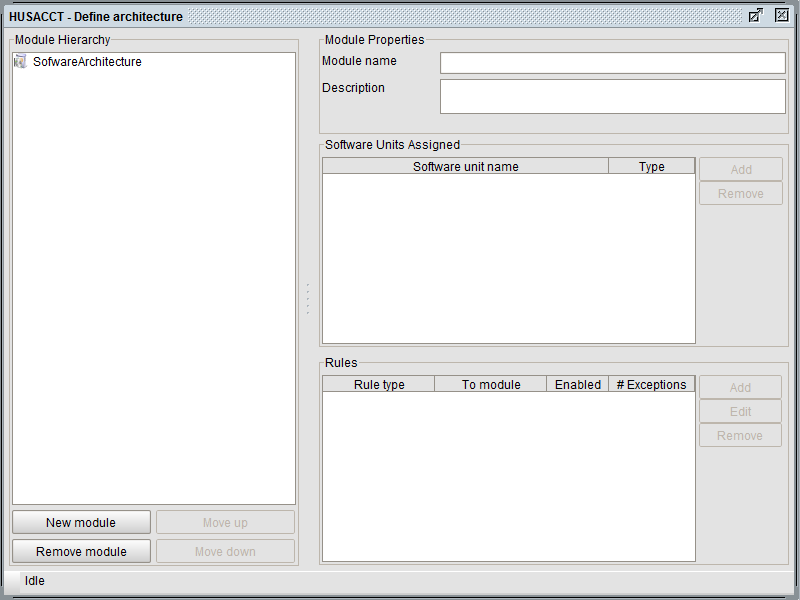
|  |  |  |
| --- | --- | --- |
| Define architecture |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user can define a logical architecture then map an application to it. | |
| Precondition | A workspace is open. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Define -> Define architecture or types Ctrl + D |  |
|  |  | 1. System makes a request for the ‘Define architecture’ internalframe from the Define-service |
|  |  | 1. System opens the ‘Define architecture’ internalframe |
| Postcondition | The ‘Define architecture’ internalframe is opened. | |

#### User interface

##### **Step 1**



##### **Postcondition**

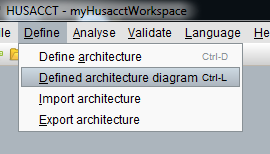


### Use case Defined architecture diagram

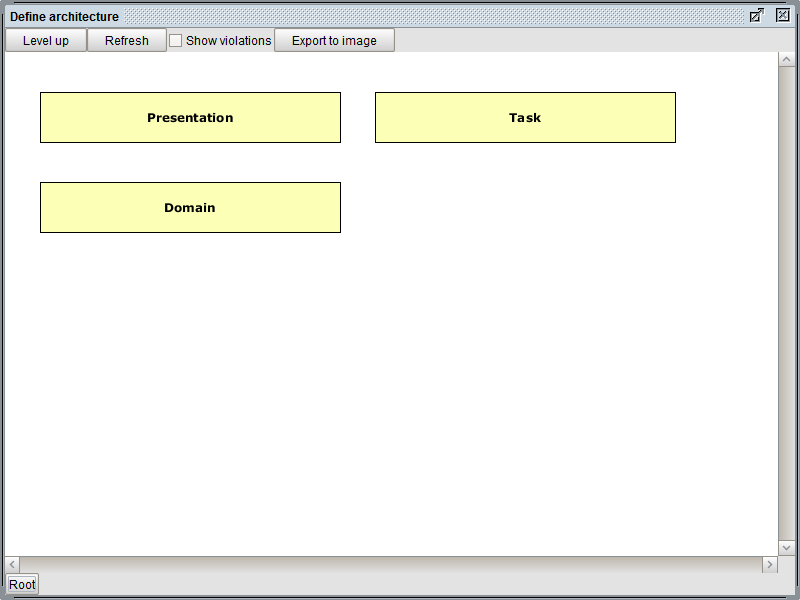
|  |  |  |
| --- | --- | --- |
| Defined architecture diagram |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user is able to open a graphical view of the defined architecture | |
| Precondition | A workspace is open, the architecture is defined. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Define -> Defined architecture diagram or types Ctrl + L |  |
|  |  | 1. System makes a request for the ‘Defined architecture diagram’ internalframe from the Graphics-service |
|  |  | 1. System opens the ‘Defined architecture diagram’ internalframe |
| Postcondition | The ‘Defined architecture diagram’ internalframe is opened. | |

#### User interface

##### **Step 1**



##### **Postcondition**

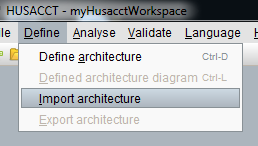
****

### Use case Import Architecture

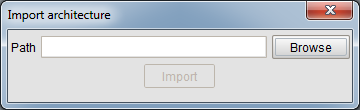
|  |  |  |
| --- | --- | --- |
| Import architecture |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user is able to import previously defined architecture. | |
| Precondition | A workspace is open. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Define -> Import architecture |  |
|  |  | 1. System shows ‘Import architecture’-dialog |
|  | 1. Actor specifies the location of the file |  |
|  | 1. Actor clicks ‘Import’ |  |
|  |  | 1. System transforms selected file to an xml-document |
|  |  | 1. System sends xml-document to Define-service |
| Postcondition | The Define-service processes xml-document. | |
| Transformation rules | System transforms selected file content to an xml-document. | |
| Constraints | The selected file must be .xml. | |

#### User interface

##### **Step 1**



##### **Step 2**

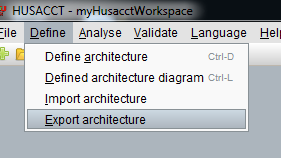


### Use case Export Architecture

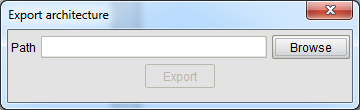
|  |  |  |
| --- | --- | --- |
| Export architecture |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user is able to export defined architecture. | |
| Precondition | A workspace is open, architecture is defined. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Define -> Export architecture |  |
|  |  | 1. System shows ‘Export architecture’-dialog |
|  | 1. Actor specifies the location of the file |  |
|  | 1. Actor clicks ‘Export’ |  |
|  |  | 1. System requests the architecture definition from the Define-service as xml-document |
|  |  | 1. System saves xml-document to specified location |
| Postcondition | The architecture is saved to an xml-file at the specified location. | |
| Constraints | The filename must be alphanumeric, the only special symbols allowed are the dash (‘-‘) and the underscore (‘\_’). Only one dot (‘.’) is allowed in the filename (to prevent filenames like: myWorkspace.xml.xml.xml). Filename may not be empty. | |

#### User interface

##### **Step 1**

****

##### **Step 2**

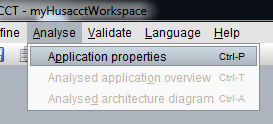


### Use case Application properties

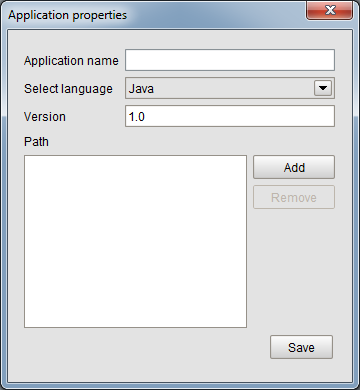
|  |  |  |
| --- | --- | --- |
| Application properties |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user is able to specify the application properties | |
| Precondition | A workspace is open, architecture is defined. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Analyse -> Application properties |  |
|  |  | 1. System shows ‘Application properties’-dialog |
|  | 1. Actor specifies the name, programming language, versionnumber and path of the application |  |
|  | 1. Actor clicks ‘Save’ |  |
|  |  | 1. System sends the application properties to the Define-service |
|  |  | 1. System requests the Analyse-service to analyse the application |
| Postcondition | The application is analysed and the application properties are saved. The application can be mapped to the defined architecture. | |
| Constraints | The application name must be alphanumeric, the only special symbols allowed are the dash (‘-‘) and the underscore (‘\_’). The application name may not be empty. | |

#### User interface

##### **Step 1**

****

##### **Step 2**

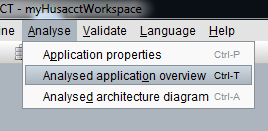


### Use case Analysed application overview

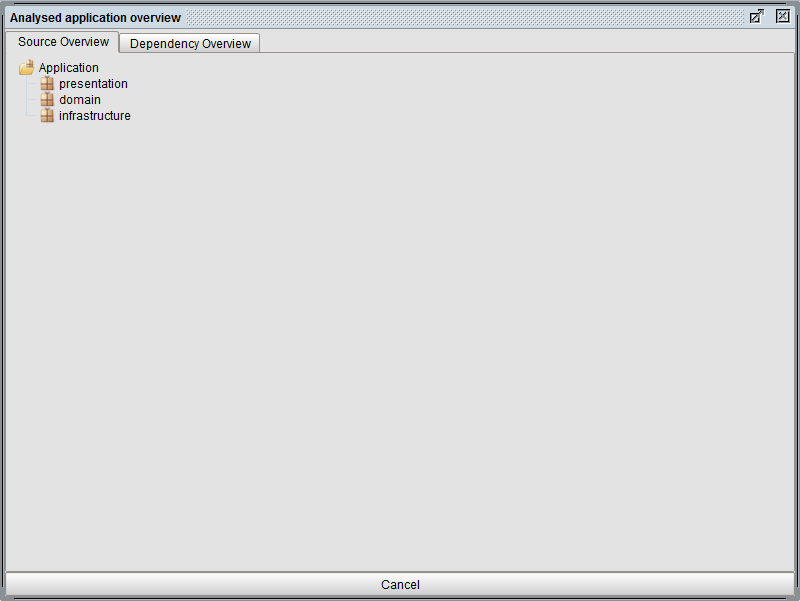
|  |  |  |
| --- | --- | --- |
| Analysed application overview |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user can show an overview of the analysed application. | |
| Precondition | A workspace is open, architecture is defined and an application is analysed. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Analyse -> Analysed architecture overview or types Ctrl + T |  |
|  |  | 1. System makes a request for the ‘Analysed architecture overview’ internalframe from the Analyse-service |
|  |  | 1. System opens the ‘Analysed architecture overview’ internalframe |
| Postcondition | The ‘Analysed architecture overview’ internalframe is opened. | |

#### User interface

##### **Step 1**



##### **Postcondition**

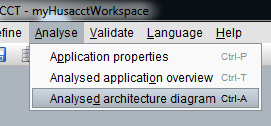
****

### Use case Analysed application diagram

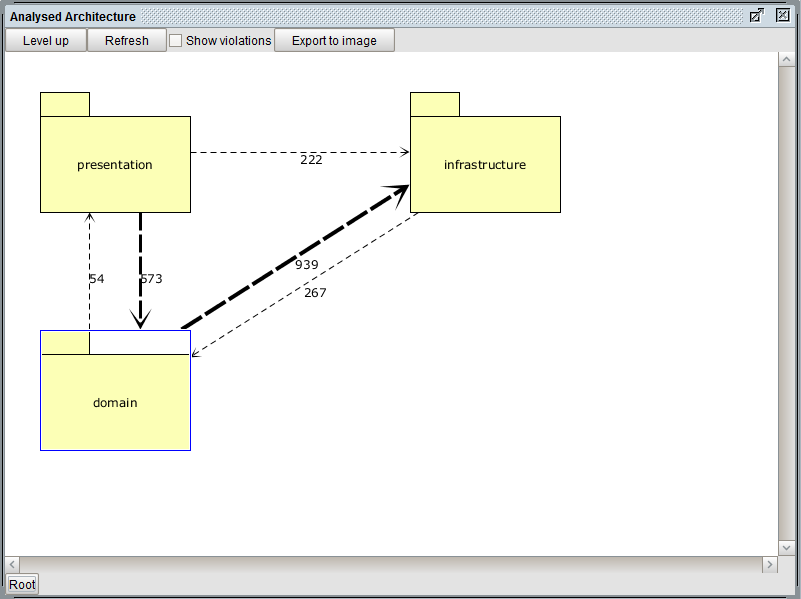
|  |  |  |
| --- | --- | --- |
| Analysed application diagram |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user can show a graphical view of the analysed application. | |
| Precondition | A workspace is open, architecture is defined and an application is analysed. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Analyse -> Analysed architecture diagram or types Ctrl + A |  |
|  |  | 1. System makes a request for the ‘Analysed architecture diagram’ internalframe from the Analyse-service |
|  |  | 1. System opens the ‘Analysed architecture diagram’ internalframe |
| Postcondition | The ‘Analysed architecture diagram’ internalframe is opened. | |

#### User interface

##### **Step 1**



##### **Postcondition**

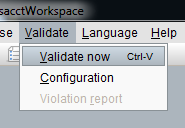


### Use case Validate

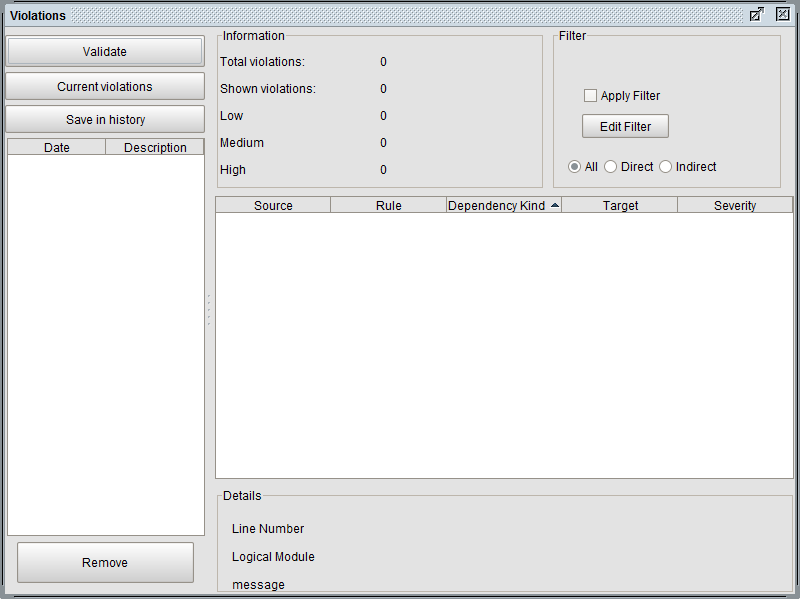
|  |  |  |
| --- | --- | --- |
| Validate |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user can validate an application. | |
| Precondition | A workspace is open, architecture is defined, an application is analysed and mapped to the architecture. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Validate -> Validate now or types Ctrl + V |  |
|  |  | 1. System makes a request for the ‘Violations’ internalframe from the Validate-service |
|  |  | 1. System opens the ‘Violations’ internalframe |
| Postcondition | The ‘Violations’ internalframe is opened. | |

#### User Interface

##### **Step 1**



##### **Step 3**

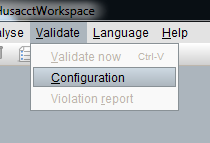


### Use case Validate configuration

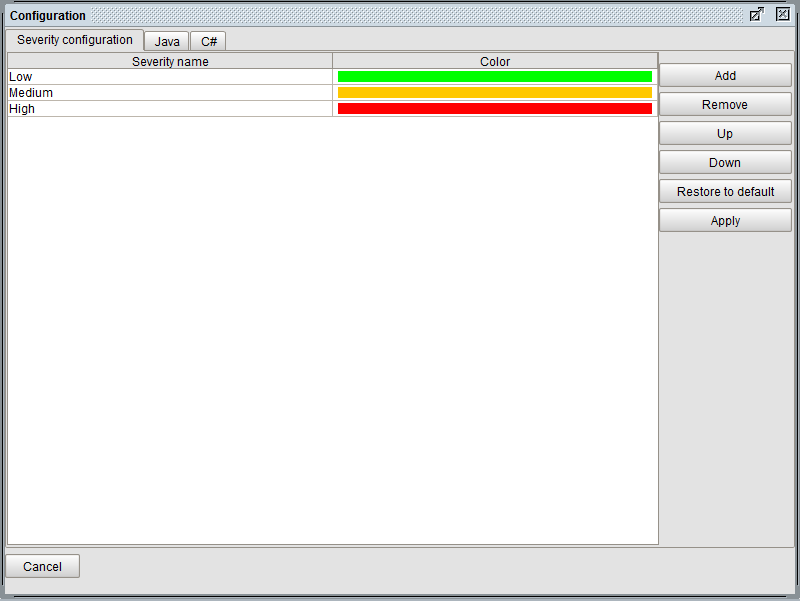
|  |  |  |
| --- | --- | --- |
| Validate configuration |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user can configure the violations and the way they are shown. | |
| Precondition | A workspace is open. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Validate -> Configuration |  |
|  |  | 1. System makes a request for the ‘Configuration’ internalframe from the Validate-service |
|  |  | 1. System opens the ‘Configuration’ internalframe |
| Postcondition | The ‘Configuration’ internalframe is opened. | |

#### User Interface

##### **Step 1**



##### **Step 3**

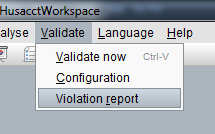


### Use case Export violation report

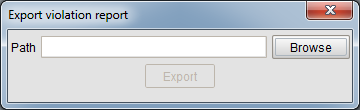
|  |  |  |
| --- | --- | --- |
| Export violation report |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user can export the violations. | |
| Precondition | A workspace is open, architecture is defined, an application is analysed and mapped to the architecture. The application is validated. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Validate -> Export violation report |  |
|  |  | 1. System show the ‘Export violation report’-dialog |
|  | 1. Actor clicks ‘Browse’ |  |
|  |  | 1. System makes a request for the filetypes from the Validate-service |
|  | 1. Actor specifies the location of the file and specifies the filetype |  |
|  | 1. Actor clicks ‘Export’ |  |
|  |  | 1. System sends the file and file-extension to the Validate-service |
| Postcondition | The Validate-service creates a report and saves the report to the specified location. | |
| Constraints | The filename must be alphanumeric, the only special symbols allowed are the dash (‘-‘) and the underscore (‘\_’). Only one dot (‘.’) is allowed in the filename (to prevent filenames like: myReport.pdf.html.xml). Filename may not be empty. | |

#### User Interface

##### **Step 1**



##### **Step 3**

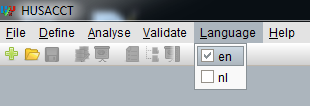


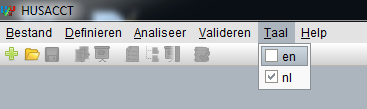
### Use case Change Language

|  |  |  |
| --- | --- | --- |
| Validate configuration |  |  |
| Version | 1.0 | |
| Priority | Must have | |
| Actors | User | |
| Summary | The user can change the language the application is in. | |
| Precondition | The Husacct-tool is opened. | |
| Main scenario | **Actor actions** | **System actions** |
|  | 1. Actor clicks Language -> Language that isn’t selected |  |
|  |  | 1. System checks the values with the appropriate resourcefile |
|  |  | 1. System notifies the listeners of the Locale-service |
|  |  | 1. System changes the language of the GUI |
| Postcondition | The language of the GUI is changed. | |

#### User interface

##### **Step 1**





### Toolbar

A toolbar is implemented to increase the ease of use of the HUSACCT-tool. All the buttons in the toolbar are directly linked to the items in the menubar.

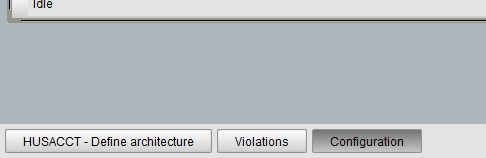


The buttons are from left to right:

* New workspace
* Open workspace
* Save workspace
* Define architecture
* Defined architecture diagram
* Application properties
* Analysed application overview
* Analysed application diagram
* Validate

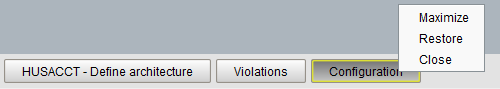
### Taskbar

The taskbar is shown when an internalframe is opened. The taskbar is implemented to give a clear overview of all the opened internalframes.



Left-click on a button will put the corresponding frame to the front.

Right-click on a button will open a contextmenu.



Maximize, maximizes the frame.  
Restore, sets the size of the frame to the default and centers the frame.  
Close, close the frame.

## Functional Requirements

|  |  |  |
| --- | --- | --- |
| **CFR** | **Description** | **Priority** |
| 1 | Ability to call the define service. This service is responsible to handle several operations which has to do with defining architecture rules. | Must have |
| 2 | Ability to call the analyse service. This service is responsible to handle several operations which has to do with analyzing. | Must have |
| 3 | Ability to call the validate service. This service is responsible to handle several operations which has to do with validating. | Must have |
| 4 | Ability to save project. | Must have |
| 5 | Ability to close project. | Should have |
| 6 | Ability to open project. | Must have |
| 7 | Ability to open a specific view | Should have |
| 8 | Ability to close a specific view. | Should have |
| 9 | Ability to create a new project. | Must have |
| 10 | Ability to show the architectural graphics. | Must have |

## Non-functional requirements

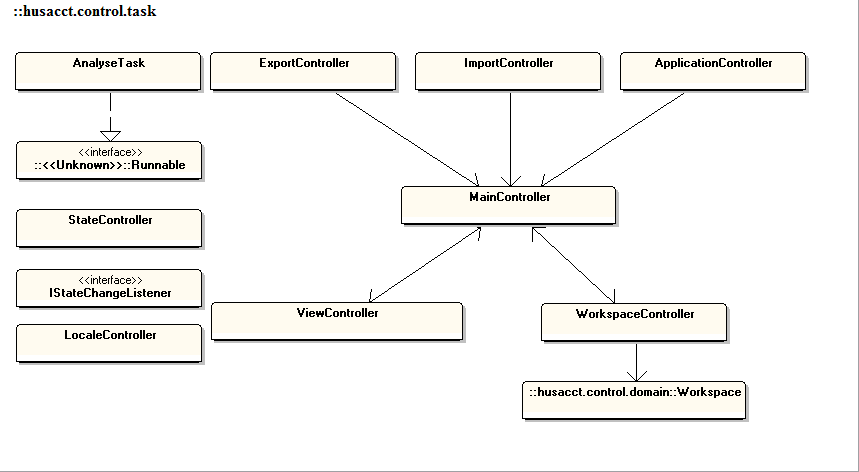
|  |  |  |
| --- | --- | --- |
| **NFR** | **ISO 9123 attr.** | **Requirement** |
| **1.** | **Functionality** |  |
| 1.1 | Suitability | Configuration of the user language must be possible. |
| 1.2 | Interoperability | Communication with other applications (Maven, Sonar, Ant ...) must be supported. E.g. activate, import; export. |
|  | **Reliability** |  |
|  | Maturity | The tool must not go down in case of a failure, but generate a meaningful error message. |
|  | **Usability** |  |
|  | Understandability | The division in required steps (manual and automated) must be clear to the user of the tool. |
|  | Understandability | It must be clear to the user what he/she is working on |
|  | **Maintainability** |  |
|  | Analyseability  Testability | Taking over the development of the tool by other development teams must be unproblematic. |
|  | **Portability** |  |
|  | Adaptability | Plug-ins for IDE’s (Eclipse, Sonar ...) must be provided. |
|  | Installability | The tool must be easy to install. |
|  | Installability | The tool must provide a complete installation package (no extra downloads needed). |
| 4.4 | Adaptability | The tool must be usable in several operating systems. |

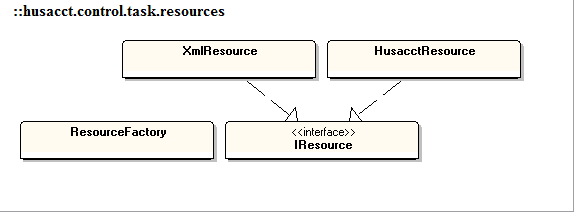
## Decisions and justification

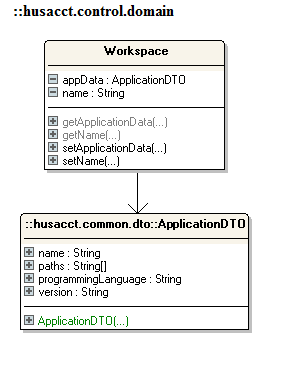
|  |  |
| --- | --- |
| **Decision** | **Justification** |
| In order to ensure the user language of the SACC-tool is independent, we will  use ResourceBundles in Java. We will implement the observer pattern to notify the other services of a language change. | NF1.1 |
| In order to ensure other development teams can easily maintain the source code of the SACC-tool, we make use of GIT combined with GITHUB. | NF4.1 |
| In order to ensure the SACC-tool is operating system independent, we develop  the GUI in Java Swing. | NF5.4 |
| To support plugin development based on the tool, the presentation and task layer are separated. | NF1.2 |
| Services can call a method to display error and/or info messages on top of the main gui | NF2.1 |
| The system keeps track of the state of the workspace, making sure that buttons are enabled/disabled at the appropriate state | NF3.1 |
| Libraries are included within the project, groups/services and testsuites are separated in packages, build scripts are all relative. This ensures that when importing the project in eclipse that all dependencies are met. | NF4.1 |
| The tool is packaged including all dependencies | NF5.2 NF5.3 |
| It was decided not to use a tabbed panes because it wasn't possible to show several windows at once. Instead a taskbar was implemented to give an overview of the opened frames. | NF3.2 |
| JDOM2 was chosen for its ease in use and the ability to merge multiple elements into one document when saving the workspace. | FR 4 FR 6 |

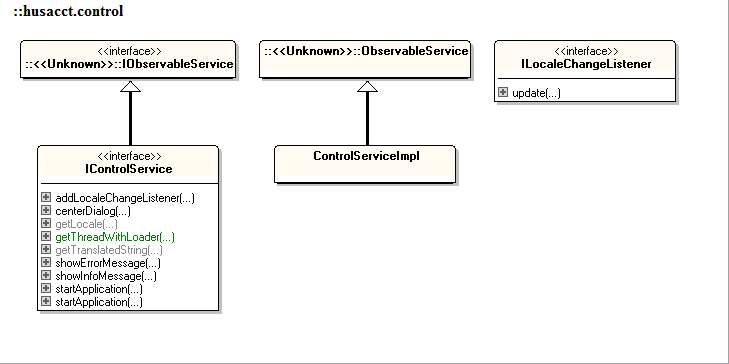
# Physical models

## Physical class diagram









## Physical software partitioning model



## Plugin component diagram

## 

### Architectural Rules

|  |  |  |
| --- | --- | --- |
|  | **Rules on the contents of a module** | Restricts the contents of a module |
|  | A module can have only one parent | - |
|  | No loops in the module hierarchy | - |
|  | Unique responsibility | - |
|  | Visibility convention | Only the interfaces are visible to the other services. |
|  | Naming convention | - |
|  | Subclass convention | - |
|  | Interface convention | - |
|  | **Rules on the legality of Dependency** |  |
|  | Is allowed to use | View is allowed to use Controller and Controller is allowed to use View  Controller is allowed to use Define, Analyse, Validate and Architecture Graphics Interfaces |
|  | Is only allowed to use | - |
|  | Is the only module allowed to use | - |
|  | Must use | - |
|  | Is not allowed to | Define, Analyse, Validate and Architecture Graphics are not allowed to use Controller |
|  | … use modules in higher layer | - |
|  | … use modules in a not directly lower layer | - |
|  | Exceptions to all those rules | Define, Analyse, Validate and Architecture Graphics are allowed to give an Exception to Controller or to add an observer to LocaleService. |
|  | **Rules on Dependency Limitation** |  |
|  | Publish Subscribe pattern (or Observer) | Observer pattern is implemented. The LocaleService is the Observerable and the other services have their own observer.  A notify to the observers is send when the user language has been changed in the GUI. |
|  | Facade pattern | The different services(Define, Analyse, Validate and Architecture Graphics) do have a façade(interface) which the controller communicates with. |
|  | Data Transfer Object | - |
|  | Error messages conform X | When an export or import fails an Exception is sent to us by the appropriate service. |
|  | Communication protocol conform Y | When an export or import succeeds we get a message (Boolean) confirming the import/export. |
|  | No cycles in the dependency | - |