|  |
| --- |
| Hogeschool Utrecht |
| Attachment A |
| Test scripts |
| Architecture Graphics, System documentation |

|  |
| --- |
| **Team Architecture Graphics**  Tom de Bruijn  Johan van der Slikke  Guido van Tricht  Patrick van der Willik  13-06-2012 |

# Black Box Test

## Test Data

### Defined architecture

**Data:**  
HUSACCT-BenchmarkApplication - <https://github.com/HUSACCT/HUSACCT-BenchmarkApplication/>

|  |  |
| --- | --- |
| **Layers** | **Included packages** |
| Presentation | Presentation.\* |
| Domain | domain.\* |
| Infrastructure | infrastructure.\* |

Components in domain: modules: domain infrastructure and presentation

Packages in domain: blog facebook flickr foursquarealternative google\_plus gowalla hyves language lastfm linkedin locationbased music netlog orkut pinterest shortcharacter spotify stumbleupon

Classes in flickr: Flickr FlickrPicture Tag

**Defined rules:**

|  |
| --- |
| Presentation is not allowed to use Infrastructure |

### Analyse architecture zie githubrepositories

**Data:**  
HUSACCT-BenchmarkApplication - <https://github.com/HUSACCT/HUSACCT-BenchmarkApplication/>

## View Defined Architecture

Precondition: The user has defined an architecture through the define User Interface and the define service.

Actions

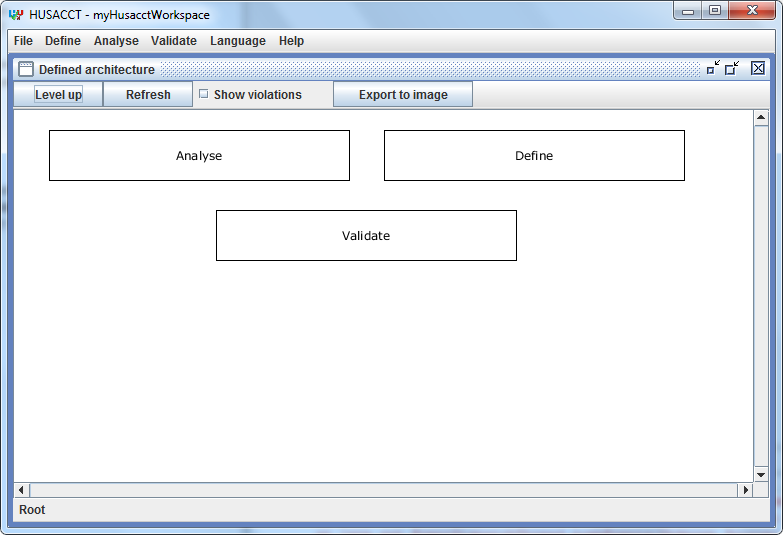
1. In the menu bar the user clicks “Show logical architecture graphics” (name subject to change.)
2. The Architecture Graphics Frame is shown for the defined architecture. It should show the defined layers.   
   Data: presentation, domain, infrastructure.
   1. If these layers are mapped to analysed software units, dependency lines are shown with a number next to them representing the amount of dependencies.

Figure Action 1

Figure Action 2

* 1. These dependency lines between layers should be thicker or thinner compared to other lines based on the amount of dependencies. More dependencies results in a thicker line.

|  |  |  |  |
| --- | --- | --- | --- |
| **Expected Result** | **Realized Result** | **Satisfying?** | **Solution** |
| The defined architecture is shown. This must be exactly the same as what has been defined. |  |  |  |

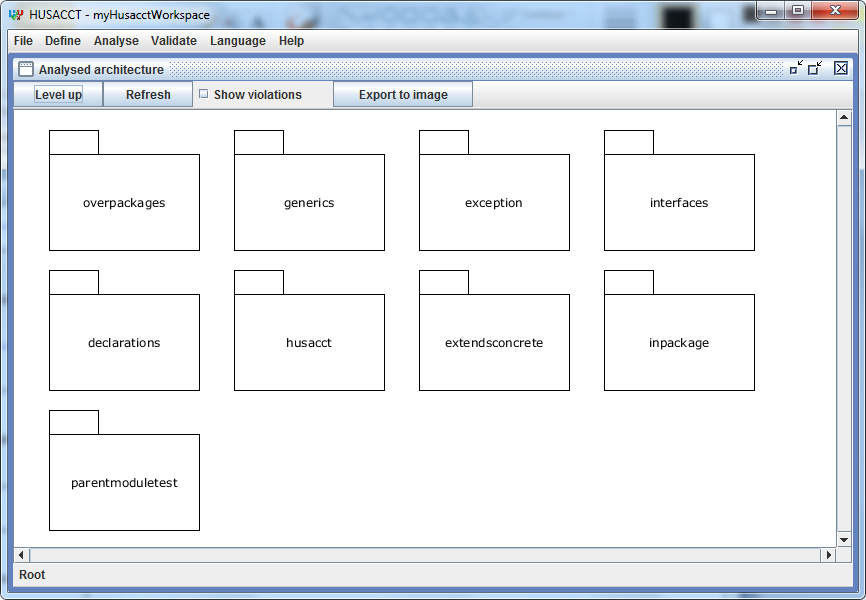
## View Analyzed Architecture

Figure Action 1

Precondition: The user has analysed the source code of a selected application.

Actions:

1. The user opens the analysed architecture GUI through the menu bar by clicking “Analyse” -> “Show analysed architecture graphics” (name subject to change). (Figure 3)
2. The Architecture Graphics Frame opens. (Figure 4) It should contain the analysed architecture.  
   Data: husacct: .
   1. Dependency lines are shown with a number next to them representing the amount of dependencies.
   2. Optional feature: These dependency lines between layers should be thicker or thinner compared to other lines based on the amount of dependencies. More dependencies results in a thicker line.



|  |  |  |  |
| --- | --- | --- | --- |
| **Expected Result** | **Realized Result** | **Satisfying?** | **Solution** |
| The defined architecture is shown. This must be exactly the same as the architecture analyzed from the source code. |  |  |  |

## Zoom on Defined Architecture

Precondition: The user has defined an architecture and has opened the logical Architecture Graphics frame.

Actions:

1. Layer zoom:
   1. The user double clicks on the layer [Data: Domain].
   2. The system should clear the drawing’s existing figures, but this is not visible to the user.
   3. The system draws the following in the empty drawing:
      1. Data: modules: domain infrastructure and presentation
2. Component zoom:
   1. The user double clicks on the component [Data: domain].
   2. The system should clear the drawing’s existing figures, but this is not visible to the user.
   3. The system draws the following in the empty drawing:
      1. Data: Packages: blog facebook flickr foursquarealternative google\_plus gowalla hyves language lastfm linkedin locationbased music netlog orkut pinterest shortcharacter spotify stumbleupon
3. Package zoom:
   1. The user double clicks on the package [Data: flickr].
   2. The system should clear the drawing’s existing figures, but this is not visible to the user.
   3. The system draws the following in the empty drawing:
      1. Data: Class: Flickr FlickrPicture Tag

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action type** | **Expected Result** | **Realized Result** | **Satisfying?** | **Solution** |
| Layer zoom | The child architecture from the Layer is shown. |  |  | - |
| Component zoom | The child architecture from the Component is shown. |  |  | - |
| Package zoom | The child architecture from the Package is shown. |  |  | - |

## Zoom on Analysed Architecture

Precondition: The user has analysed an application and has opened the analysed Architecture Graphics frame.

Actions:

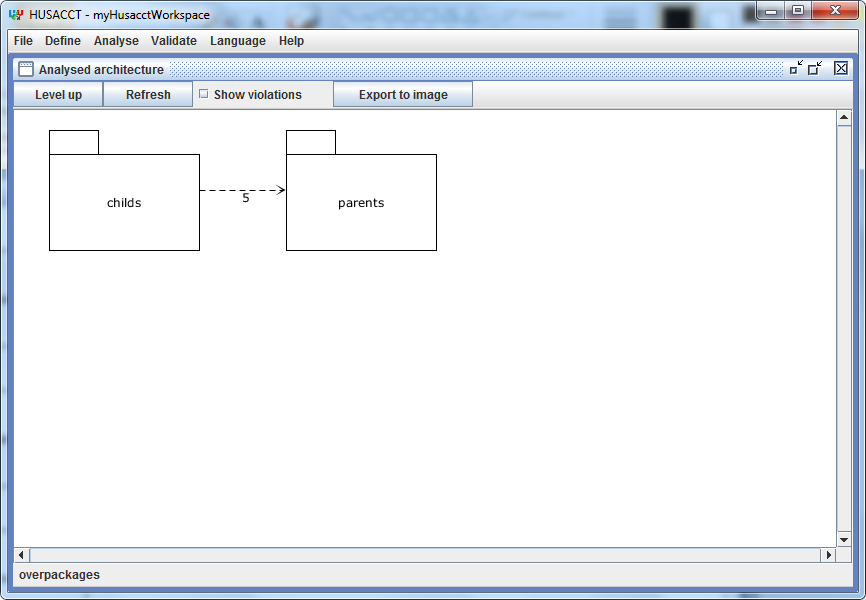
1. Package zoom:
   1. The user double clicks on the package [Data: domain]. (Figure 4)
   2. The system should clear the drawing’s existing figures, but this is not visible to the user.
   3. The system draws the following in the empty drawing:
      1. Data: Packages: blog facebook flickr foursquarealternative google\_plus gowalla hyves language lastfm linkedin locationbased music netlog orkut pinterest shortcharacter spotify stumbleupon

Figure4 Action 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action type** | **Expected Result** | **Realized Result** | **Satisfying?** | **Solution** |
| Package zoom | The child architecture from the Package is shown. |  |  | - |

## C:\Users\Guido\Intraserve\Dropbox\Thema Opdracht specialisatie\Results\Construction I\Test Screenshots\Select module.pngShow Violations on Defined Architecture

Precondition: The user has defined an architecture through the define User Interface and the define service, and has opened the Defined Architecture Graphics.

Actions

1. In the graphics menu bar the user clicks “Show violations” (name subject to change.)
2. The violated modules and dependencies are shown in a color.
   1. Violations in modules and dependencies are shown in a color based on the severity of a violation.(These colors can be defined by the user)
      1. Yellow: Low
      2. Orange: Medium
      3. Red: High

Figure Action 1

* 1. The system draws the following violations:
     1. Data: violations detected and shown between the two modules. (see rules)\* They are present on every zoom level.

|  |  |  |  |
| --- | --- | --- | --- |
| **Expected Result** | **Realized Result** | **Satisfying?** | **Solution** |
| The violations of the shown modules are shown. |  |  |  |

## C:\Users\Guido\Intraserve\Dropbox\Thema Opdracht specialisatie\Results\Construction I\Test Screenshots\Select module.pngShow Violations on Analysed Architecture

Precondition: The user has analysed an architecture through the analyse User Interface and the analyse service, and has opened the Analysed Architecture Graphics.

Actions

1. In the graphics menu bar the user clicks “Show violations” (name subject to change.)
2. The violated modules and dependencies are shown in a color.
   1. Violations in modules and dependencies are shown in a color based on the severity of a violation.(These colors can be defined by the user)
      1. Yellow: Low
      2. Orange: Medium

Figure Action 1

* + 1. Red: High
  1. The system draws the following violations:
     1. Data: violations between presentations and infrastructure. They are present on various physical paths are set and they are shown on every zoom level. \*

|  |  |  |  |
| --- | --- | --- | --- |
| **Expected Result** | **Realized Result** | **Satisfying?** | **Solution** |
| The violations of the shown modules are shown. |  |  |  |

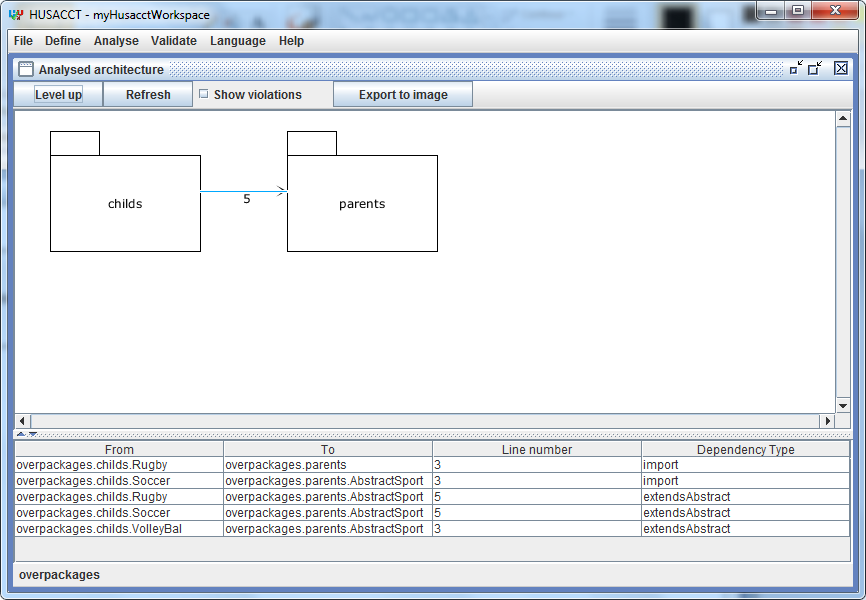
## C:\Users\Guido\Intraserve\Dropbox\Thema Opdracht specialisatie\Results\Construction I\Test Screenshots\Select module.pngShow properties of selected figure

Precondition: The user has analysed or defined an architecture through the analyse or define User Interface and the analyse or define service, and has opened the Analysed or Defined Architecture Graphics.

Actions

1. The user selects a figure [Data: domain].
   1. Figure is a physical figure.
   2. Figure is a logical figure.
2. The properties view is shown
   1. When the violations are shown, these violations are shown in the properties of a module [Data: more than one violation is visible in the properties pane].
   2. When the dependencies are shown in the properties of a dependency [Data: more than one dependency is visible in the properties pane].

Figure Action 1



|  |  |  |  |
| --- | --- | --- | --- |
| **Expected Result** | **Realized Result** | **Satisfying?** | **Solution** |
| The dependencies of the physical selected module are shown. |  |  |  |
| The dependencies of the logical selected module are shown. |  |  |  |
| The violations of the physical selected module are shown. |  |  |  |
| The violations of the logical selected module are shown. |  |  |  |

Figure Action 2

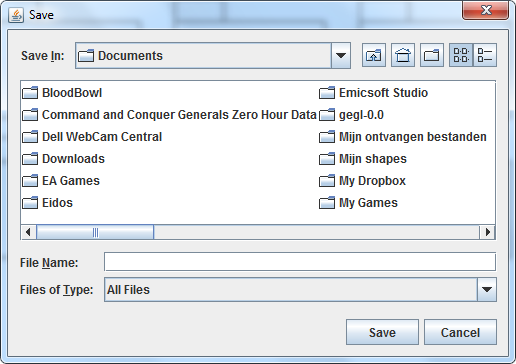
## C:\Users\Guido\Intraserve\Dropbox\Thema Opdracht specialisatie\Results\Construction I\Test Screenshots\Zoomed.pngExport to image

Precondition: The user has analysed or defined an architecture through the analyse or define User Interface and the analyse or define service, and has opened the Analysed or Defined Architecture Graphics.

Actions

1. In the graphics menu bar the user clicks “Export to image” (name subject to change.)
2. The Export to Image Frame is shown for the shown architecture. It should show the documents folder.
3. The user selects an folder to save the file image in, defines a “File Name” and clicks “Save” [Data: ~/Desktop/export-graphics.png].

Figure Action 1



|  |  |  |  |
| --- | --- | --- | --- |
| **Expected Result** | **Realized Result** | **Satisfying?** | **Solution** |
| The image is saved in the defined folder with the defined file name. |  |  |  |

Figure Action 2

# White Box Tests

In the HUSACCT project itself several JUnit tests are included. Much of the Graphics Service cannot be tested however, as it is all part of the JHotDraw library which can only be tested manually.