|  |
| --- |
| HUSACCT |
| Architecture Notebook – Architecture Graphics |
|  |

|  |
| --- |
| Team 5  29-5-2012 |

# Table of Contents

[Table of Contents 1](#_Toc326072345)

[Architectural significant requirements 2](#_Toc326072346)

[Functional 2](#_Toc326072347)

[Non Functional 2](#_Toc326072348)

[Decisions and Justification 3](#_Toc326072349)

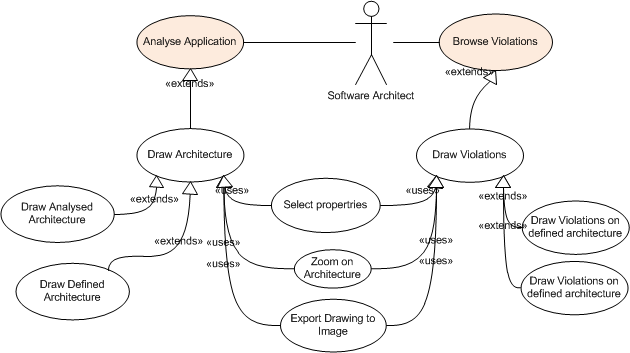
[Component Model 4](#_Toc326072350)

[Physical software partitioning model 0](#_Toc326072351)

# Architectural significant requirements

The requirements for HUSACCT 2012 are described below.

## Functional



## Non Functional

|  |  |  |
| --- | --- | --- |
| **Id** | **Non-functional Requirement** | **ISO 9126 Quality Attribute** |
| NF1 | This version should only be able to analyze Java and C#-code. But the system should be able to analyze other programming languages in a later stage. | Maintainability.Changeability |
| NF2 | The system should be used in different operating environment. | Functionality.Interopability |
| NF3 | This version should only be able to be used as stand-alone application, eclipse plugin and maven plugin. But the system should be able to be used in any other kind of plugin in a later stage. | Maintainability.Changeability |
| NF4 | The system should be easy to use. | Usability.Operability |
| NF5 | The application should perform smoothly | Efficiency.Time Behaviour |
| NF6 | The application should limit memory usage as much as possible | Efficiency.Efficiency Compliance |
| NF7 | The application should remain responsive during CPU intense tasks | Efficiency.Resource utilization / |

# Decisions and Justification

Some important decisions are described below with their justification.

|  |  |
| --- | --- |
| **Decision** | **Justification** |
| We are going to use the Java library JHotDraw. With this library we will be able to draw the architecture of Java based source code as for other programming languages. | NF1 |
| We are going to use the Java library JHotDraw. This library can be used in Windows, Linux and Mac. | NF2 |
| We use the Factory Pattern for an abstract solution for the usage of the different DTOs provided by the other teams. | NF1 |
| We will use JInternalFrames to show our graphics view. This decision was made by the Control team. | NF3 |
| We’ve separated presentation and task logic into two separate layers. The task layer is does not contain direct knowledge of which technologies (JHotDraw) used in the presentation layer. | NF2 |
| Logic of the file system for exporting the images of a drawing is present in our service (abstraction layer) so that control does not have to make a separate use case for this feature. |  |
| We’ve added calls to the graphics interface to allow the direct call to show violations in a Drawing. This allows for less user interaction. | NF4 |
| We’ve separated calls to retrieve the drawing JInternalFrames and to draw them. This way we decrease the amount of requests in case the drawing has already been drawn. | NF3 |
| Data is requested from the other services on a level by level basis. This way we only receive information we are using in the drawing and we do not contain any logic of the structure of the analysed architecture of an application. | NF3 |
| Data such as dependencies and violations are retrieved by multiple calls to the analyse and validate services for each combination of the DTOs. This way we do not contain any logic of (Analysed)ModuleDTOs to DependencyDTOs and ViolationDTOs. |  |
| Although not used decorators are present in the presentation layer for figures. This allows for more dynamic behavior to be added later on in development. Decorators allow certain behavior or graphical additions to be added. | NF1 |
| We’ve created separate controllers for the analyse and define services. The logic that controls the drawing mechanism is present in the DrawingController. The separate controllers handle only the communication with the analyse, define and validate services. |  |
| We manage all types of DTOs and what they represent (figures) in a FigureMap. This contains HashMaps that links figures to DTOs for easy retrieval of DTOs without putting the logic in the DrawingController. | NF1 |
| A UserInputListener is implemented into the controllers to handle requests from the presentation layer. This allows for other usages as well in the future such as a context sensitive right-click mouse menu. |  |
| Other services (analyse, define and validate) do not inform us of any changes in their components thus we implemented a refresh option to allow users to redraw the drawings based on the new data. | NF4 |
| In order to position the drawn figures in the drawing we make use of LayoutStrategies. We aim to implement two different kind of LayoutStrategies. One that will position the figures on a table basis (columns and rows) and one that will take into account the shortest routes to other elements for dependency and violation lines. |  |
| We are using threading in order to keep the application responsive during CPU intensive work | NF4 / NF7 |
| We’re providing multi-input methods for commands in order to provide the user with a rich user experience | NF4 |

# 

# C:\Users\Guido\Intraserve\Dropbox\Thema Opdracht specialisatie\Results\Construction II\29 mei bespreking\Componenten Model.pngComponent Model

|  |  |  |
| --- | --- | --- |
| Component | Classes | Description |
| GraphicsService | IGraphicsService, GraphicsServiceImpl | Handles all the request received in the service and sends requests to other components to get the needed information. |
| Task | DefinedController, AnalysedController, DrawingController, FigureMap | Handles all the requests from the GraphicsService Component to create the needed information. |
| Presentation | GraphicsFrame, DrawingView, Drawing | Builds up the frames. |
| Figures | BaseFigure, RelationFigure, ModuleFigure, ParentFigure, ClassFigure, AbstractClassFigure, ComponentFigure, SubsystemFigure, LayerFigure, PackageFigure, InterfaceFigure, FigureFactory, RoundedLiner, AbsoluteLocator, ElbowLiner | This is the part where JHotDraw is most active. The Figures Component is responsible for creating all the figures and lines of the architecture. |
| MenuBars | ContextMenu, GraphicsMenuBar, ZoomLocationBar, LocationButtonActionListener | This component creates all the Menu bars and handles the button clicks. |
| LineStrategies | ElbowLineSeparationStrategy, ConnectorLineSeparationStrategy, DistatiatedChopRectangleConnector, ILineSeparationStrategy | Here the strategy is selected for the view of the architecture. |
| Util | ListUtils, DrawingDetail, DrawingLayoutStrategy, Geom, UserInputListener | This component is used for drawing and handling some minor details of the view. |
| Threading | DrawingMultiLevelThread, DrawingSingleLevelThread, DrawingLinesThread | This component handles the threads to create the architecture view. |
| Abstraction | FileManager | The Abstraction Component is used to export the shown architecture to a image. |

# Physical software partitioning model



|  |  |
| --- | --- |
| Service | Description |
| ControllService | The ControllService is used for handling the language. |
| DefineService | The DefineService is used to receive the necessary data to create a Logical Architecture. |
| AnalyseService | The AnalyseService is used to receive the necessary data to create a Physical Architecture and displaying data about violations and dependencies. |
| ValidateService | The ValidateService is used to receive the necessary data to show the violations on top of the shown Physical Architecture. |