Design Document
Institute for Computer Graphics and Vision
Graz University of Technology
Graz

Cerberus
Design Document

Marc Streit
marc.streit@student.tugraz.at

DI Michael Kalkusch
kalkusch@icg.tugraz.at

July 2006

Contents

Chapter 1

Introduction

This project aims at manipulation and visualization of DAN-Micro array data enriched by medical clinical data. The goal is to provide a tool for medical experts and microbiologists.

Chapter 2

Used libraries

2.1 Java OpenGL (JOGL)

The JOGL project hosts the development version of the Java Binding for the OpenGL API, and is designed to provide hardware-supported 3D graphics to applications written in Java ((JOG06).

2.2 Standard Widget Toolkit (SWT)

SWT is an open source widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented ((SWT06)).

2.3 JGraph

JGraph is open source graph component available for Java which uses the Swing GUI library. For more details visit (JGG06).

Chapter 3

Class Design

3.1 Modules

The Cerberus Software can be devided in several module packages. The packages itself are intended to be as seperate as possible with a clear interface for moudle intercommunication. The simple reason for this approach is the easier changeability and extendability. In figure 3.1 the module design is visualized using an UML package diagram.

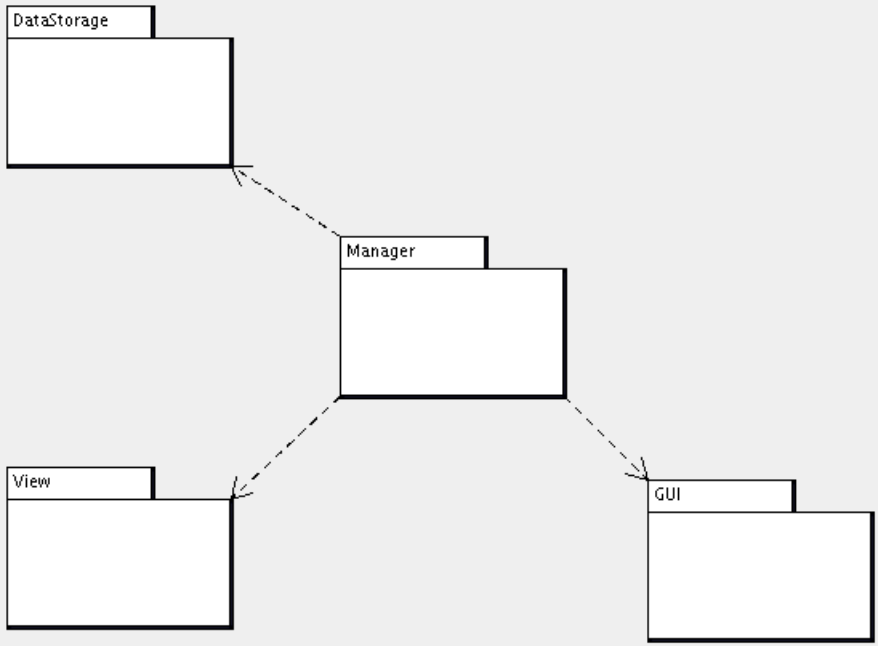

Figure 3.1: *Cerberus Modules*

In the following sections the modules are explained in more detail.

3.1.1 Manager

All components are handled by Mangers. This opens the possibility to use Abstract Factories (Design Pattern "Abstract Factory"). Since all above described components are only Interfaces, all components / objects can be created by factories (Design Pattern "Factory").

3.1.2 Data

DataStorage

Arrays filled with INT, FLOAT, DOUBLE, STRING, BOOLEAN

Selection

Define regions on DataStorage-Objects

Set

Holds a collection of Selections which refer to a DataStorage. Pipe-and-Filers Design Pattern (DataStorage-Selection-Set) Selection and Set can be accessed by Iterators. Design Pattern "Iterator".

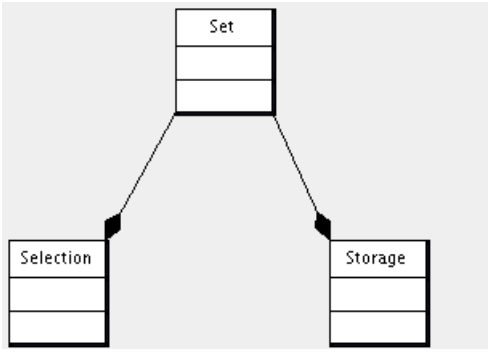

Figure 3.2: *Data Module Class Design*

3.1.3 GUI

The application will be built using the SWT. The Frame class is the basis of all GUI elements in the Cerberus application. The inheritence structure is shown in figure 3.3 This class is implemented as an interface which makes it basically possible to exchange the GUI library SWT with Swing. The Swing dependency is needed anyway because JGraph is used for visualizing and editing Graphs (needed for Pathways). JGraph takes only an Swing Frame for drawing. Therefore the Swing Frame needs to be embedded in a SWT Frame. Obviously this is a workaround but JGraph is a very powerful Graph Library on which we don't want to surrender. The Jogl render area can be embedded in an SWT Fram without problems.

3.1.4 View

In figure 3.4 the class design of the view module is visualized in UML. All special view types (e.g. heatmaps, pathways, etc.) are inherited from a general View interface. The abstract interface hides the implementation of the specific Views. The Views are created by a View-Factory (which follows the Abstract Factory Design Pattern).

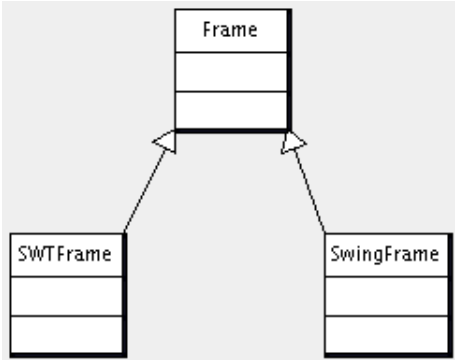

Figure 3.3: *GUI Module Class Design*

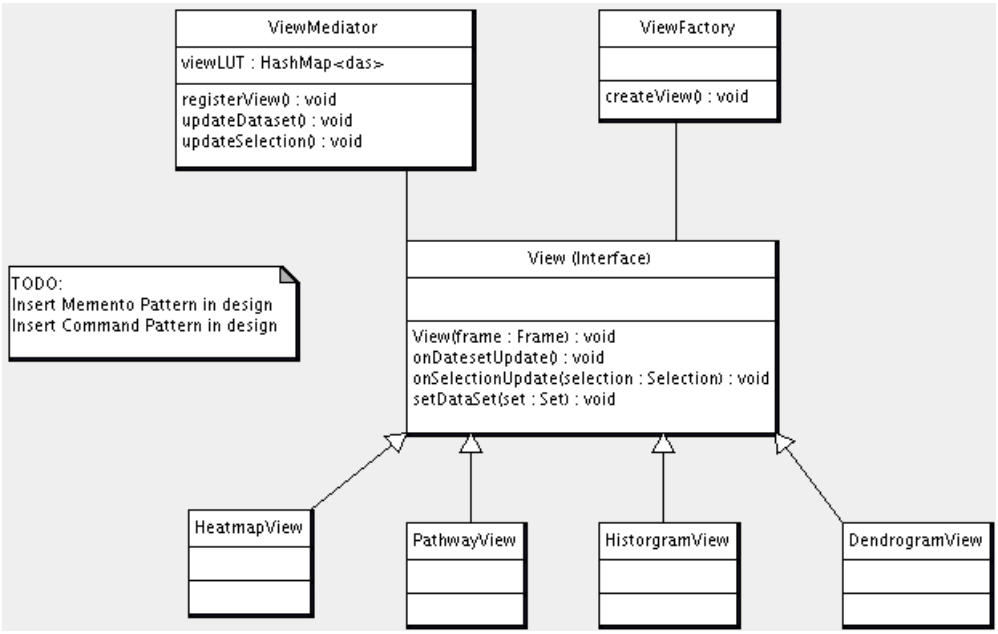

Figure 3.4: *View Module Class Design*

DataViewMediator

The View objects are stored in a HashMap in the ViewMediator. The key in the Map is the Set ID of the Dataset and the value the View Object. The map can then be used as a lookup table for finding out the relations among the View objects. Generally (in most cases) a View influences another View only if they operate on the same Set of data. The ViewMediator follows the well known Mediator Design Pattern, where several Objects can communicate over a common module. Therefore the usage of the Mediator Pattern eliminates the need of Objects to refer to each other directly which reduces the implementation effort. Also in case of a new View type it has to only registered in the ViewMediator without knowing an referring to the other Views. In the Mediator class special update methods are implemented, which are called when in a View some kind of event occurs. The View calls on of these update models (as the case may be) and the update method in the Mediator finds out all related Views and triggers the update. The update method in the view knows then how to react on that update. For example a Heatmap has to

visualize an selection completely different as an Dendrogram.

The ViewMediator reacts on two basic events.

- **selectionUpdate:** The selection update is triggered when the user selects a portion or a region of a View, which influences other dependent Views.
- **dataSetUpdate:** The dataSetUpdate is called when the Set of data changes in a View. For example when a clustering algorithm is performed on the Set of data which results in an update of all Views, which visualize these

3.1.4.1 Heatmap View

3.1.4.2 Dendrogram View

A Dendrogram is a tree diagram that is mostly used to show the result of clustering alogrithms.

3.1.4.3 Pathway View

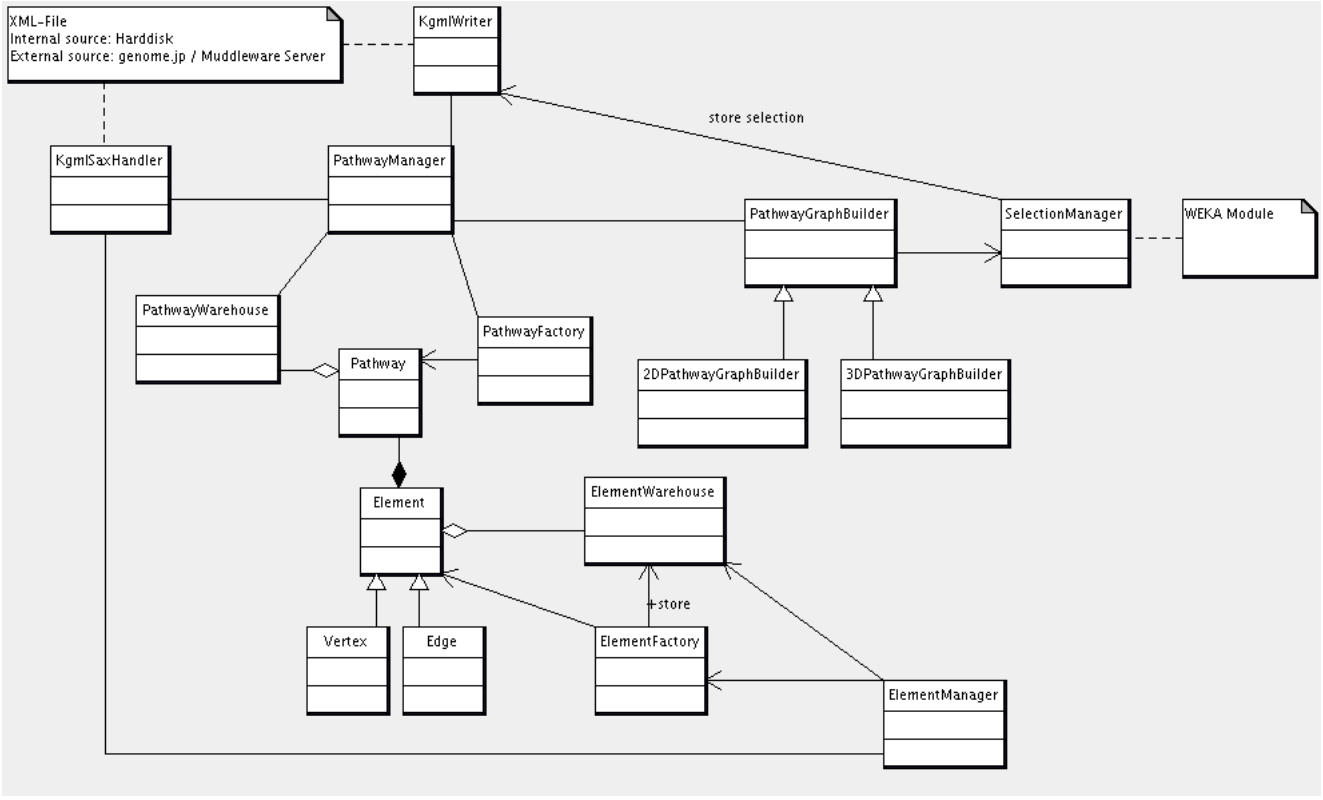

Figure 3.5: *Pathway Class Design*

3.1.4.4 Histogram View

3.1.4.5 Combination and interaction of Views

List of Figures

Bibliography

[JGG06] JGraph Ltd. (Editor): *Java Graph Visualization and Layout*. `http://www.jgraph.com`. Version: 2006

[JOG06] *JOGL - Java OpenGL*. `https://jogl.dev.java.net/`. Version: 2006

[SWT06] Eclipse (Editor): *SWT: The Standard Widget Toolkit*. `http://www.eclipse.org/swt/`. Version: 2006