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).

2.4 WEKA

> Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes ((WEK06)).

Chapter 3

Class Design

3.1 Module Overview

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.

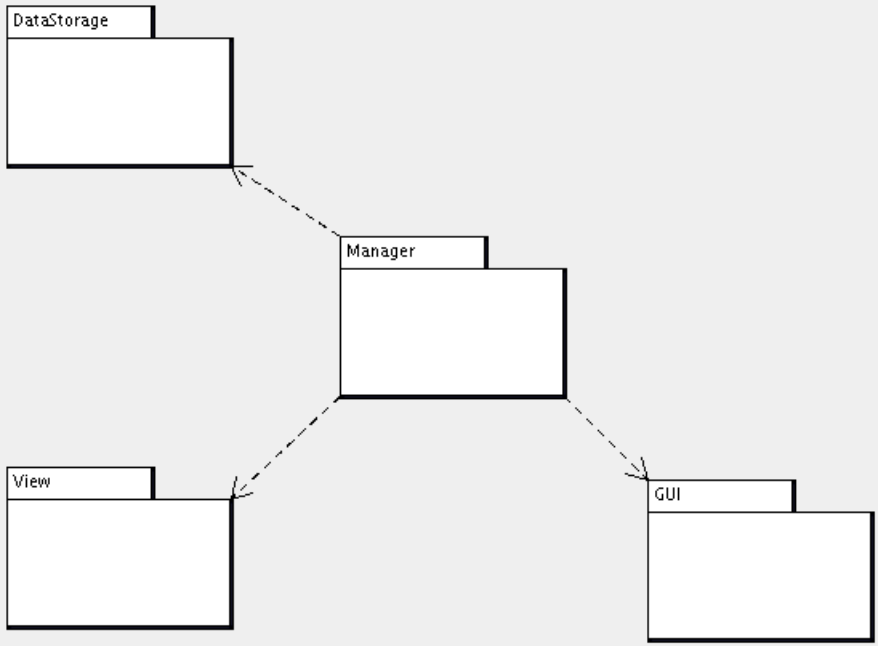

Figure 3.1: *Cerberus Modules*

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

3.2 Manager Module

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").

3.3 Data Module

3.3.1 Internal Data Management

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".

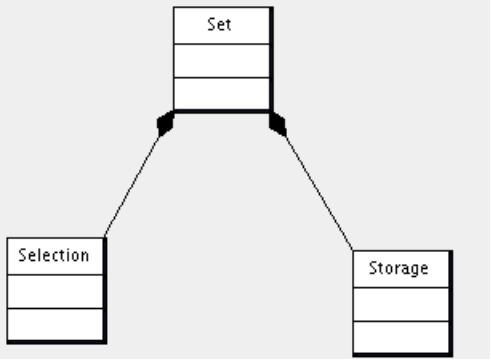

Figure 3.2: *Data Module Class Design*

3.3.2 External Data

3.3.2.1 Micro-Array Data file

- **Description:** One file stores one micro-array experiment containing all Genes/Proteins.
- **Extension:** *.grp
- **File size:** 8 MB
- **Dimension:** [38000 Genes] x [1 probe] x [10 values + (15 derived statistical values)]
- **Format:** ASCII, header, table of values

3.3.2.2 Clinical Excel files

- **Description:** Each row represents one patient; each column contains data on the patient.
- **Extension:** *.xls and *.csv
- **File size:** $>$ 1 MB
- **Dimension:** [40 medical parameters] x [2000 patients]
- **Format:** EXCEL-file, Header row, table of values

3.3.2.3 Pathway data file

- **Description:** Biological and medical pathways. For details see section 3.5.3.1 on page 10.
- **Extension:** *.xml
- **File size:** each $<$ 1 MB
- **Dimension:** [3000] x [directed cyclic graph]
- **Format:** XML-file

3.4 GUI Module

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.

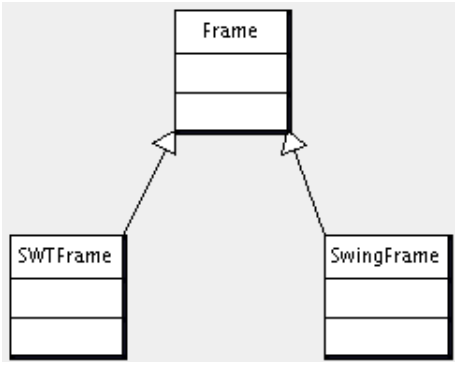

Figure 3.3: *GUI Module Class Design*

3.4.1 Alternative Application Styles

3.4.1.1 Multiple Document Interface (MDI)

The MDI approach is a controverial issue but without doubt it has it's advantages. An simple MDI example can be seen in figure 3.4. For our Cerberus Framework we can exclude the MDI GUI approach because of the usage of SWT as a GUI toolkit. With SWT it is not (out of the box) possible to create a clean MDI application.

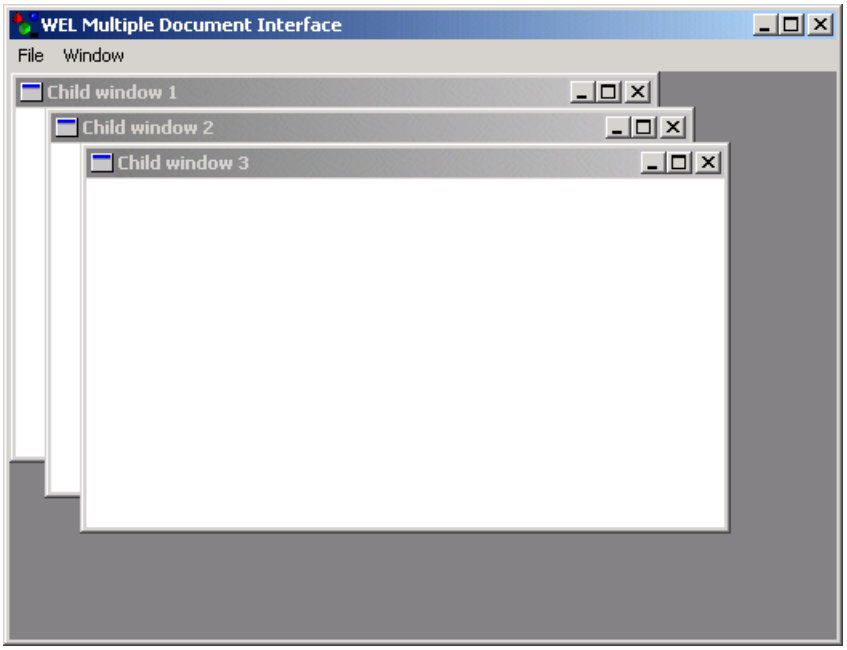

Figure 3.4: *MDI*

3.4.1.2 Tree Managed Views combined with Grid Layout

This layout consists of one overall window in which the widgets are placed. On the left side a tree appears where the views are listed. It is possible to arrange the views on a grid layout. These grid sheets can also be managed by tabs that are well known from the broweser (i.e. Tabbed Browsing Concept).

3.4.1.3 Free window layout

In a free window layout all windows are genereated from one application but the windows appear to the user as independent windows. The layout is known from the GNU GIMP project under Linux. A simple example of that GUI design can be seen in figure 3.5. We apply this layout for working on multiple desktops (in most cases a dual monitor system). We can visualize the overview of several depending views on one screen and a detailed specific view on the second screen.

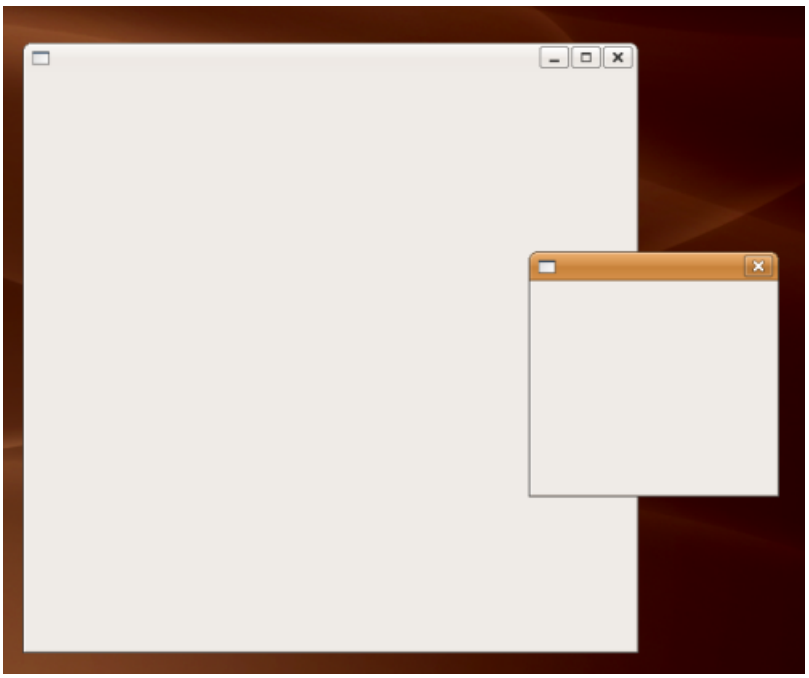

Figure 3.5: *Free Window Layout*

3.5 View Module

In figure 3.6 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).

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 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 a Dendrogram.

The ViewMediator reacts on two basic events.

- **selectionUpdate:** The selection update is triggered when the user selects a portion

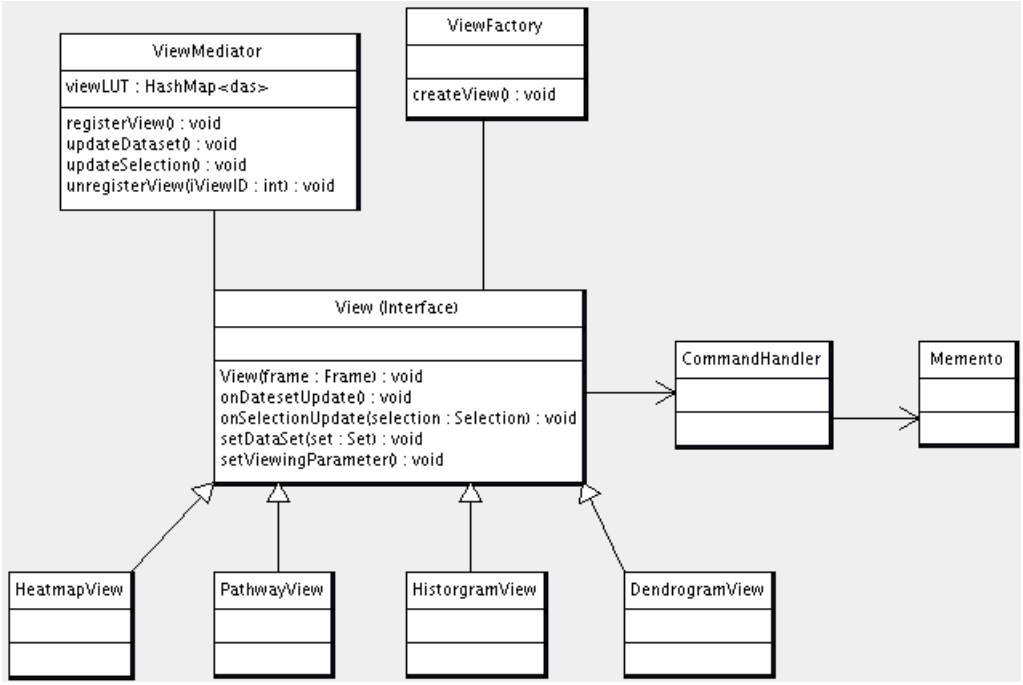

Figure 3.6: *View Module Class Design*

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

3.5.1 Heatmap View

3.5.2 Dendrogram View

Description

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

3.5.3 Pathway View

3.5.3.1 Description

Collection of Proteins, that interact with each other and thus occur co-regulated or counter-regulated in the micro-array data-sets. A pathway is a directed cyclic graph consisting of Proteins, which can be traversed in forward or backward order. One Protein may occur several times in different nodes. The edges of the graph can be tagged with additional information. For instance required or produced biochemical energy (ADP +Pi <==> ATP) or (NADP <==> NADPH). Some pathways are irreversible and thus only traverse

in one direction, like "Glukose-Spalltung = Glykolyse". Once this kind of pathway is triggered it is "executed". Other Pathways like "Umbau von Zucker" or the Fat-Cycle can be traversed back and forth.

3.5.3.2 Class Diagram

The UML class diagram is shown in figure 3.7. Basicall a pathway graph from the computer science point of view is a cyclic graph that consists of nodes and edges. Nodes and edges are inherited from the base class Element. Elements are created by the ElementFactory and are stored in the ElementManager (Note: The ElementFactory is integrated in the ElementManager and exists only as a logical entity in the design.) The pathway consists of elements and are stored in the PathwayManager. The ElementManager and the PathwayManager are implemented as a Singleton. The KgmlSaxHandler uses the Java SAX parser API for XML. Therefore this class implements the parsing of the XML input pathway file (for details see section 3.3.2.3 on page 7. The extracted data is feed into the PathwayManager and the ElementManager. When the pathways and the elements are created, the data is ready to be drawn in a graph using the JGraph library. This is implemented in the PathwayGraphBuilder classes.

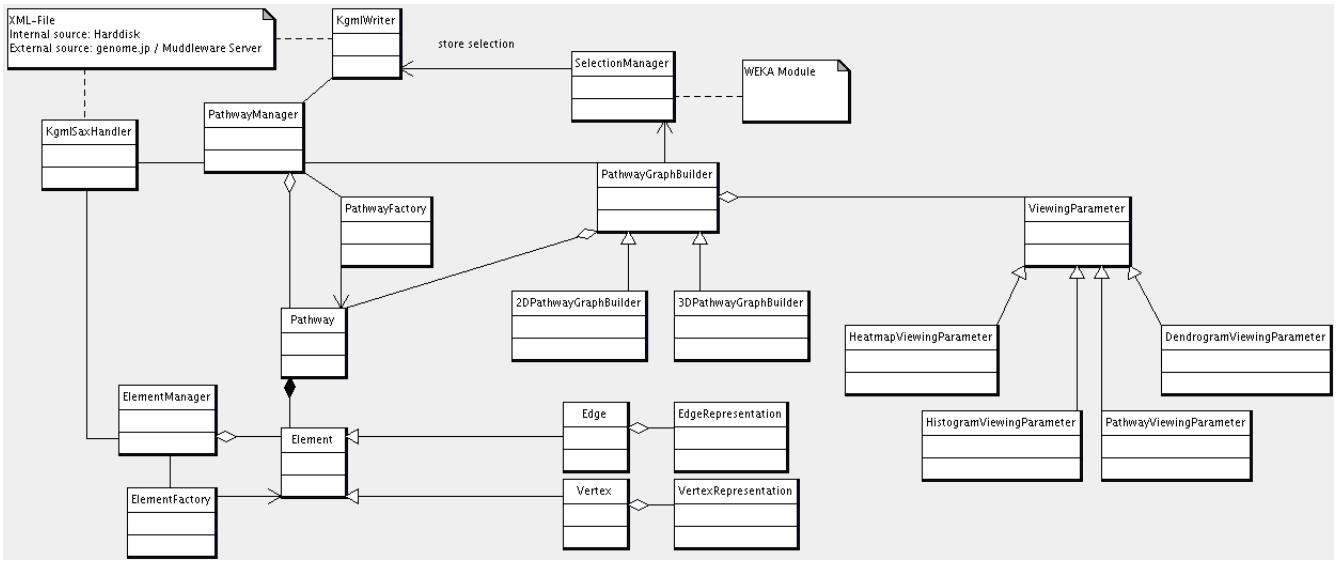

Figure 3.7: *Pathway Class Design*

3.5.4 Histogram View

3.5.5 Alternative visualization techniques

It seems that a 2.5D version of the heatmap has a lot of potential to improve the state of the art heatmap visualization technique. In this concecpt the 3rd dimension contains information about the quantity of a specific gene (like a historgram in the third dimension). For example when the clustering algorithmn merges some genes this information can be

visualized in an elegant way. Also in comination with the pathways this visualization method needs to be explored in the future.

3.5.6 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

[WEK06] University of Waikato (Editor): *Weka 3: Data Mining Software in Java*. `http://www.cs.waikato.ac.nz/ml/weka/`. Version: 2006