

A Versatile Scientific Programming Environment for Cardiovascular Data Analysis

A. Schlemmer

T. Hajek, U. Parlitz, S.E. Lehnart, N. Wessel, H. Malberg and S. Luther

Max Planck Research Group Biomedical Physics,
Max Planck Institute for Dynamics and Self-Organization, Göttingen

<http://www.bmp.ds.mpg.de/>
alexander.schlemmer@ds.mpg.de

April 13, 2010



Aims

- ▶ Platform for cardiovascular data analysis in multi-disciplinary environments

Aims

- ▶ Platform for cardiovascular data analysis in multi-disciplinary environments
- ▶ Intuitive graphical user interface

Aims

- ▶ Platform for cardiovascular data analysis in multi-disciplinary environments
- ▶ Intuitive graphical user interface
- ▶ High level of interactivity

Aims

- ▶ Platform for cardiovascular data analysis in multi-disciplinary environments
- ▶ Intuitive graphical user interface
- ▶ High level of interactivity
- ▶ Performant application of algorithms

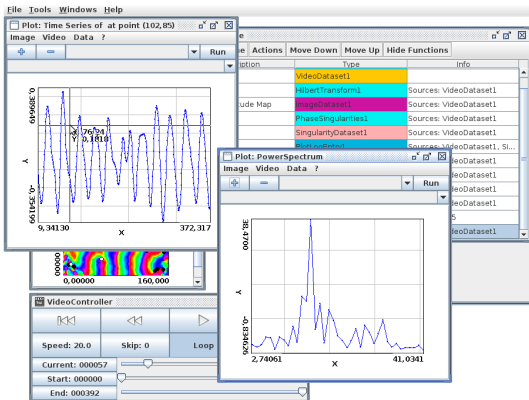
Aims

- ▶ Platform for cardiovascular data analysis in multi-disciplinary environments
- ▶ Intuitive graphical user interface
- ▶ High level of interactivity
- ▶ Performant application of algorithms
- ▶ Efficient and standardized development of algorithms

Aims

- ▶ Platform for cardiovascular data analysis in multi-disciplinary environments
- ▶ Intuitive graphical user interface
- ▶ High level of interactivity
- ▶ Performant application of algorithms
- ▶ Efficient and standardized development of algorithms
- ▶ Reproducibility of results

OpenAnalyser



OpenAnalyser is an open source-licensed (GNU GPL) Java application for scientific data analysis with a graphical user interface (GUI) and integrated plugin system.

The Java Platform

- ▶ Platform independence

The Java Platform

- ▶ Platform independence
- ▶ Powerful API including GUI-toolkit

The Java Platform

- ▶ Platform independence
- ▶ Powerful API including GUI-toolkit
- ▶ Prevalence

The Java Platform

- ▶ Platform independence
- ▶ Powerful API including GUI-toolkit
- ▶ Prevalence
- ▶ Robustness

The Java Platform

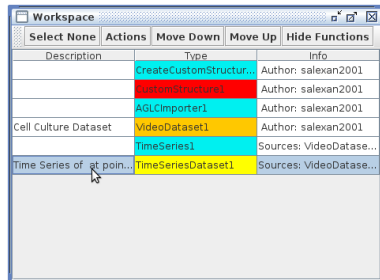
- ▶ Platform independence
- ▶ Powerful API including GUI-toolkit
- ▶ Prevalence
- ▶ Robustness
- ▶ Modularity and dynamic features

The Java Platform

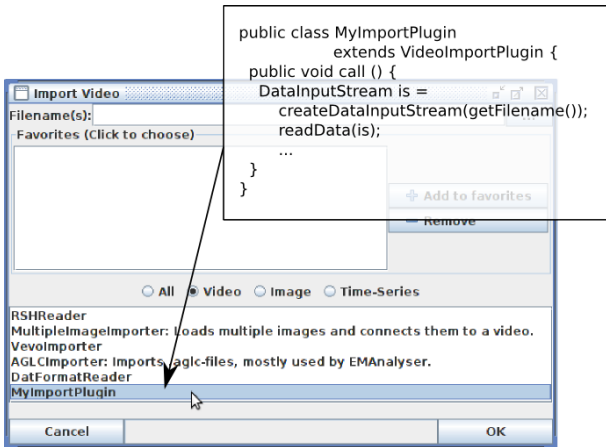
- ▶ Platform independence
- ▶ Powerful API including GUI-toolkit
- ▶ Prevalence
- ▶ Robustness
- ▶ Modularity and dynamic features
- ▶ Integration of other programming languages, e.g. Python, C++ or FORTRAN

The Workspace

- ▶ Overview over data, analysis steps and results
- ▶ Visualization and analysis algorithms applicable via context menus

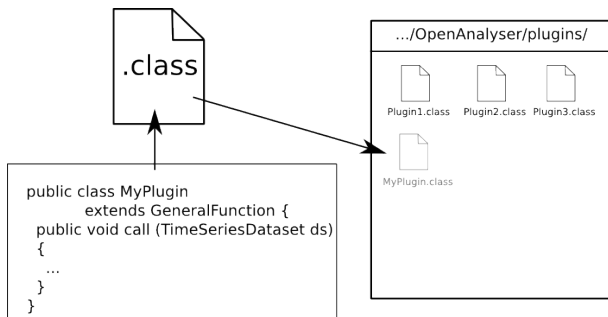


Data Import



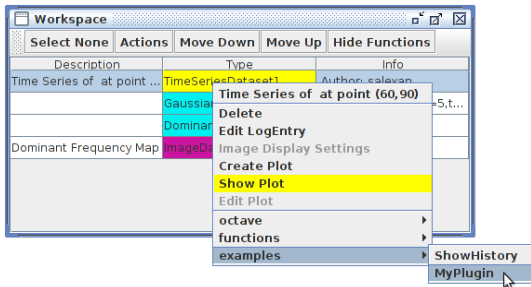
- ▶ Import plugins facilitate the integration of data formats.

Plugin-System



- ▶ Simple, yet powerful and flexible plugin-system for development of algorithms for OpenAnalyser
- ▶ Permits users to add their own algorithms

Integration of Algorithms



- ▶ New algorithms are integrated into the GUI and can even be reloaded at runtime

Parameters

The image shows a tree view of a plugin named 'MyPlugin'. The tree contains several nodes: 'saveData = false', 'threshold = 1.7', 'parameters = (X1=0, Y1=0, Z1=0, T1=0, X2=-1, Y2=-1, Z2=1, T2=-1)', 'timestamp = Sa Apr 10 18:54:27 MESZ 2010', 'author = salexan', and 'description ='. A box on the left contains two annotations: '@ParameterDescription public boolean saveData = false;' with an arrow pointing to the 'saveData' node, and '@ParameterDescription public double threshold = 1.7;' with an arrow pointing to the 'threshold' node. To the right, a dialog window titled 'Eingabe' (Input) is shown. It has a green question mark icon and the text 'Please enter new double:'. Below this text is a text input field containing the value '1.7'. At the bottom of the dialog are two buttons: 'OK' and 'Abbrechen' (Cancel).

- ▶ Simple declaration of parameters needed by plugins
- ▶ Automatic integration of parameters into user-interface

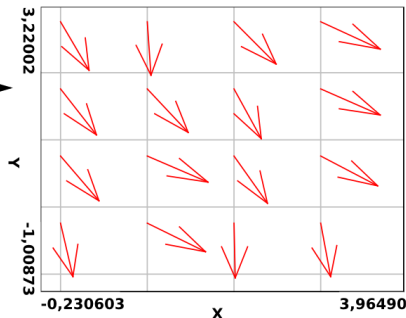
Visualization-System

```
public class VectorVisualizer
    extends GeneralVisualizer {

    private VectorDataset dataset;

    public void paint(Graphics2D g) {
        for (int i = 0; i < dataset.vectors.size(); i++) {
            Arrow vector = dataset.vectors.get(i);
            DoubleXY pos = vector.pos;

            ...
        }
    }
}
```



- ▶ OpenAnalyser offers basic visualization functionality (e.g. xy-plots, video display, ...)
- ▶ Support for custom visualization methods (e.g. for custom datatypes) via visualization plugins

Workspace storage and logging mechanism

Workspace		
Select	None	Actions
	Move Down	Move Up
	Hide	Functions
Description	Type	Info
	AGLCImporter1	Author: salexan
	VideoDataset1	Author: salexan
	TimeSeries1	Sources: VideoDataset...
Time Series of at point ...	TimeSeriesDataset1	Sources: VideoDataset...
	GaussianSmoothing1	smoothImages, size=5, t...
	DominantFrequencyMap1	Sources: VideoDataset...
Dominant Frequency Map	ImageDataset1	Sources: VideoDataset...

AGLCImporter1_objectData.xml	948 Bytes
DominantFrequencyMap1_objectData.xml	985 Bytes
GaussianSmoothing1_objectData.xml	1,3 KB
ImageDataset1_colorPalette.dat	3,2 MB
ImageDataset1_image.dat	150,0 KB
ImageDataset1_objectData.xml	2,7 KB
TimeSeries1_objectData.xml	909 Bytes

testworkspace.oaw

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--@ version="1.0.0.0" class="java.beans.XMLDecoder"-->
3 <object class="java.util.HashMap">
4 <void method="put">
5 <string>timestamp</string>
6 <long>1278834939758</long>
7 </void>
8 <void method="put">
9 <string>temporalSize</string>
10 <int>5</int>
11 </void>
12 <void method="put">
13 <string>author</string>
14 <string>salexan</string>
15 </void>
16 <void method="put">
17 <string>description</string>
18 <string></string>
19 </void>
20 <void method="put">
21 <string>std</string>
22 <double>1.0</double>
23 </void>
```

- ▶ Workspace objects are mapped to xml- and binary-data

Workspace storage and logging mechanism

Workspace		
Select None	Actions	Move Down Move Up Hide Functions
Description	Type	info
	AGLCImporter1	Author: salexan
	VideoDataset1	Author: salexan
	TimeSeries1	Sources: VideoDataset...
Time Series of at point ...	TimeSeriesDataset1	Sources: VideoDataset...
	GaussianSmoothing1	smoothImages, size=5,t...
	DominantFrequencyMap1	Sources: VideoDataset...
Dominant Frequency Map	ImageDataset1	Sources: VideoDataset...

AGLCImporter1_objectData.xml	948 Bytes
DominantFrequencyMap1_objectData.xml	985 Bytes
GaussianSmoothing1_objectData.xml	1,3 KB
ImageDataset1_colorPalette.dat	3,2 MB
ImageDataset1_image.dat	150,0 KB
ImageDataset1_objectData.xml	2,7 KB
TimeSeries1_objectData.xml	909 Bytes

zip
testworkspace.oaw

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--@ version="1.0.0.0" class="java.beans.XMLDecoder"-->
3 <object class="java.util.HashMap">
4 <void method="put">
5 <string>timestamp</string>
6 <long>1278834939758</long>
7 </void>
8 <void method="put">
9 <string>temporalSize</string>
10 <int>5</int>
11 </void>
12 <void method="put">
13 <string>author</string>
14 <string>salexan</string>
15 </void>
16 <void method="put">
17 <string>description</string>
18 <string></string>
19 </void>
20 <void method="put">
21 <string>std</string>
22 <double>1.0</double>
23 </void>
```

- ▶ Workspace objects are mapped to xml- and binary-data
- ▶ Storage in container-files or databases possible

Workspace storage and logging mechanism

Workspace		
Select	None	Actions
	Move Down	Move Up
	Hide Functions	
Description	Type	Info
	AGLCImporter1	Author: salexan
	VideoDataset1	Author: salexan
	TimeSeries1	Sources: VideoDataset...
Time Series of at point ...	TimeSeriesDataset1	Sources: VideoDataset...
	GaussianSmoothing1	smoothImages, size=5,t...
	DominantFrequencyMap1	Sources: VideoDataset...
Dominant Frequency Map	ImageDataset1	Sources: VideoDataset...

AGLCImporter1_objectData.xml	948 Bytes
DominantFrequencyMap1_objectData.xml	985 Bytes
GaussianSmoothing1_objectData.xml	1,3 KB
ImageDataset1_colorPalette.dat	3,2 MB
ImageDataset1_image.dat	150,0 KB
ImageDataset1_objectData.xml	2,7 KB
TimeSeries1_objectData.xml	909 Bytes

testworkspace.oaw

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--@ version="1.0.0.0" class="java.beans.XMLDecoder"-->
3 <object class="java.util.HashMap">
4 <void method="put">
5 <string>timestamp</string>
6 <long>1278834939758</long>
7 </void>
8 <void method="put">
9 <string>temporalSize</string>
10 <int>5</int>
11 </void>
12 <void method="put">
13 <string>author</string>
14 <string>salexan</string>
15 </void>
16 <void method="put">
17 <string>description</string>
18 <string></string>
19 </void>
20 <void method="put">
21 <string>std</string>
22 <double>1.0</double>
23 </object>
```

- ▶ Workspace objects are mapped to xml- and binary-data
- ▶ Storage in container-files or databases possible
- ▶ Each step in the analysis procedure can be reproduced

Conclusion

- ▶ OpenAnalyser is designed for collaborative data analysis in large-scale, cross-disciplinary research networks
- ▶ OpenAnalyser combines an intuitive GUI with a powerful programming framework for algorithm development and application
- ▶ Users can integrate existing code easily via versatile plugin system
- ▶ OpenAnalyser is work in progress. Your suggestions, comments and contributions are welcome!

Thank you!

Thank you for your attention!

Download and more Information about OpenAnalyser:

<http://www.bmp.ds.mpg.de/software.html>

The project is supported by:

- ▶ BMBF FKZ 01EZ0905/6
- ▶ European Community FP7 EUTrigTreat (HEALTH-F2-2009-241526)