Implementation of Interactive Statistical Graphics Software
Platform Independence

- We got tired of “Your software is great, but we don’t use Macs!”
- Academic software environments are heterogeneous
  ⇒ so are the software projects.
- Potential users of academic software are 95% Windows
  ⇒ can not be ignored!
- Cross platform development (i.e. maintaining multiple code bases at least in parts) is far too expensive!
- R put up a standard in being available for “any” platform
  ⇒ that’s what people more and more expect
- Still only one choice ⇒ JAVA
JAVA

- Really available on almost all platforms
  (Even Sun-independent implementations on LINUX)

- “write once – run everywhere” still dominates “write once – debug everywhere”

- Knows how to deal with graphics
  - AWT
  - JFC
  - 2D
  - PS printing

- Package/Library system

- Developing (e.g. 1.5 implements native hardware acceleration)
Decomposition of Graphs

- In an object-oriented programming environment/language, an effective definition of the graphical objects is key.

- Typical Objects
  - plot primitives
    - points
    - lines
    - boxes
  - axes
  - plot specifics

- Example: Histogram
  - primitives: boxes
  - axes
    - x: range
    - y: count or probability
  - plot specifics
    - origin and width control
Layers

- 4 layers can be defined to group the different plot components
  - Interaction layer
  - Selection layer
  - Object layer
  - Background layer

- The layers are defined according to their update frequencies from least frequent update to most frequent update, i.e.
  interaction ➤ selection/highlighting ➤ objects ➤ axes (background)

- Very important to speed up drawing times!
Object Hierarchy: Data

- Data are abstracted in data set objects.
- If the data source is a flat file, the actual data resides in double arrays (one per variable).
- For data in databases, only the connection information is stored.
- For convenience a Table object exists ‘above’ the data set class.

All plot for categorical data access the data only through the Table class!
ASCII Files

- (At least for statisticians) most data sits in flat files

- Mondrian accepts tab separated ASCII files with headers
  - Export as `.tsv’ in Excel
  - in R use: `write.table(mydata, "myfile", quote = F, sep = "\t ", row.names = F)`

- ASCII files can include an optional set of polygons for map plotting
  For each area, there must be a description via a closed polygon of x- and y-coordinates, and a matching link in the data matrix.
  (A “standard” for map files would be nice …)

- Missing values are **not** supported.
Database Connections

• Connecting directly to databases has many advantages
  + no data handling within Mondrian
  + works very efficient on categorical and summarized data
  + scales up to “any” size of data set
  + natural management of missing values (NULLs)
  + selection and linking translates directly in SQL clauses
  + data is always up-to-date
  + far wider range of data problems

• Disadvantages are
  – access of single records can be comparatively expensive
  – update strategies for DB changes must be included

• Mondrian uses elementary JDBC functionality
  … still at an experimental stage – proof of concept
Object Hierarchy: Graphs

- Mondrian graphs use JFC and Java2D
- Base class is the SelectableFrame which handles all
  - selections
  - zooming
  - event handling
- All plots are derived from the SelectableFrame
- Object hierarchy was deliberately choose to be very flat, to speed up computation
Event Handling

- Mondrian implements two custom events
  - `dataChanged`
  - `selectionChanged`

- `dataChanged` is fired, whenever a plot changed data, e.g. reorder of categories or transformations

- `selectionChanged` is fired whenever the selection changed

- Both event types are distributed by a parent event listener!
Interface Conventions

• There is a tight and consistent mapping of interactions
  
  - Selections
    click and drag ➞ create a selection rectangle / brush
    click on selection rectangle handle ➞ resize this selection
    popup-trigger on selection rectangles ➞ alter this selection
  
  - Interrogation
    popup-trigger on objects ➞ interrogation
    shift click in selection rectangle ➞ interrogation
  
  - Alterations
    meta-click and drag ➞ zoom in/out
    (middle click on Windows)
    popup-trigger on background ➞ get plot options
    alt-click and drag ➞ reorder objects
    page-up /-down ➞ cycle through views
    arrows up/down and left/right ➞ increase/decrease plot parameters
Communication with R

• Big problem in statistical software development:
  Methods are re-implemented over and over again whenever systems, packages and/or programming languages change

• Examples of custom development
  – Adding a linear regression to a scatterplot ✔
  – Adding a lowess smoother to a scatterplot ✘

• Better solution: Re-use of existing and tested components

• Right now R is the cheapest and best available source of statistical routines, written in R-code, C and Fortran

• Problem: R has no decent interface other than the REPL-loop
Talking to R via Rserve

- Rserve (developed by Simon Urbanek) runs R as a background process and communicates via socket connections with R
- Many potential clients can talk to Rserve – for Mondrian we need the JAVA-client.
- Example: Adding a density function to a histogram

```java
try {
    RConnection c = new RConnection();
    double[] xVal = data.getRawNumbers(tablep.initialVars[0]);
    c.assign("x", xVal);

    RList l = c.eval("density(x, bw="+bWidth+", from="+xMin+", to="+xMax+")").asList();
    double[] dx = (double[]) l.at("x").getContent();
    double[] dy = (double[]) l.at("y").getContent();

    if( displayMode.equals("Histogram") && !CDPlot )
        for( int f=0; f<dx.length-1; f++ ) {
            bg.drawLine( (int)userToWorldX( dx[f] ), (int)userToWorldY( dy[f] ),
                          (int)userToWorldX( dx[f+1] ), (int)userToWorldY( dy[f+1] ));
        }
    ...
    c.close();
} catch(RSrvException rse) {System.out.println("Rserve exception: "+rse.getMessage());}
```
How to use Mondrian “from Outside”

• Mondrian was never designed to offer single classes to other applications
  (almost all classes would need to be “exported” in order to use a single plot)

• Nonetheless, Mondrian can be invoked from other applications

• If Mondrian runs within the same VM as the caller application, functions like
  – Get selected points
  – Set selection
  – Data changed

  can be used.
What is still missing?

• Clean up, clean up, clean up, clean up, clean up, …

• Better data reader, especially for handling NAs

• Completion of the DB interface for all data access functions

• Saving of preferences

• Methods to communicate with other JAVA apps

• More enhanced graphics …

• …
The “Skeletons in the Closet”

- Mouse-Event and Mouse-Modifier model is clumsy …
  (... and sometimes source of errors)

- General event model is based on JAVA 1.1, should be updated

- Should make use of Java2D throughout the application

- General hot selection concept does not really fit into Mondrian

- Handling of missing values would be a pain

- …
Wrap Up & Outlook

• Implementation of interactive software is relatively expensive …

• … but, once you know how to do it right, it is far from being rocket science.

• Important
  “Don’t think of implementing a tool, but of implementing properties and strategies!”
  (Comp. scientists often think in terms of a tool not of the problem)

• After ~5 years of development of Mondrian, many concepts could be clarified – finding the “right” interface is always a challenge

• iPlots for R will bring ISG to the “masses”, once it is completed.