Reflexion on Software Quality and Maintenance

Alexandre Bergel
abergel@dcc.uchile.cl
13/04/2010

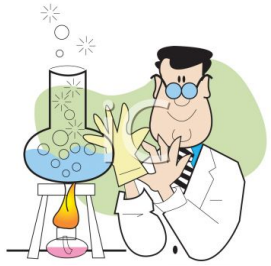
Goal of this lecture

Understanding some of the *design rules* in object-oriented systems

See *practical problems* of class inheritance

See few research results in the field of *software visualization*

the research icon



The *research* icon indicates topics that are not mandatory for the controls and the exam

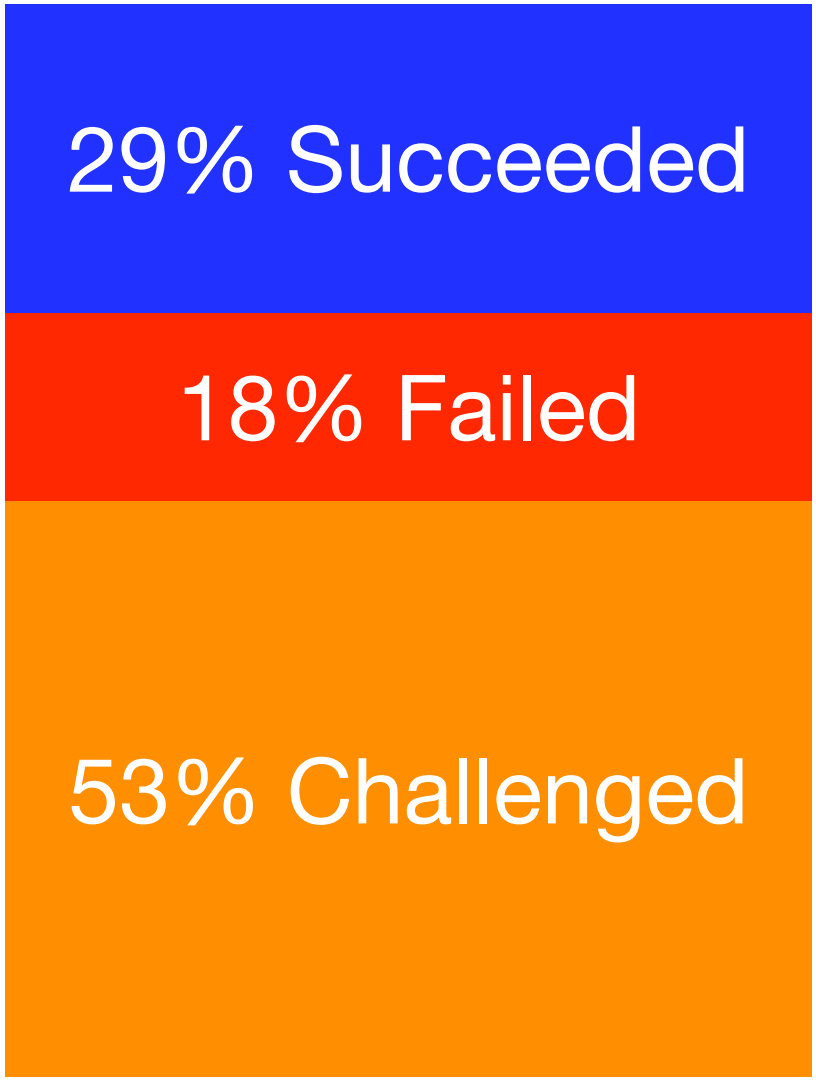
Roadmap

1. Real life maintenance problems
2. Software Quality
3. Example of code quality
4. Software visualization help taming software complexity 

Roadmap

1. **Real life maintenance problems**
2. Software Quality
3. Example of code quality
4. Software visualization help taming software complexity 

Software is complex

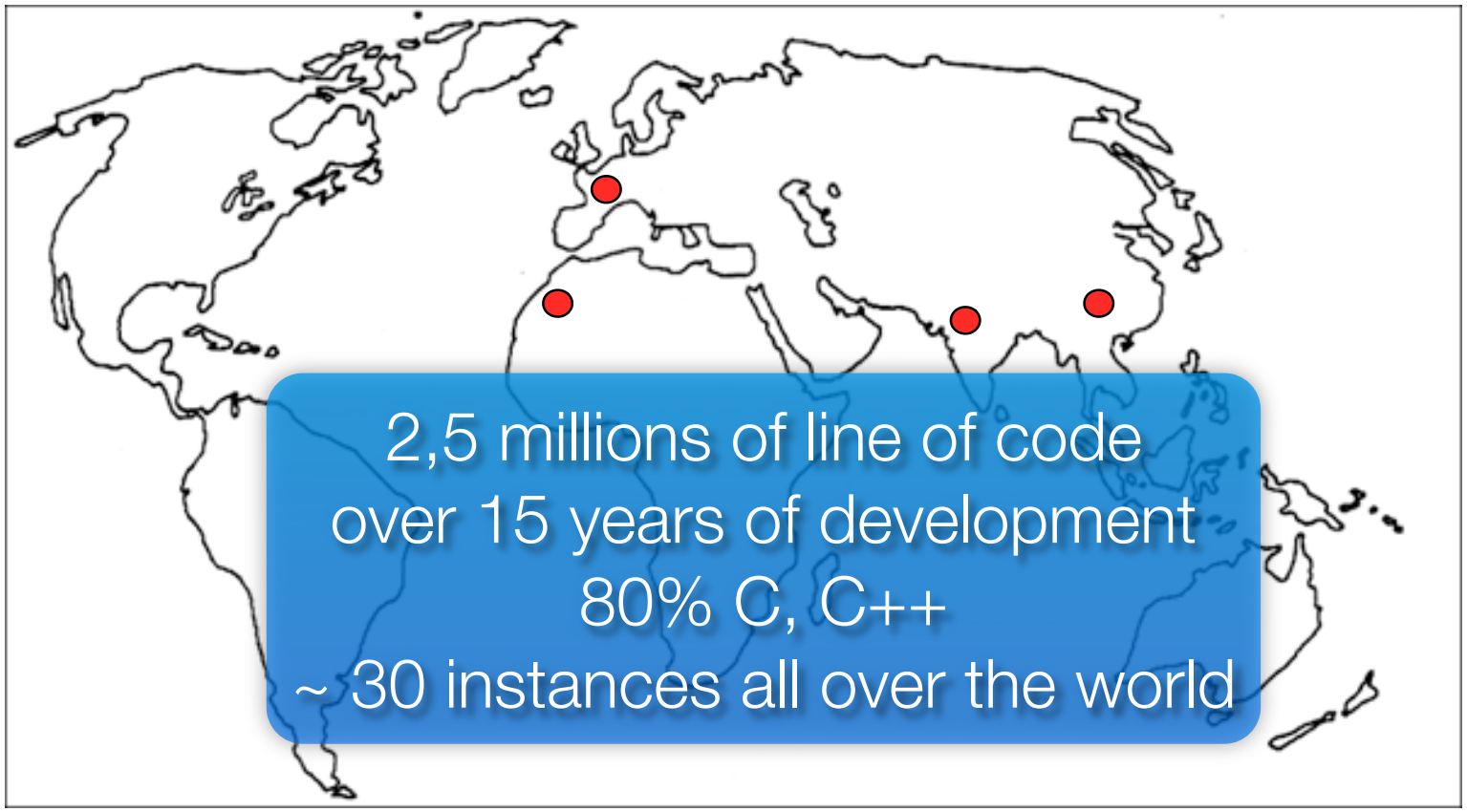


The Standish Group, 2004

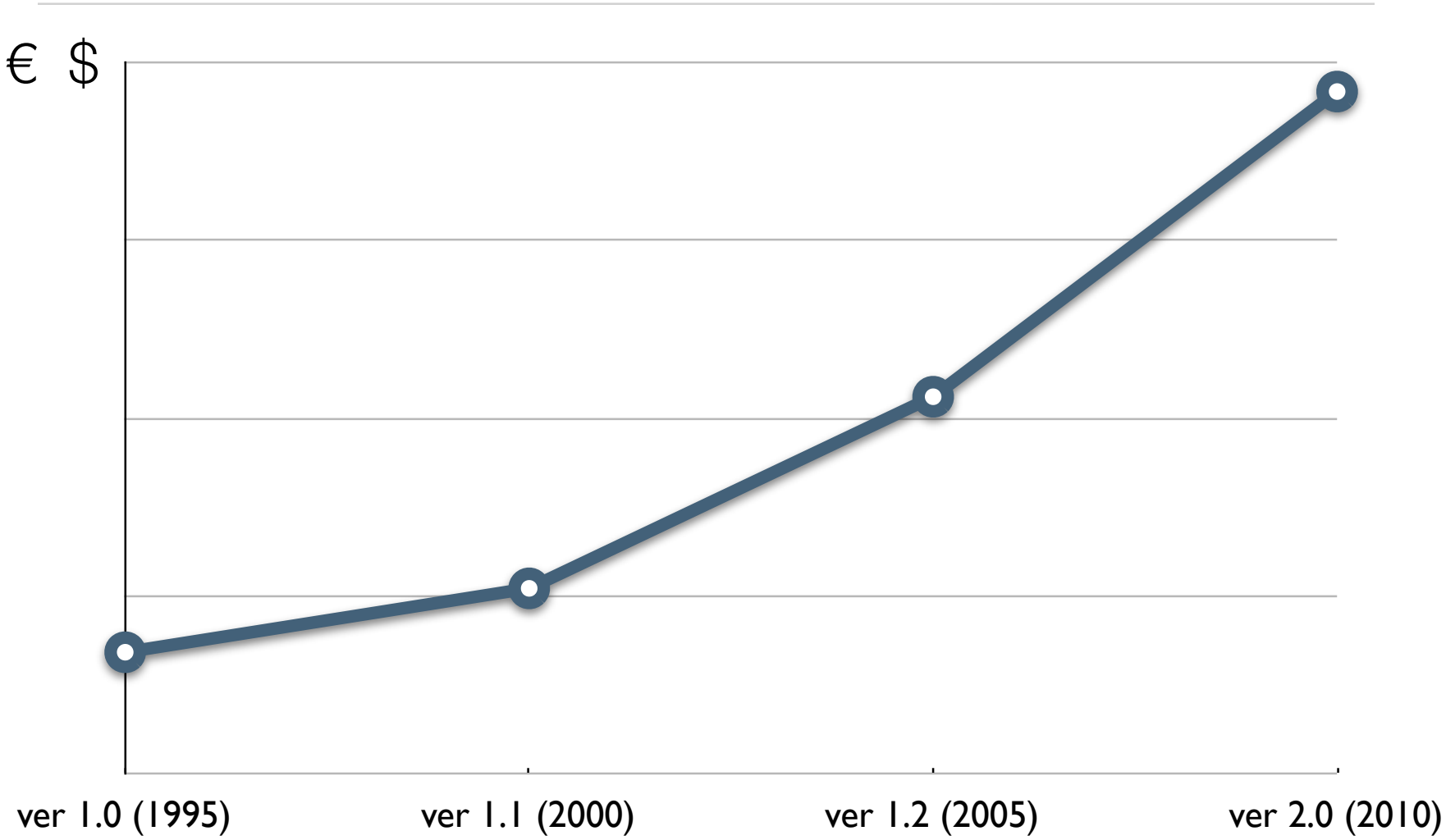
Let’s study some real world examples

Construction sites for an European truck maker

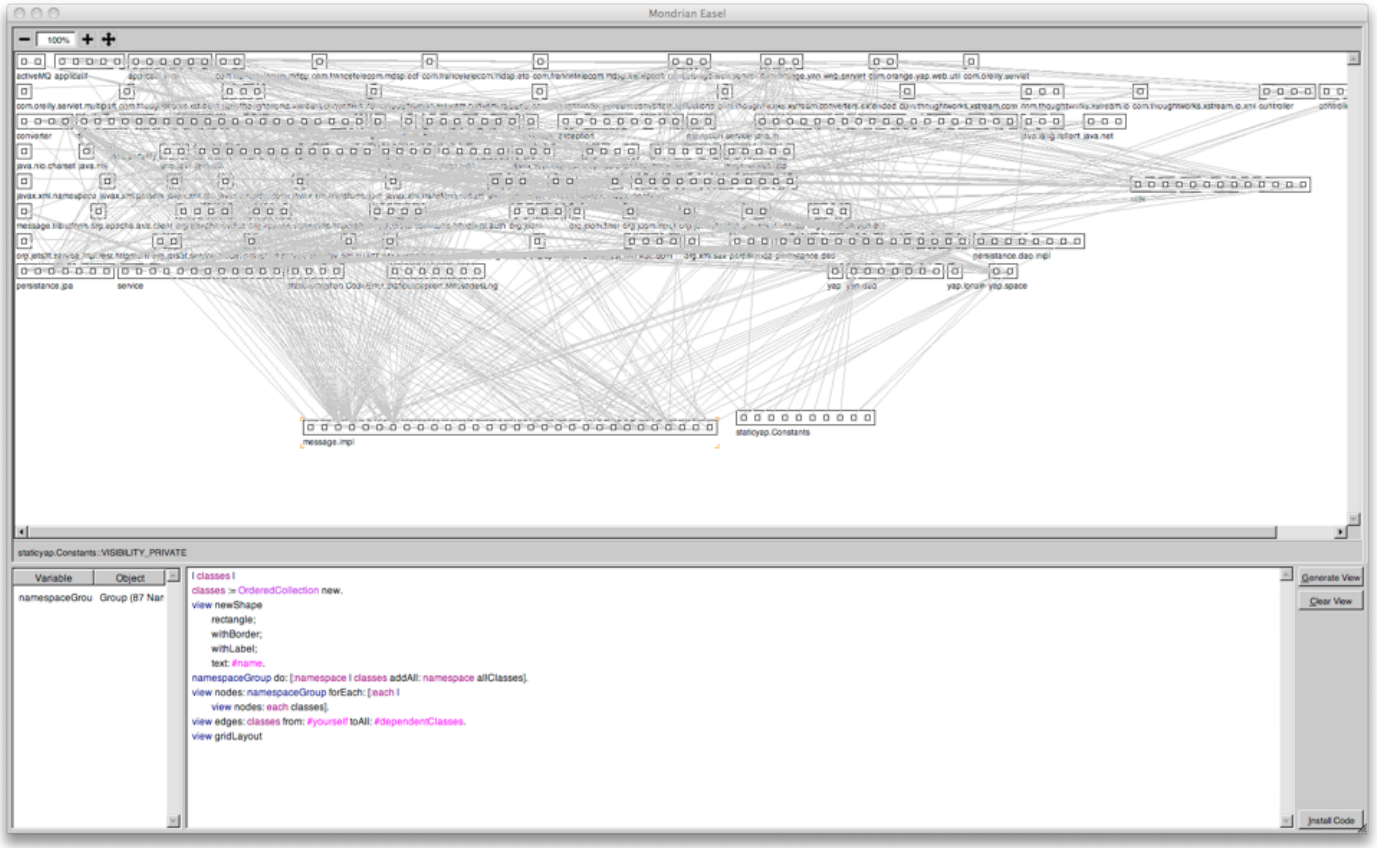
Construction sites for an European truck maker



Cost of feature addition



Large software in a French telecom company

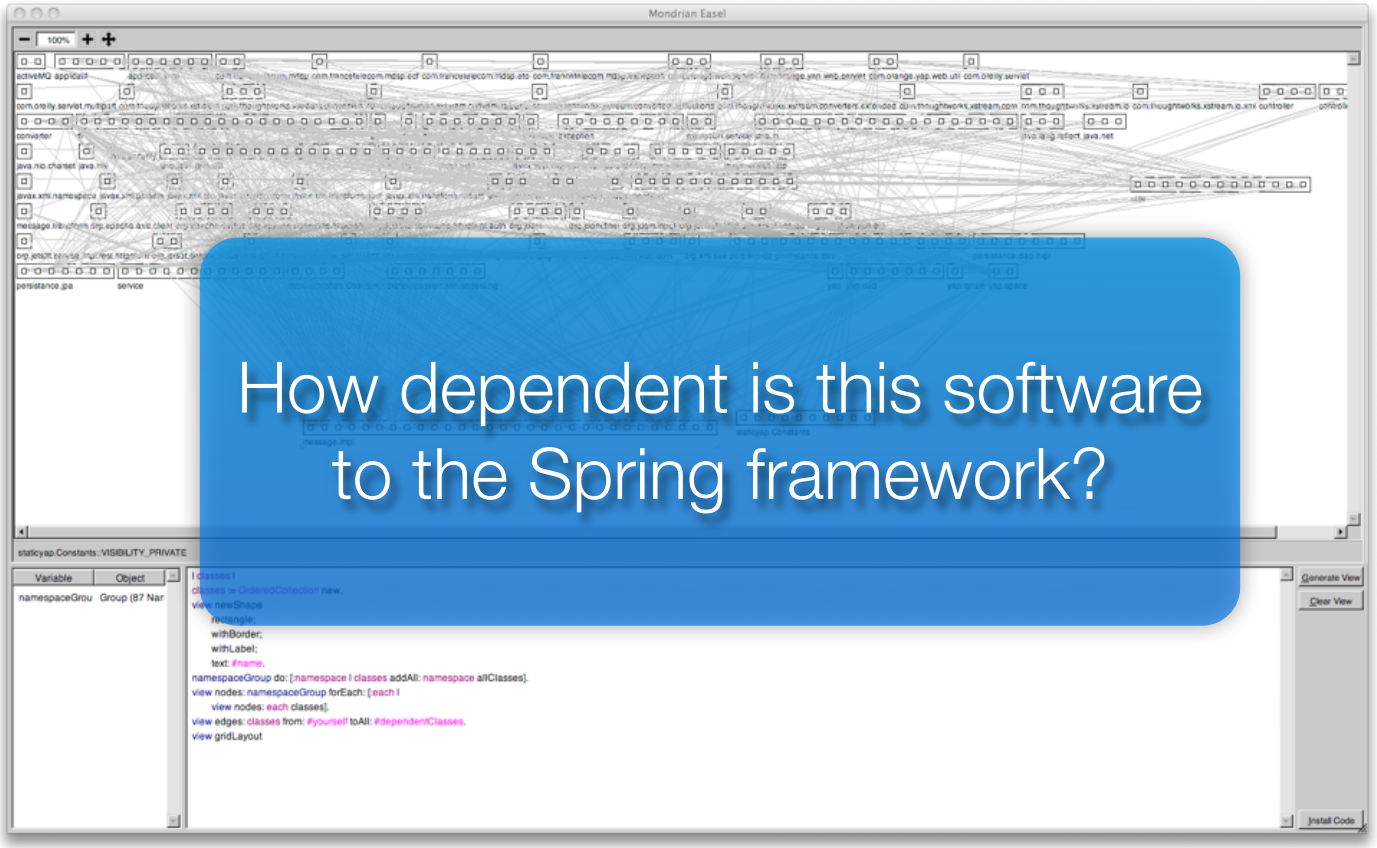


~100 packages

~ 500 classes

Paris, 2008

Large software in a French telecom company



~100 packages

~ 500 classes

Paris, 2008

Typical large scale long living systems

- Large
  - thousands of classes
  - hundreds of packages
- Undocumented - knowledge loss
- Lack of structure overview (layers, cycles, core)
- Possibly written in ADA or Cobol
- Multi developers
- Multi years development

Roadmap

1. Real life maintenance problems
2. **Software Quality**
3. Example of code quality
4. Software visualization help taming software complexity 

Quality plays a major rôle in maintaining software

But quality is vague...

Extendibility

Efficiency

Portability

Robustness

Correctness

Timeliness

Reusability

Compatibility

Ease of use

Roadmap

1. Real life maintenance problems
2. Software Quality
3. **Example of code quality**
4. Software visualization help taming software complexity 

```c
/* Hello World program */

#include<stdio.h>

int main()
{
  printf("Hello World");

}
```

no return statement in the function

```
/* Hello World program */

#include<stdio.h>

int main()
{
  printf("Hello World");

}
```

```
/* Hello World program */

#include<stdio.h>

int main()
{
    printf("Hello World");

}
```

no return statement in the function

9 lines of code
7 seconds

In Mozilla:

dom/base/nsDOMWindowUtils.cpp

```
/* -*- Mode: C++; tab-width: 2; indent-
tabs-mode: nil; c-basic-offset: 2 -*- */
/* ***** BEGIN LICENSE BLOCK *****
 * Version: MPL 1.1/GPL 2.0/LGPL 2.1
...
#include "nsIDOMHTMLCanvasElement.h"
#include "nsICanvasElement.h"
#include "gfxContext.h"
#include "gfxImageSurface.h"
...
```

belong to the core

In Mozilla:

dom/base/nsDOMWindowUtils.cpp

```
/* -*- Mode: C++; tab-width: 2; indent-
tabs-mode: nil; c-basic-offset: 2 -*- */
/* ***** BEGIN LICENSE BLOCK *****
 * Version: MPL 1.1/GPL 2.0/LGPL 2.1
...
#include "nsIDOMHTMLCanvasElement.h"
#include "nsICanvasElement.h"
#include "gfxContext.h"
#include "gfxImageSurface.h"
...
```

belong to gfx package

belong to the core

In Mozilla:

dom/base/nsDOMWindowUtils.cpp

```
/* -*- Mode: C++; tab-width: 2; indent-
tabs-mode: nil; c-basic-offset: 2 -*- */
/* ***** BEGIN LICENSE BLOCK *****
 * Version: MPL 1.1/GPL 2.1
...
#include "nsIDOMHTMLCanvasElement.h"
#include "nsICanvasElement.h"
#include "gfxContext.h"
#include "gfxImageSurface.h"
...
```

695 lines of code
3 minutes

belong to gfx package

In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {
  String s;
  if (b instanceof TitledBorder) {
       return ((TitledBorder) b).getTitle();
  } else if (b instanceof CompoundBorder) {
       s = getBorderTitle(((CompoundBorder)
                          b).getInsideBorder());
  if (s == null) {
    s = getBorderTitle(((CompoundBorder)
                          b).getOutsideBorder());
  }
  return s;
   } else {
      return null;
   }
...
```

In Swing:

JComponent.java

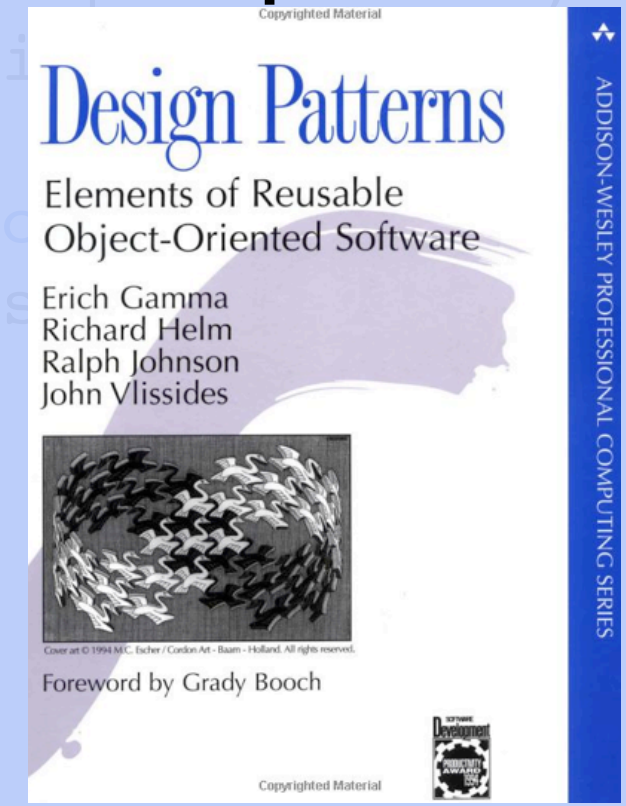
```
protected String getBorderTitle(Border b) {
   String s;
   if (b instanceof TitledBorder) {
         return ((TitledBorder) b).getTitle();
   } else if (b instanceof CompoundBorder) {
         s = getBorderTitle(((CompoundBorder)
                        b).getInsideBorder());
   if (s == null) {
     s = getBorderTitle(((CompoundBorder)
                        b).getOutsideBorder());
   }
   return s;
    } else {
       return null;
    }
...
```

In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {
   String s;
   if (b instanceof TitledBorder) {
         return ((TitledBorder) b).getTitle();
   } else if (b instanceof C
         s = getBorderTitle((
                        b).getI
   if (s == null) {
     s = getBorderTitle(((Co
                        b).getO
   }
   return s;
    } else {
       return null;
    }
...
```

Need Double Dispatch



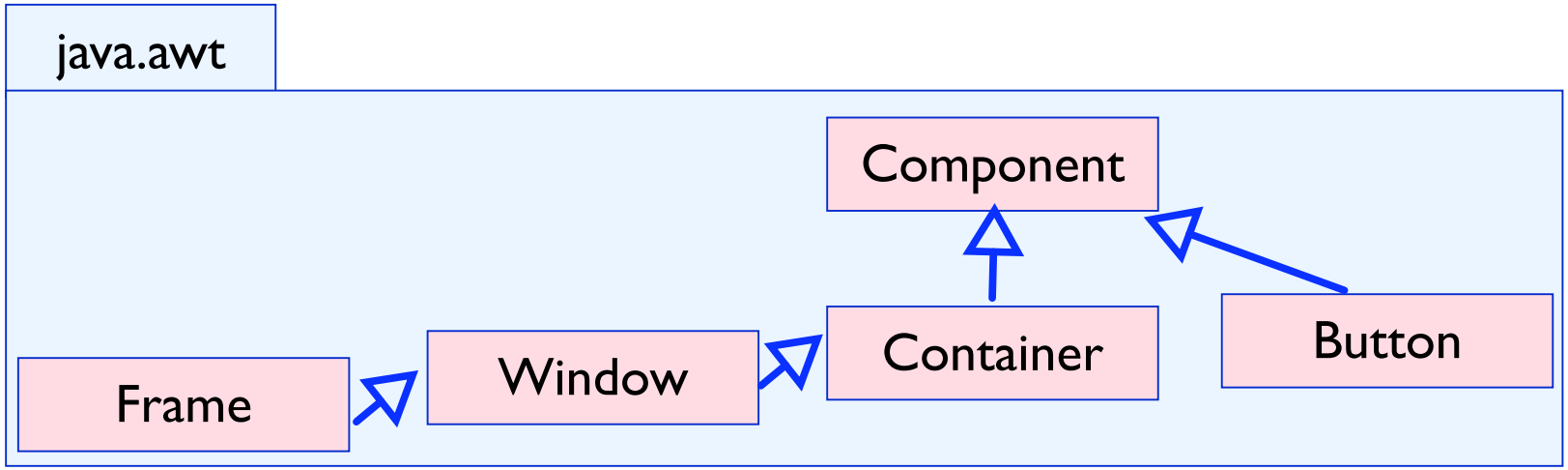
In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {
   String s;
   if (b instanceof TitledBorder) {
         return ((TitledBorder) b).getTitle();
   } else if (b instanceof CompoundBorder) {
         s = getBorderTitle(((CompoundBorder)
                  b).getInsideBorder());
   if (s == null) {
     s = getBorderTitle(((CompoundBorder)
                  b).getOutsideBorder());
   }
   return s;
    } else {
       return null;
    }
...
```

5471 lines of code
12 minutes

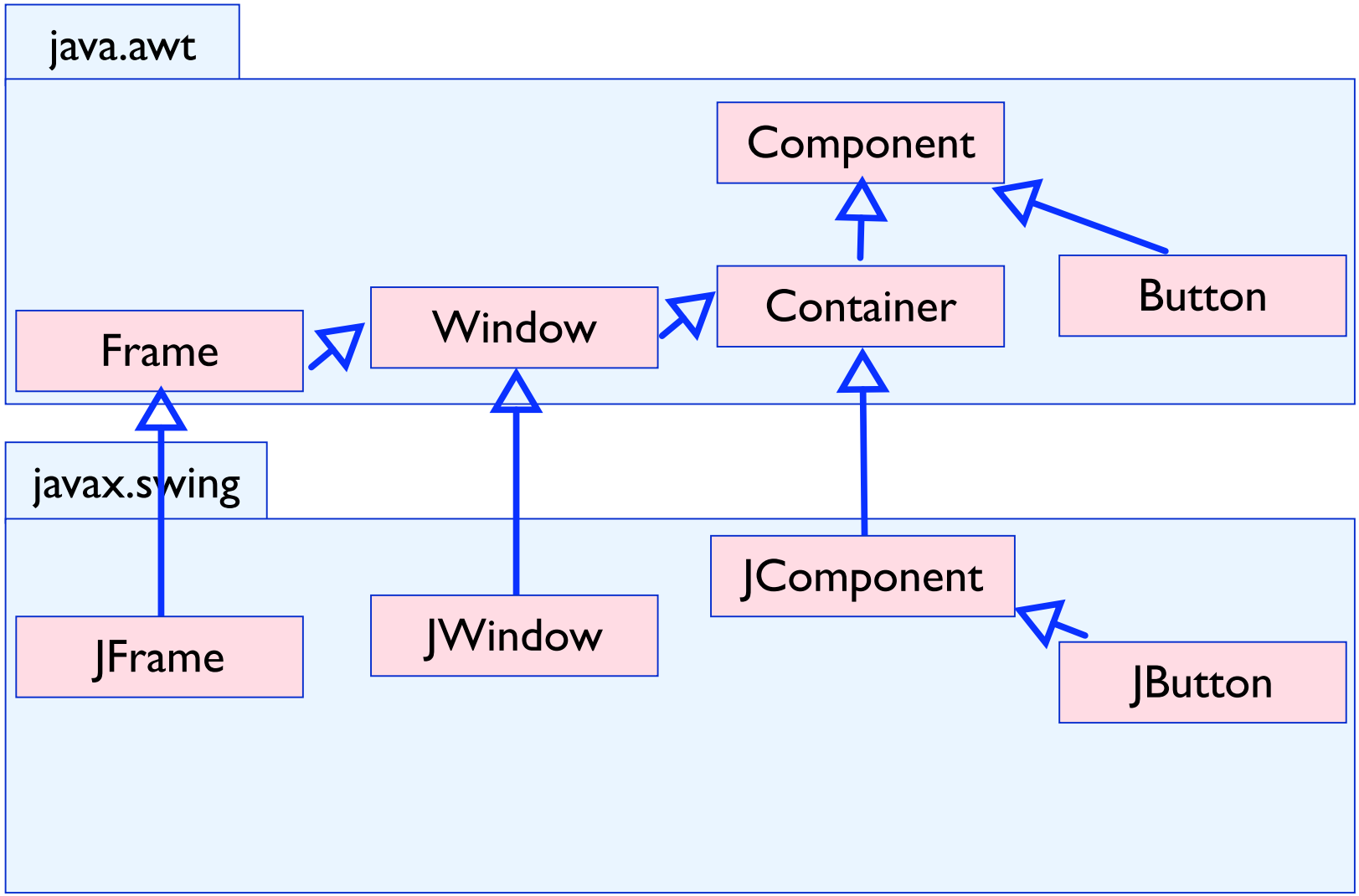
Presentation of AWT



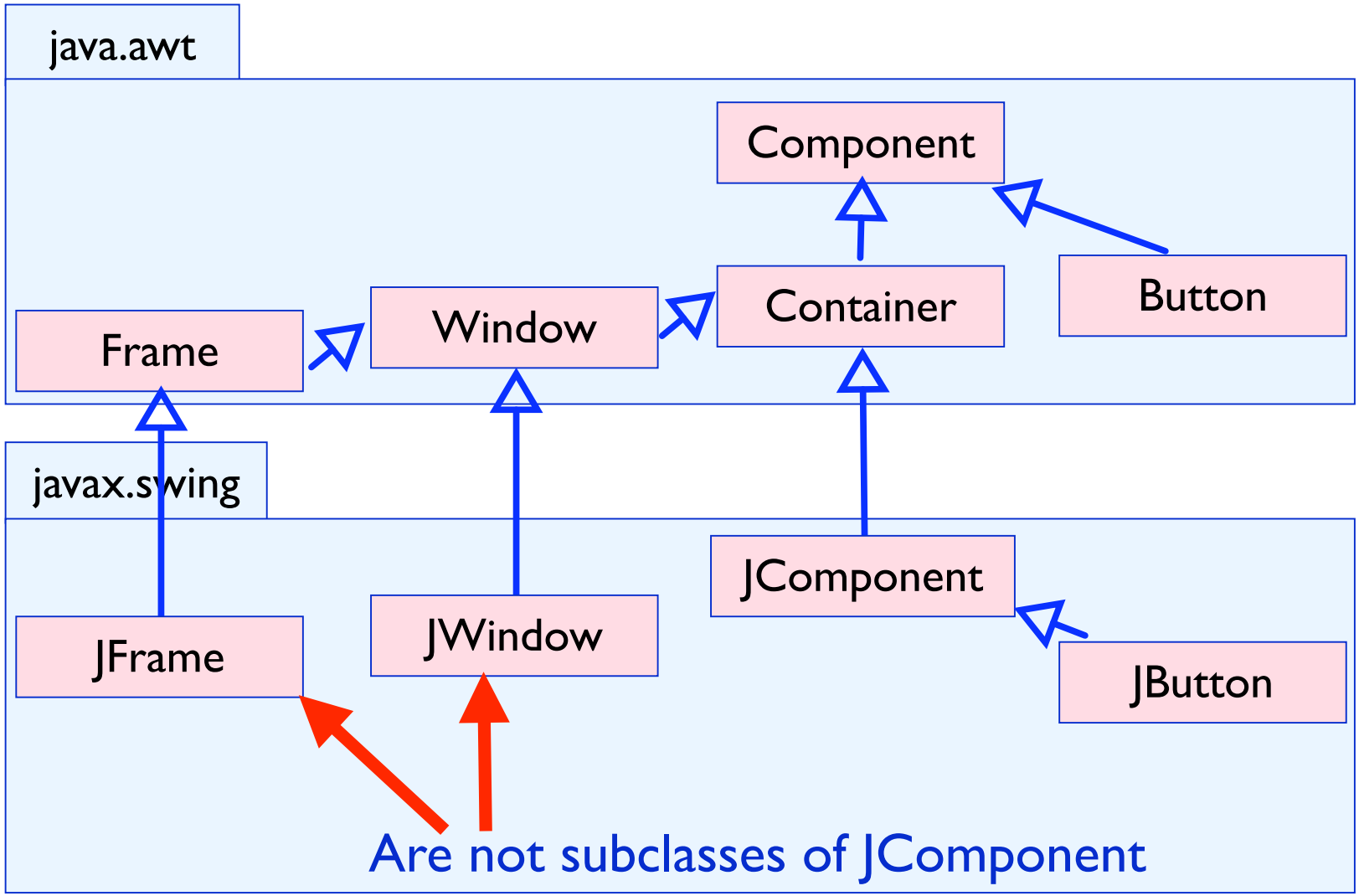
In the AWT framework:

- Widgets are components (i.e., inherit from Component)
- A frame is a window (Frame is a subclass of Window)

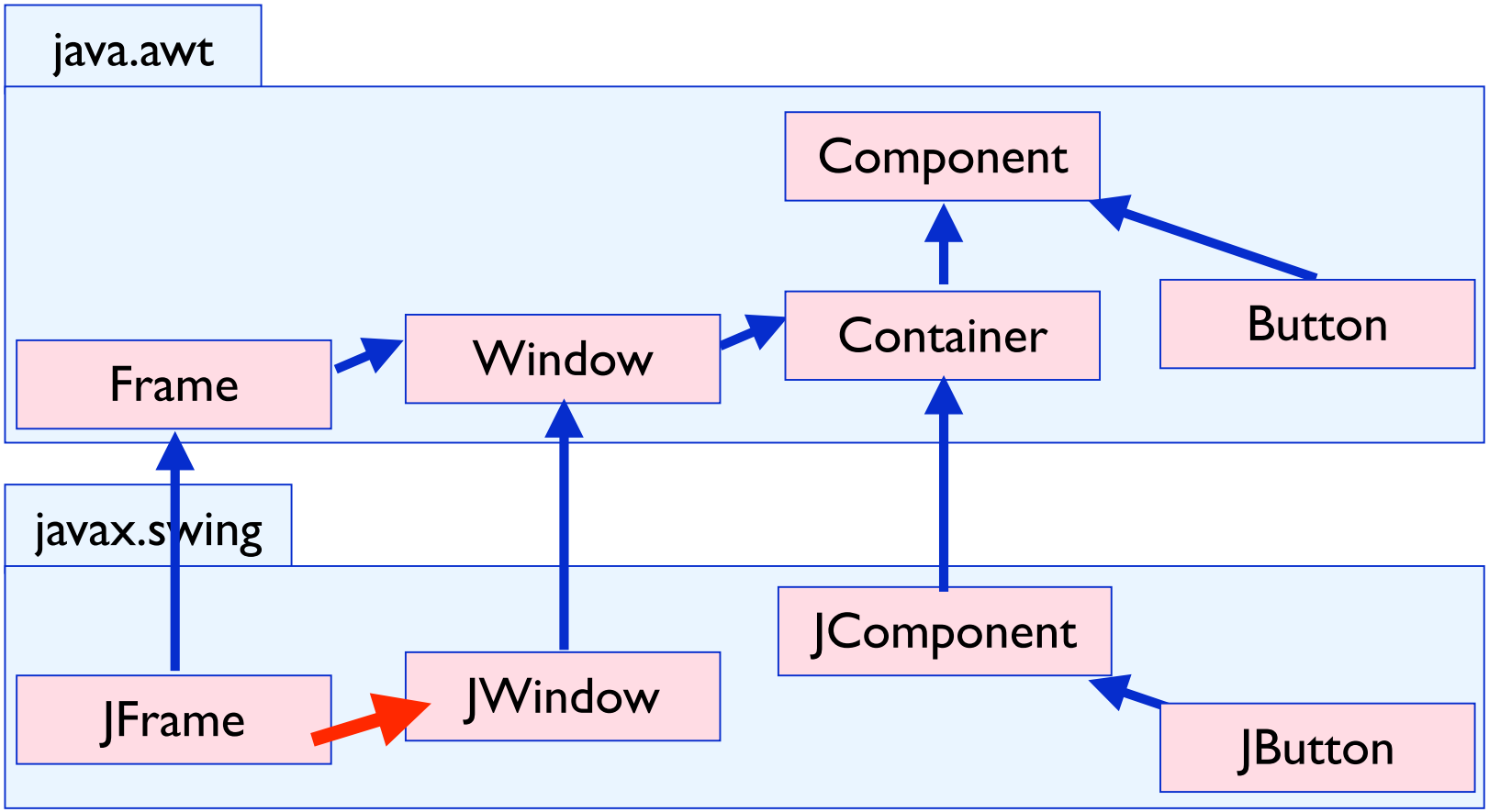
Swing at the top of AWT



Problem #1: Brocken Inheritance

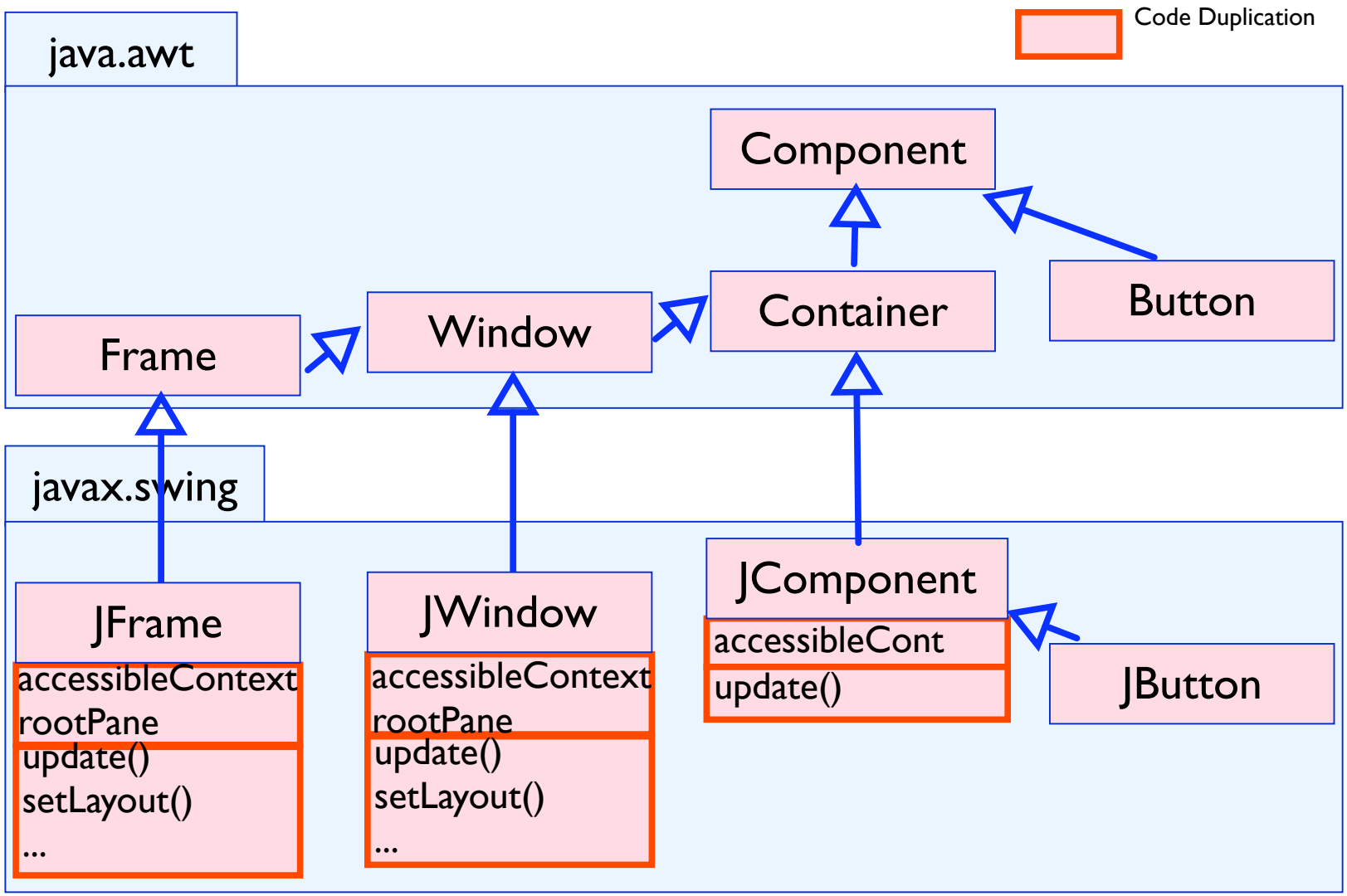


Problem #1: Brocken Inheritance



Missing inheritance link between JFrame and JWindow

Problem #2: Code Duplication



Problem #3: Explicit Type Checks and Casts

```
public class Container extends Component {
  Component components[] = new Component [0];
  public Component add (Component comp) {...}
}

public class JComponent extends Container {
  public void paintChildren (Graphics g) {
    for (; i>=0 ; i--) {
      Component comp = getComponent (i);
      isJComponent = (comp instanceof JComponent);
      ...
      ((JComponent) comp).getBounds();
    }
  }}
```

Supporting Unanticipated Changes

AWT couldn't *be enhanced* without risk of *breaking* existing code

Swing is, therefore, *built on the top* of AWT using *subclassing*

As a result, *Swing is a big mess internally*!

Why do we care to have a messy Swing ?

Swing appeared in 1998, and *has not evolved since*!

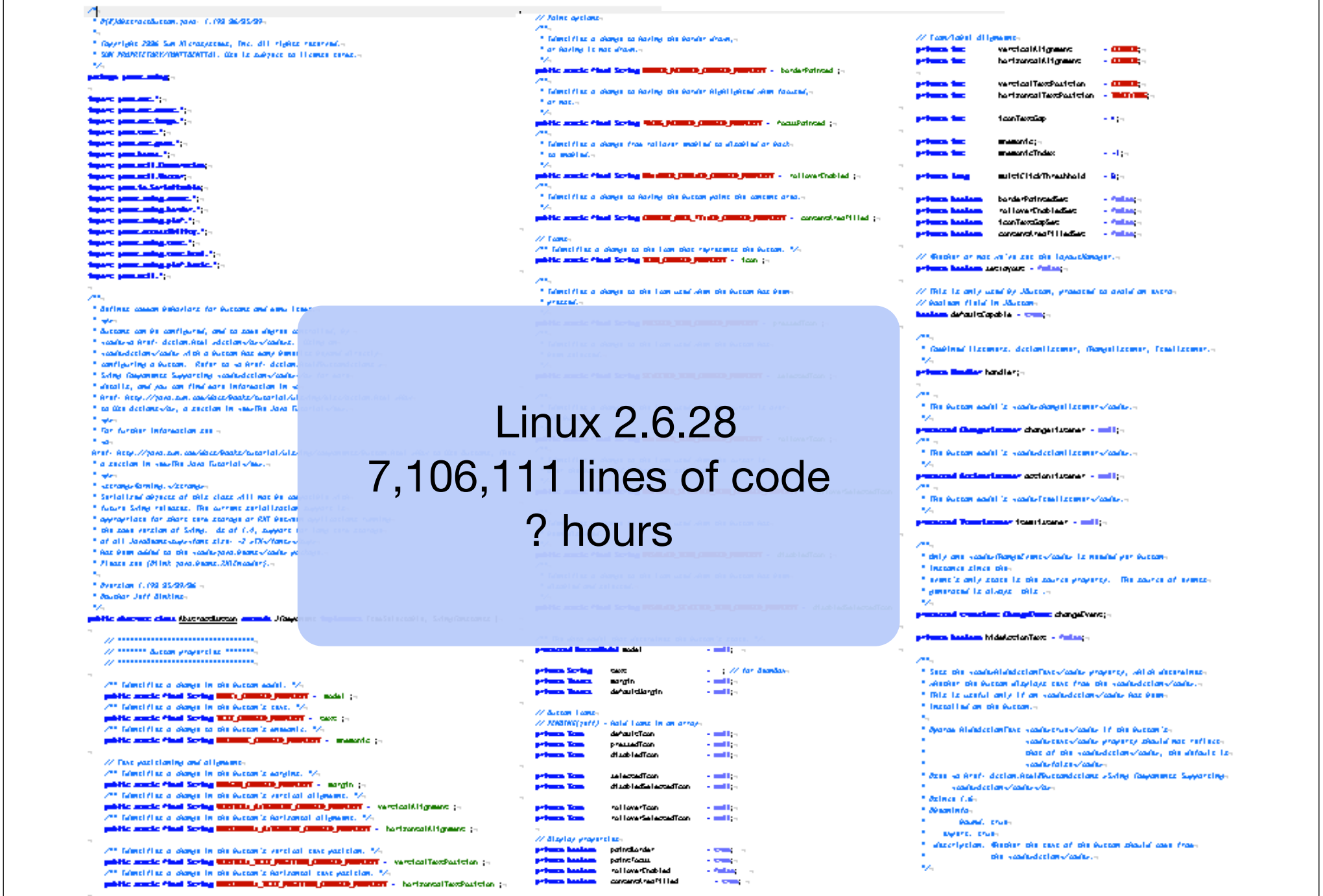
Swing is too heavy to be ported to PDA, cellphones, ...

SWT is *becoming* a *new standard*.

Either a system evolves, or it is dead. [Lehmans74]

```java
/*
 * @(#)AbstractButton.java	1.192 06/05/29
 *
 * Copyright 2006 Sun Microsystems, Inc. All rights reserved.
 * SUN PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 */
package javax.swing;

import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.text.*;
import java.awt.geom.*;
import java.beans.*;
import java.util.Enumeration;
import java.util.Vector;
import java.io.Serializable;
import javax.swing.event.*;
import javax.swing.border.*;
import javax.swing.plaf.*;
import javax.accessibility.*;
import javax.swing.text.*;
import javax.swing.text.html.*;
import javax.swing.plaf.basic.*;
import java.util.*;

/**
 * Defines common behaviors for buttons and menu items.
 * <p>
 * Buttons can be configured, and to some degree controlled, by
 * <code><a href="Action.html">Action</a></code>s.  Using an
 * <code>Action</code> with a button has many benefits beyond directly
 * configuring a button.  Refer to <a href="Action.html#buttonActions">
 * Swing Components Supporting <code>Action</code></a> for more
 * details, and you can find more information in <a
 * href="http://java.sun.com/docs/books/tutorial/uiswing/misc/action.html">How
 * to Use Actions</a>, a section in <em>The Java Tutorial</em>.
 * <p>
 * For further information see
 * <a
 href="http://java.sun.com/docs/books/tutorial/uiswing/components/button.html">How to Use Buttons, Check
 * a section in <em>The Java Tutorial</em>.
 * <p>
 * <strong>Warning:</strong>
 * Serialized objects of this class will not be compatible with
 * future Swing releases. The current serialization support is
 * appropriate for short term storage or RMI between applications running
 * the same version of Swing.  As of 1.4, support for long term storage
 * of all JavaBeans<sup><font size="-2">TM</font></sup>
 * has been added to the <code>java.beans</code> package.
 * Please see {@link java.beans.XMLEncoder}.
 *
 * @version 1.192 05/29/06
 * @author Jeff Dinkins
 */
public abstract class AbstractButton extends JComponent implements ItemSelectable, SwingConstants {

    // *********************************
    // ******* Button properties *******
    // *********************************

    /** Identifies a change in the button model. */
    public static final String MODEL_CHANGED_PROPERTY = "model";
    /** Identifies a change in the button's text. */
    public static final String TEXT_CHANGED_PROPERTY = "text";
    /** Identifies a change to the button's mnemonic. */
    public static final String MNEMONIC_CHANGED_PROPERTY = "mnemonic";

    // Text positioning and alignment
    /** Identifies a change in the button's margins. */
    public static final String MARGIN_CHANGED_PROPERTY = "margin";
    /** Identifies a change in the button's vertical alignment. */
    public static final String VERTICAL_ALIGNMENT_CHANGED_PROPERTY = "verticalAlignment";
    /** Identifies a change in the button's horizontal alignment. */
    public static final String HORIZONTAL_ALIGNMENT_CHANGED_PROPERTY = "horizontalAlignment";

    /** Identifies a change in the button's vertical text position. */
    public static final String VERTICAL_TEXT_POSITION_CHANGED_PROPERTY = "verticalTextPosition";
    /** Identifies a change in the button's horizontal text position. */
    public static final String HORIZONTAL_TEXT_POSITION_CHANGED_PROPERTY = "horizontalTextPosition";

    // Paint options
    /**
     * Identifies a change to having the border drawn,
     * or having it not drawn.
     */
    public static final String BORDER_PAINTED_CHANGED_PROPERTY = "borderPainted";
    /**
     * Identifies a change to having the border highlighted when focused,
     * or not.
     */
    public static final String FOCUS_PAINTED_CHANGED_PROPERTY = "focusPainted";
    /**
     * Identifies a change from rollover enabled to disabled or back
     * to enabled.
     */
    public static final String ROLLOVER_ENABLED_CHANGED_PROPERTY = "rolloverEnabled";
    /**
     * Identifies a change to having the button paint the content area.
     */
    public static final String CONTENT_AREA_FILLED_CHANGED_PROPERTY = "contentAreaFilled";

    // Icons
    /** Identifies a change to the icon that represents the button. */
    public static final String ICON_CHANGED_PROPERTY = "icon";

    /**
     * Identifies a change to the icon used when the button has been
     * pressed.
     */
    public static final String PRESSED_ICON_CHANGED_PROPERTY = "pressedIcon";
    /**
     * Identifies a change to the icon used when the button has
     * been selected.
     */
    public static final String SELECTED_ICON_CHANGED_PROPERTY = "selectedIcon";

    /**
     * Identifies a change to the icon used when the cursor is over
     * the button.
     */
    public static final String ROLLOVER_ICON_CHANGED_PROPERTY = "rolloverIcon";
    /**
     * Identifies a change to the icon used when the cursor is
     * over the button and it has been selected.
     */
    public static final String ROLLOVER_SELECTED_ICON_CHANGED_PROPERTY = "rolloverSelectedIcon";

    /**
     * Identifies a change to the icon used when the button has
     * been disabled.
     */
    public static final String DISABLED_ICON_CHANGED_PROPERTY = "disabledIcon";
    /**
     * Identifies a change to the icon used when the button has been
     * disabled and selected.
     */
    public static final String DISABLED_SELECTED_ICON_CHANGED_PROPERTY = "disabledSelectedIcon";


    /** The data model that determines the button's state. */
    protected ButtonModel model                = null;

    private String     text                    = ""; // for BeanBox
    private Insets     margin                  = null;
    private Insets     defaultMargin           = null;

    // Button icons
    // PENDING(jeff) - hold icons in an array
    private Icon       defaultIcon             = null;
    private Icon       pressedIcon             = null;
    private Icon       disabledIcon            = null;

    private Icon       selectedIcon            = null;
    private Icon       disabledSelectedIcon    = null;

    private Icon       rolloverIcon            = null;
    private Icon       rolloverSelectedIcon    = null;

    // Display properties
    private boolean    paintBorder             = true;
    private boolean    paintFocus              = true;
    private boolean    rolloverEnabled         = false;
    private boolean    contentAreaFilled         = true;

    // Icon/Label Alignment
    private int        verticalAlignment       = CENTER;
    private int        horizontalAlignment     = CENTER;

    private int        verticalTextPosition    = CENTER;
    private int        horizontalTextPosition  = TRAILING;

    private int        iconTextGap             = 4;

    private int        mnemonic;
    private int        mnemonicIndex           = -1;

    private long       multiClickThreshhold    = 0;

    private boolean    borderPaintedSet        = false;
    private boolean    rolloverEnabledSet      = false;
    private boolean    iconTextGapSet          = false;
    private boolean    contentAreaFilledSet    = false;

    // Whether or not we've set the LayoutManager.
    private boolean setLayout = false;

    // This is only used by JButton, promoted to avoid an extra
    // boolean field in JButton
    boolean defaultCapable = true;

    /**
     * Combined listeners: ActionListener, ChangeListener, ItemListener.
     */
    private Handler handler;

    /**
     * The button model's <code>changeListener</code>.
     */
    protected ChangeListener changeListener = null;
    /**
     * The button model's <code>ActionListener</code>.
     */
    protected ActionListener actionListener = null;
    /**
     * The button model's <code>ItemListener</code>.
     */
    protected ItemListener itemListener = null;

    /**
     * Only one <code>ChangeEvent</code> is needed per button
     * instance since the
     * event's only state is the source property.  The source of events
     * generated is always "this".
     */
    protected transient ChangeEvent changeEvent;

    private boolean hideActionText = false;

    /**
     * Sets the <code>hideActionText</code> property, which determines
     * whether the button displays text from the <code>Action</code>.
     * This is useful only if an <code>Action</code> has been
     * installed on the button.
     *
     * @param hideActionText <code>true</code> if the button's
     *                       <code>text</code> property should not reflect
     *                       that of the <code>Action</code>; the default is
     *                       <code>false</code>
     * @see <a href="Action.html#buttonActions">Swing Components Supporting
     *      <code>Action</code></a>
     * @since 1.6
     * @beaninfo
     *        bound: true
     *    expert: true
     *  description: Whether the text of the button should come from
     *               the <code>Action</code>.
     */
```


J2EE
3,430,663 lines of code
? hours


Linux 2.6.28
7,106,111 lines of code
? hours

How large is your project?

1’000’000 lines of code

How large is your project?

1’000’000 lines of code

* 2 = 2’000’000 seconds

How large is your project?

1’000’000 lines of code

* 2 = 2’000’000 seconds

/ 3600 = 560 hours

How large is your project?

1’000’000 lines of code

* 2 = 2’000’000 seconds

/ 3600 = 560 hours

/ 8 = 70 days

How large is your project?

1’000’000 lines of code

* 2 = 2’000’000 seconds

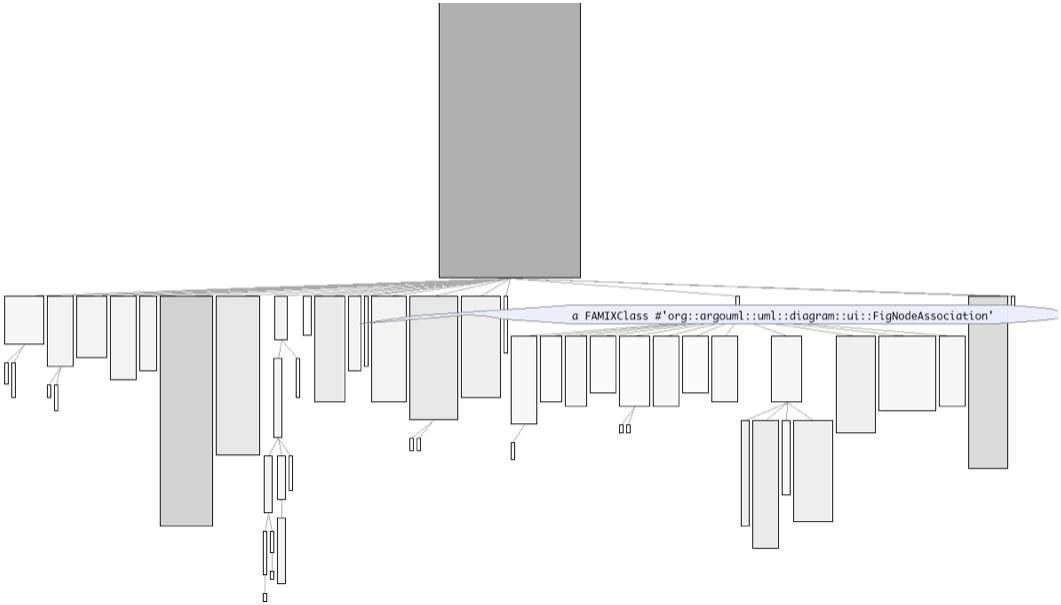
/ 3600 = 560 hours

/ 8 = 70 days

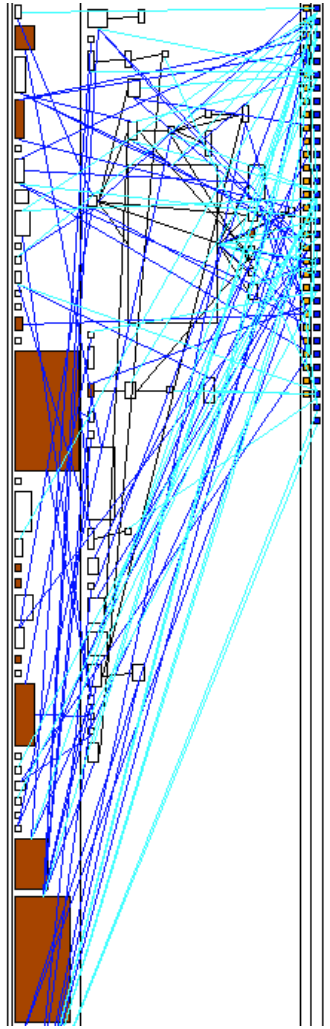
/20 = 3 months

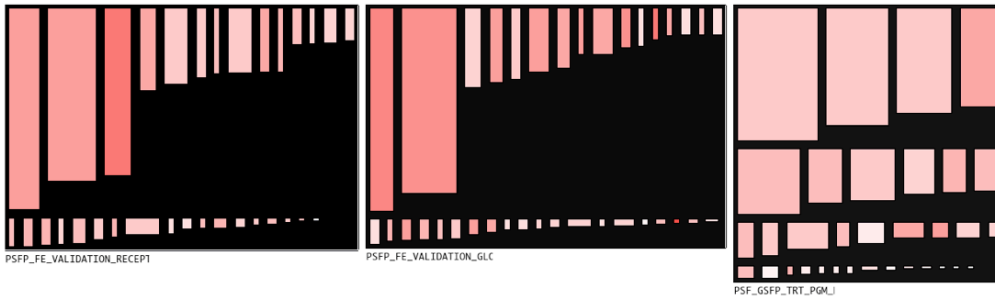
Roadmap

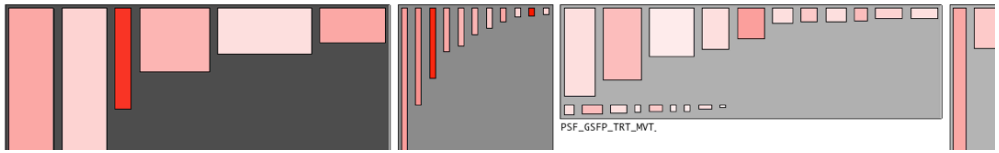
1. Real life maintenance problems
2. Software Quality
3. Example of code quality
4. **Software visualization help taming software complexity** 

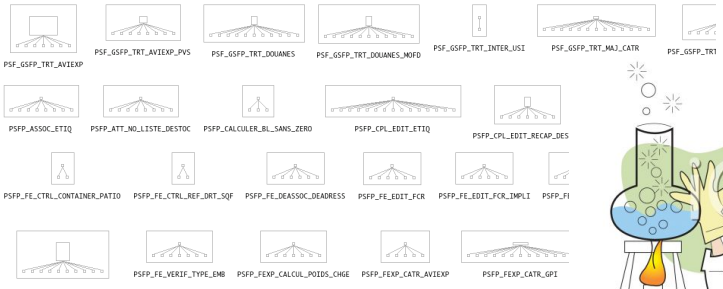


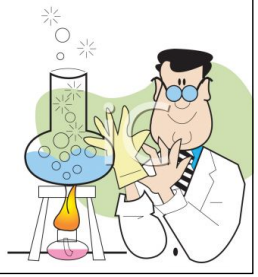
Software maps

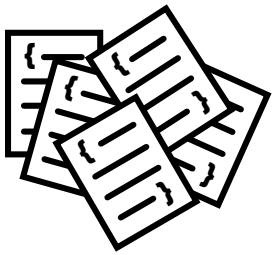












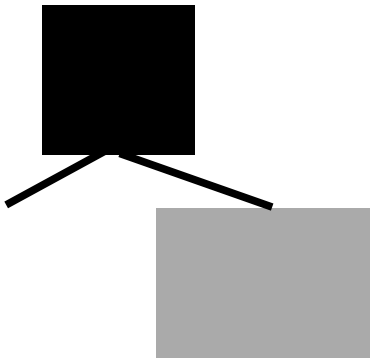
source code

McCabe = 21

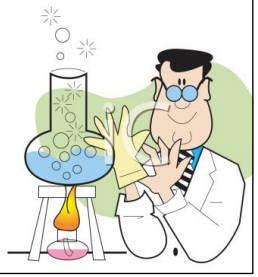
NOM = 102

LOC = 753,000

metrics

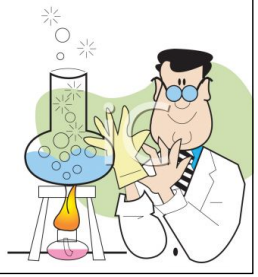


maps

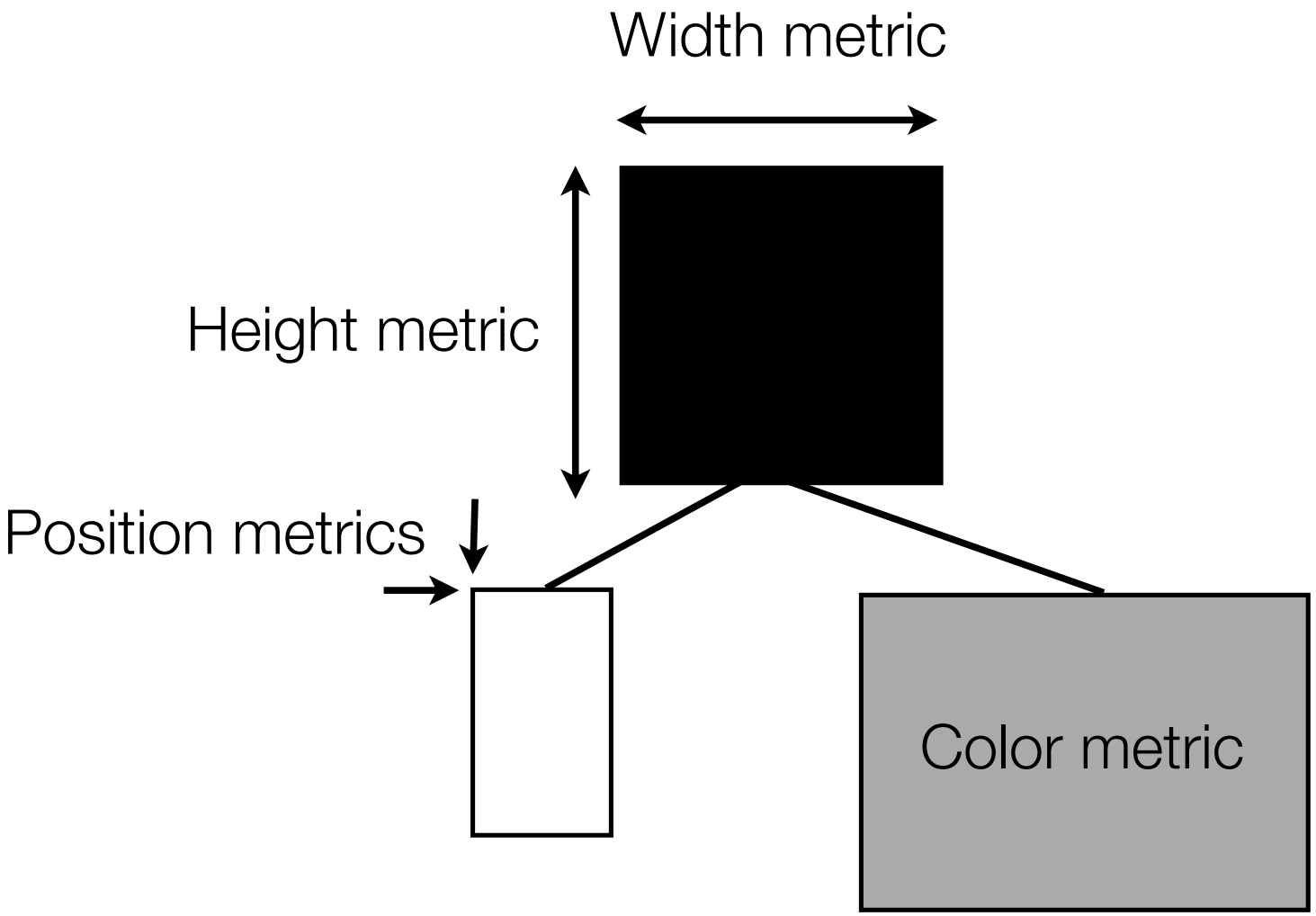


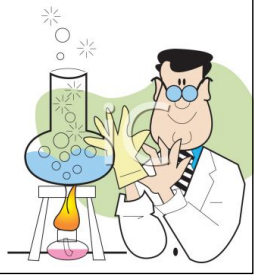
Metrics compress the system into numbers

NOM	NOC	DUPLINES
LOC	NOCmts	NAI
TCC	NOPA	NOA
WMC	WLOC	NI
CYCLO	WNOC	...
ATFD	WOC	
HNL	MSG	

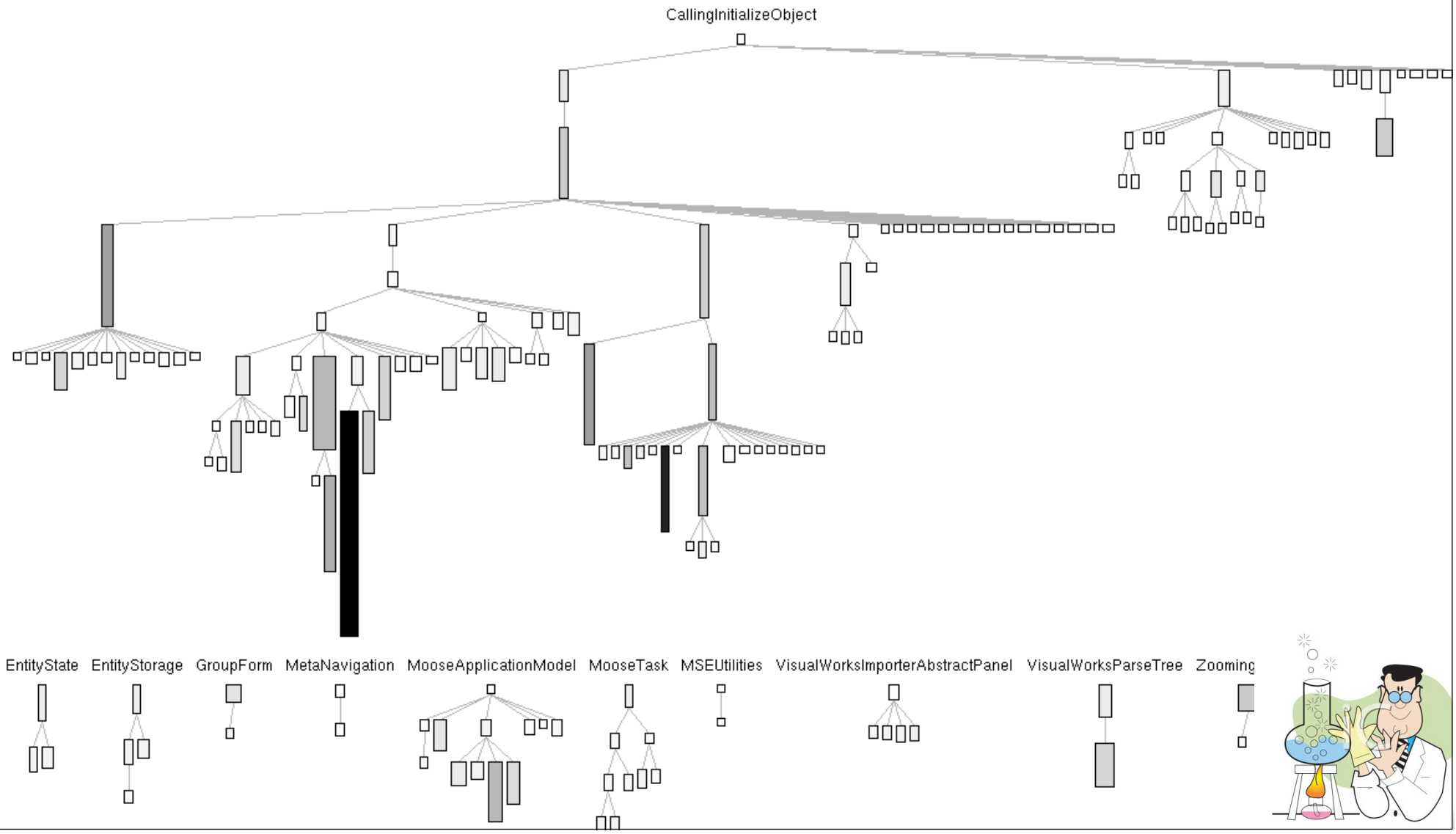


Polymetric views shows up to 5 metrics



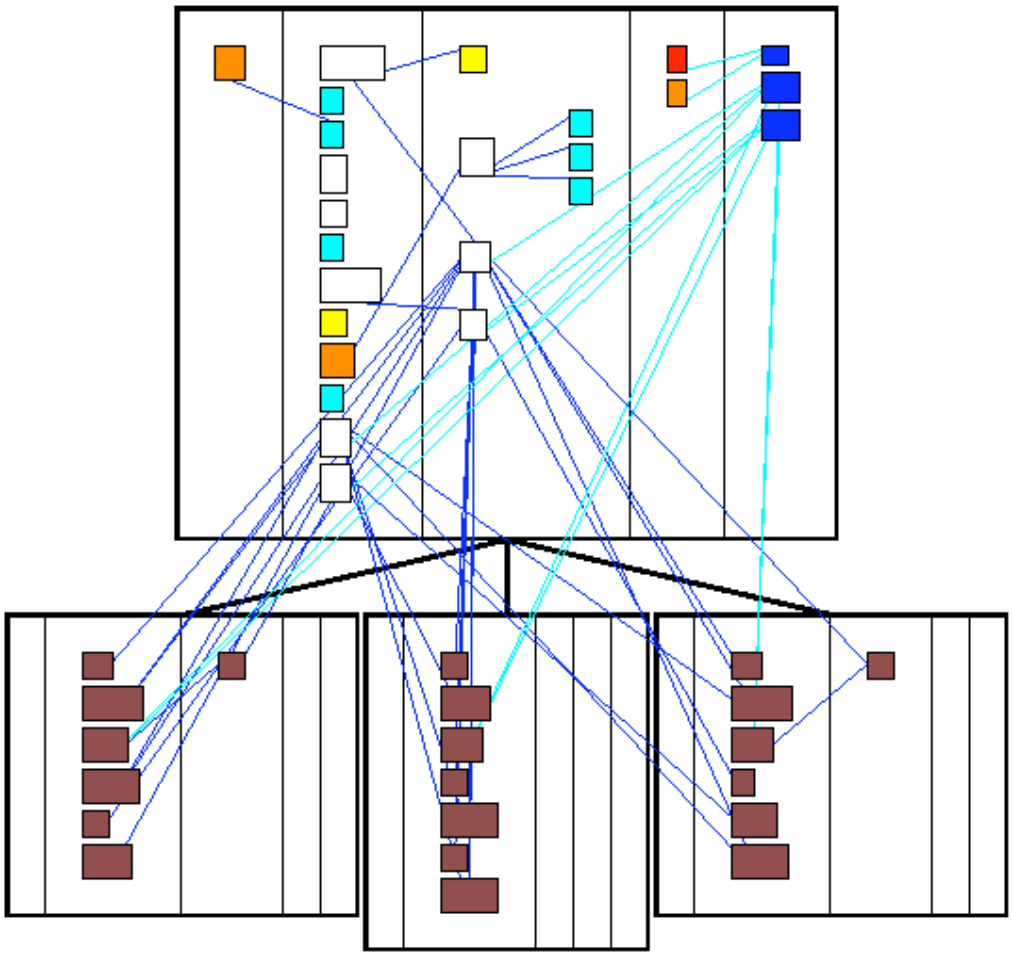


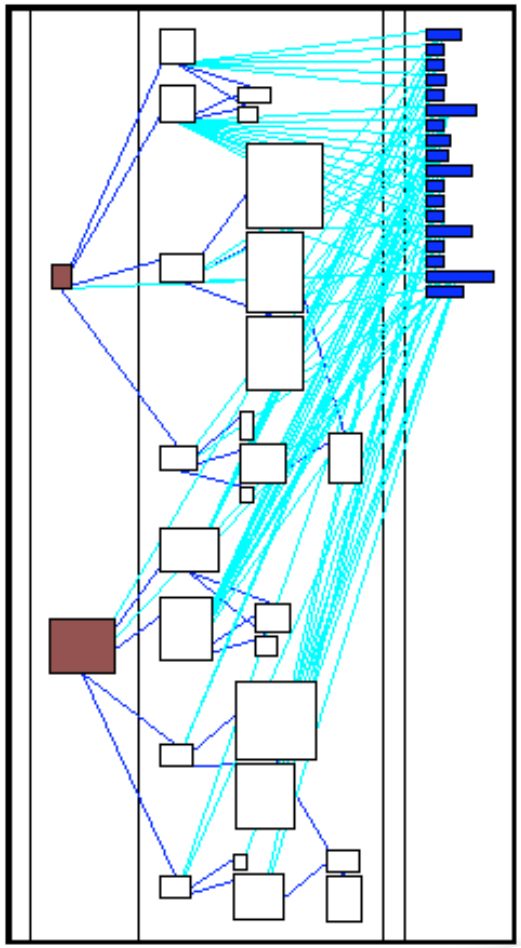
System complexity shows class hierarchy



Class blueprint

shows class internals

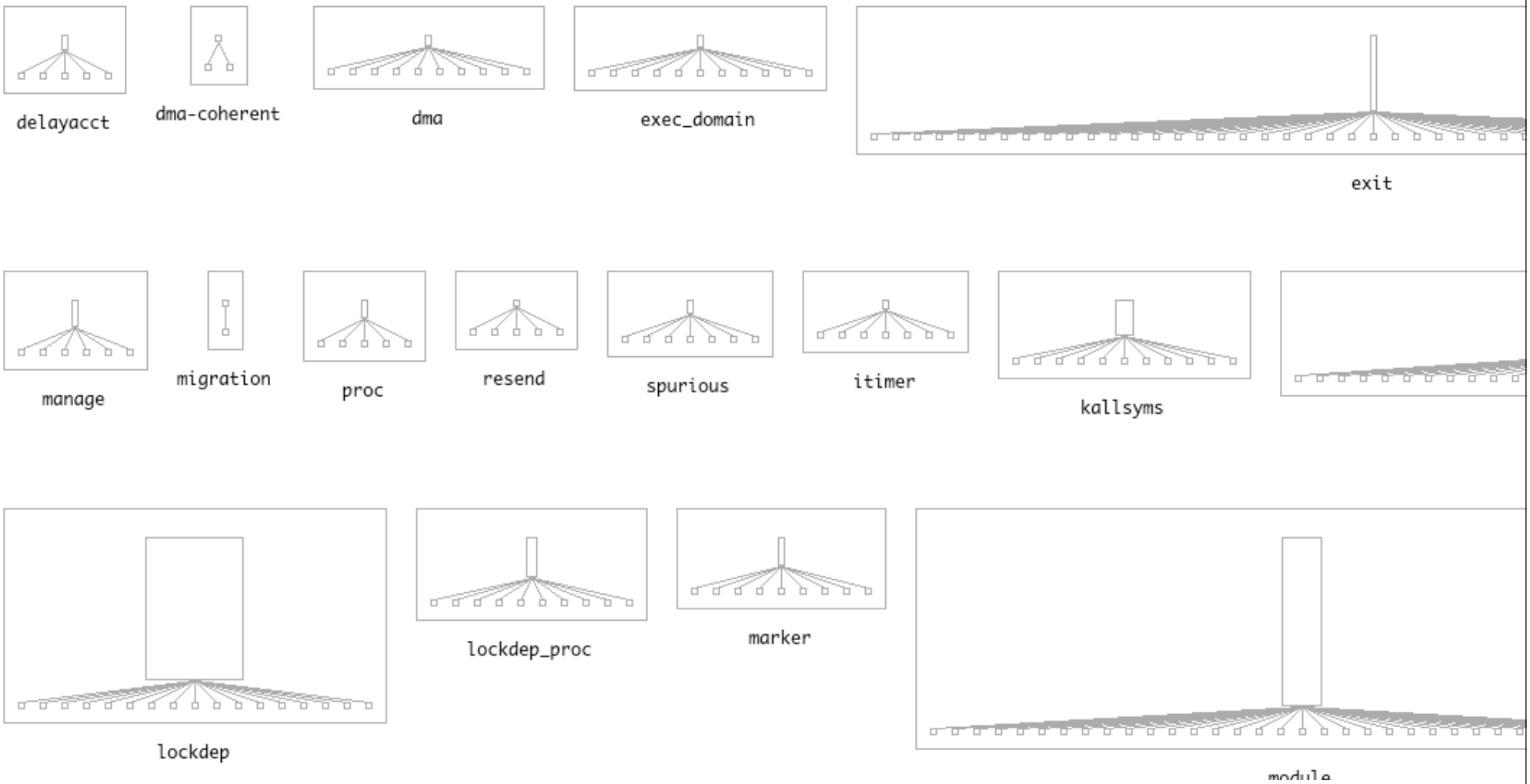




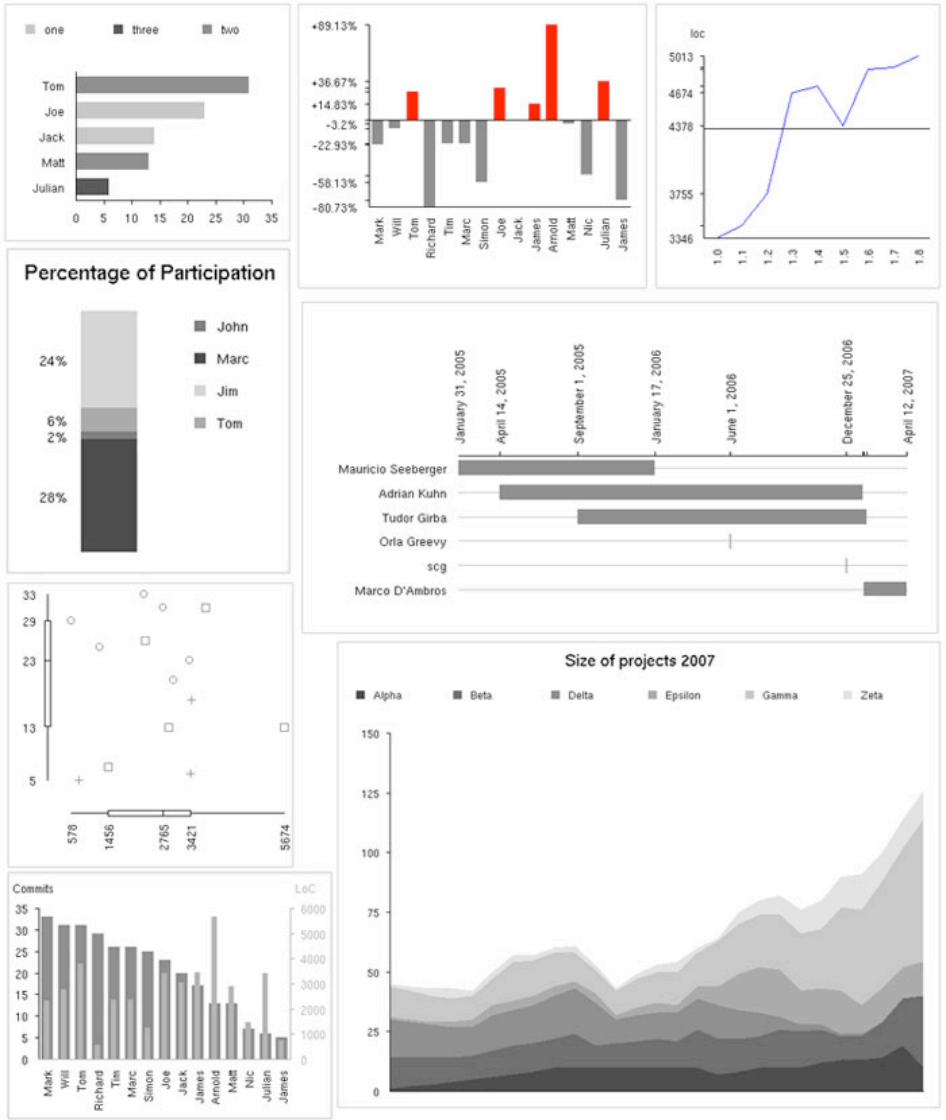
Module complexity
for C modules

Module dependencies

for C modules

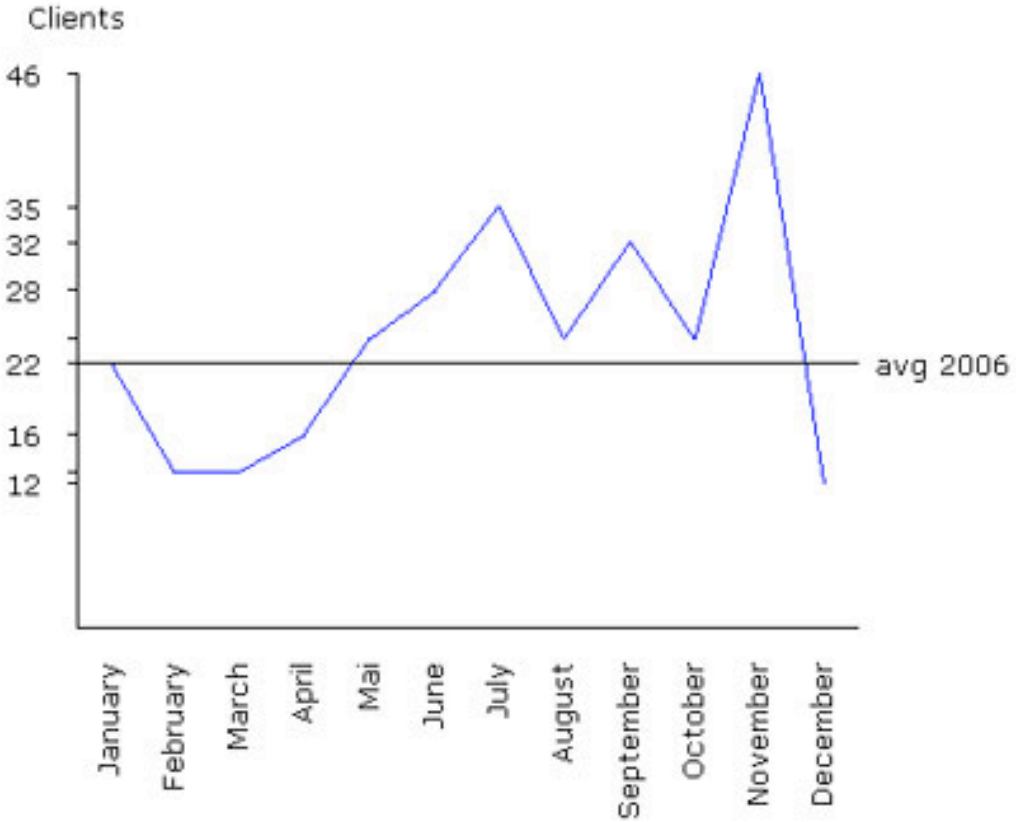


EyeSee

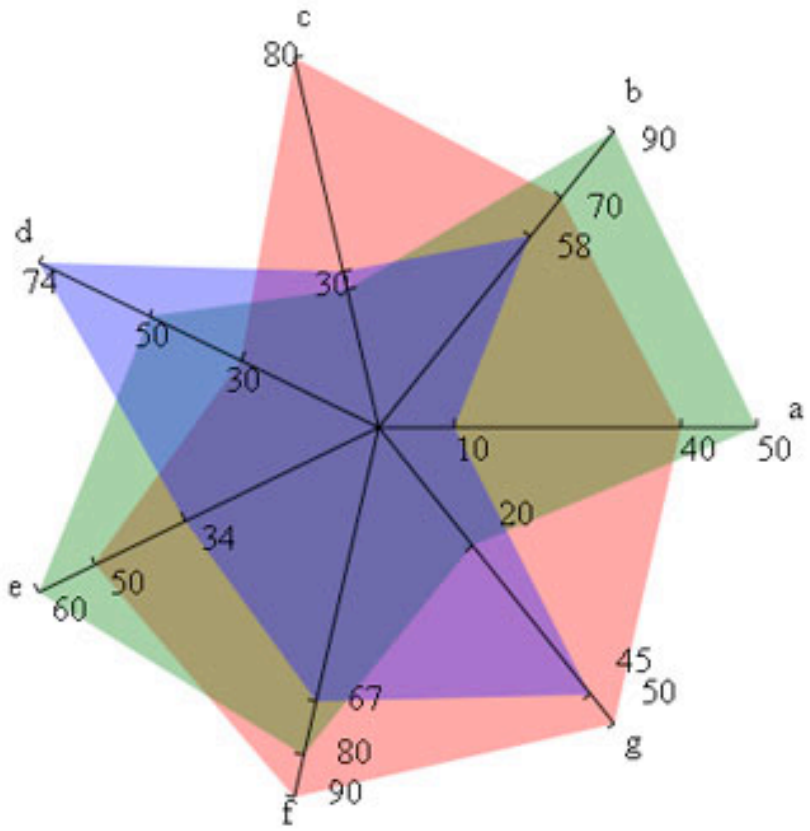


Matthias Junker and
Markus Hofstetter
2007

University of Bern,
Switzerland

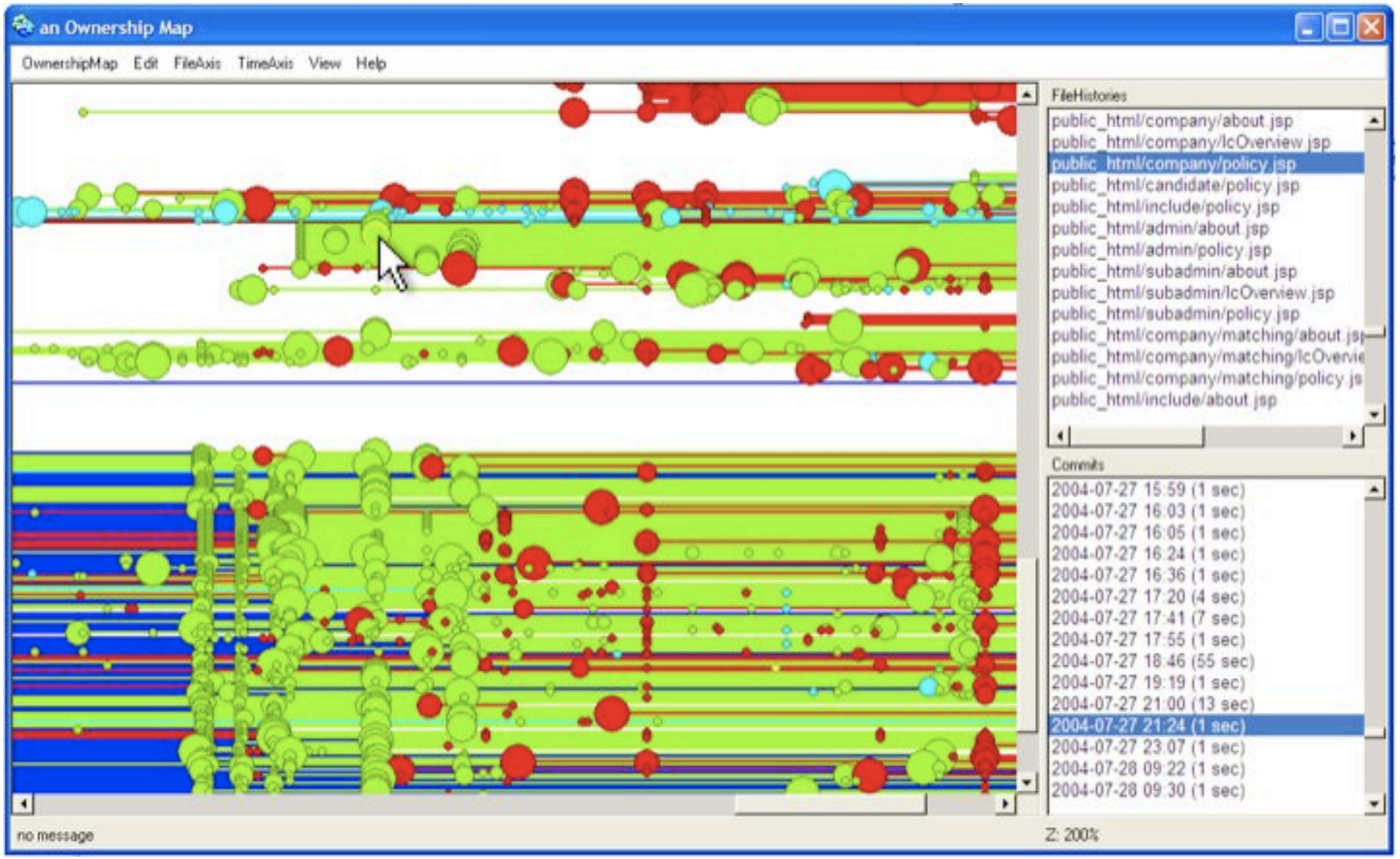



```
diag := DiagramRenderer new.
diag lineDiagram
   y: #income;
   identifier: #identifier;
   defaultColor: #blue;
   yAxisLabel: 'Clients';
   valueAxis;
   deviationValue: clients2006 avg;
   deviationDescription: 'avg 2006';
   models: clients2007.
diag open.
```

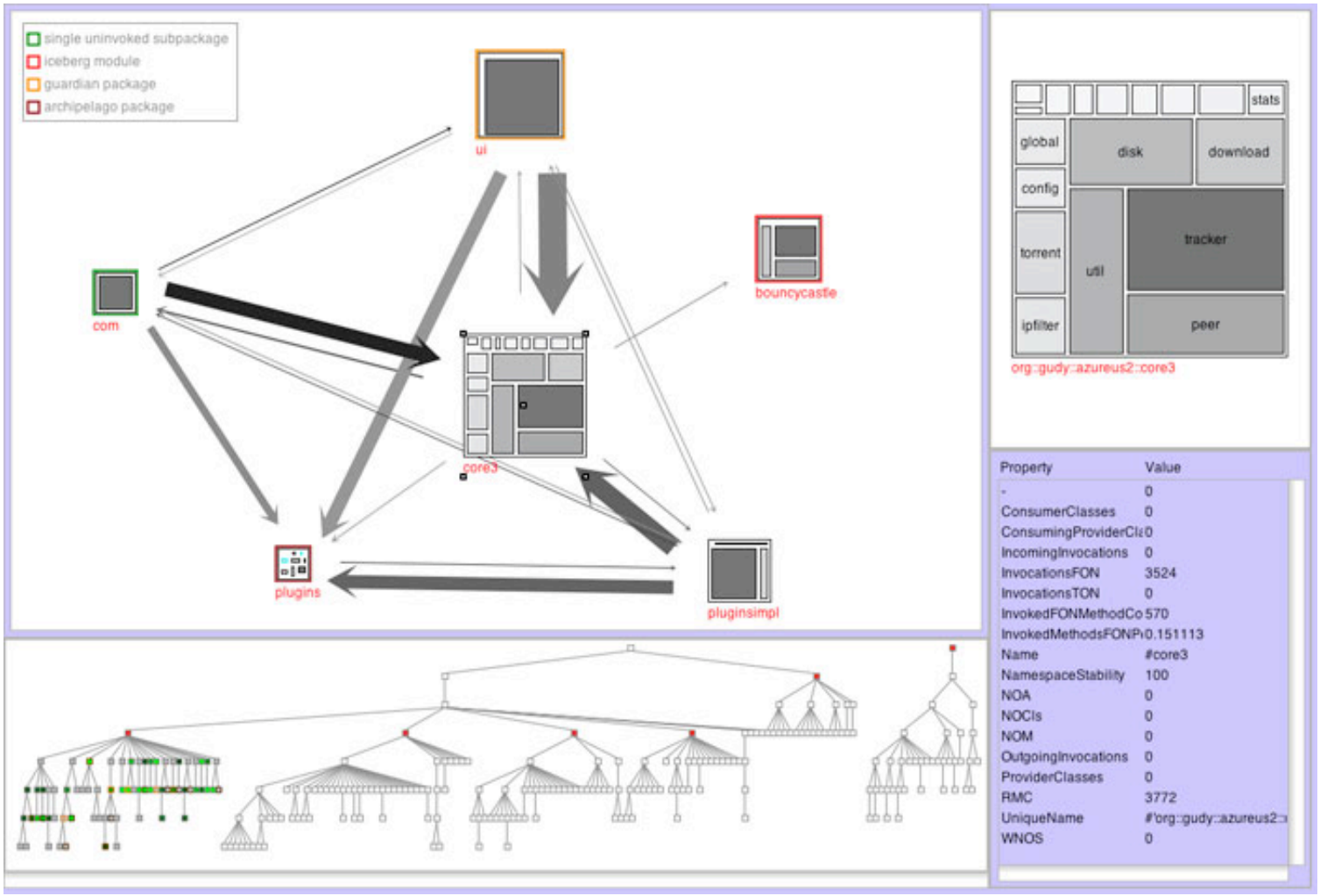



```
diag := DiagramRenderer new.
diag spider
  attributes: #(#a #b #c #d #e  #f #g );
  valueAxis;
  models: self spiderModel.
diag cairo.
```

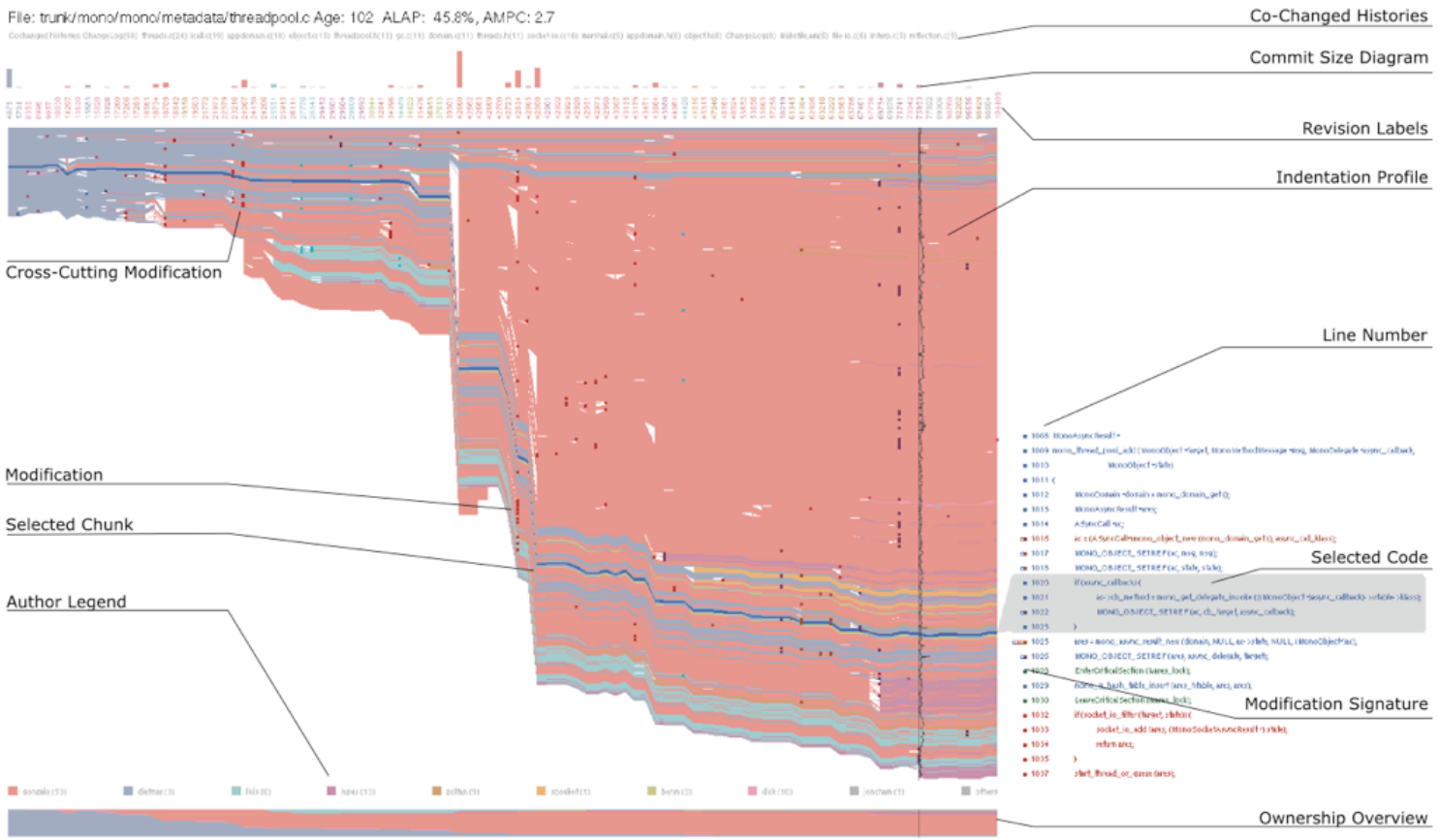
Chronia (Seeberger, Uni Berna)



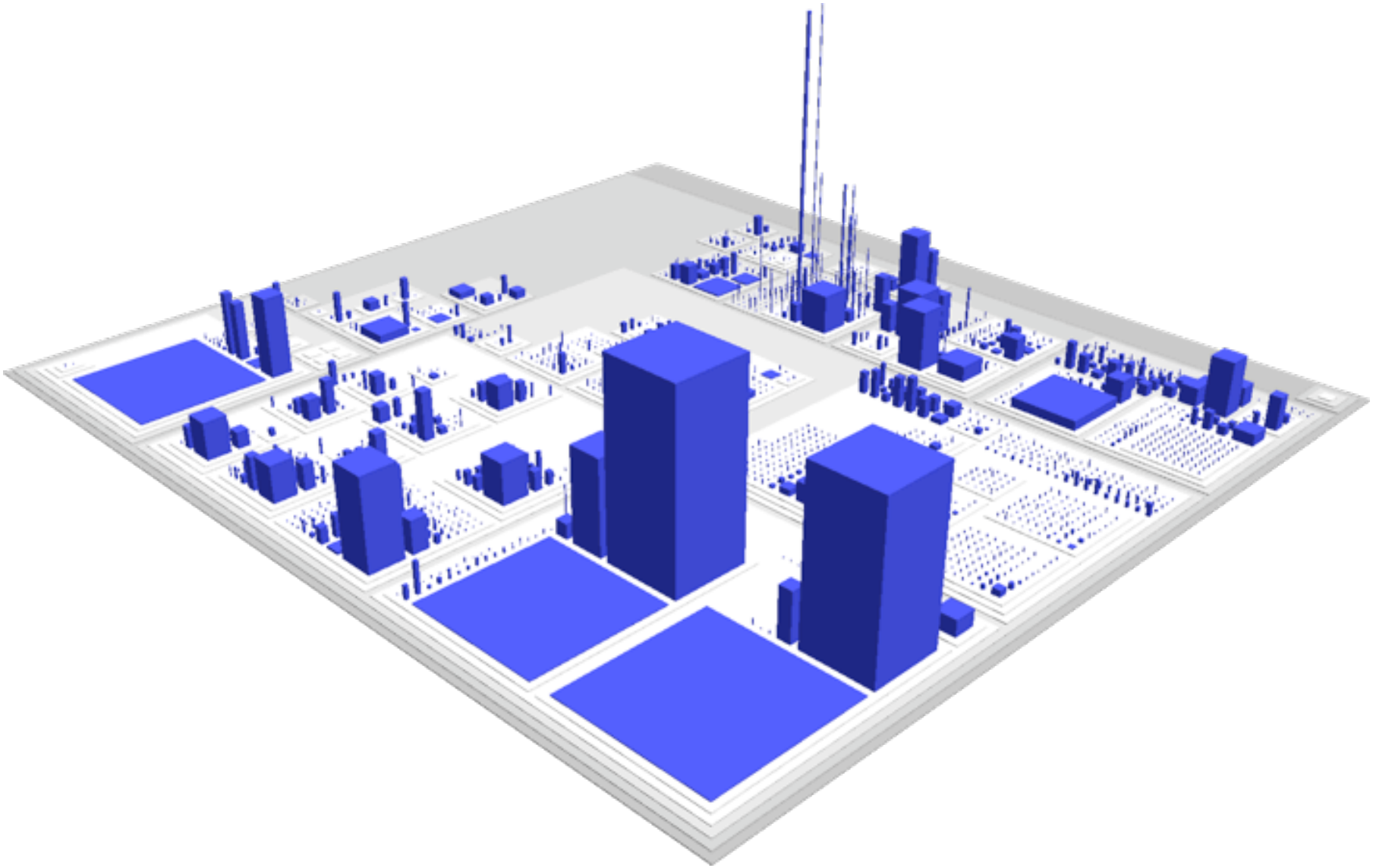
Softwarenaut (Lungo, Uni Lugano)



Yellow submarine (Univ Berna)



CodeCity (Wettel, Univ Lugano)



Conclusion

Software quality is hard to achieve

Difficult and varied topic

- No definitive answer
- Process? Information flow? Program execution?

Insuring software quality require external dedicated tools (e.g., profilers, code coverage)

What you should know!

What is class inheritance

When to use it

What are the difficult when designing a class hierarchy

Understand what are the real challenges in designing long-living software systems

Can you answer to these questions?

Will the problem identified in the Java Swing Library exist in C++?

Why using instanceof is considered a bad programming practice?

What is the essential differences between object-oriented programming and procedural programming?

License

http://creativecommons.org/licenses/by-sa/2.5

creative commons

COMMONS DEED

Attribution-ShareAlike 2.5

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.