



# Lessons learnt by building Intensive

Andy Kellens

Johan Brichau

Sergio Castro

Kim Mens



Programming Technology Lab - Vrije Universiteit Brussel

Département d'Ingénierie Informatique - Université catholique de Louvain

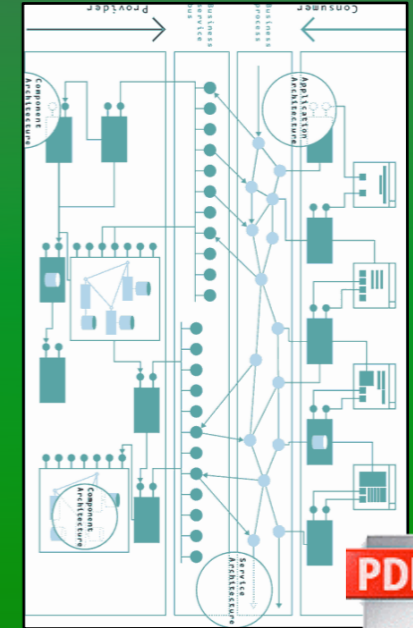


# Intensional Views in 60 seconds



INTENSIVE  
INTENSIVE

## Architectural Design Documents



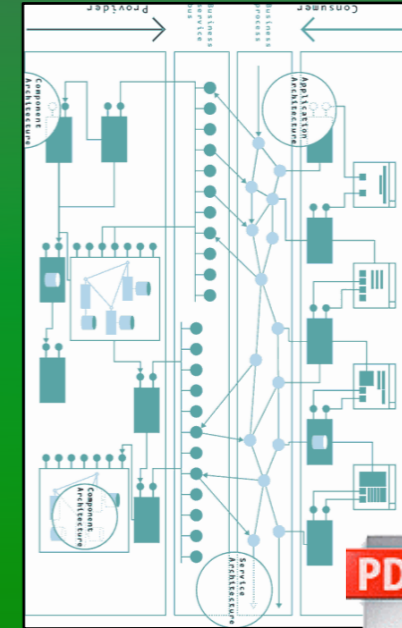
PDF



# Intensional Views in 60 seconds

- ▶ **Coding conventions**
- ▶ **Protocols**
- ▶ **Design patterns**
- ▶ **Modular structure**
- ▶ **Best practices**
- ▶ **...**

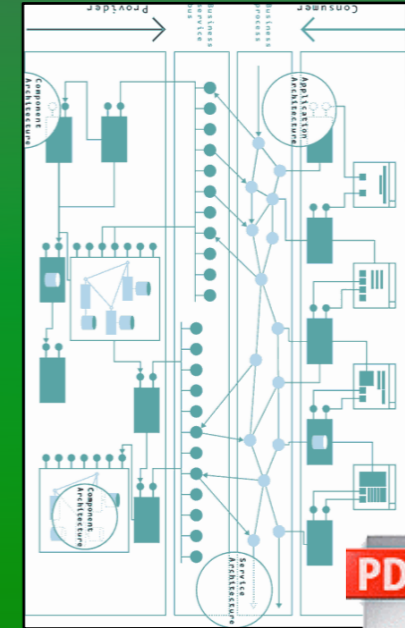
## Architectural Design Documents



# Intensional Views in 60 seconds

Developer guidelines  
System design structure  
ADL, UML, English...

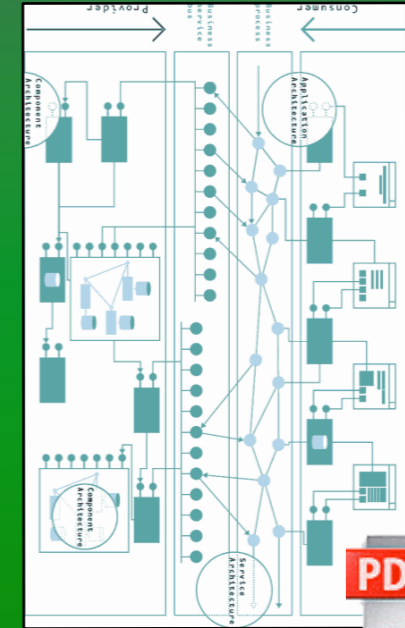
## Architectural Design Documents



# Intensional Views in 60 seconds

Developer guidelines  
System design structure  
ADL, UML, English...

## Architectural Design Documents

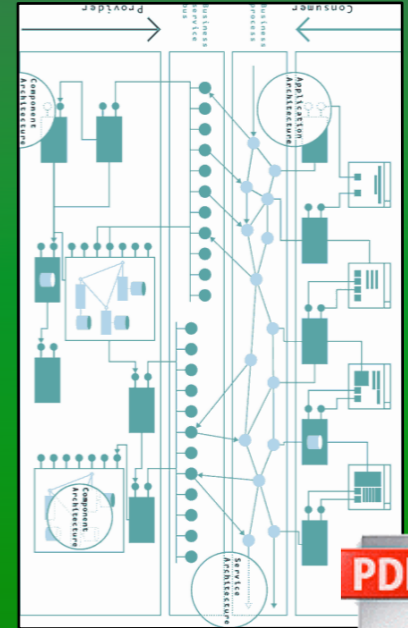


# Intensional Views in 60 seconds



Developer guidelines  
System design structure  
ADL, UML, English...

## Architectural Design Documents



## Software Implementation

```
import java.io.*;
import java.util.*;

/**
 * Comment line program to copy a file to another directory.
 */
public class CopyFile {
    // constant values for the source and destination
    private static final int SOURCE_PATH = 1;
    private static final int DEST_PATH = 2;

    // program options (initialized to default values)
    private static int sourcePath = SOURCE_PATH;
    private static int destPath = DEST_PATH;
    private static boolean copyRecursive = true;
    private static int overwrite = OVERWRITE_NONE;

    public static long copyFile(String src, File destDir)
        throws IOException {
        return copyFile(src, destDir, true);
    }

    private static long copyFile(String src, File destDir,
        boolean recursive) throws IOException {
        File srcFile = new File(src);
        if (!srcFile.exists())
            return 0;

        if (!srcFile.isDirectory())
            return copyFile(src, destDir, recursive);

        File destDirFile = new File(destDir, srcFile.getName());
        if (destDirFile.exists()) {
            if (overwrite == OVERWRITE_NONE)
                return 0;
            if (overwrite == OVERWRITE_REPLACE)
                destDirFile.delete();
        }

        if (recursive) {
            File[] files = srcFile.listFiles();
            if (files != null)
                for (File file : files)
                    copyFile(file.getAbsolutePath(), destDir, recursive);
        }

        return copyFile(srcFile.getAbsolutePath(), destDirFile, recursive);
    }

    public static void main(String[] args) {
        if (args.length < 2)
            System.out.println("Usage: java CopyFile <sourcePath> <destPath>");
        else {
            sourcePath = Integer.parseInt(args[0]);
            destPath = Integer.parseInt(args[1]);
        }
    }
}

public class HappyNewYear implements Runnable {
    private static NumberFormat
    formatter = NumberFormat.getInstance();
    private JFrame frame;
    private JLabel label;
    private long newYearsMillis;
    private String message;

    public HappyNewYear(JFrame frame, JLabel label) {
        // store
        this.frame = frame;
        this.label = label;
        // compute
        Calendar cal =
        Calendar.getInstance();
        int nextYear =
        cal.get(Calendar.YEAR) + 1;
        nextYear;
        cal.set(Calendar.YEAR,
        nextYear);
        cal.set(Calendar.MONTH,
        Calendar.JANUARY);
    }

    public void run() {
        // store argument
        this.frame = frame;
        this.label = label;
        // compute
        Calendar cal = new
        GregorianCalendar();
        int nextYear =
        cal.get(Calendar.YEAR) + 1;
    }
}

public class FileDownload {
    public static void download(String
    address, String localFileName) {
        OutputStream out =
        null;
        URLConnection conn =
        null;
        InputStream in = null;
        try {
            URL url
            = new URL(address);
        }
    }
}

public class HappyNewYear implements Runnable {
    private static
    NumberFormat formatter =
    NumberFormat.getInstance();
    private JFrame
    frame;
    private JLabel
    label;
    private long
    newYearsMillis;
    private String
    message;

    public
    HappyNewYear(JFrame
    frame, JLabel label)
    {
        // store argument
        this.frame =
        frame;
        this.label =
        label;
    }
}

public class HappyNewYear implements Runnable {
    private static NumberFormat
    formatter = NumberFormat.getInstance();
    private JFrame frame;
    private JLabel label;
    private long newYearsMillis;
    private String message;

    public HappyNewYear(JFrame
    frame, JLabel label)
    {
        // store argument
        this.frame =
        frame;
        this.label =
        label;
        // compute
        Calendar cal = new
        GregorianCalendar();
        int nextYear =
        cal.get(Calendar.YEAR) + 1;
    }
}
```



Source code  
(Java / C / Cobol /...)

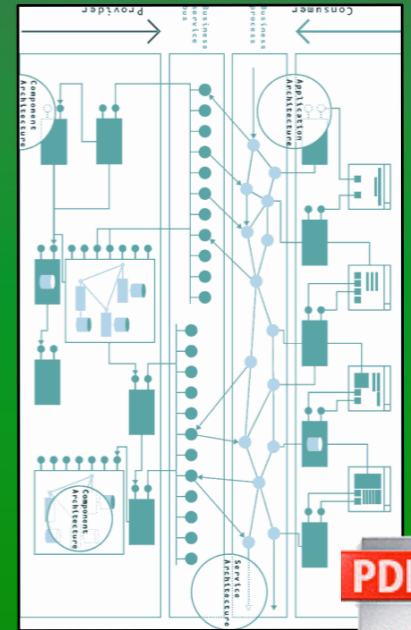


# Intensional Views in 60 seconds



Developer guidelines  
System design structure  
ADL, UML, English...

## Architectural Design Documents



## Software Implementation

```
import java.io.*;
import java.util.*;

public class CopyFile {
    // constants for the source and destination
    private static final int SOURCE_PATH = 1;
    private static final int DEST_PATH = 2;
    private static final int FILE_NAME = 3;

    // program options (initialized to default values)
    private static int sourcePath = SOURCE_PATH;
    private static int destPath = DEST_PATH;
    private static String fileName = "copy.txt";

    // constructor
    public CopyFile(int sourcePath, int destPath, String fileName) {
        this.sourcePath = sourcePath;
        this.destPath = destPath;
        this.fileName = fileName;
    }

    // main method
    public static void main(String[] args) {
        CopyFile copyFile = new CopyFile(sourcePath, destPath, fileName);
        copyFile.run();
    }

    // run method
    private void run() {
        try {
            // create the source file
            File sourceFile = new File("src/" + fileName);
            if (!sourceFile.exists()) {
                sourceFile.createNewFile();
            }

            // create the destination file
            File destFile = new File("dest/" + fileName);
            if (!destFile.exists()) {
                destFile.createNewFile();
            }

            // copy the file
            FileInputStream in = new FileInputStream(sourceFile);
            FileOutputStream out = new FileOutputStream(destFile);
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = in.read(buffer)) != -1) {
                out.write(buffer, 0, bytesRead);
            }
            in.close();
            out.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

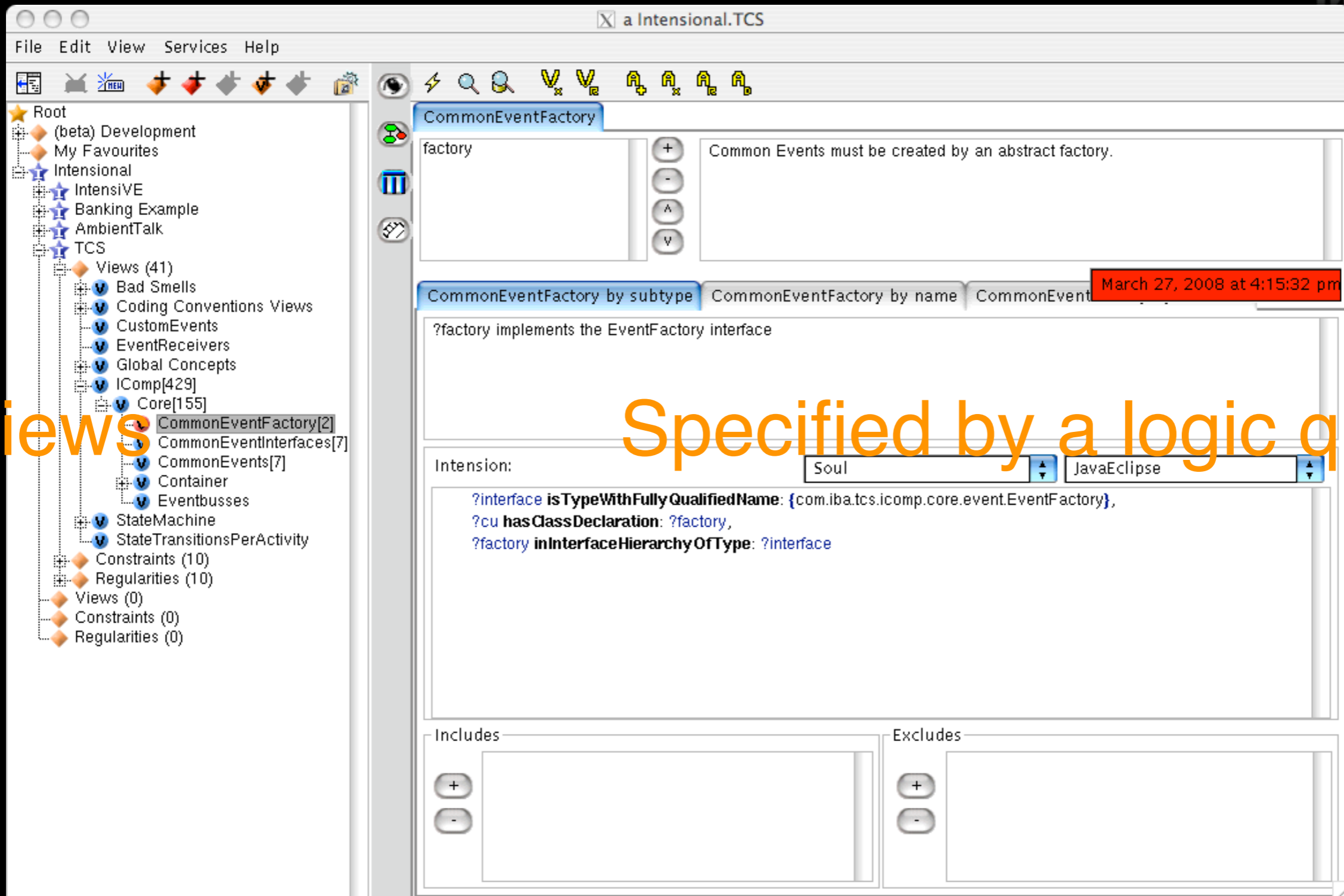


Source code  
(Java / C / Cobol /...)









File Edit View Services Help

CommonEventFactory

factory + Common Events must be created by an abstract factory.

CommonEventFactory by subtype CommonEventFactory by name CommonEventFactory by name

March 27, 2008 at 4:15:32 pm

?factory implements the EventFactory interface

Intension:

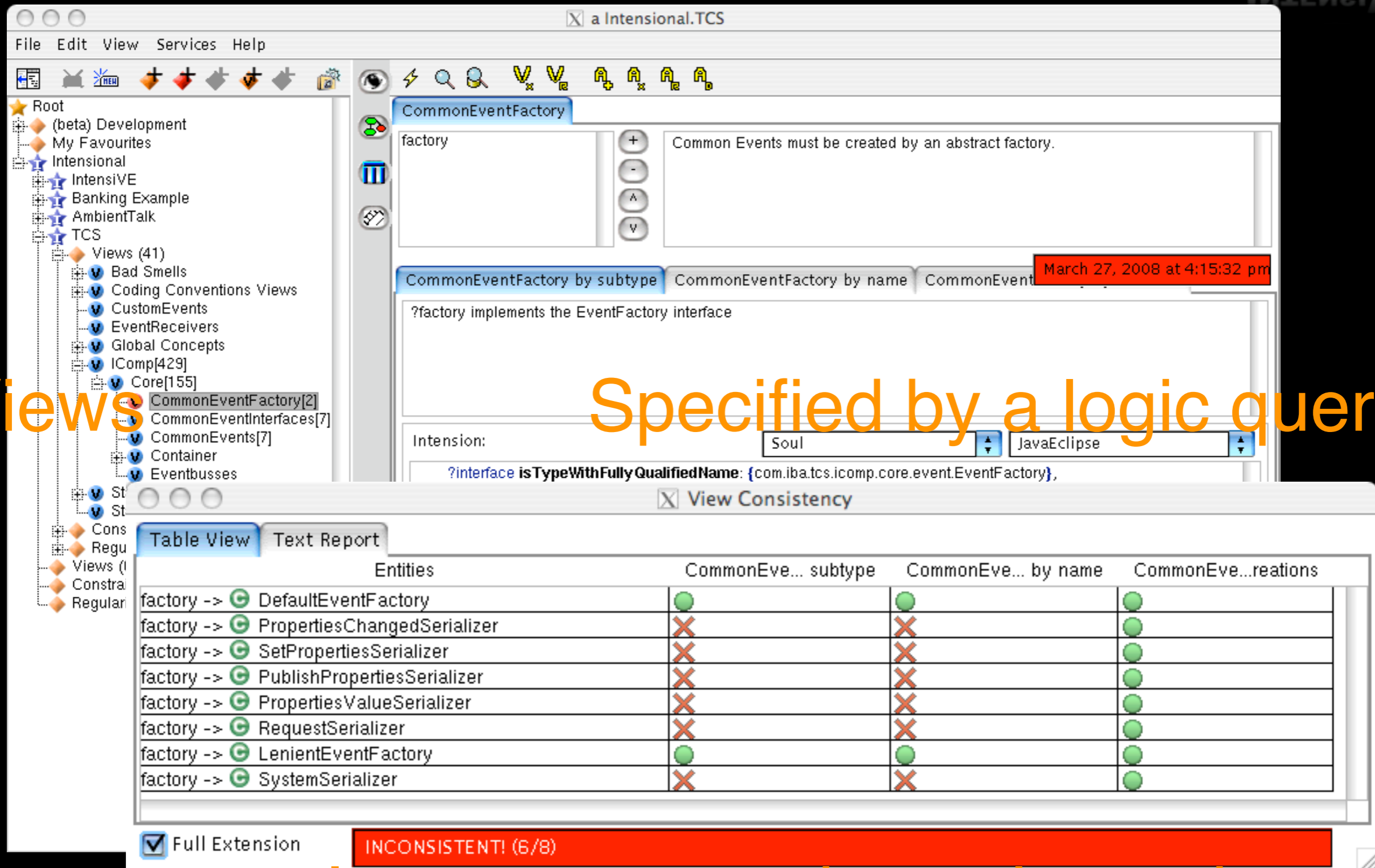
Soul JavaEclipse

?interface isTypeWithFullyQualifiedName: {com.iba.tcs.icomp.core.event.EventFactory},  
?cu hasClassDeclaration: ?factory,  
?factory inInterfaceHierarchyOfType: ?interface

Includes Excludes

+ -

Views Specified by a logic query



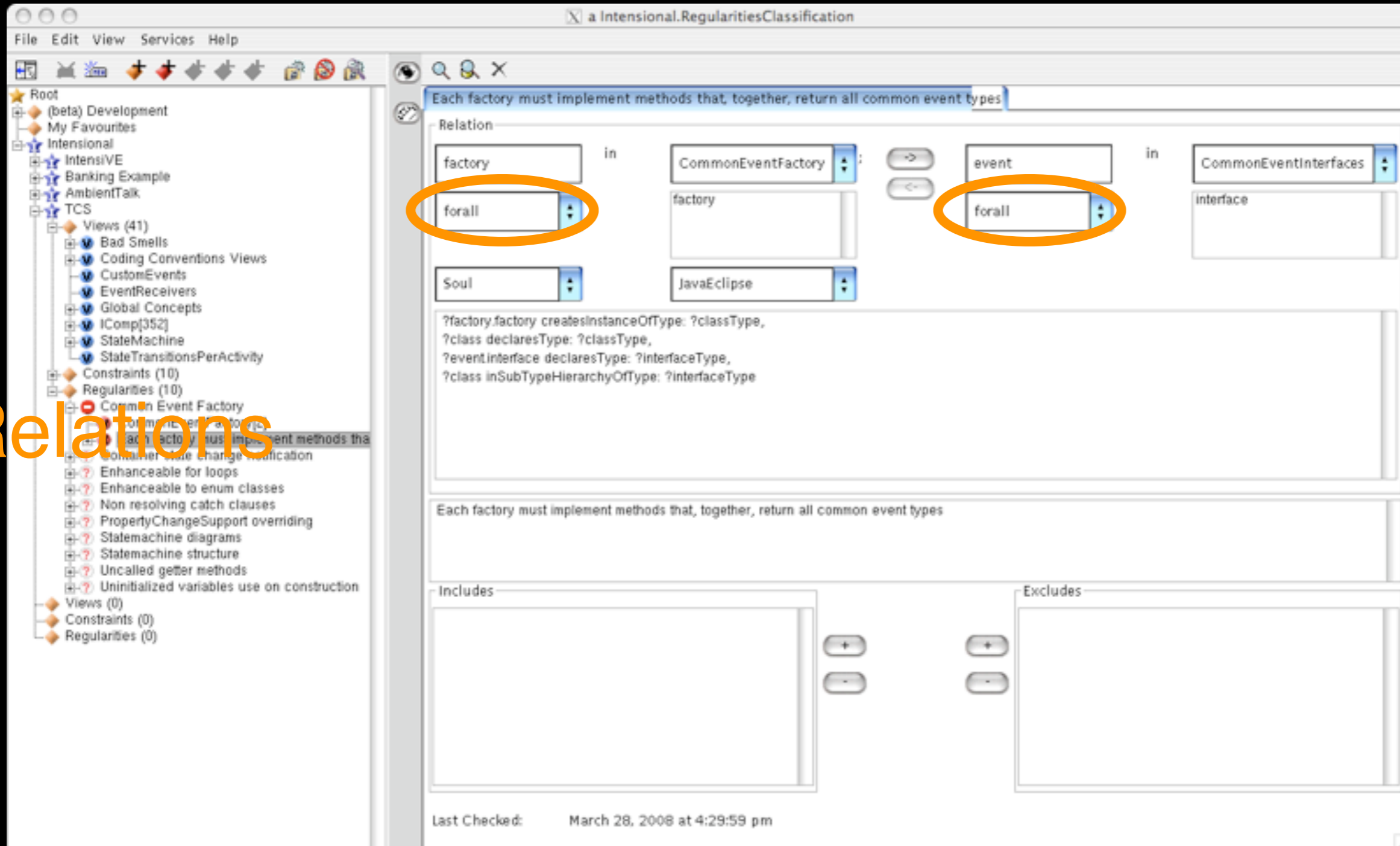
The screenshot shows the IntensiVE IDE interface. On the left is a project tree with 'Intensional' and 'TCS' folders. The main editor displays a logic query for 'CommonEventFactory' with a text area containing the query: '?factory implements the EventFactory interface'. Below the query, there are dropdown menus for 'Intension:' (set to 'Soul') and 'JavaEclipse'. A date stamp 'March 27, 2008 at 4:15:32 pm' is visible. In the foreground, a 'View Consistency' window is open, showing a table with columns for 'Entities', 'CommonEve... subtype', 'CommonEve... by name', and 'CommonEve...reations'. The table lists various factory classes and their consistency status. A red bar at the bottom of the consistency window reads 'INCONSISTENT! (6/8)'. A 'Full Extension' checkbox is checked.

Entities	CommonEve... subtype	CommonEve... by name	CommonEve...reations
factory -> DefaultEventFactory	●	●	●
factory -> PropertiesChangedSerializer	✗	✗	●
factory -> SetPropertySerializer	✗	✗	●
factory -> PublishPropertiesSerializer	✗	✗	●
factory -> PropertiesValueSerializer	✗	✗	●
factory -> RequestSerializer	✗	✗	●
factory -> LenientEventFactory	●	●	●
factory -> SystemSerializer	✗	✗	●

Views

Specified by a logic query

Impose constraints: alternatives



File Edit View Services Help

a Intensional.RegularitiesClassification

Each factory must implement methods that, together, return all common event types

Relation

factory in CommonEventFactory : event in CommonEventInterfaces

forall

forall

Soul JavaEclipse

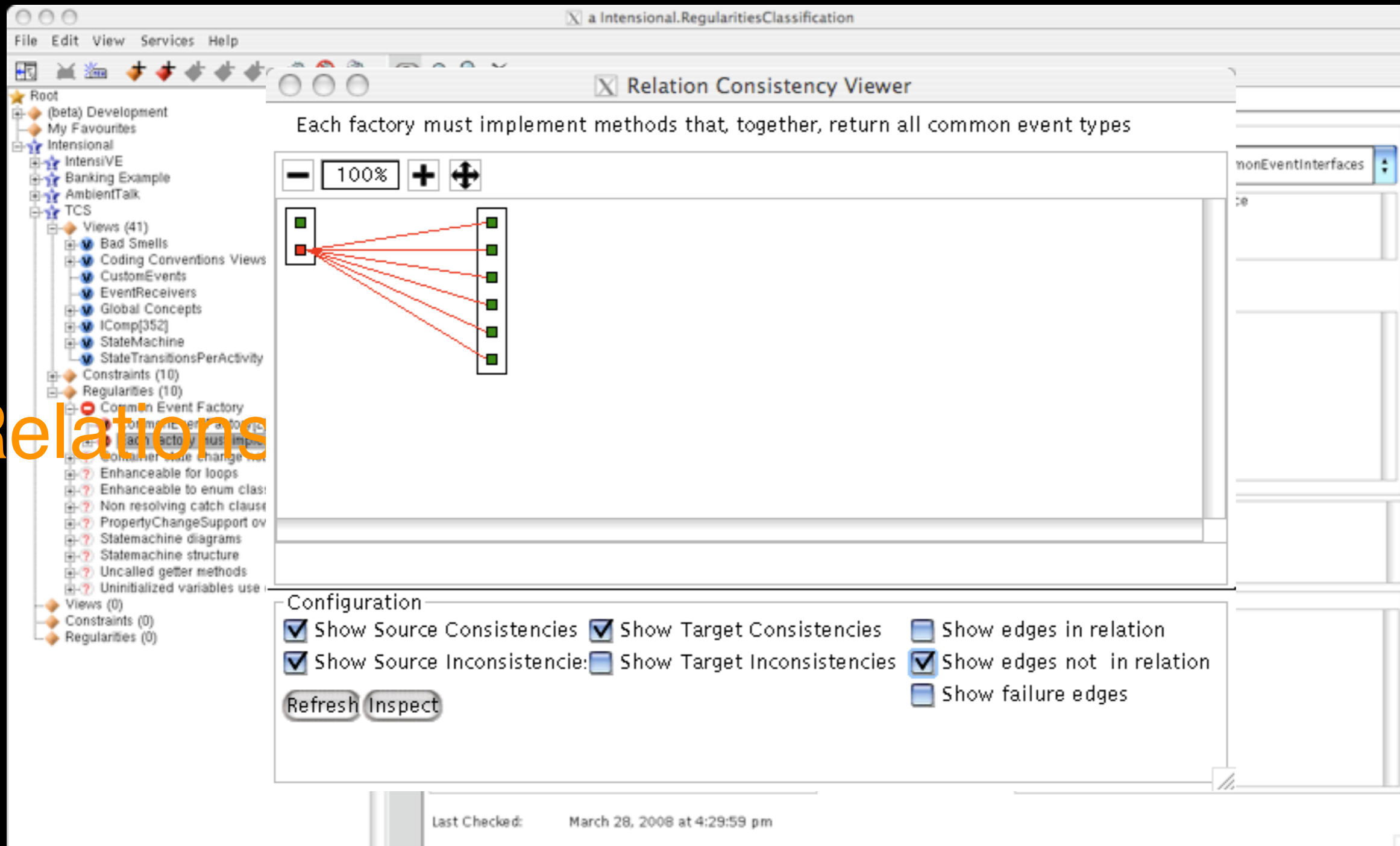
?factory.factory createsInstanceOfType: ?classType,  
?class declaresType: ?classType,  
?event.interface declaresType: ?interfaceType,  
?class inSubTypeHierarchyOfType: ?interfaceType

Each factory must implement methods that, together, return all common event types

Includes Excludes

Last Checked: March 28, 2008 at 4:29:59 pm

Relations



Each factory must implement methods that, together, return all common event types

100%

Configuration

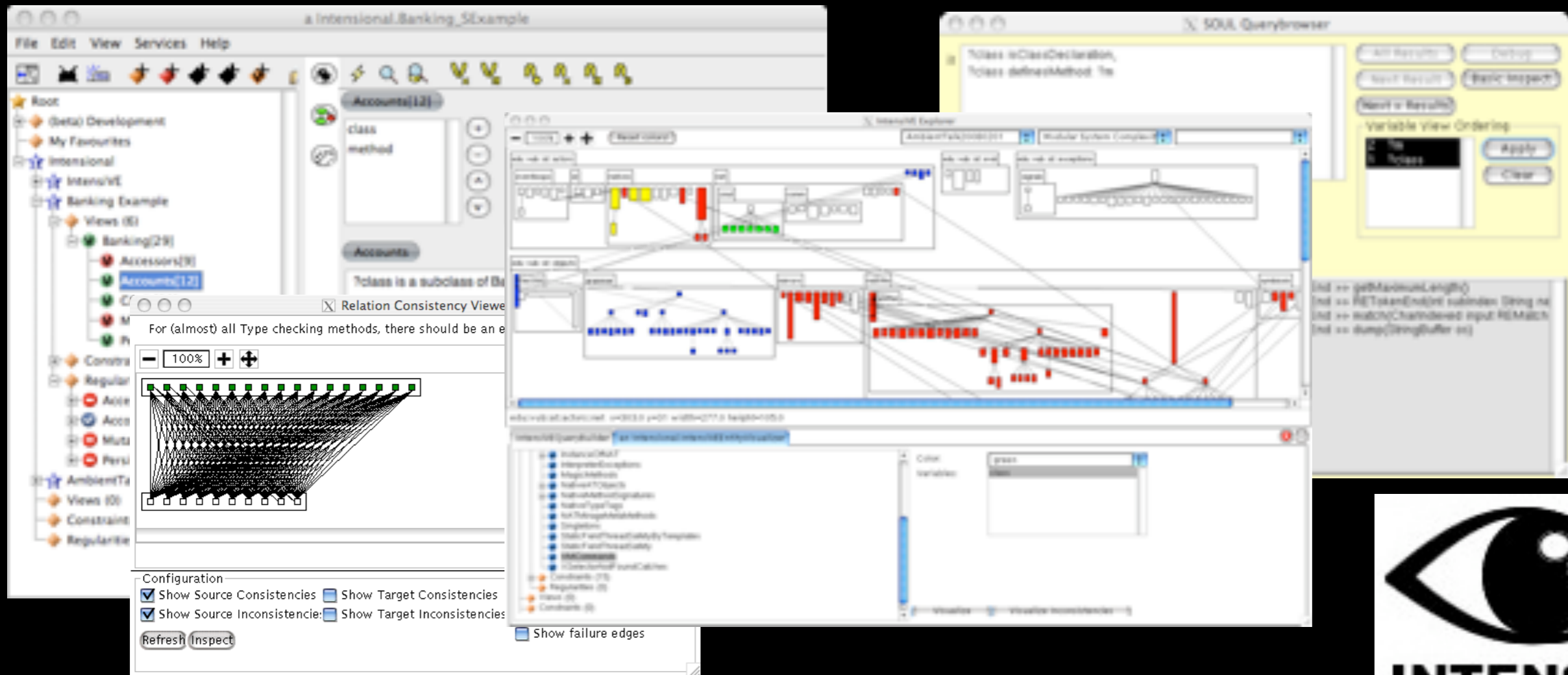
- Show Source Consistencies
- Show Target Consistencies
- Show edges in relation
- Show Source Inconsistencies
- Show Target Inconsistencies
- Show edges not in relation
- Show failure edges

Refresh Inspect

Last Checked: March 28, 2008 at 4:29:59 pm

Relations

## The Intensional Views Environment



The screenshot displays the IntensiVE IDE interface. On the left is a project browser showing a tree structure with folders like 'Intensional' and 'Banking Example'. The main workspace is divided into several panes. The top center pane shows a class browser with a diagram of class relationships and methods. The bottom center pane shows a list of classes and their attributes. The foreground pane is a 'Relation Consistency View' window with a title bar that says 'Relation Consistency View' and a close button. It contains a dense network graph with nodes and edges, a zoom control set to 100%, and a configuration panel at the bottom with checkboxes for 'Show Source Consistencies', 'Show Source Inconsistencies', 'Show Target Consistencies', and 'Show Target Inconsistencies', along with 'Refresh' and 'Inspect' buttons. To the right, the 'SOA Querybrowser' window is visible, showing a query result table and a 'Variable View Ordering' section with 'Apply' and 'Clear' buttons.



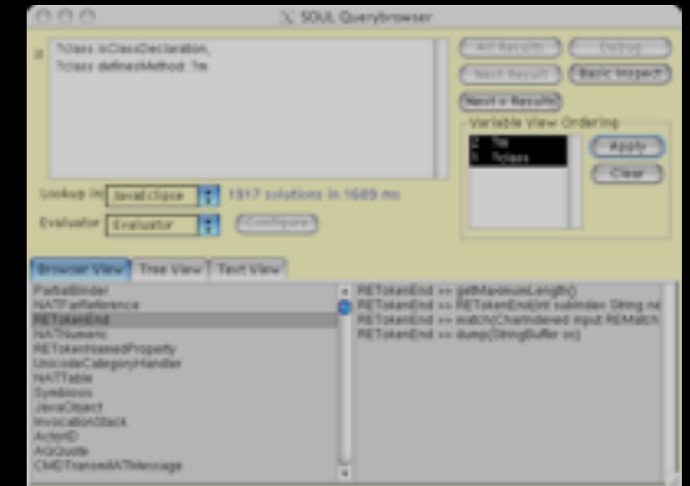
# About IntensiVE's implementation



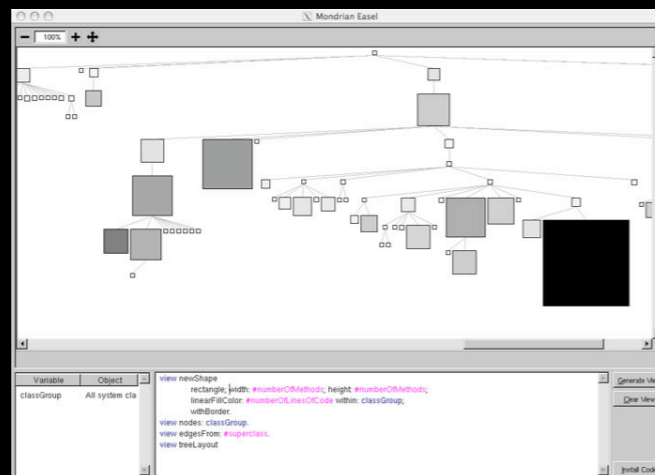
## VisualWorks Smalltalk



## Based upon the SOUL Language



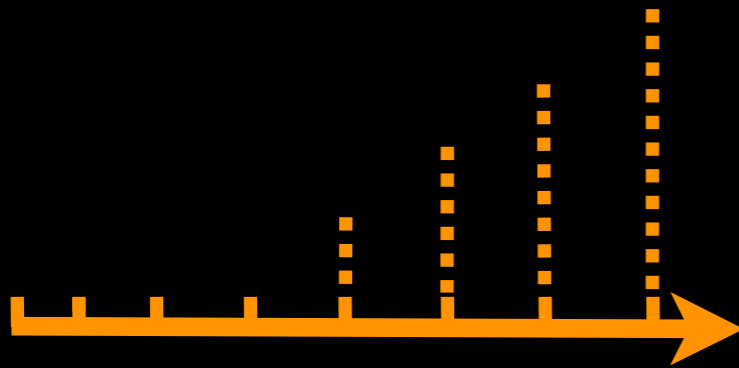
## Different technologies/components



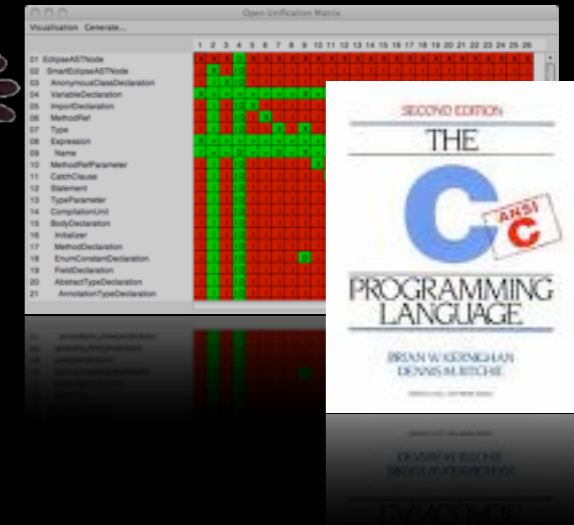
# Considerations



## Changing emphasis



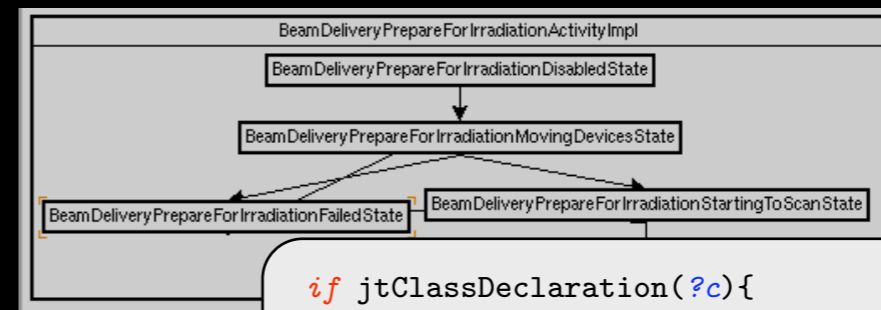
## Language independence



## Why Smalltalk?

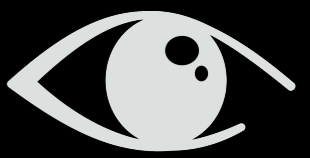


## Customized Extensions (Framework)

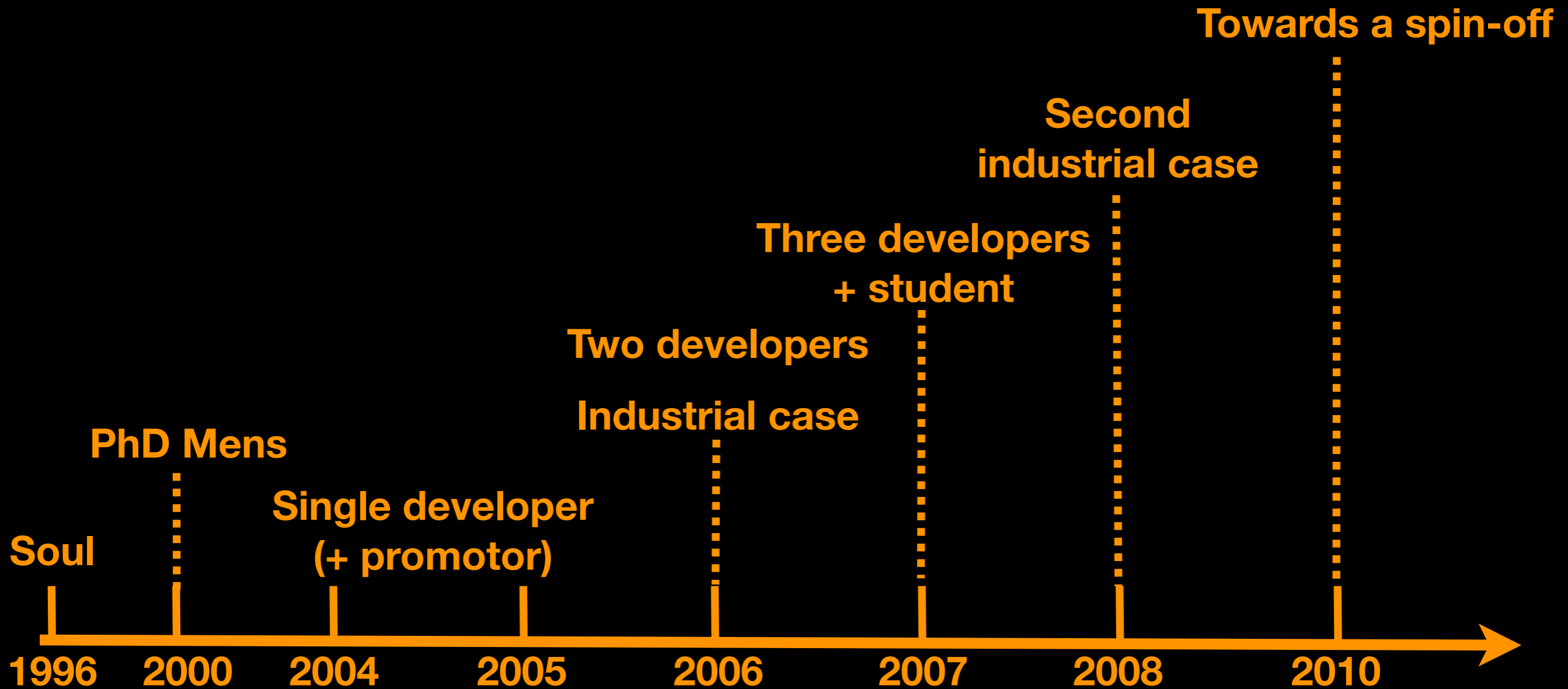


```
if jtClassDeclaration(?c){  
  class ?c {  
    private ?type ?field;  
    public ?type ?name() { return ?field; }  
  }  
}
```

# Time line



INTENSIVE  
INTENSIVE

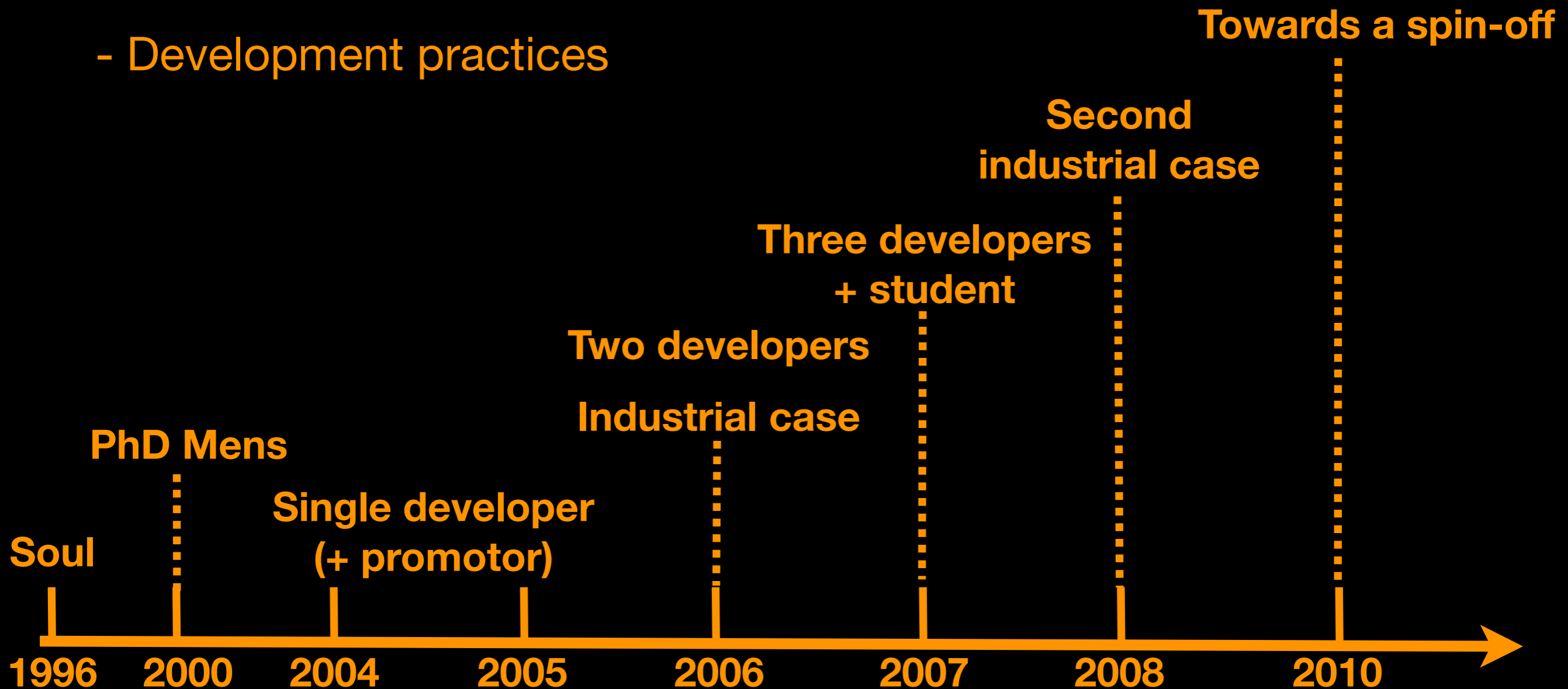


# Time line



## ► In order to cope with increasing complexity

- Guidelines
- Architecture
- Development practices



## ▶ **Dynamic typing**

- Easy to support various source-code models
- Rapid prototyping
  - try out new ideas with the least amount of effort
- Open environment
- Extensive meta-object protocol
- Ease of integration

Lots of different technologies: StarBrowser, JavaConnect, Mondrian, ...

## ▶ **Facilitates language independence**

- Java and Smalltalk
- C(++) and Cobol
- Each kind of object can belong to an intensional view



# Language independence



Open Unification Matrix

Visualisation Generate...

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
01 EclipseASTNode	X	X	X	D	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
02 SmartEclipseASTNode	I	X	I	D	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
03 AnonymousClassDeclaration	I	I	X	D	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
04 VariableDeclaration	X	=	=	=	=	=	=	=	X	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
05 ImportDeclaration	I	I	I	D	X	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
06 MethodRef	I	I	I	D	I	X	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
07 Type	I	I	I	D	I	I	X	I	X	I	I	I	I	I	I	I	I	I	I	I	I	I	X	I	I	D
08 Expression	X	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
09 Name	I	=	=	D	=	=	D	=	X	=	=	=	=	=	=	=	=	X	=	=	=	X	X	=	=	D
10 MethodRefParameter	I	I	I	D	I	I	I	I	I	X	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
11 CatchClause	I	I	I	D	I	I	I	I	I	I	X	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
12 Statement	I	I	I	D	I	I	I	I	I	I	I	X	I	I	I	I	I	I	I	I	I	I	I	I	I	I
13 TypeParameter	I	I	I	D	I	I	I	I	I	I	I	I	X	I	I	I	I	I	I	I	I	I	I	I	I	I
14 CompilationUnit	I	I	I	D	I	I	I	I	I	I	I	I	I	X	I	I	I	I	I	I	I	I	I	I	I	I
15 BodyDeclaration	I	I	I	D	I	I	I	I	I	I	I	I	I	I	X	I	I	I	I	I	I	I	I	I	I	I
16 Initializer	I	I	I	D	I	I	I	I	I	I	I	I	I	I	I	X	I	I	I	I	I	I	I	I	I	I
17 MethodDeclaration	I	I	I	D	I	I	I	I	I	I	I	I	I	I	I	I	X	I	I	I	I	I	I	I	I	I
18 EnumConstantDeclaration	I	I	I	D	I	I	I	I	D	I	I	I	I	I	I	I	I	X	I	I	I	I	I	I	I	I
19 FieldDeclaration	I	I	I	D	I	I	I	I	I	I	I	I	I	I	I	I	I	I	X	I	I	I	I	I	I	I
20 AbstractTypeDeclaration	I	I	I	D	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	X	I	I	I	I	I
21 AnnotationTypeDeclaration	I	I	I	D	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	X	I	I	I	I
22 PackageDeclaration	I	I	I	D	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	X	I	I	I
23 TypeDeclaration	I	I	I	D	I	I	D	I	D	I	I	I	I	I	I	I	I	I	I	I	I	I	I	X	I	D
24 EnumDeclaration	I	I	I	D	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	X	I	I
25 AnnotationTypeMemberDeclaration	I	I	I	D	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	X	I
26 TypeBinding	I	I	I	D	I	I	X	I	X	I	I	I	I	I	I	I	I	I	I	I	I	I	I	X	I	X

## ▶ Extensible implementation

- Design patterns
- OO best practices

## ▶ Example extensions

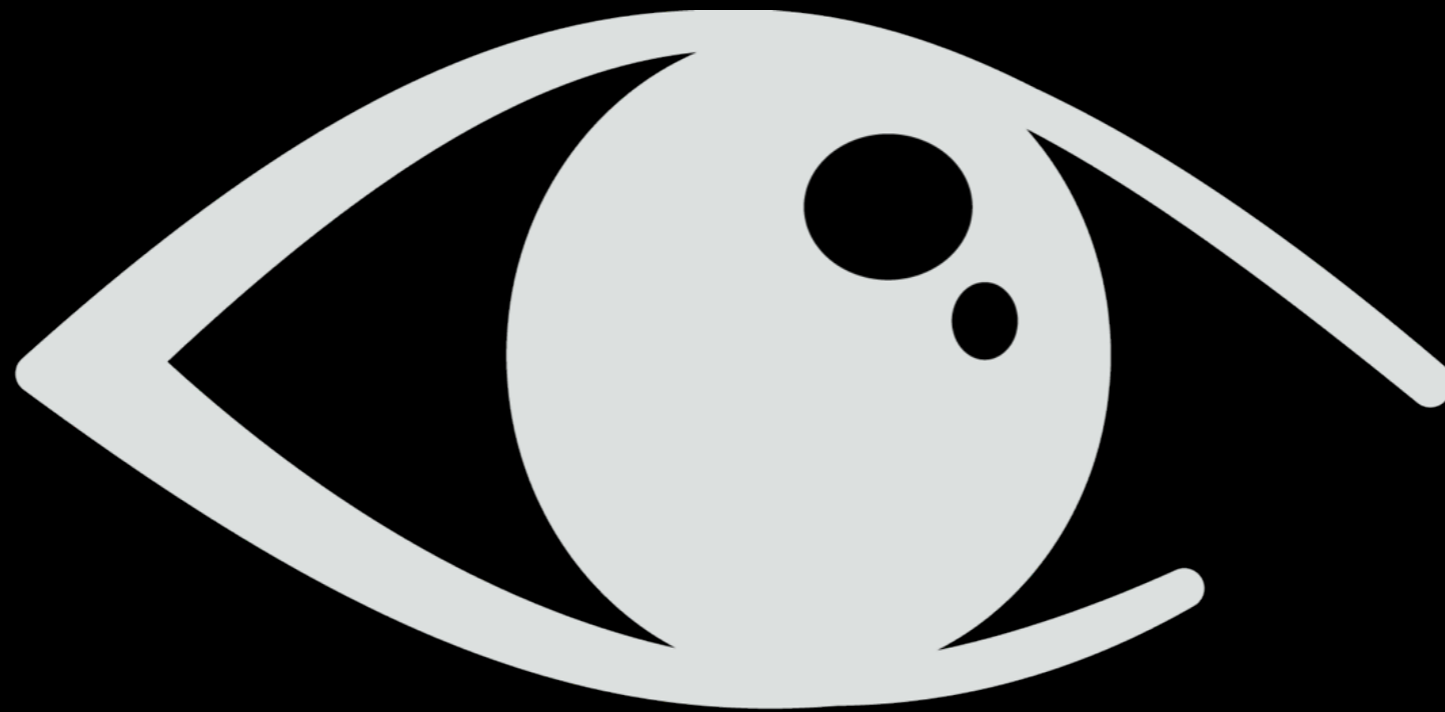
- State diagram extraction
- Integration with Moose/Mondrian
- Different query languages
- Different flavours of intensional views (e.g. Fuzzy)

# Eat your own dogfood!



We are our own customer

<http://www.intensional.be>



**INTENSIVE**

**INTENSIVE**