

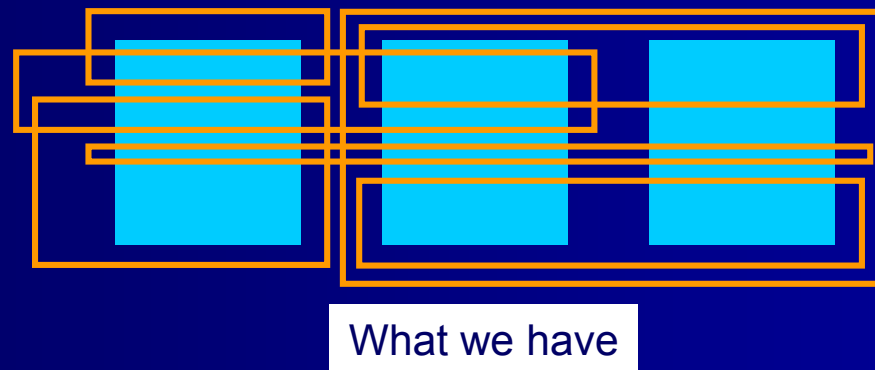
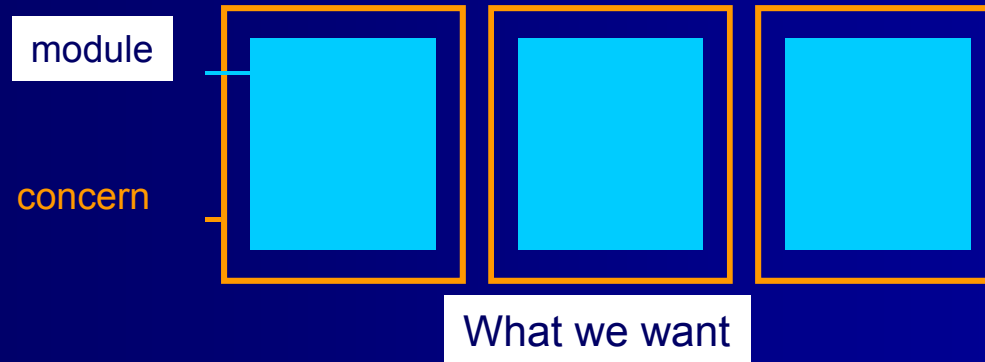
COMP 400 Technical Project

Extendible ConcernMapper Plug-in

Mohammad Usman Ahmed

Supervised by Martin Robillard

A case for ConcernMapper



Mohammad Usman Ahmed
Supervised by Martin Robillard

The ConcernMapper Application Domain

- Support software development involving scattered concerns



- Support advanced SOC research
 - Aspect mining
 - Feature location
 - Re-factoring
 - Software investigation

Tracking concerns with ConcernMapper

The screenshot displays the Eclipse IDE with several windows. The **ConcernMapper** window is highlighted with an orange border and shows a tree view of concerns. The **Package Explorer** window shows the project structure, and the **Class Explorer** window shows the methods of a class. Red arrows and circles highlight the mapping of concerns to code elements.

ConcernMapper View:

- Autosave feature
 - Autosave
 - autosaveFile
 - autosave()
 - getAutosaveFile()
 - getFlag(int)
 - recoverAutosave(View)
 - Buffer
 - autosaveFile
 - autosave()
 - getAutosaveFile()
 - getFlag(int)
 - recoverAutosave(View)
 - jEdit
 - propertiesChanged()
 - LoadSaveOptionPane
 - autosave
 - _init()
 - _save()

Package Explorer View:

- ActionListHandler.java
- ActionSet.java
- Autosave.java
 - Autosave
 - timer

Class Explorer View:

- addOrRemoveMarker(char, int)
- autosave()
- beginCompoundEdit()
- checkFileStatus()
- checkModTime(View)
- close()
- commitTemporary()
- contentInserted(int, int, IntegerArray)
- createPosition(int)
- endCompoundEdit()
- finishLoading()
- finishSaving(View, String, String, boolean, ...)
- fireContentInserted(int, int, int, int)
- fireContentRemoved(int, int, int, int)
- fireFoldLevelChanged(int, int)
- fireTransactionComplete()
- getAutosaveFile()
- getBooleanProperty(String)
- getBufferChangeListeners()
- getContextSensitiveProperty(int, String)
- getDefaultRootElement()
- getFile()
- getFlag(int)
- getOldAtLine(int)
- getFoldLevel(int)

Goal: to make the concern models easy to create both, manually and programmatically

Plug-in Development - AttributeFigure.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Plug-ins Hierarchy

CH.ifa.draw.figures

- AbstractLineDecoration.java
- ArrowTip.java
- AttributeFigure.java
 - AttributeFigure
 - fgDefaultAttributes
 - serialVersionUID
 - getDefaultAttribute(String)
 - initializeAttributes()
 - attributeFigureSerializedDataVersion
 - fAttributes [Attribute Manager]
 - AttributeFigure()
 - draw(Graphics)
 - drawBackground(Graphics)
 - drawFrame(Graphics)

Elements in concerns are highlighted in the JDT views. Preferences allow to customize highlighting settings.

Add fields and methods to concerns by dragging them directly from most JDT views onto the appropriate concern.

return getDefaultAttr

the named attribute to the new value

```
public void setAttribute(String name, Object value)
{
    if (fAttributes == null)
        fAttributes = new FigureAttributes();
    fAttributes.set(name, value);
    changed();
}
```

Save concern models to files using the toolbar buttons.

Load a saved concern model either by selecting "Load Concerns" from the file's context menu, or simply by dropping the concern model file onto the ConcernMapper view.

Create new concerns

ConcernMapper

- Attribute Management
 - AttributeFigure
 - fAttributes
 - getAttribute(String)
 - setAttribute(String, Object)
 - FigureAttributes
 - FigureAttributes()
 - Command Design Pattern
 - AbstractCommand
 - execute()
 - Command
 - CommandButton
 - actionPerformed(ActionEvent)

The ConcernMapper view can contain different concerns.

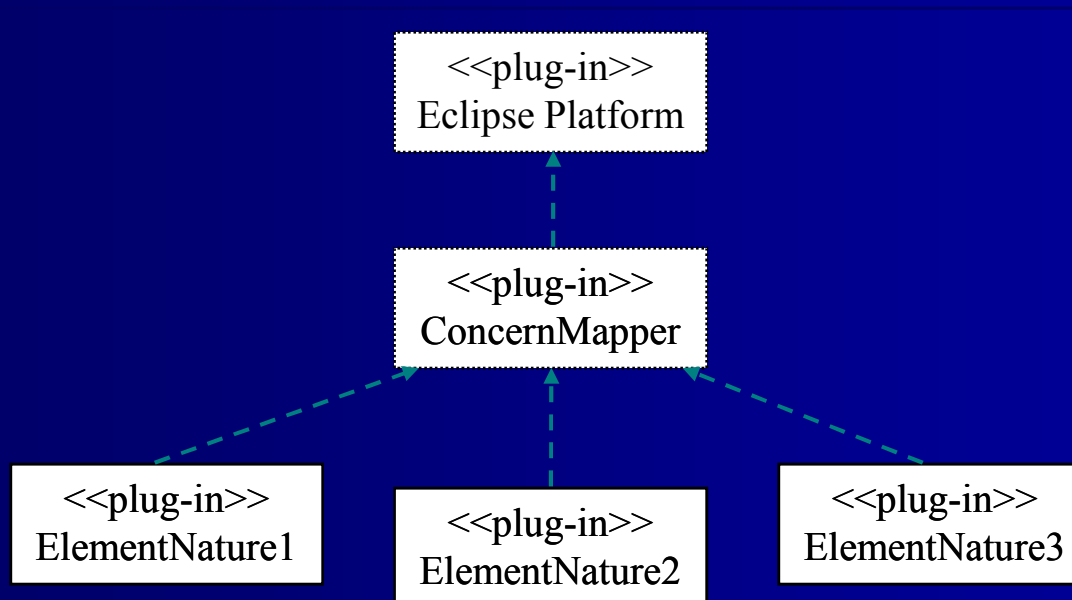
Quantify to which extent each element belongs to a concern (between 0 and 100), and to filter out elements whose membership certainty is below a certain threshold.

Filter matched 1 of 43 items

Description	Resource	In Folder	Location
ODO : test FastBufferedUpdateStr...	StandardD...	JHotDraw5.3/src/CH/ifa/d...	line 142

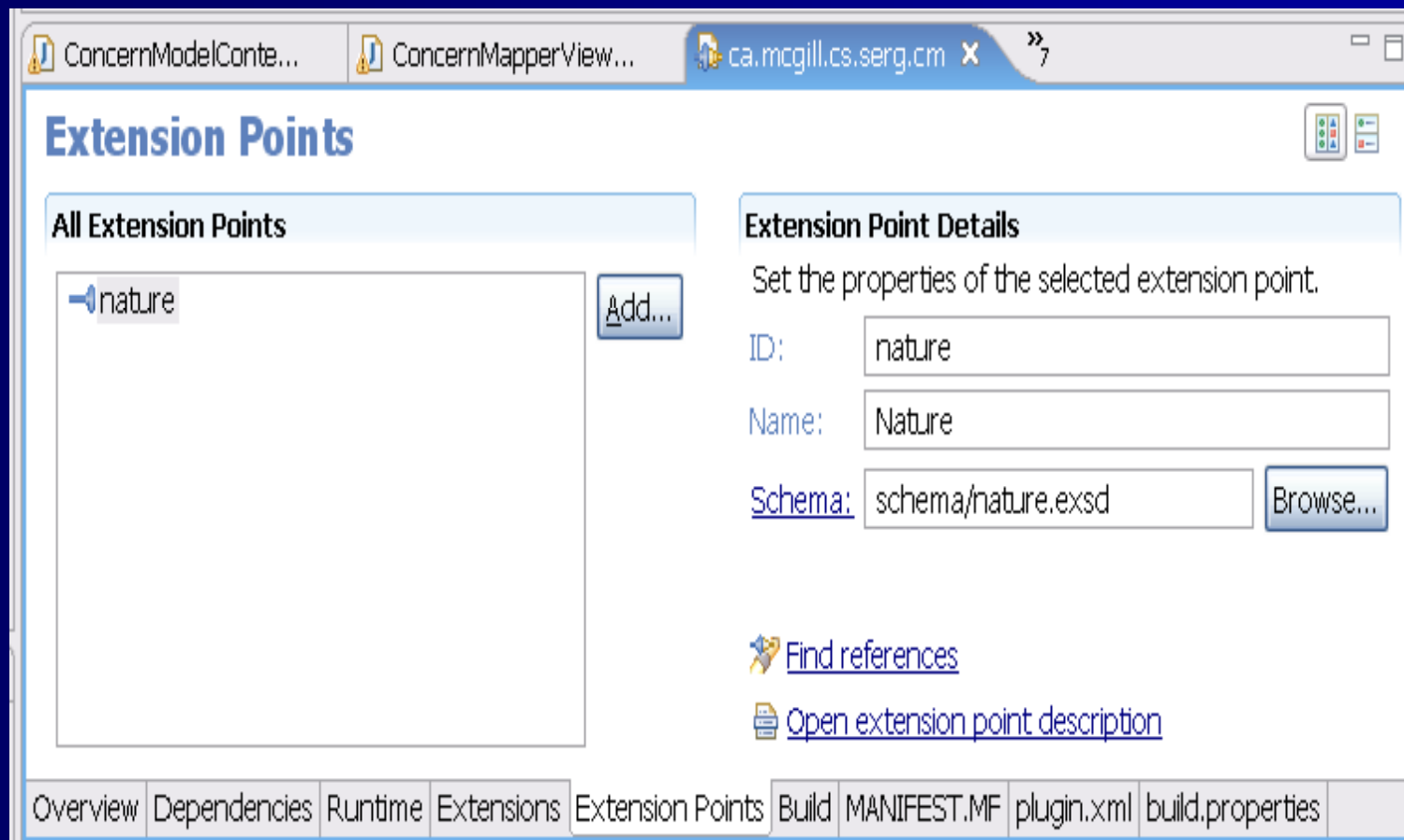
The new plug-in extension architecture

- One generic ConcernMapper plug-in
- Multiple nature extensions



Mohammad Usman Ahmed
Supervised by Martin Robillard

Solution: Creation of the Extension-Point



Solution: Creation of the Extension-Point

The screenshot shows the ConcernMapperView application with the 'Nature' concern selected. The interface is divided into two main panels: 'Extension Point Elements' and 'Compositor Details'.

Extension Point Elements

The following XML elements and attributes are allowed in this extension point:

- extension**
 - Sequence**
 - nature (1 - *)**
 - sorter**
 - labelProvider**
 - point**
 - id**
 - name**
 - nature**
 - id**
 - name**
 - class**
 - sorter**
 - id**
 - name**
 - class**
 - labelProvider**

Buttons: **New Element**, **New Attribute**

Compositor Details

Properties for the Sequence compositor.

Min Occurences: 1

Max Occurences: 1 ☐ Unbounded

Type: sequence

DTD approximation:
(nature+ , sorter , labelProvider)

Solution: Creation of the Extension-Point

The screenshot displays the Eclipse IDE's 'Extensions' dialog box. The 'All Extensions' list on the left contains the following items:

- org.eclipse.ui.views
- org.eclipse.ui.perspectiveExtensions
- org.eclipse.ui.popupMenus
 - ca.mcgill.cs.serg.cm.popup (objectContributor)
 - ca.mcgill.cs.serg.cm.addToConcernObjectCon
 - ca.mcgill.cs.serg.cm.addToConcernEditorCont
 - ca.mcgill.cs.serg.cm.addToConcernObject2Co
 - ca.mcgill.cs.serg.cm.addToConcernObject2Co
 - Add to Concern (action)
- org.eclipse.ui.decorators
- org.eclipse.ui.preferencePages
- org.eclipse.help.toc
- org.eclipse.core.runtime.preferences
- ca.mcgill.cs.serg.cm.nature
 - ca.mcgill.cs.serg.cm.nature.jfield (nature)
 - ca.mcgill.cs.serg.cm.nature.jmethod (nature)
 - ca.mcgill.cs.serg.cm.nature.file (nature)
 - ca.mcgill.cs.serg.cm.nature.sorter (sorter)
 - ca.mcgill.cs.serg.cm.nature.labelProvider (labelProvider)

The 'Extension Element Details' panel on the right shows the configuration for the 'nature' extension:

Set the properties of "nature"

id*: ca.mcgill.cs.serg.cm.nature.file

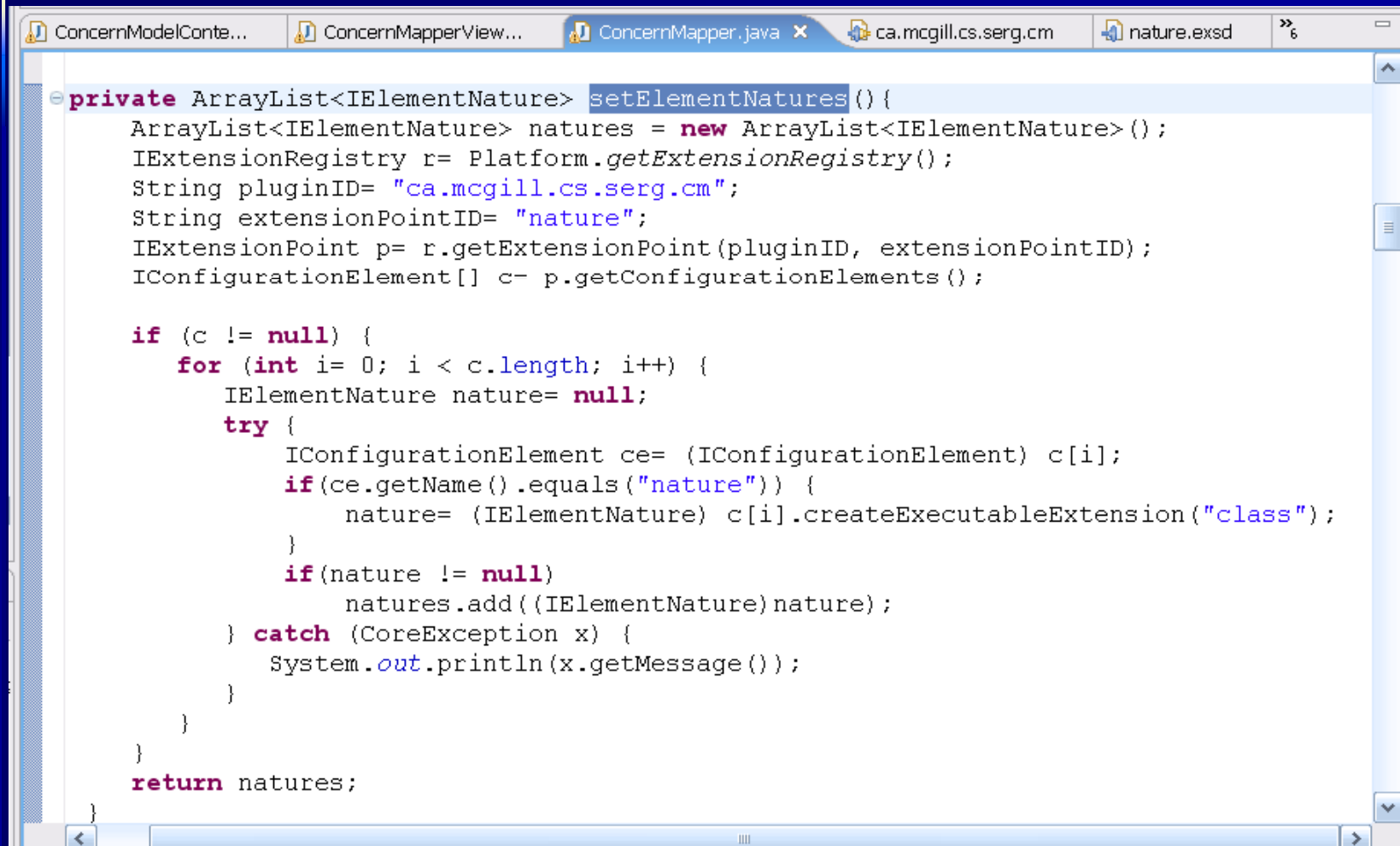
name*: ca.mcgill.cs.serg.cm.nature.file

class*: cm.nature.ResourceFile [Browse...](#)

The bottom of the dialog shows a 'Body Text' section and a tabbed interface with the following tabs: Overview, Dependencies, Runtime, Extensions, Extension Points, Build, MANIFEST.MF, plugin.xml, build.properties.

Extension Processing in Eclipse

A nature callback object is retrieved from the extension



```
private ArrayList<IElementNature> setElementNatures() {
    ArrayList<IElementNature> natures = new ArrayList<IElementNature>();
    IExtensionRegistry r= Platform.getExtensionRegistry();
    String pluginID= "ca.mcgill.cs.serg.cm";
    String extensionPointID= "nature";
    IExtensionPoint p= r.getExtensionPoint(pluginID, extensionPointID);
    IConfigurationElement[] c= p.getConfigurationElements();

    if (c != null) {
        for (int i= 0; i < c.length; i++) {
            IElementNature nature= null;
            try {
                IConfigurationElement ce= (IConfigurationElement) c[i];
                if(ce.getName().equals("nature")) {
                    nature= (IElementNature) c[i].createExecutableExtension("class");
                }
                if(nature != null)
                    natures.add((IElementNature) nature);
            } catch (CoreException x) {
                System.out.println(x.getMessage());
            }
        }
    }
    return natures;
}
```

Example: Using ConcernMapper in tracking this project's changes

The screenshot displays the Eclipse IDE interface for the 'Plug-in Development - ConcernModelContentProvider.java - Eclipse SDK' project. The Package Explorer on the left shows the project structure, including the 'src' folder and the 'ca.mcgill.cs.serg.cm' package. The Outline view on the right shows the 'buildDynamicStructure() [aView]' method. The main editor window displays the implementation of the 'buildDynamicStructure()' method in the 'ConcernModelContentProvider' class. A context menu is open over the 'buildDynamicStructure()' method in the Package Explorer, showing options like 'References', 'Declarations', 'Delete', 'Workspace', 'Project', 'Hierarchy', and 'Working Set...'. The bottom status bar shows the file path 'ca.mcgill.cs.serg.cm.views...' and the text 'Read Me Trim (Bottom)'.

Plug-in Development - ConcernModelContentProvider.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run FieldAssist Window Help

Package Explorer Hierarchy

src

ca.mcgill.cs.serg.cm

actions

decorators

ConcernMapper

ConcernModelContentProvider

addMap(Map<Object, IConcernMapperViewNode>, Object, ConcernMapperViewNode)

buildDynamicStructure()

References

Declarations

Delete

Workspace Ctrl+G

Project

Hierarchy

Working Set...

Outline

getElements(Object)

buildDynamicStructure() [aView]

getChildren(Object)

addMap(Map<Object, IConcernMapperViewNode>, Object, ConcernMapperViewNode)

getParent(Object)

hasChildren(Object)

return lReturn;

private Object[] buildDynamicStructure()

List<ConcernNode> lReturn = new ArrayList<>();

for (String lConcernName : aConcernModel.getConcernNames())

{

Map<Object, IConcernMapperViewNode> lNodeMap = new HashMap<>();

ConcernNode lConcernNode = new ConcernNode(lConcernName, lNodeMap);

lConcernNode.setParent(aConcernModel);

lReturn.add(lConcernNode);

}

Error ... Tasks Probl... Call Hi... Console Search SVN R... Synch...

'ca.mcgill.cs.serg.cm.views.ConcernModelContentProvider.buildDynamicStructure'

ca.mcgill.cs.serg.cm.views - src - ca.mcgill.cs.serg.cm

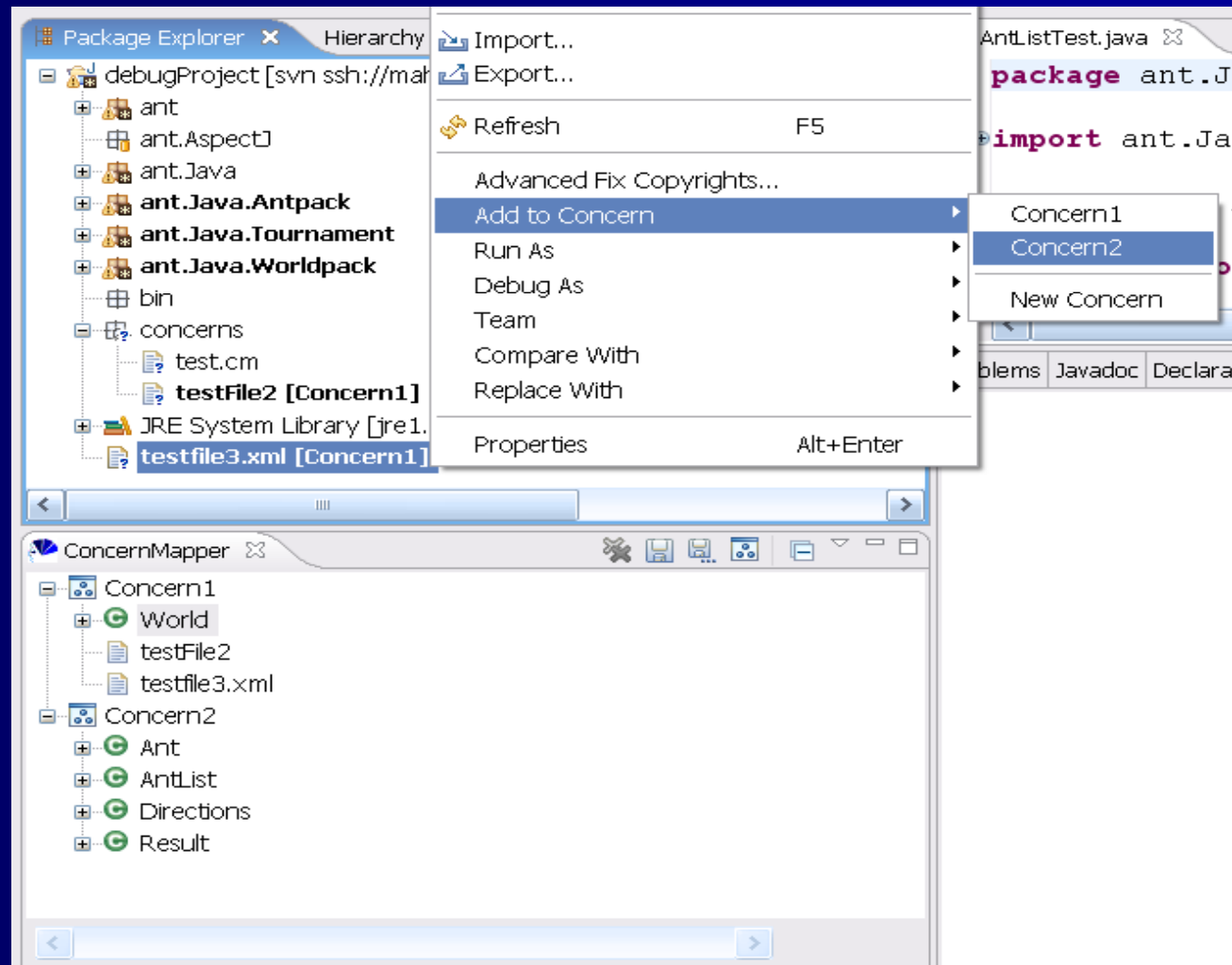
ConcernModelContentProvider 151 4/12/07 11:49 PM Usman

buildDynamicStructure() [aView]

ca.mcgill.cs.serg.cm.views... ca.mcgill.cs.serg.cm/src

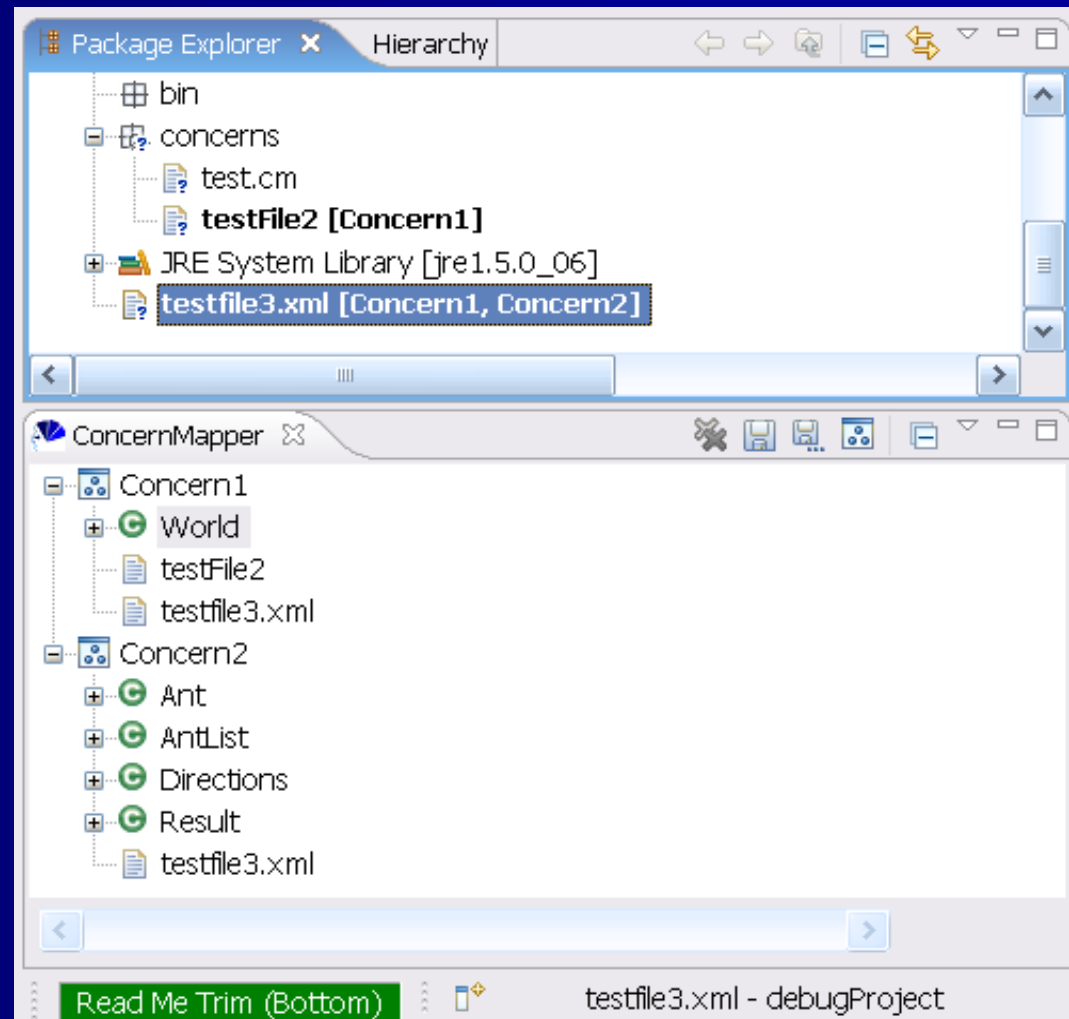
Read Me Trim (Bottom)

Resource files in the new plug-in



Resource files in the new plug-in

- XML file added to Concern2
- Can now be opened directly from the concern view



The architecture of ConcernMapper

