

Lecture 15

Refactoring Reconstruction

Today's Agenda

- Motivation for Refactoring Reconstruction
- Refactoring Reconstruction
 - UMLDiff: some slides borrowed from Zhenchang Xing (U.Alberta)

Today's Agenda

- Synthesis of Refactoring Reconstruction Techniques
- API Evolution Support
- Bug Cache (MSR Part II)

Motivation for Reconstructing Refactorings from Two Versions

Motivation for Reconstructing Refactorings from Two Versions

Motivation for Reconstructing Refactorings from Two Versions

I. Detecting Possible Sources of Errors

- Incomplete refactorings can be sources of errors
- e.g. `BarChart.draw()` and `PieChart.draw()` override `Chart.draw()`
- e.g. `Chart.draw()` and `PieChart.draw()` were renamed to `Chart.paint()` and `PieChart.paint()` but not `BarChart.draw()`.

Motivation for Reconstructing Refactorings from Two Versions

2. Capturing Intent of Changes

- Better empirical studies of code changes
- Reduce # of conflicts in version merging

Motivation for Reconstructing Refactorings from Two Versions

3. Capturing and Replaying Changes

- Automated update of client code: e.g. if a parameter was added to an API, then method invocations in program code using the API is automatically adapted.

4. Longer, continuous evolution history

- eRose system: when identifying related changes, inferred renamings can be used to combine rules of the previous instance and rules of the new instance

Motivation for Reconstructing Refactorings from Two Versions

5. Relation to Software Metrics

- Assess what kinds of refactorings increase what kinds of quality metrics

[Source: Identifying Refactorings from Source-Code Changes, Peter Weissgerber and Stephan Diehl ASE 2006]

Design Evolution Analysis in support of Evolutionary Software Development

Zhenchang Xing
University of Alberta

Supported by



Why is He Unhappy?

```
public class Course {
    private MonitorableQueue waitingList = new MonitorableQueue();

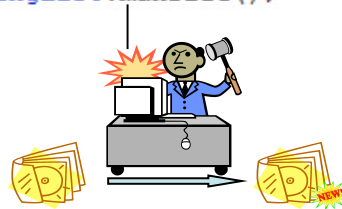
    private Queue backupTAs = new Queue();

    public void addToWaitingList(Collection waitingStudents) {
        waitingList.offerMany(waitingStudents);
    }

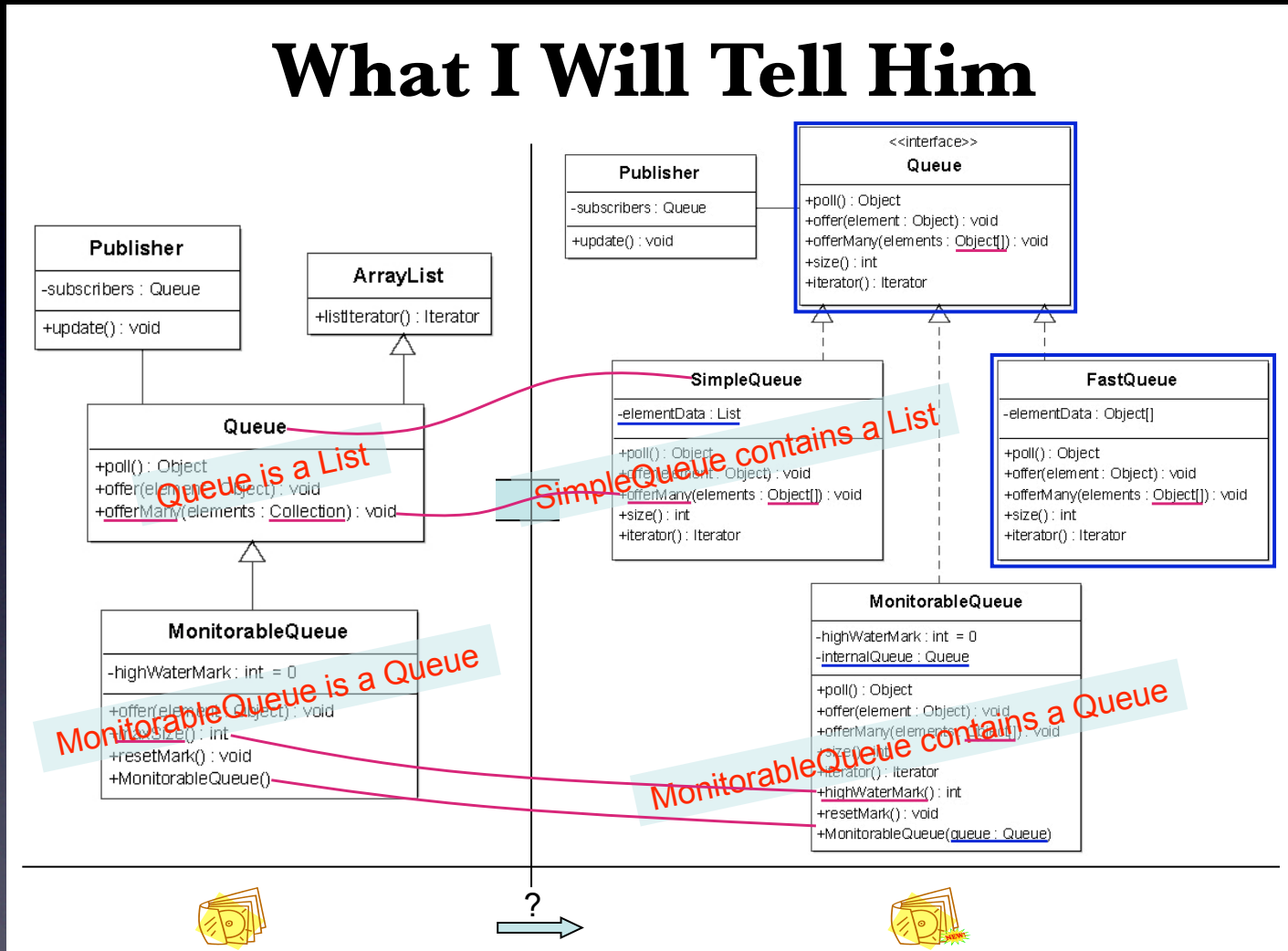
    public void enrolFromWaitingList(int howmany) {
        List list = this.waitingList;
    }

    public void notifyBackupTAs() {
        for(Iterator iterator = backupTAs.listIterator(); iterator.hasNext();) {
        }
    }

    public void reportEnrolmentStatistics() {
        int historyHigh = waitingList.maxSize();
    }
}
```



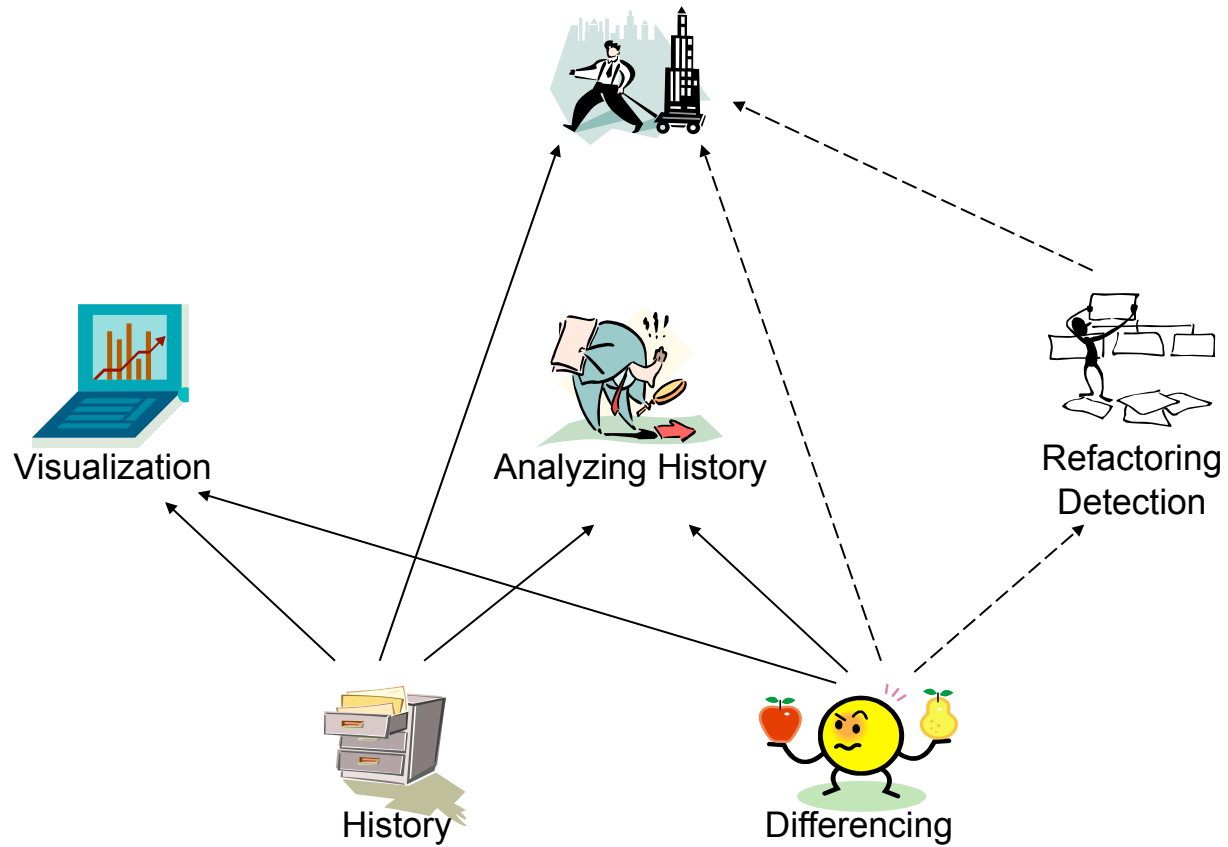
What I Will Tell Him



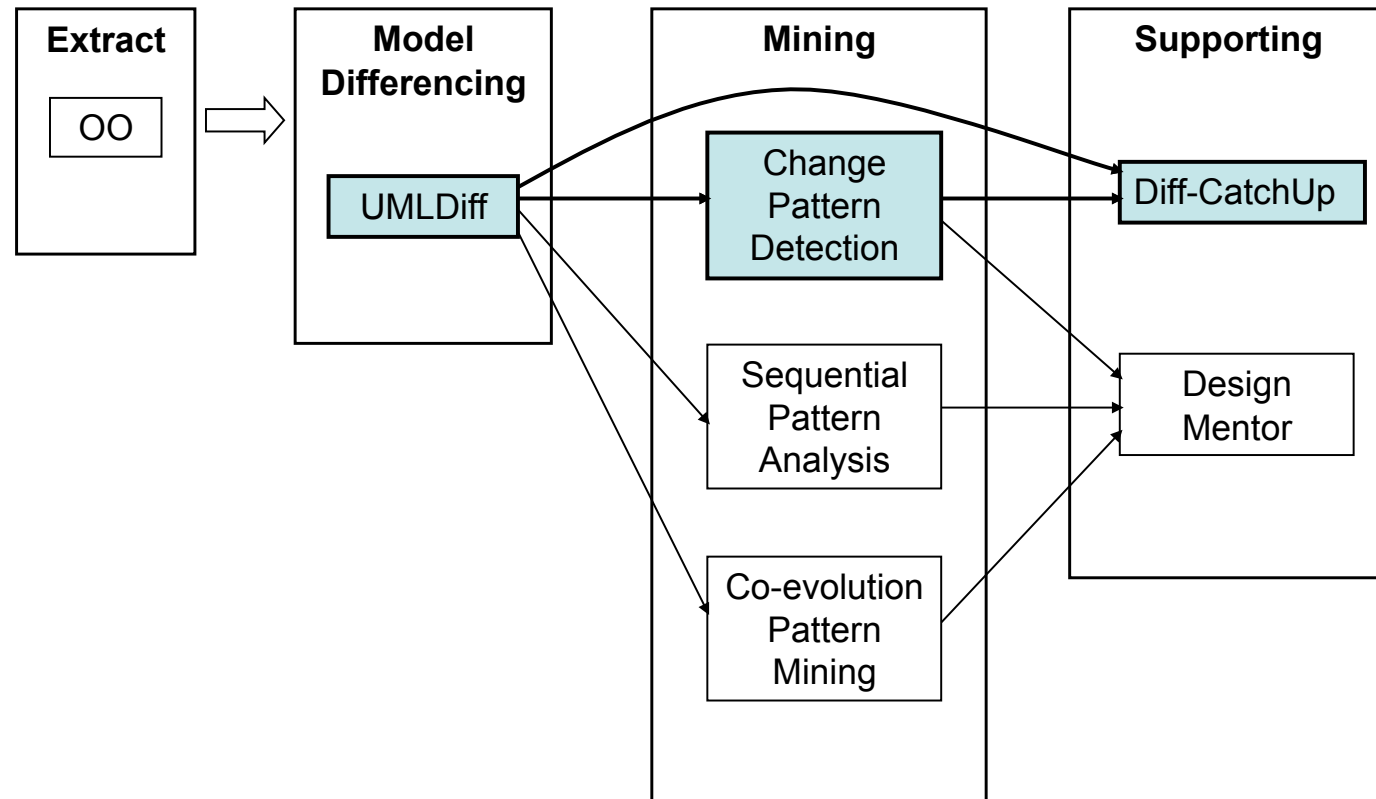
The Research Questions

- What exactly has been changed in the design context and how?
- Why has it been changed in the way it has?
- How can this information be used to support developers and in what tasks?

The World



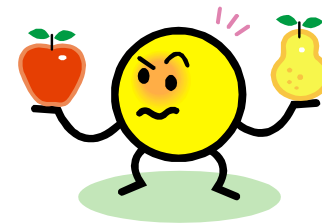
The Methodology



Model differencing with UMLDiff

Journal of Automated Software Engineering, 2007
The 20th ACM/IEEE International Conference on Automated Software Engineering, 2005

What exactly has been changed and How?



Heuristics in UMLDiff

- Additions and removals are easy
- Renamings are difficult
 - Lexical similarity of names and comments:
 - LCS, Adjacent pair
 - Structural similarity of relations
- Moves are even harder
 - The context from and to which elements are moved
 - Relationships: inheritance, containment, usage
 - Lexical and structural similarity of source and target contexts
 - The number of potential moves
- What if a set of elements are all renamed and/or moved?
 - Multiple rounds of renaming/move recognition

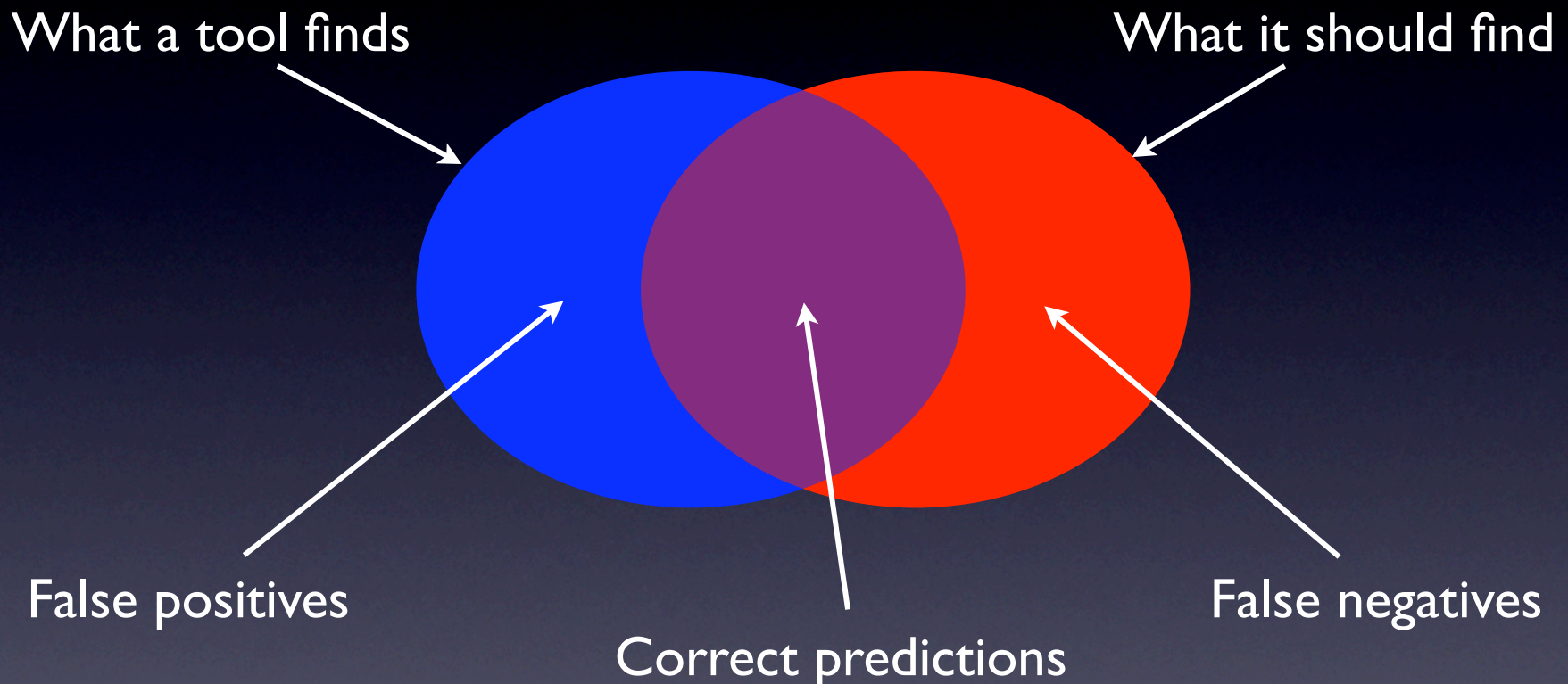
UMLDiff Process

- **Input:** Model_{before} and Model_{after}
- **UMLDiff** is a heuristic differencing algorithm
 1. Mapping model elements
 - Lexical and structural similarity
 2. Mapping relationships
 - The same relation type and the model elements they relate are mapped
 3. Recognizing extract/inline operations (not limited to class internals)
 - Usage dependency changes
 4. Compare attributes of mapped model elements
- **Output:** A set of elementary design change facts
 - Additions, removals, matches, renamings, moves of model elements
 - Extract and inline operations
 - Changes to relationships (inheritance, association, usage)
 - Changes to attributes (visibility, deprecation-status, ...)

Evaluation

- How did they create the ground truth?
 - Use a very low threshold 1% and manually inspect all of them
 - Changes identified by UMLDiff and the ones UMLDiff missed, which were manually added through their manual inspection using JDEvAn tool
- Precision
- Recall

Precision vs. Recall



precision = % of returned entities are relevant

recall = what % of relevant entities are
returned

How good is UMLDiff?

	HtmlUnit	JFreeChart	Eclipse JDT
Type	Unit testing framework for web apps	Java library for drawing charts	IDE and Plugin-based framework
Major releases	11 (~4 years)	31 (~5 years)	6 (~3 years)
Average #Class	~200	~450	~4000
Renamings* (Precision) [Threshold 0.3] (Recall)	97.2% 98.5%	95.2% 96.4%	93.8% 96.6%
Moves* (Precision) [Threshold 0.4] (Recall)	99.5% 99.9%	91.1% 97.1%	84.8% 90.3%

*Results with heuristics: Name, Comment, Structure, Src/TrgContext, #PotentialMoves, TransitiveUsage, Round=3

Evaluation

JDEvAn in Eclipse

The screenshot displays the Eclipse IDE with the JDEvAn tool. The main window, titled "Design Evolution Analysis - Two-way compare of jfreechart0.9.4/src with jfreechart0.9.5/src - Eclipse SDK", shows an "Inheritance Change Tree" for the project "jfreechart_catchup_0911_094_r40_m30". The tree is expanded to show the "com.jrefinery.chart" package, with "ValueAxis" selected. A context menu is open over "ValueAxis", listing options such as "Next ChangeTree", "Previous ChangeTree", "Refresh", "Show Only Changed Entities", "Hide Entities", "Diff Class Usage", and "Correct Wrong Move".

Below the main window, a secondary window displays a table of changes between the two versions:

origname	legacyname	srcname	trgname
com.jrefinery.chart	com.jrefinery.chart.axis	com.jrefinery.chart.Tick	com.jrefinery.chart.axis.Tick
com.jrefinery.chart	com.jrefinery.chart.axis	com.jrefinery.chart.TickUnit	com.jrefinery.chart.axis.TickUnit
com.jrefinery.chart	com.jrefinery.chart.axis	com.jrefinery.chart.TickUnits	com.jrefinery.chart.axis.TickUnits
com.jrefinery.chart	com.jrefinery.chart.axis	com.jrefinery.chart.ValueAxis	com.jrefinery.chart.axis.ValueAxis
com.jrefinery		com.jrefinery.chart.VerticalAxis	com.jrefinery.chart.axis.VerticalAxis
com.jrefinery		com.jrefinery.chart.axis.VerticalCategory	com.jrefinery.chart.axis.VerticalCategory
com.jrefinery		com.jrefinery.chart.axis.VerticalLogarithmic	com.jrefinery.chart.axis.VerticalLogarithmic
com.jrefinery		com.jrefinery.chart.axis.VerticalNumber	com.jrefinery.chart.axis.VerticalNumber
com.jrefinery		com.jrefinery.chart.axis.VerticalSymbolic	com.jrefinery.chart.axis.VerticalSymbolic
com.jrefinery		com.jrefinery.chart.plot.CategoryPlot	com.jrefinery.chart.plot.CategoryPlot
com.jrefinery		com.jrefinery.chart.plot.CategoryPlotContainer	com.jrefinery.chart.plot.CategoryPlotContainer
com.jrefinery		com.jrefinery.chart.plot.CategoryPlot.getParent()	com.jrefinery.chart.plot.CategoryPlotContainer.getParent()
com.jrefinery		com.jrefinery.chart.plot.CategoryPlot.setParent()	com.jrefinery.chart.plot.CategoryPlotContainer.setParent()
com.jrefinery		com.jrefinery.chart.plot.CategoryPlot.setParent()	com.jrefinery.chart.plot.CategoryPlotContainer.setParent()
com.jrefinery		com.jrefinery.chart.plot.CombinedXYPlot	com.jrefinery.chart.plot.CombinedXYPlot

The status bar at the bottom of the secondary window shows: "{id:12815}{R:2,M:3,MO:2}{Moved from com.jrefinery.chart.ValueAxis}"

JDEvAn Viewer in Eclipse

The screenshot displays the Eclipse IDE with the JDEvAn Visualizer tool open. The main window shows a UML class diagram with several classes and their relationships. The classes are:

- `java.lang.Object` (Superclass)
- `ValueAxis` (Superclass)
- `NumberAxis` (Subclass of ValueAxis)
- `HorizontalNumberAxis` (Subclass of NumberAxis)
- `ColorBarAxis` (Subclass of ValueAxis)
- `HorizontalColorBarAxis` (Subclass of ColorBarAxis)
- `ColorBar` (Subclass of HorizontalColorBarAxis)

Annotations and transformations are visible:

- Replace Inheritance with Delegation:** A note indicates that `ColorBar` is no longer a `ValueAxis`, but instead composes a `ValueAxis` object to reuse axis-related features.
- Extract Superclass:** A note indicates that `ObjectTable`, `PaintTable`, and `StrokeTable` classes have been processed to reduce duplication.
- Method Transformations:** A transformation is shown: `Call super(...) or super.xxx(...)` [No longer existing] `[setMaximumValue(double)[Renamed] -> setMaximumAxisValue(double)[Matched]]`.

The bottom panels show the Outline, Package Explorer, and Properties views. The Package Explorer shows the package structure, and the Properties view shows details for the selected class, `HorizontalColorBarAxis -> ColorBar`.

Property	Value
As source	3 shown outgoing relations
As target	1 shown incoming relations
Category	Class
ID	55971 <-> 70472
Location	Point(8, 155)
Moved entities	
Name	HorizontalColorBarAxis -> ColorBar
Size	Dimension(-1, -1)
UMLDiff status	Renamed
Visibility	public

Synthesis of Refactoring Reconstruction Techniques

Method	Program Element Characteristics	Versions
Origin Analysis 2005	name similarity, code metrics, calls	two complete versions selected manually
UMLDiff 2005	name similarity, code relationships	two complete versions selected manually
M. Kim et al. 2007	name similarity	two complete versions selected manually
S. Kim et al. 2005	name similarity, code metrics, calls, textual similarity	two complete versions selected manually
Dig et al. 2006	syntactical similarity, code relationships	two complete versions selected manually
Weissgerber et al. 2006	structural and code clone differences	all change sets between two versions
SemDiff 2008	structural and outgoing call differences	all change sets between any versions

[Source: Recommending Adaptive Changes from Framework Evolution, Barthelemy Dagenais and Martin Robillard, ICSE 2008]

API-Evolution Support with Diff-CatchUp

IEEE Transactions on Software Engineering, 2007

How can this information be used to support
developers and in what tasks?



Diff-CatchUp Approach

- Automatically recover the evolution of framework APIs
 - UMLDiff and change-pattern queries
- Suggest ways to migrate client applications
 - Refactored API
 - Present the **refactorings** that the API is involved in and its renaming/move **counterparts** in new version if any
 - Removed (deprecated, visibility-restricted, no-longer-inherited, and class-made-abstract) API
 - Locate “**voluntary**” migration examples
 - Recommend **replacing APIs**

Migrate to Refactored API

- RenameMethod(maxSize(), highWaterMark())

Prob #1: The method `maxSize()` is undefined for the type `MonitorableQueue`

Reason: The method name changed

Solution: Update the method call with new name

- ChangeParamType(offerMany(...), Collection, Object[])

Prob #2: The method `offerMany(Object[])` in the type `MonitorableQueue` is not applicable for the argument (`Collection`)

Reason: Parameter type changed

Solution: Obtain `Object[]` from `Collection`
(e.g. `Collection.toArray()`)



Migrate to Refactored API

- **RenameClass**(Queue, SimpleQueue)
- **ExtractInterface**(SimpleQueue, Queue)
- **AddAbstraction**(FastQueue, Queue)
- **AddAbstraction**(MonitorableQueue, Queue)

Prob #3: Cannot instantiate the type Queue

Reason: The Queue represents a newly introduced interface in the new version.

The original class Queue is renamed as SimpleQueue.

Solution: Create SimpleQueue object, or

See if the interface Queue's other implementation classes can be used as well.



Migrate to Refactored API

- **ReplaceInheritanceWithDelegation**(MonitorableQueue, SimpleQueue, internalQueue, Queue)
- **ReplaceInheritanceWithDelegation**(SimpleQueue, ArrayList, elementData, List)
- **ExtractInterface**(SimpleQueue, Queue)
- **AddAbstraction**(MonitorableQueue, Queue)

Prob #4: Type mismatch: cannot convert `MonitorableQueue` to `List`

Reason: `MonitorableQueue` is no longer `SimpleQueue`, which is no longer `List`

Solution: Stop using `MonitorableQueue` as `List` object
May use it as a `Queue` object



Migrate to “Removed” API

- **ReplaceInheritanceWithDelegation**(
SimpleQueue, ArrayList, elementData, List)

Prob #6: The method `listIterator()` is undefined for the type
`Queue`

Reason: The original `Queue` class used to be a `List`; it inherits
`listIterator()` from its superclass `ArrayList`, but
no longer doing so.

This is essentially a “**removed**” API.
How am I going to **replace** it?



Diff-CatchUp in Eclipse

The screenshot displays the Eclipse IDE interface with the 'API Catchup' tool active. The main editor shows a Java file named 'BaseImageServlet.java' with a switch statement. A context menu is open over the 'new PlotFit' line, with 'Catchup API Evolution' selected. Below the editor, the 'API Proposals' panel lists several suggestions, including 'com.jrefinery.data.MovingAverage.createMovingAverage' (added) and 'com.jrefinery.chart.demo.DemoDatasetFactory.createHighLowDataset' (renamed). The 'Usage Examples' panel shows examples for the selected method. On the right, the 'Problems' view lists 43 errors, with 'Fatal Errors (43 of 43 items)' expanded to show a list of unresolved types and methods. A 'Go To' dialog is also visible, showing a search for 'com.jrefinery.chart.data'.

```
switch (type) {
    case 21:
        // moving avg
        MovingAveragePlotFitAlgorithm mavg =
            new MovingAveragePlotFitAlgorithm();
        mavg.setPeriod(30);
        PlotFit pf = new PlotFit(chartData, mavg);
        xyData = pf.getFit();
        break;
    case 22:
        // linear fit
        pf = new PlotFit(chart
        xyData = pf.getFit();
        break;
    case 0:
```

Message	Status	Support
com.jrefinery.data.MovingAverage.createMovingAverage(XYDataset,String,long,long)	added	0.5
com.jrefinery.data.TimeSeriesCollection.TimeSeriesCollection()	matched	0.6666667
com.jrefinery.chart.demo.DemoDatasetFactory.createHighLowDataset()	renamed	0.5
com.jrefinery.data.DatasetUtilities.sampleFunction2D(Function2D,double,double,int,String)	matched	0.1666667
com.jrefinery.data.JDBCXYDataset.JDBCXYDataset(Connection,String)	renamed	0.1666667

Message	Support
com.jrefinery.chart.demo.JFreeChartDemoBase.createCombinedAndOverlaidChart1()	2.0

- Axis cannot be resolved to a type
- JdbcCategoryDataset cannot be resolved
- JdbcCategoryDataset cannot be resolved
- JdbcPieDataset cannot be resolved to a
- JdbcPieDataset cannot be resolved to a
- JdbcXYDataset cannot be resolved to a
- JdbcXYDataset cannot be resolved to a
- LinearPlotFitAlgorithm cannot be resolve
- MovingAveragePlotFitAlgorithm cannot b
- MovingAveragePlotFitAlgorithm cannot b
- PiePlot cannot be resolved to a type
- PiePlot cannot be resolved to a type
- PiePlot cannot be resolved to a type
- Plot cannot be resolved to a type
- PlotFit cannot be resolved to a type
- PlotFit cannot be resolved to a type
- PlotFit cannot be resolved to a type
- The import com.jrefinery.chart.Axis can
- The import com.jrefinery.chart.data can
- The method setTitle(ArrayList) is undef
- ValueAxis cannot be resolved to a type

How Good is Diff-CatchUp?

Type of problem	#broken API	#success proposal	%
JFreechart			
ImportNotFound	17	17	100
UndefinedType+ImportNotFound+UndefinedName	254	247	97.2
InvalidClassInstantiation	1	1	100
UndefinedMethod/Constructor	180	151	83.9
ParameterMismatch	54	54	98.1
UndefinedField+UndefinedName	33	29	87.9
UsingDeprecatedType	3	3	100
UsingDeprecatedMethod/Constructor	35	34	97.1
Total	577	535	92.7
HTMLUnit			
UndefinedType	1	1	100
UndefinedMethod/Constructor	11	9	81.8
ParameterMismatch	3	3	100
UsingDeprecatedType	1	0	0
UsingDeprecatedMethod/Constructor	10	7	70
Total	26	20	76.9

32

My thought on Refactoring Reconstruction Research

- Promising ways to allow programmers to understand code changes at a high level
- Still long ways to go to automatically reconstruct design intent from source code
- It can be applied to mining software repository research.
- This is a challenging problem:
 - heuristics-based, often requiring many similarity thresholds
 - hard to evaluate this type of work in general.

Preview for Monday after Spring Break

- First of all--- have a fun & productive spring break!
- Crosscutting Concerns
 - Why some code changes are crosscutting?
 - Read Visitor Pattern from Design Patterns book--- We may have a quiz on crosscutting concerns (using the visitor pattern code example) on Monday.