

A Generic Approach to Supporting Diagram Differencing and Merging for Collaborative Design

Akhil Mehra¹, John Grundy^{1, 2} and John Hosking¹

¹Dept. Computer Science and ²Dept. Electrical and
Computer Engineering
University of Auckland, New Zealand

ASE 2005

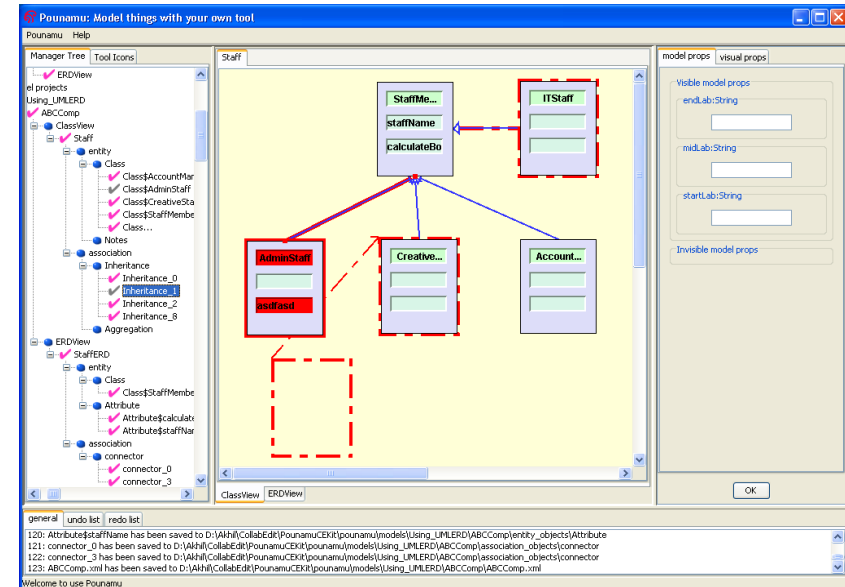
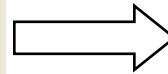
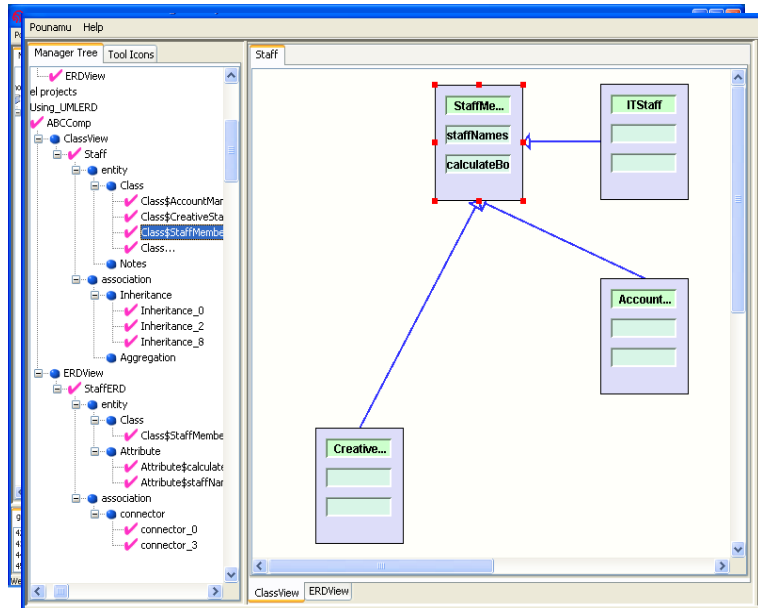
Outline

- Motivation
- Version control of software artifacts
- Differentiation and merging
- Our approach: visual artifact differentiation & merging via plug-in components
- Examples of usage
- Architecture & Design
- Future work
- Conclusions

Motivation

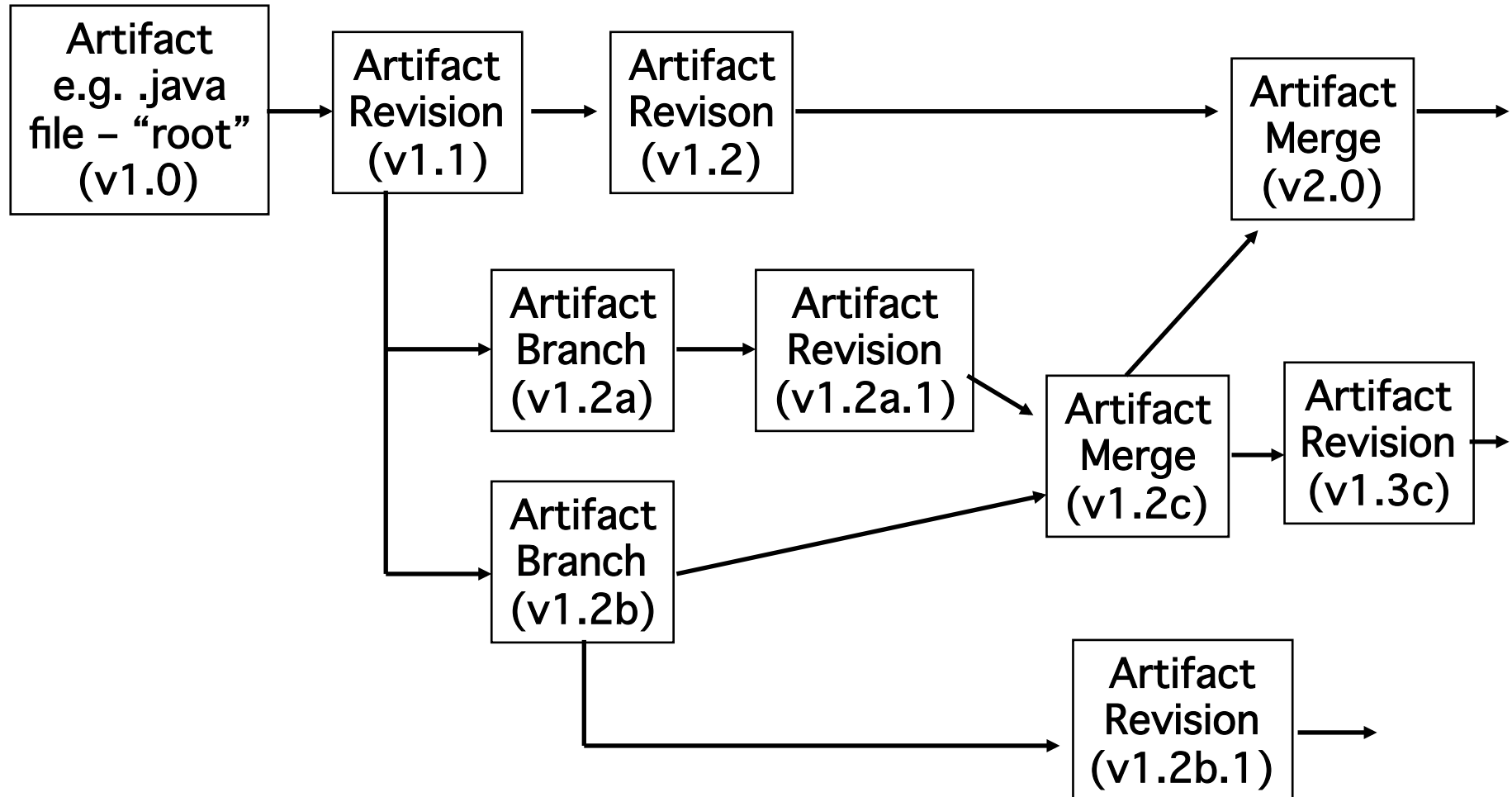
- To support collaborative, asynchronous work we need:
 - Support for multiple versions of software artifacts & configuration management
 - Ability to compare versions (“differentiation”)
 - Ability to combine versions (“merging”)
- Good support exists for textual & XML-based versioning differentiation and merging
- Limited or no support for visual design artifact versioning, differentiation and merging
- Wanted to add such support to a design environment meta-tool for use across wide variety of design tools

Example - Requirements



- Need to:**
1. Create alternate version
 2. Edit alternate
 3. Compare to repository
 4. Display changes
 5. Merge some/all changes
 6. Check back into repository

Version Control 101



Differentiation & Merging

- Find changes (differentiation) between two documents that are alternative versions (share the same root document)
- Classical textual comparison - “diff” of documents d1 and d2:
 - Several algorithms developed
 - Usual approach is to identify “islands of similarity” between d1 and d2
 - Then build set of line additions/deletions/changes that if applied to d1 would convert it to d2
 - Set of additions/deletions is a “delta” between d1 and d2
 - Merging applies all or some of delta to d1 - if all applied, get d2 else if some applied, get d3, a merging of some changes from d1 and some from d2 producing a third alternate, d3
- Similar algorithms for XML (hierarchical document) diffing:
 - Find add/delete of element/attribute nodes; changed values

Eclipse Example

The screenshot shows the Eclipse IDE interface with the following components:

- Top Bar:** Team Synchronizing - DiagramEditPart.java - Eclipse SDK
- Menu Bar:** File, Edit, Navigate, Search, Project, Tomcat, Run, Window, Help
- Toolbar:** Standard Eclipse development icons.
- Left Panel (CVS Compare):** Shows a project tree for 'MaramaEditor' with files like DiagramEditPart.java, MaramaSketchedShapeEditPart.java, and PounamuShapeEditPart.java.
- Right Panel (Java Source Compare):** Compares 'Local File (1.1.6.1.6.4)' and 'Remote File (1.1.6.1.4.2)'. Both files show the `showTargetFeedback` method. The local file uses `TimestampedPoint` and `getFigure().add(line)`, while the remote file uses `Point` and `line.addPoint(points.get(i))`.
- Bottom Panel:** Tasks, Problems, Console. A message states 'A console is not available.'
- Taskbar:** Windows taskbar with 'start' button, 'Inbox for john-g@cs...', 'ASE 2005 Papers - Mi...', 'Microsoft PowerPoint ...', 'D:\java\ eclipse', 'Team Synchronizing ...', 'EN', and system tray icons.

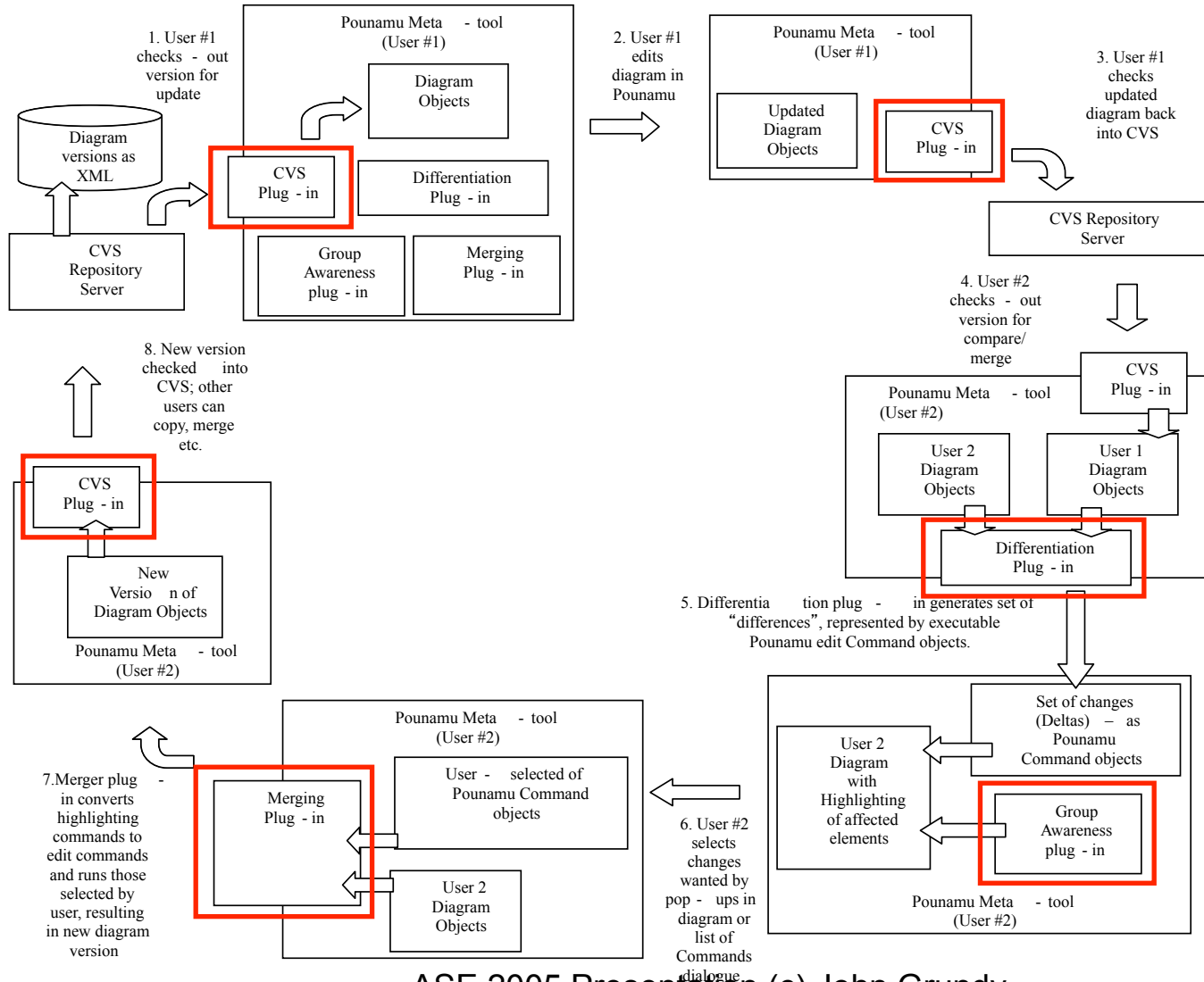
Problem: doesn't work very well for visual artifacts

- In CASE and other design tools:
 - Diffing of text files or XML files commonly used to compare e.g. underlying UML models
 - Cognitive gap between diff of diagram model and its visual representation
 - Hard to visualise what changes really are and control merging
- What we really want:
 - Diffing using diagram data structures (graph)
 - Presentation of deltas in-situ in diagrams
 - Selective accept/reject of changes by user

Our Approach

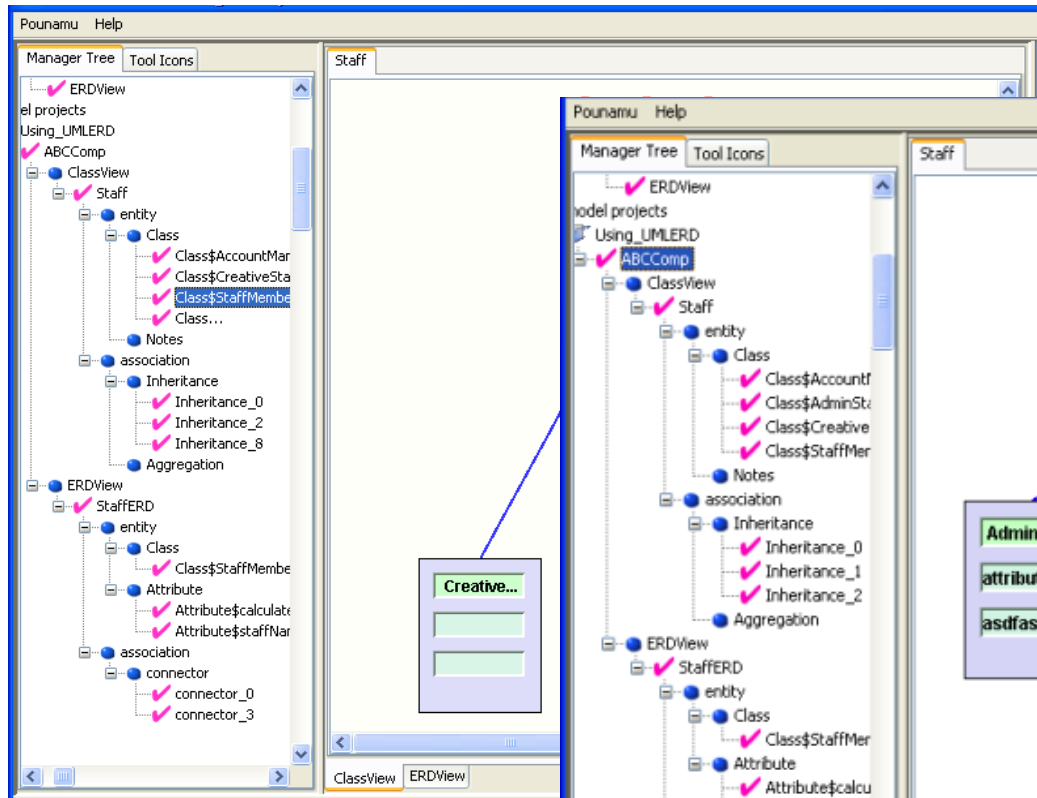
- Added set of plug-ins to Pounamu visual design meta-tool to support version management, diffing, merging:
 - Plug-in to support CVS check-in/out of diagrams
 - Plug-in to support visual diffing of diagram versions - editing Command objects synthesized as delta representation - works for ANY diagram type
 - Plug-in to visualise Command object deltas in one of the diagram versions (actually reused from a collaborative editing plug-in...)
 - Plug-in to allow user to selectively accept/reject delta items
 - runs Commands on a diagram to accept changes
- Plug-ins added to single-user meta-tool - no code change made to this to support check-out/diff/highlight/merge/check-in!

Our Approach

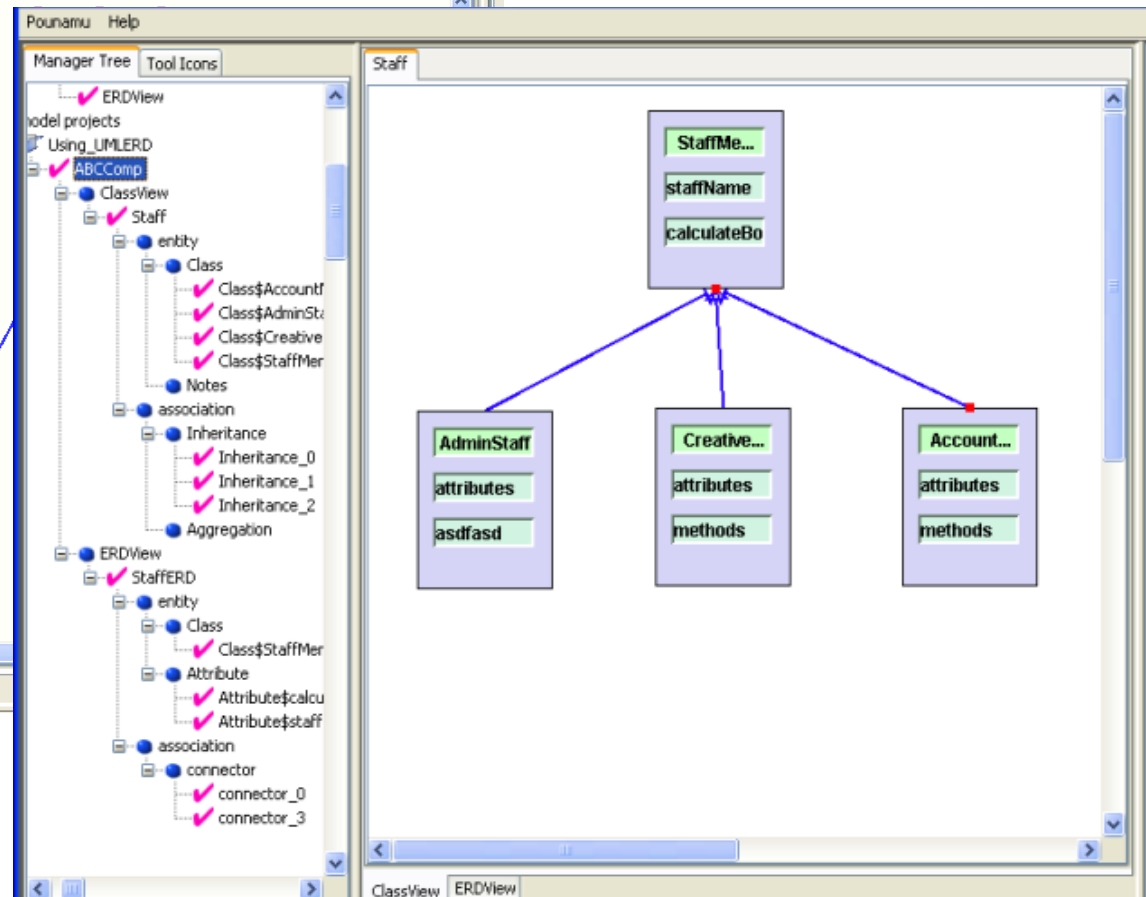


Examples of using

John's version:



Akhil's version:



CVS Check-out by Akhil

Pounamu CVS Configuration Dialogue

CVS Parameters

Protocol: :pserver:
Server: Ameh010-pc
Port: 2401
Folder: /CVS/cvsrepo
User Name: john
Password: ****

Get Versions

File Versions Available

User Name	Version	Date	Comments
tim	1.5	2005/01/16 04:36...	Added project wit...
john	1.4	2005/01/16 04:28...	Added with ERD ...
tim	1.3	2005/01/16 04:17...	Added with modifi...
john	1.2	2005/01/16 04:07...	Modle Project add...
john	1.1	2005/01/16 03:01...	Add Comments B...

Differentiate

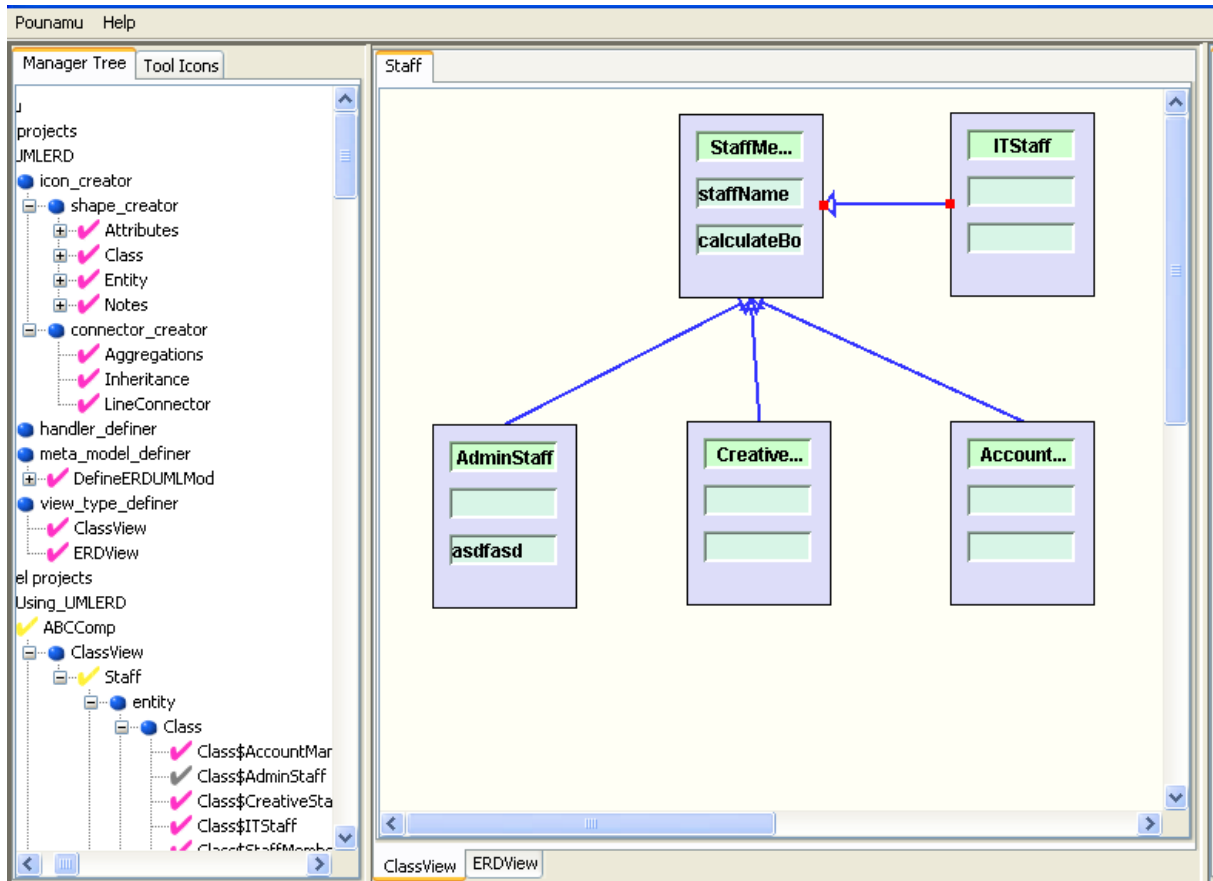
Differencing

The screenshot shows the Pounamu software interface. The main window displays a UML class diagram with the following elements:

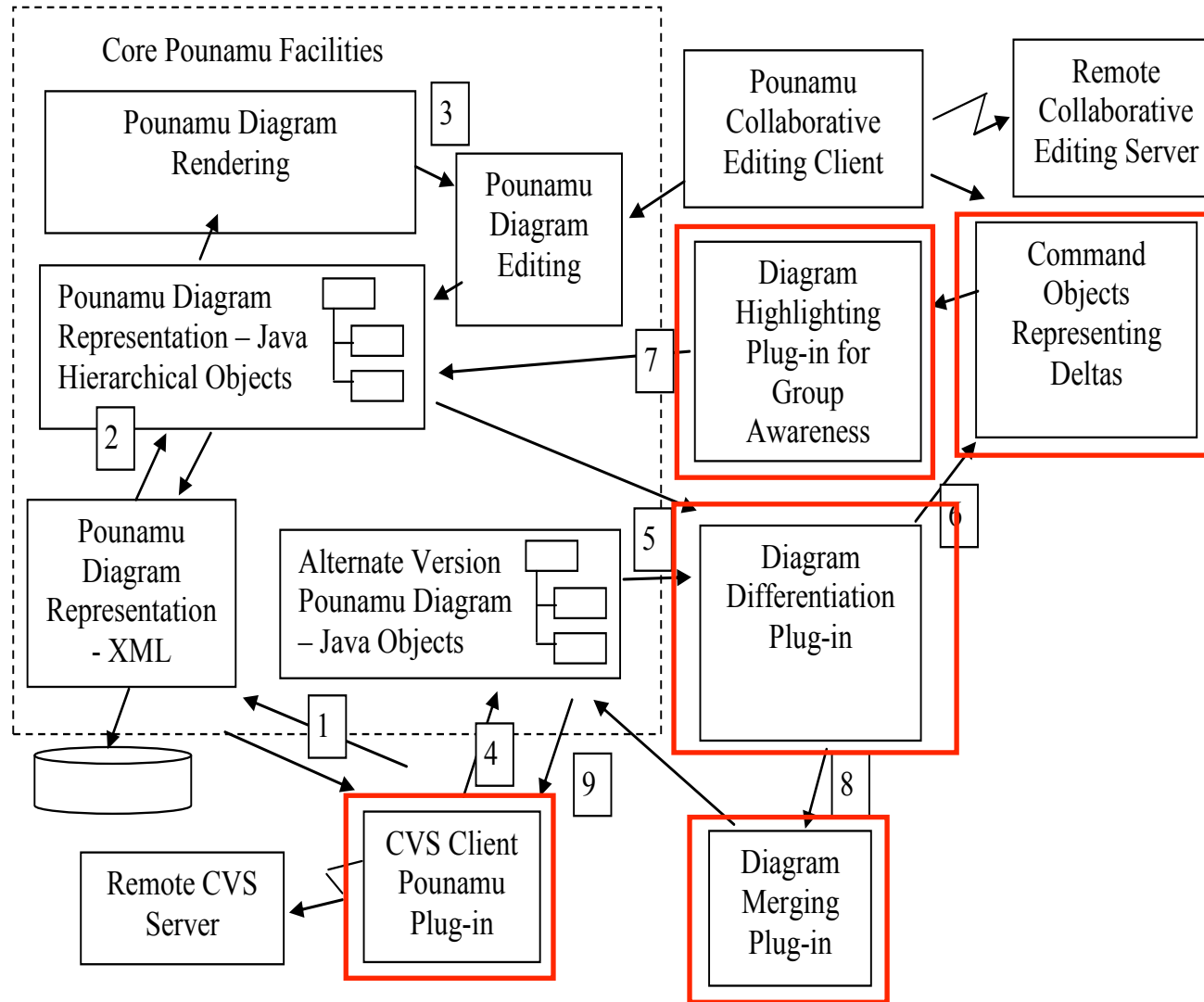
- Class Diagram:**
 - StaffMe...** (Class) with attributes `staffName` and `calculateBo`.
 - ITStaff** (Class) associated with **StaffMe...** via a dashed red arrow.
 - AdminStaff** (Class) associated with **StaffMe...** via a solid red arrow.
 - Creative...** (Class) associated with **StaffMe...** via a solid blue arrow.
 - Account...** (Class) associated with **StaffMe...** via a solid blue arrow.
 - Inheritance:** **AdminStaff**, **Creative...**, and **Account...** inherit from **StaffMe...** (indicated by solid lines with hollow triangle heads).
- Manager Tree (Left):** Shows a hierarchical view of the model, including 'Staff' and 'StaffMe...'.
- model props (Right):** A panel for configuring model properties, with fields for `endLab:String`, `midLab:String`, and `startLab:String`.
- Table Window (Bottom Right):**

Serial Nu...	Event	Artifact A...	Status	User Name
1	Shape Mo...	93852D5B...	Accepted	John
2	Shape Mo...	40DA9DD0...	Accepted	John
3	Shape Mo...	0F3CB872...	Accepted	John
4	Add Shape	70EDFE40...	Accepted	John
5	Change Pr...	70EDFE40...	Executing	John
6	Change Pr...	82ED1667...	Executing	John
7	Change Pr...	82ED1667...	Executing	John
8	Change Pr...	82ED1667...	Executing	John

Merging



Architecture

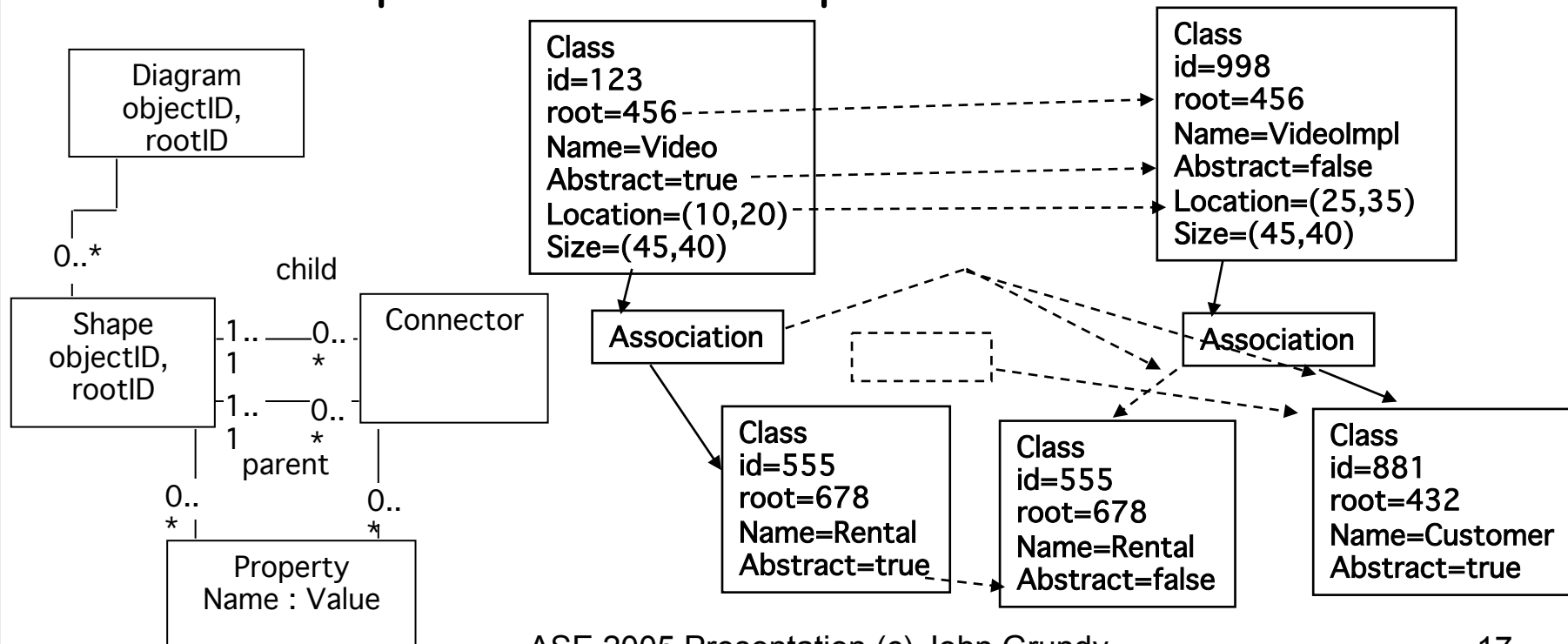


Differencing Algorithm

- Differencing does 2 passes over diagram datastructure - shape then connector comparison
- Uses a “root ID” for each shape to determine which items in two versions share same root version
- Compares attribute values
- Compares position, size
- Distinguishes “contains” and “related to” connectors
- Builds Create/DeleteShape, SetProperty, MoveShape, ResizeShape, Create/DeleteConnector Commands
- Highlighting temporarily annotates diagram with Command list info

Differencing

- Pounamu diagram data structure:
 - All element use “root ID” to assist related item identification
 - Sub-shape vs related shape info also used



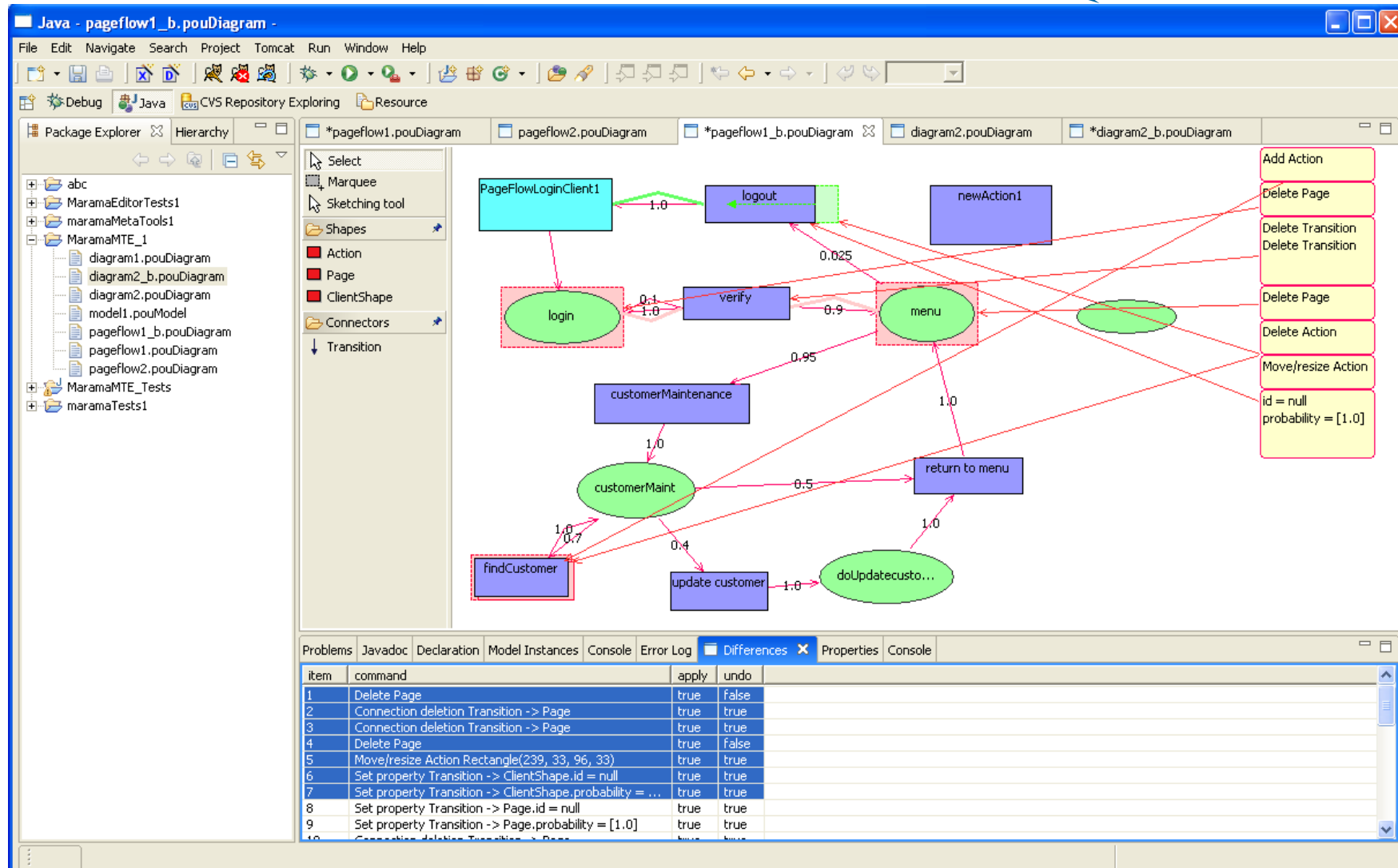
Evaluation

- Usability analysis via survey and Cognitive Dimensions
- Gutwin's groupware assessment framework
- Visibility - displays differences in-situ
- Viscosity - user can accept/reject changes directly
- Hidden dependencies - reduces
- Consistency - of change representation/acceptance
- Error-proneness and hard mental operations - reduced
- Presence & authorship - clear
- Awareness of change - explicit
- Intention awareness - some support

Current/Future work

- Incorporated into Eclipse-based meta-tool (Marama)
- Extended Command visualisation support
- Word-like tracking annotations
- Use of semantic constraint checking to enhance differencing and ordering of Commands planned
- Extend to provide highlighting re-configuration by users:
 - Change appearance of annotations
 - Extend comparison mechanism
 - Change way user can accept/reject

Eclipse Example



The screenshot displays the Eclipse IDE interface for editing a UML PageFlow diagram. The main workspace shows a complex flow of pages and actions. Pages are represented by rectangles (e.g., PageFlowLoginClient1, login, verify, menu, customerMaintenance, customerMaint, findCustomer, update customer, doUpdatecusto..., return to menu) and actions by ovals (e.g., login, verify, menu, customerMaint, doUpdatecusto...). Transitions connect these elements with arrows and associated probabilities (e.g., 1.0, 0.1, 0.9, 0.025, 0.95, 1.0, 0.5, 1.0, 0.4, 1.0, 0.7). A toolbar on the right side of the diagram provides various manipulation tools such as 'Add Action', 'Delete Page', 'Delete Transition', 'Move/resize Action', and 'id = null probability = [1.0]'. The bottom of the IDE features a 'Differences' window with a table of changes:

item	command	apply	undo
1	Delete Page	true	false
2	Connection deletion Transition -> Page	true	true
3	Connection deletion Transition -> Page	true	true
4	Delete Page	true	false
5	Move/resize Action Rectangle(239, 33, 96, 33)	true	true
6	Set property Transition -> ClientShape.id = null	true	true
7	Set property Transition -> ClientShape.probability = ...	true	true
8	Set property Transition -> Page.id = null	true	true
9	Set property Transition -> Page.probability = [1.0]	true	true
10	Connection deletion Transition -> Page	true	true

Conclusions

- Version control for visual software artifacts requires differentiation/merging support as with textual/XML artifacts
- Comparing graph-based visual designs is more complex in some ways; easier in others
- We have prototyped generic algorithm for differencing as set of plug-ins to the meta-tools Pounamu and Marama (itself a set of Eclipse plug-ins)
- Differencing produces set of editing Commands
- We reuse a collaborating editing plug-in to provide Command highlighting facility
- We support partial or full change merging via Command execution on versions

References

- Mehra, A., Grundy, J.C. and Hosking, J.G., Adding Group Awareness to Design Tools Using a Plug-in, Web Service-based Approach, In Proceedings of the Sixth International Workshop on Collaborative Editing Systems, CSCW 2004, Chicago, November 6, 2004.
- Mehra, A., Grundy, J.C. and Hosking, J.G. Supporting Collaborative Software Design with a Plug-in, Web Services-based Architecture, In ICSE 2004 Workshop on Directions in Software Engineering Environments, Grundy, Welland and Stoeckle (eds), IEE Press.
- Grundy, J.C. and Hosking, J.G. Engineering plug-in software components to support collaborative work, Software - Practice and Experience, Vol. 32, No. 10, August 2002, Wiley, 983-1013.
- Grundy, J.C. Engineering component-based, user-configurable collaborative editing systems, Engineering for Human-Computer Interaction, Chatty, S. and Dewan, P. Eds, February 1999, Kluwer Academic Publishers.
- Grundy, J.C., Mugridge, W.B., Hosking, J.G., Amor, R.W. Support for Collaborative, Integrated Software Development, in Proceedings of the 7th Conference on Software Engineering Environments (SEE'95), IEEE CS Press, Netherlands, April 5-7, 1995, pp. 84-94.