

732 ISDE and Meta tools Section

- **Aims:** introduction to research issues associated with software tools
- Focus is primarily on **visual tools** - ie tools that use some visual metaphor to assist in software design and implementation
- **Topics** (approx no of lectures):
 - Software Tools Introduction (1)
 - ISDE Frameworks (1)
 - Visual languages & DSLs (2 + 1 for class exercise)
 - Meta tools, meta modelling, MDA (3 + Assmt)
 - Design patterns for framework development (1)
- **Me:** Professor John Hosking (also Prof John Grundy)
Room 303.487
john@cs.auckland.ac.nz

How this section runs

- There is no textbook for this section
- Instead I will be making available research papers
 - These should be regarded like a required text
 - I will be expecting you to read these papers as homework, in some cases before the next lecture
Don't leave this till when you are studying for the exam - there will be too many of them.
 - I don't expect you to know the contents of the papers in detail
 - I will expect you to make cross linkages between the papers and be able to answer "compare and contrast" type questions on the contents
- This is a graduate level paper so an expectation is that you become familiar with research literature and be able to critique it. There will be a classroom exercise related to this.
- These skills are highly regarded by employers

Software Tools

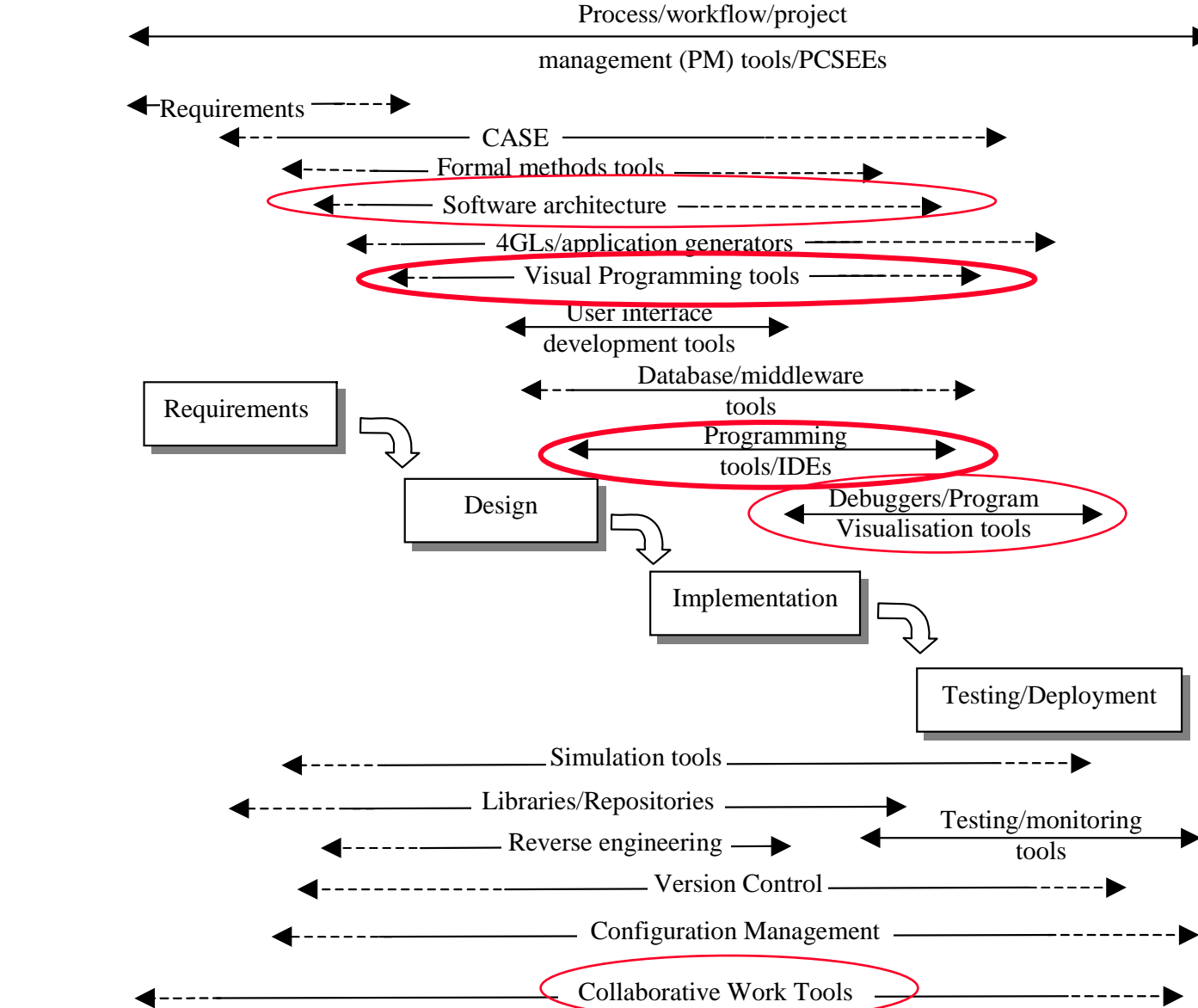
- Tools to support the **development of software**
 - Covers all aspects of the software development lifecycle
 - Covers support for a wide variety of methodologies and technologies
 - Both general purpose and domain specific
- Much research and commercial activity in this area
- Strong research focus in the CS Department at Auckland

- Resource: Software Tools, Grundy and Hosking (Chapter in Wiley Encyclopaedia of Software Engineering)

Context

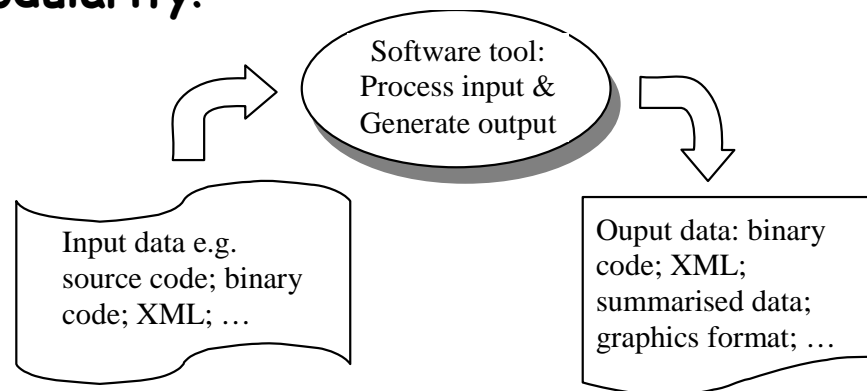
- Rapid change in software development practice in recent times:
 - **Newer development methodologies**, eg RAD, XP/Agile development, Open Source development, that focus on iterative & collaborative development
 - Need for round trip engineering support
 - Need for collaboration support
 - **New technologies to support**, partic wrt distributed systems (eg middleware, component based approaches, web services, aspects)
 - Need new modelling and support tools

Types of tool



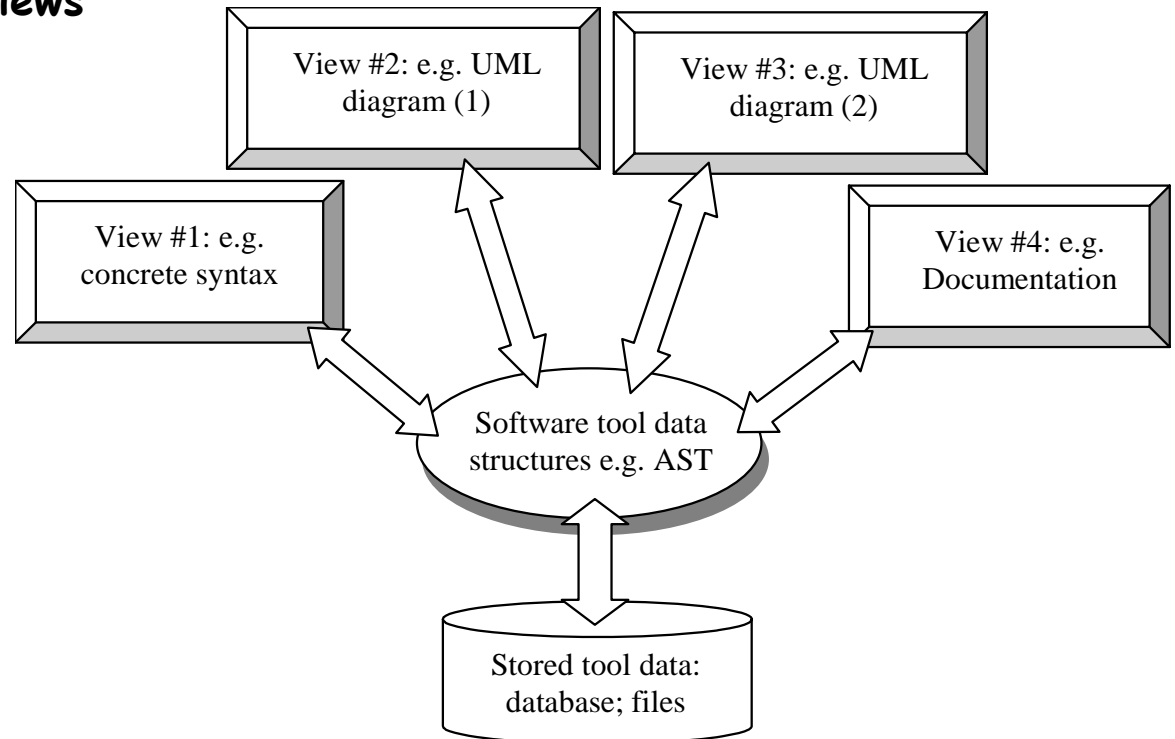
Software Tool Structure

- Batch approach
- Eg conventional compiler
- Communication between tools via files or pipes and filters
- Problems with inter-tool consistency, need for interchange formats, slow turnaround, etc
- But good modularity!



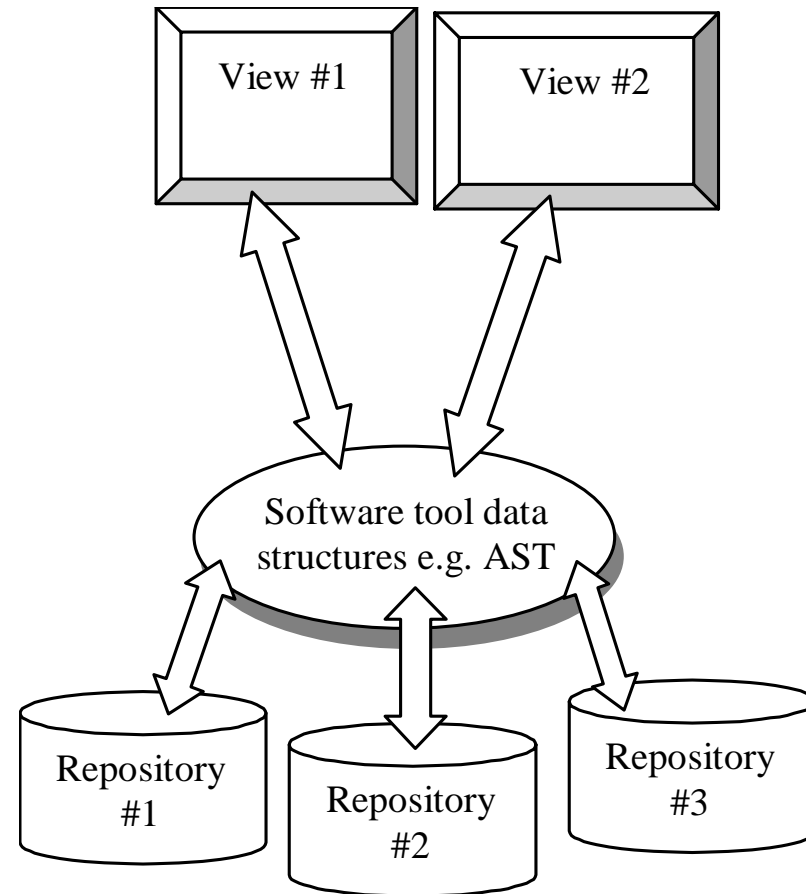
Software Tool Structure

- Interactive, with multiple views, and incremental consistency between views
- **Issues**
- View consistency
 - Difficult problem
- Repository design
 - R/ODBMS
 - Eg PCTE
 - Custom file
- Efficient editing
- Tool tailoring
- Notation tailoring



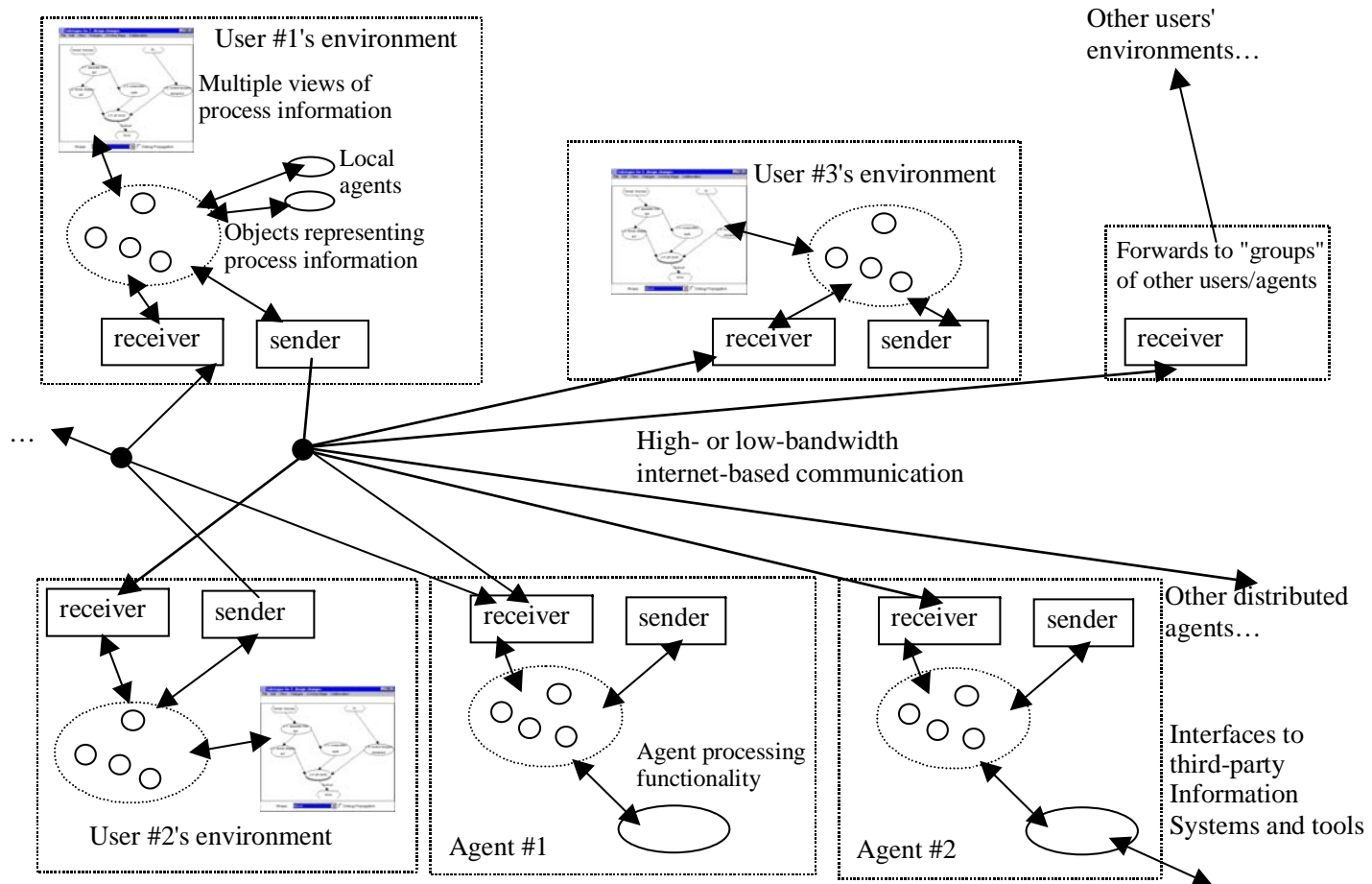
Software Tool Structure

- **Federated repositories**
- **Partition data for**
 - **Efficiency**
 - **Ease of construction**
- **Decentralised with replicated data for**
 - **Robustness**
 - **Performance**



Serendipity II

- Decentralised process modelling



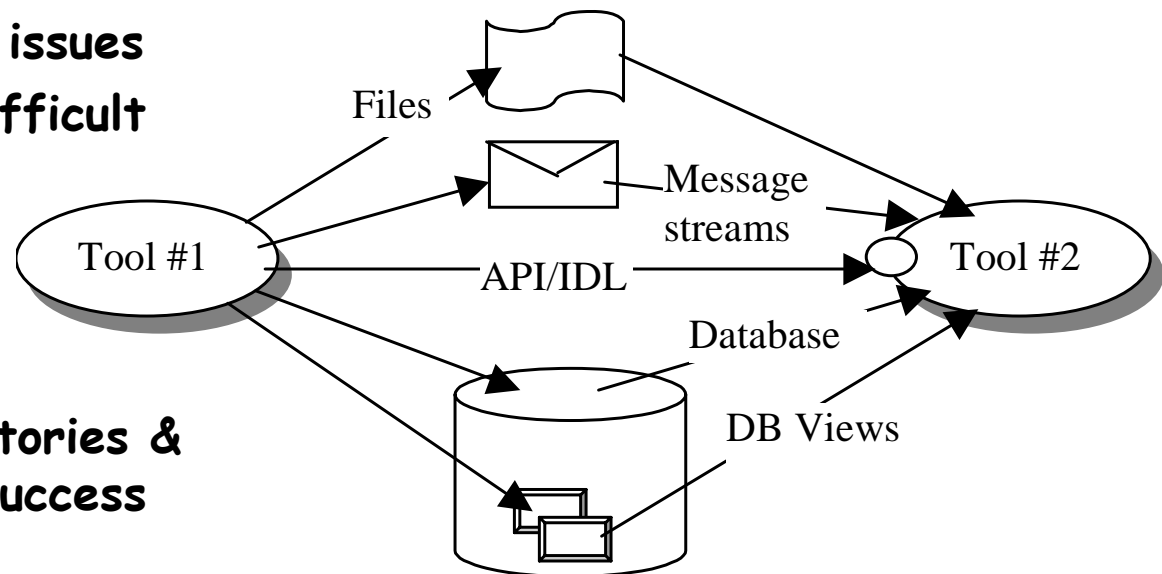
Tool integration

- Trend to using best of breed for different types of tool versus monolithic IDEs
- Need means of providing inter-tool communication for exchanging both control and data events
- Approaches
 - Data integration
 - Control integration
 - Presentation integration
 - Process integration

Data integration

- Data exchange using custom or standard exchange formats via:
 - Need for translators
 - Common formats: UML XMI (OMG), workflow exchange format

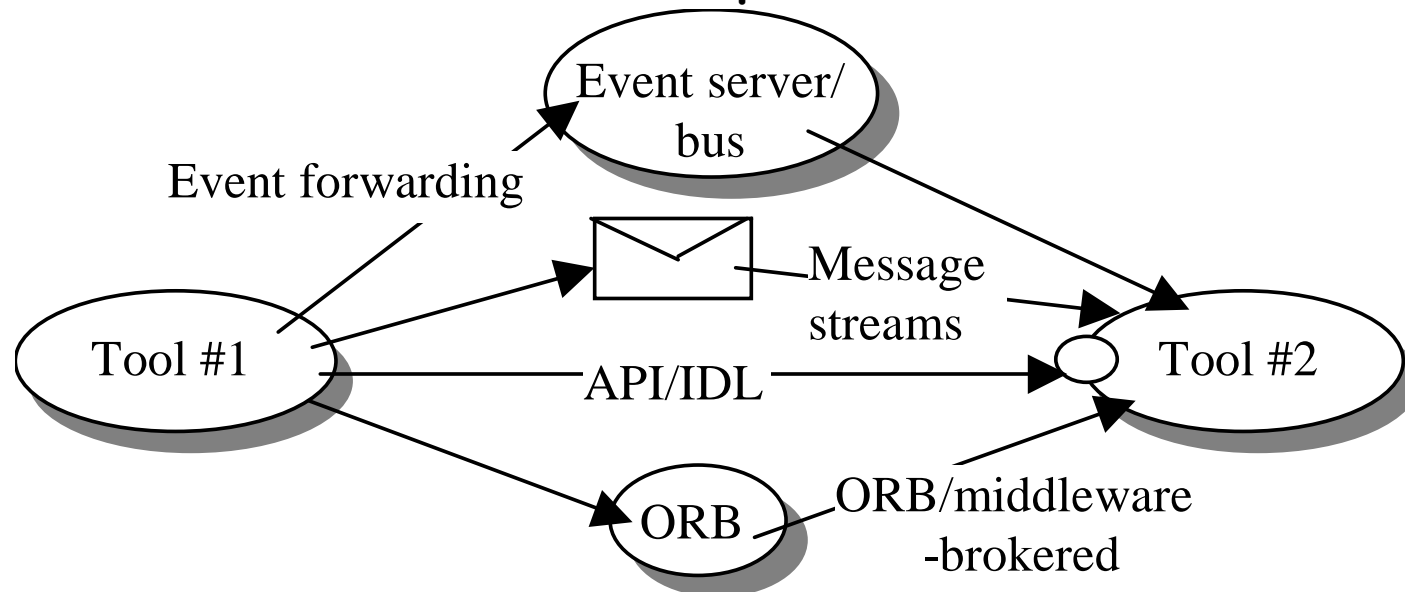
- Tighter coupling via shared database but
 - Performance issues
 - Standards difficult



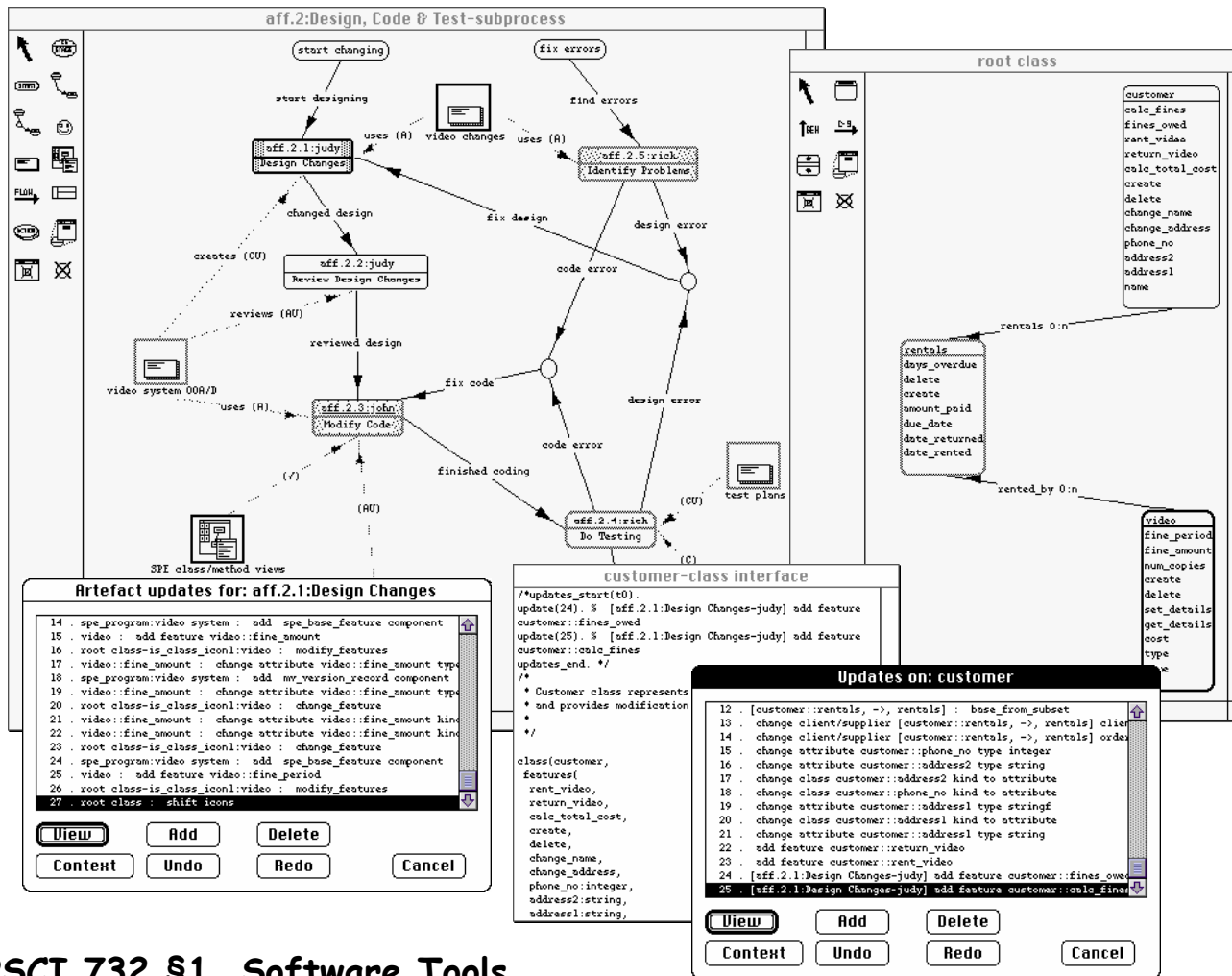
- Document repositories & version control success

Control integration

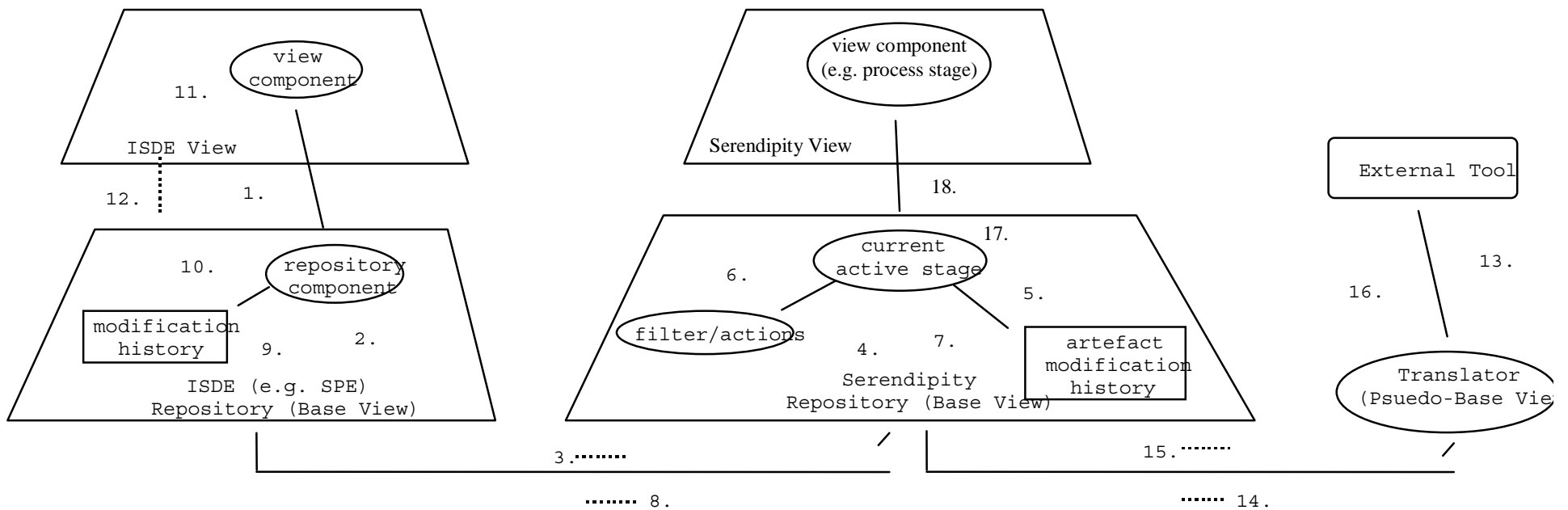
- Variety of approaches
 - Message-oriented using central message broker (eg Field, DEC FUSE)
 - Distributed object approaches eg DCOM, CORBA, web services
 - Need for common component APIs



SPE/Serendipity

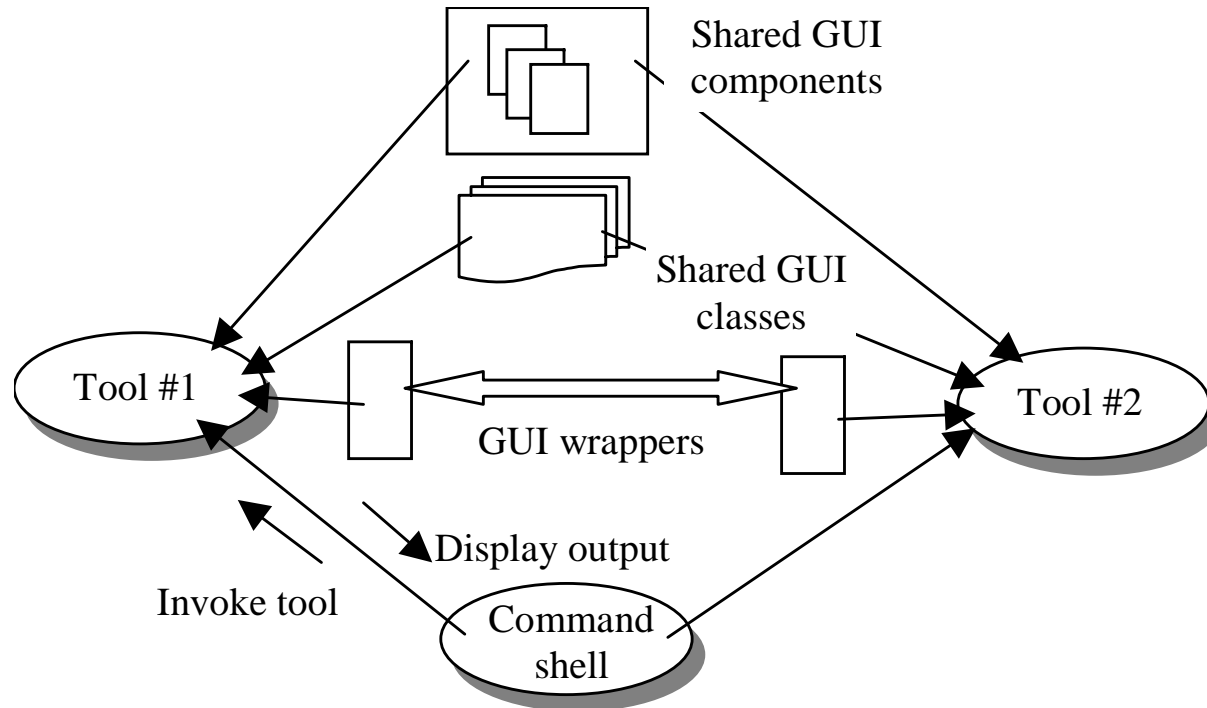


Integration architecture



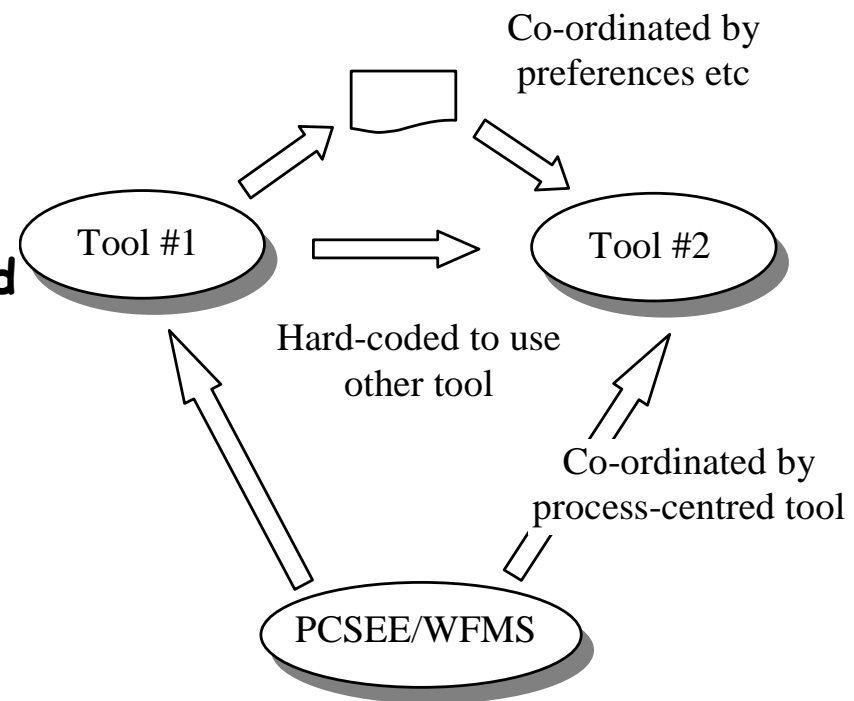
Presentation integration

- Use common interface toolkit (eg tcl/tk, MFC, JFC)
 - Still inconsistencies in usage though
 - Provides common look and feel and eg sharing of menus etc, but still need for eg data integration



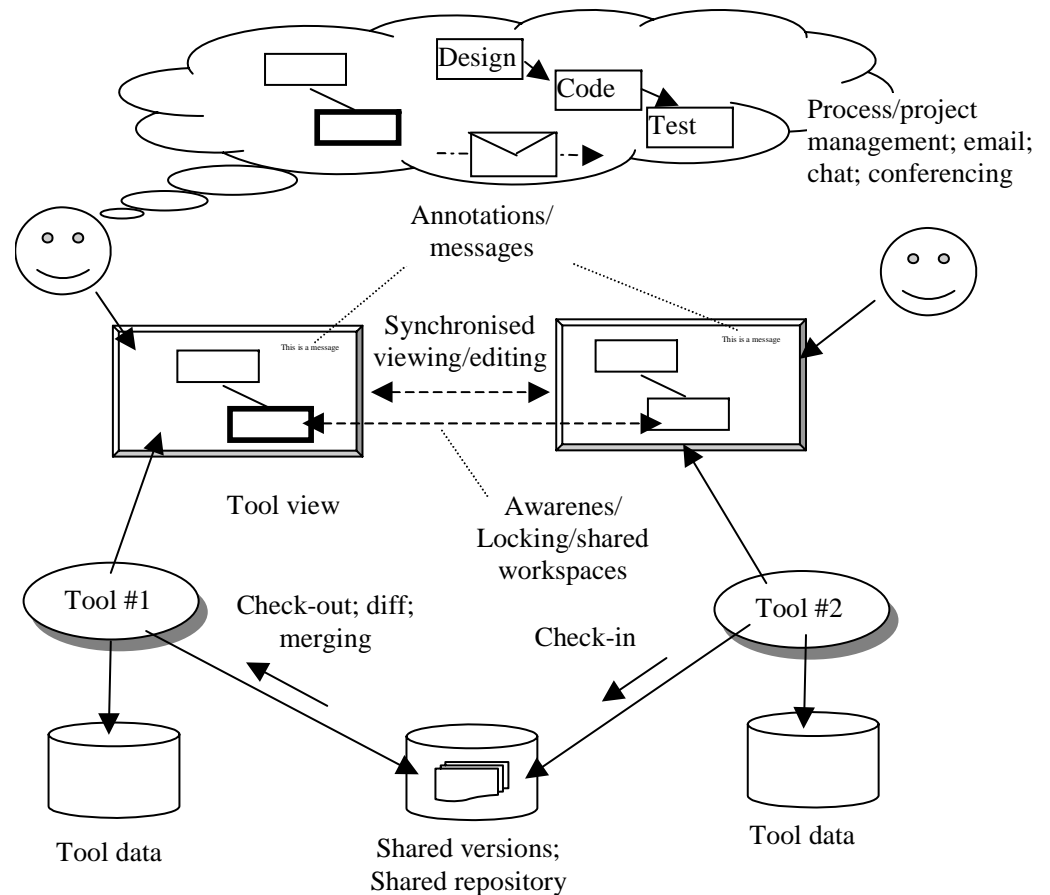
Process integration

- Important for team support, particularly for virtual teams
 - Process centred environments
 - Tight co-ordination of tool use
 - Need for detailed understanding of each tool
 - GP workflow tools to coordinate tool usage
 - Simpler but less powerful
 - Needs data, control and UI integration to work well



Collaborative work support

- Builds on tool integration approaches
 - **Coordination**
 - Project & process mmt
 - Locking of shared artefacts
 - **Comms**
 - eg chat email video audio
 - Doc annotation
 - **Composition**
 - Versioning
 - Version merging
 - Synchronous, asynchronous



Sepependity II CSCW

The screenshot displays a software interface for managing design changes and collaboration. The main window, titled "Substages for 2. design changes", features a menu bar with "File", "Edit", "View", "Changes", "Owning Stage", and "Collaboration". A status bar indicates "View sent to mark at jgrundywp.cs.waikato:". The central workspace contains a diagram with nodes: "design changes", "2.1. separate files" (containing "bill"), "2.2. resize shapes" (containing "bill"), "2.3. composites" (containing "mark"), and "2.5. correct designs" (containing "designers"). A context menu is open over the "2.5. correct designs" node, showing options: "Add Collaborator", "Current Collaborators" (selected, showing "mark (async,red)"), "Request New Changes" (with a "fix" sub-option), and "Request New View". To the right, a list of collaboration modes is shown: "0 - none", "1 - async" (highlighted), "2 - notify", "3 - present", "4 - action", and "5 - sync".

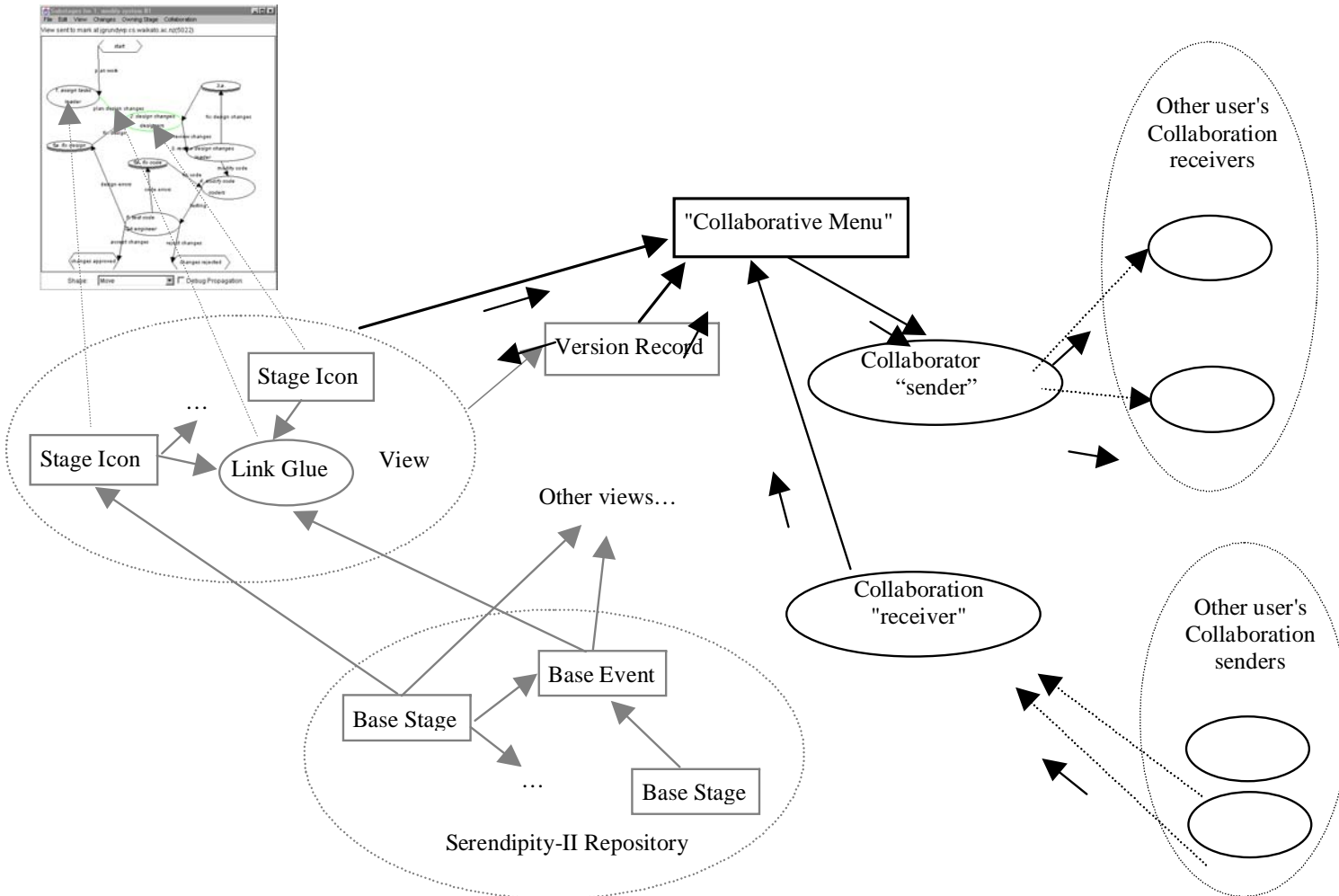
An "Add Collaborator" dialog box is open in the foreground, listing "mark", "bill", and "@historian". It includes buttons for "Add", "Query for Users", "Cancel", "Display Shapes", and "Propagation".

A "Changes for: Substages for 2. design changes" window is also visible, showing a list of change events:

- [mark] 56 SetIntValue 2.1. separate files:y 64 to 63
- [mark] 57 SetIntValue 2.1. separate files:height 36 to 37
- [mark] 58 SetIntValue 2.1. separate files:width 105 to 122
- [mark] 59 SetIntValue 2.1. separate files:y 63 to 74
- [mark] 60 SetIntValue 2.1. separate files:x 1 to 17
- 61. SetIntValue 2.2. resize shapes:x 13 to 14
- 62. SetIntValue 2.2. resize shapes:y 168 to 152
- 63. Macro change: Add Stage Icon
- 64. SetStringValue 2.4. shape colours:nameText to 2.4. shape c
- 65. SetStringValue 2.4. shape colours:parentText to bill
- [mark] 66 SetIntValue Substages for 2. design changes:x 237 to 1
- [mark] 67 SetIntValue Substages for 2. design changes:y 5 to 70
- 68. SetIntValue 2.4. shape colours:y 41 to 90**
- 69. SetIntValue 2.4. shape colours:x 194 to 220

Buttons at the bottom of the changes window include "Undo", "Redo", "Export", and "Close".

Components

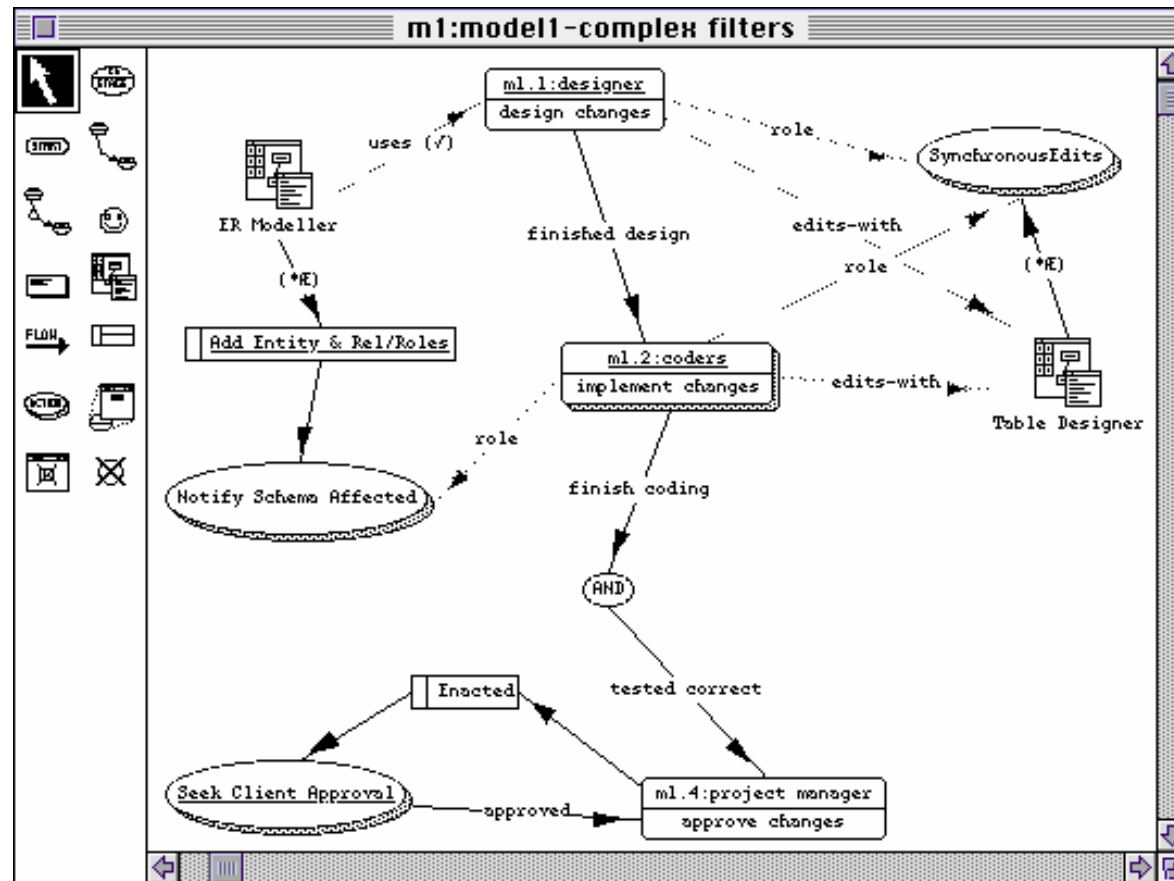


Tool automation

- **Need support for agents that assist in performing tasks related to the software development process**
 - **Analysis - eg syntax, semantics, formal consistency**
 - **Reuse - finding suitable classes etc**
 - **Reuse - instantiating frameworks**
 - **Design - critiquing design**
 - **Support - auto checkin/out from repositories**
 - **Custom - ability to construct user defined agents**
 - **Ie environment extensions**

Serendipity agent specn

- Process oriented filter and action based VL



Tool building tools

- **Need ability to specify:**
 - **Repositories**
 - **Data structures, constraints, persistency**
 - **Views**
 - **Syntax, graphical repns, consistency with repository**
 - **View editors**
 - **Interaction modes, parsing & rendering**
 - **Tool integration**
 - **Scalability & extensibility critical**
 - **Scripting support**

JComposer/Build By Wire

- Used to specify and generate JViews-based environments

User Interface

The image displays the JComposer/Build By Wire (BBW) software interface. On the left, there are two UML diagrams. The top diagram, titled 'Basic Use Case Shapes', shows a class hierarchy with 'Actors' and 'Use Cases' as abstract classes, and 'Repository' and 'BaseActor' as concrete classes. The bottom diagram, titled 'Basic View Icons & Mappings', shows a class hierarchy for view components, including 'Diagram', 'ActorIcon', 'ViewChanges', and 'ViewComp'. On the right, a 'BBW Application' window shows a Java Swing GUI with various components like 'JPanel2', 'JPanel6', and 'JTextField9'. A 'com.sun.java.swing.JPanel' dialog box is open, showing properties like 'foreground', 'background', 'font', 'alignment', 'opaque', and 'doubleBuffered'. Below the diagrams, there is a 'Shape: Move' dropdown menu.

Base CPRG

View Mappings

+ Backend code generator

Assessment

- Criteria for picking tools
- **Synergy** between **development process and tools**
 - Do tools fit process
- **Appropriate tool feature set**
 - Eg complex middleware support or embedded systems need specialised tools
- **Integration and extensibility**
 - large projects need ability to integrate addnl tools
 - General data exchange format support for portability to new tools
 - Ability to **tailor** tool
- **Usability**
 - Difficult using traditional usability approaches
 - Cognitive Dimensions approach useful here
 - Mostly focuses on UI usability

Summary

- **Have looked at:**
 - **Types of software tools**
 - **Architectures for integrating tools together**
 - **Support infrastructure for eg CSCW and tool automation**
 - **Tool building tools**
 - **Tool assessment**
- **Next lecture focus on the area of integrated software development environments (ISDEs)**