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 & DSVLs (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

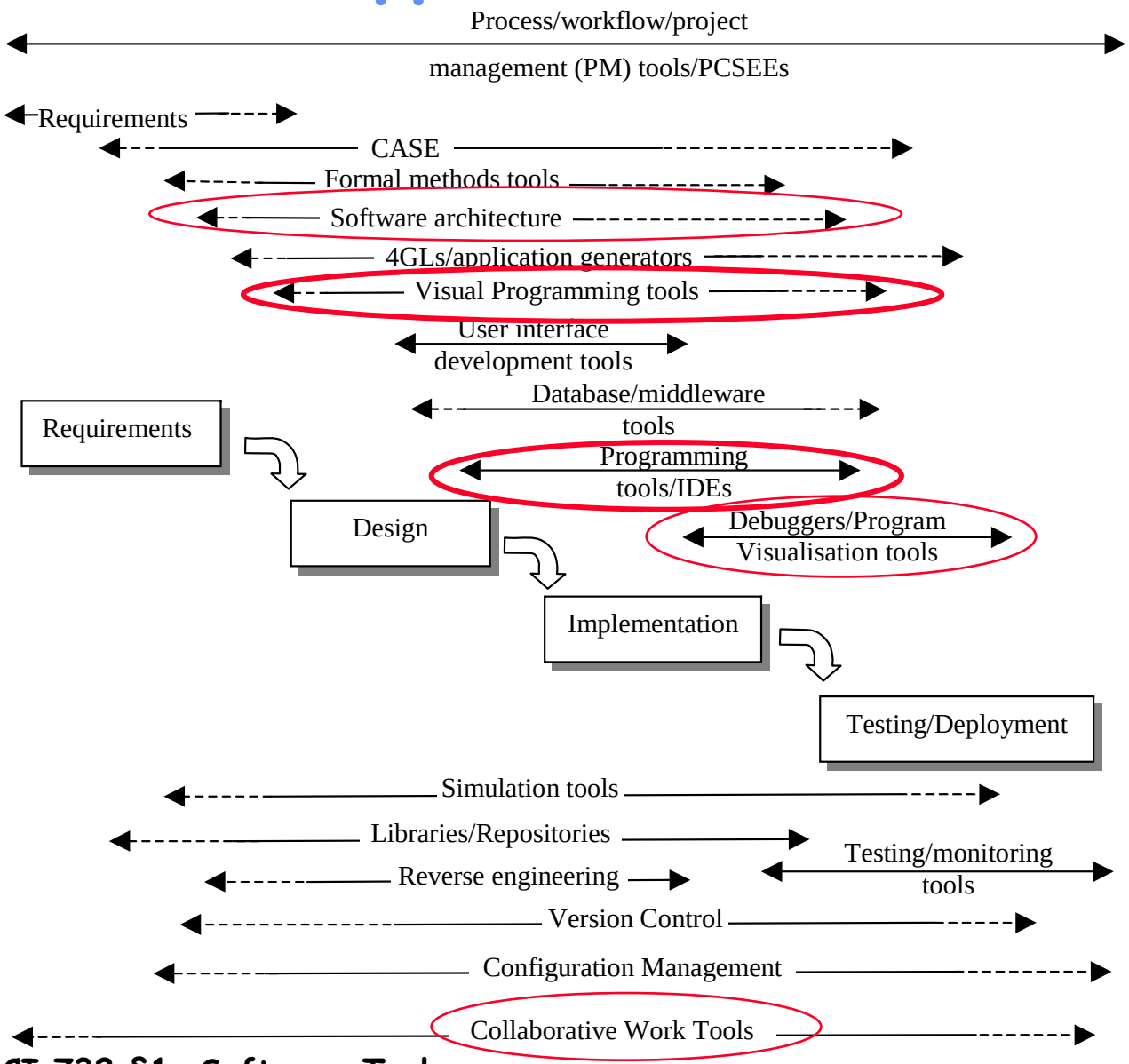
Software Tools

- 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

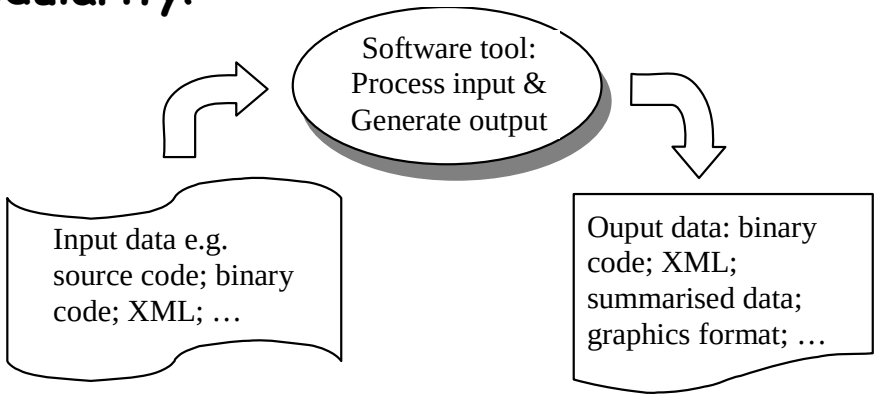
- Rapid change in software development practice in recent times:
  - Newer development methodologies, eg RAD, XP/Agile development, Open Source development, that focus on iterative & collaborative development
    - Need for round trip engineering support
    - Need for collaboration support
  - New technologies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

Types of tool



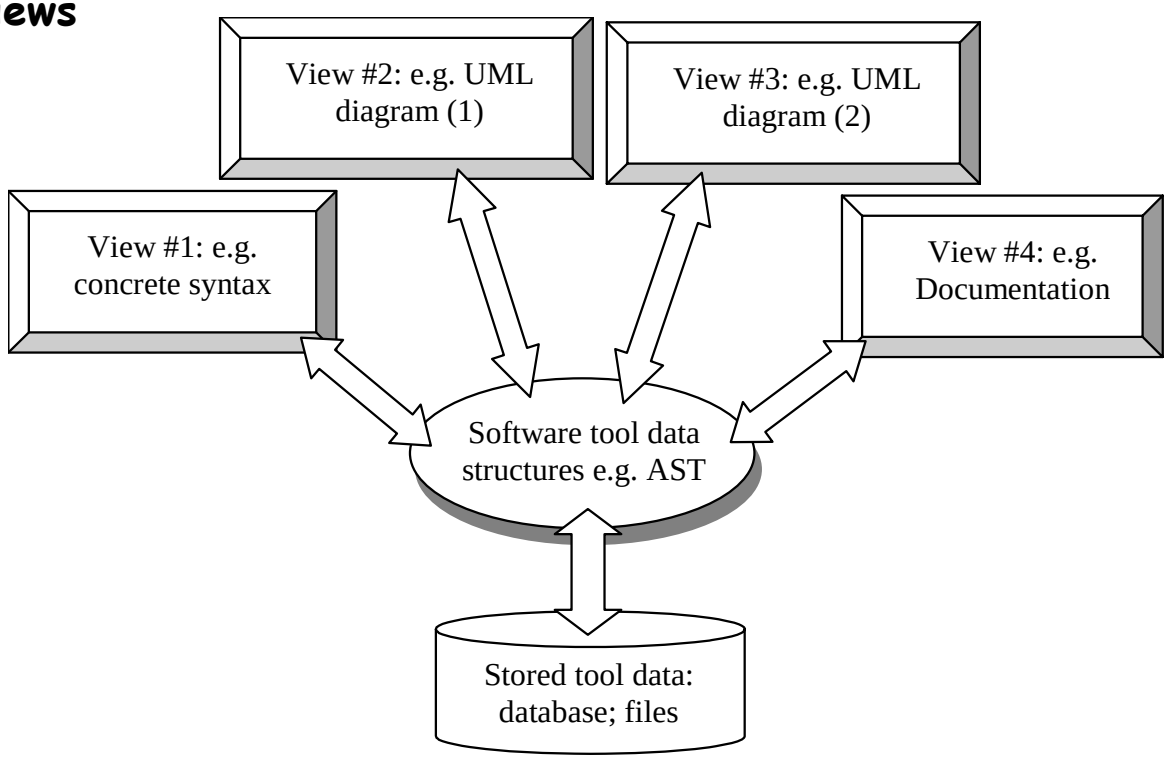
Software Tool Structure

- 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!



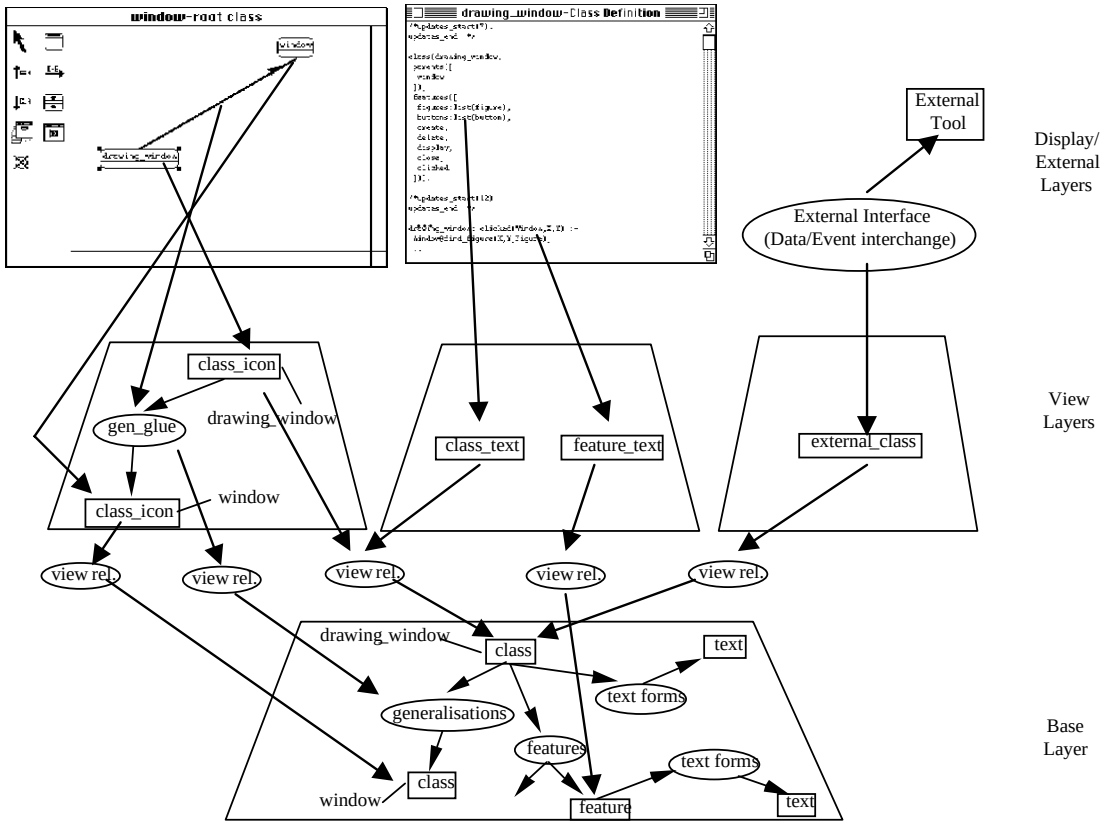
Software Tool Structure

- 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



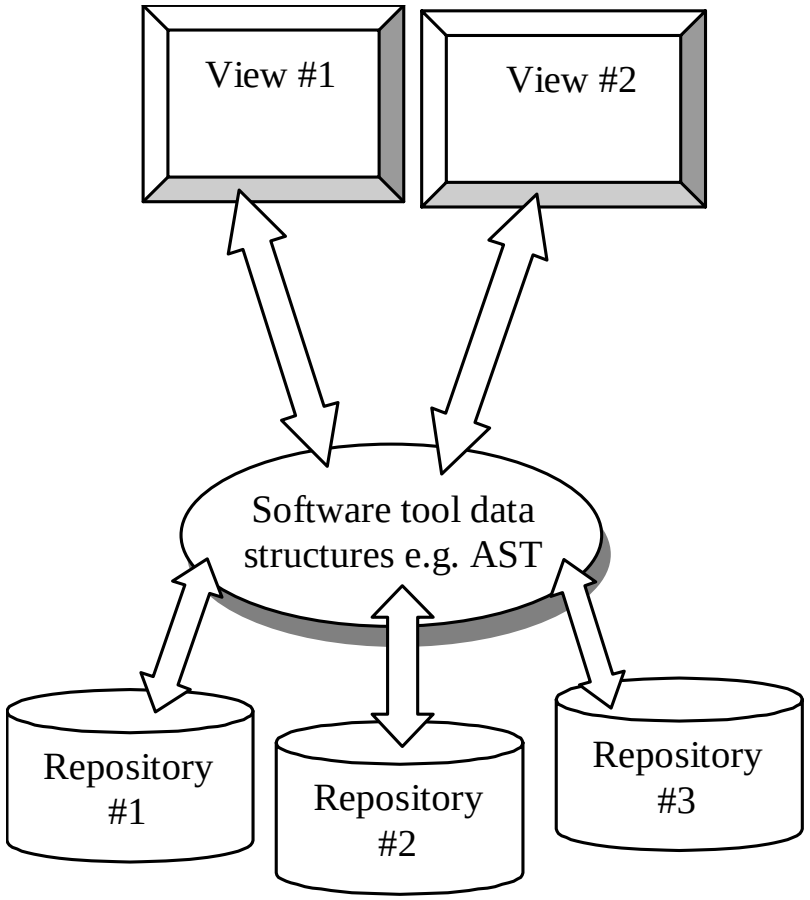
MViews/JViews work

- **Multiple views, multiple notations, shared repository using custom file storage**



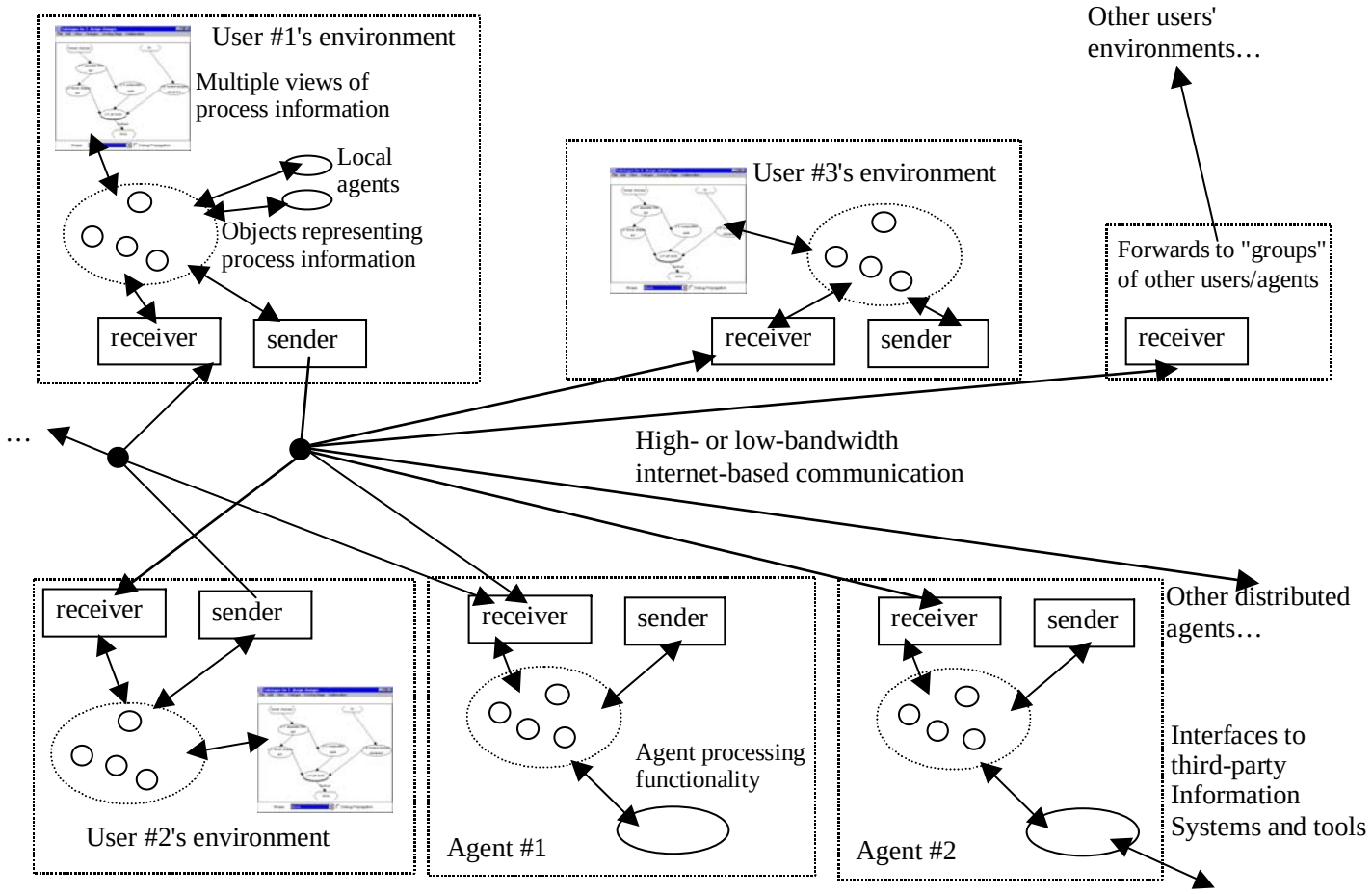
Software Tool Structure

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



Serendipity II

- **Decentralised process modelling**

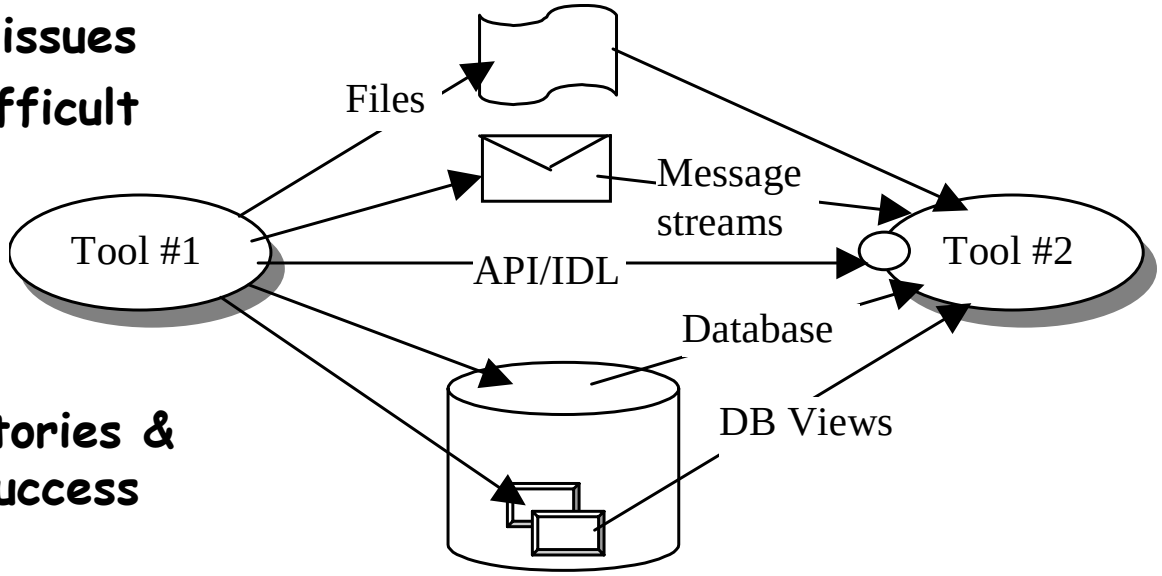


Tool integration

- 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

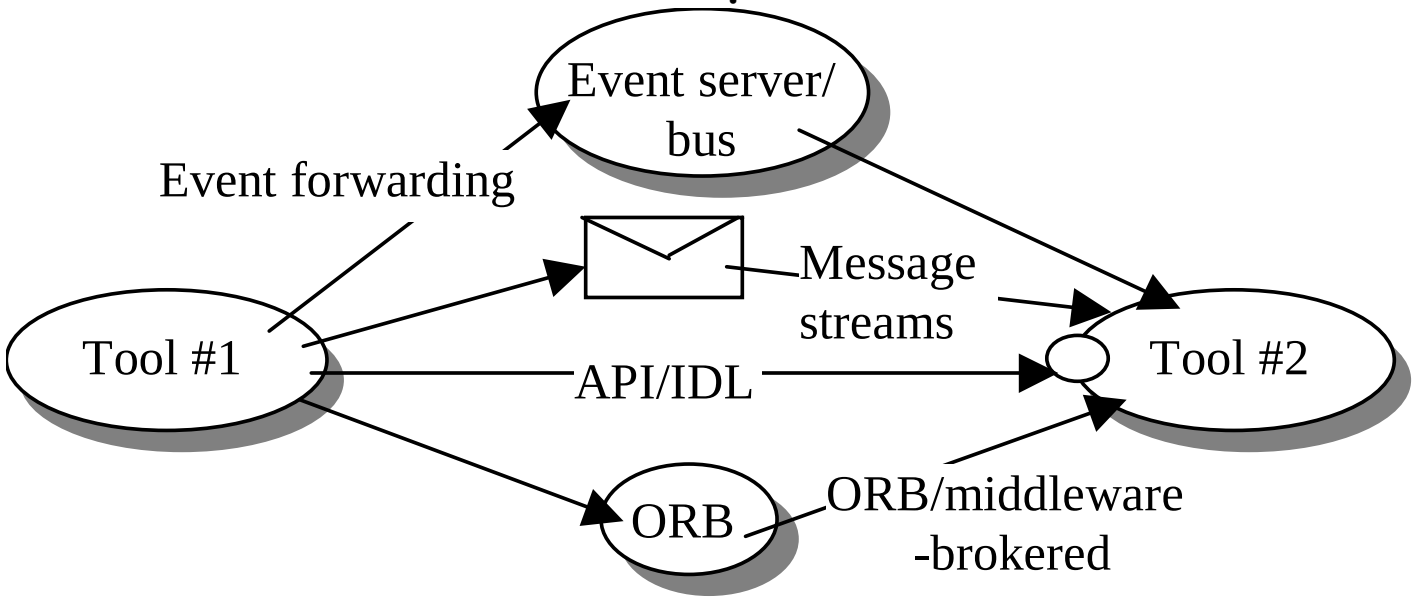
Data integration

- 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

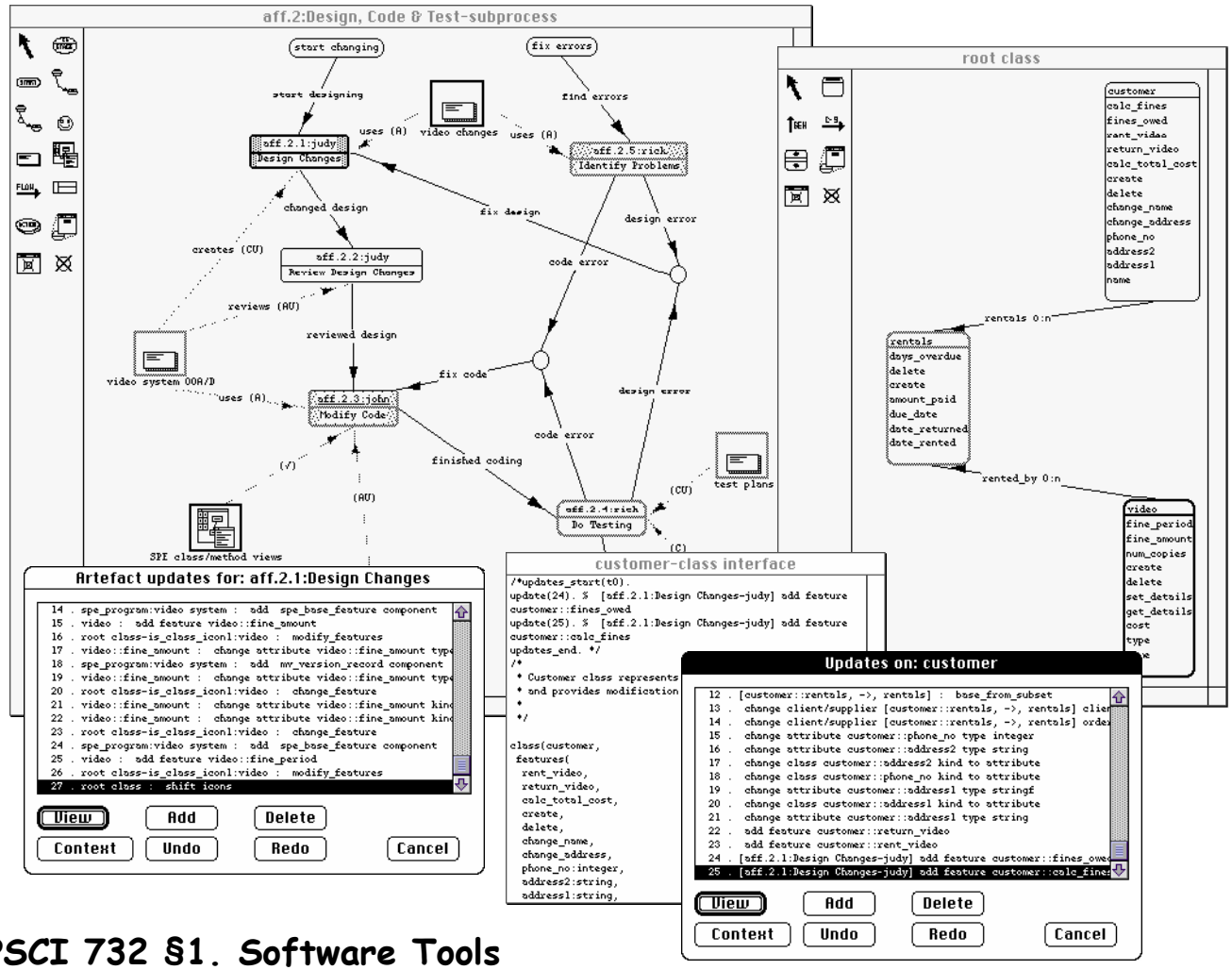


Control integration

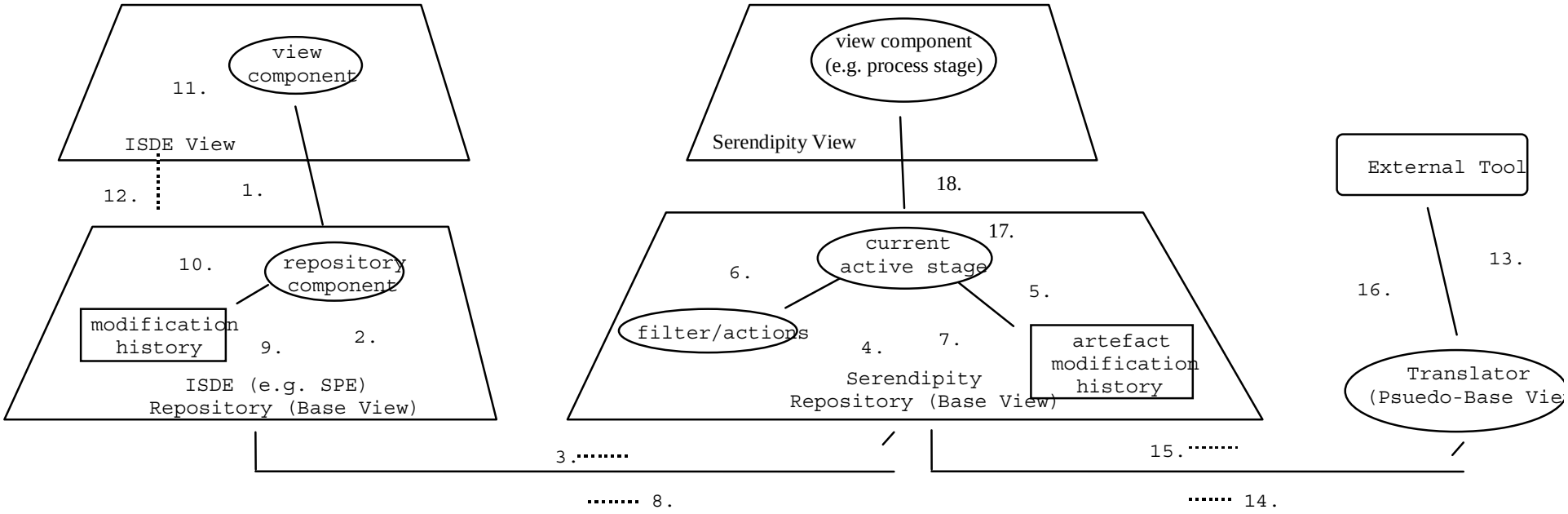
- Variety of approaches
  - Message-oriented using central message broker (eg Field, DEC FUSE)
  - Distributed object approaches eg DCOM, CORBA, web services
    - Need for common component APIs



SPE/Serendipity

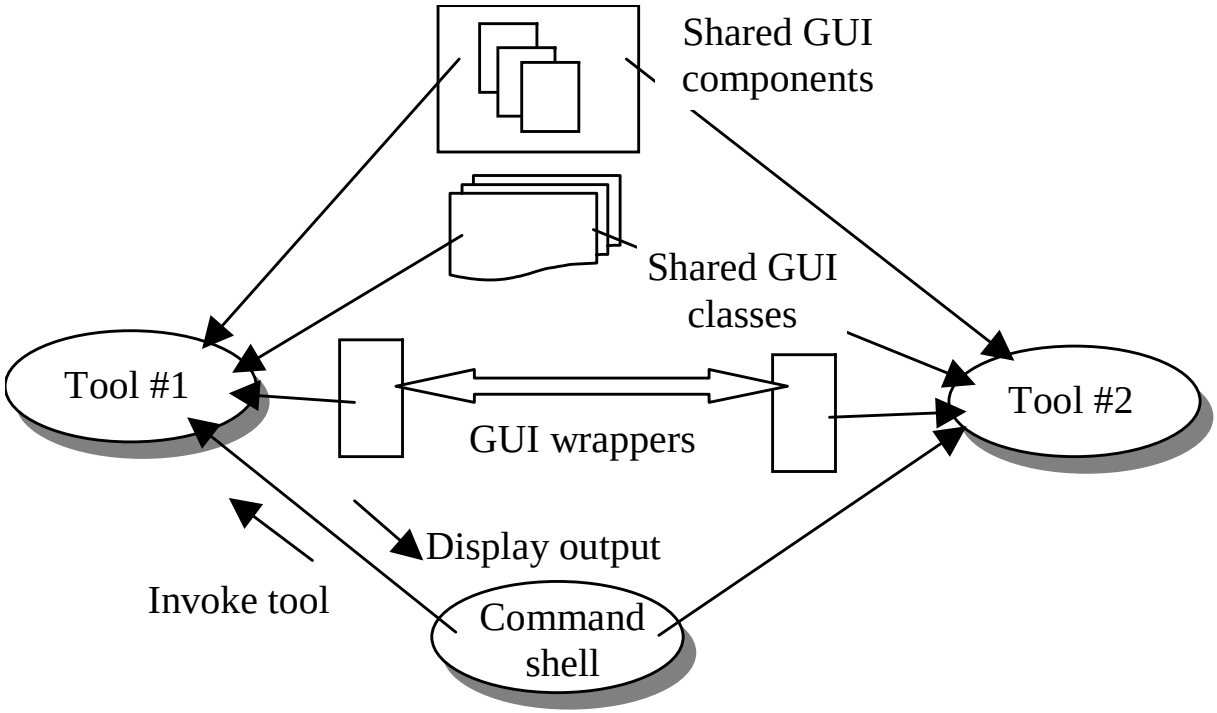


Integration architecture



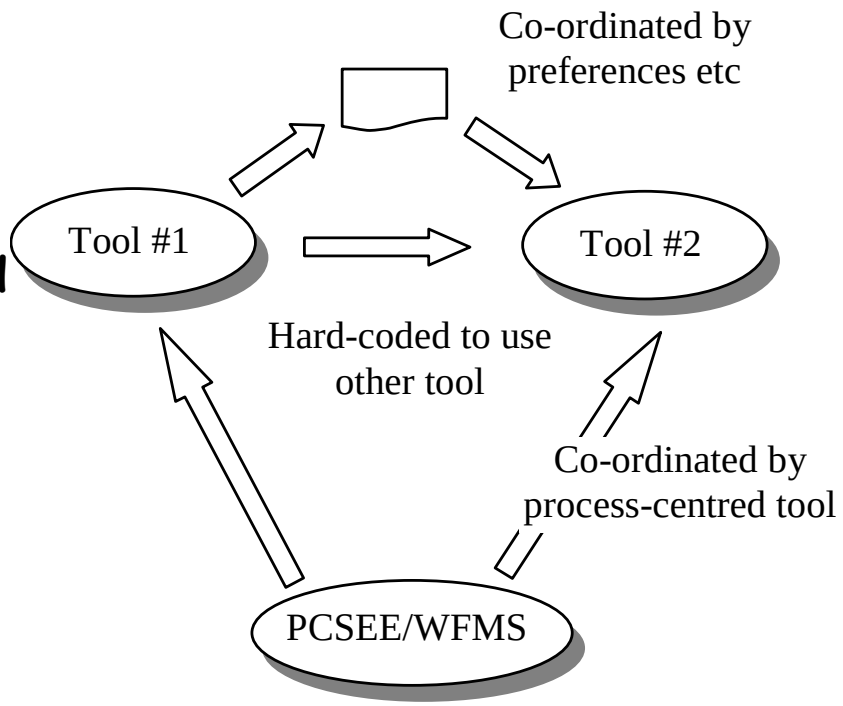
Presentation integration

- 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



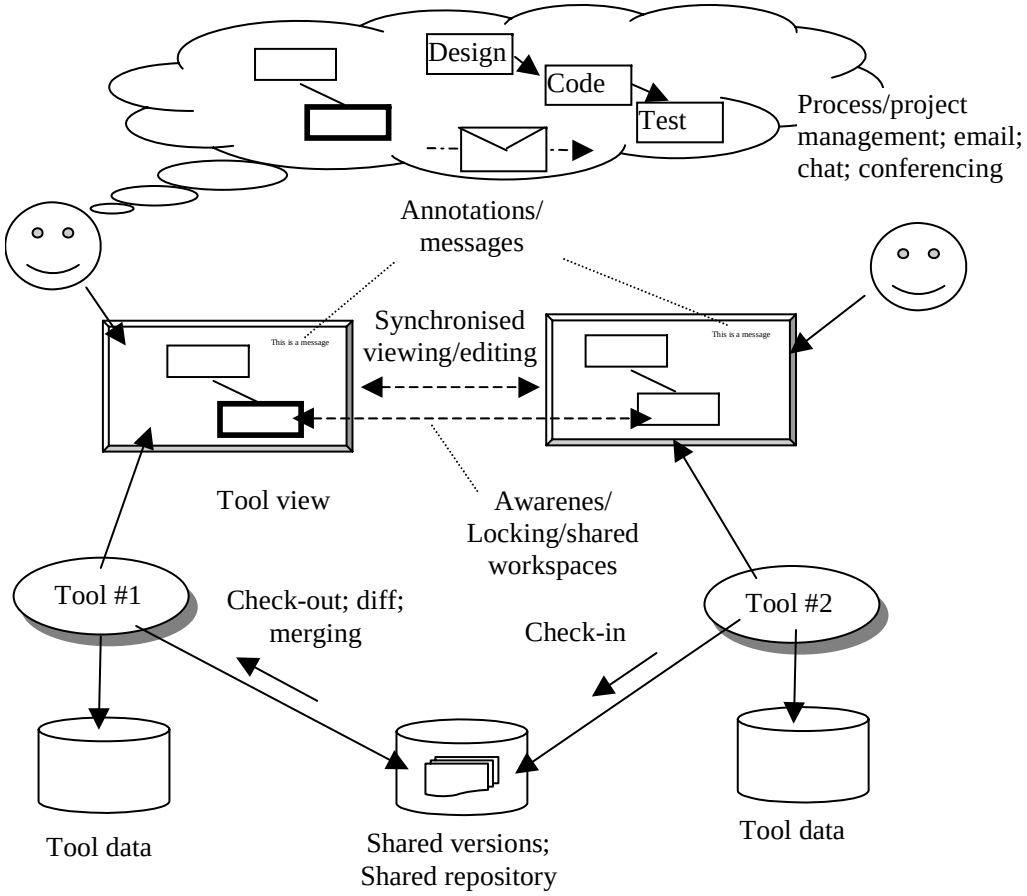
Process integration

- 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

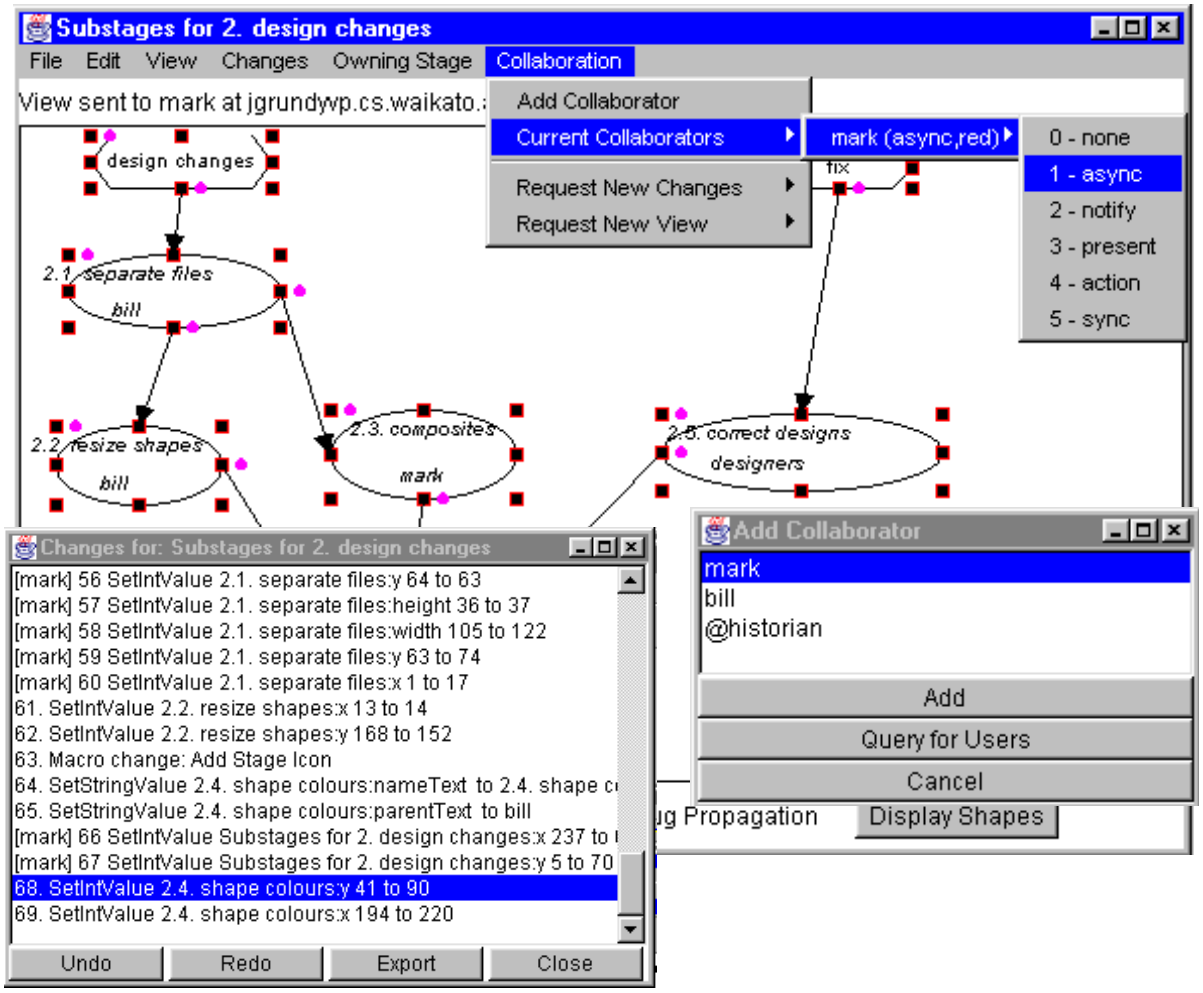


Collaborative work support

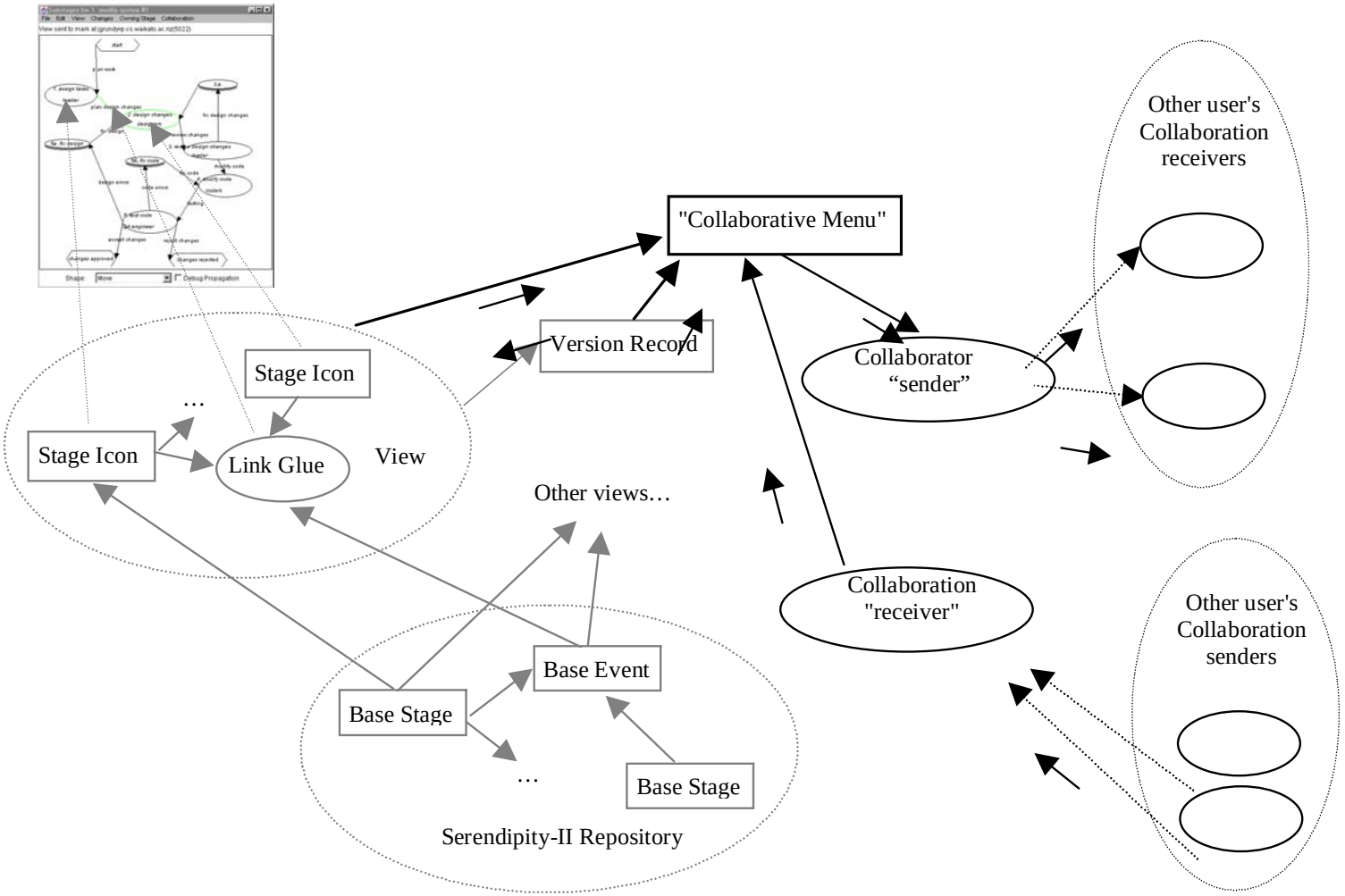
- Builds on tool integration approaches
  - Coordination
    - Project & process mmt
    - Locking of shared artefacts
  - Comms
    - eg chat email video audio
    - Doc annotation
  - Composition
    - Versioning
    - Version merging
  - Synchronous, asynchronous



Sependipity II CSCW



Components

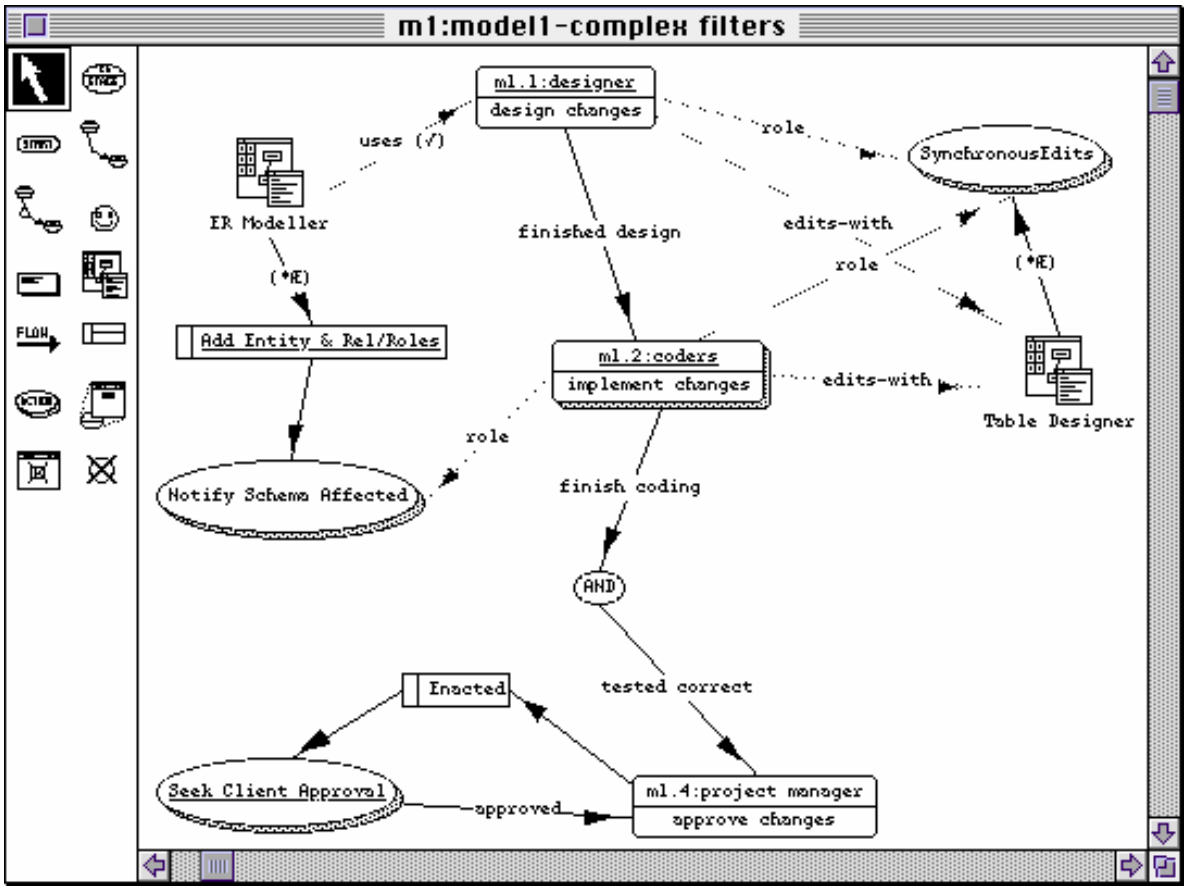

Other user's Collaboration receivers
"Collaborative Menu"
Version Record
Collaborator "sender"
Stage Icon
...
Stage Icon
Link Glue
View
Other views...
Collaboration "receiver"
Other user's Collaboration senders
Base Event
Base Stage
...
Base Stage
Serendipity-II Repository

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



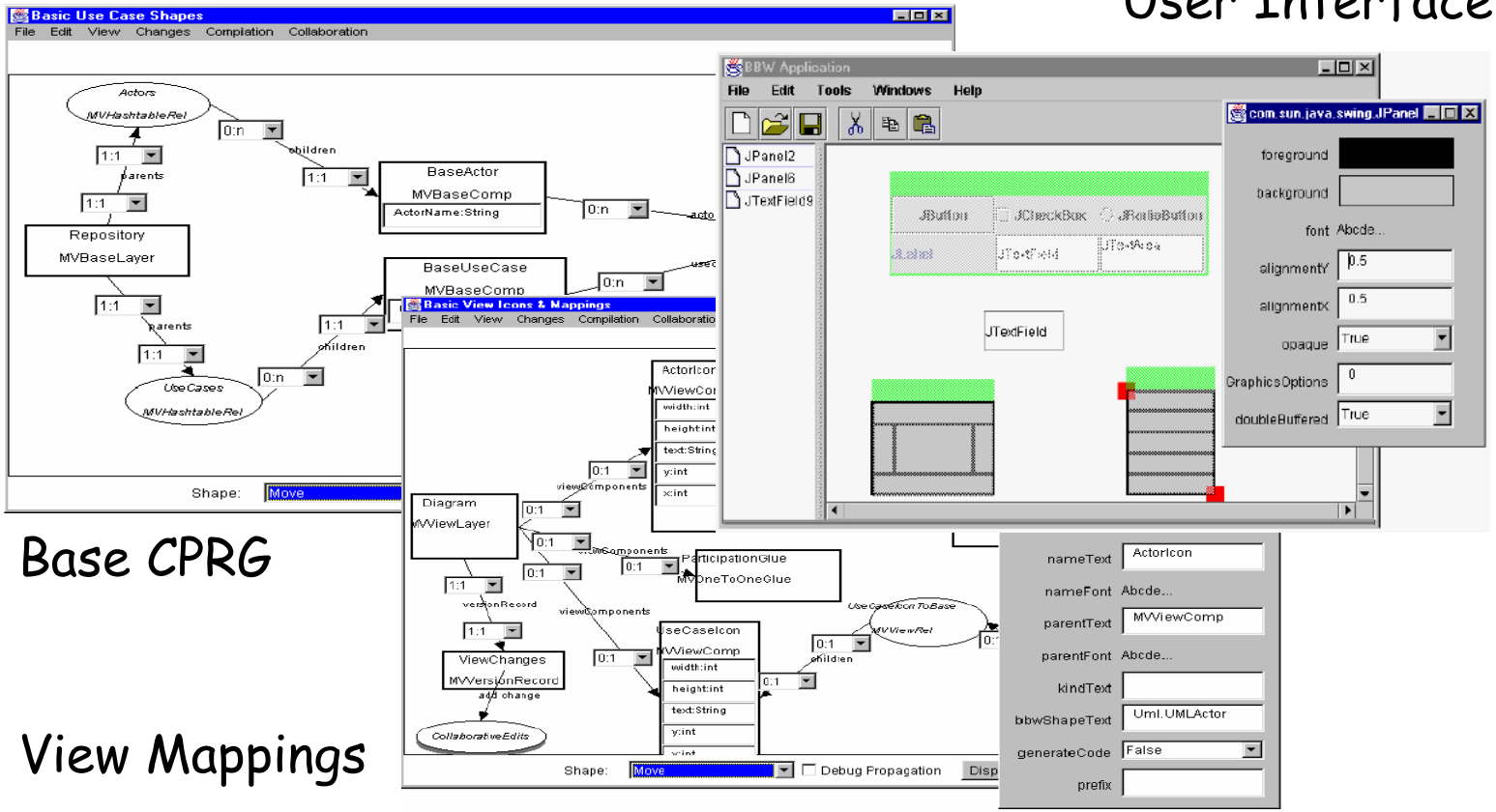
Tool building tools

- 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



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 tolls 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)