732 Software 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)
  - Visual Notations (2)
  - UML + meta modelling (1)
  - Pounamu meta tool (2-3 + Assmt)
  - Other meta tools and tool frameworks (2-3)
  - Ancillary material (2)
- Me: Professor John Hosking
  Room 303.387
  john@cs.auckland.ac.nz

How this section runs

- There is no textbook for this section
- Instead I will be handing out 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

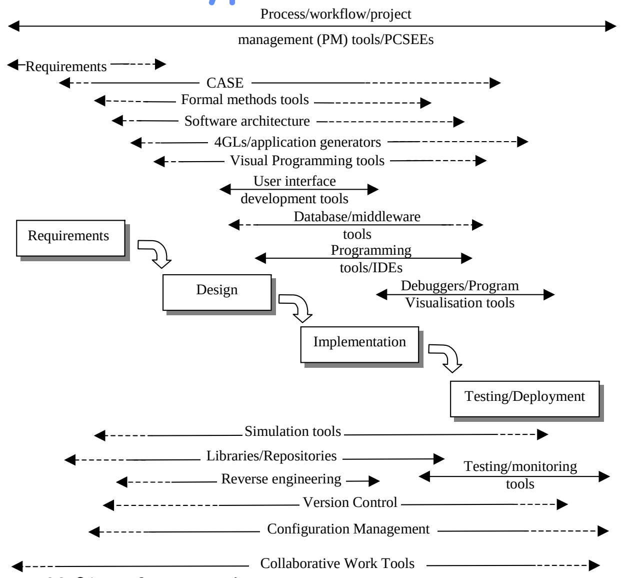
Software Tools

- 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

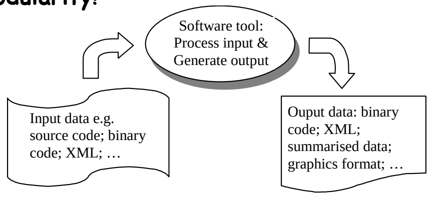
- Rapid change in software development practice in recent times:
  - Newer development methodologies, eg RAD, XP/Agile development, Open Source development, that focus on iterative & collaborative development
    - Need for round trip engineering support
    - Need for collaboration support
  - New technologies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

Types of tool



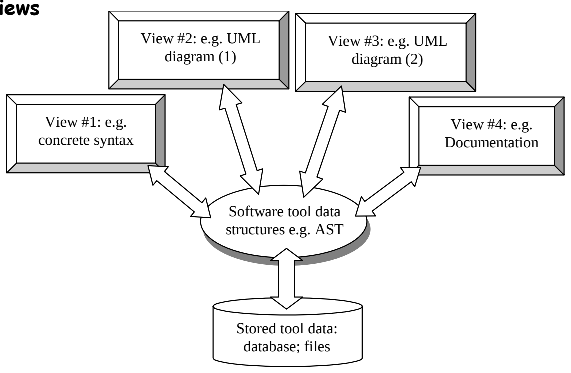
Software Tool Structure

- 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: Process input & Generate output
Input data e.g. source code; binary code; XML; …
Ouput data: binary code; XML; summarised data; graphics format; …

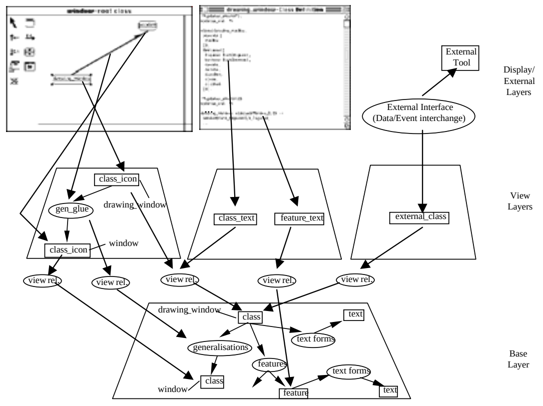
Software Tool Structure

- 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


View #2: e.g. UML diagram (1)
View #3: e.g. UML diagram (2)
View #1: e.g. concrete syntax
View #4: e.g. Documentation
Software tool data structures e.g. AST
Stored tool data: database; files

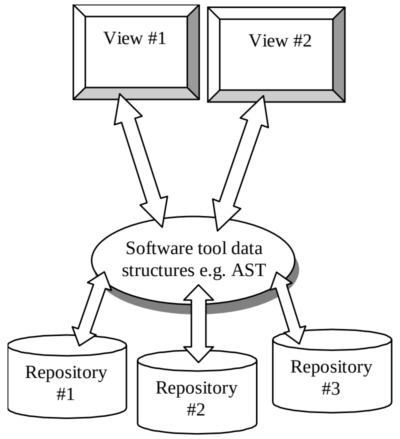
MViews/JViews work

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



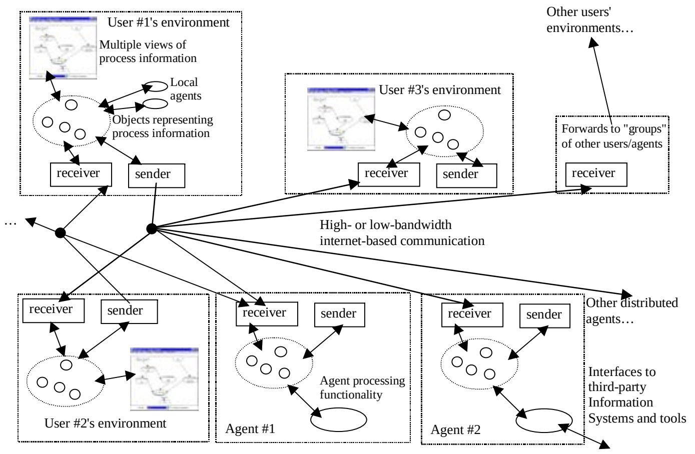
Software Tool Structure

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



Serendipity II

- Decentralised process modelling

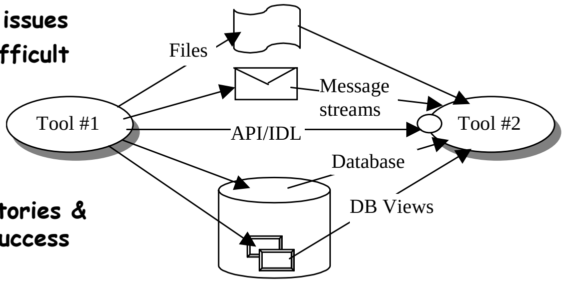

User #1's environment
Multiple views of process information
Local agents
Objects representing process information
receiver
sender
User #3's environment
receiver
sender
Other users' environments…
Forwards to "groups" of other users/agents
receiver
…
High- or low-bandwidth internet-based communication
receiver
sender
User #2's environment
receiver
sender
Agent processing functionality
Agent #1
receiver
sender
Agent #2
Other distributed agents…
Interfaces to third-party Information Systems and tools

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

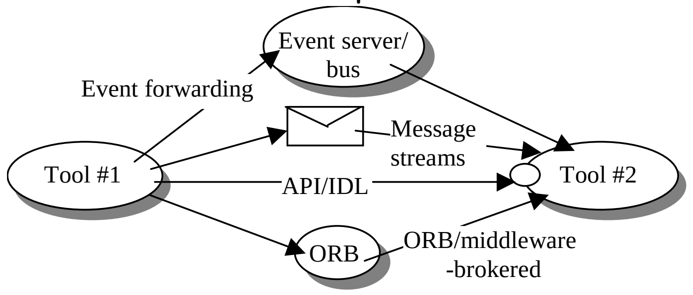
- Data exchange using custom or standard exchange formats via:
  - Need for translators
  - Common formats: UML XMI (OMG), workflow exchange format
- Tighter coupling via shared database but
  - Performance issues
  - Standards difficult



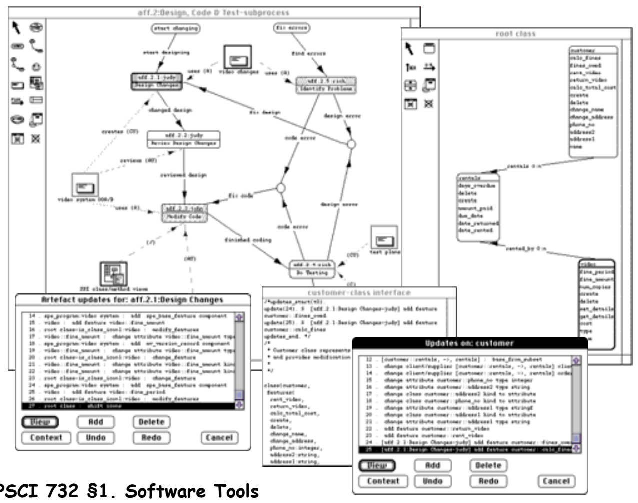
- Document repositories & version control success

Control integration

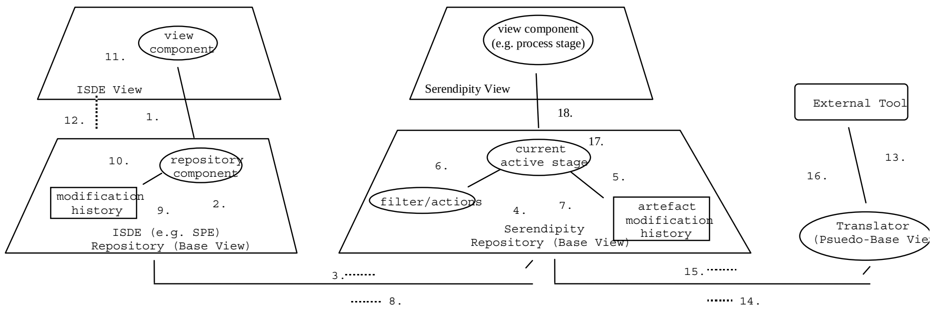
- Variety of approaches
  - Message-oriented using central message broker (eg Field, DEC FUSE)
  - Distributed object approaches eg DCOM, CORBA, web services
    - Need for common component APIs



SPE/Serendipity

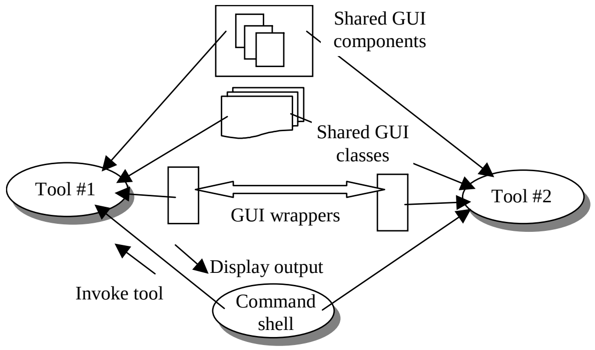


Integration architecture



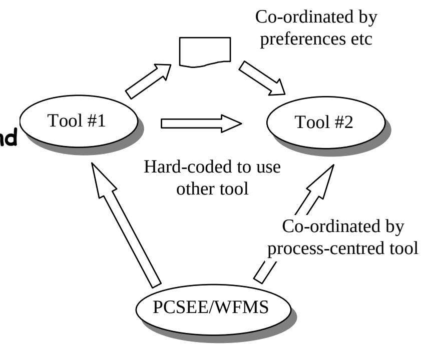
Presentation integration

- 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



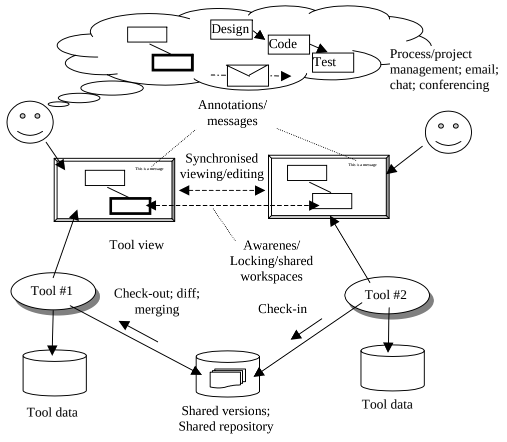
Process integration

- 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

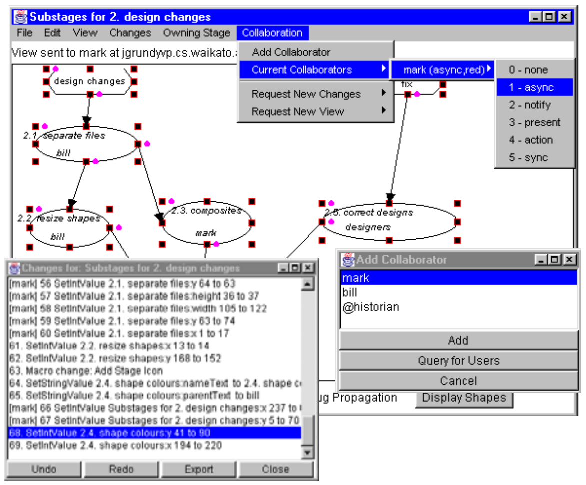

Co-ordinated by preferences etc
Tool #1
Tool #2
Hard-coded to use other tool
Co-ordinated by process-centred tool
PCSEE/WFMS

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


Design
Code
Test
Process/project management; email; chat; conferencing
Annotations/ messages
Synchronised viewing/editing
Tool view
Awarenes/ Locking/shared workspaces
Tool #1
Tool #2
Check-out; diff; merging
Check-in
Tool data
Shared versions; Shared repository
Tool data

Sependipity II CSCW

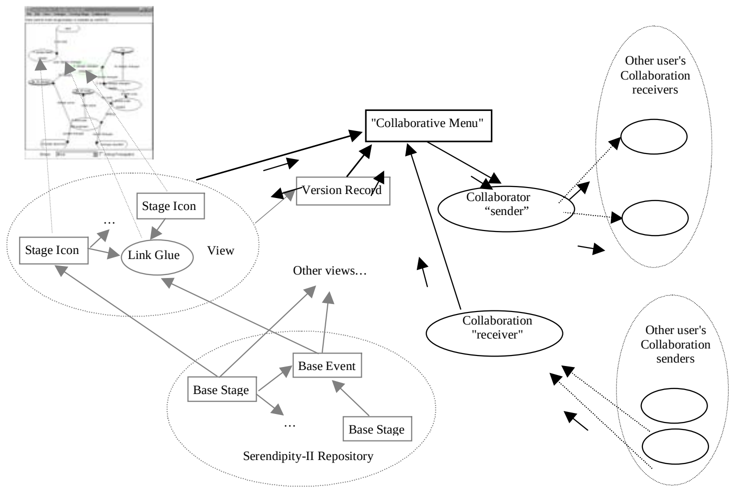


View sent to mark at jgrundywp.cs.waikato.

Changes for: Substages for 2. design changes

```
[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
[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
```

Components

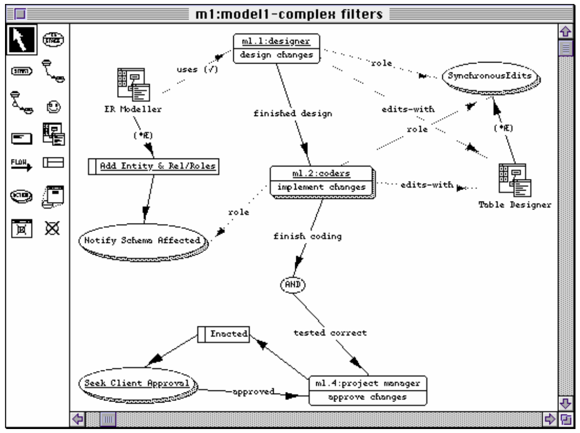


Tool automation

- 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



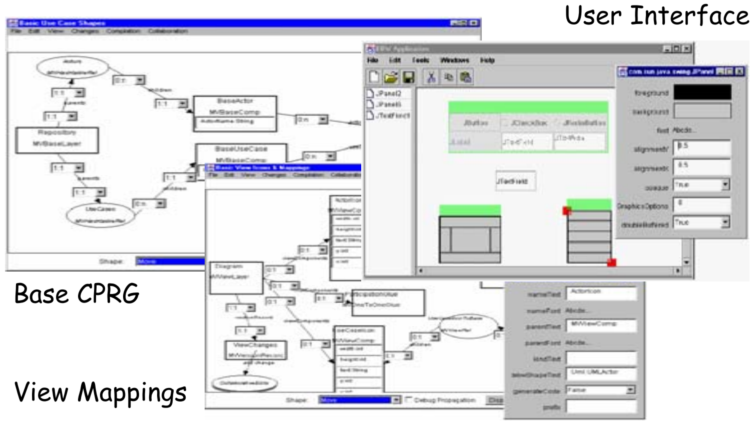
Tool building tools

- 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



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 visual notations