

Component-based Methods, Architectures and Tools



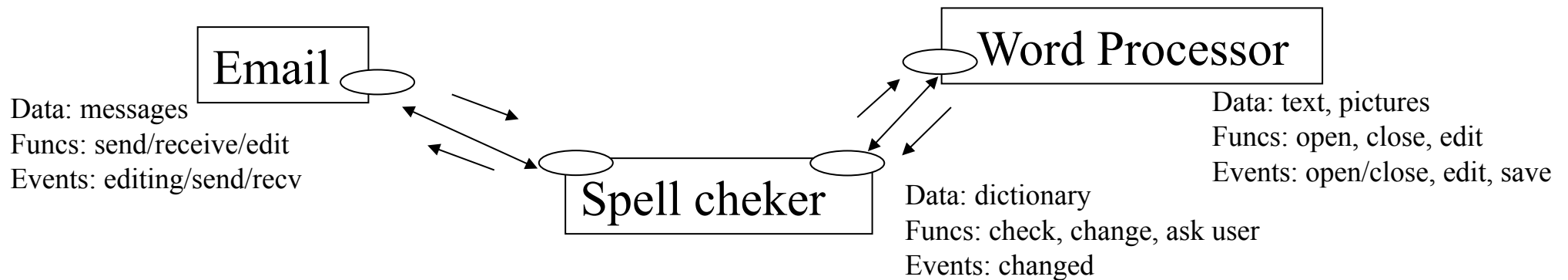
John Grundy
Dept of Computer Science
University of Auckland

Overview

- What are component-based systems???
- Recent UoA work on component-based systems:
 - JViews
 - Jcomposer & Serendipity-II
 - Aspect-oriented Component Engineering
 - SoftArch
- Putting it all together...
- What does the future hold?

Software Components

- Idea of discrete, “pluggable” software components:



- Isolate functions/non-functional characteristics
- Interact via well-defined interfaces/events
- Compose to form systems (sometimes end users!)
- Domain-specific & reusable...

Our Use of Components



Multi-user design environments

Groupware & E-commerce Applications

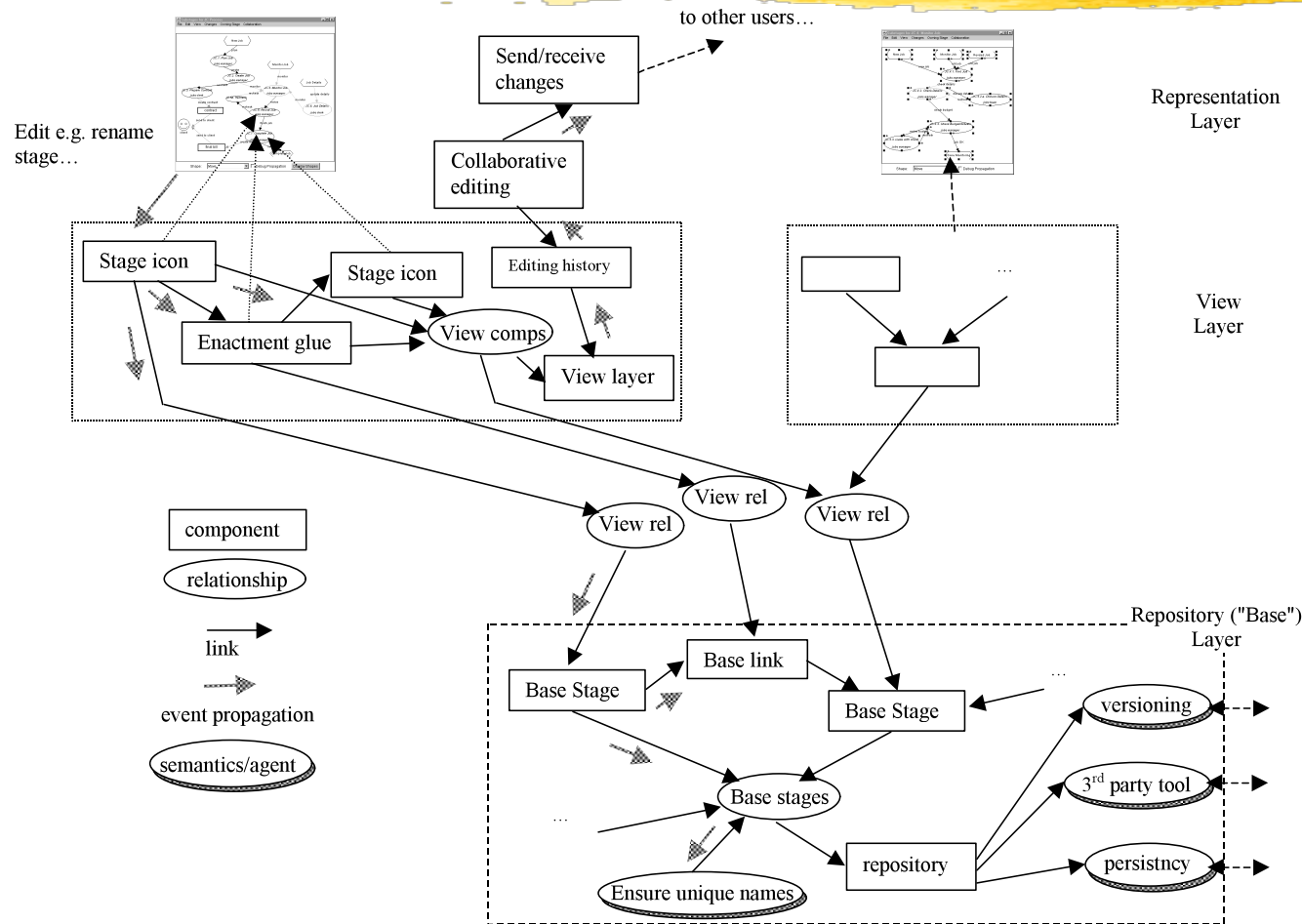
Use components to design & implement...

⇒ **Need Component Framework...**

JViews Framework

- Architecture for building event-based software engineering tools (originally, anyway...)
- Abstractions:
 - uses extended JavaBeans component model
 - multiple view support
 - repository, distribution support
 - multi-user support
 - extensible user interfaces
 - limited tool integration support
 - many reusable components from framework

JViews Architecture Example



- JViews structure of Ser-II tool
- Comps for repository (model); views; collaboration; persistency; tool integration etc

Tool Support

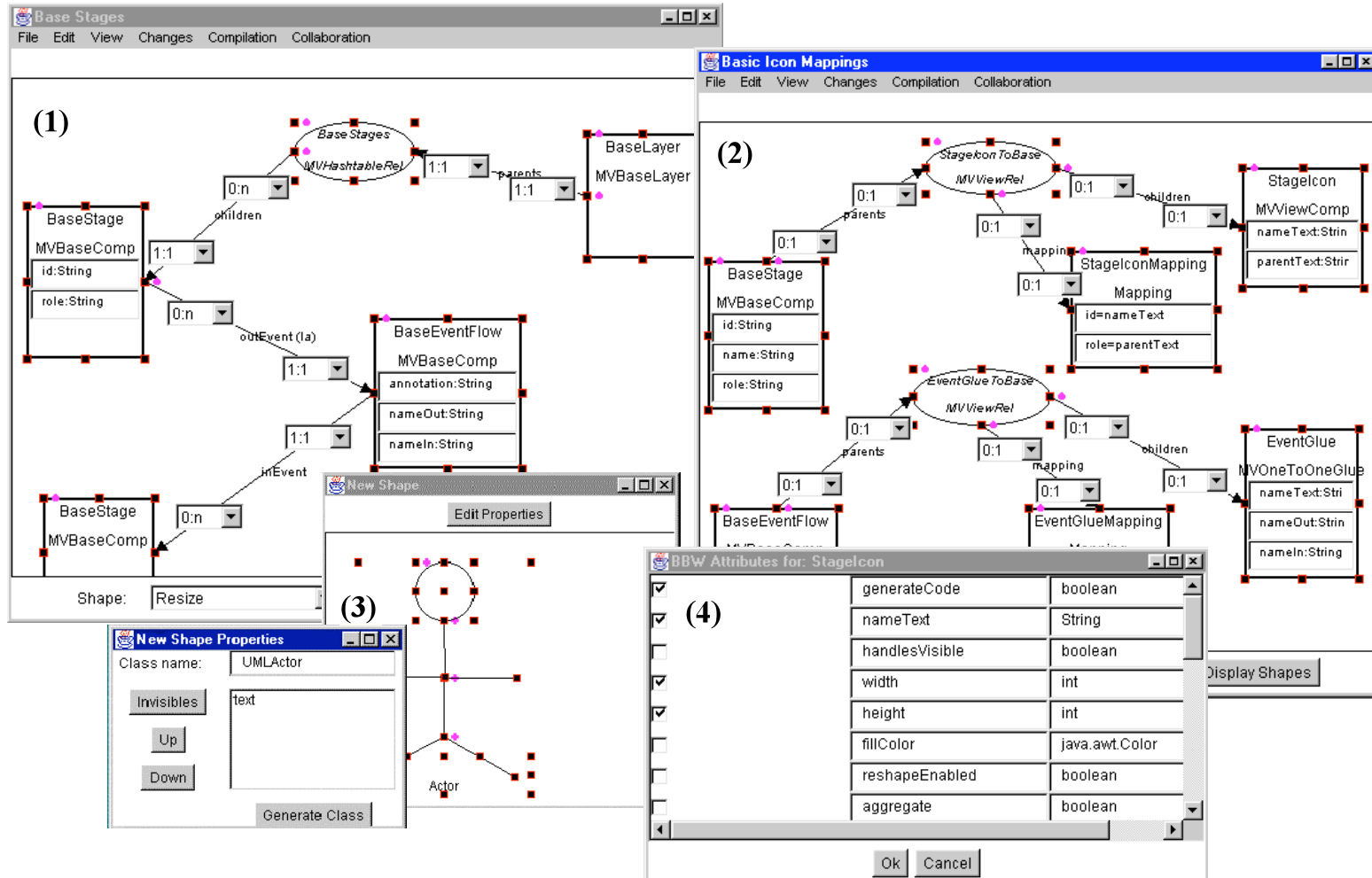


Building with Jviews hard -
need some tools to help...

⇒ **Jcomposer & BuildByWire - metaCASE
Tools**

⇒ **Serendipity-II - Process Support Tool**

JComposer/BuildByWire



Serendipity-II

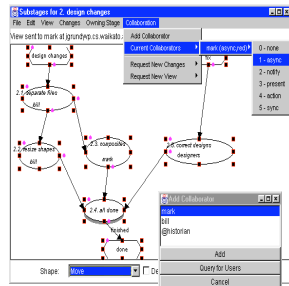
- Process modeling & enactment environment
- Process modeling:
 - multiple, visual views (overlapping & hierarchical)
 - multiple user editing support
 - event processing visual language
- Process enactment:
 - decentralised enactment engine; view highlights
 - decentralised to-do lists, task automation (“agents”)
 - tool integration

Ser-II: Example of Use

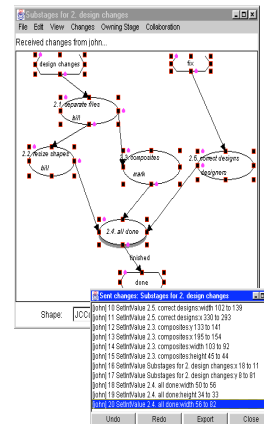
The screenshot displays a complex software development process visualization tool with several windows:

- Simple Software Process #1:** Shows a sequence of stages: 1. plan changes (by john), 2. design changes (by mark), 3. review changes (by john, mark), 4. code changes (by bill), and 5. retest. Transitions between stages are labeled with terms like 'plan changes', 'errors', 'review', 'code', and 'test'.
- Remember enactments:** A diagram showing the flow from 'Enacted Stage' to 'Events' and then to 'Store Change'.
- Changes for: Simple Software Process #1:** A list of changes including:
 - 56. SetStringValue 5b:text to 5b
 - 57. SetStringValue 5b:parentName to OR
 - 58. Macro change: Add Event Glue
 - 59. SetStringValue errors:nameText to errors
 - 60. Macro change: Add Event Glue
 - 61. SetStringValue errors:nameText to errors
 - [john] 62 Macro change: Move 5. retest
 - [john] 63 Macro change: Move 4. code changes
 - [john] 64 Macro change: Resized window
- Enacted Stages:** A list of stages: mark, *3. review changes, john, 3. review changes, *1. plan changes.
- Enactment history for :3. review changes:** A list of actions:
 - 0. EnactedStage: 3. review changes(by john, review,), 2. d
 - 1. FinishedStage: 3. review changes(by john, john, errors),
 - 2. EnactedStage: 3. review changes(by john, review,), 2. d
 - 3. EnactedStage: 3. review changes(by mark, mark,),
 - 4. MakeCurrentStage: 3. review changes(by john, john,),
 - 5. MakeCurrentStage: 3. review changes(by mark, mark,), do r

Collaborative Work...



(a) John's view



(b) Mark's View

Other applications...

The screenshot displays a web-based application interface. On the left, a 'File Tree' window shows a hierarchical structure of itinerary items: 'Root', '20th November 1998: Sydney to Melbourne' (with sub-items 'QF412 10:30am' and 'Stay in: Hotel Alexander, Melbourne'), and '22nd November 1998: Melbourne to Adelaide'. Below the tree is a text field labeled 'Itinerary for:' containing the name 'John Grundy'. A map of Australia is shown with red lines indicating flight paths between Sydney, Melbourne, and Adelaide. The main window, titled 'Qantas Airways Limited - Netscape', displays a 'Qantas Daily Schedules Display' for 'Adelaide - Perth' on '1000 hrs Thursday 26 November 1998'. It contains a table with flight details:

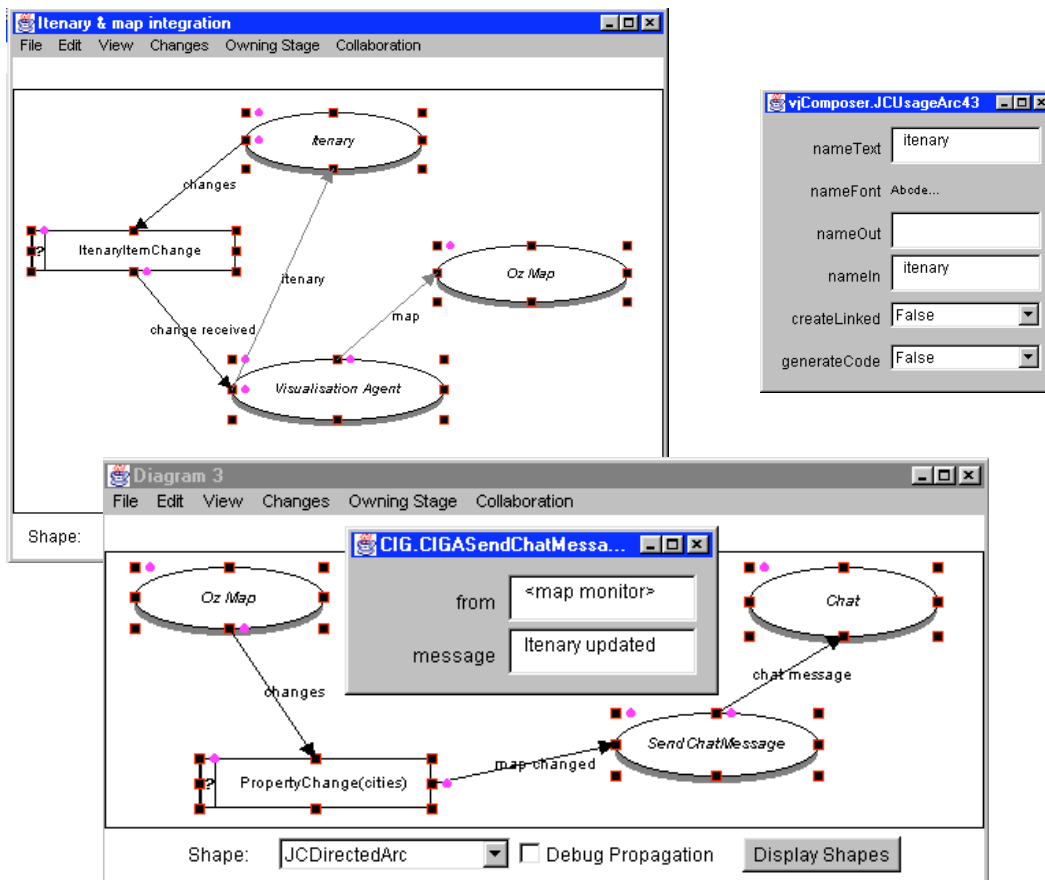
Flight Number	Departure		Arrival		Aircraft Type	Number of Stops
	City	Time*	City	Time		
561	Adelaide	0820	Perth	0905	733	0
499	Adelaide	1340	Perth	1425	734	0
899	Adelaide	1840	Perth	2055	141	1

Below the table, it notes 'QF 899 OPERATED BY AIRLINK'. A 'collaborative chat' window is open, showing a conversation between 'john' and 'mark' discussing flight options and updates to the itinerary.

- Collaborative travel itinerary planner
- Textual & visual views
- Collaboratively edit
- Built by composing comps in Ser-II

Specification in Serendipity-II

- Create/link various components
- Event handling VL from Ser-II used
- Can co-ord usage with Ser-II process models...



Component Development



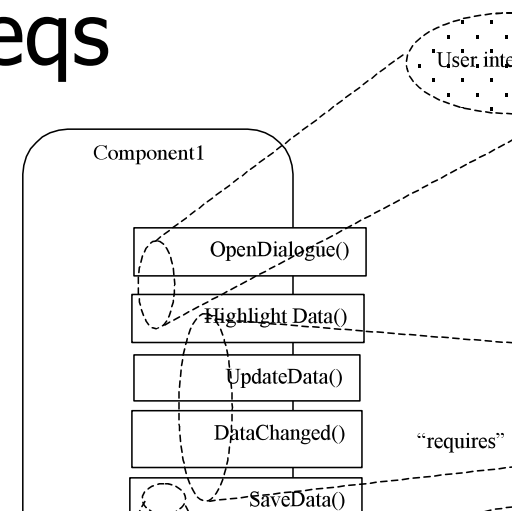
Developing Components Hard:

Requirements → Design → Code → Deployment

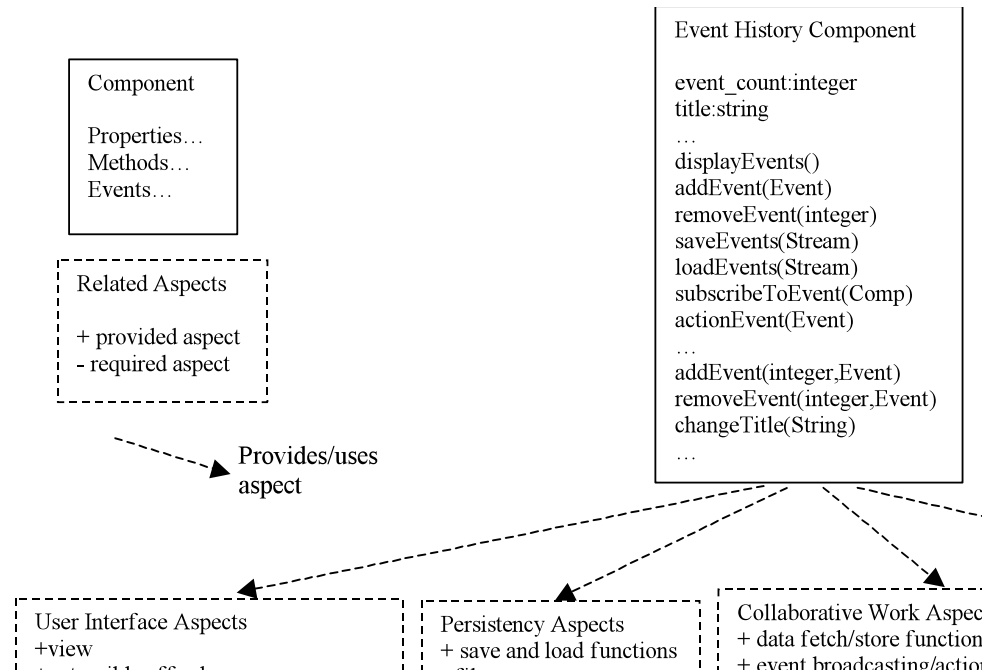
⇒ **Aspect-Oriented Component Engineering**

Aspect-oriented Component Engineering Methodology...

- Systemic perspectives on component func/non-func reqs
- Capture data, func, non-functional information
- Idea of provided & required aspects
- Often overlap
- Various kinds of aspects...



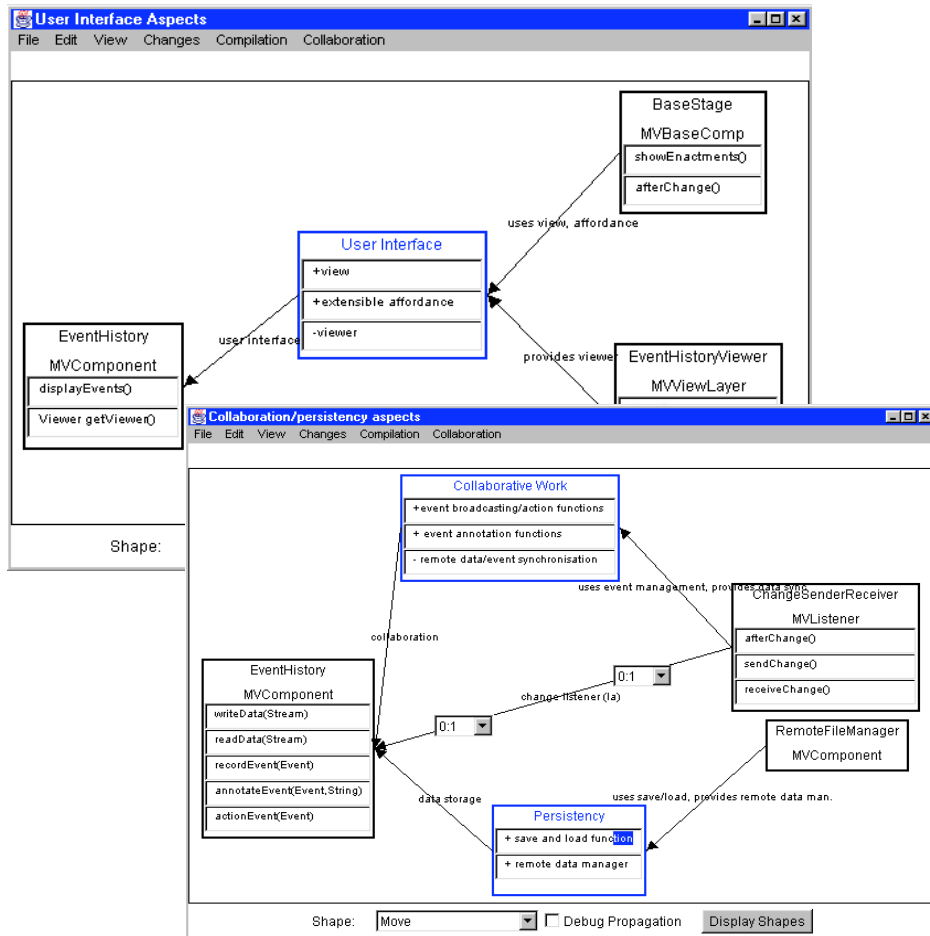
Example: JViews Event History



Implementation of Aspects

- Extended our JViews framework to support implementation of components using aspects
- Use to guide component interface & links
- Codify in component implementation for run-time use by users/other components
 - knowledge about component facilities available to users
 - aspect codification provides set of functions to examine aspects, set of patterns for component reconfiguration etc.

Tool Support: Jcomposer+; Repository & Query IF



The image shows three screenshots from the Jcomposer+ tool. The top screenshot is the "Simple Component Repository" window, displaying a list of components such as "extensible affordance", "event generation", "locking", "versioning", "encode data", "decode data", "query data", "store data", "retrieve data", "version data", and "distribution". A "GENERATE" button is highlighted, and a "Chat Server" component is visible in the background. The middle screenshot is a "Validation error for" dialog box with the text: "Warning: event history relationship not established: Need event history linked so can do querying/retransmission". The bottom screenshot is the "Aspects for:" dialog box, showing tabs for "Properties", "Relationships", "Configuration settings", and "Human interfaces", with "Relationships" selected. It includes buttons for "More Info", "Validate", and "Close".

Architectural Support



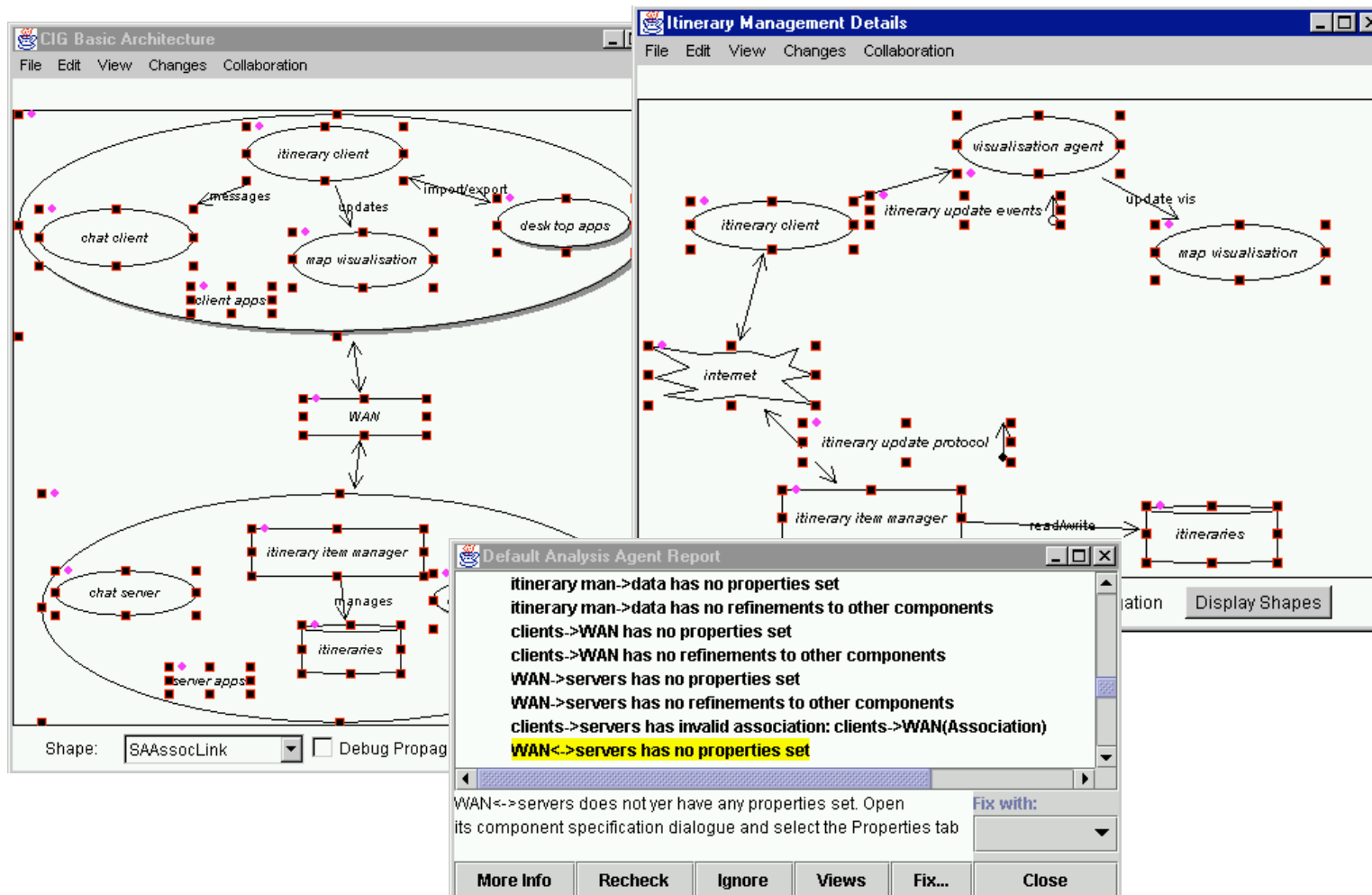
Lacked Good Software Architecture Support

⇒ **SoftArch Tool**

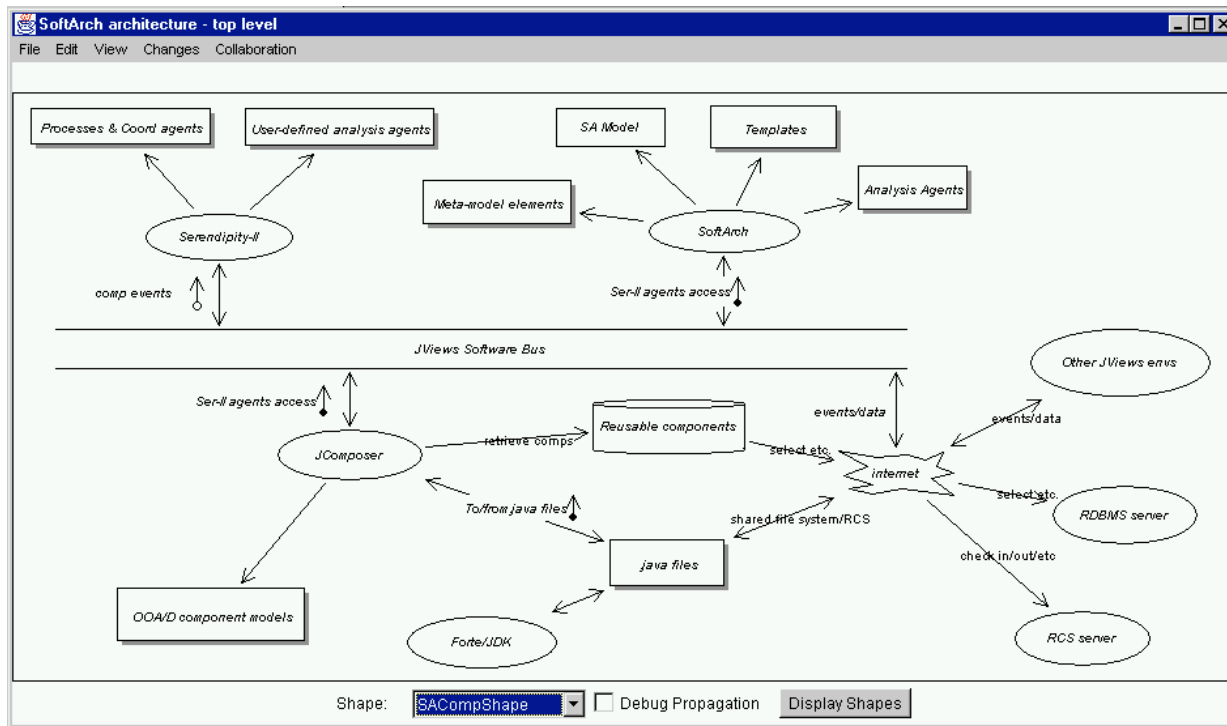
Need to put tools together...

⇒ **Integration using Jviews software bus**

SoftArch Example



A Distributed Component Engineering Environment...



- Serendipity-II:
 - processes/agents
- SoftArch:
 - High-level component groupings
- JComposer:
 - CASE/impl.
- JVisualise:
 - debugging
- Component Library:
 - reuse
- Others (DB, RCS, Forte)

Summary



- Component-based architectures:
 - work well for SEEs
 - Jviews->Jcomposer->AOCE->SoftArch->??
- Future Work:
 - Heterogeneous component systems?
 - Improved architecture/component abstractions
 - Many areas of tool enhancement
 - Further exploit agents, aspects, repositories, distribution, open systems platforms, ...

Selected References

Grundy, J.C. Multi-perspective specification, design and implementation of software components using aspects, *International Journal of Software Engineering and Knowledge Engineering*, Vol. 10, No. 6, December 2000.

Grundy, J.C., Mugridge, W.B. and Hosking, J.G. Constructing component-based software engineering environments: issues and experiences, *Journal of Information and Software Technology*, Vol. 42, No. 2, January 2000, pp. 117-128.

Grundy, J.C. Visual specification and monitoring of software agents in decentralised process-centred environments, *International Journal on Software Engineering and Knowledge Engineering*, Vol. 9, No. 4, World Scientific Publishing Company, August 1999, pp. 425-444.

Grundy, J.C., Hosking, J.G., Mugridge, W.B., Apperley, M.D. A decentralised architecture for software process modelling and enactment, *IEEE Internet Computing*, Vol. 2, No. 5, IEEE CS Press, September/November, 1998, pp. 53-62.

Grundy, J.C. *A method and support environment for distributed software component engineering*, In Proceeding of the 2000 International Conference on Software – Methods & Tools, Wollongong, Australia, Nov 6-10 2000, IEEE CS Press, pp.157-166.

Grundy, J.C. and Hosking, J.G. *High-level Static and Dynamic Visualisation of Software Architectures*, accepted to 2000 IEEE Symposium on Visual Languages, Seattle, Washington, Sept. 14-18 2000, IEEE CS Press.

Grundy, J.C. Storage and retrieval of Software Components using Aspects, In Proceedings of the 2000 Australasian Computer Science Conference, Canberra, Australia, Jan 30-Feb 3 2000, IEEE CS Press, pp 95-103.