Component-based Methods, Architectures and Tools

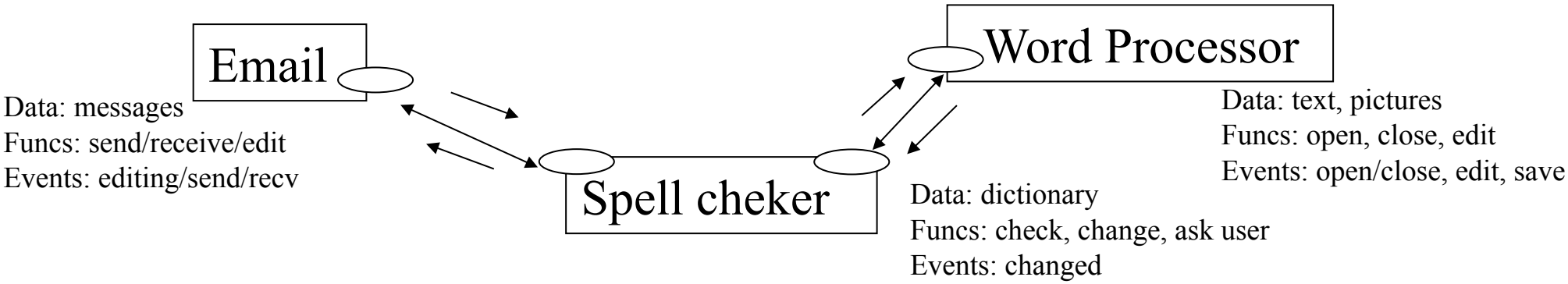
John Grundy
Dept of Computer Science
University of Auckland

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

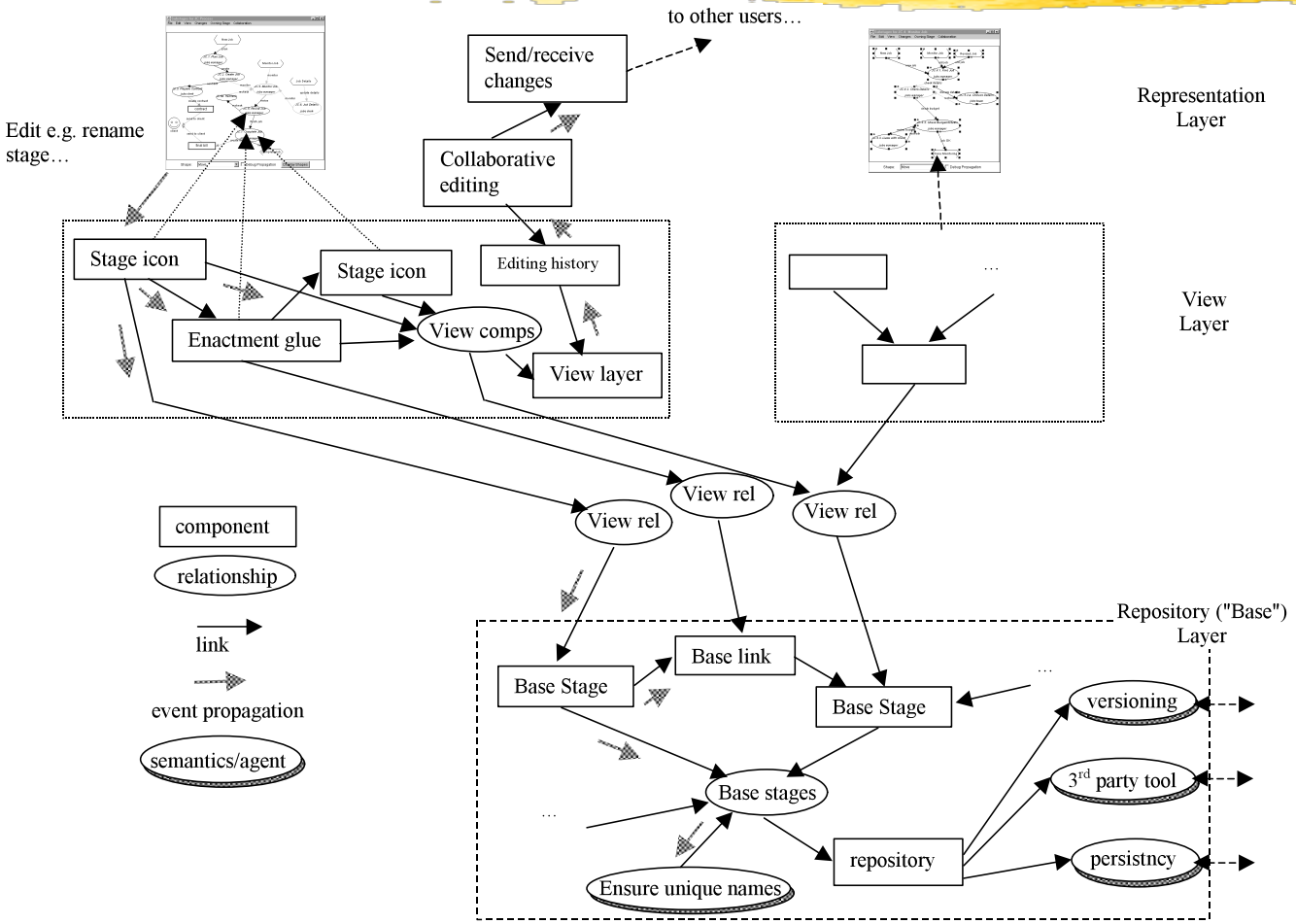
Use components to design & implement…

⇒ **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

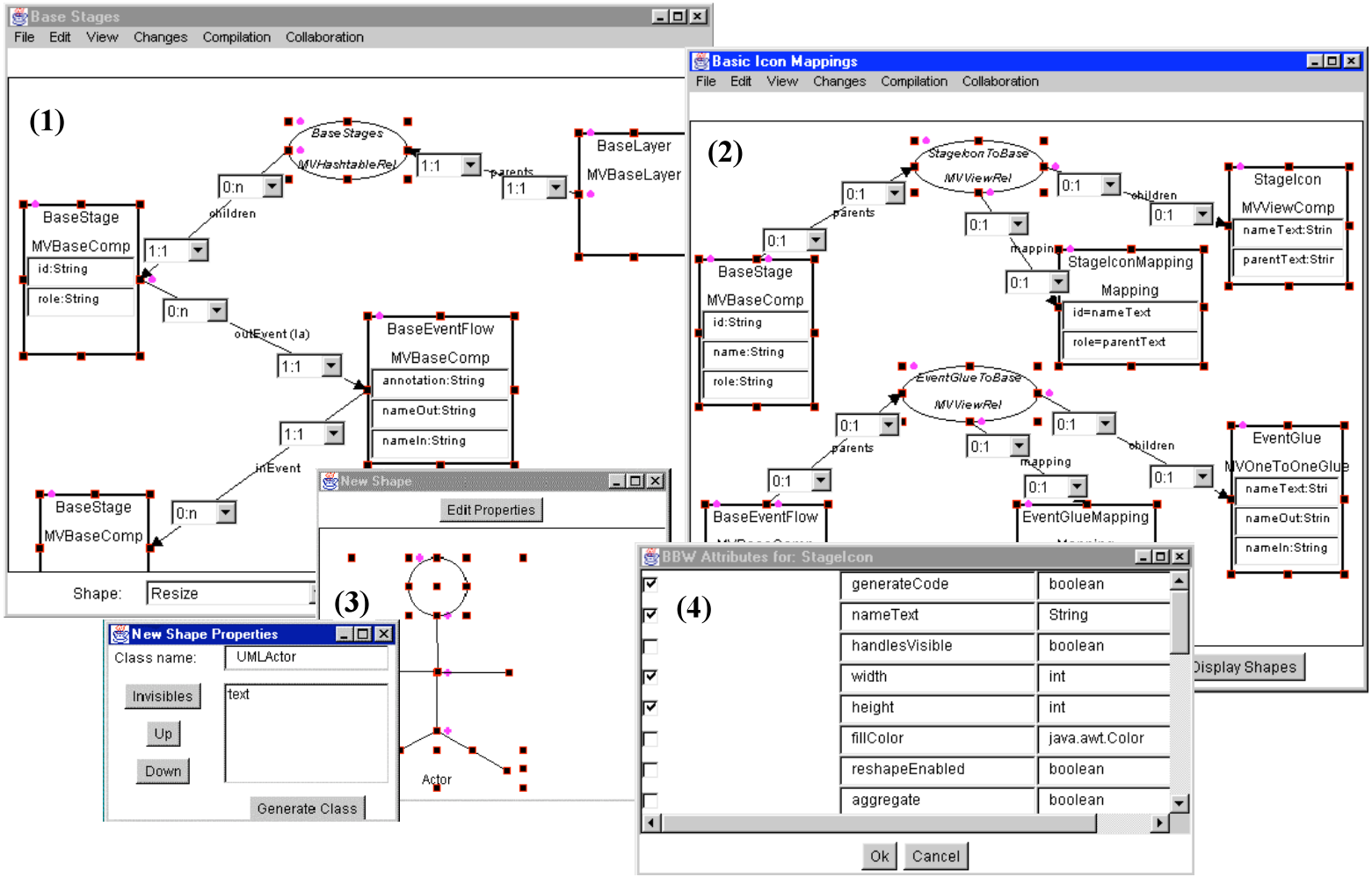
Building with Jviews hard -

need some tools to help...

⇒**Jcomposer & BuildByWire - metaCASE Tools**

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

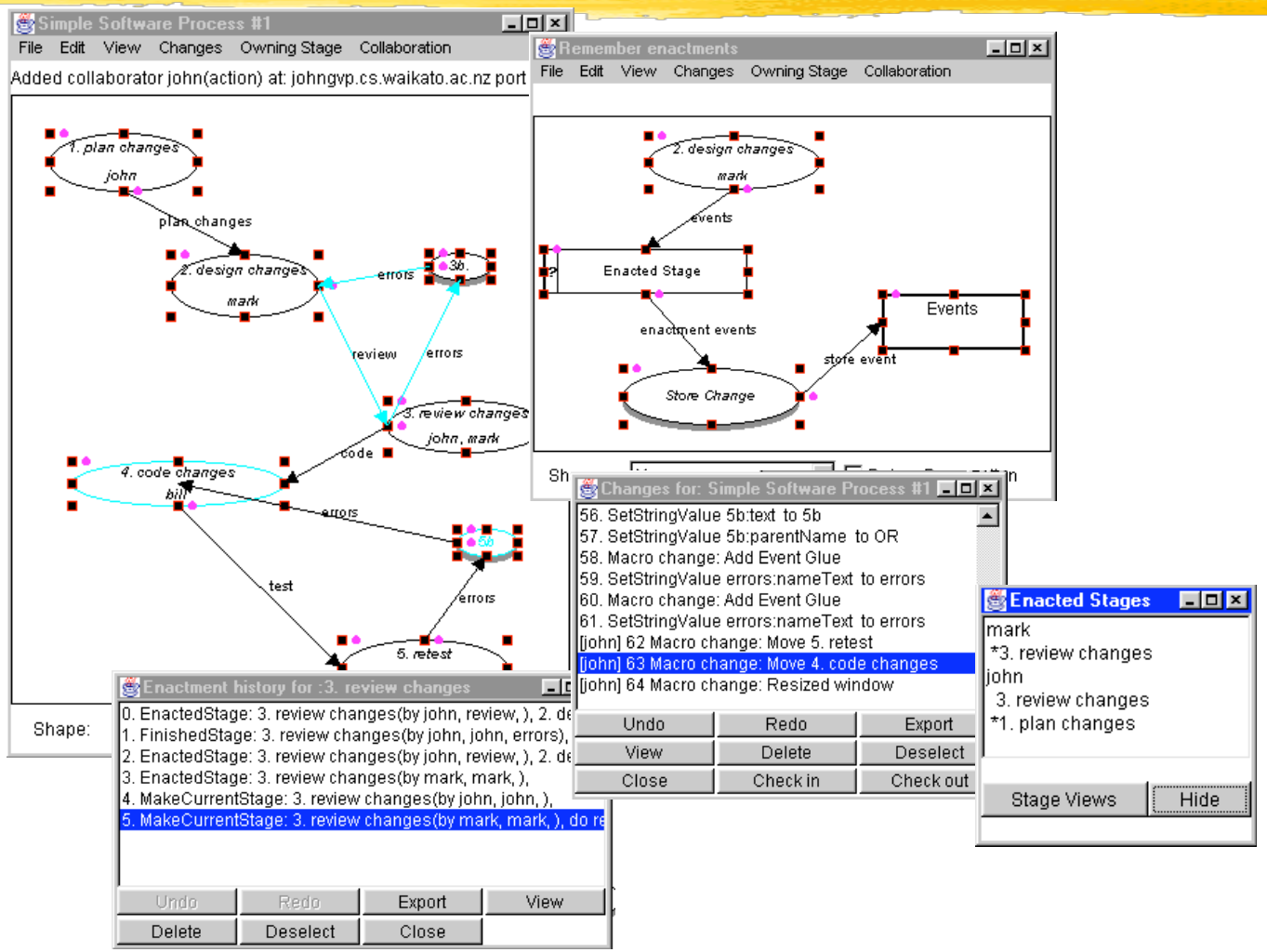
JComposer/BuildByWire



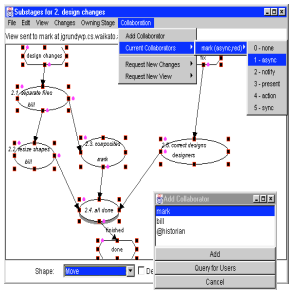
Serendipity-II

- 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

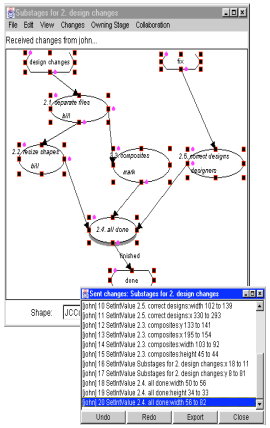
Ser-II: Example of Use



Collaborative Work...

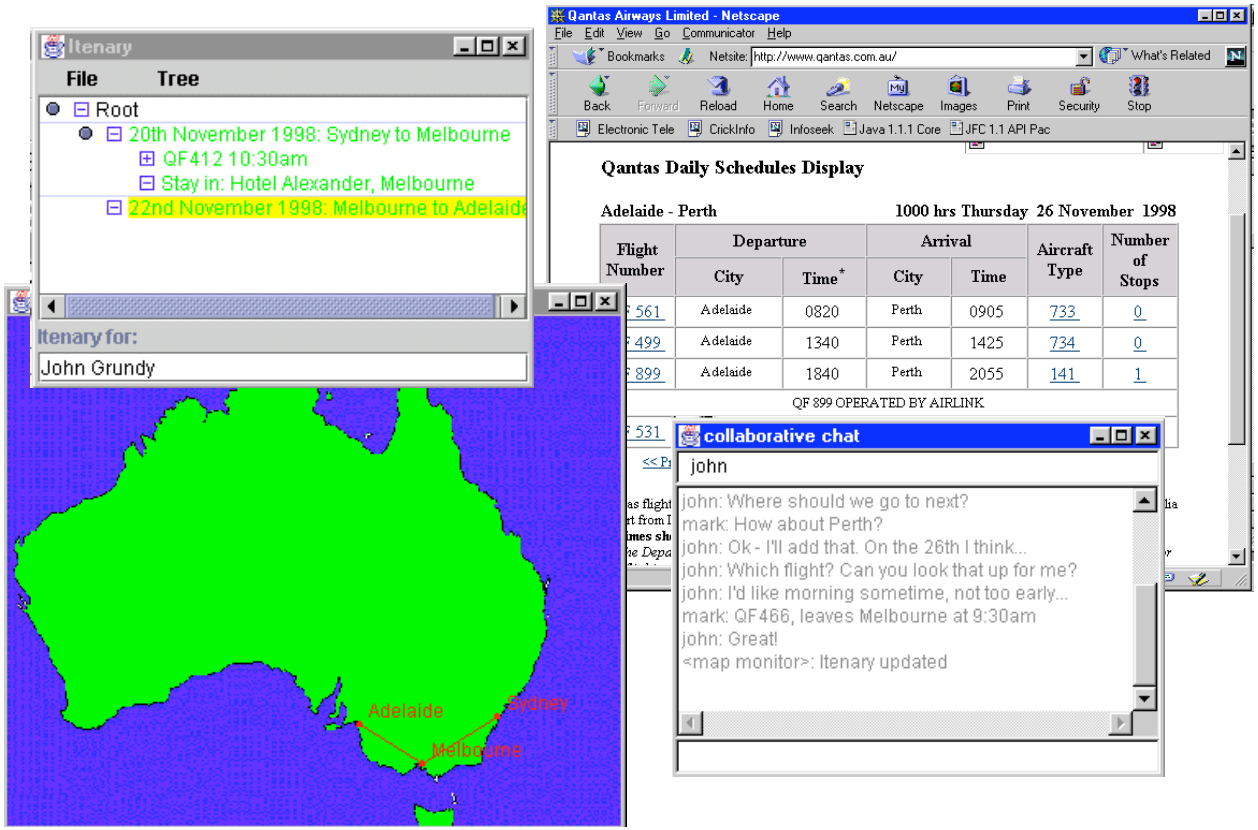


(a) John's view



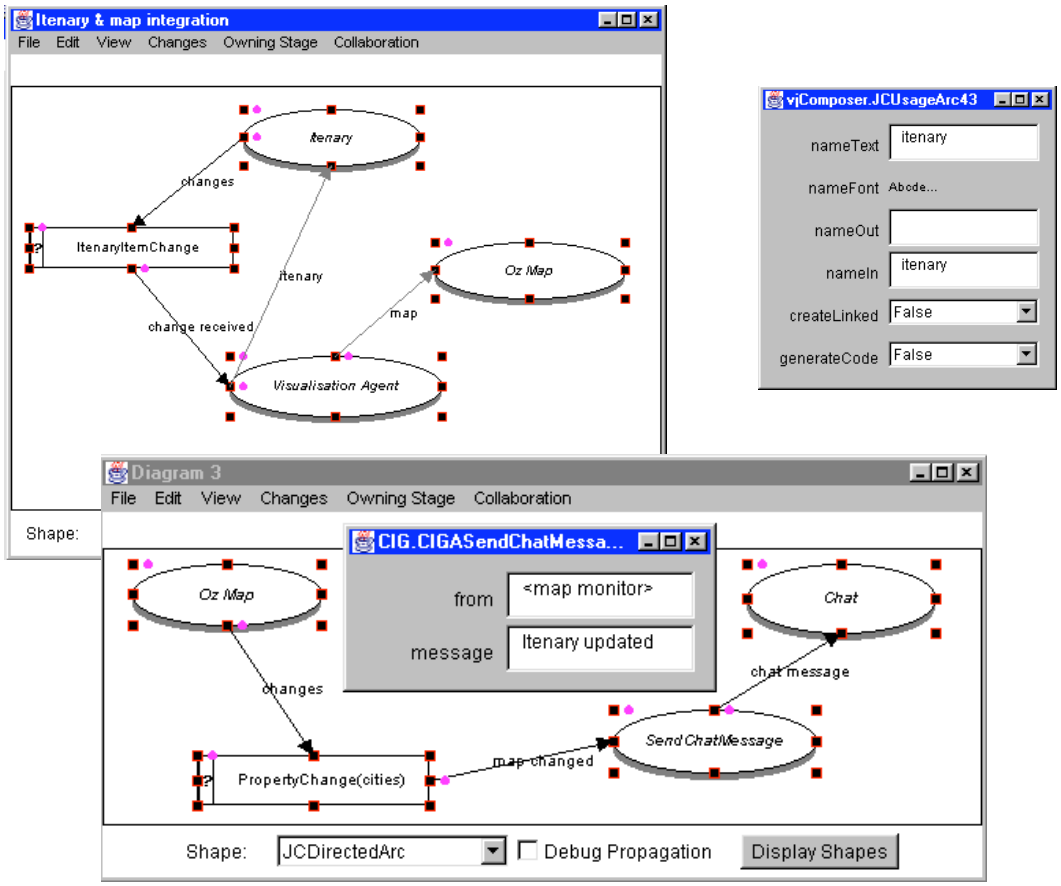
(b) Mark's View

Other applications...



- 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

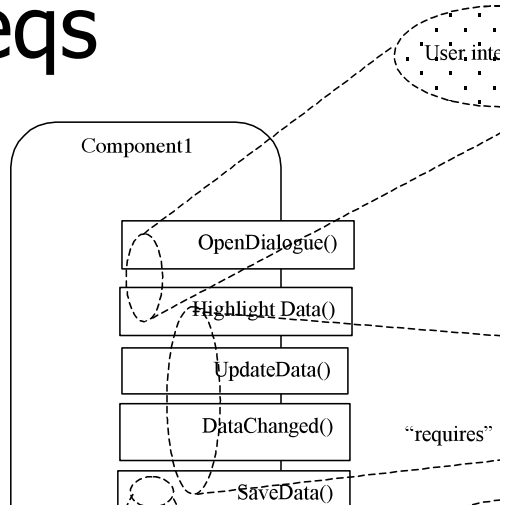
Developing Components Hard:

Requirements→Design → Code → Deployment

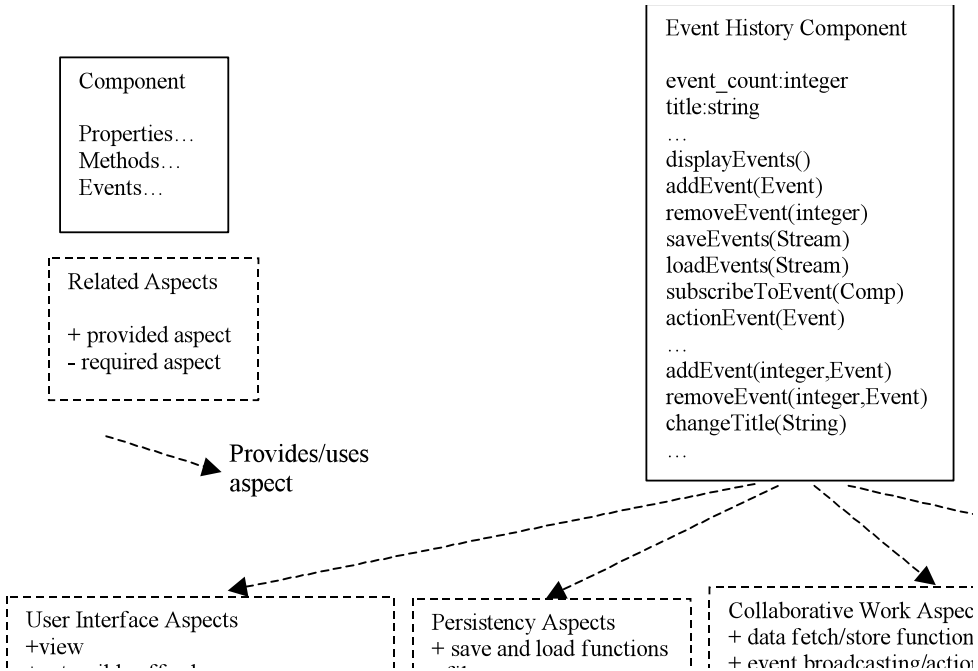
⇒**Aspect-Oriented Component Engineering**

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...



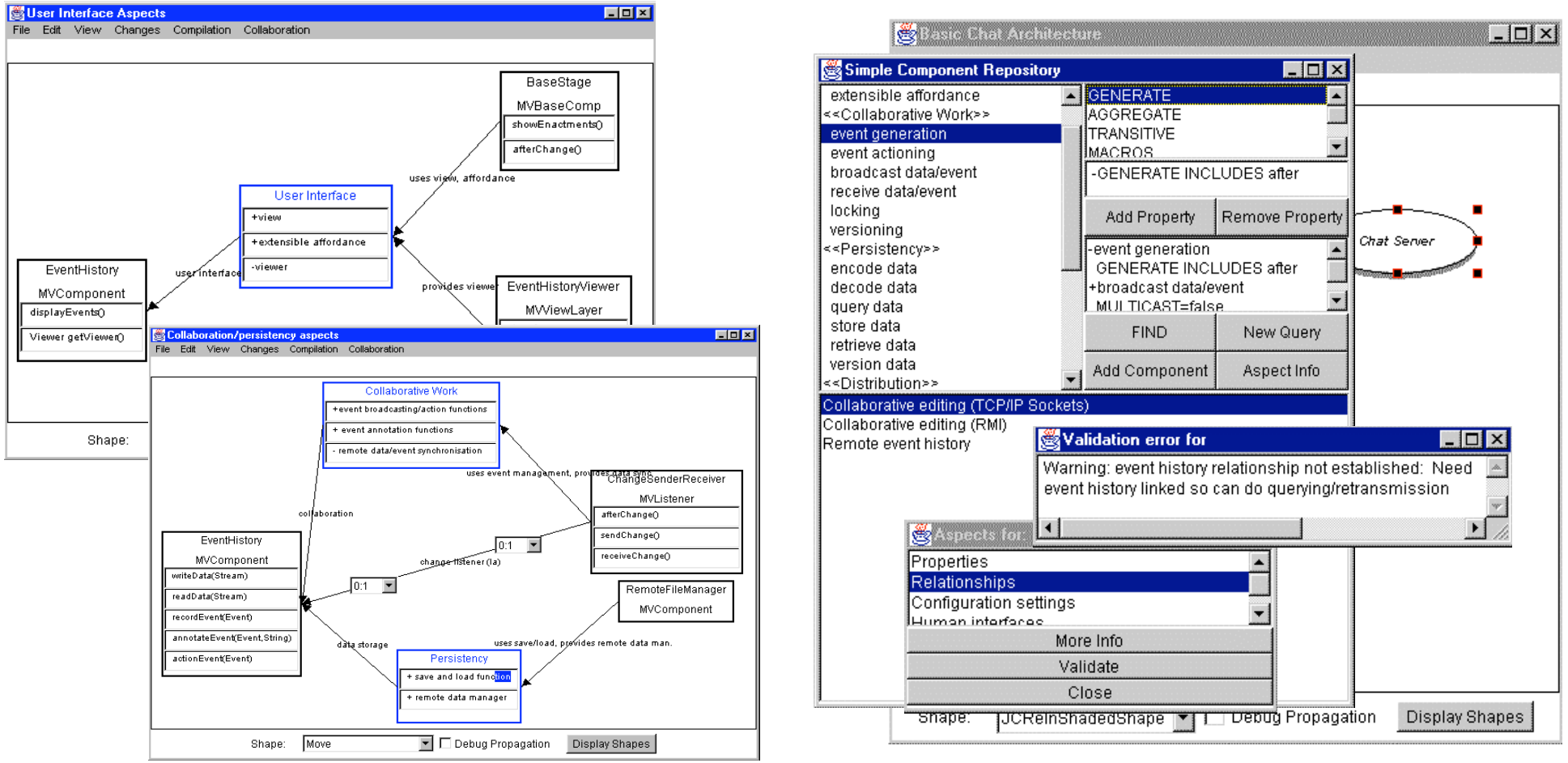
Example: JViews Event History



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



Architectural Support

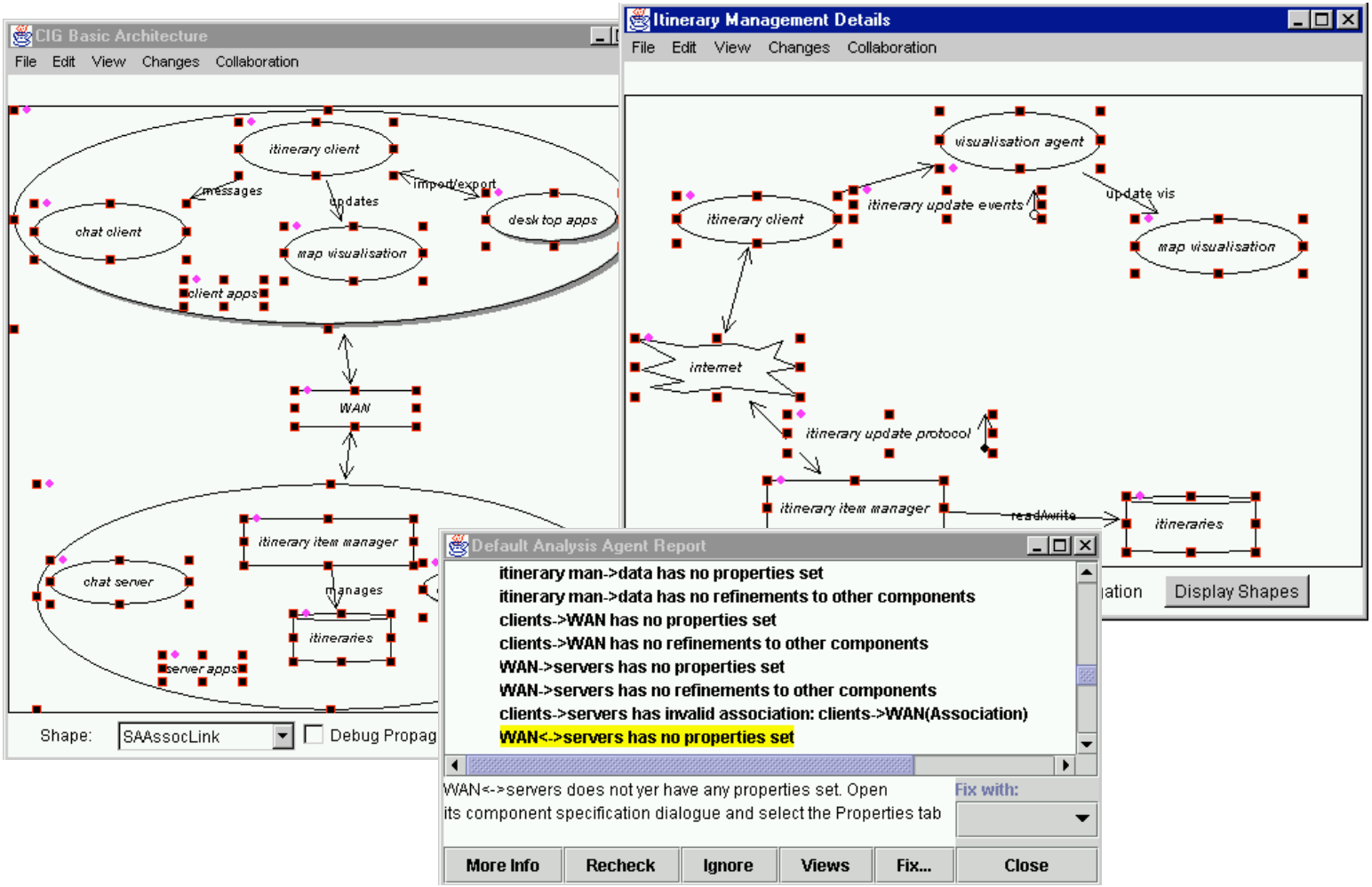
Lacked Good Software Architecture Support

⇒**SoftArch Tool**

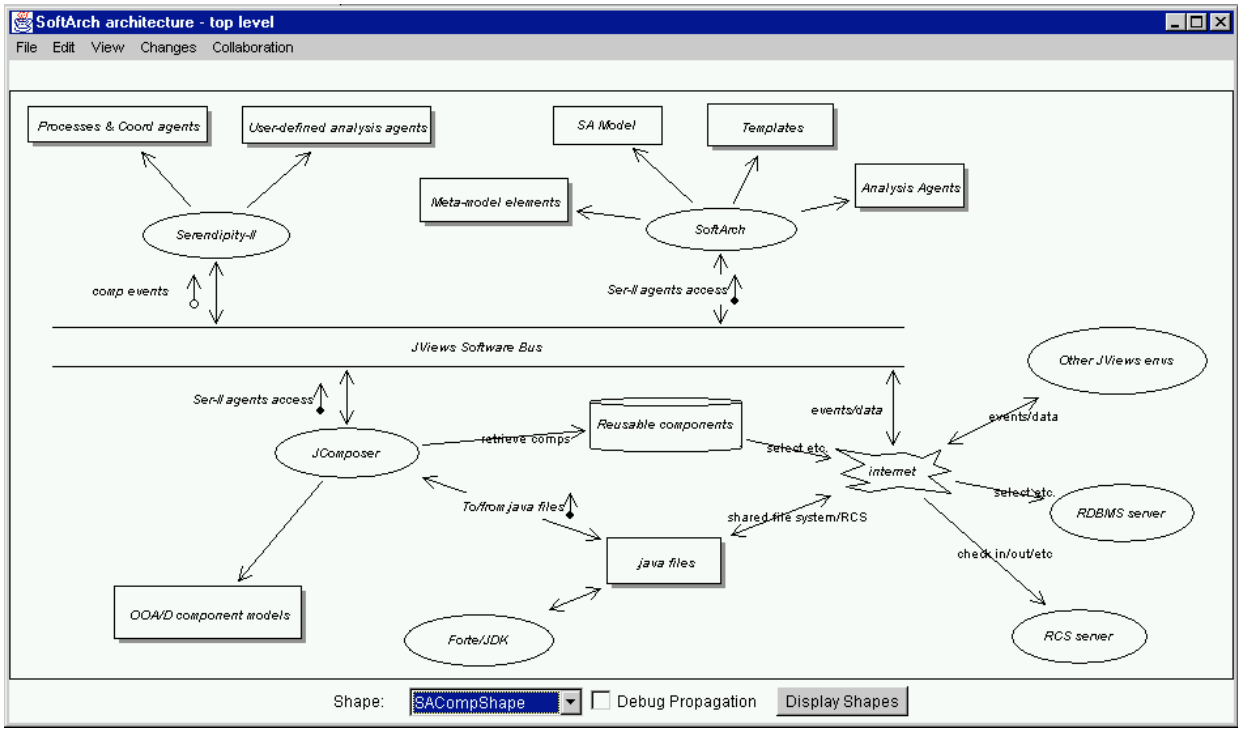
Need to put tools together…

⇒**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.

http://www.cs.auckland.ac.nz/~john-g/publications.html