

## Collaborative Tools

- **Software development is not a solitary activity**
- **Need to collaboratively design and develop software**
- **As team size increases need to formalise collaboration approaches**
- **Implies need for tool support for collaboration**
- **Much of this lecture based on Graham and Grundy's paper: "External Requirements of Groupware Development Tools"**

## Various Approaches

- **Grove, collaborative writing systems – build into tools in ad-hoc manner**
- **Suite toolkit – build into the UI infrastructure used to realise the tool**
- **SPADE-1 – event-based infrastructure, APIs**
- **COAST – collaboration components (static) can reuse when building tools**
- **JViews – Reusable components, some can be dynamically deployed/reconfigured @ run-time**

## Collaboration – Some Key Issues

- What artefacts to support for?
  - Code, diagrams, tests, design docs etc
- What kinds of collaborative editing should we support?
  - Synchronous – same time
  - Asynchronous – different time
- What degree of formality
  - Repository v shared file system, process tool v coffee room discussion
- How communicate about changes making/want to make/have been made?
- How co-ordinate updates across multiple (possibly dispersed) people?
- How do we add support to existing tools?
  - Build into tool from the start
  - Plug-and-play component support?

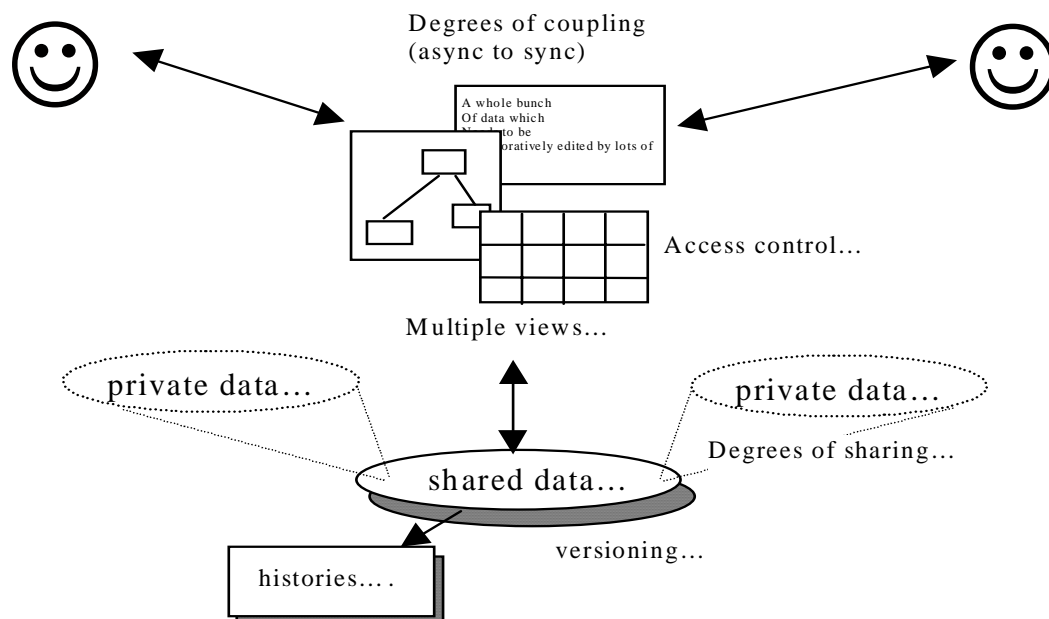
## Basic Support Features

- exchange code, diagrams, documents (asynchronous)
- see others edits (synchronous and asynchronous)
- control edits (floor control, locking)
- see what others doing/done (group awareness)
- manage revisions of code, diagrams, documents (version control)
- co-ordinate collaboration between users (turn-taking, roles, process-based support, issue management)

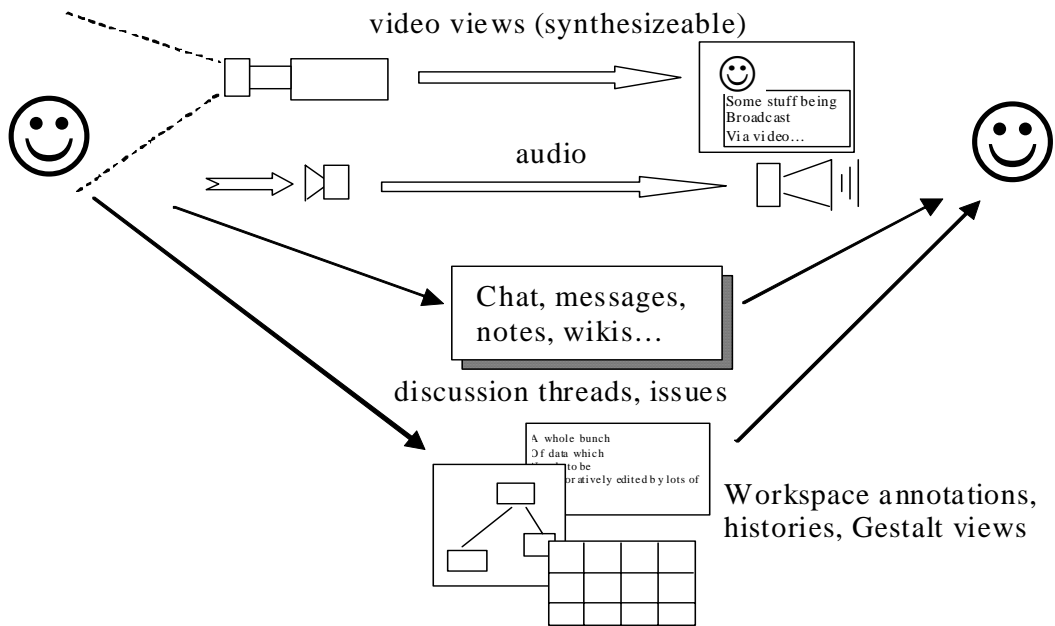
## Aspects of collaboration

- See Graham and Grundy paper (based on Calverly's Clover model)
- Collaboration (Production space)
  - Artefacts shared, viewed, modified
  - Repositories, compare/merge, synchronous edits, wikis
- Communication
  - Discussion of work/tasks
  - Chat, video conf, notes, annotations, immersive reality, wikis
- Co-ordination
  - Co-ordination between software engineers
    - Social rules -> formalised workflow
  - Locking, turn-taking, check in/out, process support, issue management

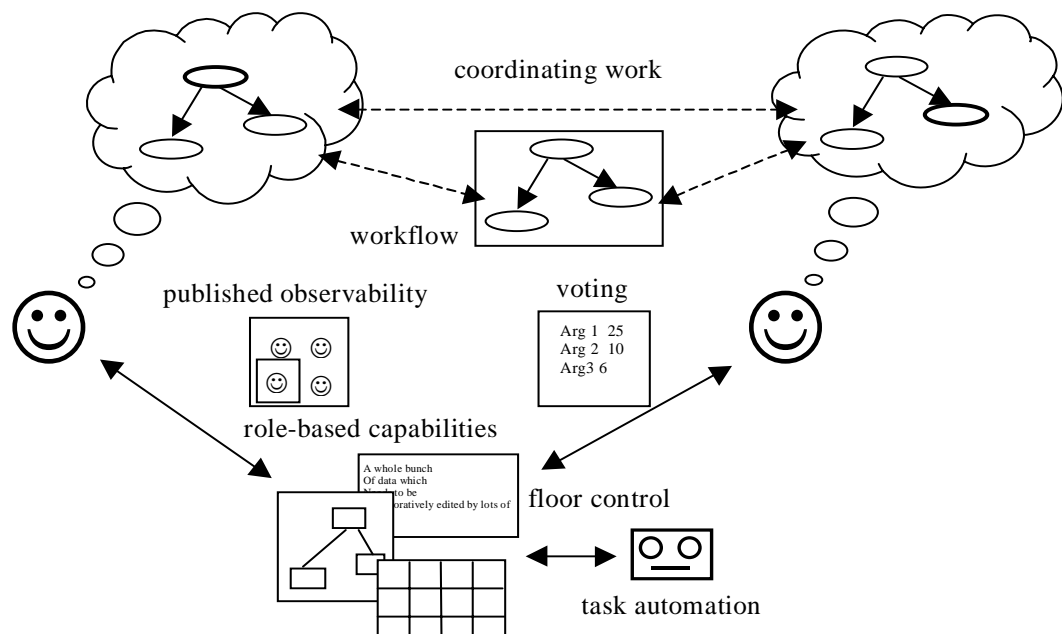
## Collaborative Editing/ "Production Space"



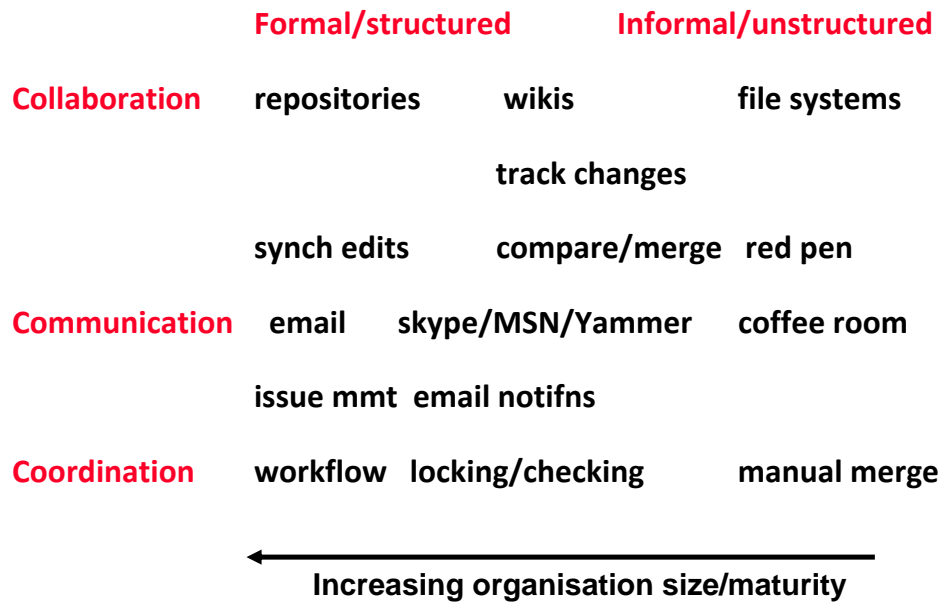
# Communication Support



# Co-ordination of Work



# Formality/informality



## Examples: Collaboration

- Synchronous editors e.g. Ellis et al Grove (1989)

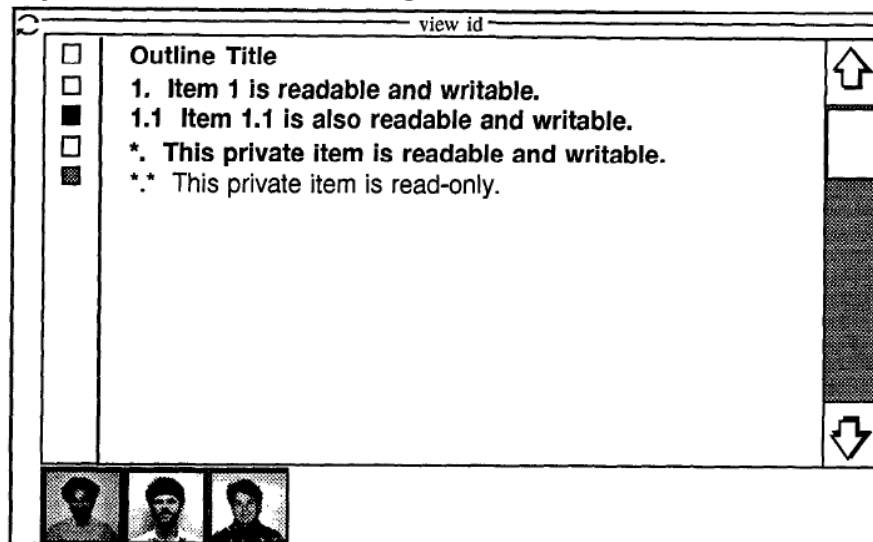
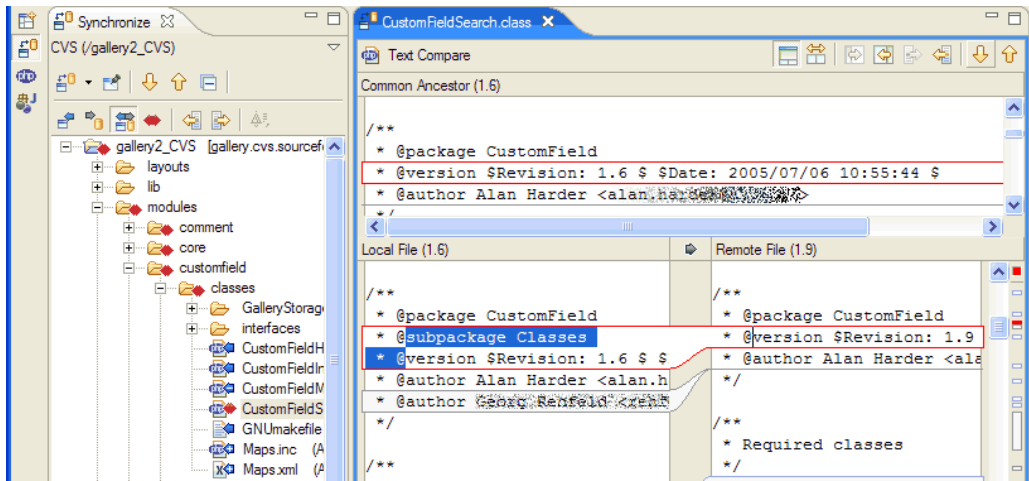


Figure 1 GROVE group window

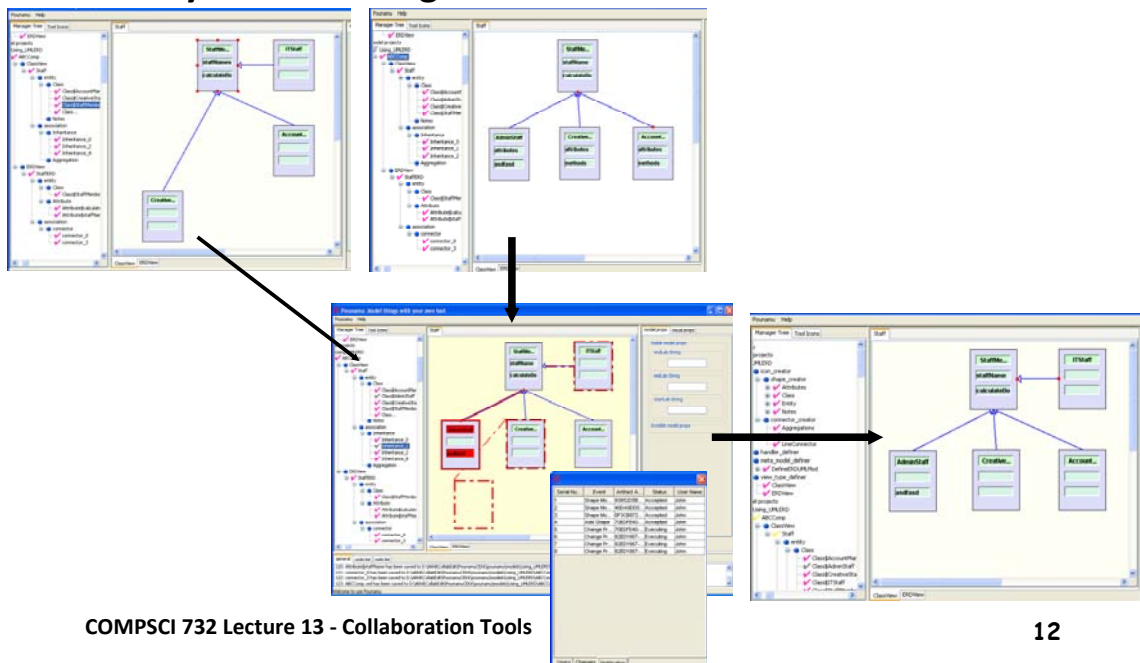
# Examples: Collaboration

- Asynchronous e.g. diffing in Eclipse



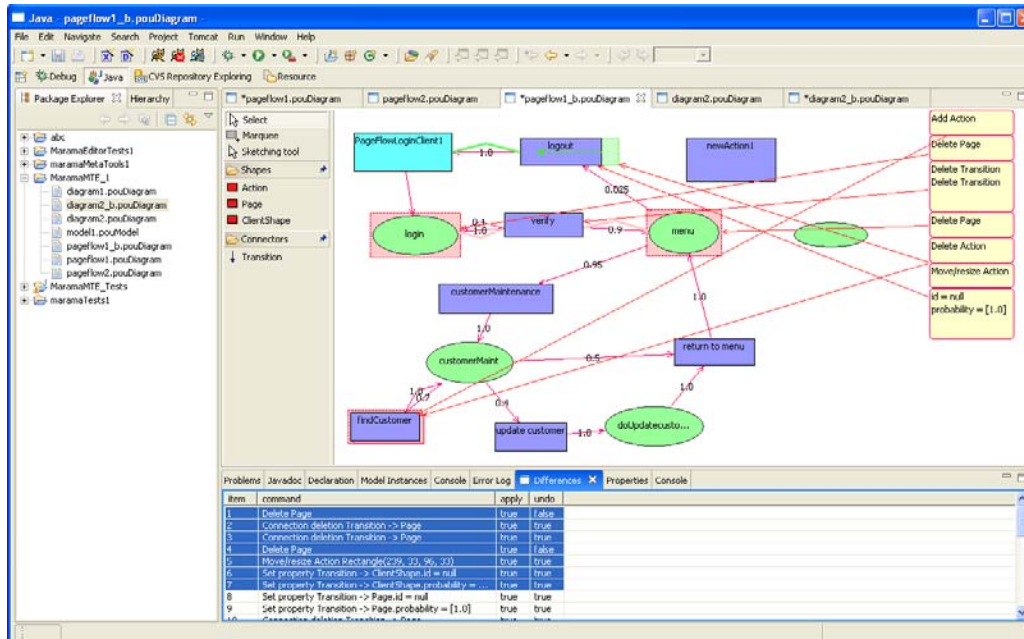
# Examples Collaboration

- Asynchronous e.g. Pounamu visual differ



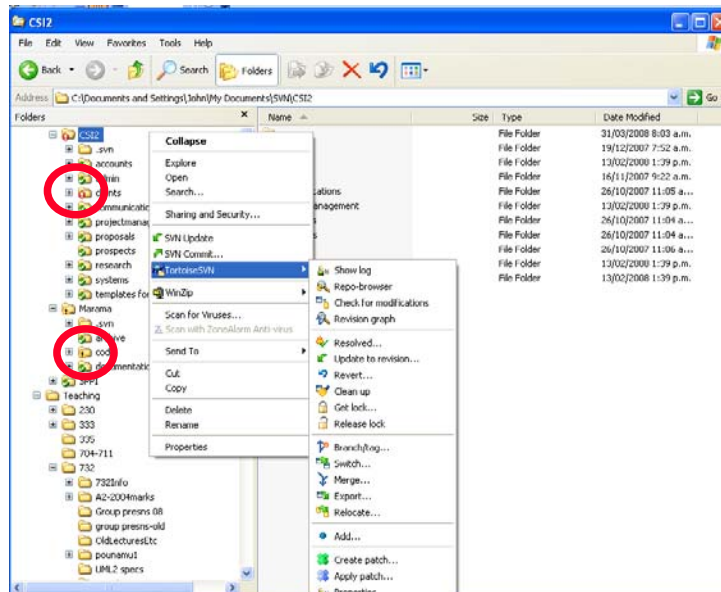
## Examples: Collaboration

- Asynchronous e.g. visual diffing in Marama



## Examples: Collaboration

- Asynchronous: Subversion – TortoiseSVN client



## Examples: Communication

- Synchronic: EVO video chat (skype on steroids) Timezone etc support



IM functions/  
presence

Encryption  
etc

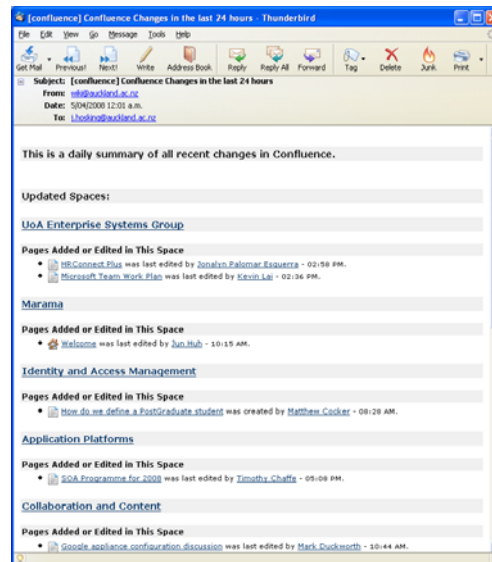
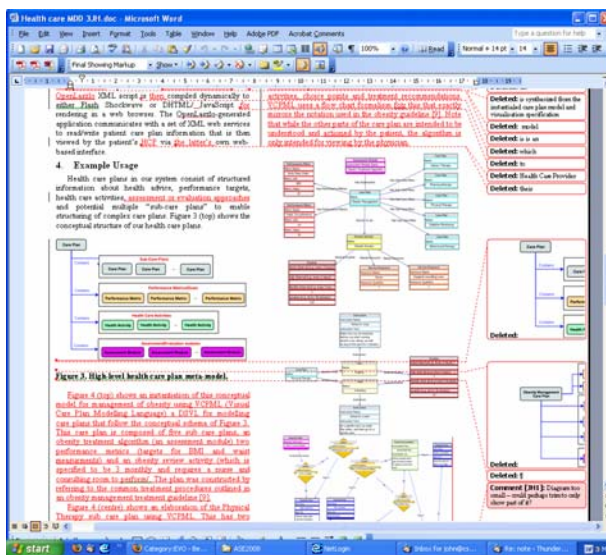
Video/audio chat

Record/  
playback

File &  
screen  
share +  
shared  
wboard

## Examples: Communication

- Asynchronous: Annotations e.g. in Word; check-in comments; Wiki comments/edits; email notifications

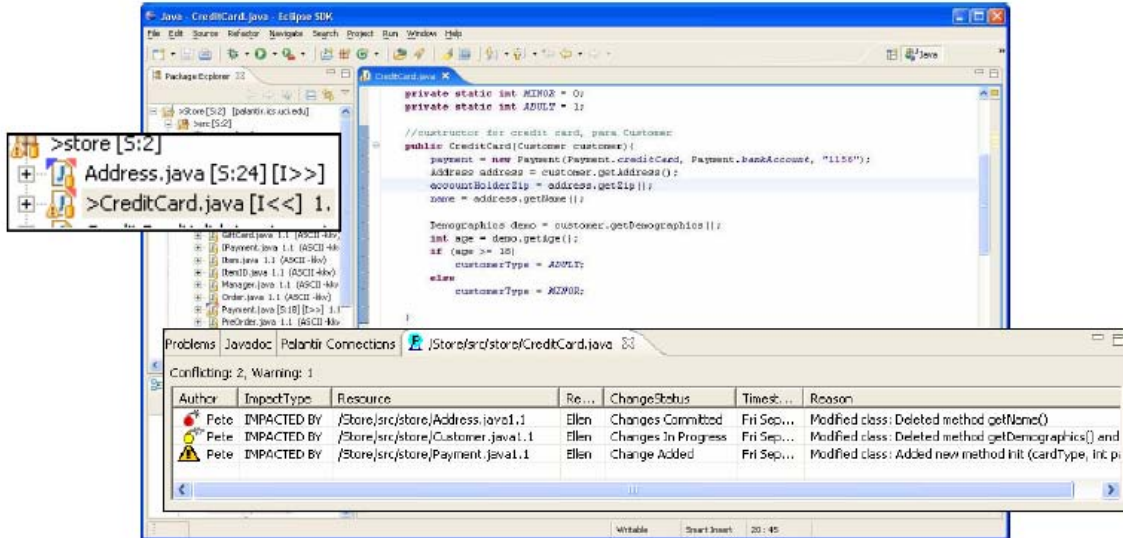






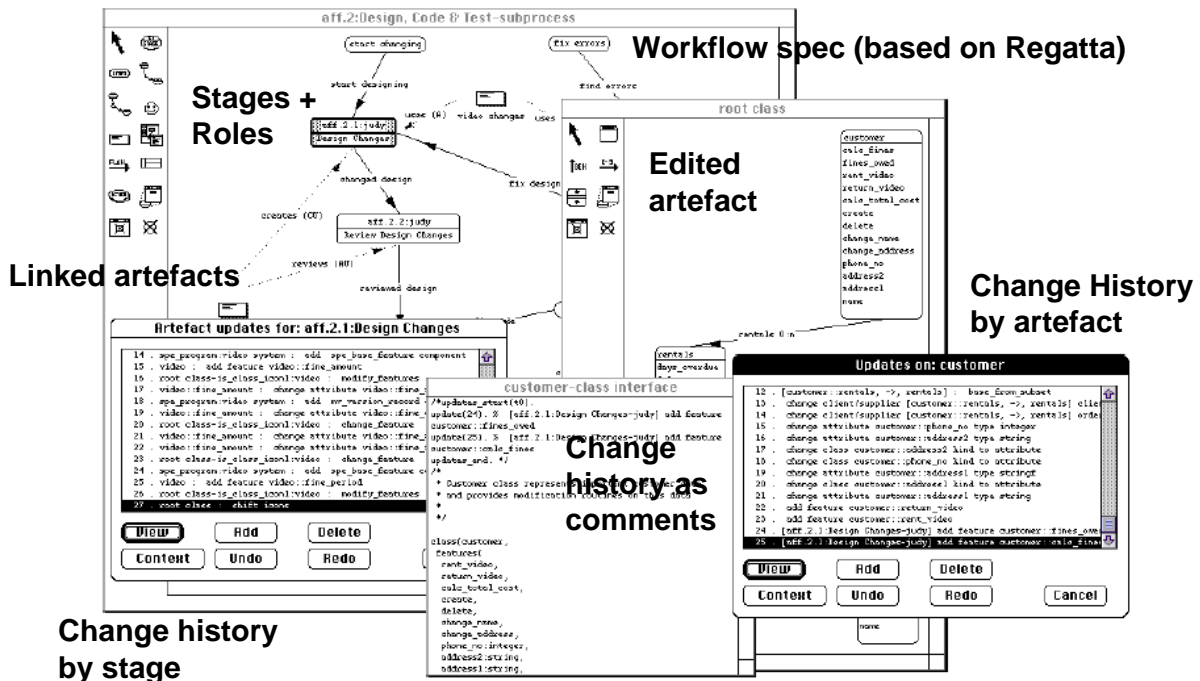
## Examples: Co-ordination

- Group awareness eg Palantir (Sarma et al) – awareness of direct and indirect conflicts when editing artefacts



## Examples: Co-ordination

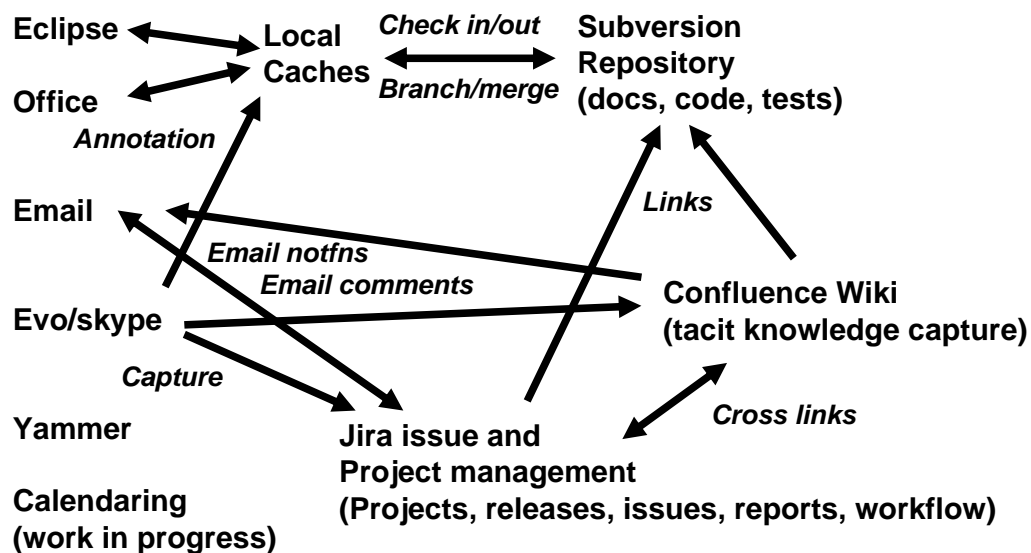
- Workflow support – Serendipity (Grundy and Hosking)

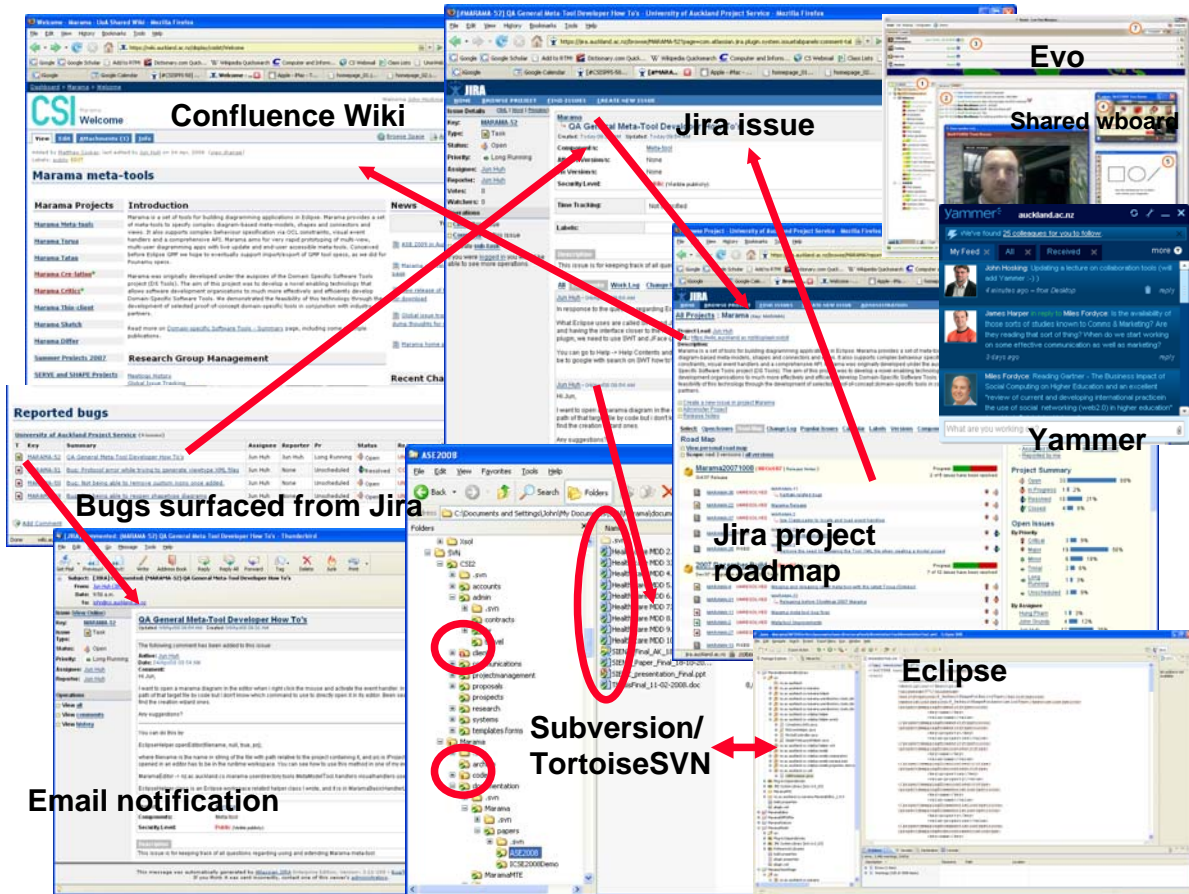


## Case study – Centre for Software Innovation

- Subversion repository for code and document sharing
  - versioned
  - code branched – Eclipse tools plus manual merge
  - MSWord track changes annotations for doc collabn
- Confluence wiki for knowledge management
  - “wiki-culture”
  - Changes auto-tracked
- Jira for issue and release management
  - Tracks projects, milestones, releases, issues
  - Tracks time, resource use, provides high level reporting
- Skype, EVO, e-mail + email notifications (Jira Confluence), Yammer, f2f, etc for communication
  - Often record skype conversations -> wiki, email -> Jira
- Jira, Confluence, SVN linked
- Extensive use of security groups, encryption etc for all of these complicates management but ensures confidentiality of client info

## CSI case study





## Supporting collaboration

- Technology needs (from Graham and Grundy)
  - Resource adaptability
    - Multiple platforms
    - Mobility
  - User preferences
  - Resuability of existing applications
    - Integration needs
    - Retrofitting legacy applications an issue
    - Modern event based/plugin based architectures help
  - Network status reporting
    - Am I being seen/listened too? Can I hear/see?
  - Fault tolerance
    - Ability to switch seamlessly between
      - Synch <-> Asynch
      - Online <-> Offline

# Example architectures

- Grundy and Hosking SP&E 2002

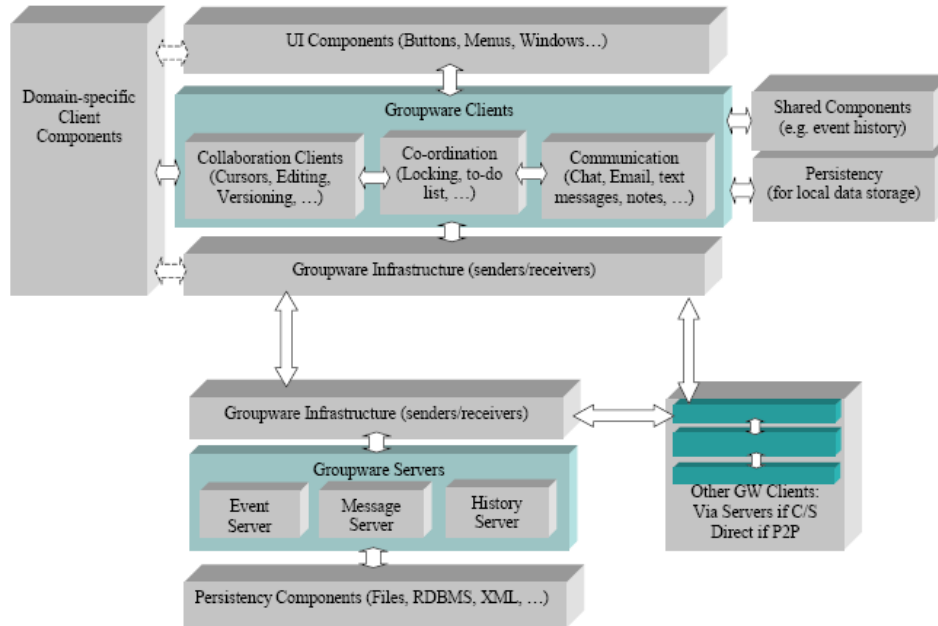
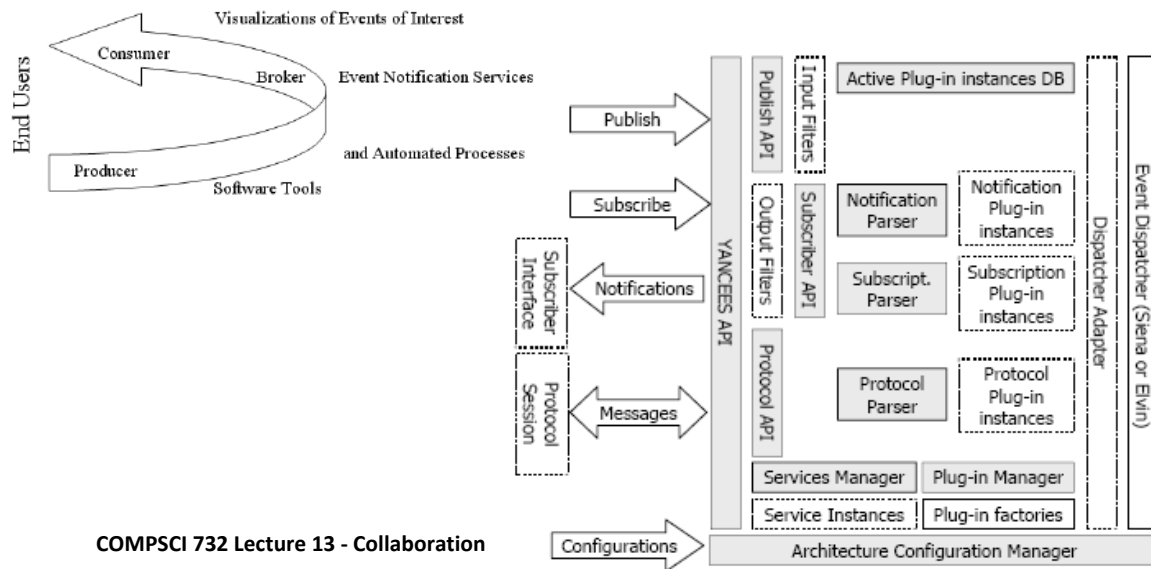


Figure 7. Basic groupware component infrastructure.

# Example Architectures

- Van der Hoek et al – Continuous coordination, YANCEES framework for event subscription/notification services



## Tutorial Exercise

**Thursday class exercise this week: 3 exercises on collaboration mechanisms – in 303.279**

- (1) Collaborative drawing**
- (2) Collaborative text editing**
- (3) Collaboration infrastructure**