Some Recent Directions in Automated Software Engineering Research

Professor John Grundy

Dept Electrical and Computer Engineering & Dept Computer Science

Outline

- What is "Automated" Software Engineering?
- Some recent ASE research @ UoA:
  - Meta-tools
  - Performance engineering via test-bed generation
  - Component discovery/integration/validation
  - Collaborative work
  - Adaptive user interfaces
  - Software architectures to support this stuff
- Conclusions

Automated Software Engineering

- Generative – try and generate code from high-level, abstract descriptions (models)
- Component-based – "build applications from bits" approach; ultimately "autonomous agents" composition
- Adaptive – components/agents discover environment and adapt to the circumstances they find themselves in
- Dynamic – ideally can do the above at run-time while the software is in use
- Formalisms necessary – specifications we can reason with; generate code from; verify vs validate models/code

Why?

- Code is too low-level for tasks we want it for – who wants to write code anyway?
- Engineering is about building models of problems/products – can we have models higher level than code? If so, can do much more with them than with program code...
- Can generate huge code base from small abstract models (if all goes to plan...)
- Some successes – domain-specific languages; IDEs; 4GLs; rapid prototyping tools; CASE/CAD tools; hardware synthesis
- Still lacking sufficient formal models for practical use
- Validation/verification become crucial issues
- Can get emergent behaviours esp. with agent-based systems

Examples from our work

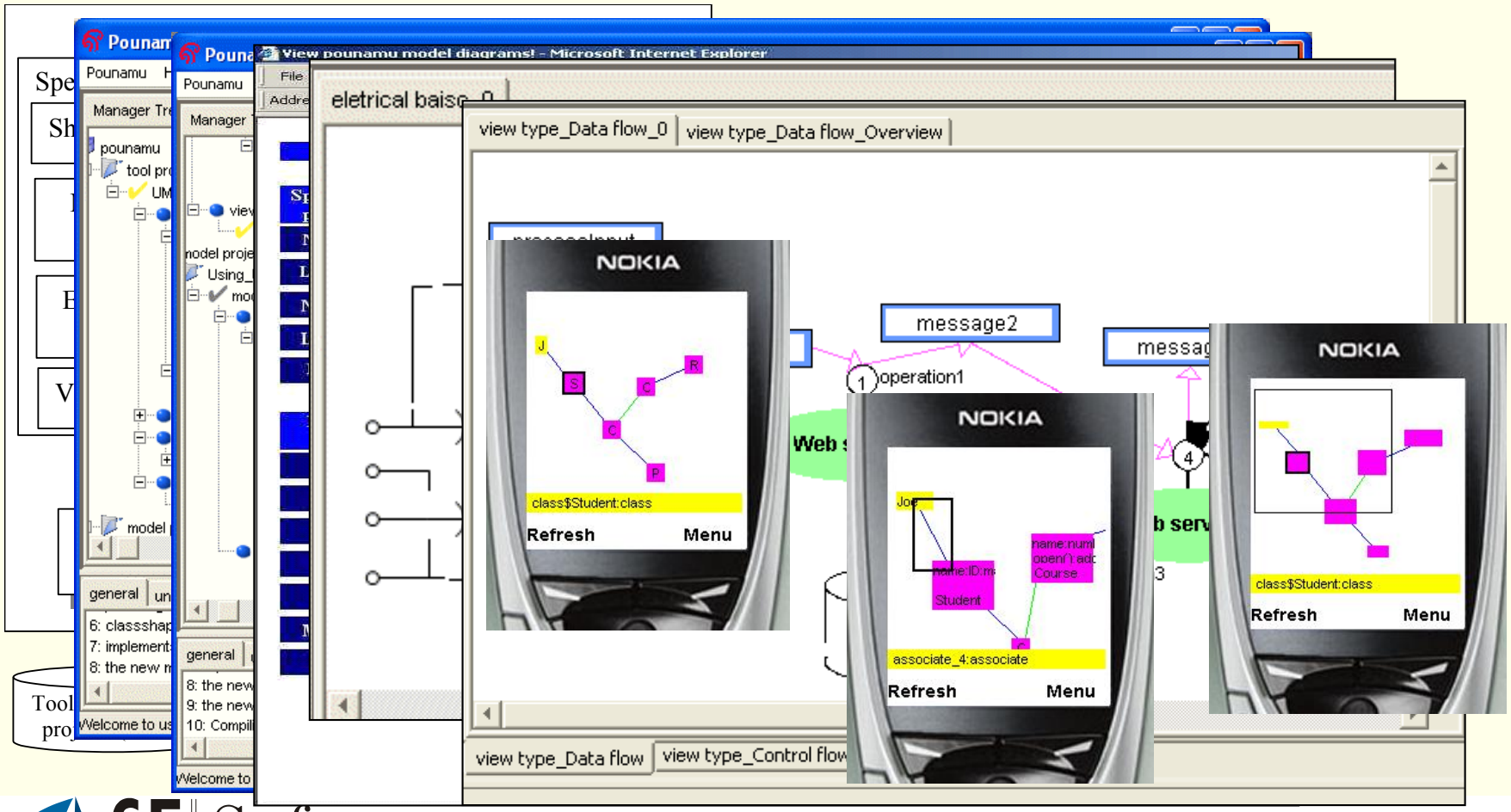
- Pounamu (a meta-CASE tool):
  - Specifying & evolving software tools
- Argo/MTE (a test-bed generator):
  - Performance engineering via test bed code generation
- Aspect-oriented component engineering:
  - Adaptive component-based systems
  - Deployed component validation
- Adaptive user interfaces:
  - E-whiteboard, PDAs, mobile phones etc.
- Some of our current/future directions...

Meta-tools: Pounamu

- A meta-tool (tool for building tools…)
- Specify visual, multiple view "tools"/applications
- Framework provides for dynamic, run-time tool modifications
- Various plug-in extensions: web-based & mobile PDA/phone-based diagramming; collaborative work support; web services APIs; dynamic tool integration

Pounamu example

HCC 2003
HCC 2004
AUIC 2005
ASWEC 2005

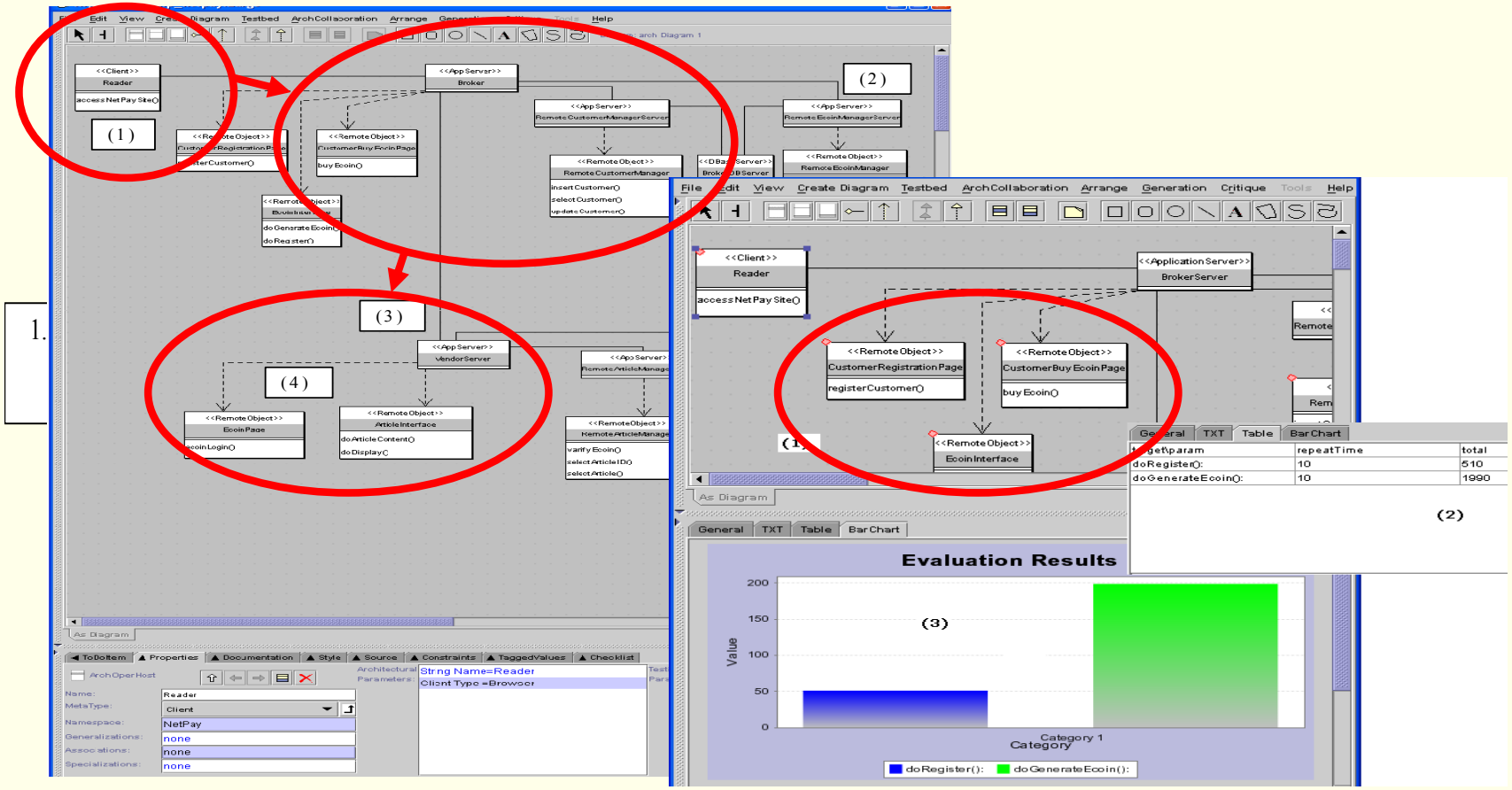


Code generation: Argo/MTE

- For performance engineering – very difficult to estimate likely system performance during design
- Our approach: generate real performance test-bed (code etc) from software architecture model
- Run performance tests and visualise results
- Extension to ArgoUML open-source CASE tool

Argo/MTE example

ASE 2001
SEKE 2004
ASE 2004
ASE J 2005

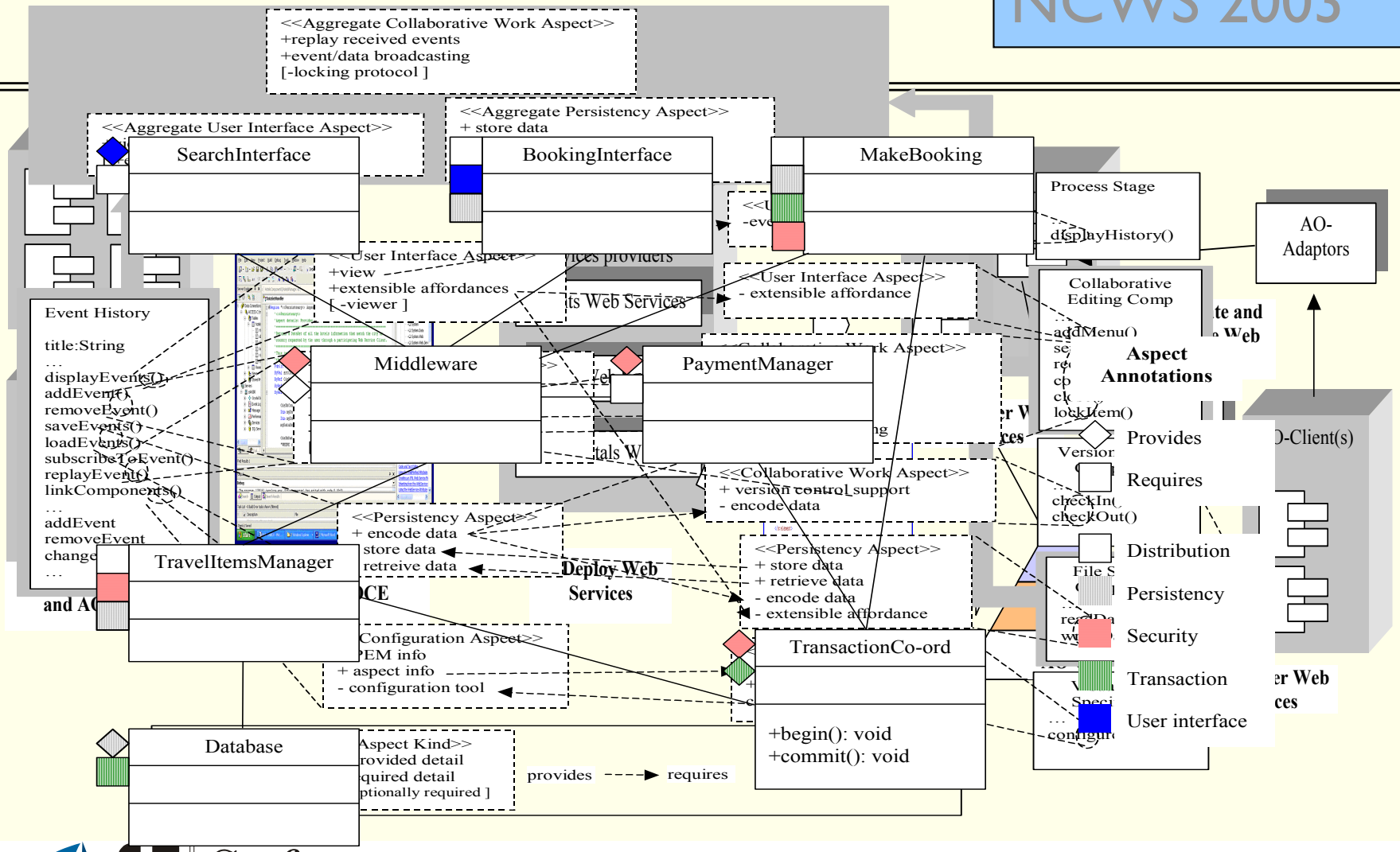


Components: AOCE

- Software components = "build from bits" model
- Problem – understanding/characterising the components to use
- Our solution – apply aspect-oriented techniques to identify cross-cutting concerns to characterise
- Developed method, basic tool support
- Can apply at run-time for dynamic adaptation too

AOCE Examples

IJSEKE 2000
S-P&E 2002
NCWS 2003

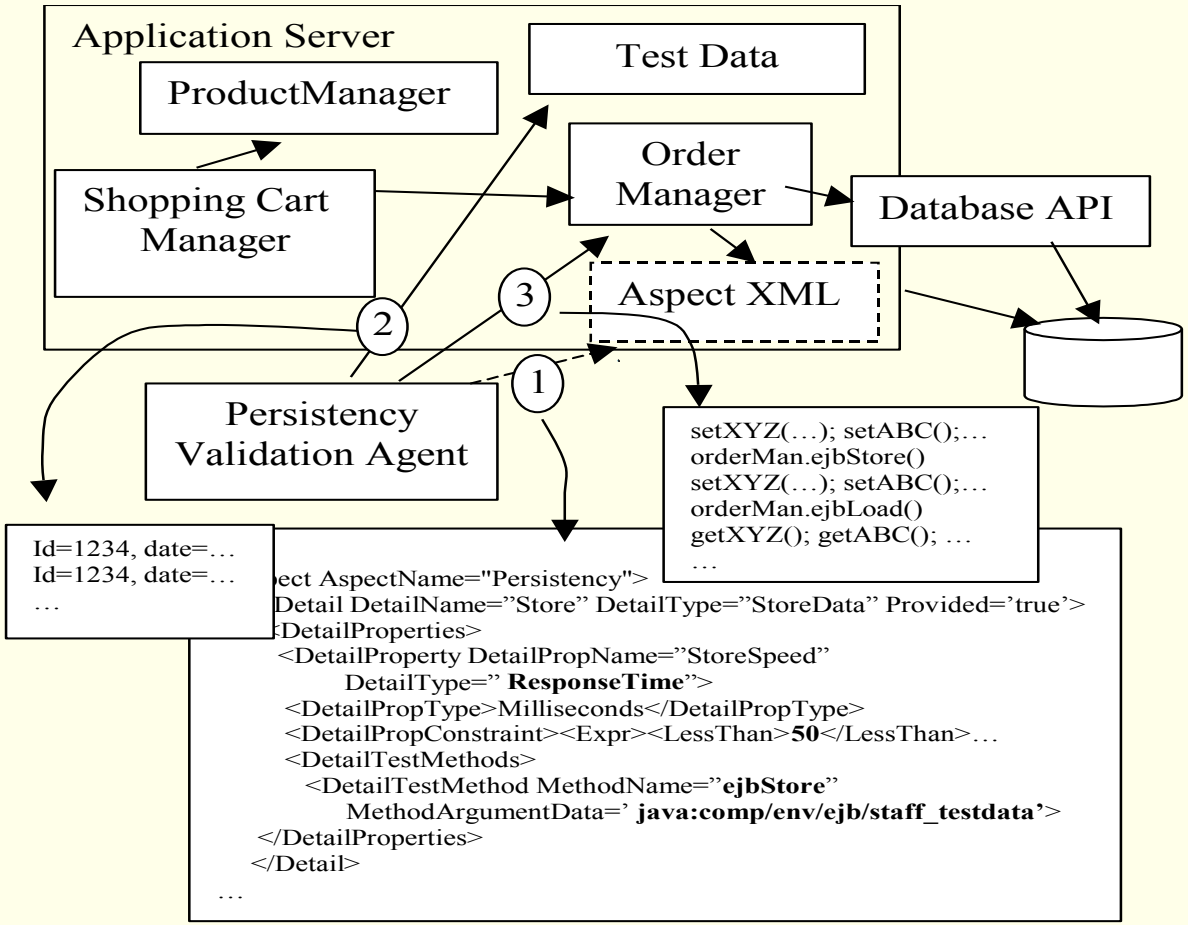


Component Validation

- How do we check deployed components meet their requirements?
- Our approach:
  - Characterise component behavioural/non-functional requirements
  - When deployed, inspect these characteristics
  - Synthesise tests to check these constraints have been met
- Requires more detailed information about components at design/run-time

Component Validation Example

ASE 2002
NCWS 2003
JSS 2004



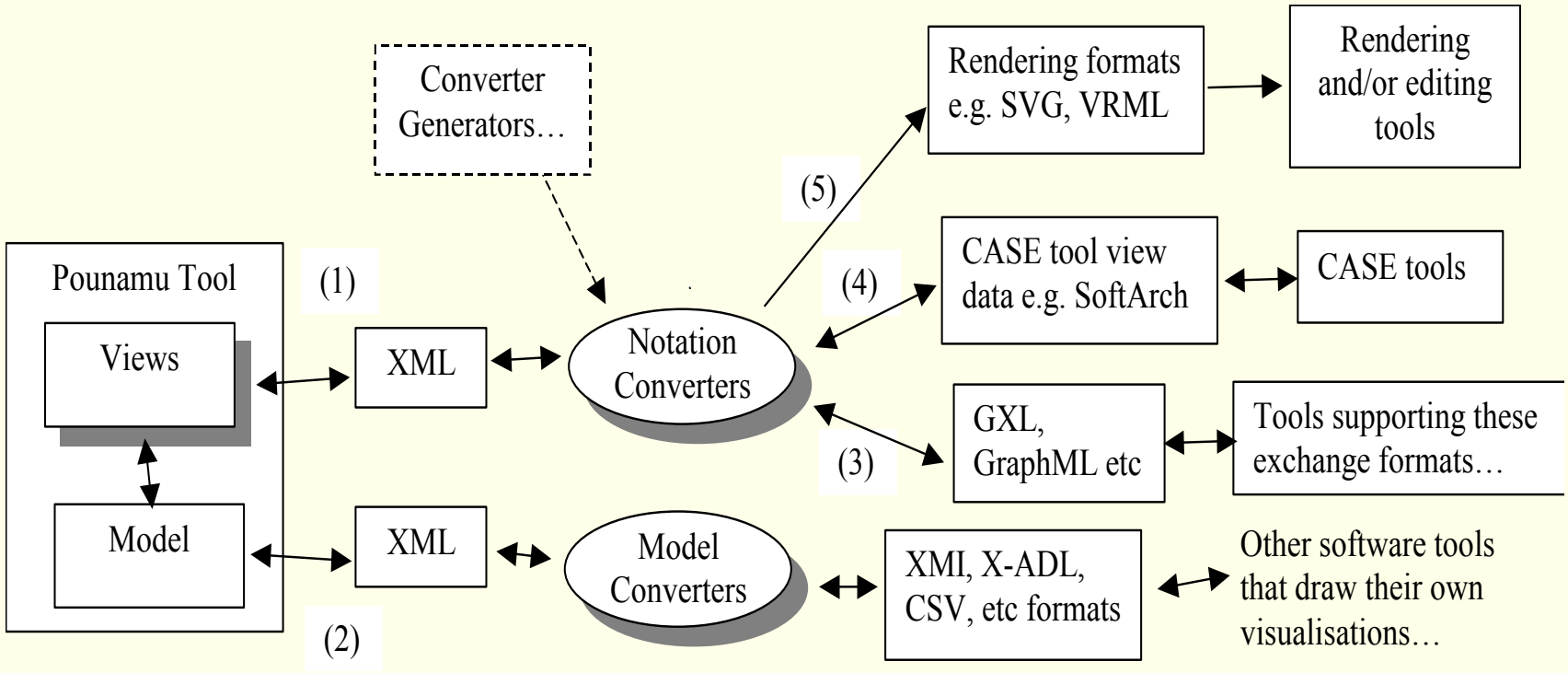
Some of our current work

- Adaptive user interfaces for tools:
  - Web browsers
  - Mobile devices
  - Sketching-based interfaces
- AOCE for web services
- Collaboration "agents"/components
- Visual languages/tools for integration
- DSLs for Data and notation translations

Adaptive UIs

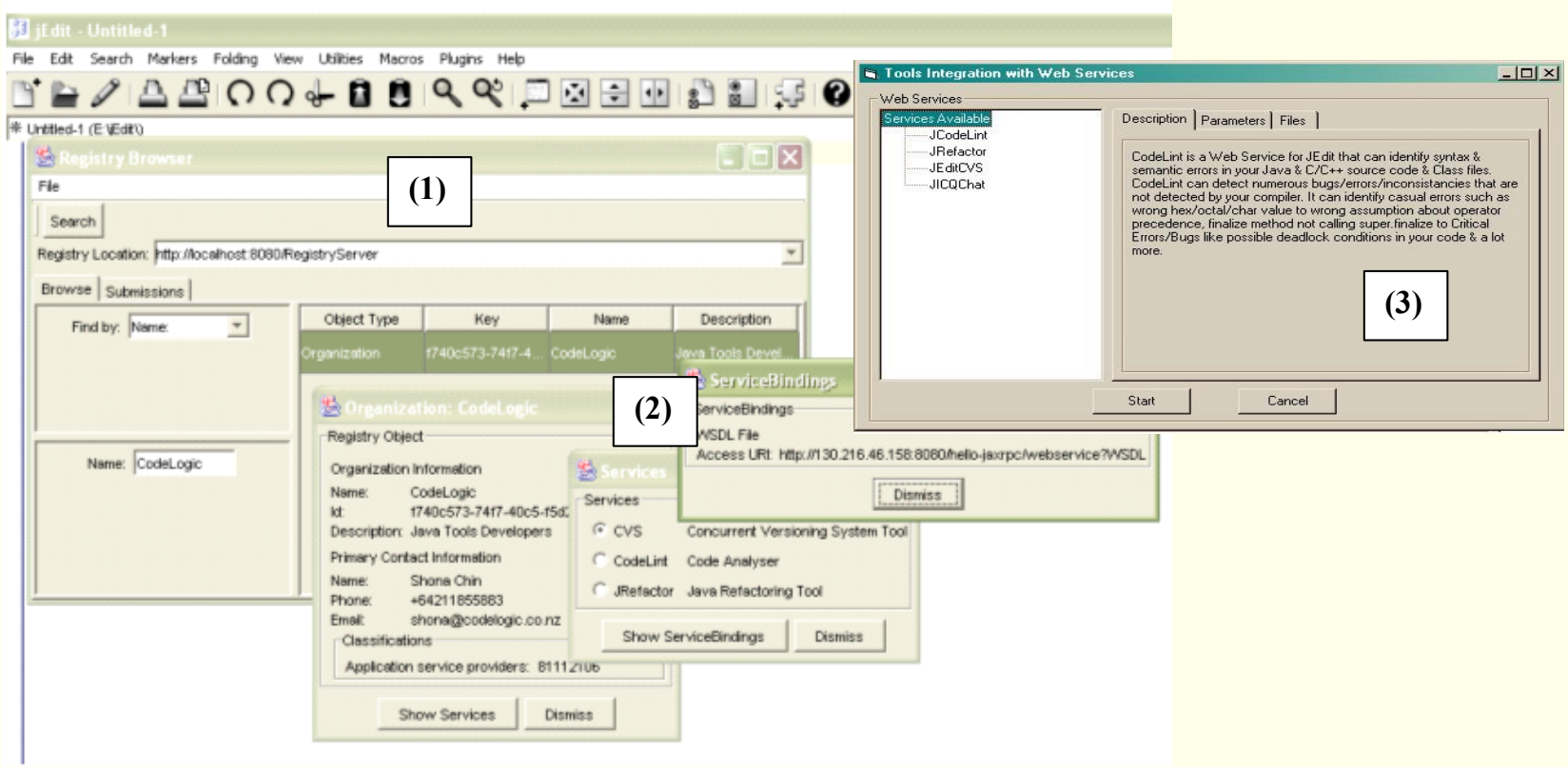
IwC 2002
HCC 2003
MUI 2003
IV 2003

How achieved



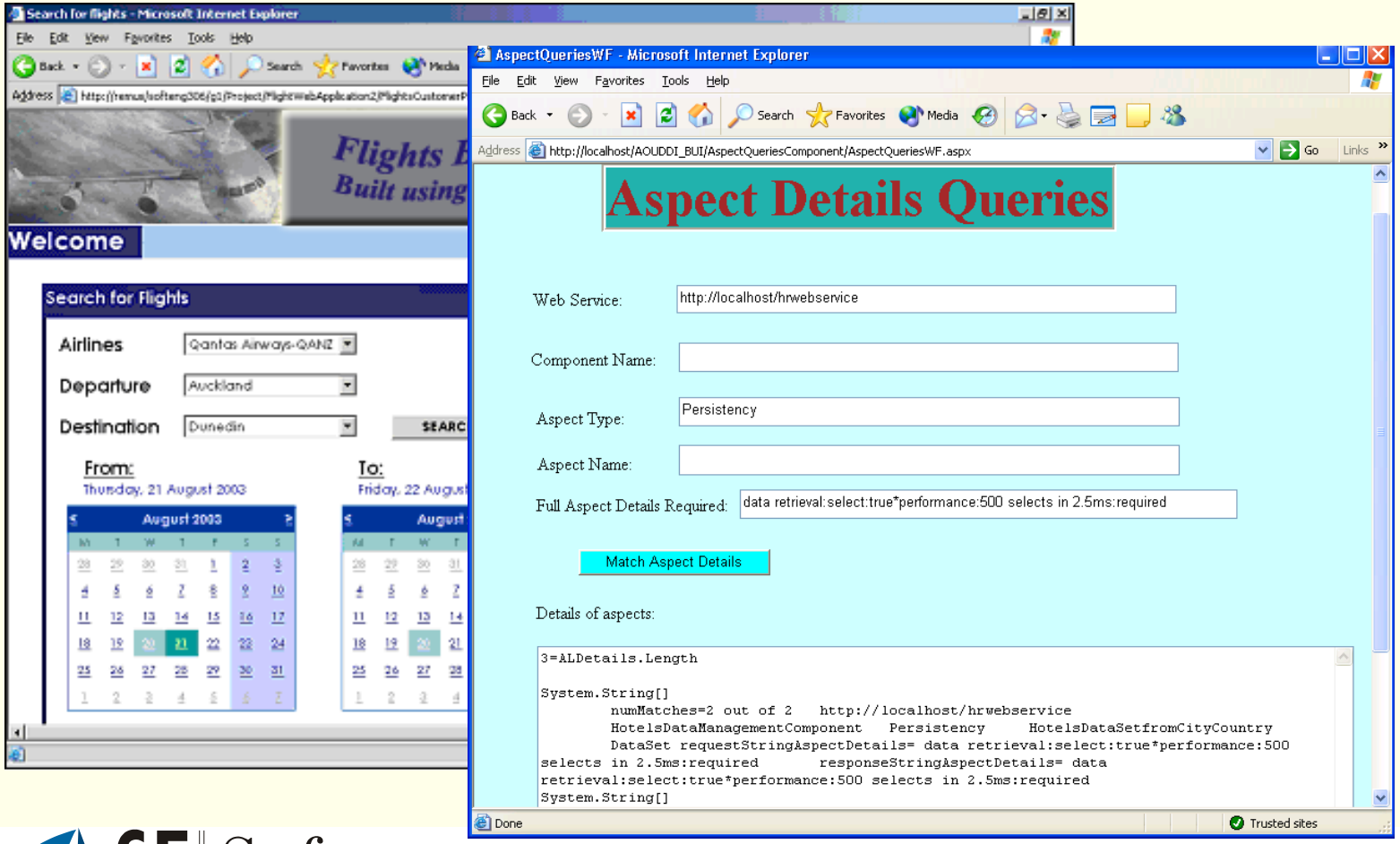
Extensible Software Tools

IST 2000
S-P&E 2002
ASWEC 2005



Web Services-extended AOCE

NCWS 2003
ASAW 2004

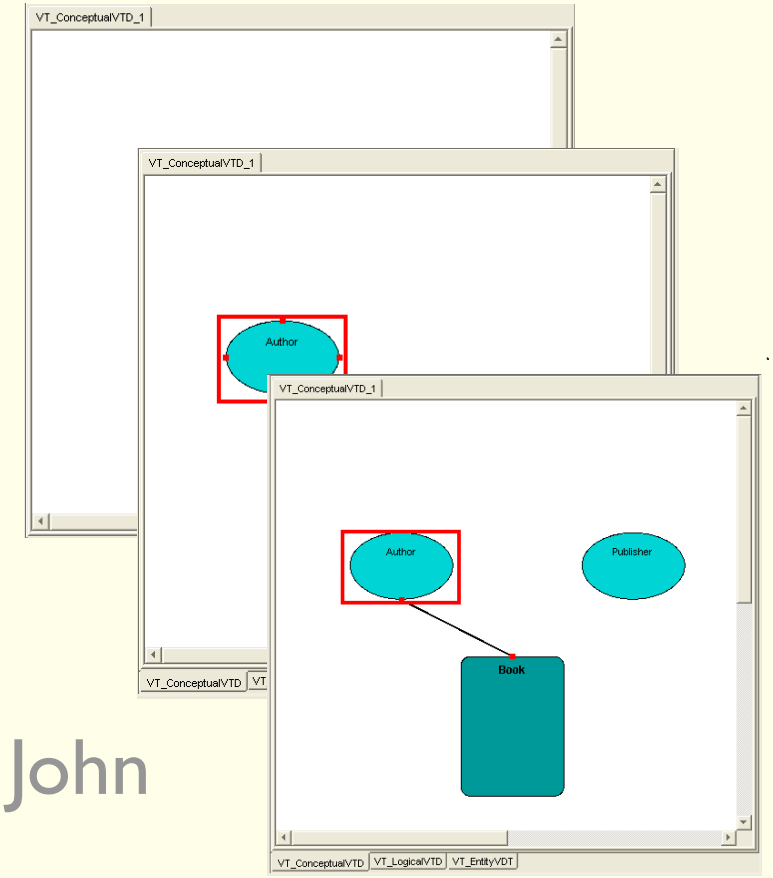


SE Software Engineering
The University of Auckland

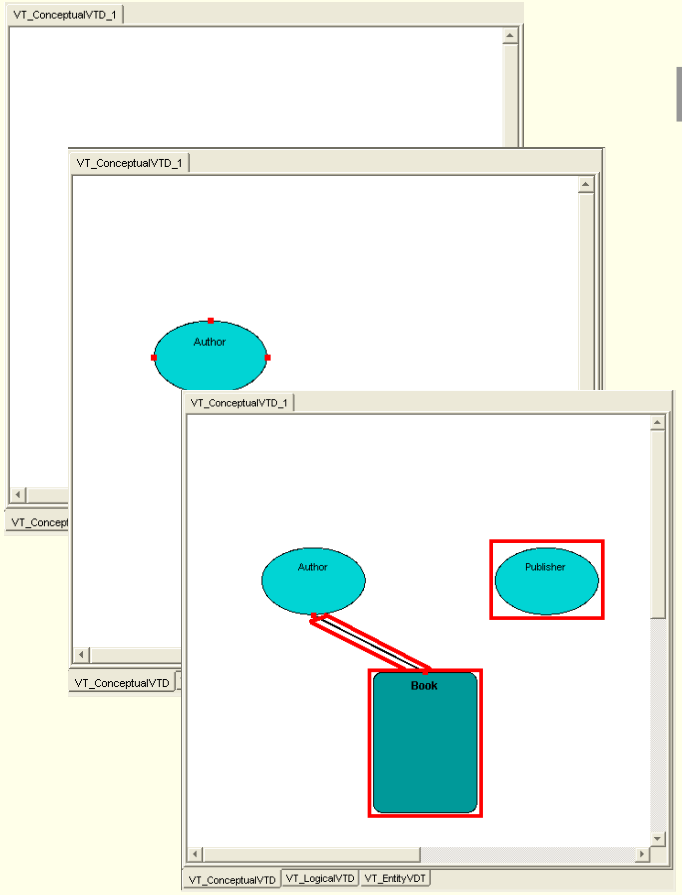
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Collaboration Components

S-P&E 2002
WoDiSEE 04



John

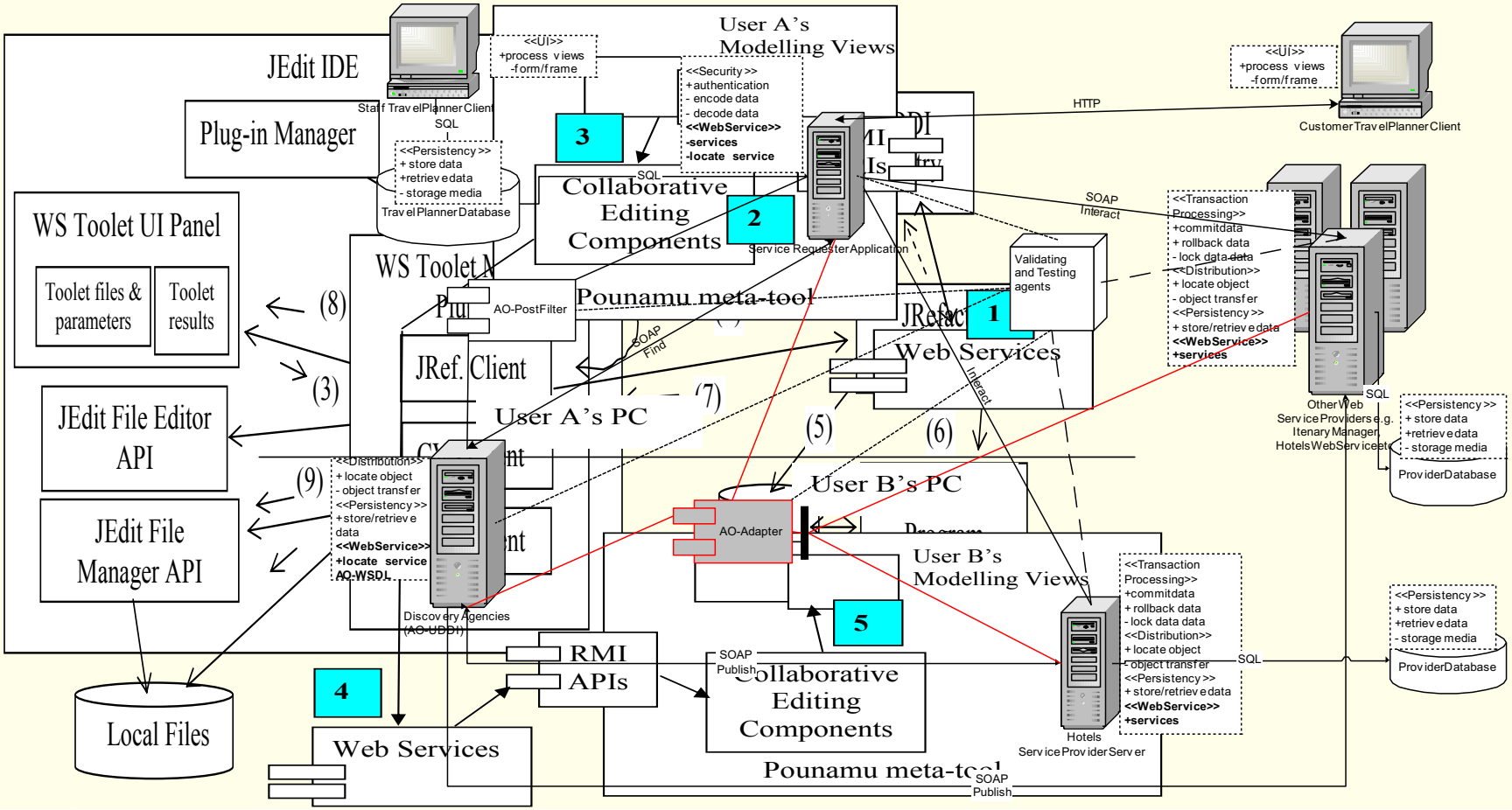


Mark

SE Software Engineering
The University of Auckland

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

How achieved/achieving…

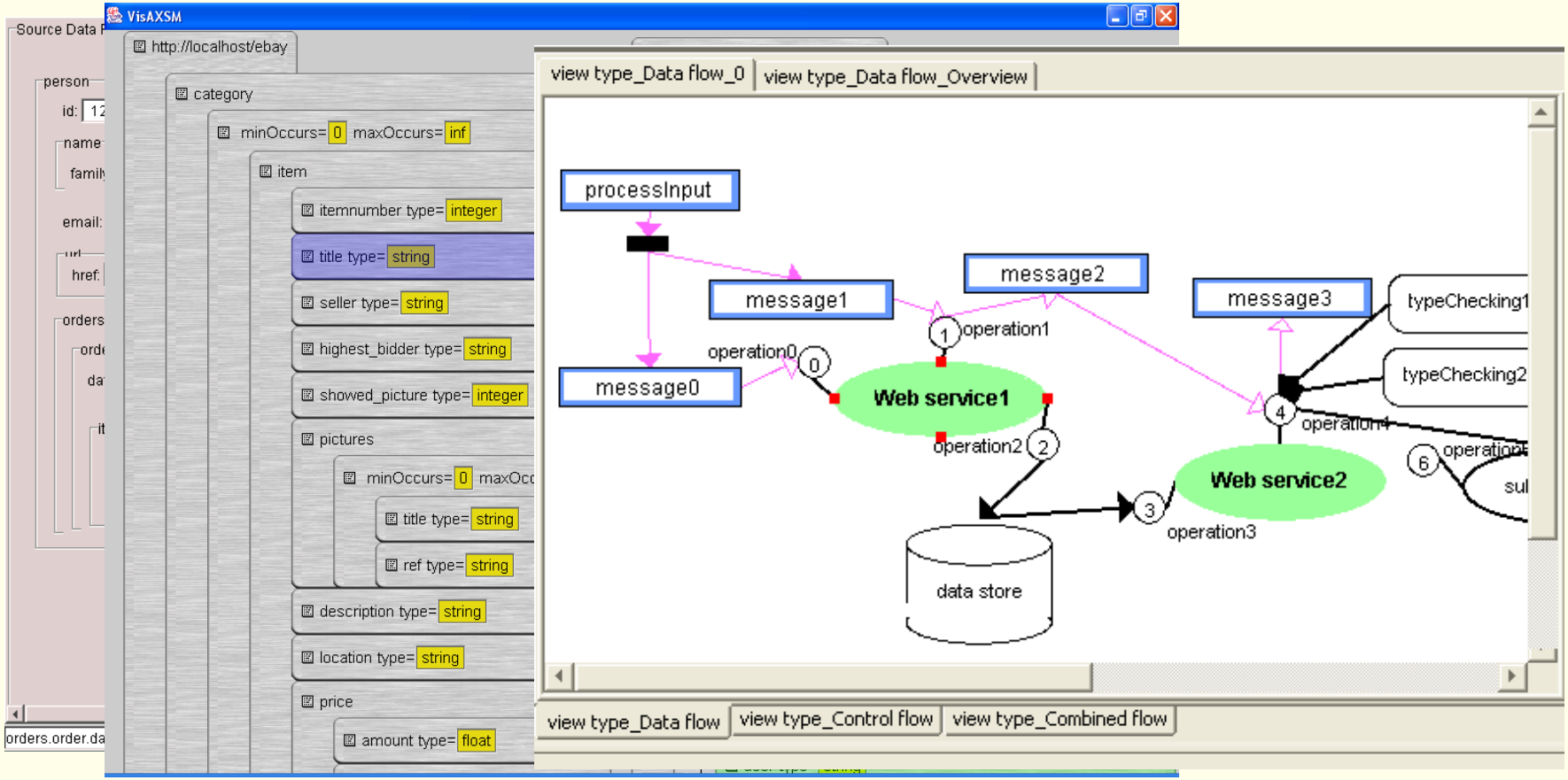


Software Engineering
The University of Auckland

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Integration Visual Languages

HCC 2002
ASE 2004
JVLC 2004



Software Engineering
The University of Auckland

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Conclusions

- Automated Software Engineering = generate/adapt software from high-level models
- Our work focuses on component-based system composition, synthesis of UIs, and architectural enhancements to assist these
- Promising results to date in these areas
- Future work includes more formal specifications of components; better tools to support composition/integration/synthesis
- Commercialisation of some of this research underway