Directions in Innovative Software Engineering

Professor John Grundy
Dept Electrical and Computer Engineering &
Dept Computer Science
University of Auckland, New Zealand

Outline

- Some trends in Software Engineering
- UoA Software Tools Group research activities
- Some examples of our current research – basic and applied:
  - Domain-specific visual language tools
  - Model-driven software engineering tools
  - Service-oriented software engineering tools
    - Aspect-oriented software engineering tools
- Some of our future work plans
- Conclusions

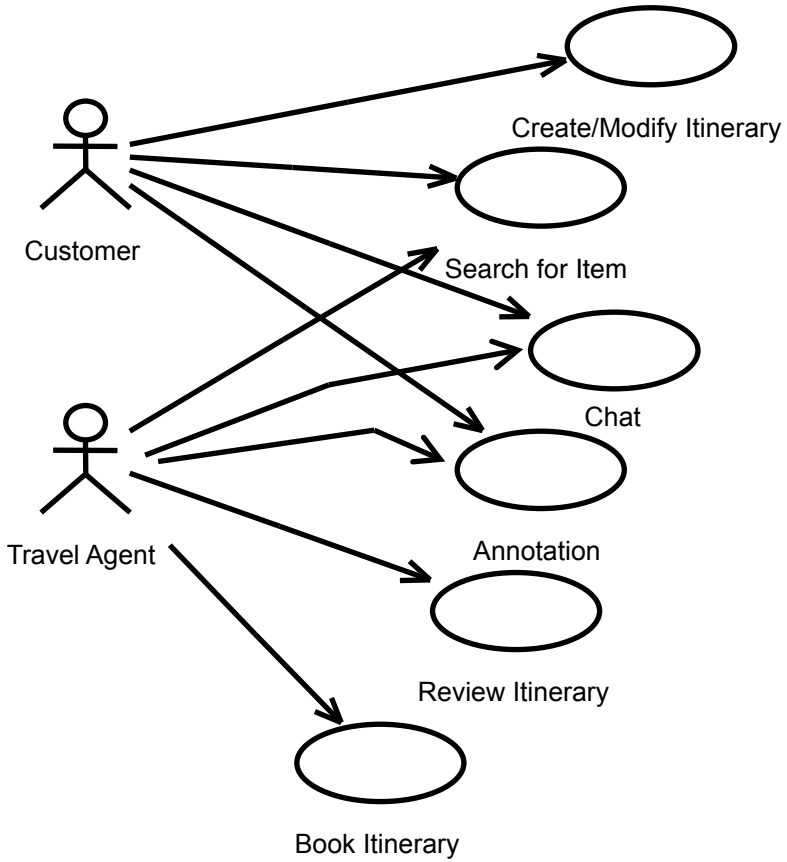
Trends

- You've probably heard this several times before ☺:
  - Software is getting ever-more complex
  - Key requirement is for higher quality software and more agile response to end-user needs
  - Moving to service-oriented delivery of (software) system solutions
  - Increasing need for self-healing, autonomic computing systems
  - Desire for "open-ness" – open source, standards, inter-operability (software, devices, networks…)
  - Ideally – end-users have ability to configure/ change systems THEMSELVES…

How can we do this??

- Model-driven development – generate from high-level models vs hack code
- Domain-specific (visual) languages and tools – close-to-end-user approaches to authoring of models
- Service-oriented architectures – compose systems as need to from reusable, 3rd party services
- Software product lines – framework of solutions vs lots of once-offs
- Aspect-oriented software development – abstract cross-cutting concerns
- Agent-oriented systems – autonomous and emergent behaviours
- Open-source software development – collaborative engineering
- Agile engineering processes – response to changing requirements
- End-user development – users do their own development ☺
- Evaluation and improvement – continuous improvement paradigm

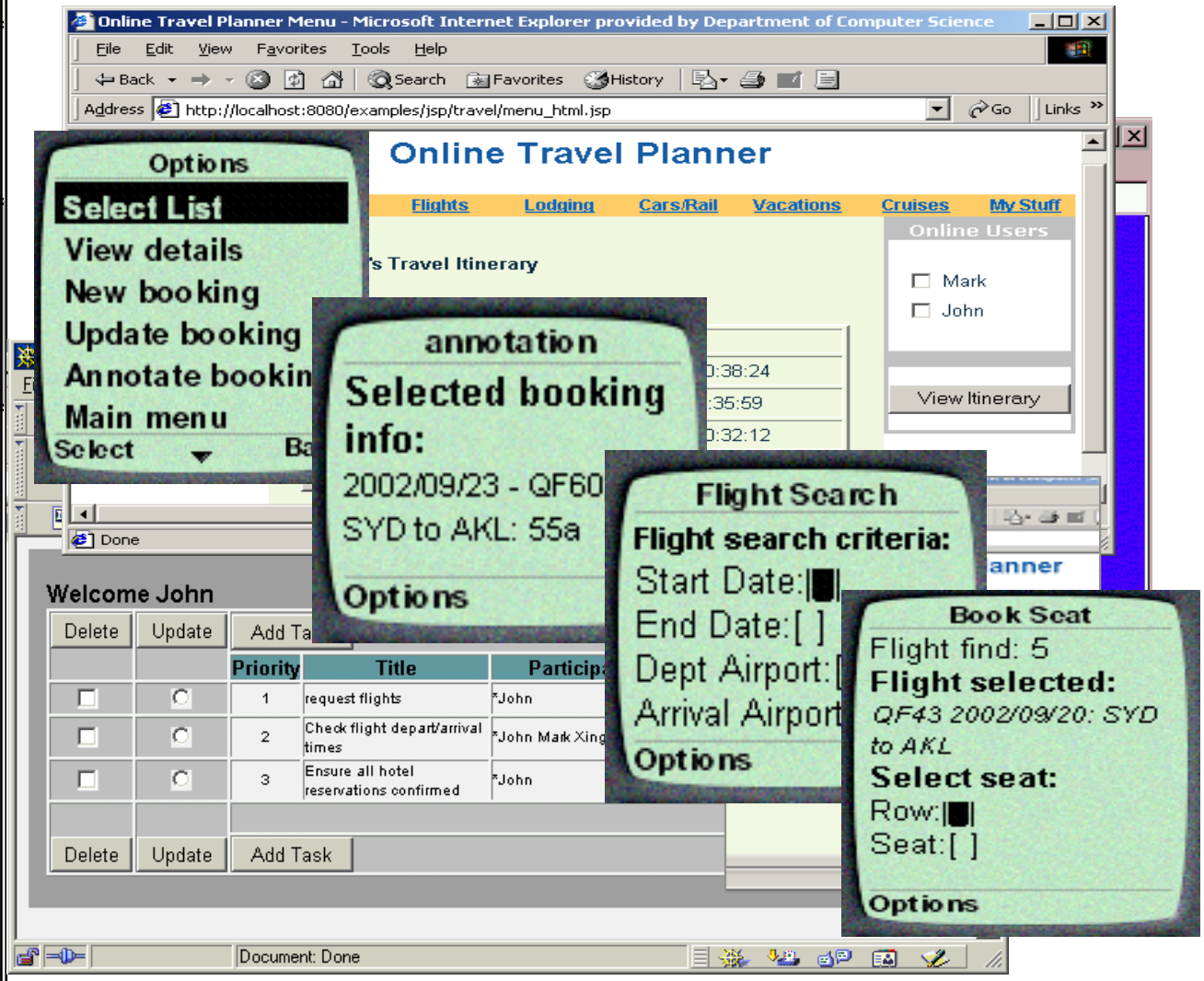
Example…



- Example: collaborative travel planner
- Want to build from services vs components/programs
- Need to integrate 3rd party components & services
- Dynamically integrate and change comps on-the-fly
- Users might want to use different devices e.g. laptop vs PDA vs phone
- Might want to share same user interface between different users

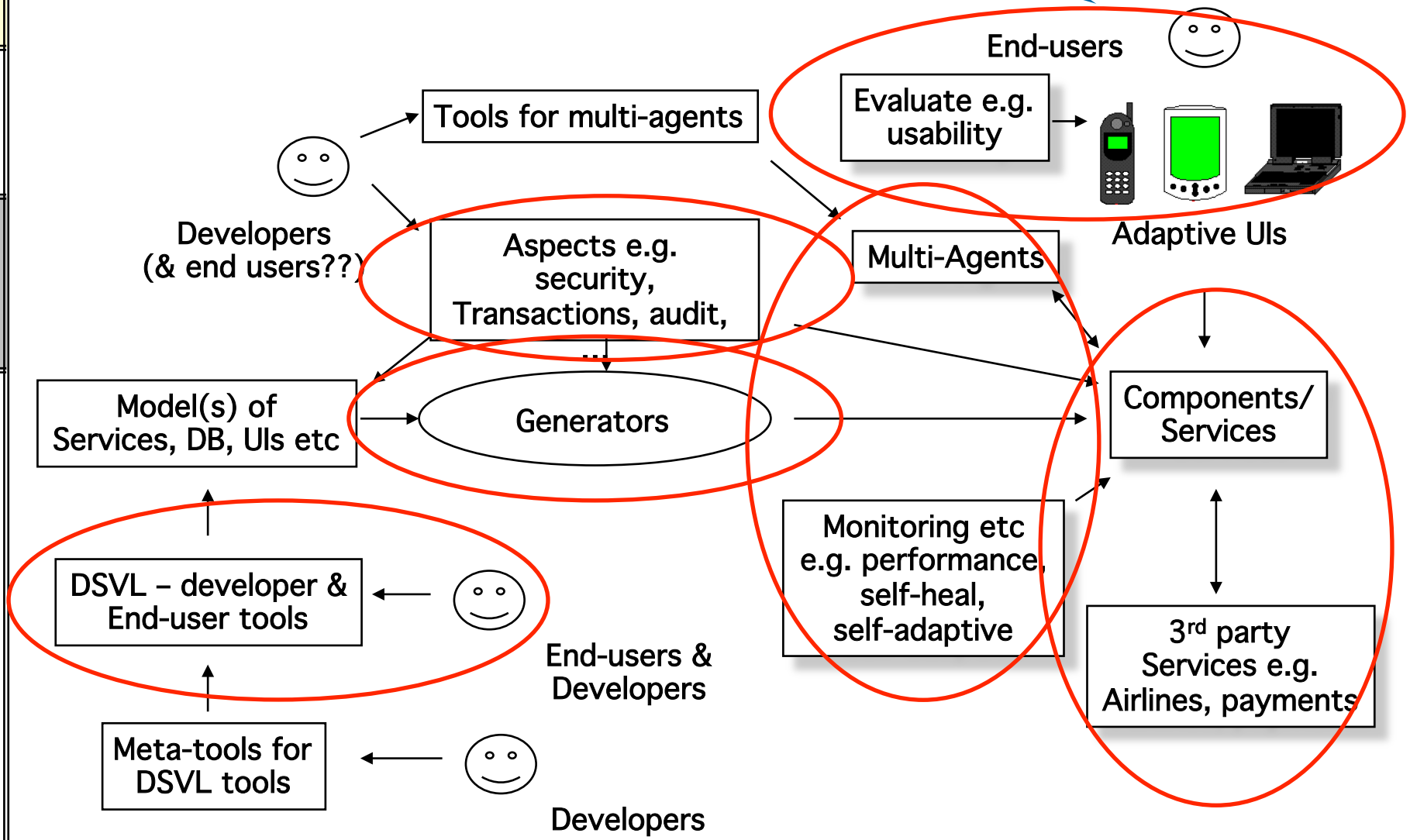
Example

Software Engineering
The University of Auckland



- Thick-client UIs & web-based/ mobile-based UIs
- Collaborative work support
- Built from set of services (some 3rd party)

(Desired) Approach



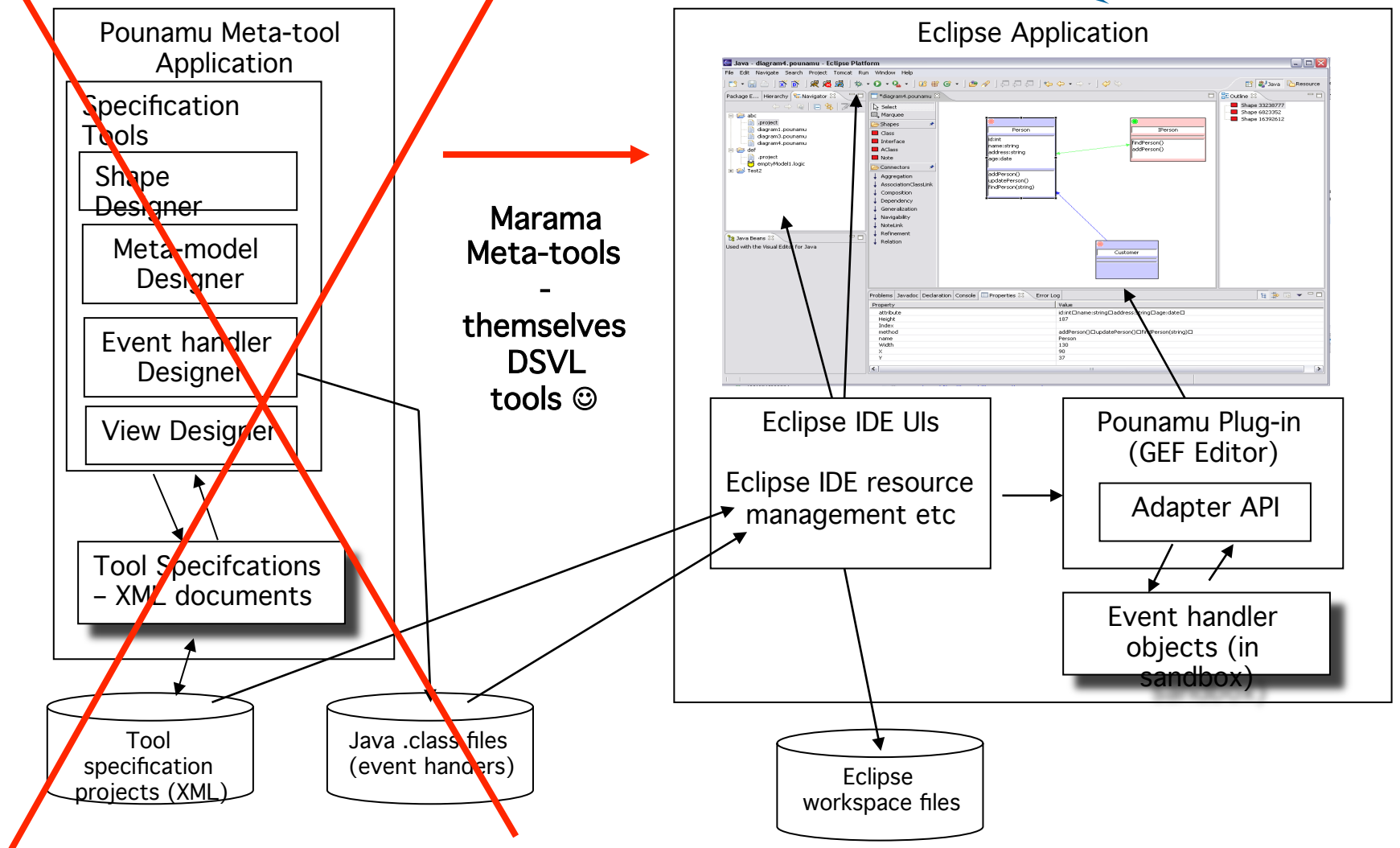
UoA Software Tools Group Research

- Meta-tools (tools for building software tools…)
- **Domain-specific visual languages**
- **Model-driven software engineering**
- Adaptive user interfaces
- Aspect-oriented software development
- **Service-oriented architectures** & adaptive architectures
- Visual languages and software visualisation
- Light-weight evaluation methods and tools
- Collaborative work support tools
- Software processes and process technology

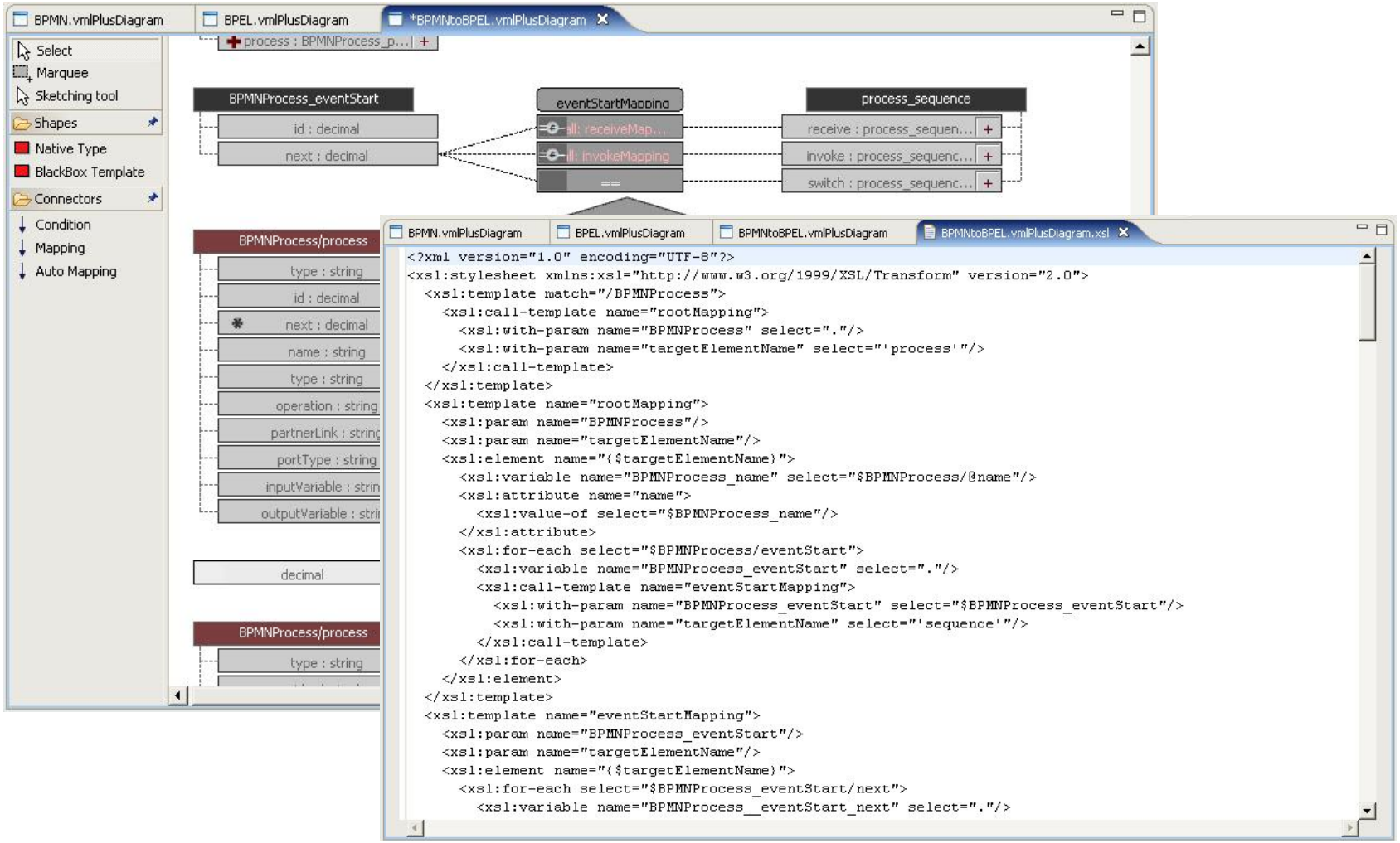
Domain-specific Visual Language Tools

- Meta-tools:
  - JViews/JComposer
  - Pounamu
  - Marama
  - Eclipse plug-ins
- Examples of DSVL tools:
  - Data mapping tools
  - Process modelling tools
  - Enterprise & software modelling tools
  - Event response behaviour tools
  - UI design tools
  - Statistical survey design tool
- Industry examples:
  - XSOL, Orion Systems, WizzBang
- (Light-weight) evaluation methods (tools are coming… ☺):
  - Cognitive Dimensions
  - Attention investment
  - Champagne prototyping (or should that be Methode Champagnois?? ☺)
  - Plus various heavy-weight usability evaluation methods

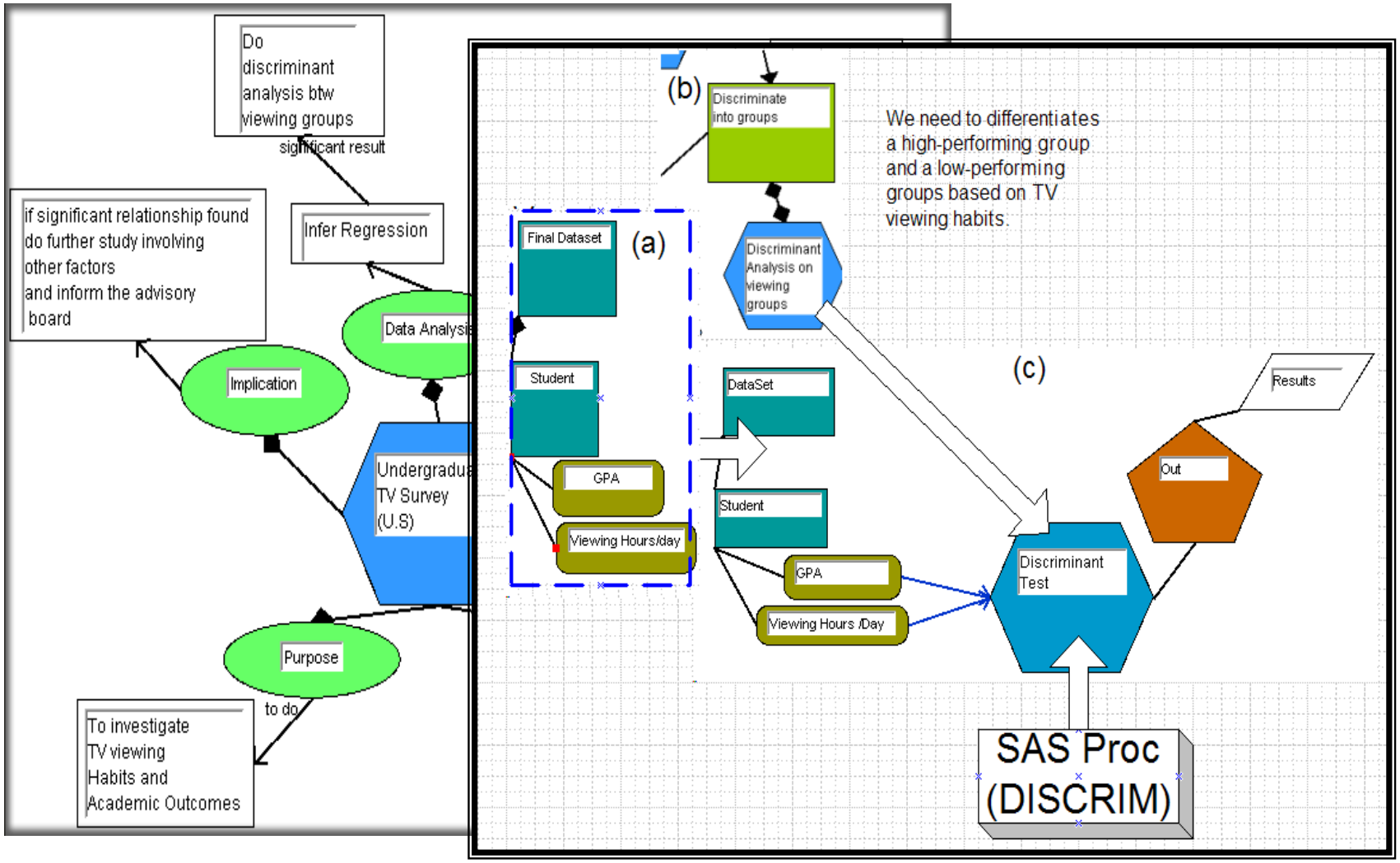
Marama - Eclipse plug-ins



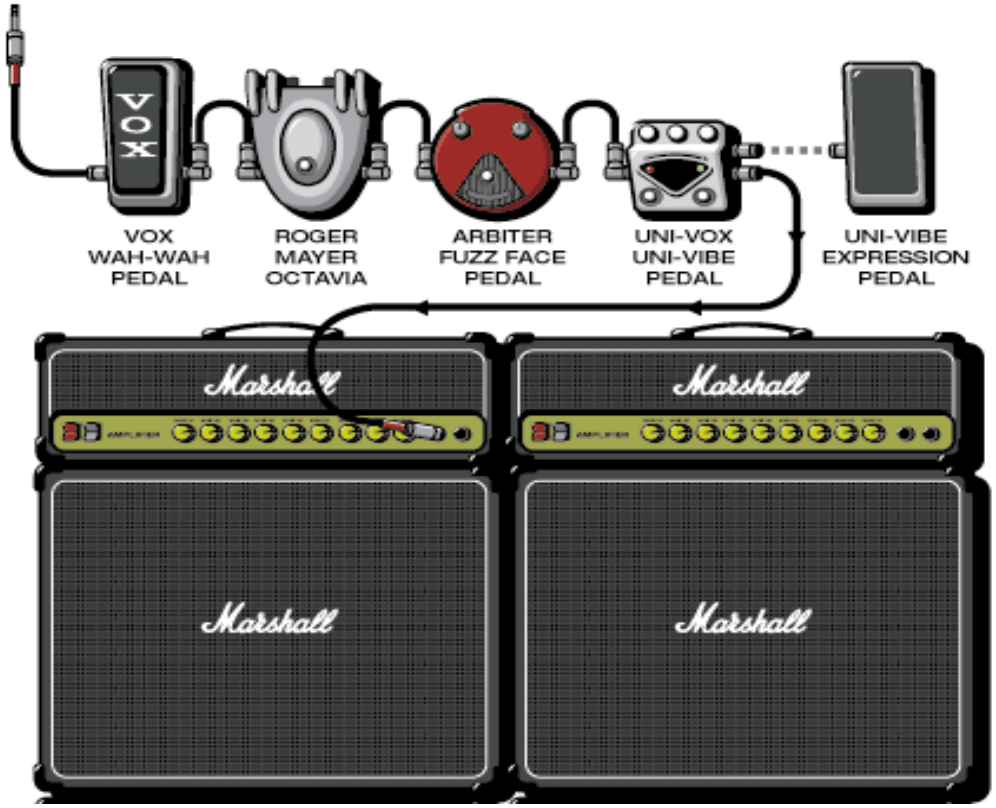
MaramaVMLPlus - complex data mapping



SDLTool – Statisical Survey Design tool

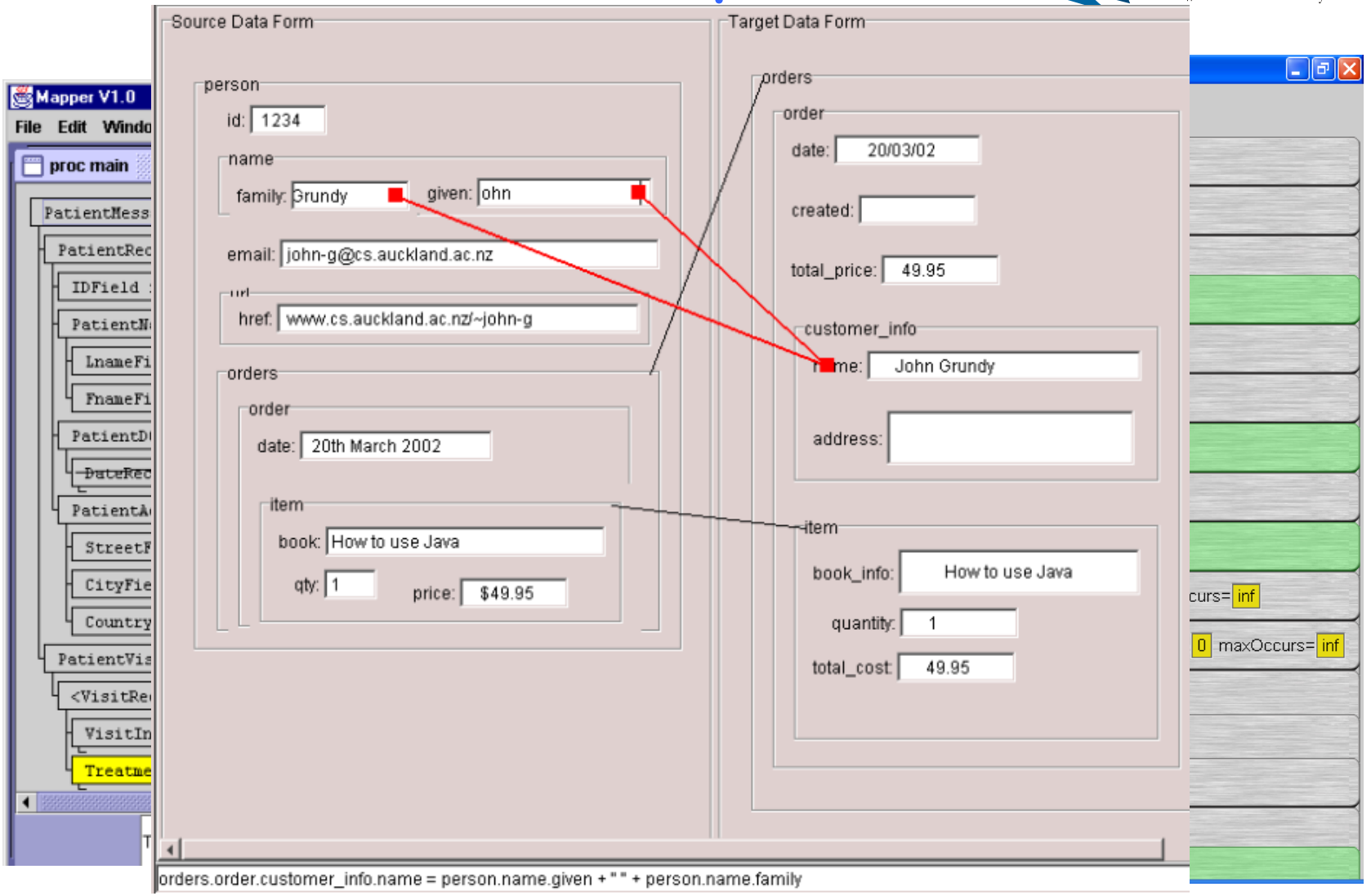


Other DSVLs…



Music – Jimi Hendrix set-up… (from guitargeek.com)

Industrial examples



DSVL Evaluation – Cognitive Dimensions Framework

- Green and Petre 1996 (since developed by Blackwell)
- Establishes a set of "dimensions" to think about the tradeoffs made in implementing visual programming *environments*
  - Means of explaining effects of design decisions
  - Has had very strong influence on the VL community
- Comes out of cognitive psychology community
- Lightweight – doesn't need large usability studies to get useful insight
- Can be used for evaluation and also as a design aid

Example
Cognitive Dimensions

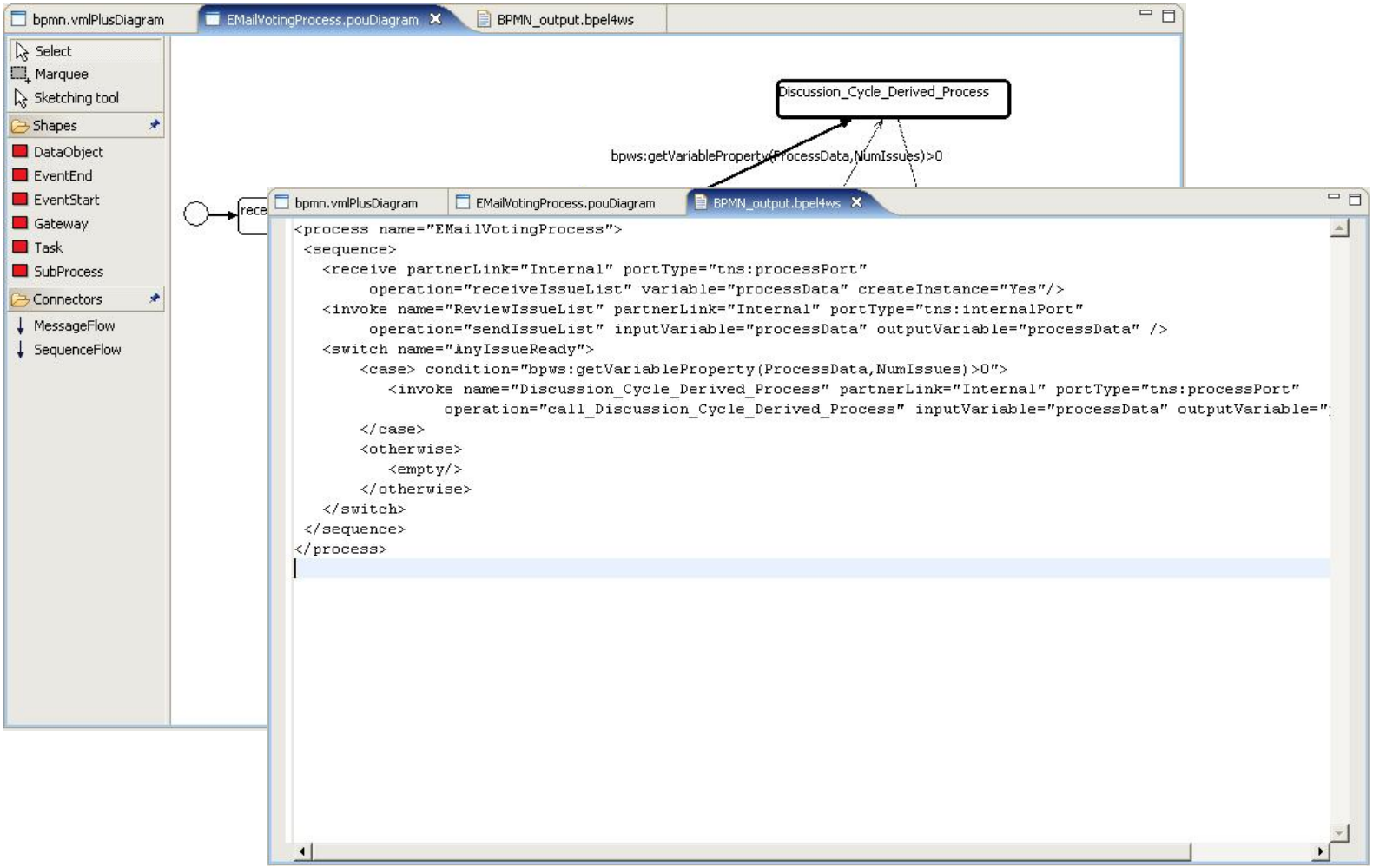
- *Abstraction gradient* What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?
- *Closeness of mapping* What 'programming games' need to be learned?
- *Consistency* When some of the language has been learnt, how much of the rest can be inferred?
- *Diffuseness* How many symbols or graphic entities are required to express a meaning?
- *Error-proneness* Does the design of the notation induce 'careless mistakes'?
- *Hard mental operations* Are there places where the user needs to resort to fingers or penciled annotation to keep track of what's happening?
- *Hidden dependencies* Is every dependency overtly indicated in both directions? Is the indication perceptual or only symbolic?

Model-driven Software Engineering Tools

- Meta-tools:
  - Marama
  - MaramaVMLPlus
  - Eclipse JET, ALT; QVT; XSLT
- Examples:
  - EML
  - MaramaMTE
  - Adaptive User Interface Technology
- Industry examples:
  - XSOL, Peace Software, Orion Systems, iVistra

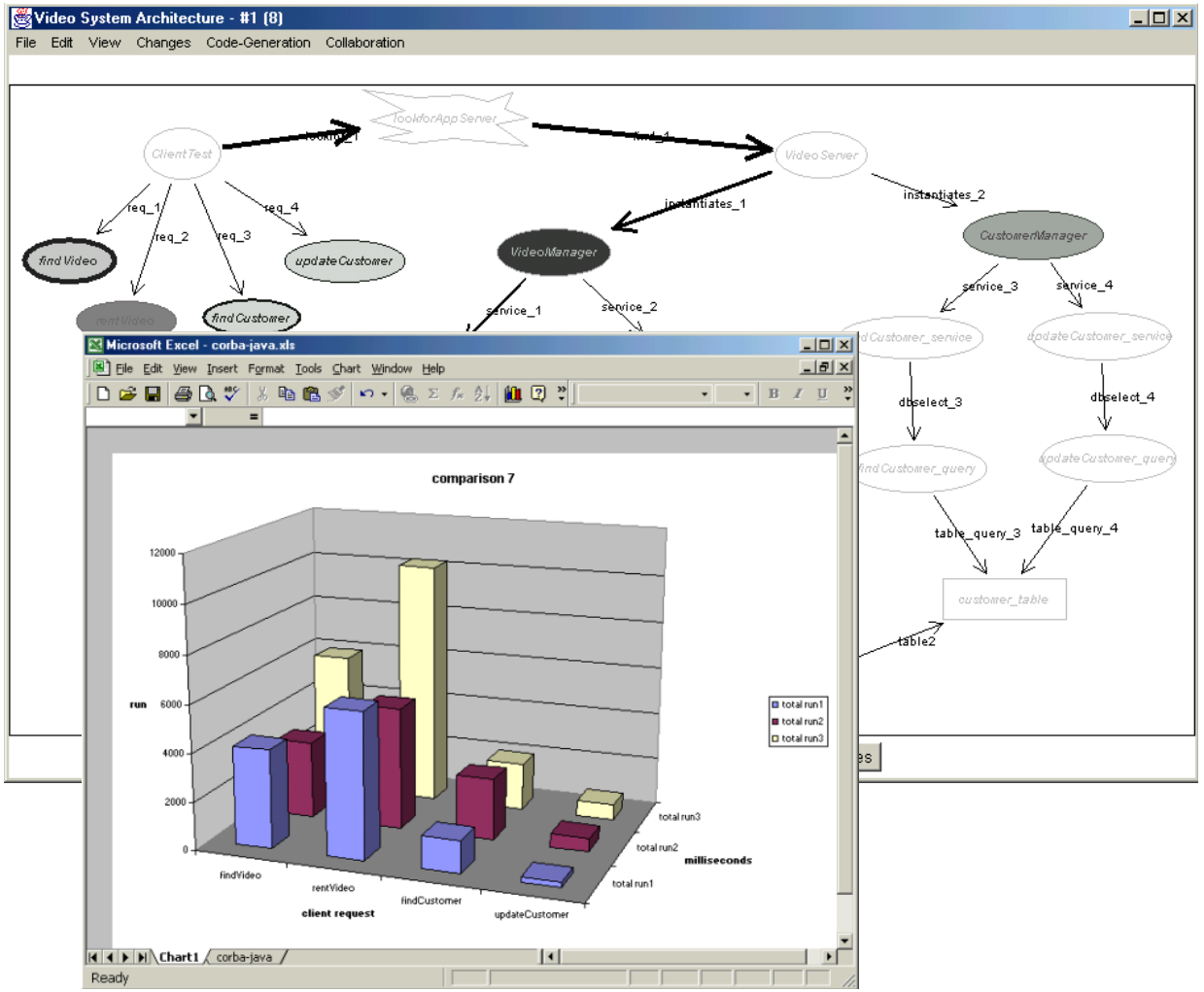
Enterprise Modelling Language

Software Engineering
The University of Auckland

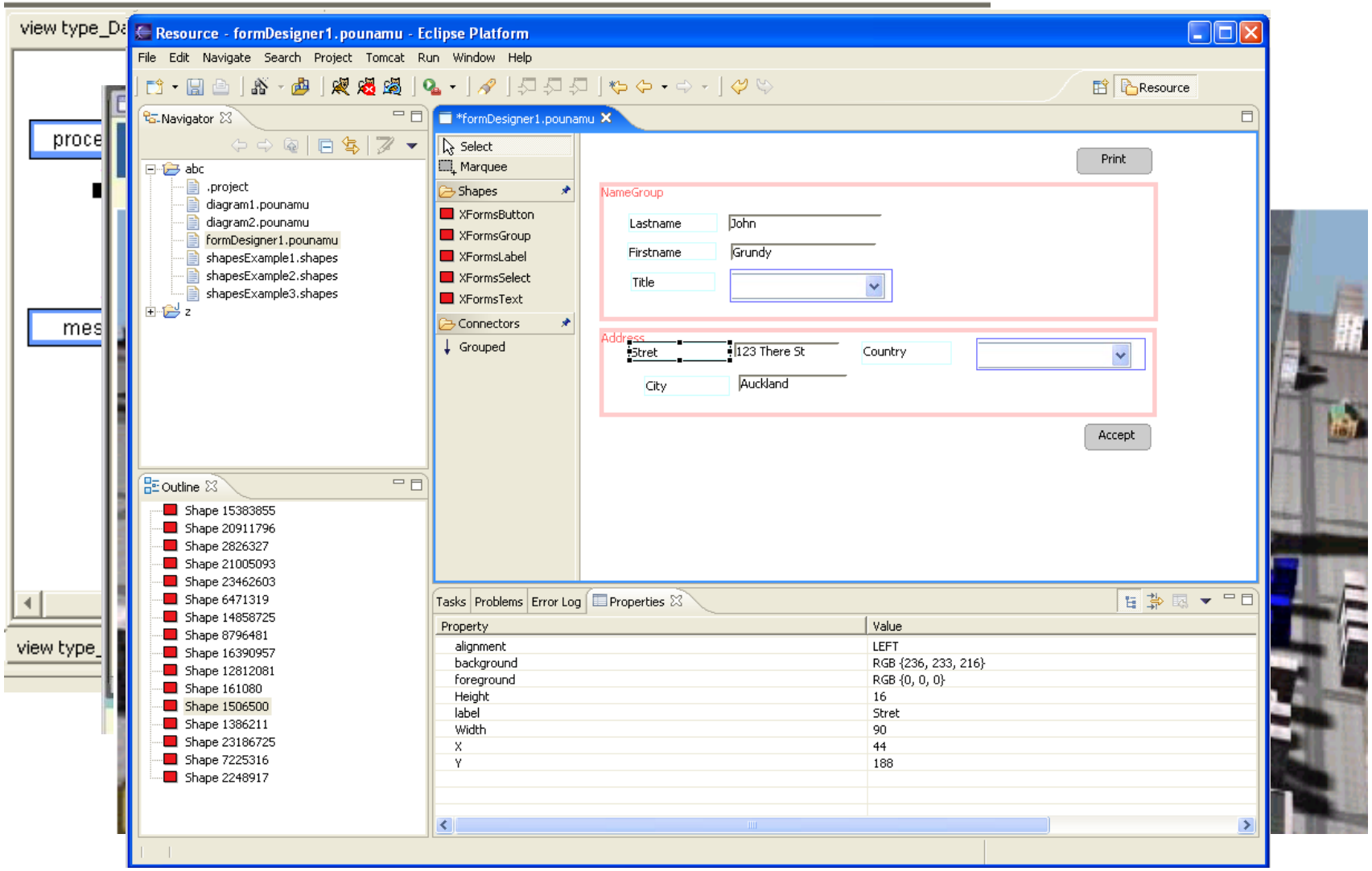


MaramaMTE - Performance Engineering tool

Software Engineering
The University of Auckland



Industrial Examples



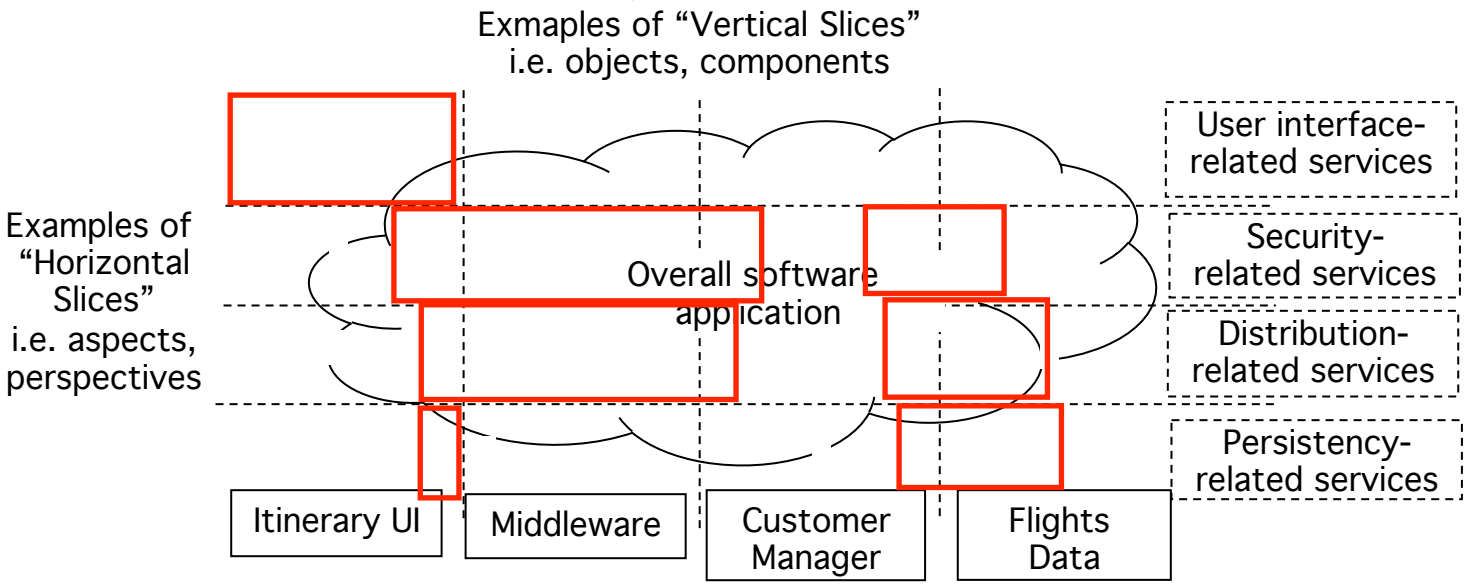
Service-oriented Software Engineering Tools

- AOCE-WS (Aspect Oriented Component Engineering)
- Enterprise Modelling Language
- ViTABaL-WS
- IMAL
- JEdit-WS
- AUIT
- …

- Industry projects:
  - XSOL, Peace Software, Orion Systems, iVistra, …

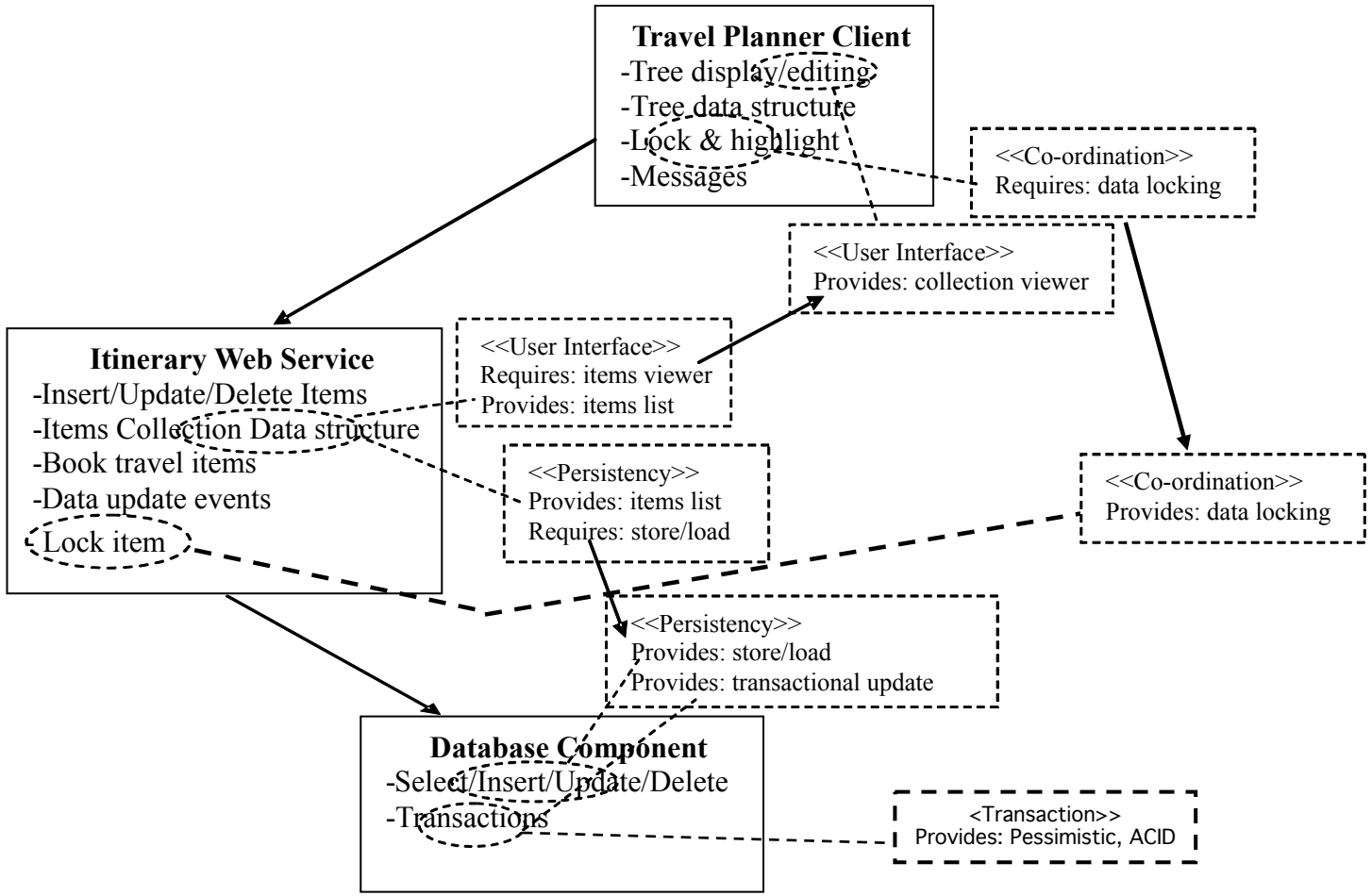
SE Software Engineering
The University of Auckland

Aspects

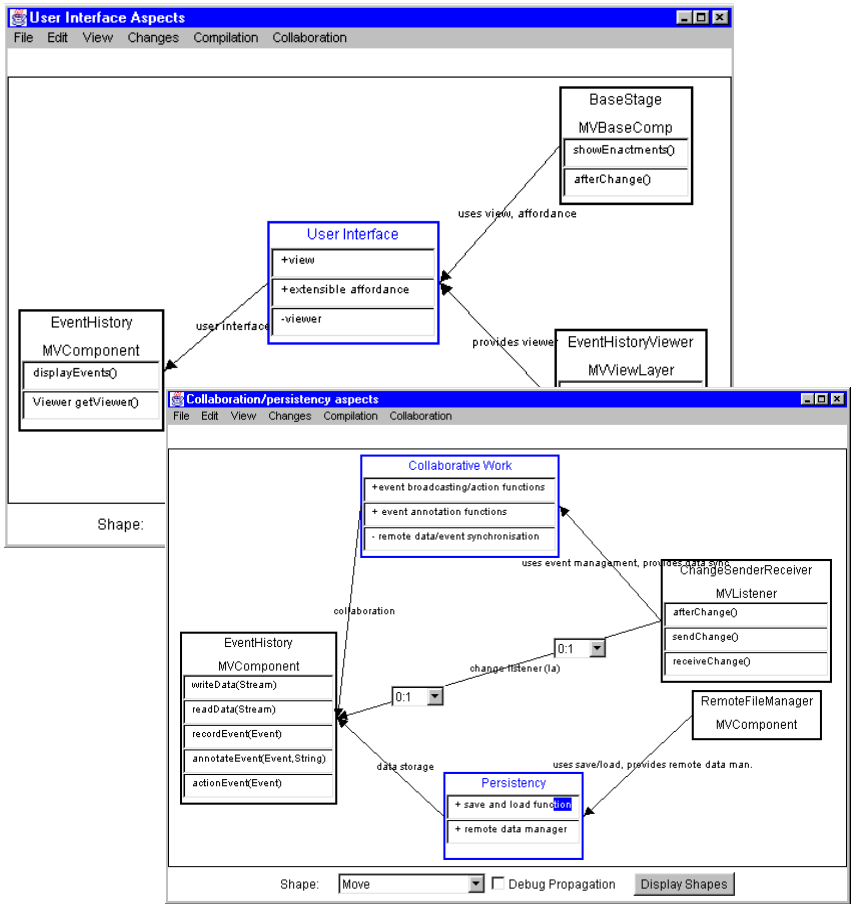


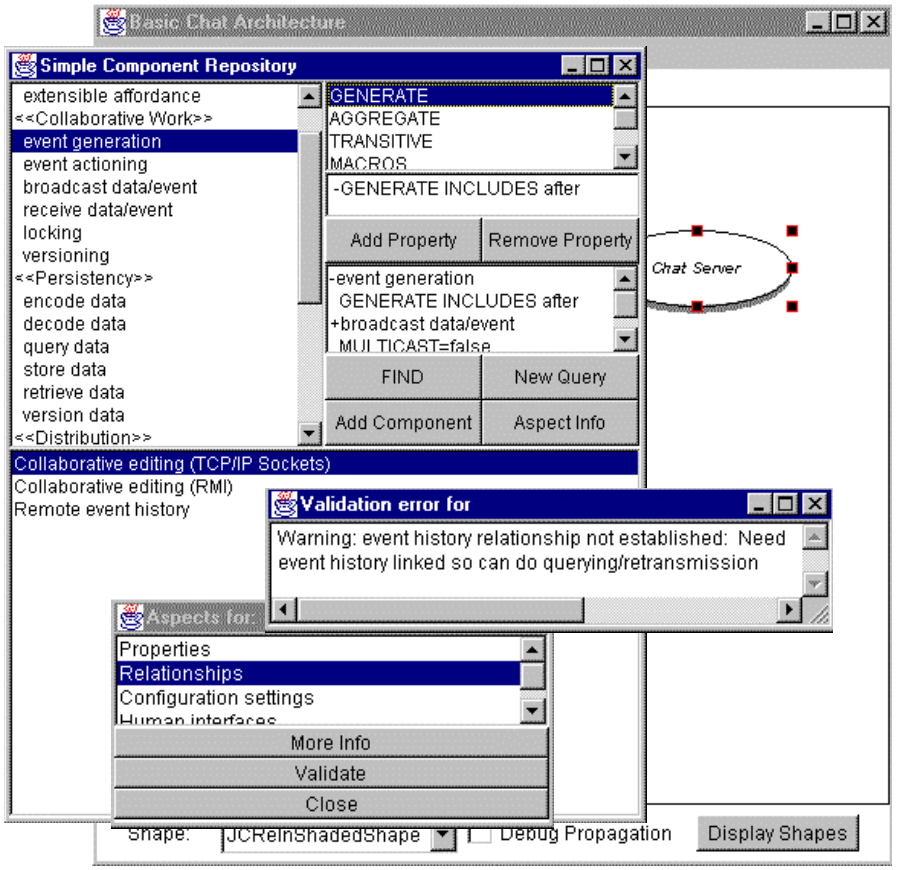
- Functional decomposition - normal approach
- Alternatives: parts of system contributing to "systemic" properties e.g. UI, persistency etc
- Systemic properties of system get spread…

AOCE-WS

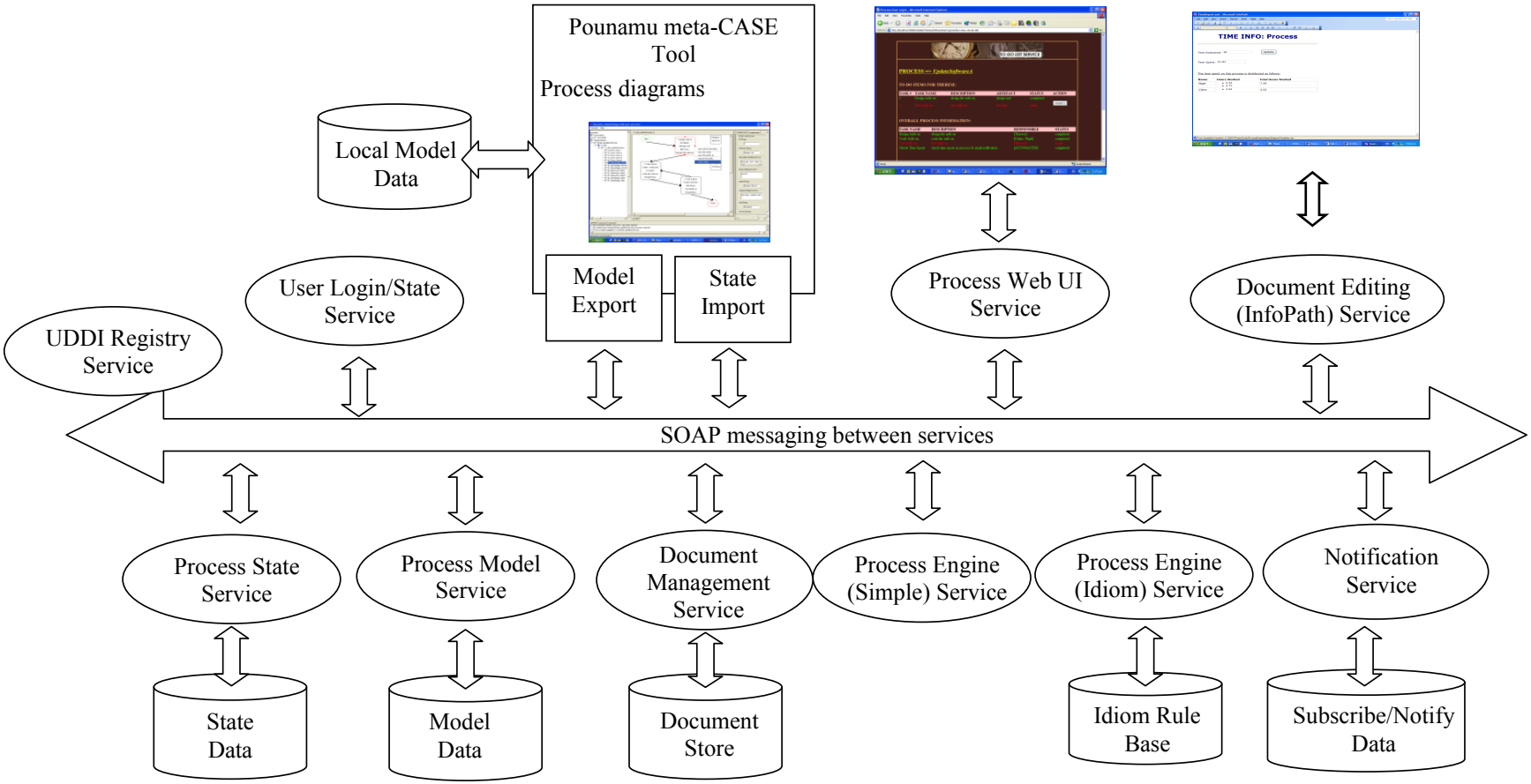

Travel Planner Client
-Tree display/editing
-Tree data structure
-Lock & highlight
-Messages
<<Co-ordination>>
Requires: data locking
<<User Interface>>
Provides: collection viewer
Itinerary Web Service
-Insert/Update/Delete Items
-Items Collection Data structure
-Book travel items
-Data update events
Lock item
<<User Interface>>
Requires: items viewer
Provides: items list
<<Persistency>>
Provides: items list
Requires: store/load
<<Co-ordination>>
Provides: data locking
<<Persistency>>
Provides: store/load
Provides: transactional update
Database Component
-Select/Insert/Update/Delete
-Transactions
<Transaction>>
Provides: Pessimistic, ACID

Tool Support for AOCE-WS

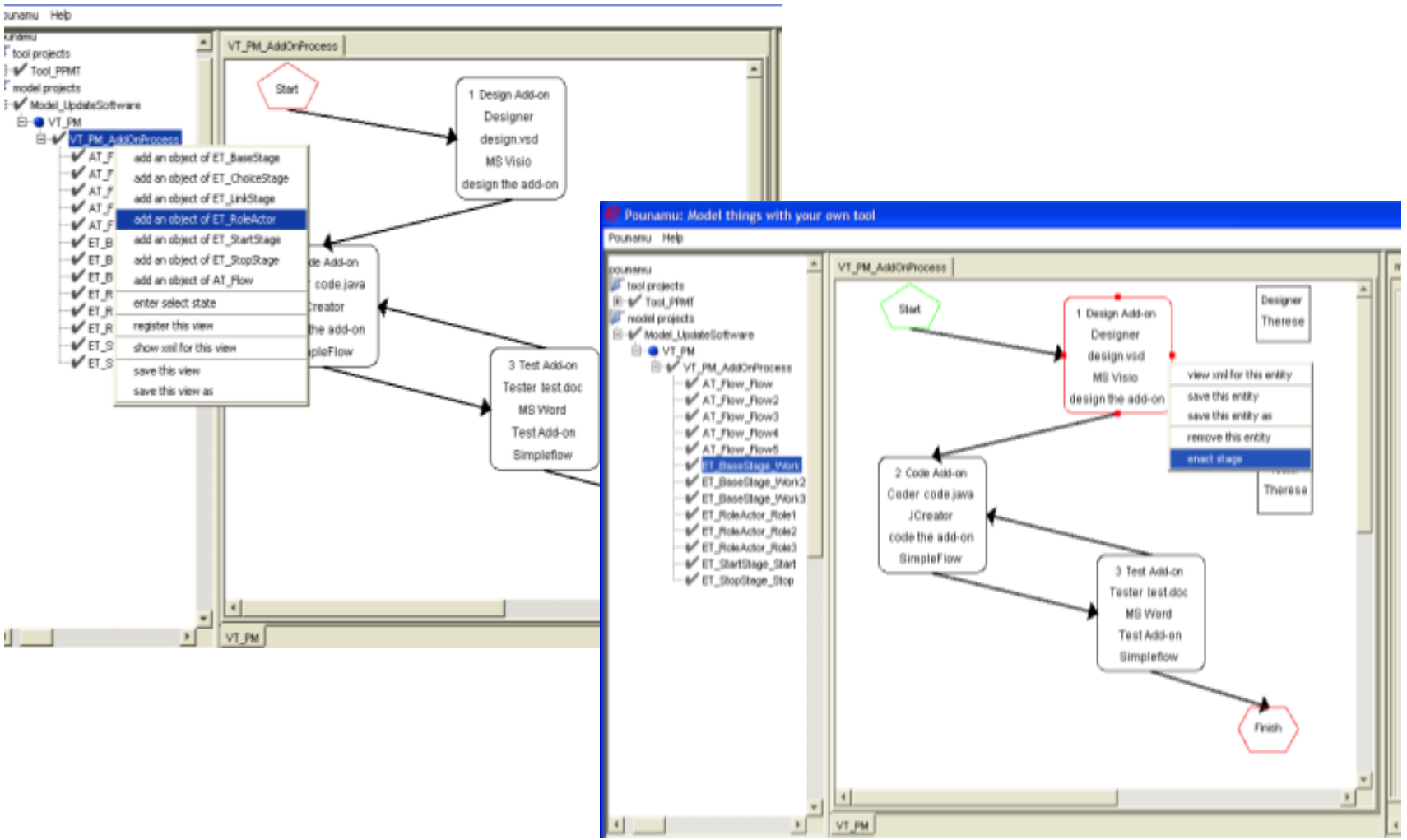




Example - IMAL

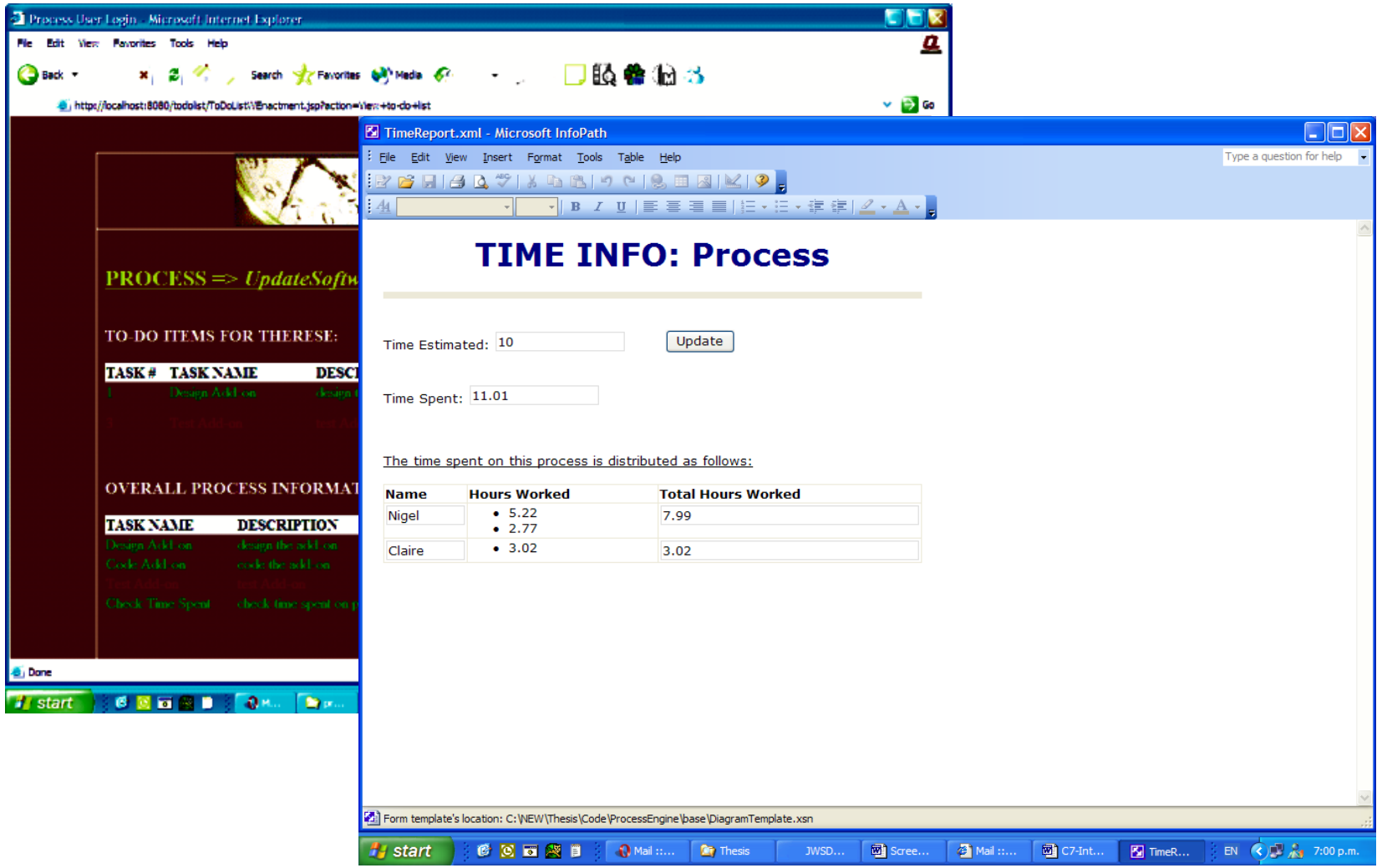


IMAL Process Modelling



IMAL Run-time

SE Software Engineering
The University of Auckland



Problems still to overcome…

- Design of DSVLs:
  - Good visual metaphors/building blocks
  - Evaluation of effectiveness
  - Building DSVL tools
  - End-user DSVLs & tools
- Model engineering:
  - Scalability and complexity of models
  - Correctness of models
  - Mapping/translating between (very) complex models
  - Integrating approaches & tools
- Service-oriented systems engineering:
  - Design of services – what is a service anyway??
  - Engineering of service-based systems – process, methods, tools
  - Scalability, performance, reliability, security, …
  - Self-healing; adpative service-based systems

Planned Future Work & Approaches

- Current/proposed projects:
  - Domain-specific Software Tools project (current FRST NERF project – sort-of very Large ARC) – meta-tools, design critique tools, collaborative work tools, model evaluation tools, MDD tools, software artefact reuse approaches
  - Several Technology NZ "TBG" projects – XSOL, Peace, WizzBang, iVistra, … - applied R&D - like DEST projects – these apply DS tools group research to practical industry problems
  - Meta-tools for DSVLs & MDD (Large FRST NERF proposal)
  - Software Improvement Research Lab (Large FRST SER proposal)
  - Aspect-oriented, agent-oriented, service-oriented systems engineering (Marsden proposal – blue skies basic research)
  - Open Systems Research Lab/Repository ("CoRE" proposal)
- Approaches to fostering academic/industry research:
  - Usability lab; Light-weight eval methods & tools
  - ICT Innovation Academy – local industry internship programme
  - Centre for Software Innovation (our "little brother to NICTA"… ☺)

Conclusions

- Building software is still hard
- Complexity management; quality assurance; dynamic change are the key nasties…
- Not just academic problems - Industry needs good solutions now!

- Solutions we are pursuing:
  - Model-driven engineering
  - Service-oriented architectures (+agents + aspects…)
  - Domain-specific visual languages & meta-tools
  - Software process & product improvement

- Working closely with industry to apply research outcomes
- Developing better tools to make all of these feasible!

References

- Grundy, J.C, Hosking, J.G., Amor, R., Mugridge, W.B., Li, M. Domain-specific visual languages for specifying and generating data mapping systems, Journal of Visual Languages and Computing, vol. 15, no. 3-4, June-August 2004, Elsevier, pp 243-263.
- Grundy, J.C., Hosking, J.G., Zhu, N. and Liu, N. Generating Domain-Specific Visual Language Editors from High-level Tool Specifications, In Proceedings of the 2006 IEEE/ACM International Conference on Automated Software Engineering, Tokyo, 24-28 Sept 2006, IEEE.
- Bossung, S., Stoeckle, H., Grundy, J.C., Amor, R. and Hosking, J.G. Automated Data Mapping Specification via Schema Heuristics and User Interaction, In Proceedings of the 2004 IEEE International Conference on Automated Software Engineering, Linz, Austria, September 20-24, IEEE CS Press, pp. 208-217.
- Stoeckle, H., Grundy, J.C. and Hosking, J.G. A Framework for Visual Notation Exchange, Journal of Visual Languages and Computing, Volume 16, Issue 3 , June 2005, Elsevier, pp.187-212.

- 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: Special Issue on Constructing Software Engineering Tools, Vol. 42, No. 2, January 2000, pp. 117-128
- Grundy, J.C., Cai, Y. and Liu, A. SoftArch/MTE: Generating Distributed System Test-beds from High-level Software Architecture Descriptions, Automated Software Engineering, Kluwer Academic Publishers, vol. 12, no. 1, January 2005, pp. 5-39.
- Cai, Y., Grundy, J.C. and Hosking, J.G. Experiences Integrating and Scaling a Performance Test Bed Generator with an Open Source CASE Tool, In Proceedings of the 2004 IEEE International Conference on Automated Software Engineering, Linz, Austria, September 20-24, IEEE CS Press, pp. 36-45.

- Grundy, J.C., Ding, G., and Hosking, J.G. Deployed Software Component Testing using Dynamic Validation Agents, Journal of Systems and Software, vol. 74, no. 1, January 2005, Elsevier, pp. 5-14
- Singh, S., Grundy, J.C., Hosking, J.G. and Sun, J. An Architecture for Developing Aspect-Oriented Web Services, In Proceedings of the 2005 European Conference on Web Services, Vaxjo, Sweden, Nov 14-16 2005, IEEE Press.
- 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, World Scientific Publishers, pp. 713-734.
- Grundy, J.C. and Hosking, J.G. Developing Software Components with Aspects: Some Issues and Experiences, Chapter 25 in Aspect-Oriented Software Development, Prentice-Hall, October 2004, pp. 585-604.