Building domain-specific, visual language software engineering tools

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

Outline

- What are domain-specific visual language software tools?
- Examples of some DSVL tools:
  - Data mapping
  - Process management/tool integration
  - Project management
  - User interface design
  - DSVL tool event specification ☺
- Some approaches to building DSVL tools
- Evaluation & future work
- Conclusions

Models in Software Engineering

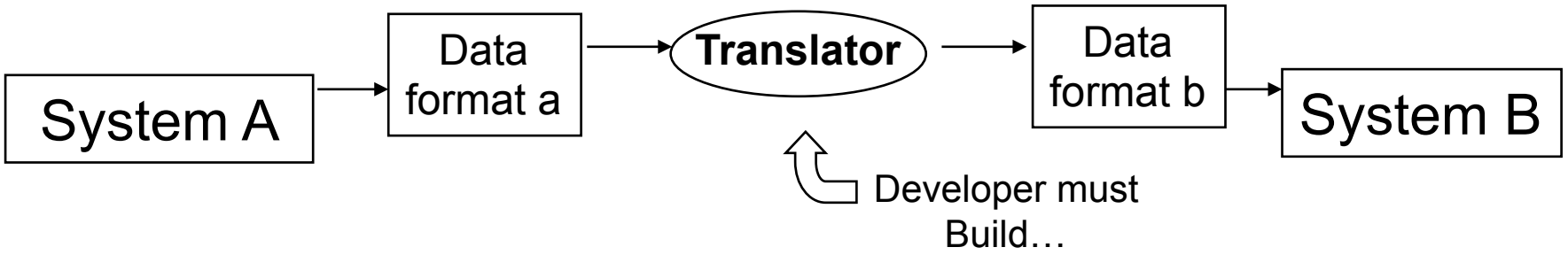
- Much of Engineering is about developing models of engineered products (or rather, models of products to engineer…)
- We've developed models for a whole range of SE "products" and activities:
  - Software processes
  - Requirements
  - Software design
  - Data structures
  - Software architecture
  - Software behaviour
  - Interface design
  - …
- We've also developed visual representations of these models – some are "abstract" (UML, ADLs); some are "concrete" e.g. WYSIWYG UI design…

But…

- Our models often get too complex, too unwieldly, hard to understand/maintain using only "abstract" or "general-purpose" model representations
- Example: any non-trivial Model-Driven Architecture application…
- Domain-specific languages (DSLs) – models that focus on expressing problems in a PART of software engineering, using less general but more expressive constructs
  - E.g. a scripting language for handling event responses
- Domain-specific visual languages provide way to represent such domain-oriented models using a wide variety of visual "metaphor(s)"
- Idea is to have a metaphor providing closer mapping to the problem domain than vanilla, general-purpose abstract model
  - E.g. show event-condition-action rules as flow charts
- DSVL tools provide environment to construct these models, configure existing components, generate code etc.

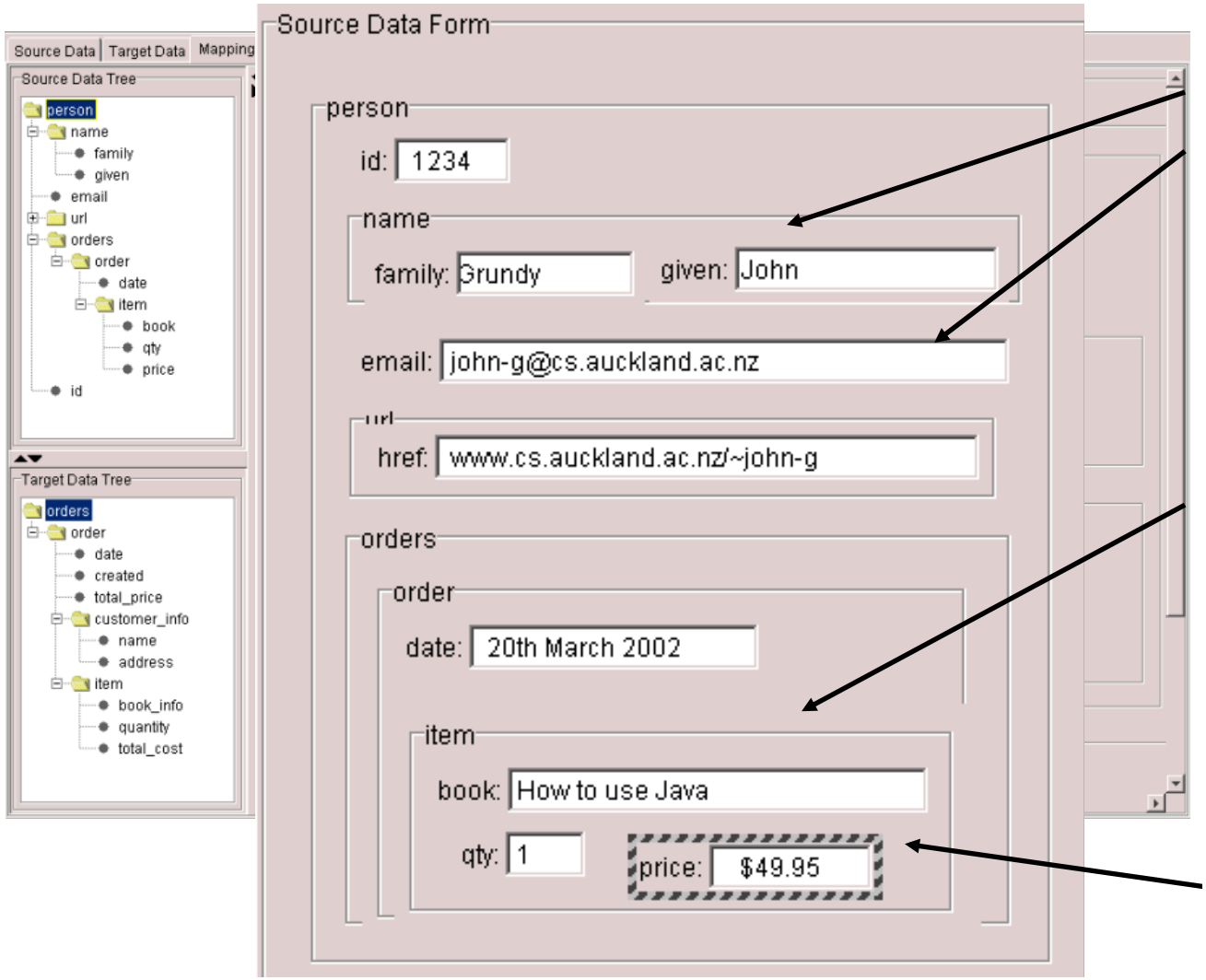
An Example: the Form-based data mapper

- Consider problem of "data mapping" between enterprise systems:



- Development of data translator tools is very tedious, time consuming and error prone using general-purpose langs/tools
- In enterprise system integration, often have "business analysts" who understand meaning of data in each domain, but not how to implement mapping tools using XSLT, Java, or even XML Spy etc.
- Idea: a new tool for translator generation – uses concept of "business forms" as the metaphor to represent source/target system data, and "mappings" between form components…

Form-based data mapping

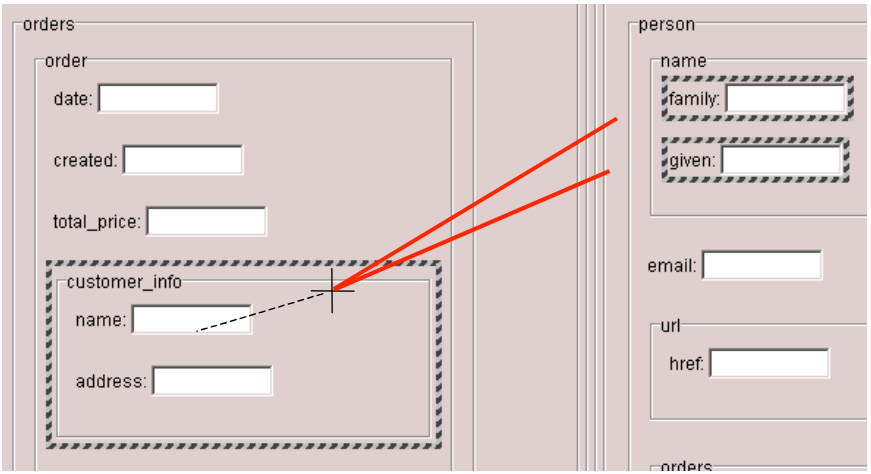


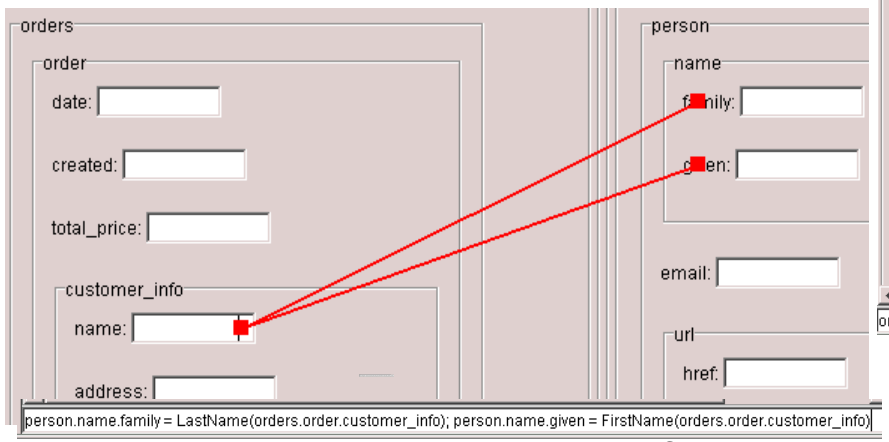
Resize

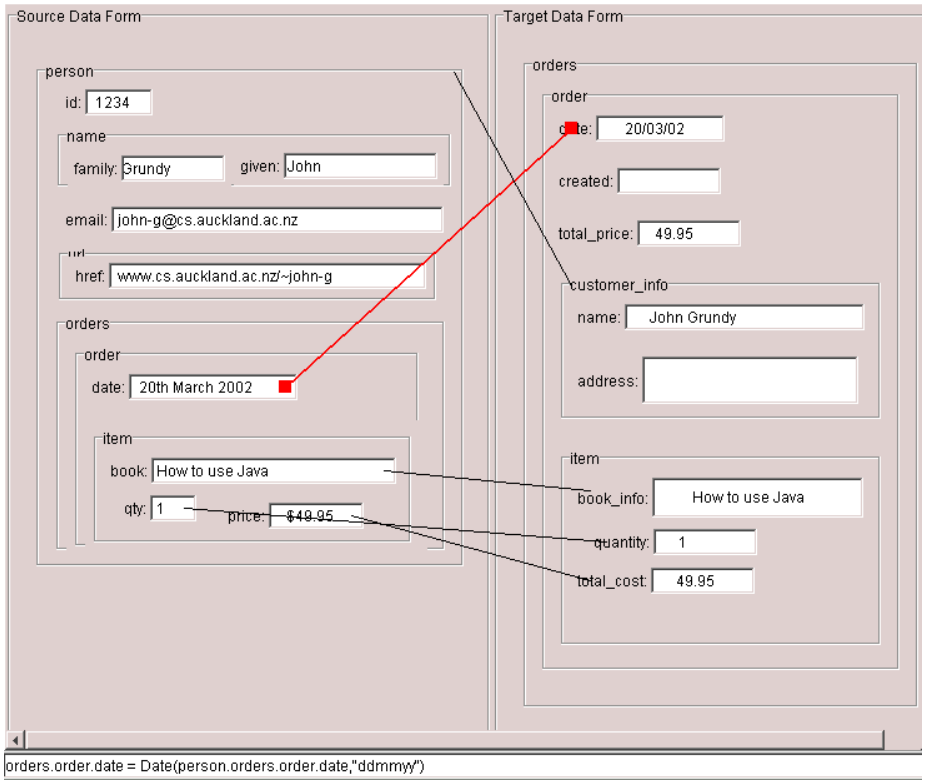
Add sub-structure

Rearrange layout

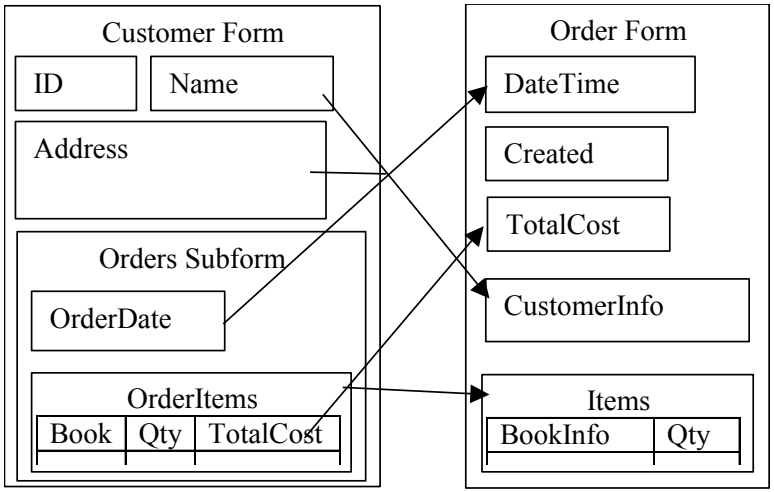
Data mapping







Code generation…



Order:
1

2

3

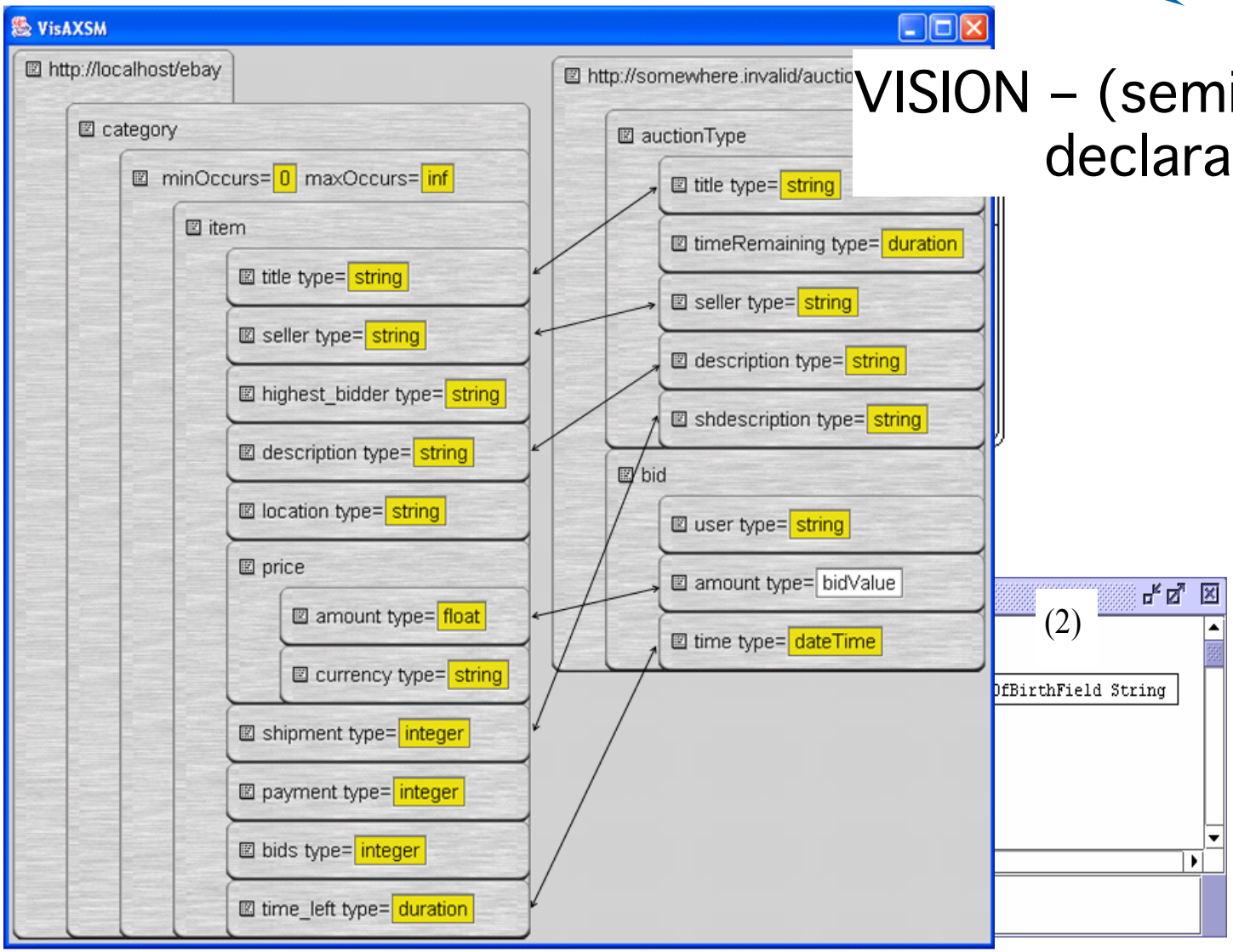
XSLT transformation script generation

4

```
<xsl:template match="/">
 <Order>
  <Number>…</Number>
  <DateTime><xsl:value-of select="/Order[1]/Order/Date"/>
    </DateTime>
  <Created>
    <xsl:value-of select="date:to-string(date:new())"/>
  </Created>
  <TotalCost><xsl:value-of
    select="sum(//OrderItem/TotalCost)"/> </TotalCost>
  <xsl:variable name="customer_id" select=
    "/Order/OrderItem[1]/CustomerSID"/>
  <CustomerInfo>
    <xsl:apply-templates select="//Customer [@id =
    $customer_id]"/>
  </CustomerInfo>
  <Items>
   <xsl:apply-templates select="//OrderItem"/>
  </Items>
 </Order>
</xsl:template>
…
```

Form-based mapper is "concrete", "Semi-declarative" DSVL…

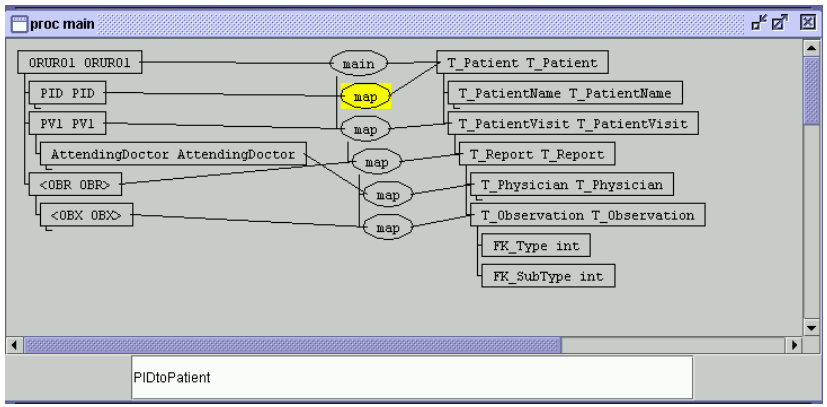
Other DSVL data mappers…

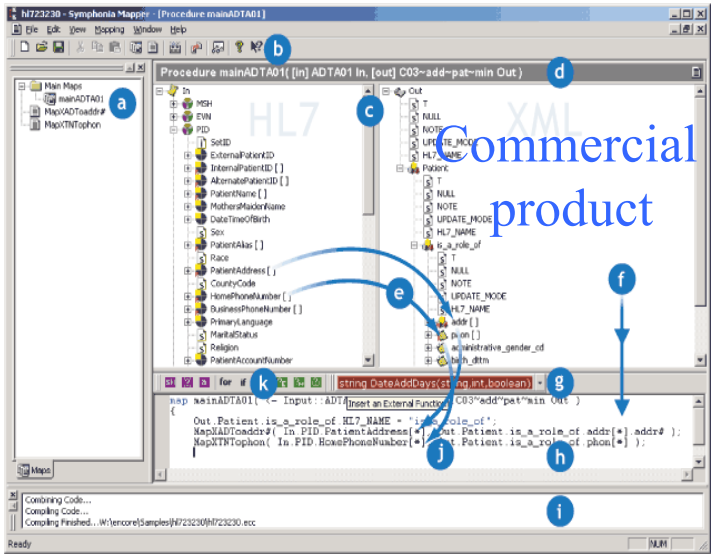


VISION – (semi-)abstract, declarative

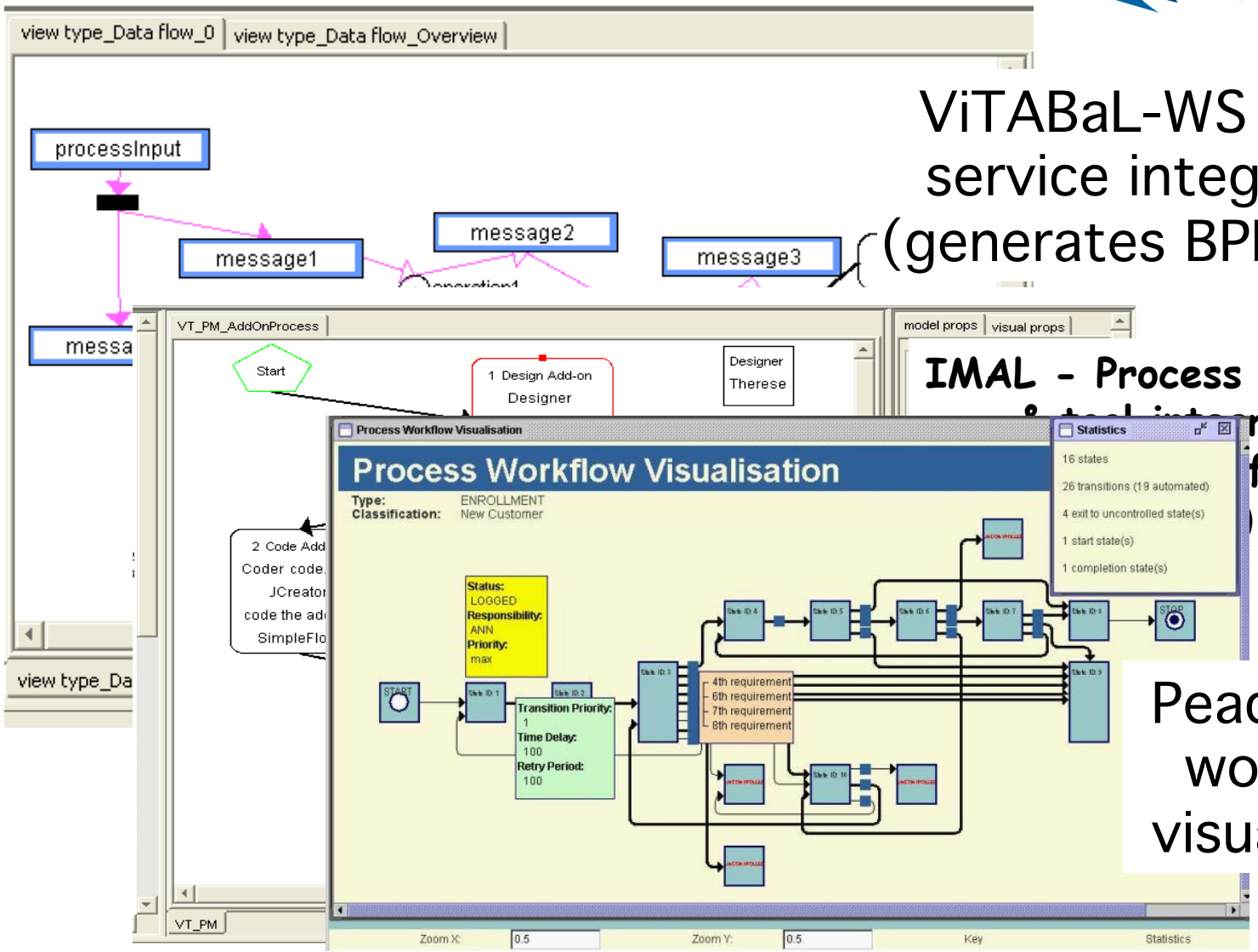
Example: Orion Systems Ltd Symphonia Message Mapper

- Health message mapping
  - Orion developed Rhapsody product from proof of concept systems
  - Large NZ software export earner
  - Follow-up DSVL tools underway





Process Management DSVLs



ViTABaL-WS – web service integration (generates BPEL4WS)

IMAL - Process modelling & tool integration foPath,

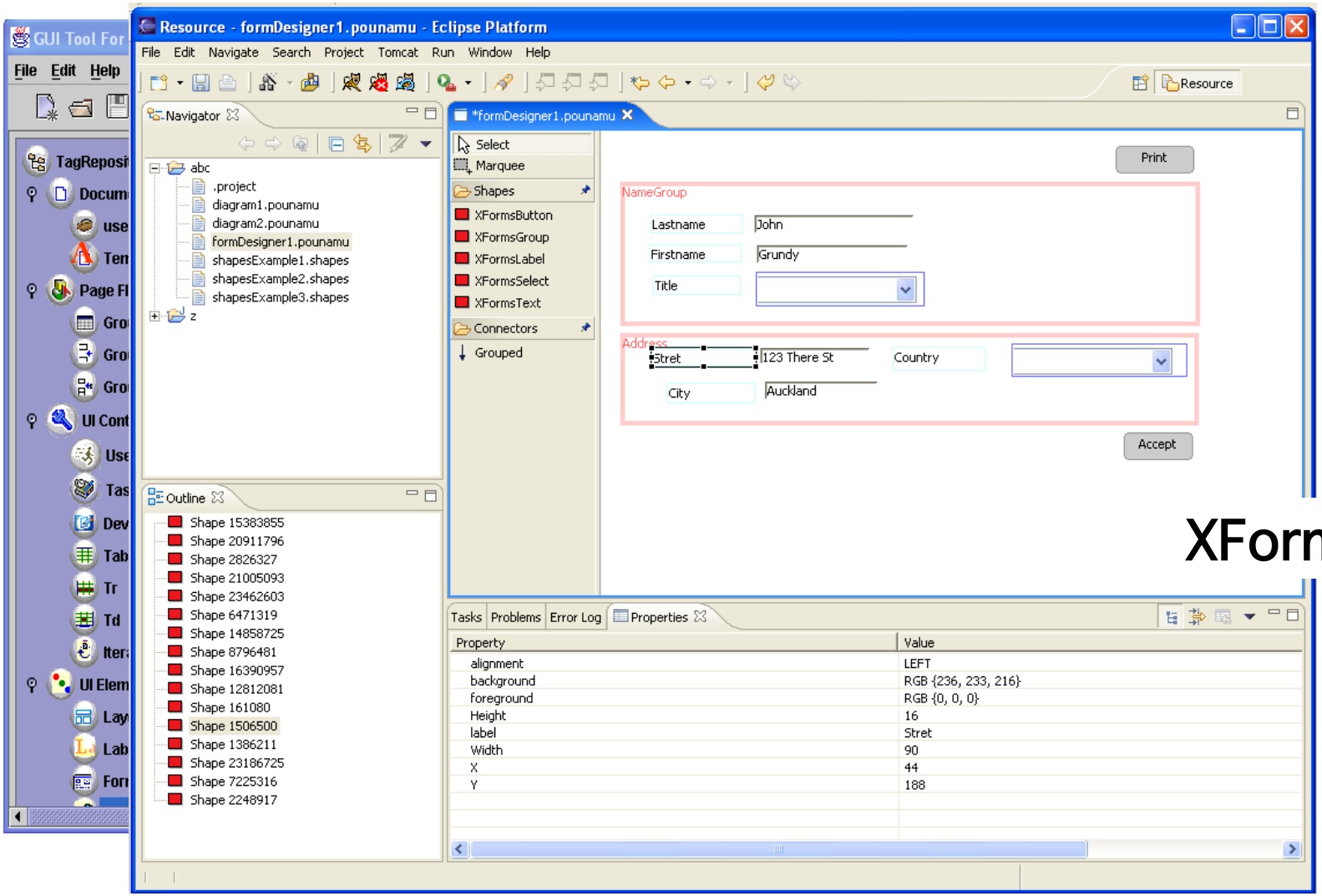
Peace WF – workflow visualisation

Project Management DSVLs

Running in a web browser...

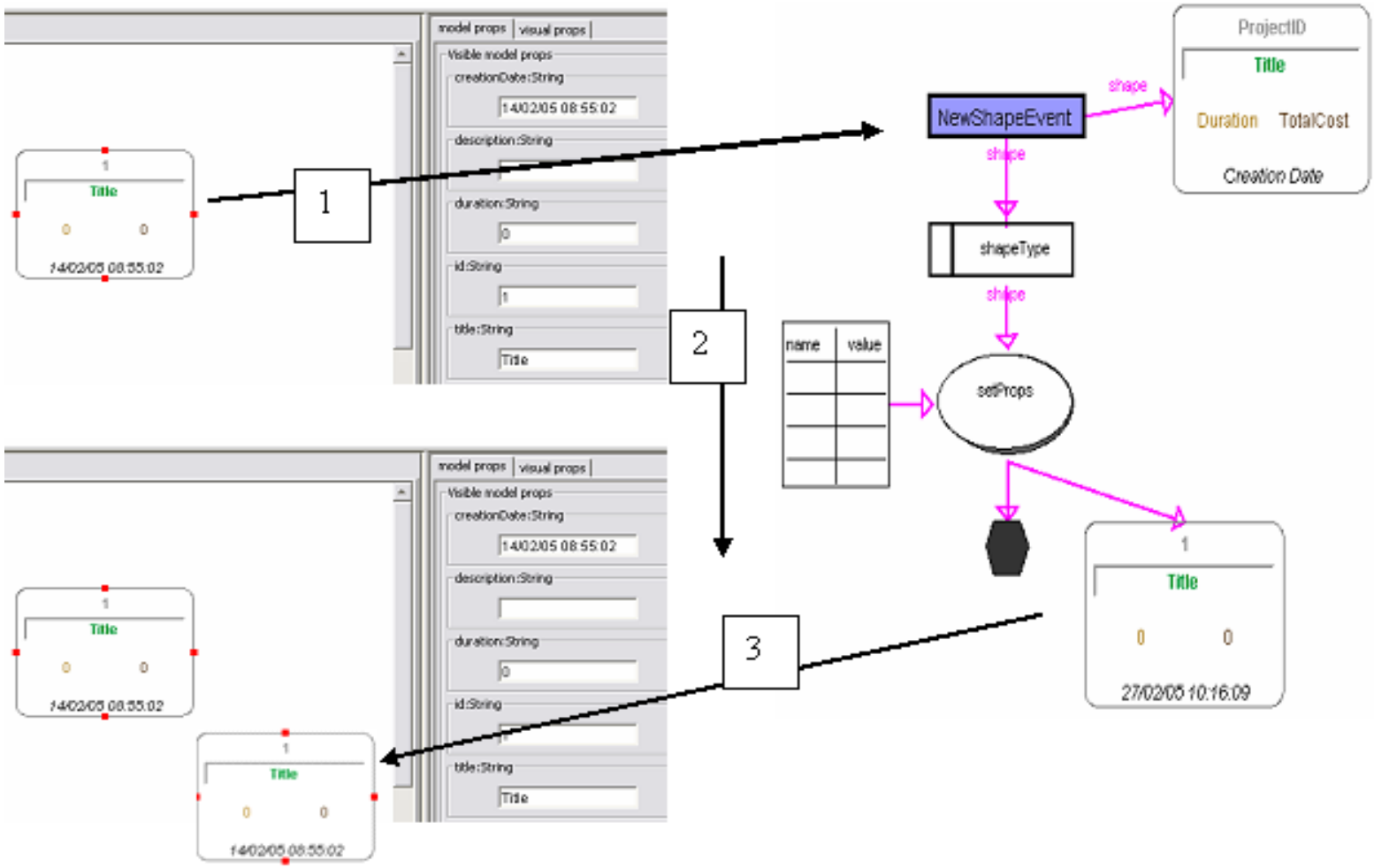
CityVis – PM data ☺

UI Design

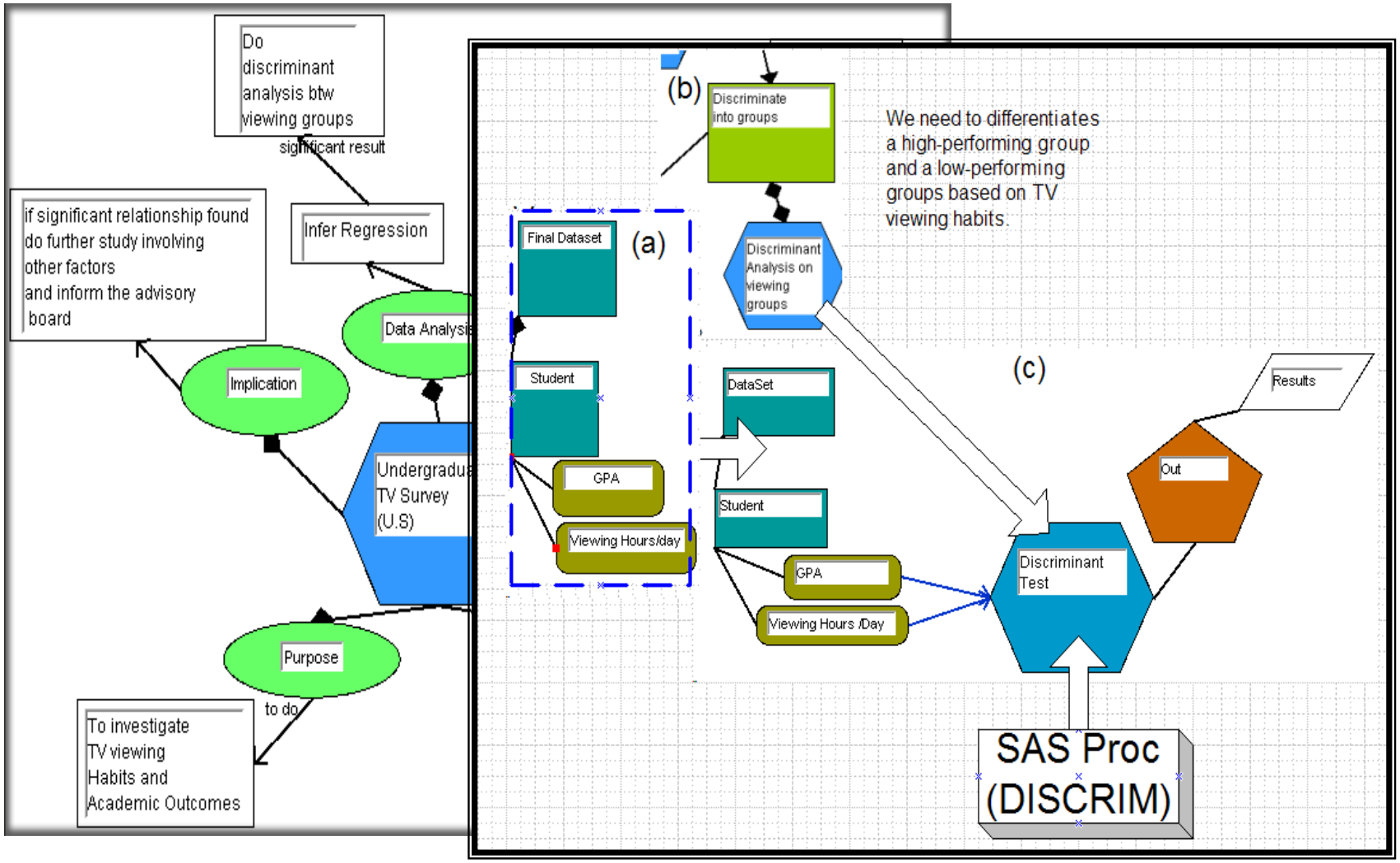


XForms

Visual event handling specification for DSVL tools



And not just for software…

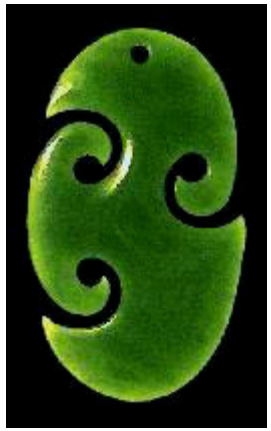


Designing DSVLs

- Little-understood…
- Our approach:
  - Identify key abstractions ("building blocks") in the target domain – leads to Domain-Specific Language
  - Identify candidate visual metaphors for DSL
  - Rapid prototype DSVL tools
  - Evaluate via:
    - Cognitive Dimensions (Greene, Blackwell)
    - Target user groups
    - Comparison to use of "conventional" languages, models, tools

Building DSVL Tools…

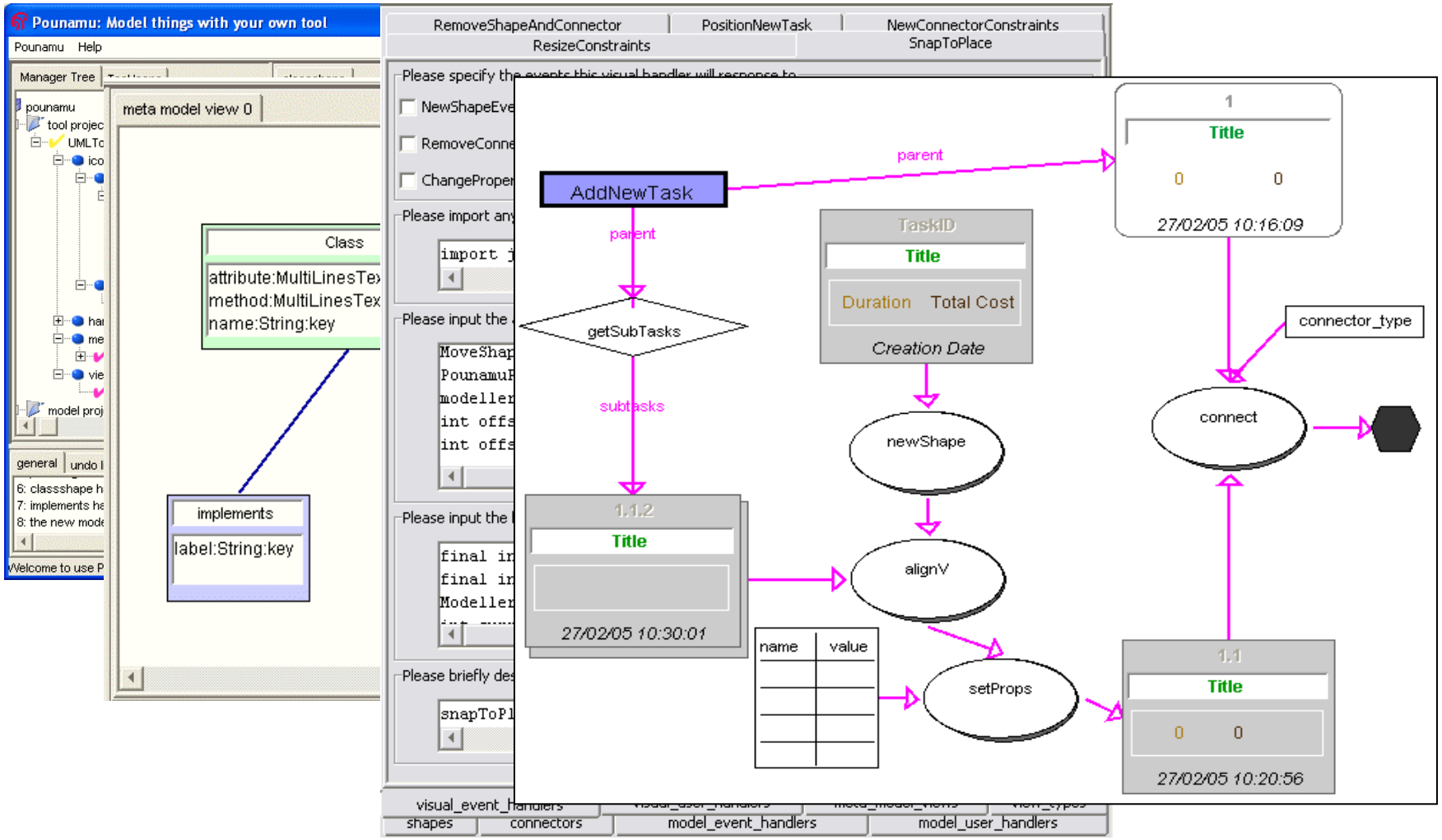
- Its hard to build these things…
  - Visual metaphor
  - Model to represent/build
  - Generate code/configurations/etc from model
  - Integrate with other tools
- Our current approach:
  - Meta-tool – visual models/meta-model
  - Import/export from model (XMI, Java, BPEL, WSDL, etc)
  - Web service/RMI APIs for other tools/plug-ins
  - Web browser, phone, Eclipse, collaboration plug-ins



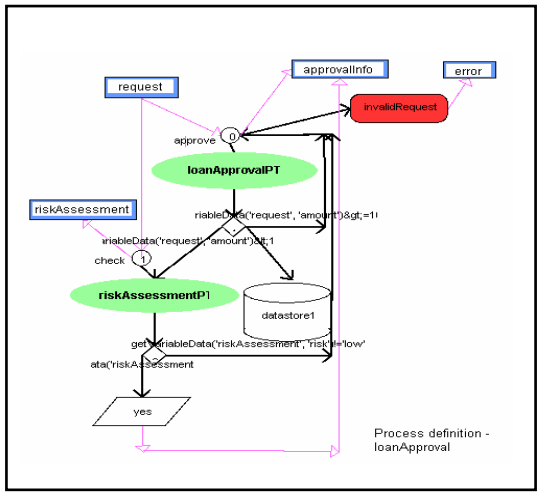
Pounamu

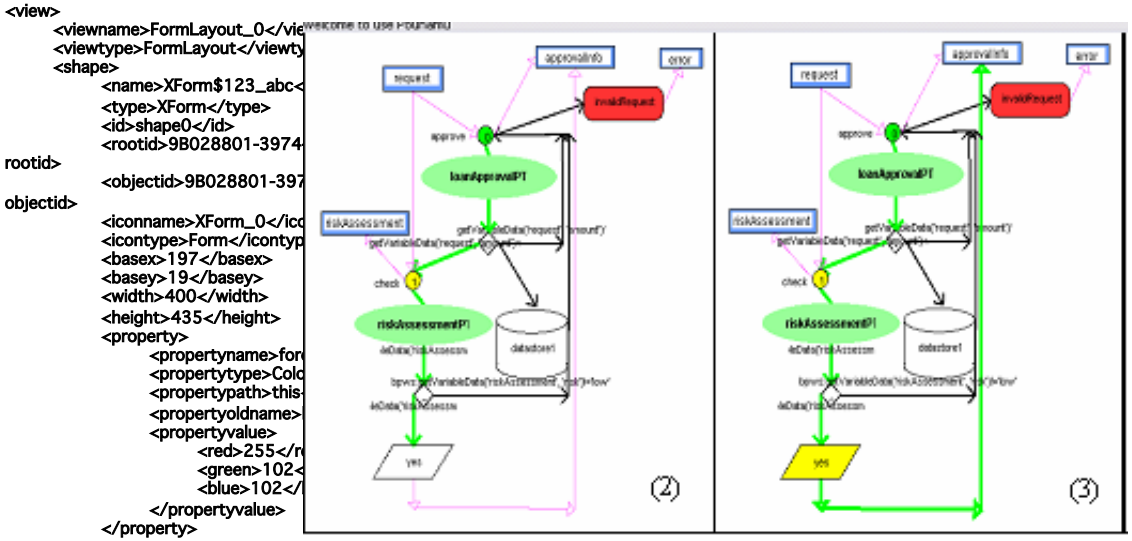
Pounamu Meta-tool Application
Specification Tools
Shape Designer
Meta-model Designer
Event handler Designer
View Designer
Modelling Tools
Modelling Views
Event Handlers
Model Entity instances
Eclipse
Eclipse Plug-in
Tool Specifcations – XML documents
Plug-ins
Web Browsers
Mobile Phones
Web editing plug-in
Web server components
Tool specification projects (XML)
Modelling projects (XML)
Web Services APIs

Meta-tools (themselves DSVLS!)



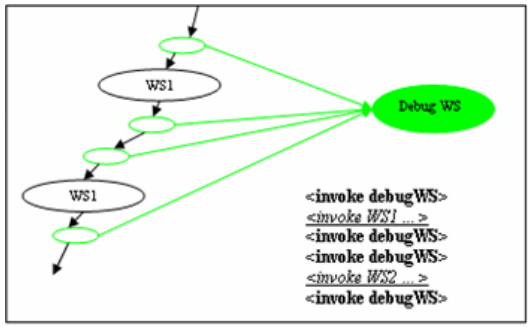
Code (data) generation



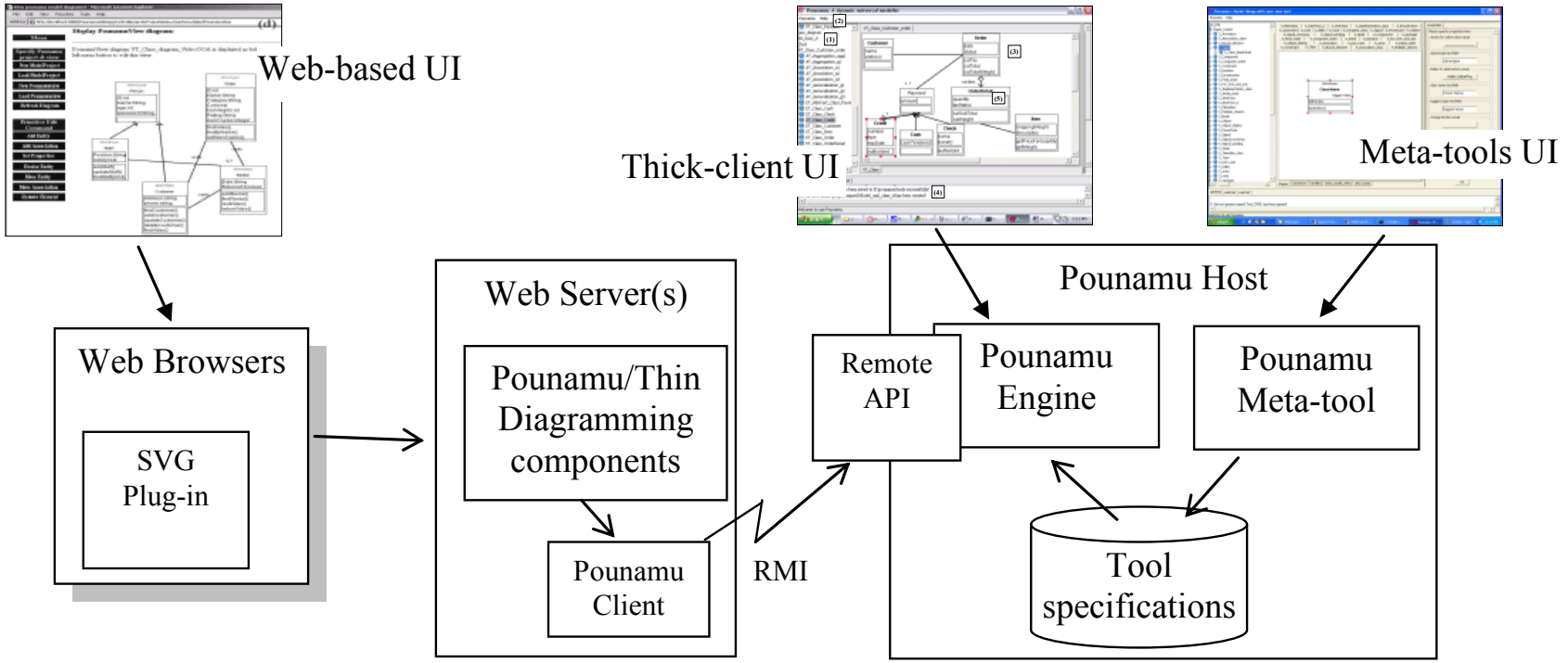


Via XSLT

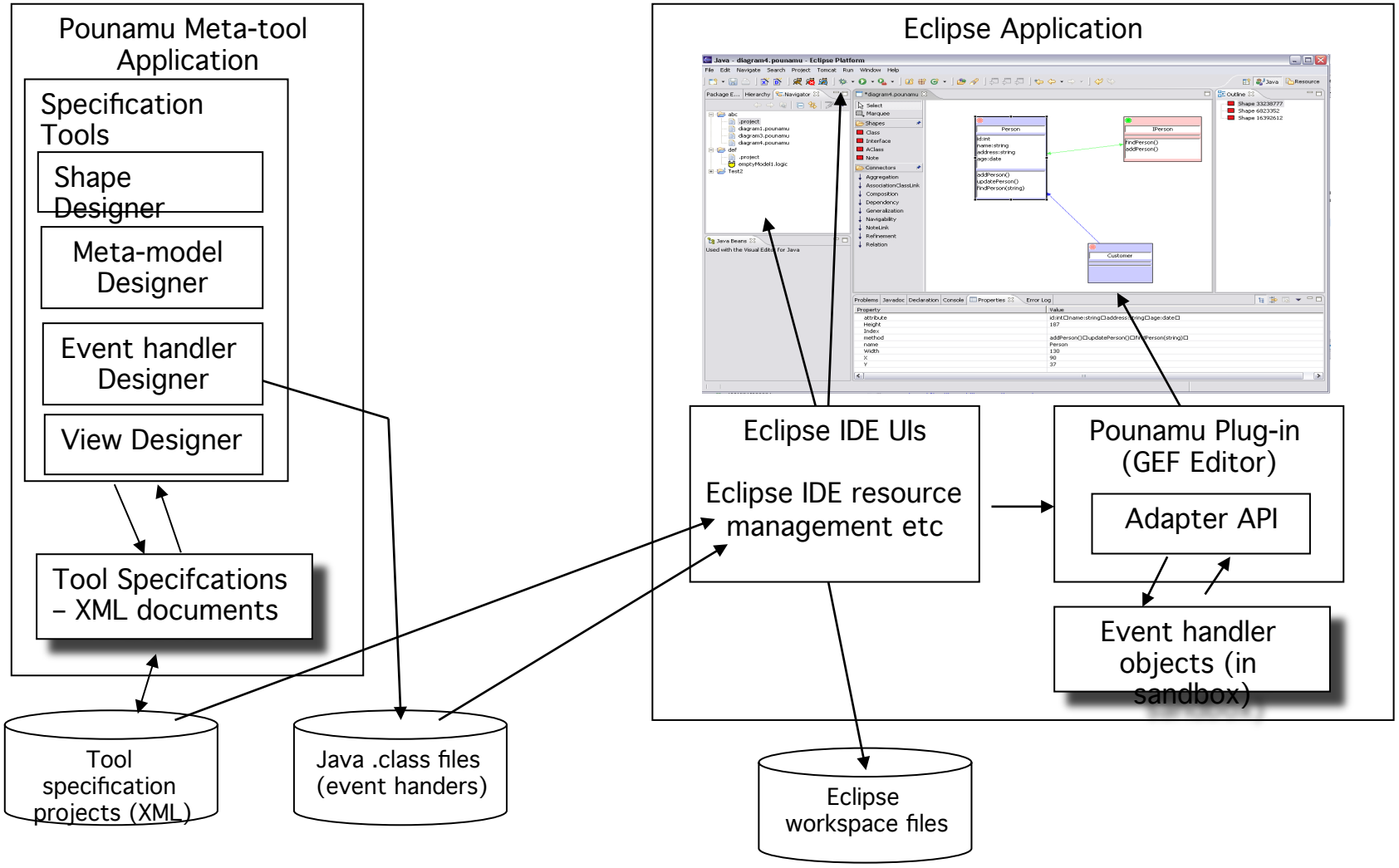
```
<receive name= "receive" partnerLink="customer"
         portType="loanApprovalPT"
         operation="approve"
         variable="request"
         createInstance="yes">
         <!-links-->
</receive>
<invoke name="invokeapprover" partnerLink="approver"
         portType="loanApprovalPT"
         operation="approve"
         inputVariable="request"
         outputVariable="approvalInfo">
         <!-links-->
</invoke>
<invoke name="invokeassessor" partnerLink="assessor"
         portType="riskAssessmentPT"
         operation="check"
         inputVariable="request"
         outputVariable="riskAssessment">
         <!-links-->
</invoke>
```



Web-based diagramming



Eclipse Plug-in



Evaluation

- Used to build range of academic & proof-of-concept industrial DSVL tools
- Proven useful for rapid prototype/evaluation of metaphors, meta-models
- Limited industrial deployment – use as modelling tool
- Still too difficult to express model transformation and code generation (ironically, data mapping… ☺)
- Closer integration with other tools needed (hence Eclipse plug-in; web services API)
- Need richer meta-model & constraint specification
- Need improved event handling specification (hence DSVL for meta-tool itself…)

Conclusions

- Our software models using general-purpose visual notations can get too complex, unwieldy, unsuitable for expressing models in various domains
- Domain-specific languages enable purpose-built model specification; DSVLs provide visual mepahor for these building these models
- DSVL tools support DSL model construction, visualisation and code/data generation/component configuration
- We're still learning how to design appropriate visual metaphors for DSVLs; building DSVL tools is hard
- Pay-off can be high…

References

- Grundy, J.C., Hosking, J.G., Li, N., Li, L., Ali, N.M., Huh, J. Generating Domain-Specific Visual Language Tools from Abstract Visual Specifications, **IEEE Transactions on Software Engineering, vol. 39**, no. 4, April 2013, pp. 487 - 515.
- Grundy, J.C., Hosking, J.G., Cao, S., Zhao, D., Zhu, N., Tempero, E. and Stoeckle, H. Experiences developing architectures for realising thin-client diagram editing tools, Software – Practice and Experience, vol. 37, no.12, Wiley, October 2007, pp. 1245-1283.
- Zhu, N., Grundy, J.C., Hosking, J.G., Liu, N., Cao, S. and Mehra, A. Pounamu: a meta-tool for exploratory domain-specific visual language tool development, Journal of Systems and Software, Elsevier, vol. 80, no. 8, pp 1390-1407.
- Grundy, J.C, Hosking, J.G., Amor, R., Mugridge, W.B., Li, M. Domain-specific visual languages for specifying and generating data mapping system, Journal of Visual Languages and Computing, vol. 15, no. 3-4, June-August 2004, Elsevier, pp 243-263
- Grundy, J.C., Mugridge, W.B., Hosking, J.G. and Kendal, P. Generating EDI Message Translations from Visual Specifications, In Proceedings of the 16th International Conference on Automated Software Engineering, San Diego, 26-29 Nov 2001, IEEE CS Press, pp. 35-42
- Li, Y., Grundy, J.C., Amor, R. and Hosking, J.G. A data mapping specification environment using a concrete business form-based metaphor, In Proceedings of the 2002 International Conference on Human-Centric Computing, IEEE CS Press
- 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.
- Kim, C. Hosking, J.G., Grundy, J.C. A Suite of Visual Languages for Statistical Survey Specification, In Proceedings of the 2005 IEEE Conference on Visual Languages/Human-Centric Computing, Dallas, Texas, 20-24 September 2005, IEEE CS Press.
- Panas, T., Berrigan, R. and Grundy, J.C. A 3D Business Metaphor for Program Visualization, In Proceedings of the 2003 Conference on Information Visualisation, London, 16-18 July 2003, IEEE CS Press.