The University of Auckland | New Zealand
2005 YEAR
PRESENTATION

SE Software Engineering
The University of Auckland

Experiences in generating applications from domain-specific visual languages

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

Outline

- What are domain-specific visual languages?
- Examples of some DSVL tools:
  - Data mapping
  - Process management/tool integration
  - User interface design
  - DSVL tool event specification ☺
- Building DSVL tools – our approach(es)
- Code generation from DSVL tools
- Conclusions

Models in Software Engineering

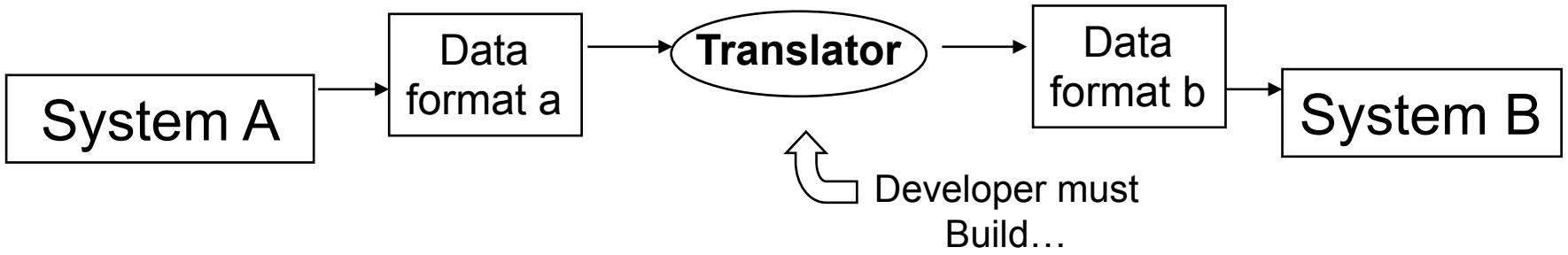
- Much of Engineering is about developing models of engineered products (or rather, models of products to engineer…)
- We've developed models for a whole range of SE "products" and activities:
  - Software processes
  - Requirements
  - Software design
  - Data structures
  - Software architecture
  - 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.

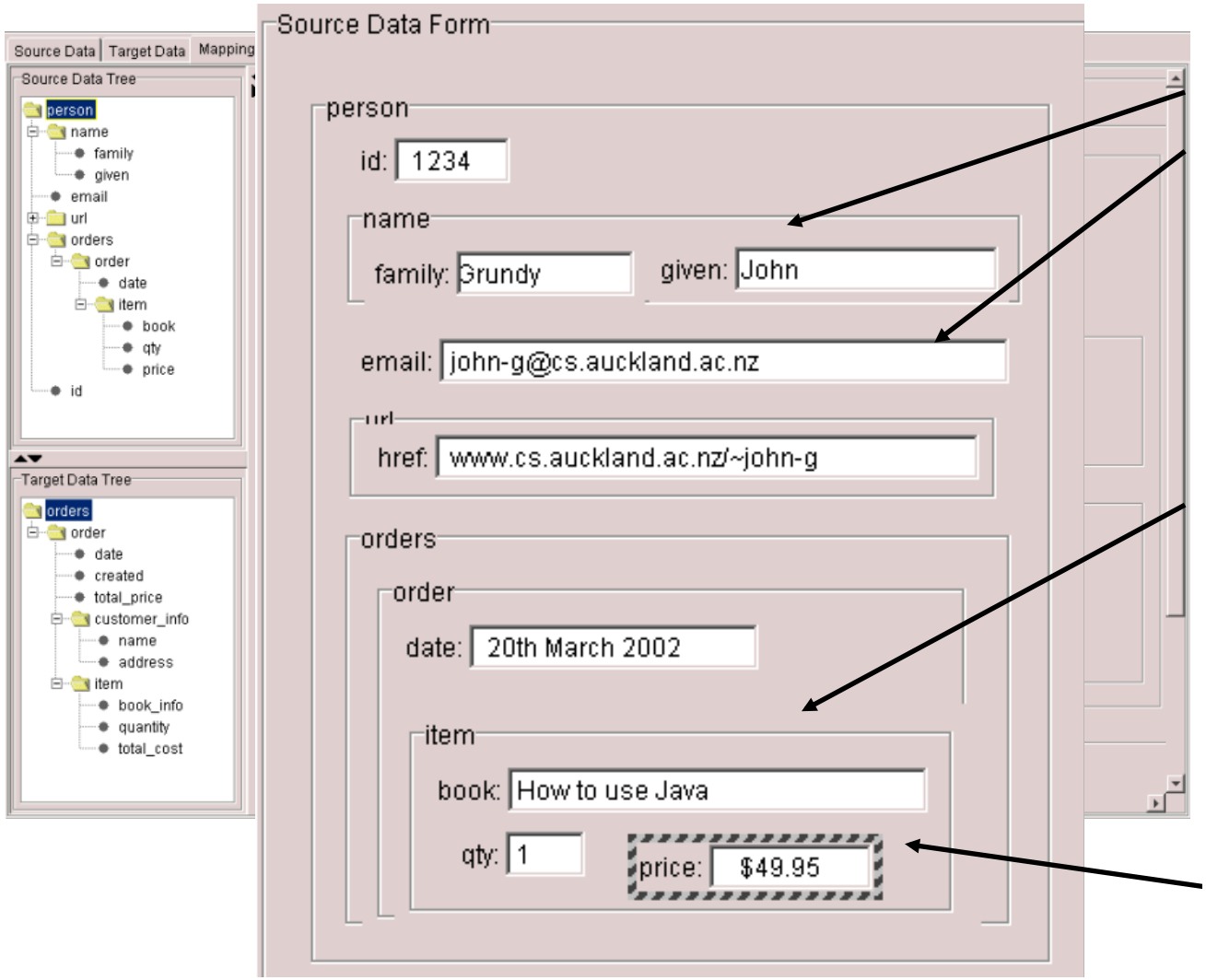
An Example: the Form-based data mapper

- 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



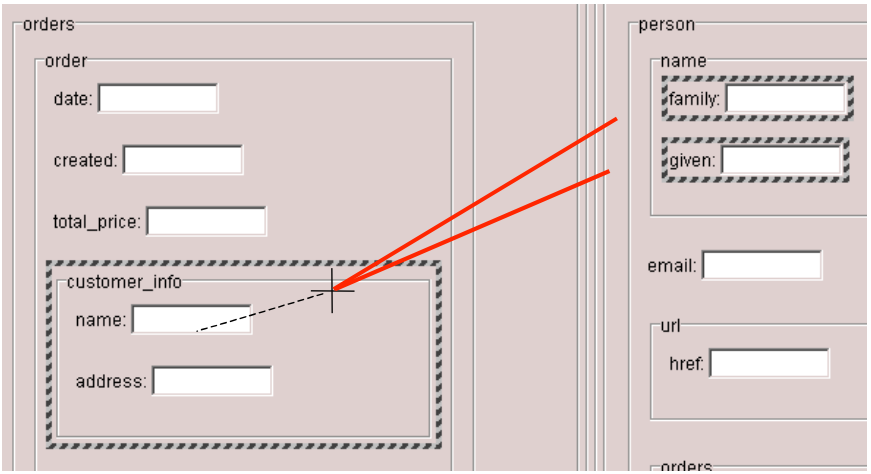
Resize

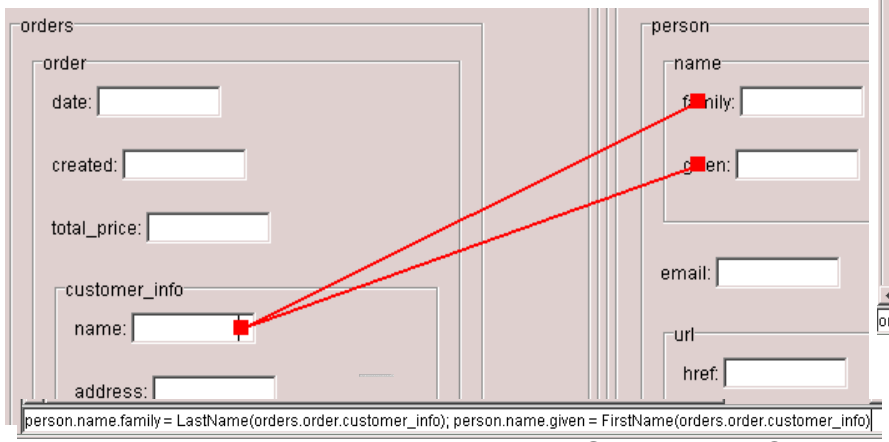
Add sub-structure

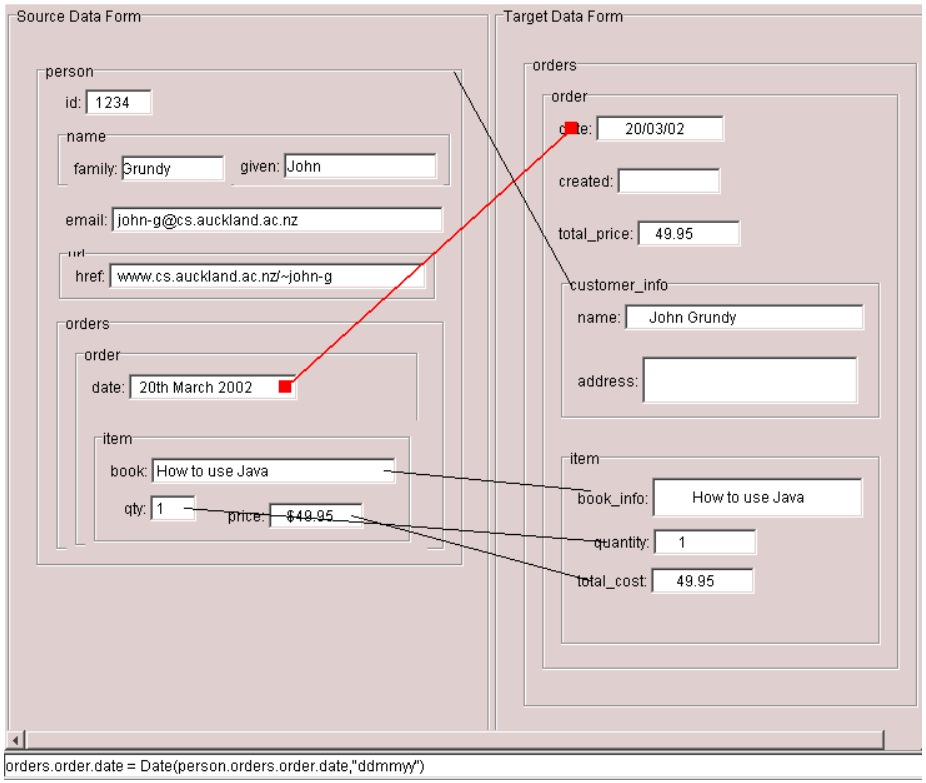
Rearrange layout

2005 YEAR PRESENTATION
The University of Auckland | New Zealand
SE Software Engineering The University of Auckland

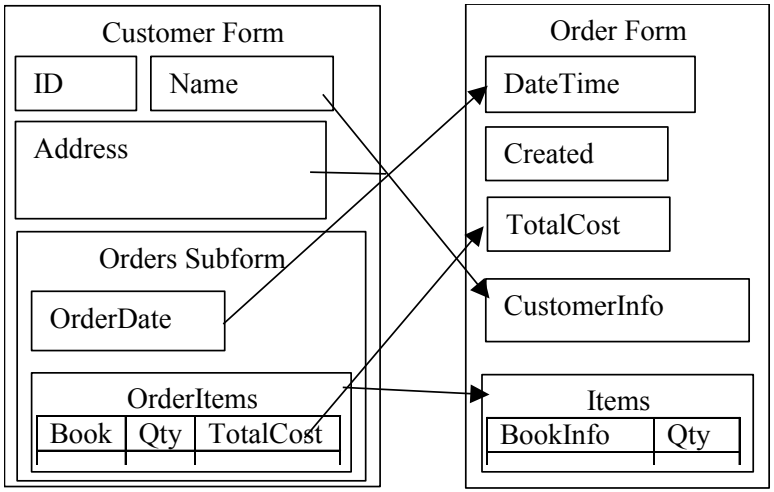
Data mapping







Code generation…



Order:
1

2

3

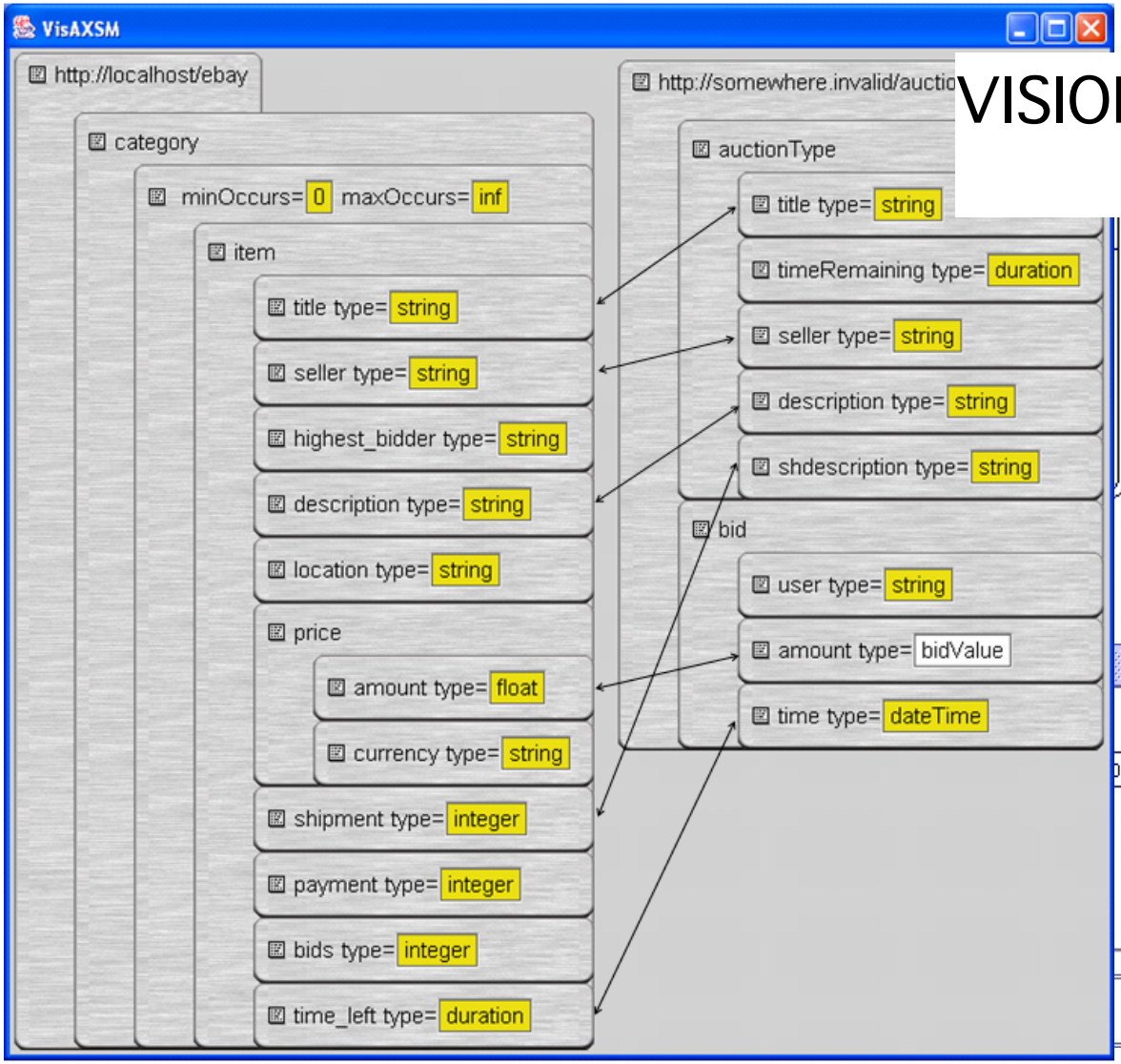
XSLT transformation script generation

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 mappers…

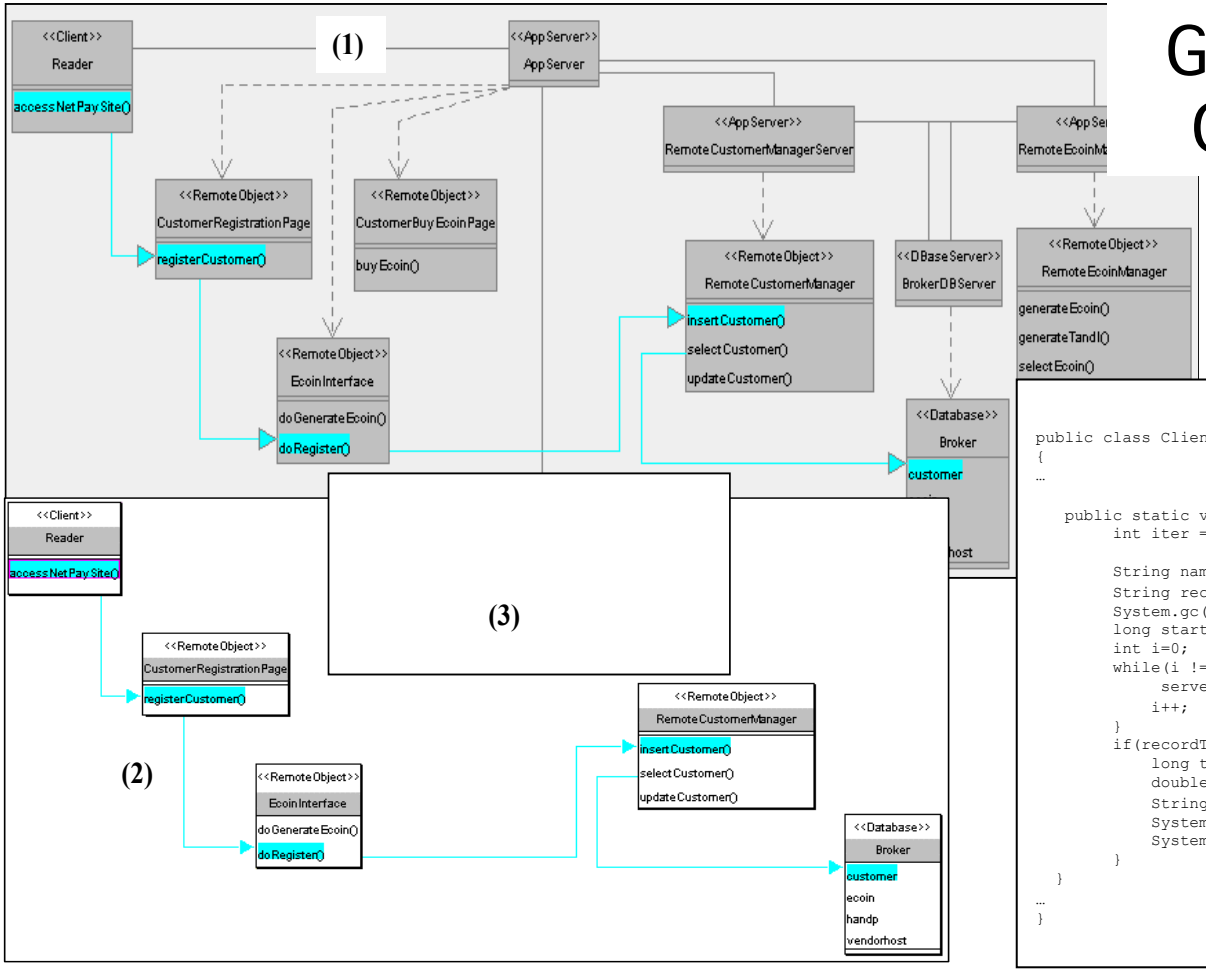


VISION – (semi-)abstract, declarative

Performance test-bed generation

Software Engineering
The University of Auckland

ArgoMTE – Generates Java, C#, JSPs, ASPs



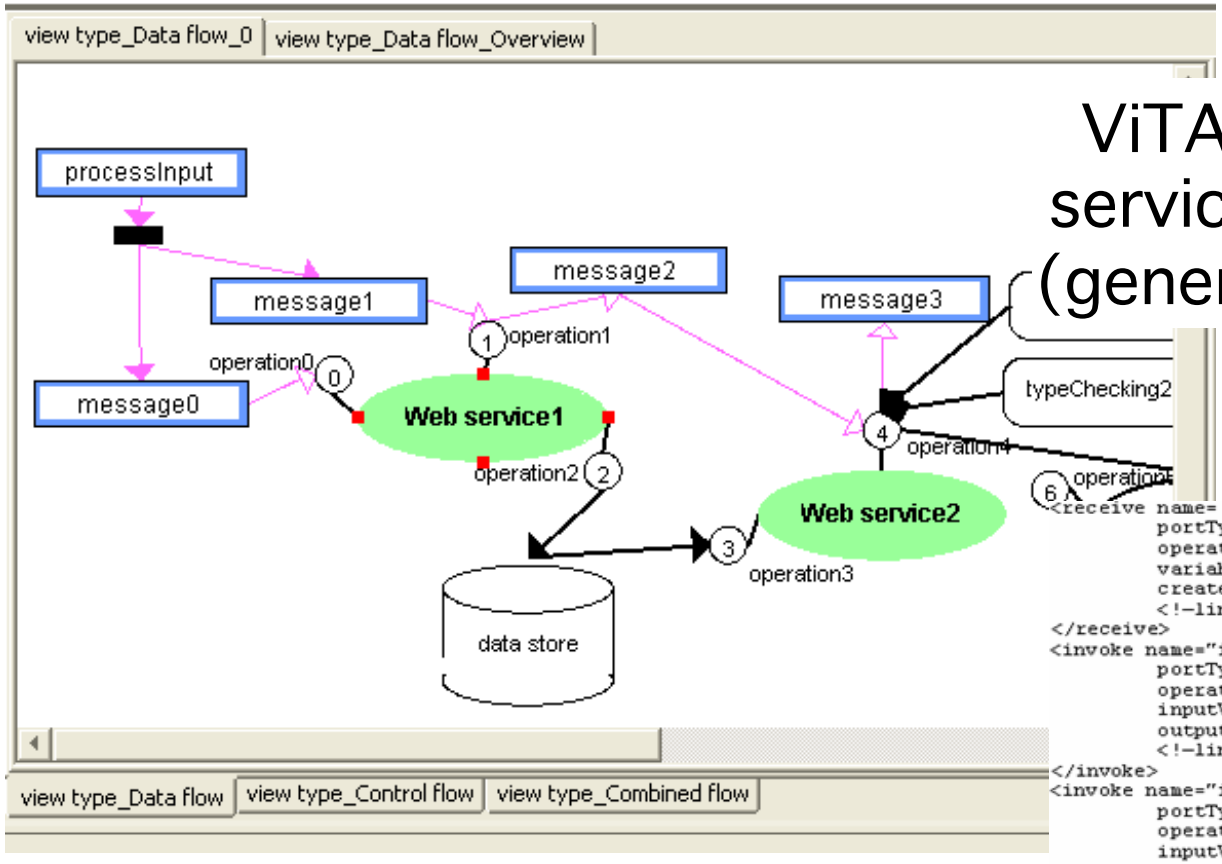
ClientTest.java

```
public class ClientTest
{
…

   public static void findVideo(VideoManager server){
        int iter = 10;

        String name = "findVideo";
        String recordTime = "yes";
        System.gc();
        long start = System.currentTimeMillis();
        int i=0;
        while(i != iter){
             server.findVideo_service ();
            i++;
        }
        if(recordTime.equals("yes")){
            long time = System.currentTimeMillis() - start;
            double elapse = (double)(time) / (double)(Math.max(1,iter));
            String perf = name+"\t"+time+"\t"+iter+"\t"+elapse;
            System.out.println(perf);
            System.err.println(perf);
        }
   }
…
}
```

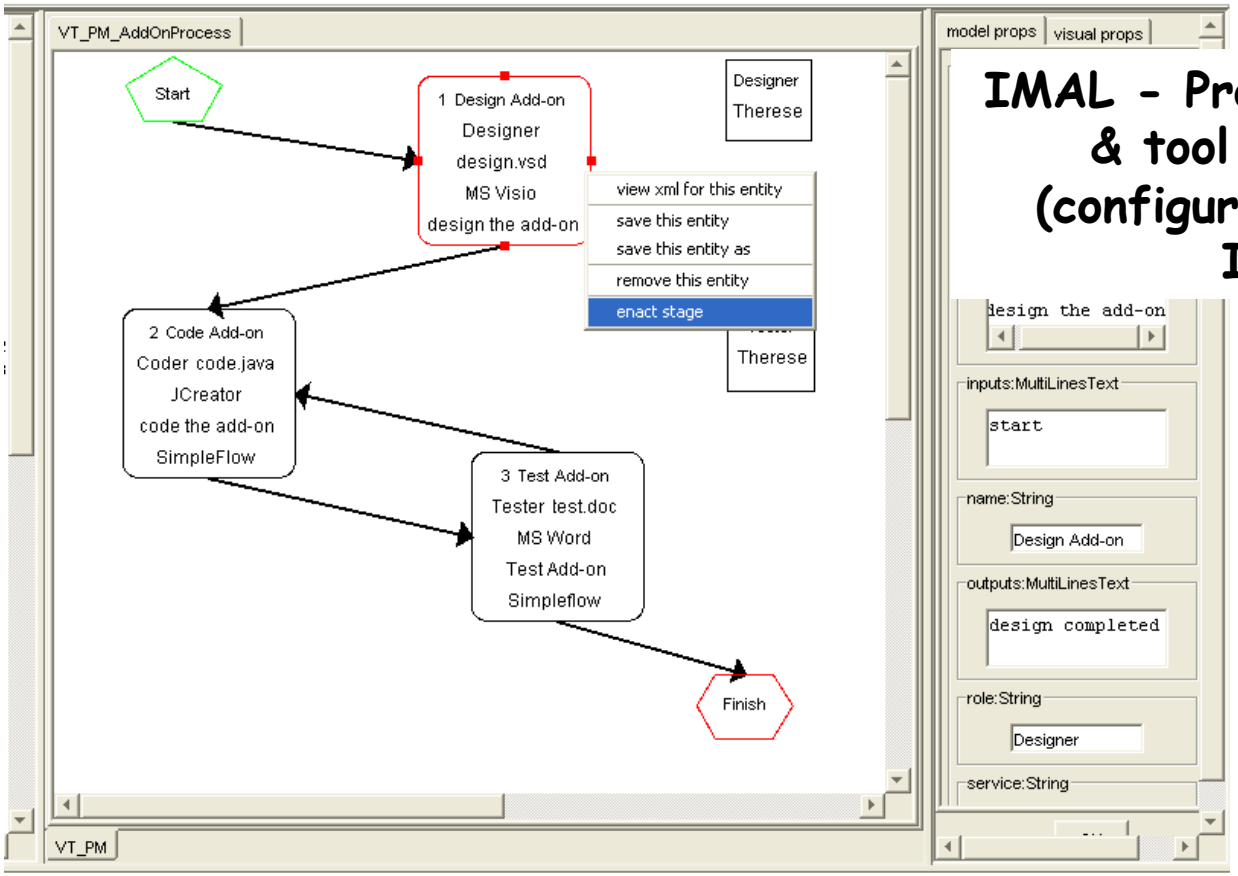
Process Management – web service orchestration



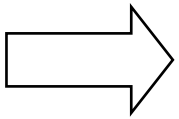
ViTABaL-WS – web service orchestration (generates BPEL4WS)

```
<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>
```

Component configuration (via web services APIs)



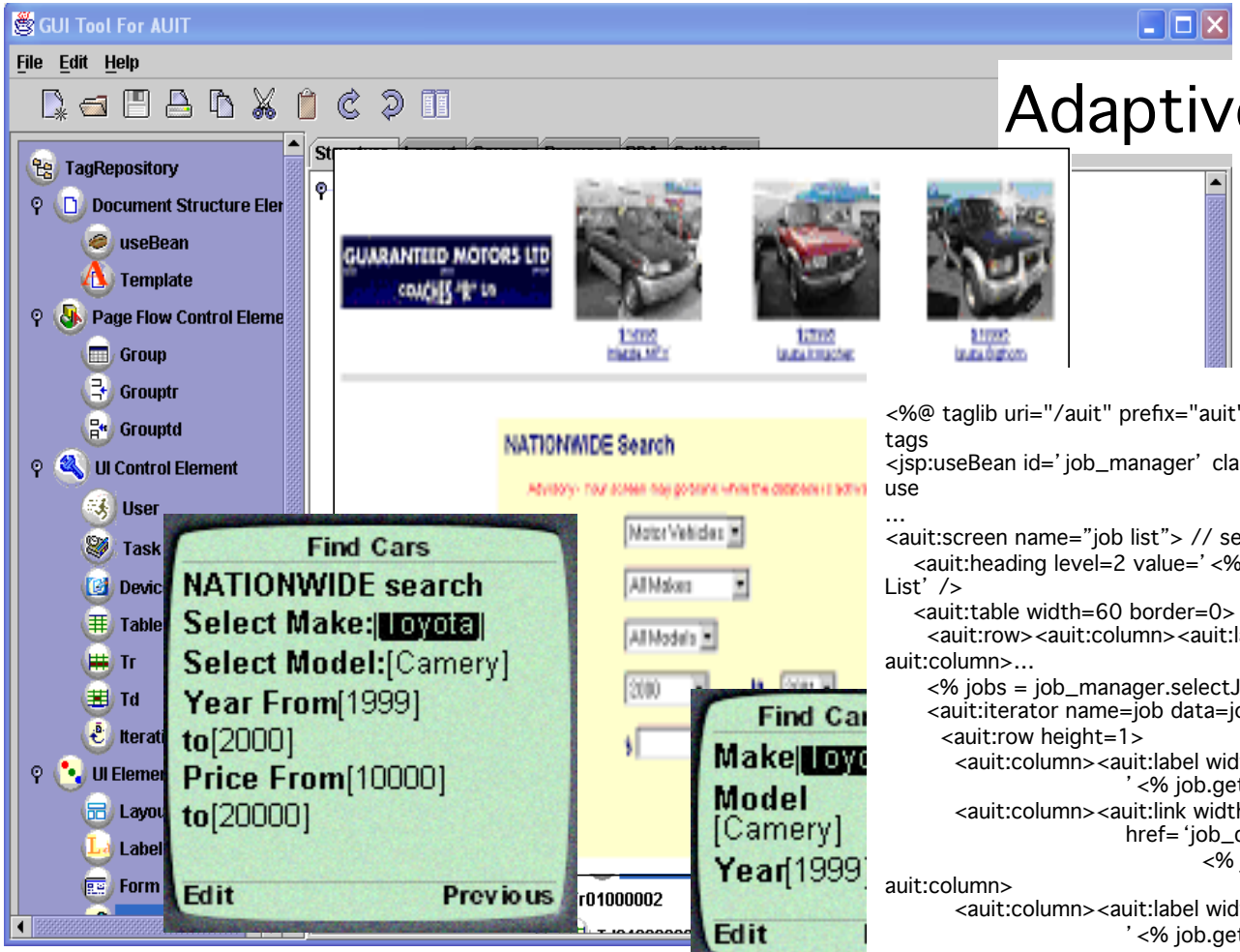
IMAL - Process modelling & tool integration (configures InfoPath, Idiom)



Idiom Rules Engine

MS InfoPath Server

UI Design

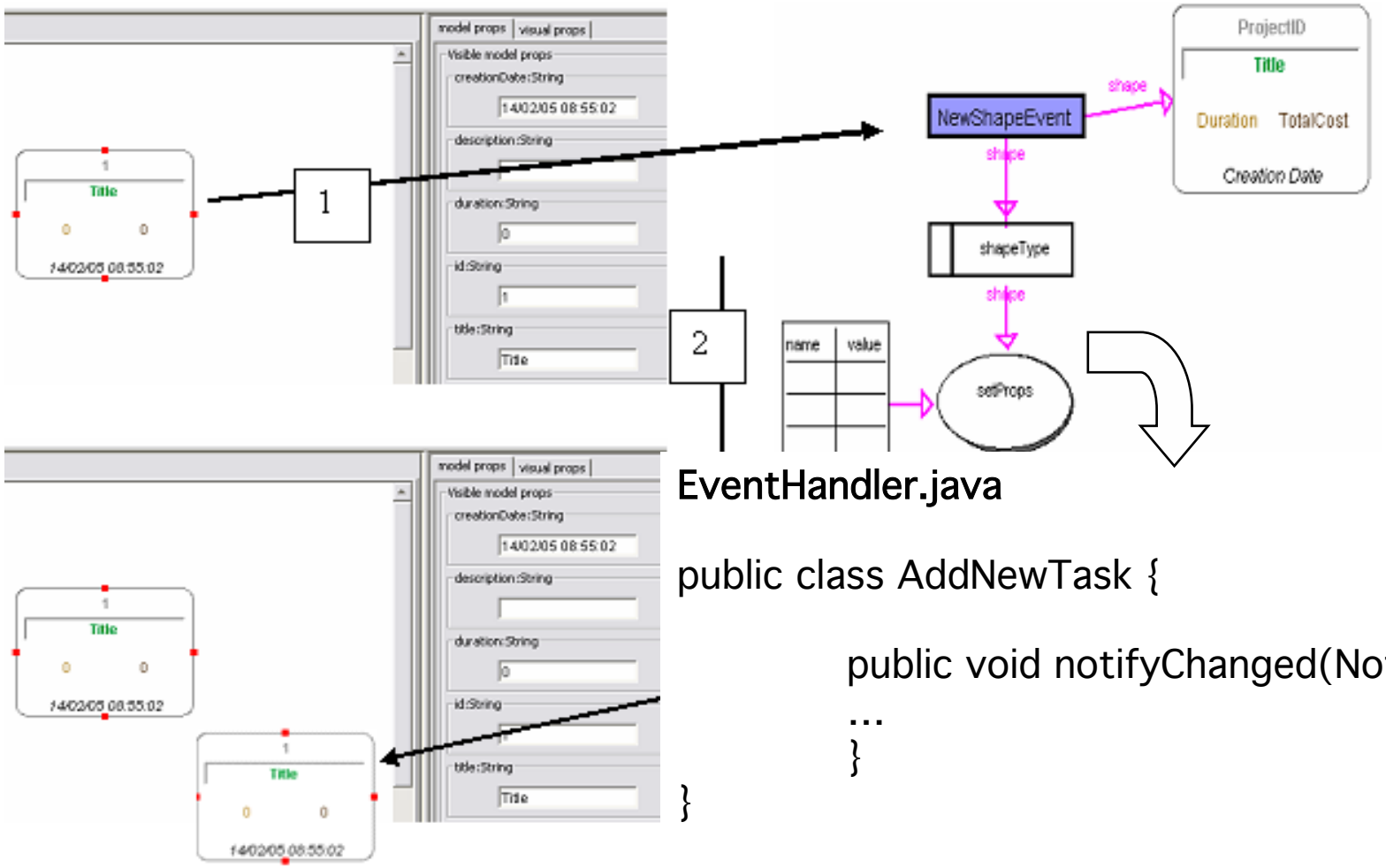


Adaptive UIs

```
<%@ taglib uri="/auit" prefix="auit" %> // page directive to access AUIT
tags
<jsp:useBean id=' job_manager'  class=' jobs.JobManager /> // JavaBeans to
use
...
<auit:screen name="job list"> // sets user/task/device information...
   <auit:heading level=2 value=' <%= AUITUser.getUserName() %>' s Job
List' />
   <auit:table width=60 border=0>
    <auit:row><auit:column><auit:label width=6 value=' Num'  /></
auit:column>...
     <% jobs = job_manager.selectJobs(AUITUser.getUserName()); %>
     <auit:iterator name=job data=jobs %>
      <auit:row height=1>
       <auit:column><auit:label width=6 value=
                          ' <% job.getJobNumber() %>'  /></auit:column>
       <auit:column><auit:link width=20 name=' <% job.getJobNumber() %>'
                          href='job_details.jsp?task=detail&job=
                                <% job.getJobNumber() %>'  /></
auit:column>
       <auit:column><auit:label width=30 value=
                          ' <% job.getInitiator() %>'  /></auit:column>

       ...
      </auit:row>
     </auit:iterator>
    </auit:table>
</auit:screen>
```

Visual event handling specification for DSVL tools - "Kaitiaki"



EventHandler.java

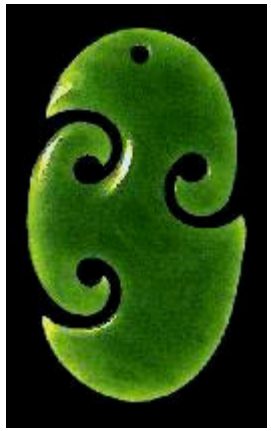
```
public class AddNewTask {

        public void notifyChanged(Notifier e) {
        ...
        }

}
```

Building DSVL Tools…

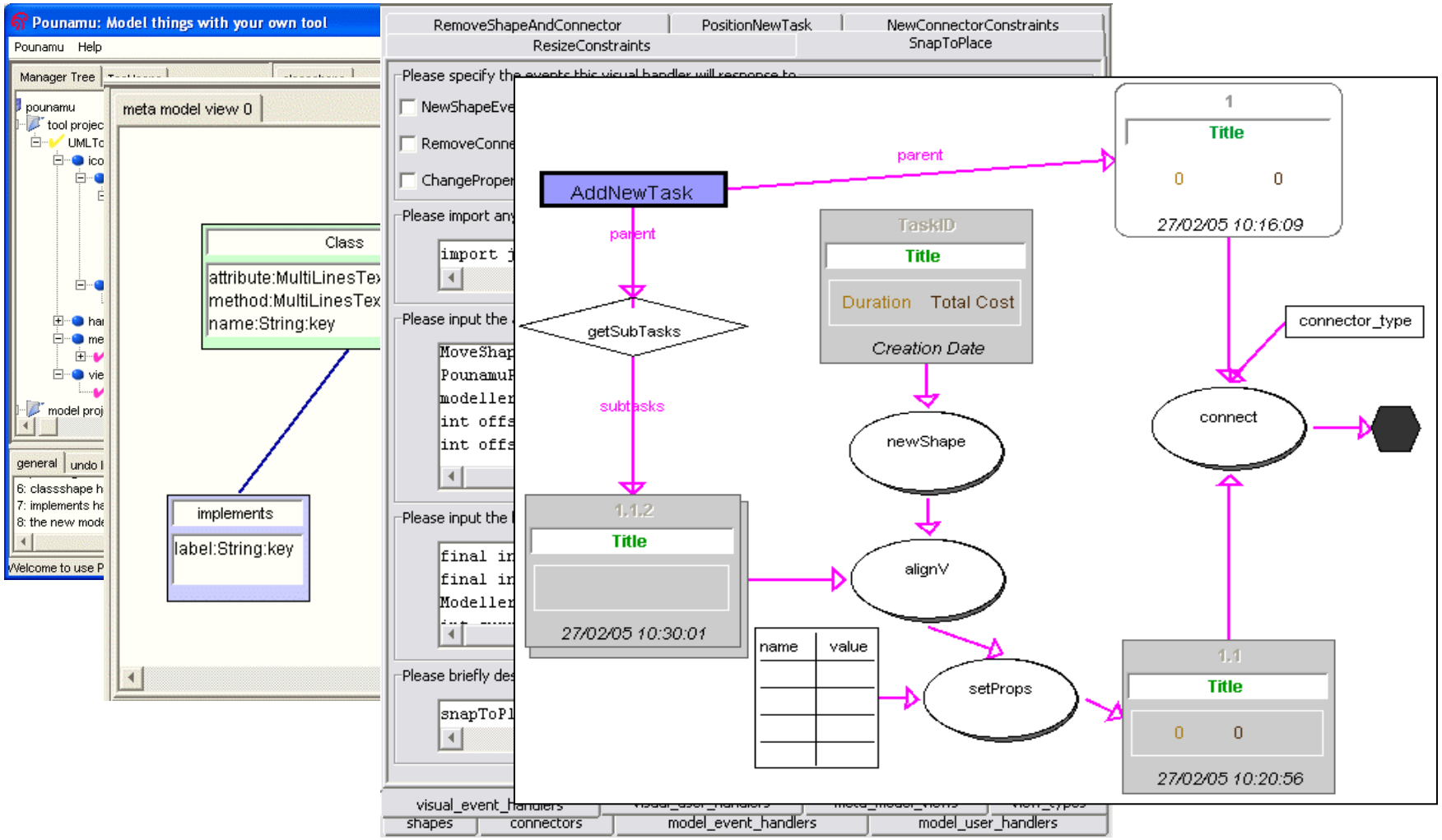
- Its hard to build these things…
  - Visual metaphor [another talk for another day…]
  - Models to represent/build; editing tool for models
  - Generate code/configurations/etc from model
  - Integrate with other tools
- Our current approach:
  - Meta-tool – visual models/meta-model
  - Import/export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



Pounamu

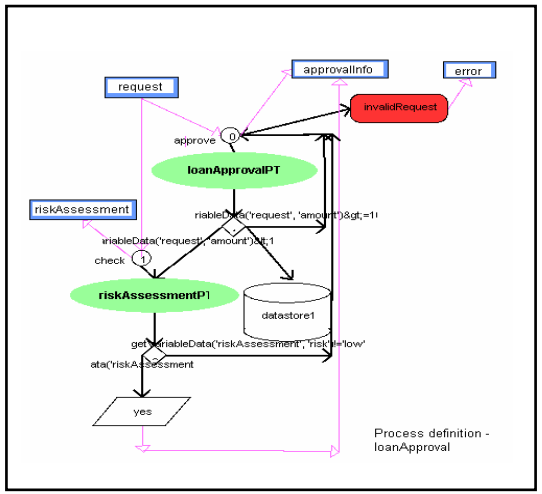
Pounamu Meta-tool Application
Specification Tools
Shape Designer
Meta-model Designer
Event handler Designer
View Designer
Modelling Tools
Modelling Views
Event Handlers
Model Entity instances
Eclipse
Eclipse Plug-in
Tool Specifcations – XML documents
Plug-ins
Code templates
XSLT Engine
Code generated
3rd party components
Web service API
Tool specification projects (XML)
Modelling projects (XML)
Web Services APIs

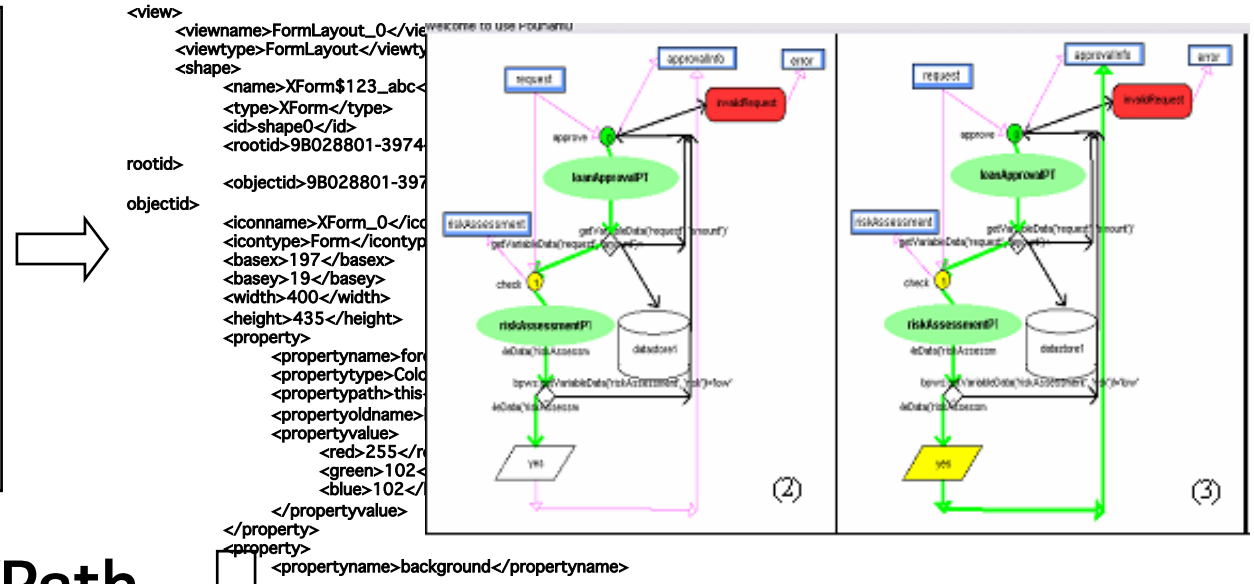
Meta-tools (themselves DSVLS!)



SE Software Engineering
The University of Auckland
2005 YEAR PRESENTATION
The University of Auckland | New Zealand

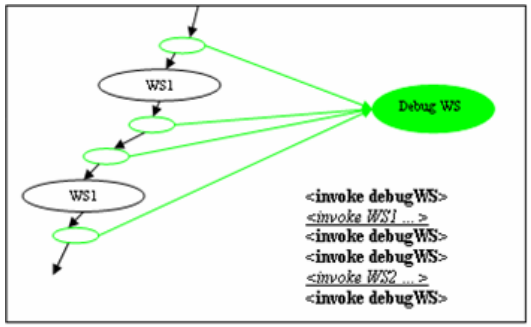
Code (data) generation





Via XSLT/XPath

```
<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>
```



2005 YEAR PRESENTATION
The University of Auckland | New Zealand

Code Generation Approaches & Experiences

- Experiences with translating XML (DS model) into:
  - XSLT (Form-based Mapper); Rimu (RVM); XSLT +Express-G (VML); XSLT or Java (Vision) – all via Java
  - BPEL (via XSLT) – ViTABaL-WS
  - Java/C#/JSPs/Ant scripts/IDLs/… (all via XSLT & Ant build scripts) – ArgoMTE
  - Java – Pounamu ECA event handlers (via Java) – Katiaki
  - Adaptive User Interface Technology – AUIT, via Java
  - Configuration of web service components – IMAL, via Java and SOAP messages

Code Generation Experiences

- Still too difficult to express model transformation and code generation (ironically, a data mapping problem… ☺)
- XSLT is nice, abstract approach but proving limited for complex transformation problems
- Java code gen. effective but too hard to maintain – whole reason for the various data mapper tools…!
- Recently built an Eclipse plug-in which also allows use of the purpose-designed Java Emitter Templates (JET) code generator now – basically JSPs
- Looking at ways to generate JET specifications for DSVL tools…

Current Work

- Code generation challenging:
  - Have DSL model for which to generate code
  - Have target code/model/configuration
- Need better meta-tools to describe this code-gen
- Our approach: YAMT (yet another mapping tool ☺) - DSVL specifically for code gen/MDA
  - Source/target models (BOTH DSVLs…)
  - Mappings between
  - Generation of code generator (meta-generator ☺)
  - Doing with the VISION tool (see ASE 2004)

Conclusions

- Models using general-purpose visual notations can get too complex, unwieldy, unsuitable for expressing things 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
- While editing tools is usually thought of as hard stuff, code gen. is v. hard too – need DSVLs for this!

References

- 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.
- 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.
- 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.
- 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,
- Liu, N., Hosking, J.G. and Grundy, J.C. A Visual Language and Environment for Specifying Design Tool Event Handling, In Proceedings of the 2005 IEEE Conference on Visual Languages/Human-Centric Computing, Dallas, Texas, 20-24 September 2005, IEEE CS Press.
- Liu, N., Grundy, J.C. and Hosking, J.G., A visual language and environment for composing web services, In Proceedings of the 2005 ACM/IEEE International Conference on Automated Software Engineering, Long Beach, California, Nov 7-11 2005, IEEE Press, pp. 321-32.
- Grundy, J.C., Hosking, J.G., Li, L. And Liu, N. Performance engineering of service compositions, ICSE 2006 Workshop on Service-oriented Software Engineering, Shanghai, May 2006.
- Gundy, 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.
- 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.
- Helland, T., Grundy, J.C. and Hosking, J.G. A Service-Oriented Architecture for Software Process Technology, In Proceedings of the 2006 Australian Conference on Software Engineering, Sydney, April 2006, IEEE CS Press.
- Zhao, D., Grundy, J.C. and Hosking, J.G. Generating mobile device user interfaces for diagram-based modelling tools, In Proceedings of the 2006 Australasian User Interface Conference, Hobart, Australia, January 2006.