SE Software Engineering
The University of Auckland

Generating domain-specific Eclipse graphical editors from high-level meta-tool specifications

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

Outline

- Domain-specific visual languages
- Meta-tool specifications in Pounamu
- Eclipse IDE
- Marama – a set of Eclipse plug-ins for DSVLs
- Example usage
- Design and Implementation
- Evaluation
- Current & Future work
- Summary

What are Domain-specific visual languages (DSVLs)?

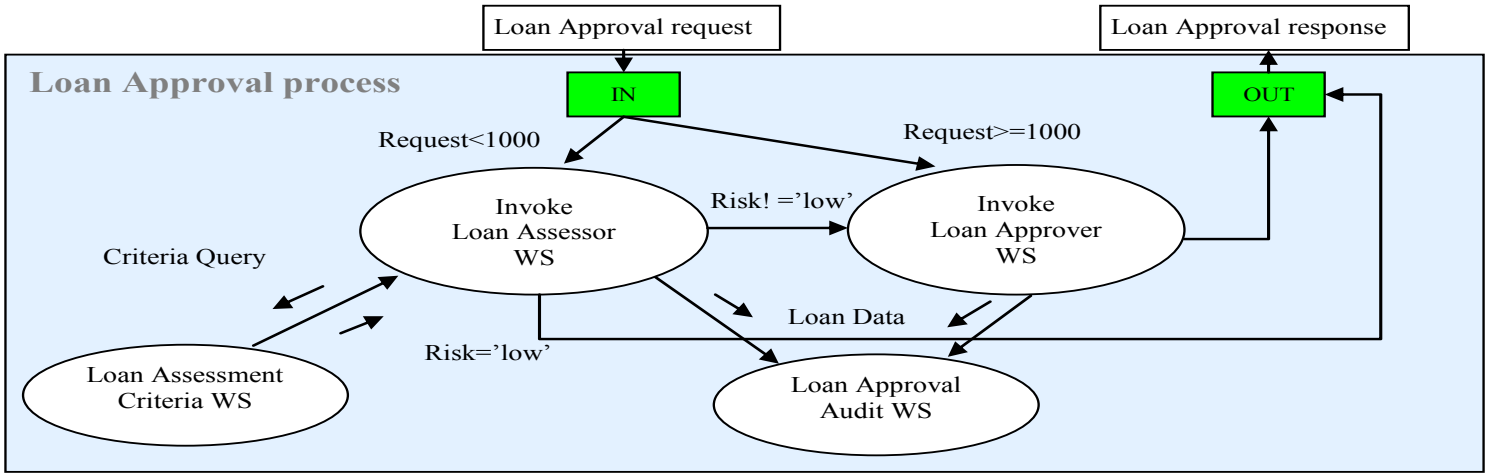
- Much of Engineering is about developing models of engineered products (or rather, models of products to engineer…)
- We've developed models for a whole range of SE "products" and activities:
  - Software processes
  - Requirements
  - Software design
  - Data structures
  - Software architecture
 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

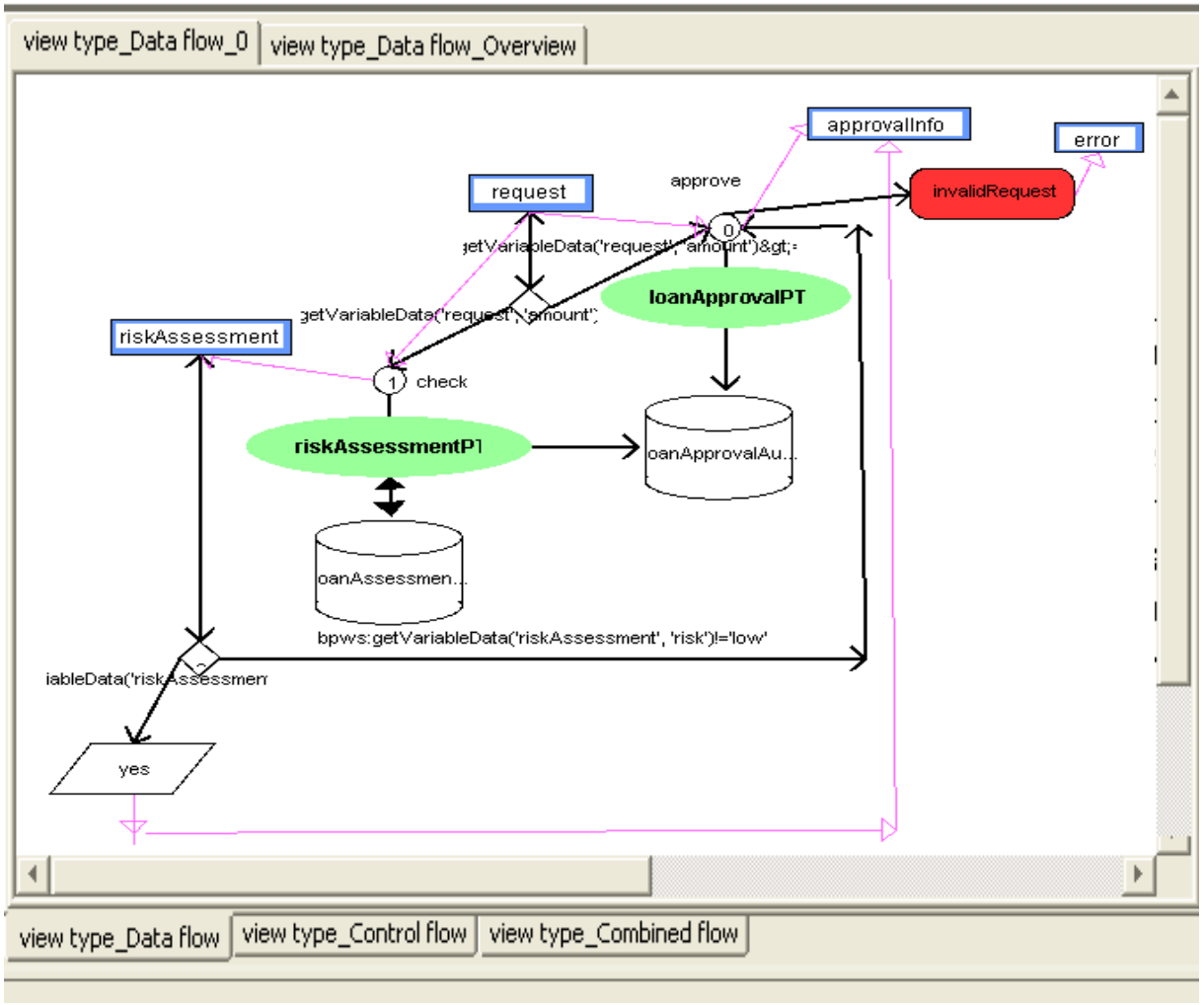
- Our models often get too complex, too unwieldly, hard to understand/maintain using only "abstract" or "general-purpose" model representations
- Example: any non-trivial Model-Driven Architecture application…
- Domain-specific languages (DSLs) – models that focus on expressing problems in a PART of 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.

Example: ViTABaL-WS Web Service composition tool

- Idea of "web services" – software components can dynamically discover, integrate, communicate with
- Want to support users specifying WS compositions
- Usual approach: code "Business Process Execution Language for Web Service (BPEL4WS)" or simplar textual specification
- Really want visual composition metaphor/tool…

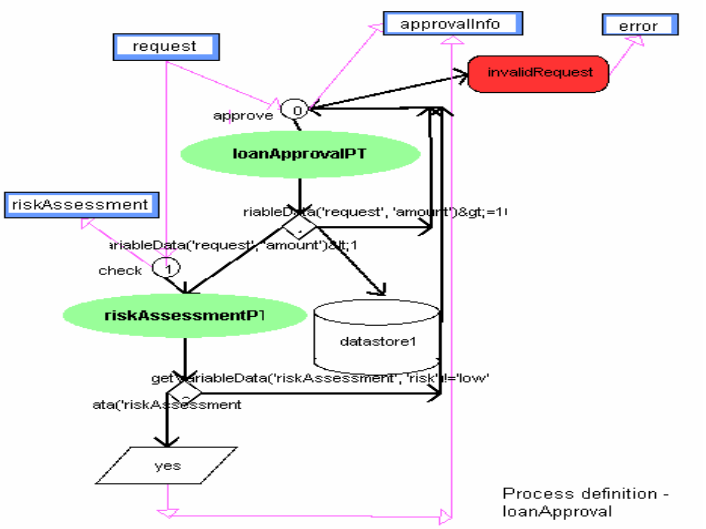


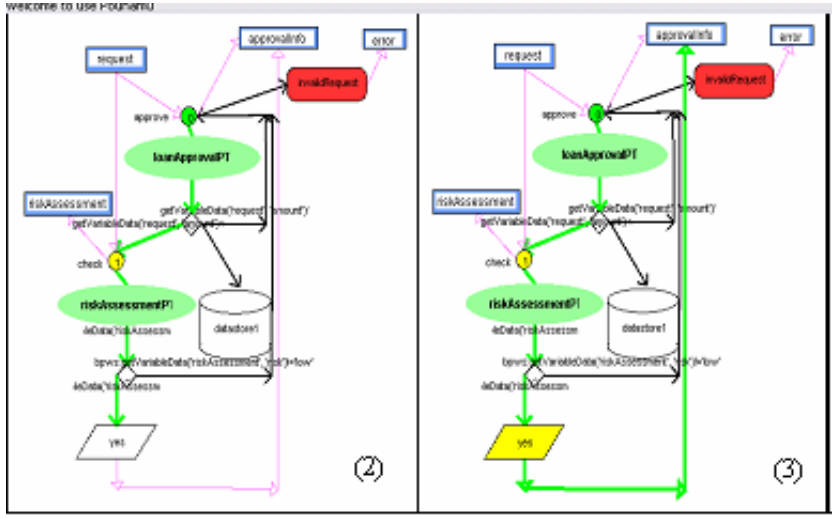
ViTABaL-WS



- Environment for modelling compositions of web services
- Uses a "tool abstraction" paradigm (metaphor)
- Generates BPEL4WS
- Provides "debugger" for running BPEL

BPEL4WS Generation & Execution



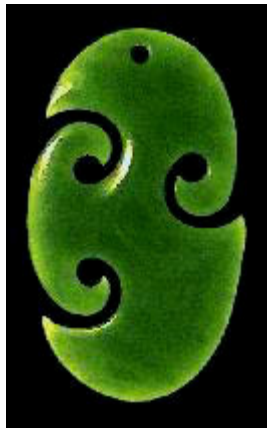


```
<receive name= "receive" partnerLink="customer"
         portType="loanApprovalPT"
         operation="approve"
         variable="request"
         createInstance="yes">
         <!-links-->
</receive>
<invoke name="invokeapprover" partnerLink="approver"
         portType="loanApprovalPT"
     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

IBM BPWS4J Workflow Engine

Building DSVL Tools…

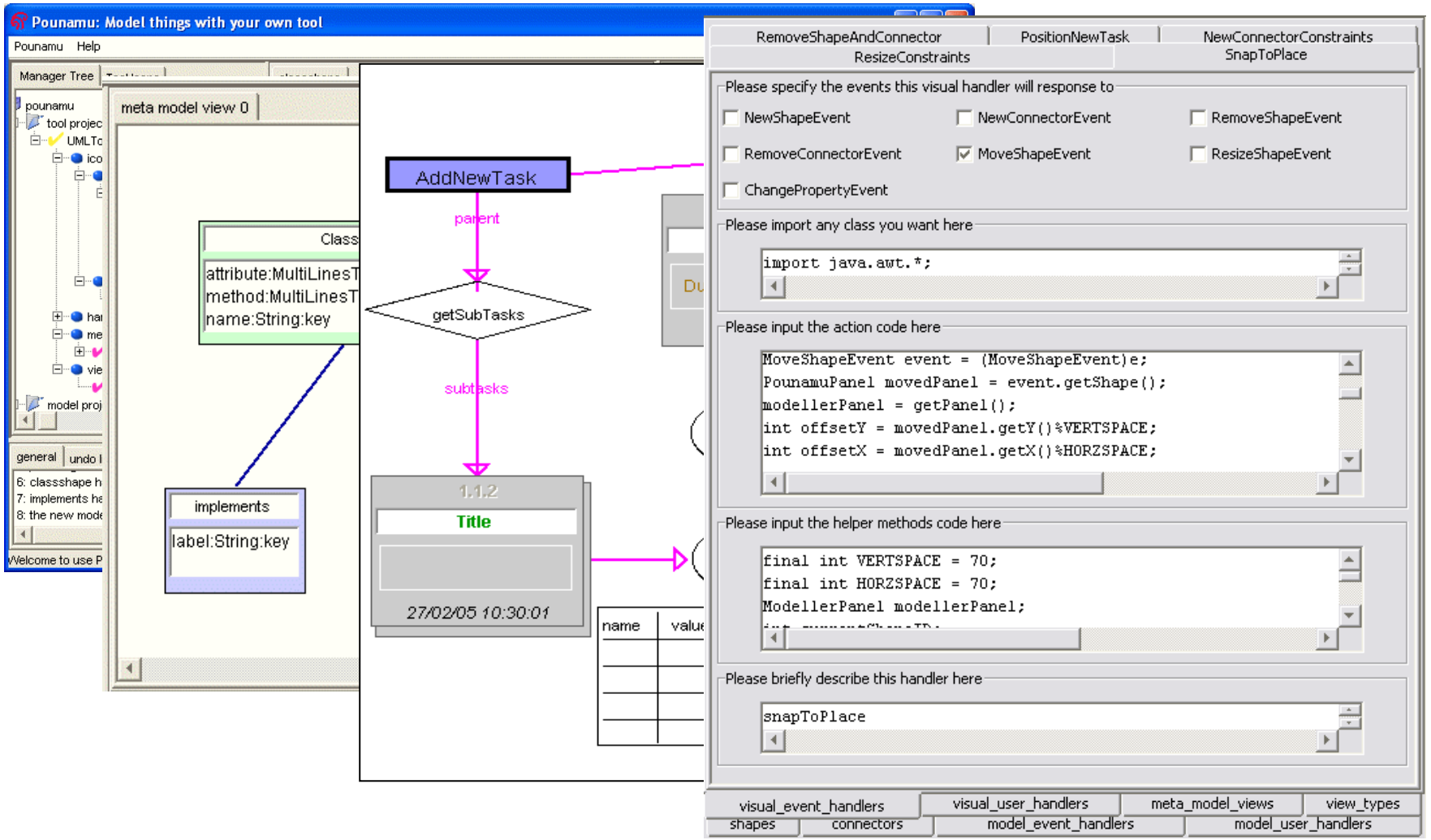
- Its hard to build these things…
  - What are the "right" visual metaphor(s)?
  - What model(s) do we need to represent/build?
  - How to generate code/configurations from model?
  - How do we achieve integration with other tools
  - How do we make them practical for users?
- Our approach to date:
  - 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, collaboration plug-ins



Pounamu

SE Software Engineering
The University of Auckland

Pounamu Meta-tools (themselves DSVLS!)

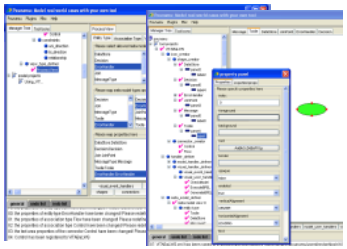


But…

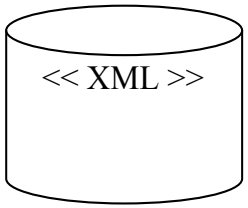
- Pounmau is stand-alone, our own IDE
- While it has good extensibility/integration support via web services API, not a "commercial quality" IDE
- Still too difficult to integrate 3rd party tools

- Solution: Use the open-source, commercial quality Eclipse IDE to realise Pounamu-specified DSVL tools
- Eclipse provides:
  - Open architecture IDE via plug-ins, very nice APIs
  - Wide range of 3rd party tools
  - Nice plug-ins & tools for building DSVL tools: EMF, GEF, JET, …
  - Very well-engineered system

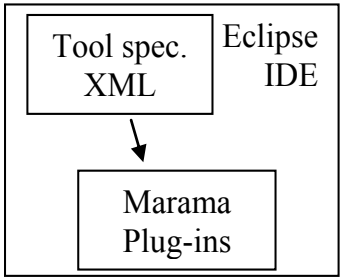
Marama ("the moon")



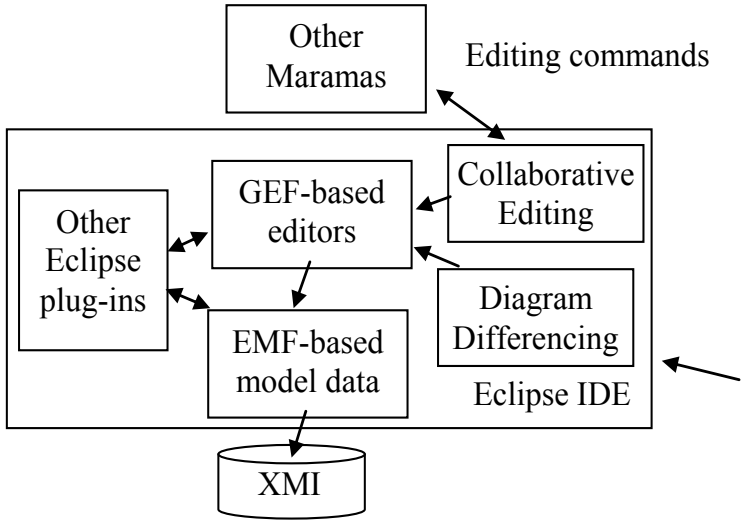
1. Visual language tool specification in the Pounamu meta-tool



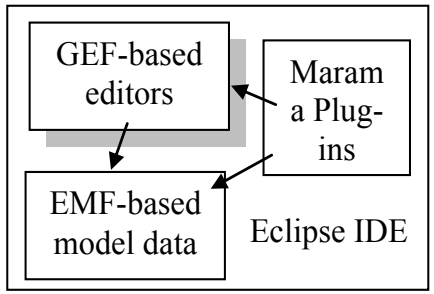
2. Pounamu saves tool specification to XML files



3. Load tool specification into Marama plug-in

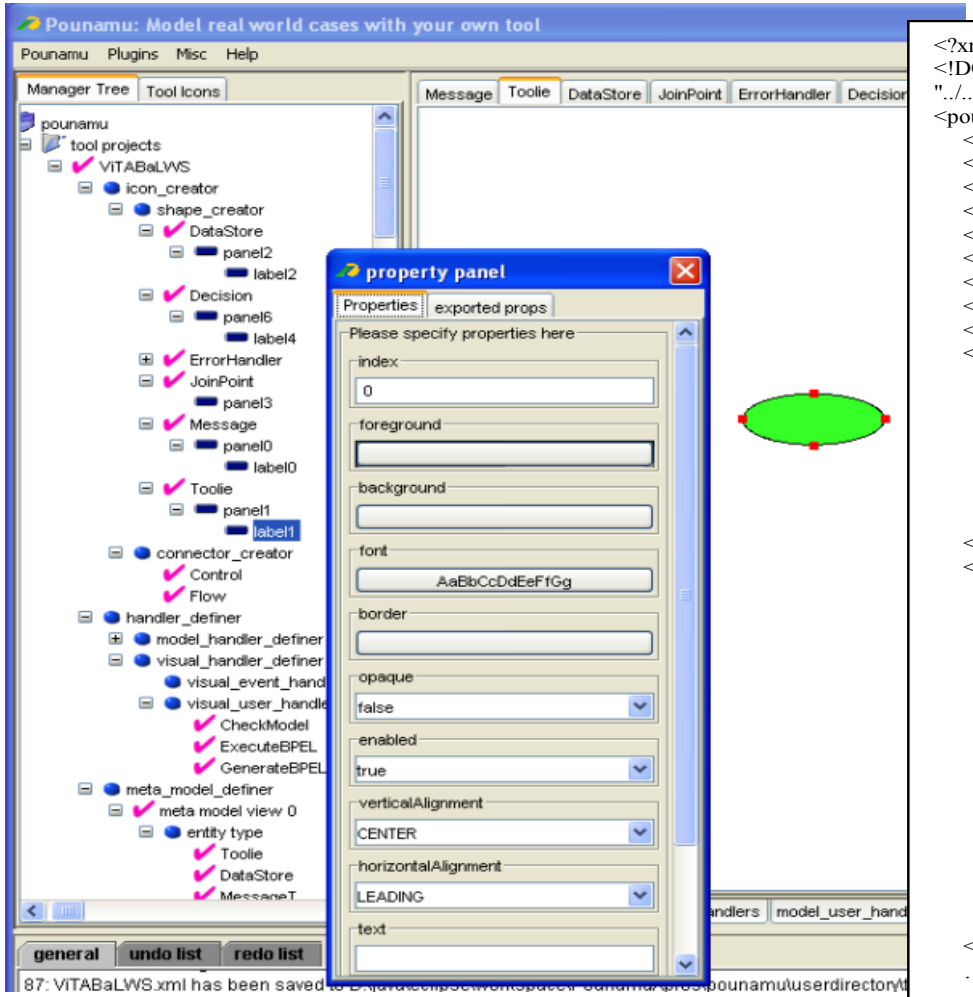


5. Modelling tool data saved/loaded as XMI; multi-user support via plug-ins



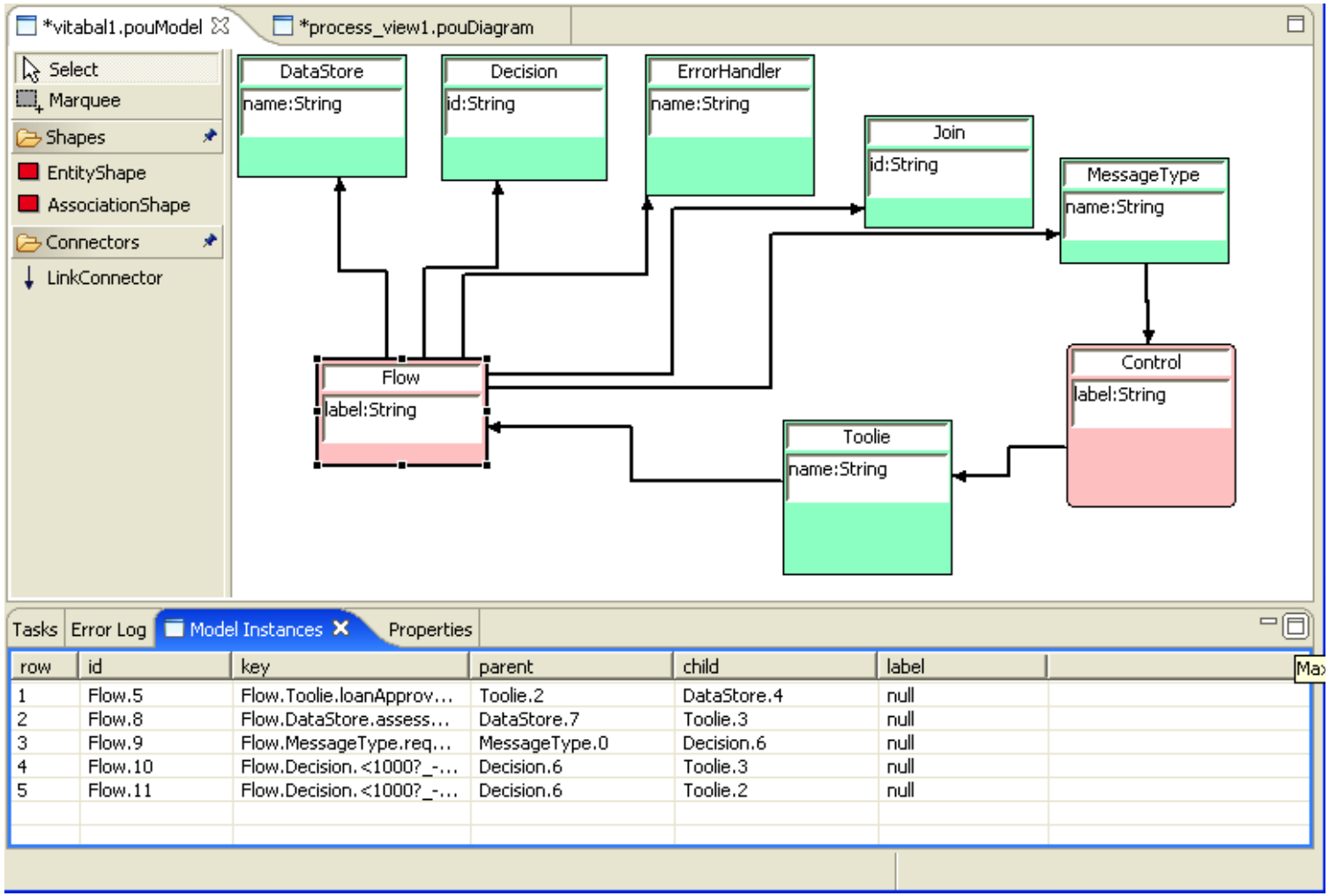
4. Marama dynamically configures GEF editors; uses EMF-based model data

Example Usage: 1. Develop Pounamu DSVL tool spec

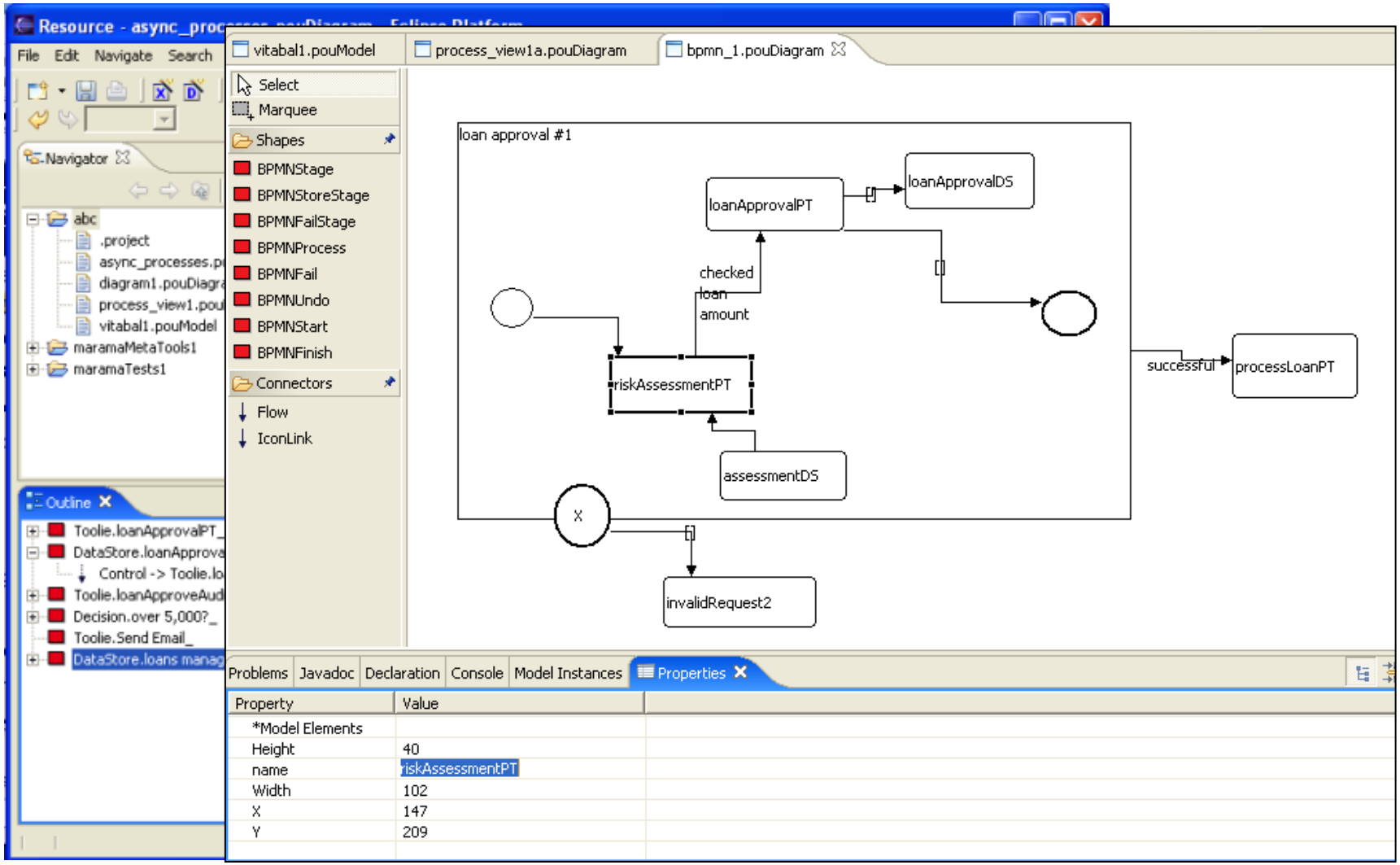



```
<?xml version="1.0"?>
<!DOCTYPE pounamushape SYSTEM
"../../../../../nonjavafiles/icon.dtd">
<pounamushape>
    <name>Toolie</name>
    <source>Toolie</source>
    <thumbnailshapetype>square</thumbnailshapetype>
    <thumbnailsizetype>ratio</thumbnailsizetype>
    <thumbnailwidth>100</thumbnailwidth>
    <thumbnailheight>100</thumbnailheight>
    <displayname>panel1</displayname>
    <type>pounamu.core.visualcomp.PounamuPanel</type>
    <path>this</path>
    <property>
        <propertyname>type</propertyname>
        <propertytype>ShapeType</propertytype>
        <propertyflag>visual</propertyflag>
        <propertypath>this</propertypath>
        <propertyvalue>
            <simplevalue>Oval</simplevalue>
        </propertyvalue>
    </property>
    <property>
        <propertyname>stroke</propertyname>
        <propertytype>BasicStroke</propertytype>
        <propertyflag>visual</propertyflag>
        <propertypath>this</propertypath>
        <propertyvalue>
            <linewidth>1.0</linewidth>
            <endcaps>2</endcaps>
            <linejoints>0</linejoints>
            <dasharray0>10.0</dasharray0>
            <dasharray1>0.0</dasharray1>
            <dasharray2>10.0</dasharray2>
            <dasharray3>0.0</dasharray3>
            <miterlimit>10.0</miterlimit>
            <dashphase>0.0</dashphase>
        </propertyvalue>
    </property>
    …
```

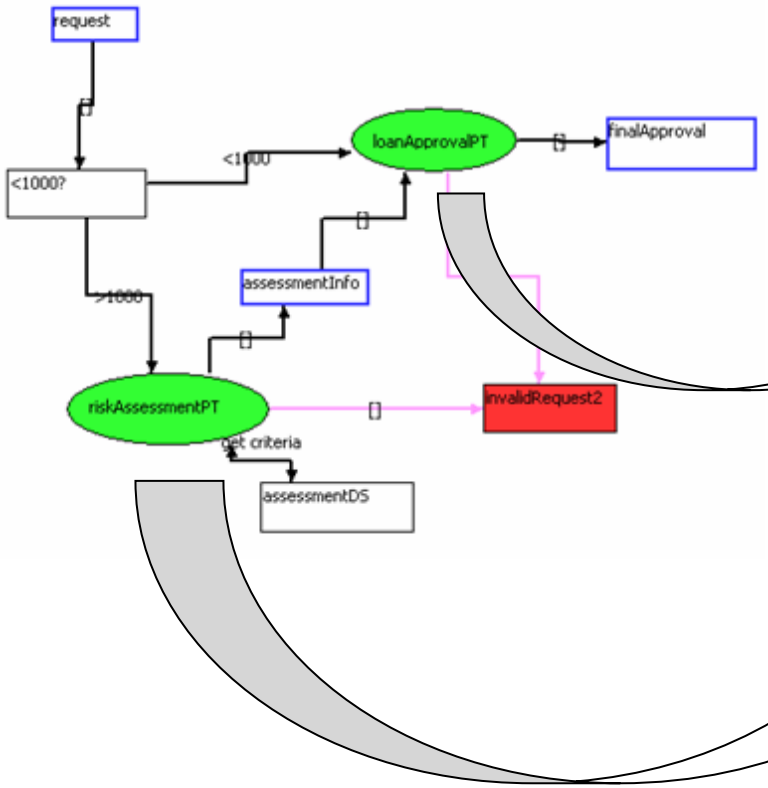
2. Load tool specification into Marama



3. Create Visual Models



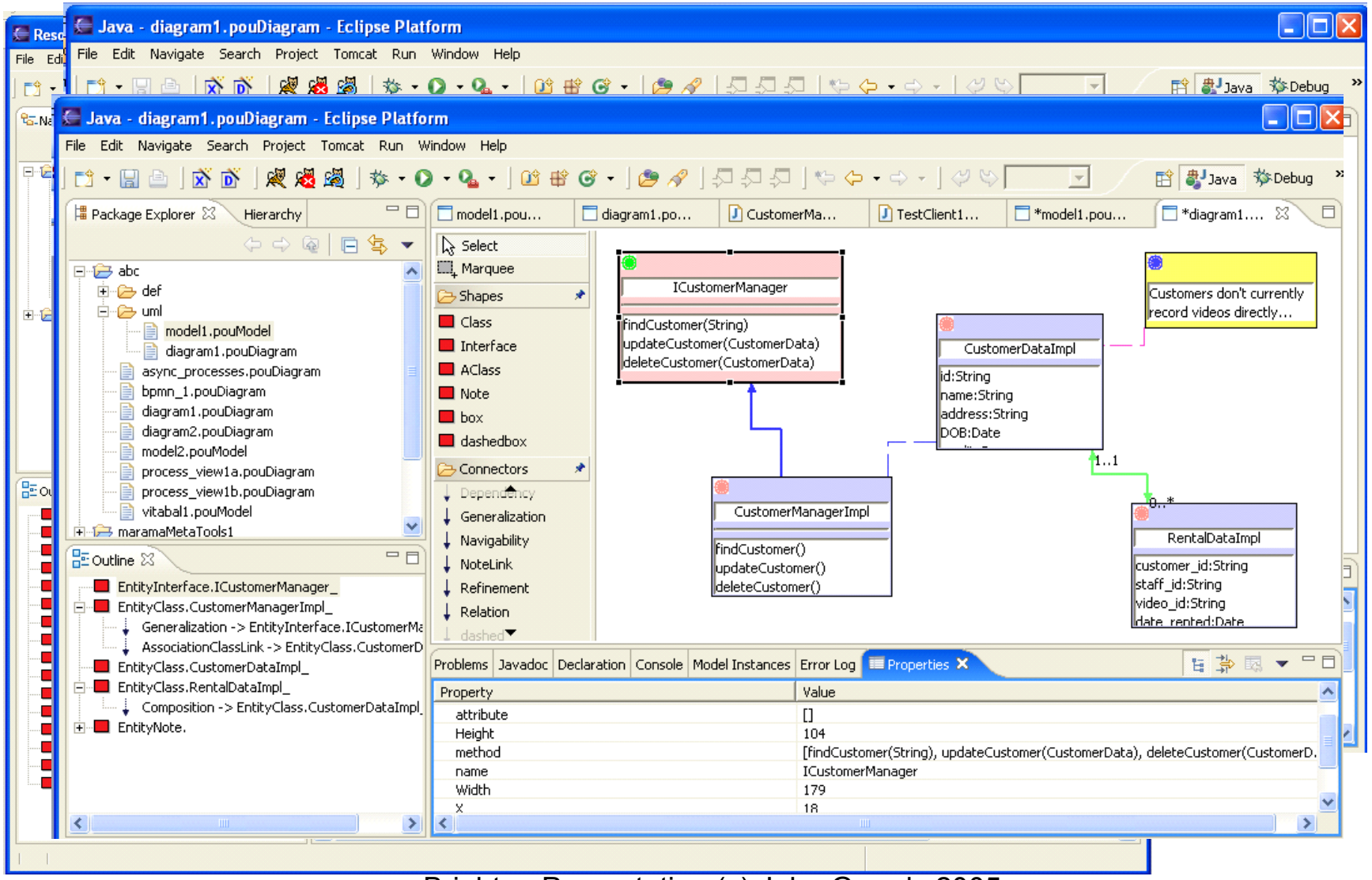
BPEL Generation



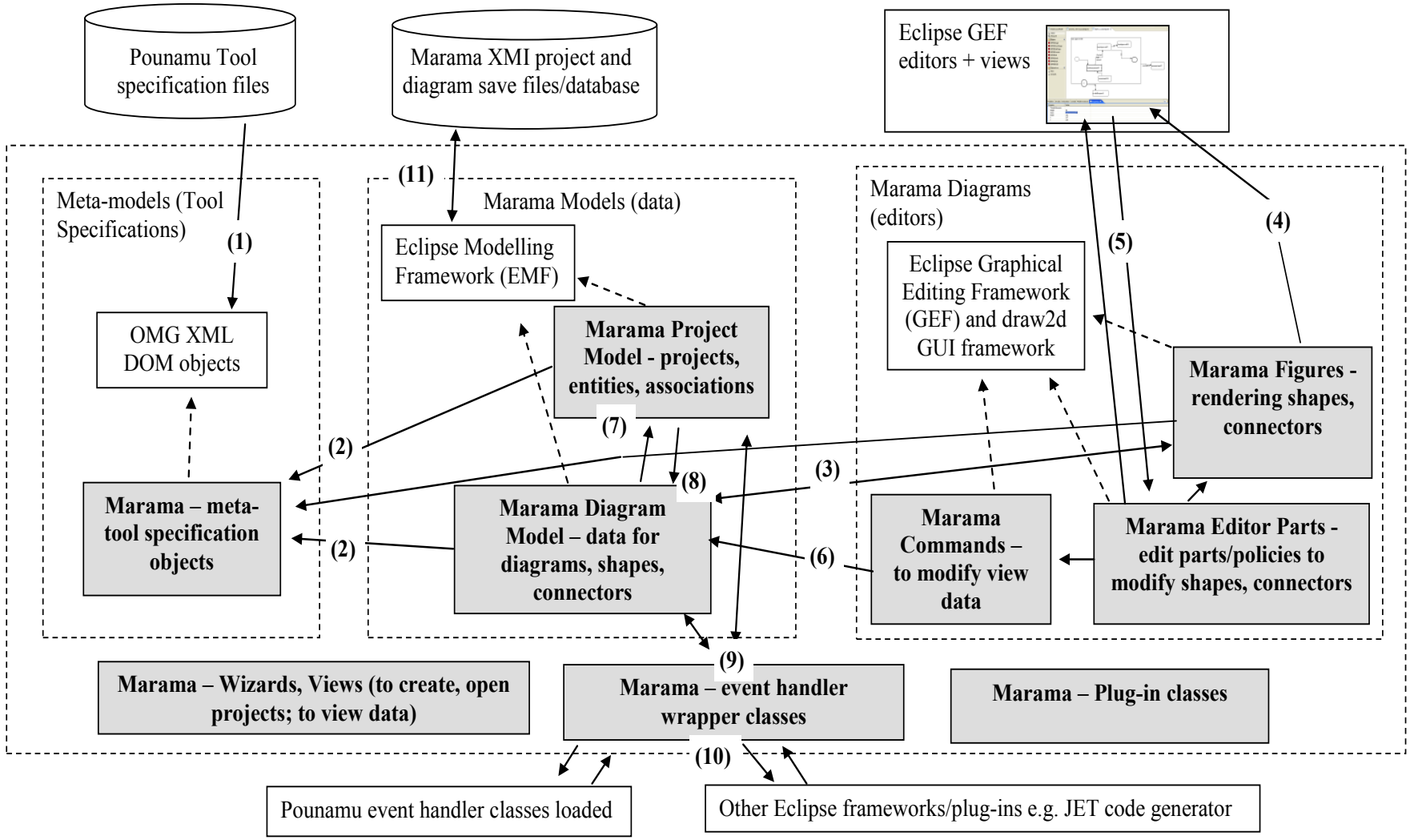

```
<receive name="receive" partnerLink="customer"
         portType="loanApprovalPT"
         operation="approve"
         inputVariable="request"
         outputVariable="FinalApproval">
         <!--links-->
<invoke name="invokeapprover"
partner  k="approver"
         rtType="loanApprovalPT"
         operation="approve"
         inputVariable="request"
         outputVariable="FinalApproval">
         <!--links-->
<invoke name="invokeassessor"
partner  nk="assessor"
         portType="riskAssessmentPT"
         operation="check"
         nputVariable="request"
         outputVariable="assessmentInfo">
         <!--links-->
```

Other Marama DSVLs...

SE Software Engineering
The University of Auckland



Marama Design



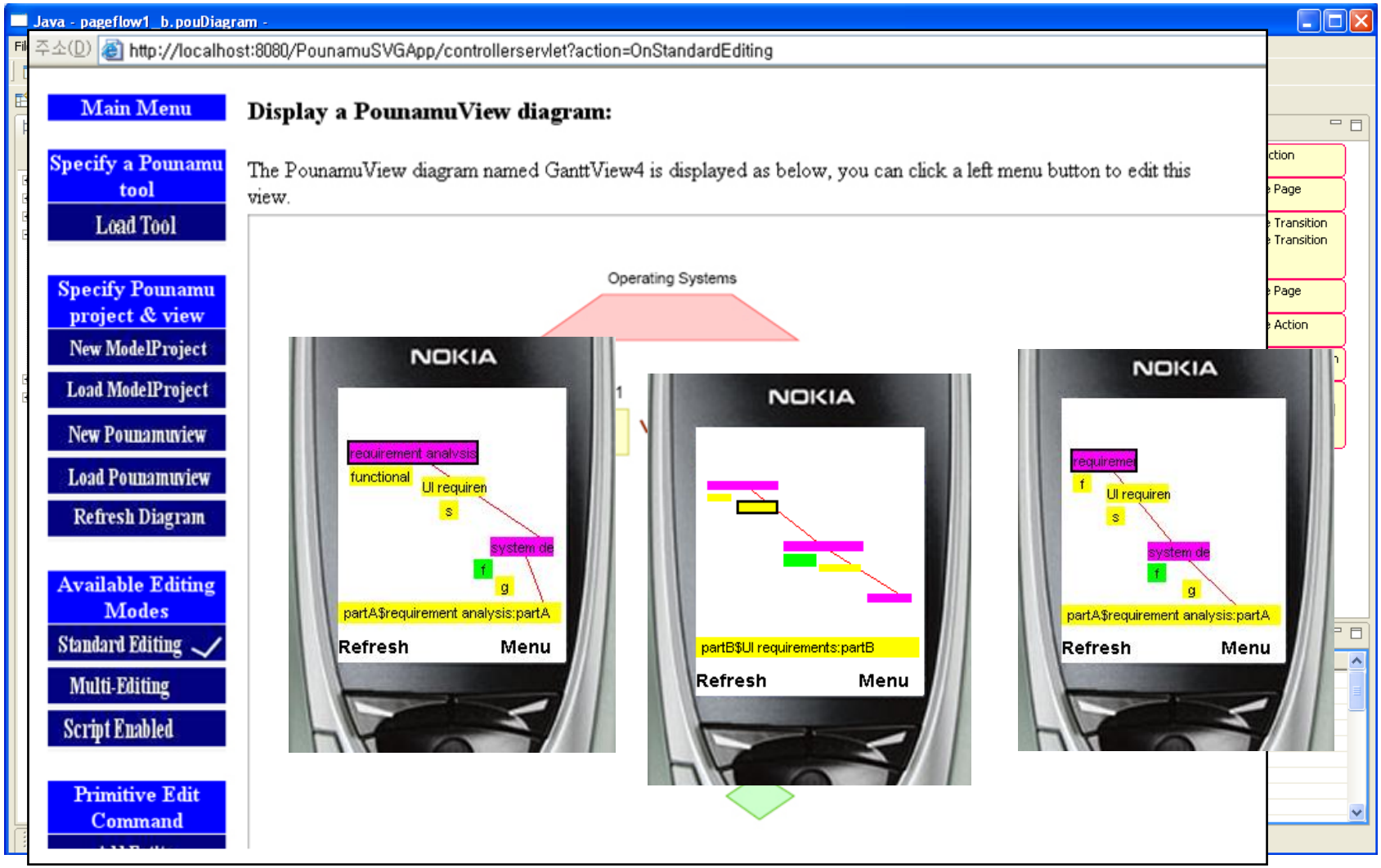
Evaluation

- Various DSVL prototype tools:
  - UML design tools
  - Web service orchestration
  - Process modelling and project management tools
  - Circuit design tool
  - Performance test bed generation tool
  - Visual data mapping tools
  - Marama meta-tools… ☺
- Cognitive Dimensions evaluation: Marama vs Pounamu
  - Better *Closeness of Mapping* ; higher tool *Viscosity*; good *Consistency* & low *Error-proneness*; much improved *Hidden dependencies* and *Juxtaposability*

Current/Future Work

- Re-implementing Pounamu meta-tools in Marama – as Marama DSVL tools!
- Enhancing specification of dependencies using spreadsheet-style formulae and event-condition-action DSVL
- Extending DSVL editing/rendering support e.g. better support composite shapes, editing in-situ of text etc
- Marama provides dynamic interpretation of Pounamu specifications – also looking at static EMF/GEF generation (hope to feed into Eclipse GMF work…)
- Collaborative work support; thin-client diagramming
- Building more DSVLs (can never have enough… ☺)

Examples: MaramaDiffer; MaramaThin



Summary

- Domain-specific visual language tools provide powerful model representation, modelling support
- Building DSVL tools is hard; integrating with existing tools challenging
- Pounamu provides meta-tool specification
- Marama provides Eclipse plug-ins to realise these specifications as near-commercial quality DSVL tools
- Allows us to deploy DSVL tools on realistic problems, with industry, and to scale DSVL research results
- Various extensions to Marama including its own meta-tools (which themselves are Marama DSVLs)

References

Zhu, N., Grundy, J.C. and Hosking, J.G. Constructing domain-specific design tools with a visual language meta-tool, CAiSE 2005 Forum, Portugul, June 2005, Springer.

Zhu, N., Grundy, J.C. and Hosking, J.G., Pounamu: a meta-tool for multi-view visual language environment construction, In Proceedings of the 2004 International Conference on Visual Languages and Human-Centric Computing, Rome, Italy, 25-29 September 2004, IEEE CS Press, pp. 254-256.

Mehra, A., Grundy, J.C. and Hosking, J.G. A generic approach to supporting diagram differencing and merging for collaborative design, In Proceedings of the 2005 ACM/IEEE International Conference on Automated Software Engineering, Long Beach, California, Nov 7-11 2005, IEEE Press, pp. 204-213

Cao, S. Grundy, J.C., Hosking, J.G., Stoeckle, H. and Tempero, E. An architecture for generating web-based, thin-client diagramming tools, In Proceedings of the 2004 IEEE International Conference on Automated Software Engineering, Linz, Austria, September 20-24, IEEE CS Press, pp. 270-273

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

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., 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.

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

Liu, A., 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. 321-324