Marama: an Eclipse-based meta-tool for generating multi-view graphical modelling tools

John Grundy

Department of Electrical & Computer Engineering
University of Auckland, New Zealand

j.grundy@auckland.ac.nz

John Hosking

Department of Computer Science,
University of Auckland, New Zealand

j.hosking@auckland.ac.nz

CSI Centre for Software Innovation

SPPI Software Process & Product Improvement

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Outline

- Models in SoftEng (and elsewhere)
- Our history in building modelling tools
- Marama motivation/requirements
- Marama overview
- Examples of Marama modelling tools
- Current & future work
- Conclusions

Models, models everywhere...

- Software engineering:
  - OOA/D, requirements, processes, networks, tests, configurations, code, …
- Construction/Engineering/Comp Systems:
  - Structures, plant, plumbing/electrics, materials, …
  - VHDL, electromagnetics, processes/tasks, …
- Health:
  - Patient diagnoses, treatments, imaging, …
- Business:
  - Processes/workflow, financial, economic (!), …
- Others:
  - Families, Friends/social/business networks, ...

Working with models

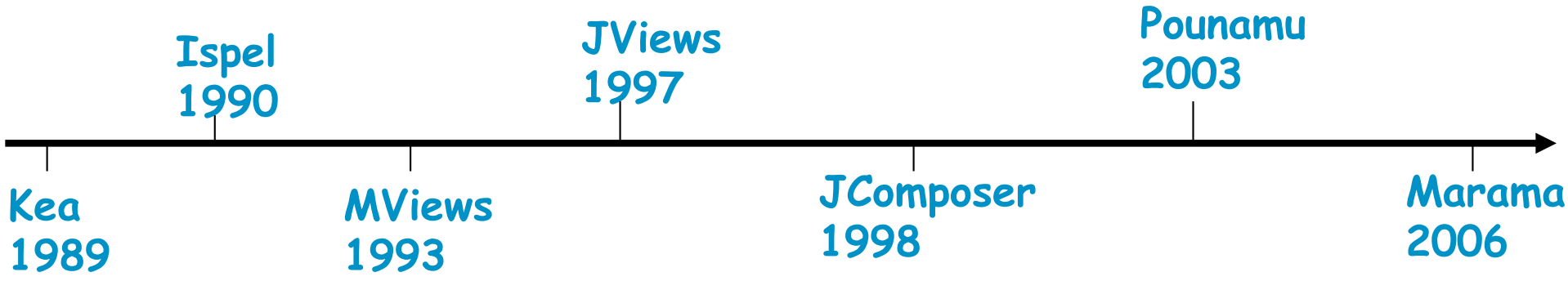
- Authoring, visualising, navigating, transforming, understanding, evolving, …
- Requires appropriate TOOLS to support these
- Tools must be usable, scalable, sharable, robust, extensible
- Ideally we want to provide *domain-specific visual languages* (DSVLs) to represent (parts of) models in "closeness of fit" to end user/domain
- We want tools to support these DSVLs
- BUT - building such DSVL modelling tools is HARD!

UoA Modelling Tools – a brief History

Design Tools-Engineering + Software

Frameworks for constructing multi-view multi-notation environments

Meta tools for specifying & constructing multi-view multi-notation environments



Applications developed using the frameworks & meta tools

Commercialisation/industry transfer

(An aside: Evolving Frameworks Pattern Language - a nice framework to describe this evolution…)

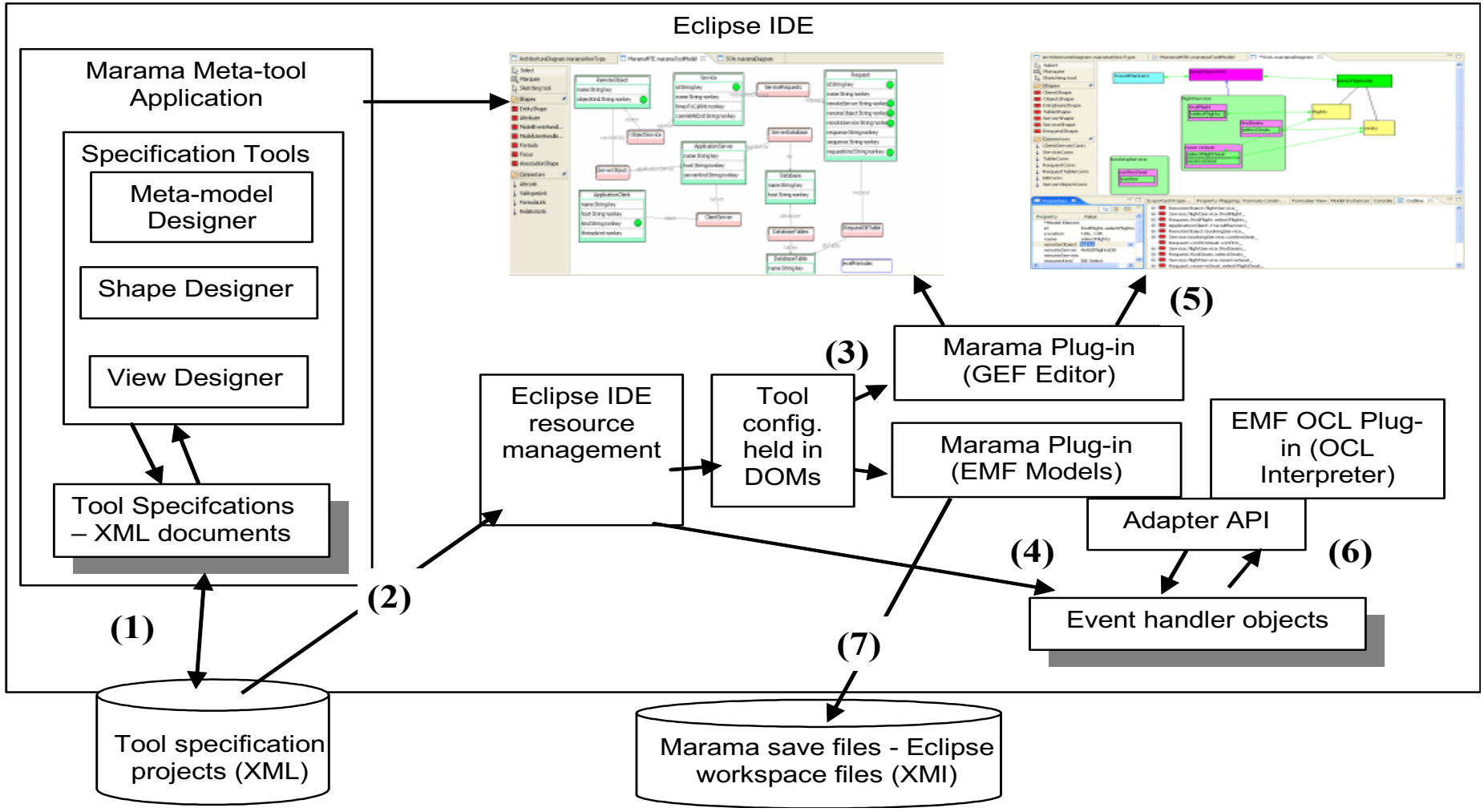
Marama – some key goals

- Make modelling tool implementation easier for:
  - Experienced domain *modellers* (may not be developers!)
  - Familiar with basic *modelling* concepts
    - Eg EER, OCL, meta models
  - Construct basic modelling tools within 1 day
    - Plus time for backend code generators etc
- Leverage strength of Eclipse platform
  - Standalone Pounamu left us with too much to support infrastructure to develop e.g. save/load, XML, GUI, remoting
    - Make use of EMF, GEF, JET, events, etc
  - Eclipse community & open source attractive
- Paper at ASE06 on early version of Marama
  - Used Pounamu metatools
  - Realised tools in Eclipse using Marama runtime plugin
- Paper at ICSE08 on (more or less) latest Marama toolset

Marama – some key requirements

- Need to be able to specify and generate:
  - Meta-model
    - represents the target model elements
  - Icons and connectors
    - visual representation(s) of model
  - Views and view to model mappings
    - View – model consistency
  - Behaviour
    - Constraints, operations
  - Model transformations
    - Backend code generation
    - Tool integration
  - Tool deployment
    - Scalable, sharable, usable, intelligent, ... tools

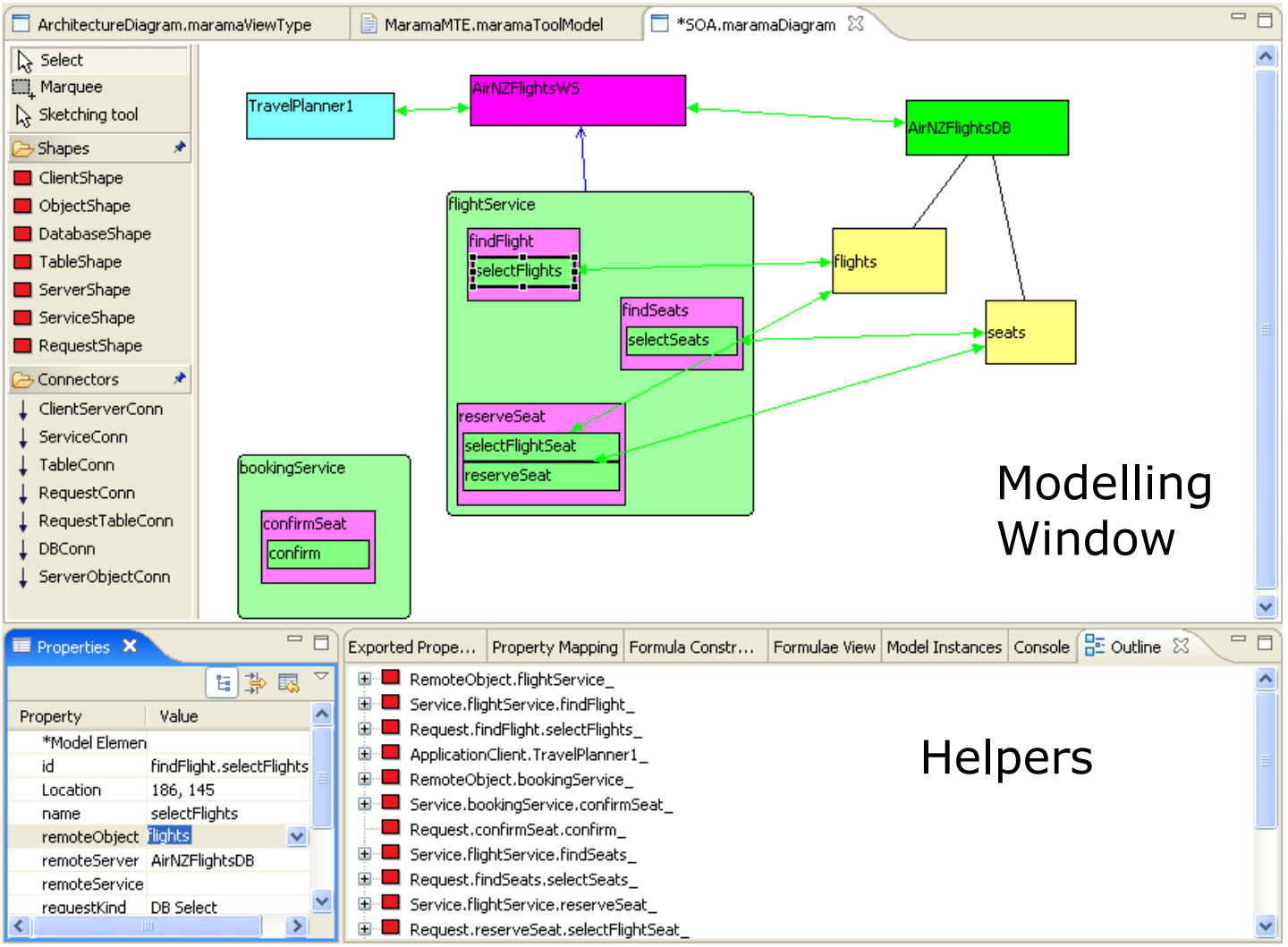
Marama – basic architecture



Example tool: MaramaMTE

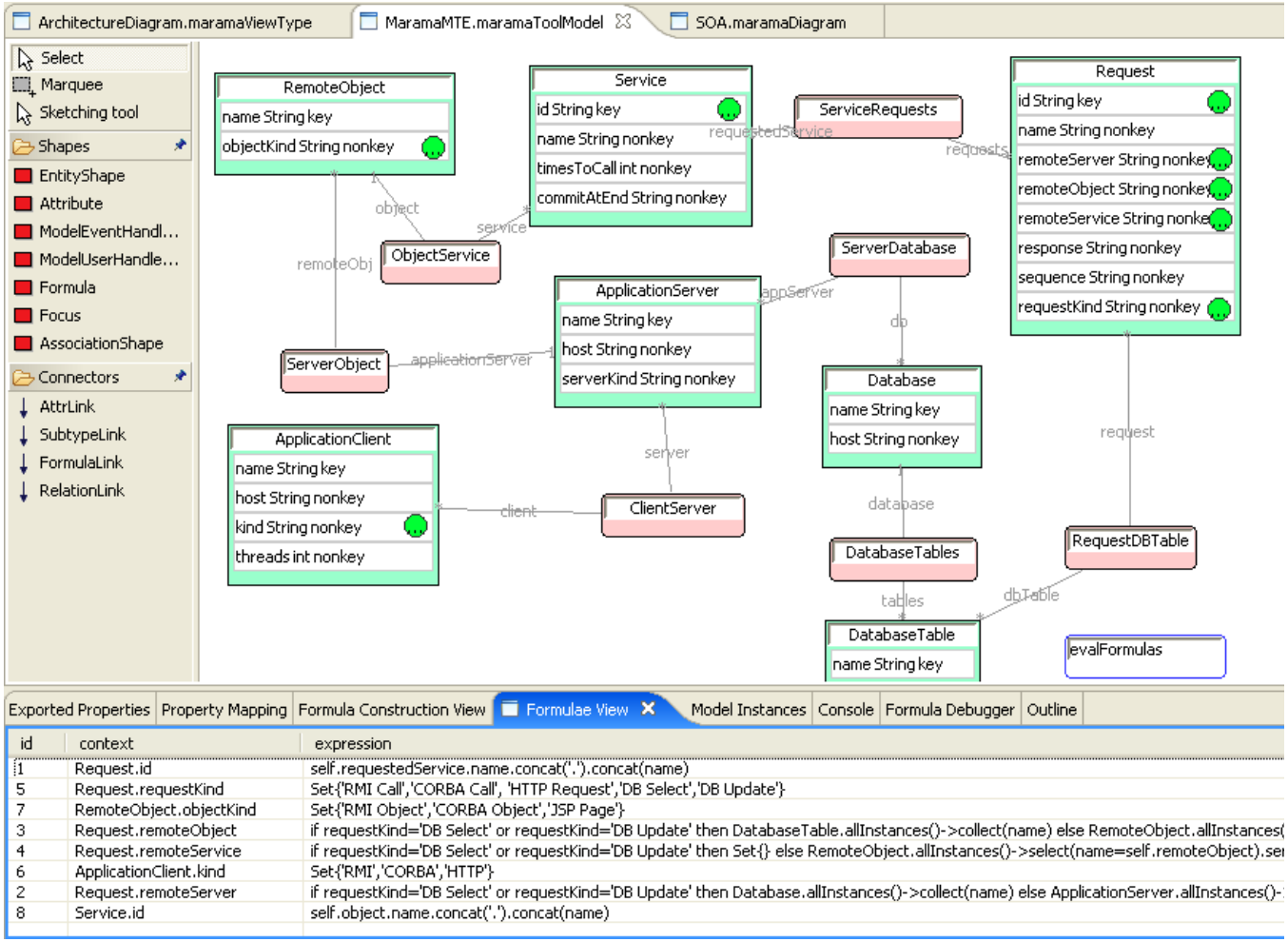
Palette

Property Sheet

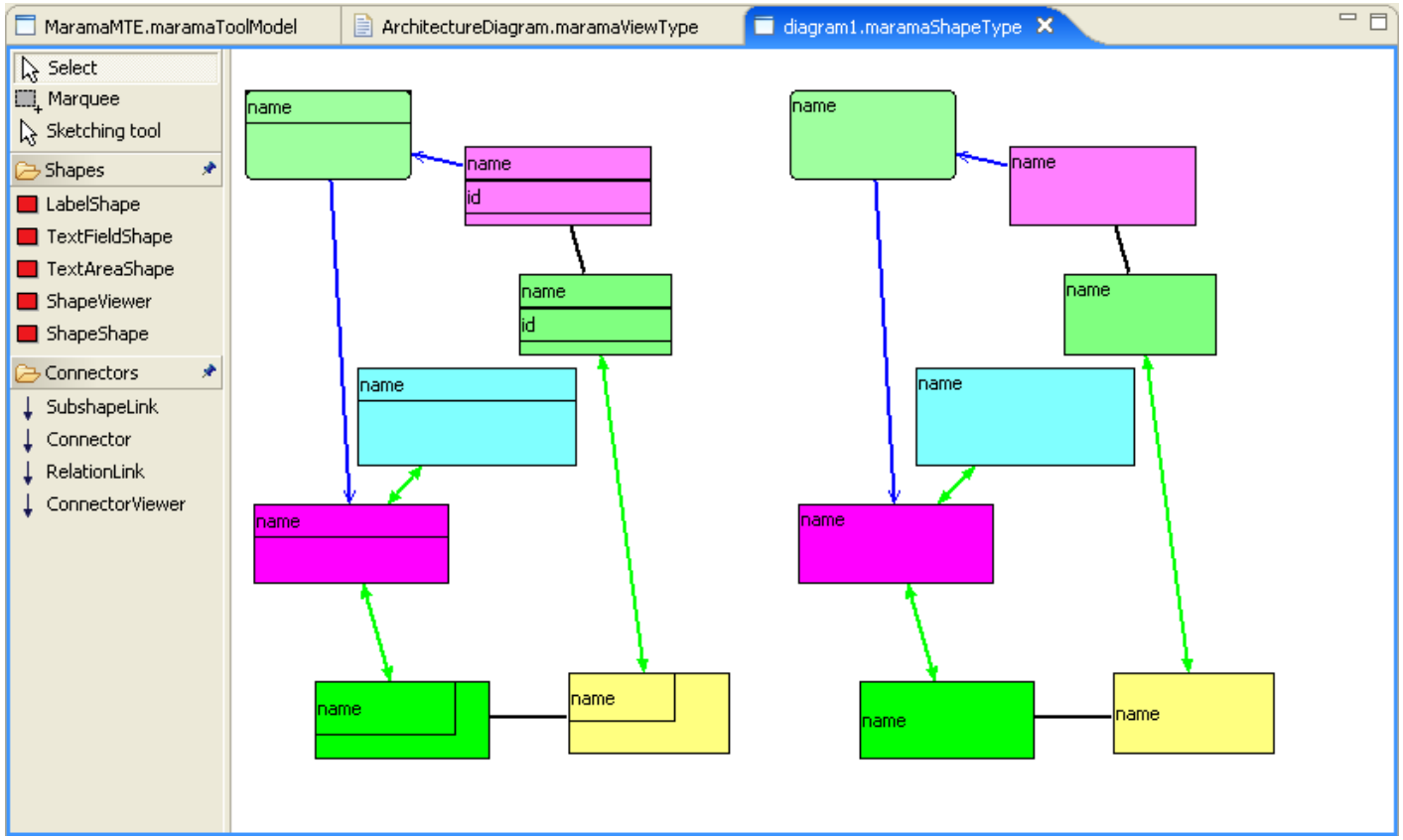


Meta model specification

- EER (KISS)
  - Entities
  - Relationships
  - Subtyping
  - Roles
  - Attributes
  - Keys
- OCL constraints (see later)
  - Attribute calcns
  - Invariants
  - Cardinalities

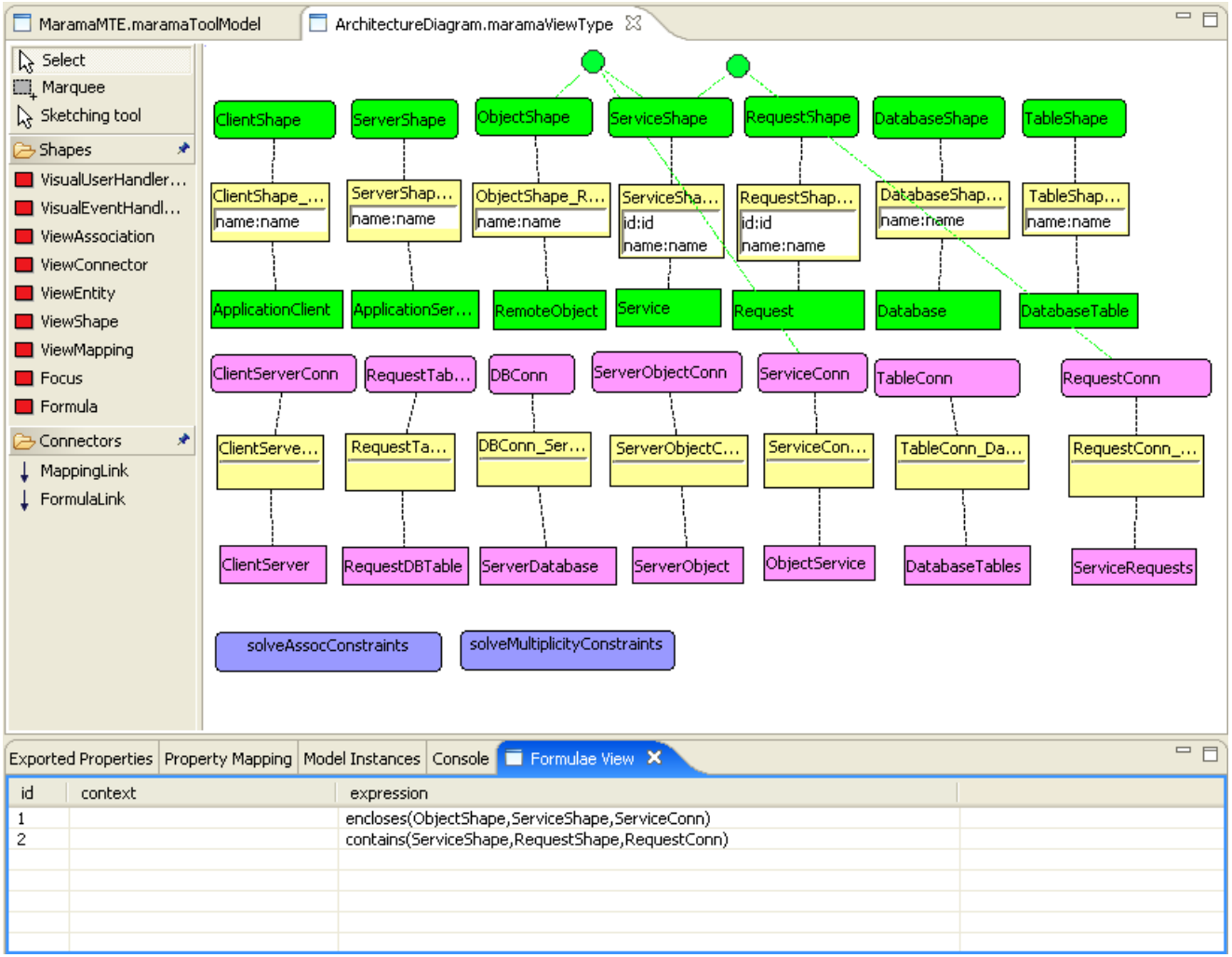


Icon and connector specification

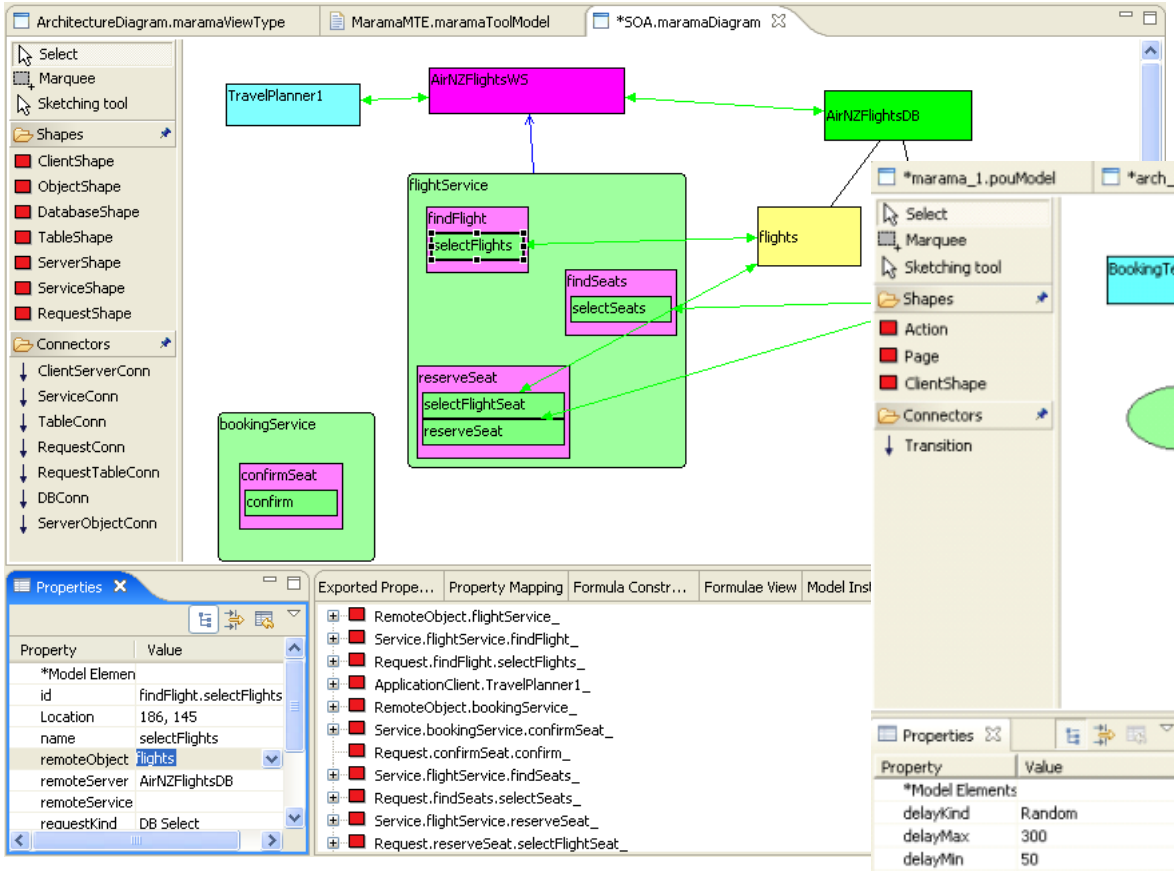


View and view-model mapping specn

- Elements in view
- Mappings
  - Entity to Icon
  - Relationship to connector
  - Attribute to property
- Constraints
  - Specialised relationships eg enclosure, containment

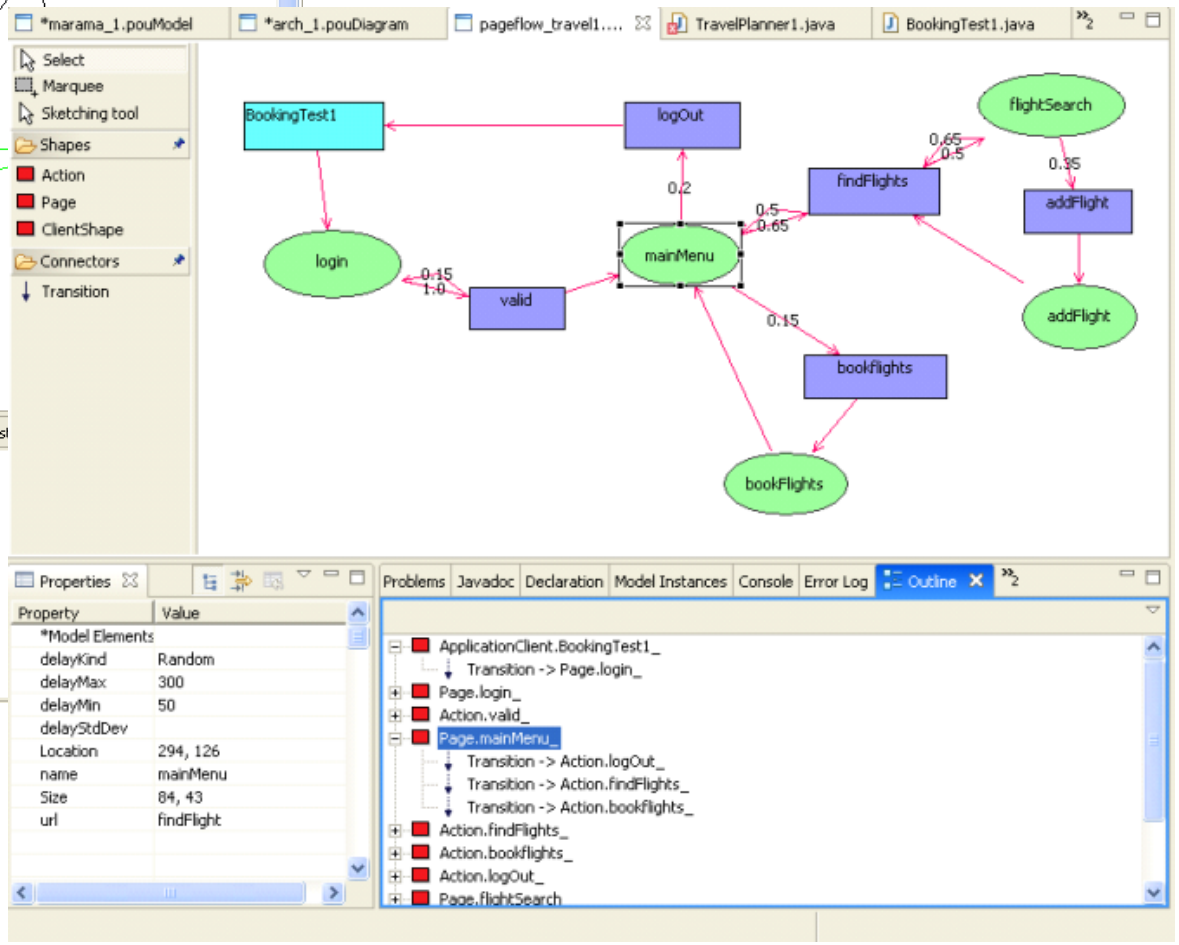


Generated tool – performance eng tool



Architecture spec

Web form interaction spec



Marama – key requirements

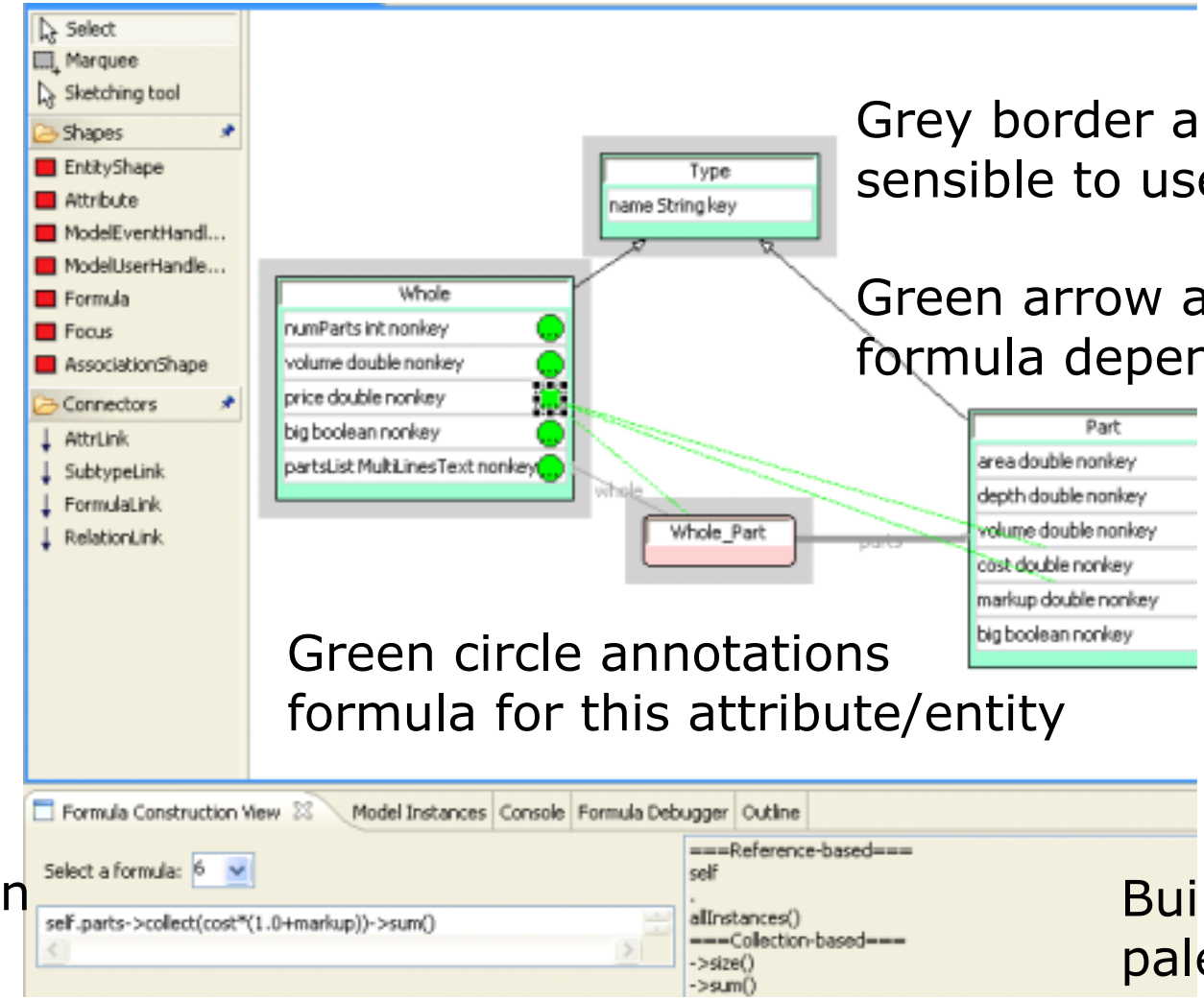
- Need to be able to specify/generate:
  - ✓ Metamodel
  - ✓ Icons and connectors
  - ✓ Views and view to model mappings
  - – Behaviour
    - Constraints, operations
  - – Model transformations
  - – Tool deployment

MaramaTatau – model level constraints

- Specification of behaviour always difficult in meta-tools:
  - Initial approach – Java event handlers (code plug-ins)
  - Clumsy to write, need detailed API knowledge etc

- MaramaTatau allows constraints to be specified as OCL expressions over the meta model elements:
  - Textual OCL expression
  - But constructed using spreadsheet approaches
  - Click and connect
  - High level visual repn

Constraint construction



Grey border annotations sensible to use in formula

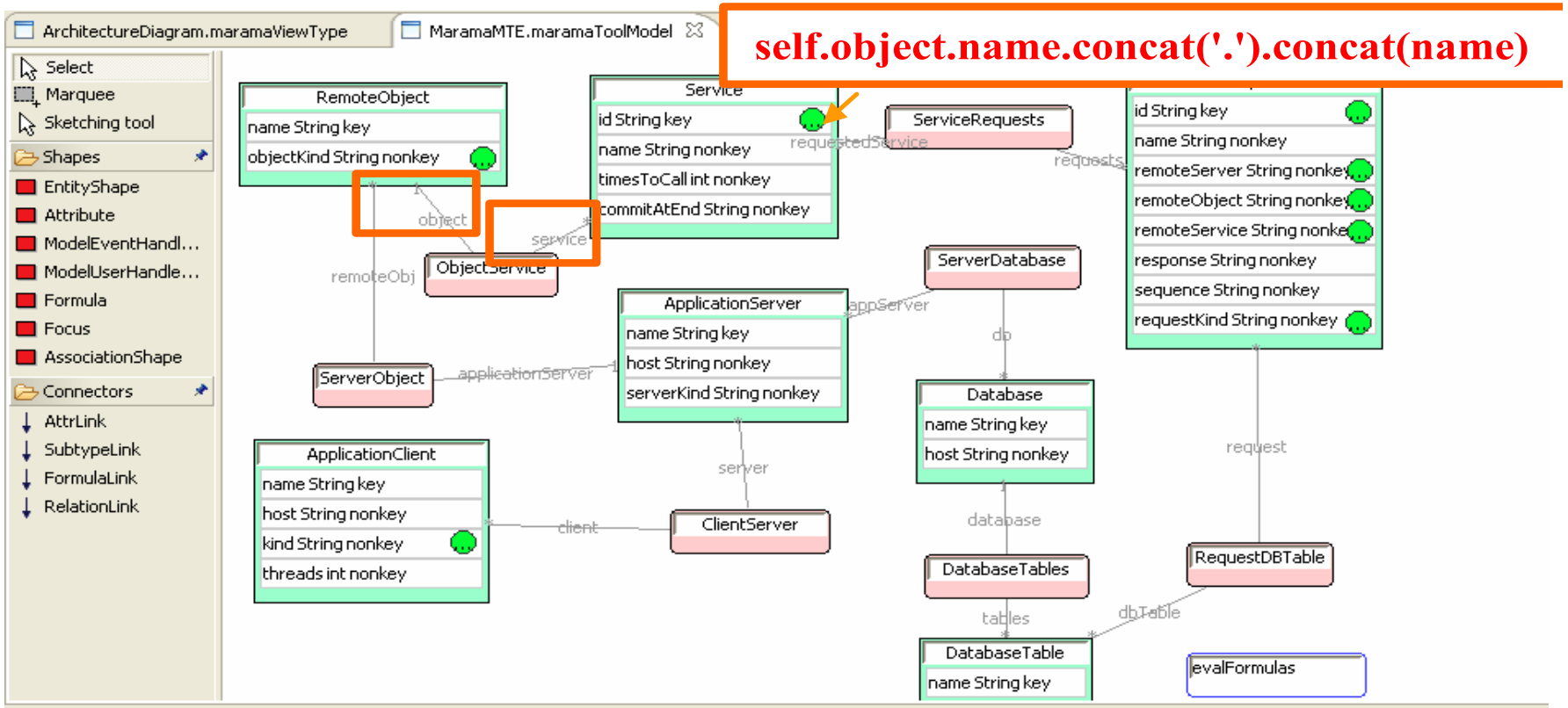
Green arrow annotations formula dependencies

Green circle annotations formula for this attribute/entity

Formula construction area

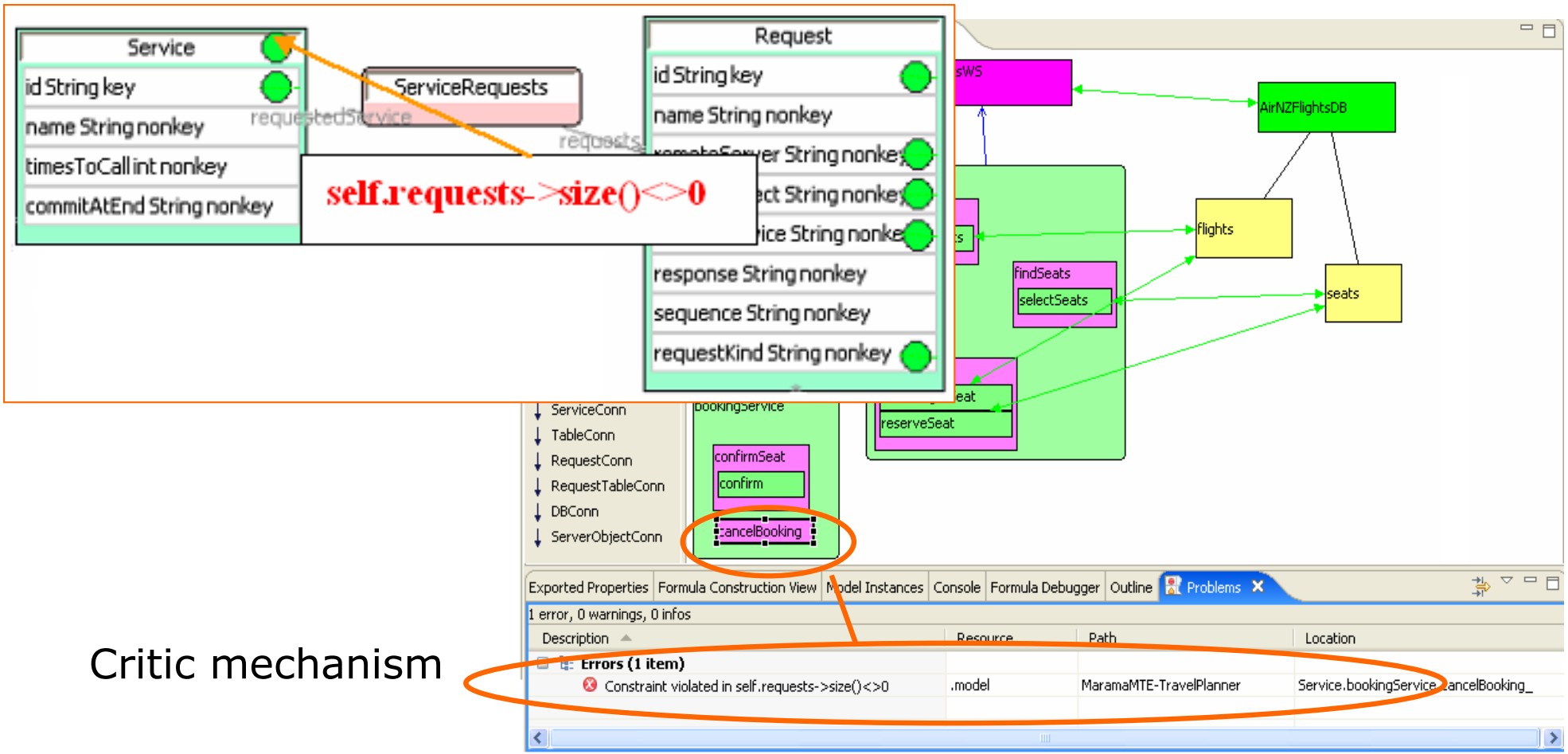
Built in function palette

MaramaMTE example



id	context	expression
1	Request.id	self.requestedService.name.concat('.').concat(name)
5	Request.requestKind	Set{'RMI Call','CORBA Call', 'HTTP Request','DB Select','DB Update'}
7	RemoteObject.objectKind	Set{'RMI Object','CORBA Object','JSP Page'}
3	Request.remoteObject	if requestKind='DB Select' or requestKind='DB Update' then DatabaseTable.allInstances()->collect(name) else RemoteObject.allInstances()->collect(nam
4	Request.remoteService	if requestKind='DB Select' or requestKind='DB Update' then Set{} else RemoteObject.allInstances()->select(name=self.remoteObject).service->collect(r
6	ApplicationClient.kind	Set{'RMI','CORBA','HTTP'}
2	Request.remoteServer	if requestKind='DB Select' or requestKind='DB Update' then Database.allInstances()->collect(name) else ApplicationServer.allInstances()->collect(name)
8	Service.id	self.object.name.concat('.').concat(name)

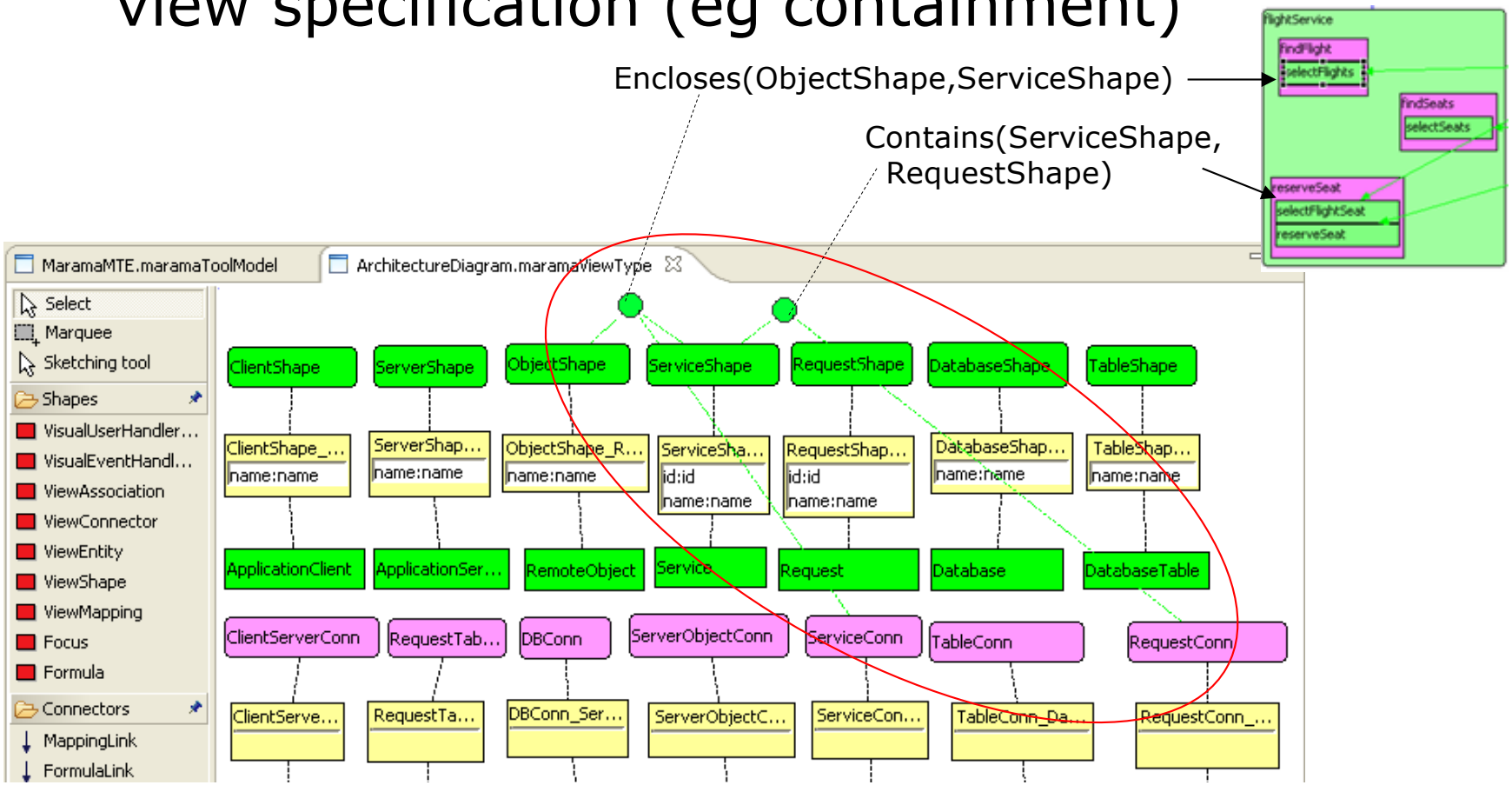
Constraint violation



Critic mechanism

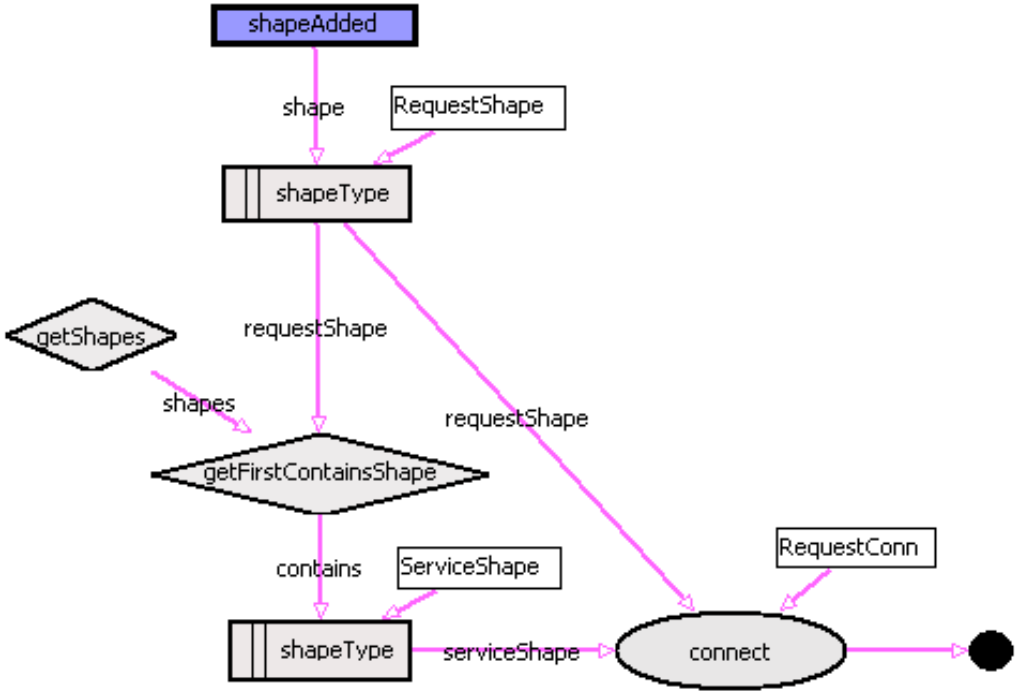
Visual constraints in views

- Can add some predefined layout constraints in view specification (eg containment)



Visual constraints in views

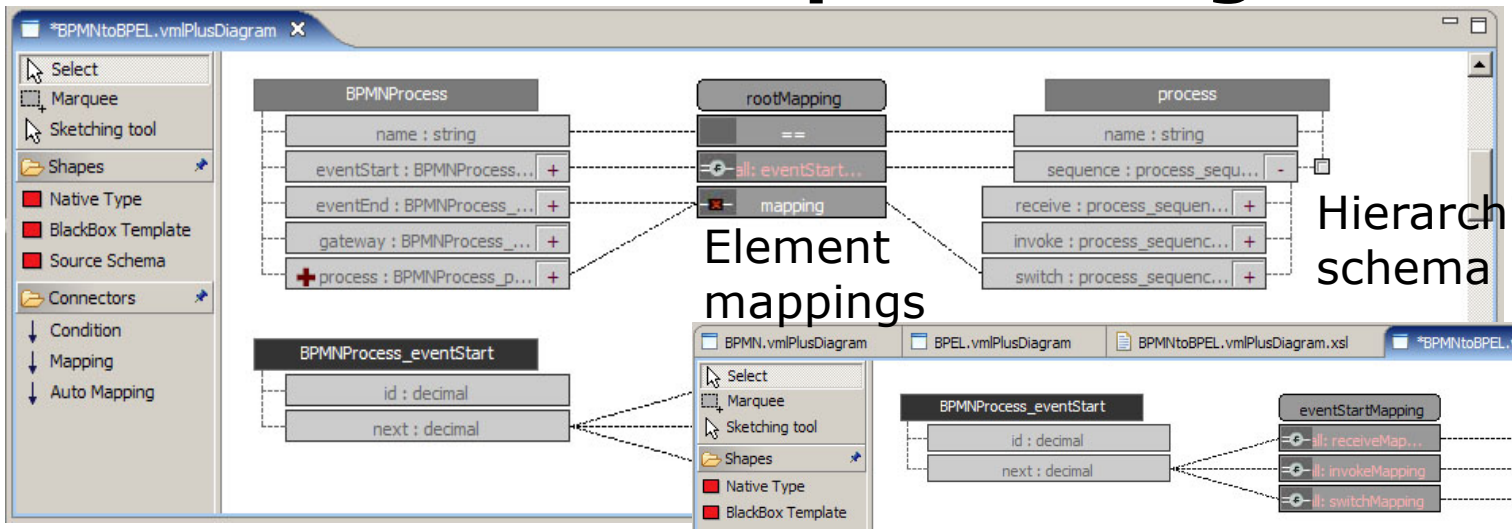
- OCL constraints in MaramaTatau – declarative; some limitations
- Kaitiaki: imperative visual event flow language for expressing view level constraints/operations
- Dataflow oriented
  - Push and pull
- Implemented in Pounamu
  - currently being ported into Marama



Marama basic requirements

- Need to be able to specify/generate:
  - ✓ Metamodel
  - ✓ Icons and connectors
  - ✓ Views and view to model mappings
  - ✓ Behaviour
  - – Model transformations
    - Backend code generation
    - Tool integration
  - – Tool deployment

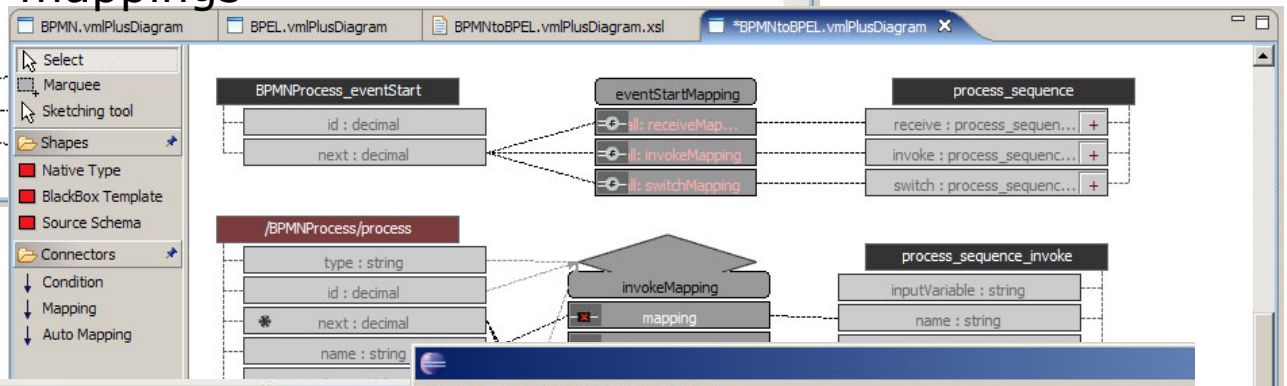
MaramaTorua –visual mapping/ model transformation specn and generation



Mapping specs

Hierarchical schema

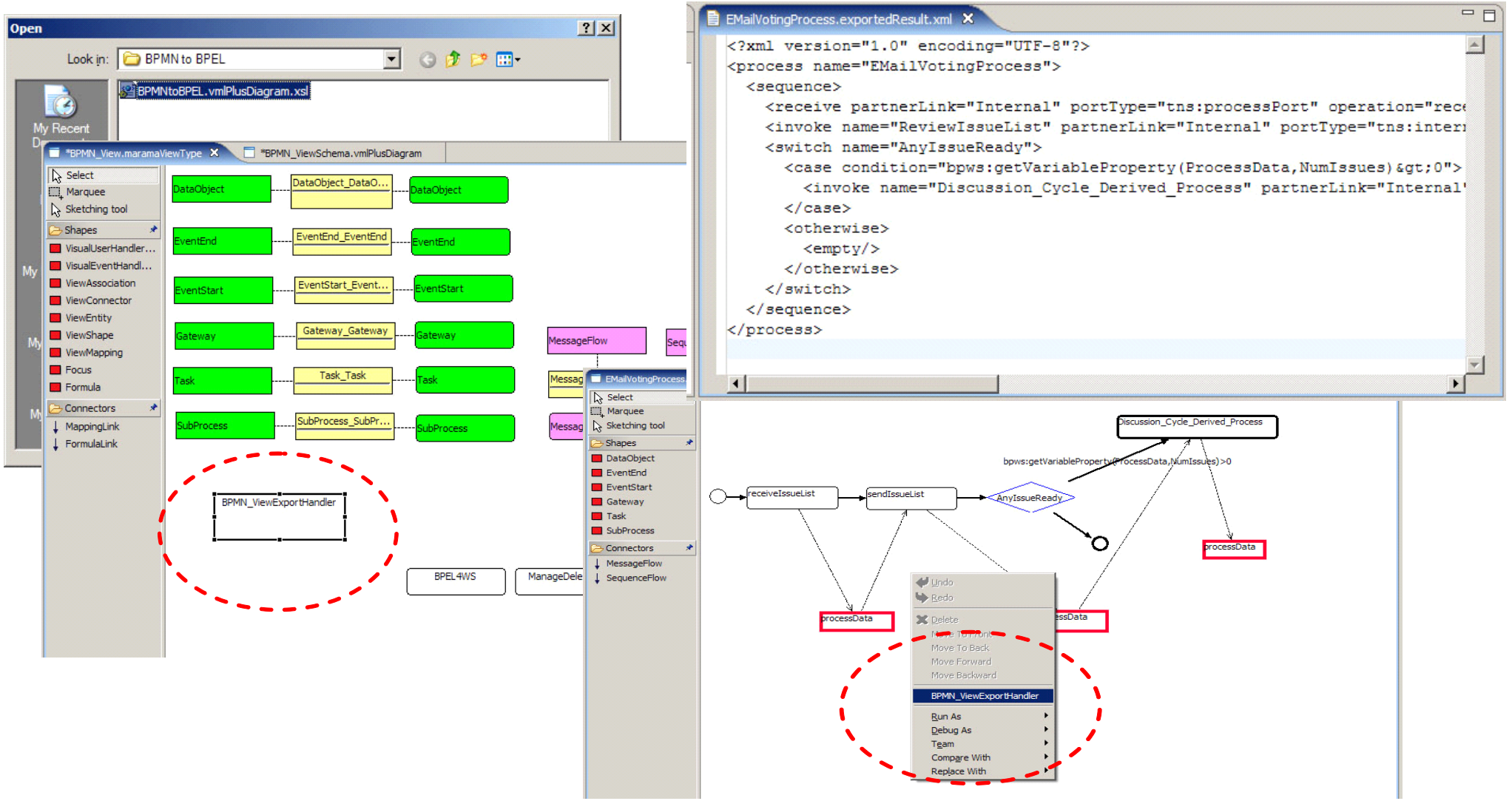
Element mappings



Generated XSLT

Mapping formula

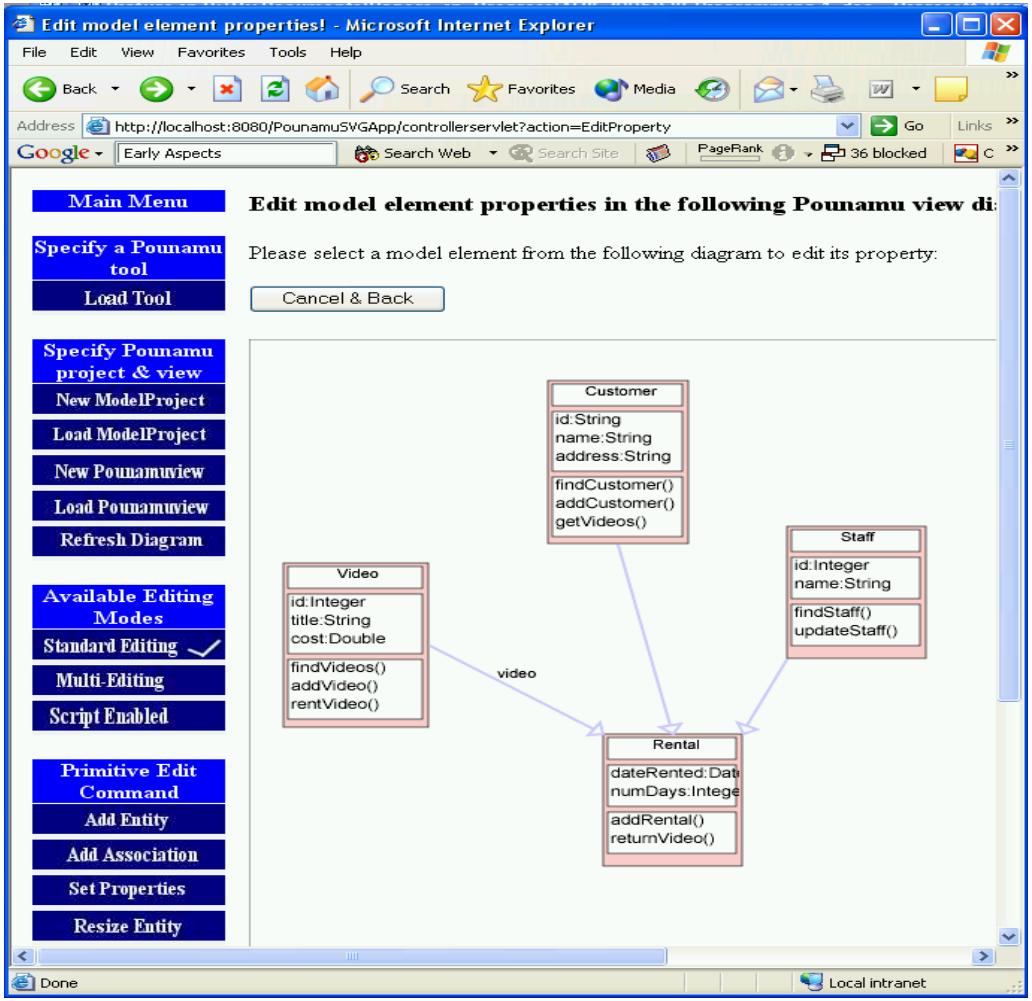
Installing mapping into a Marama tool

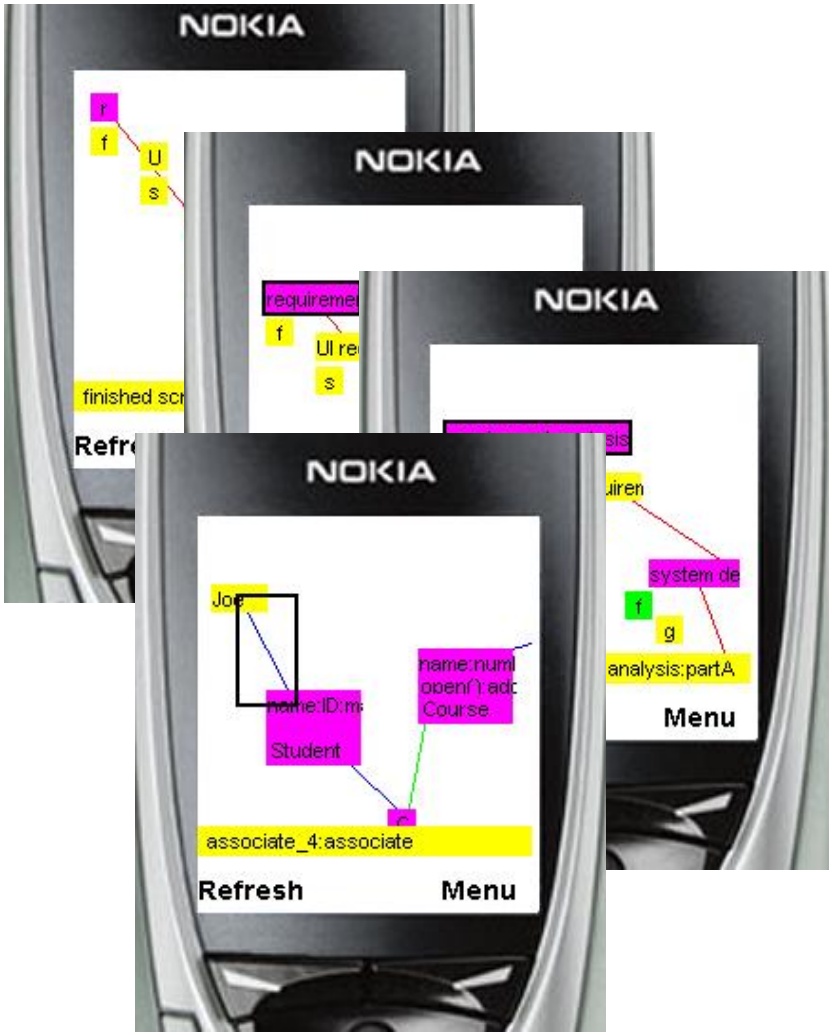


Marama – key requirements

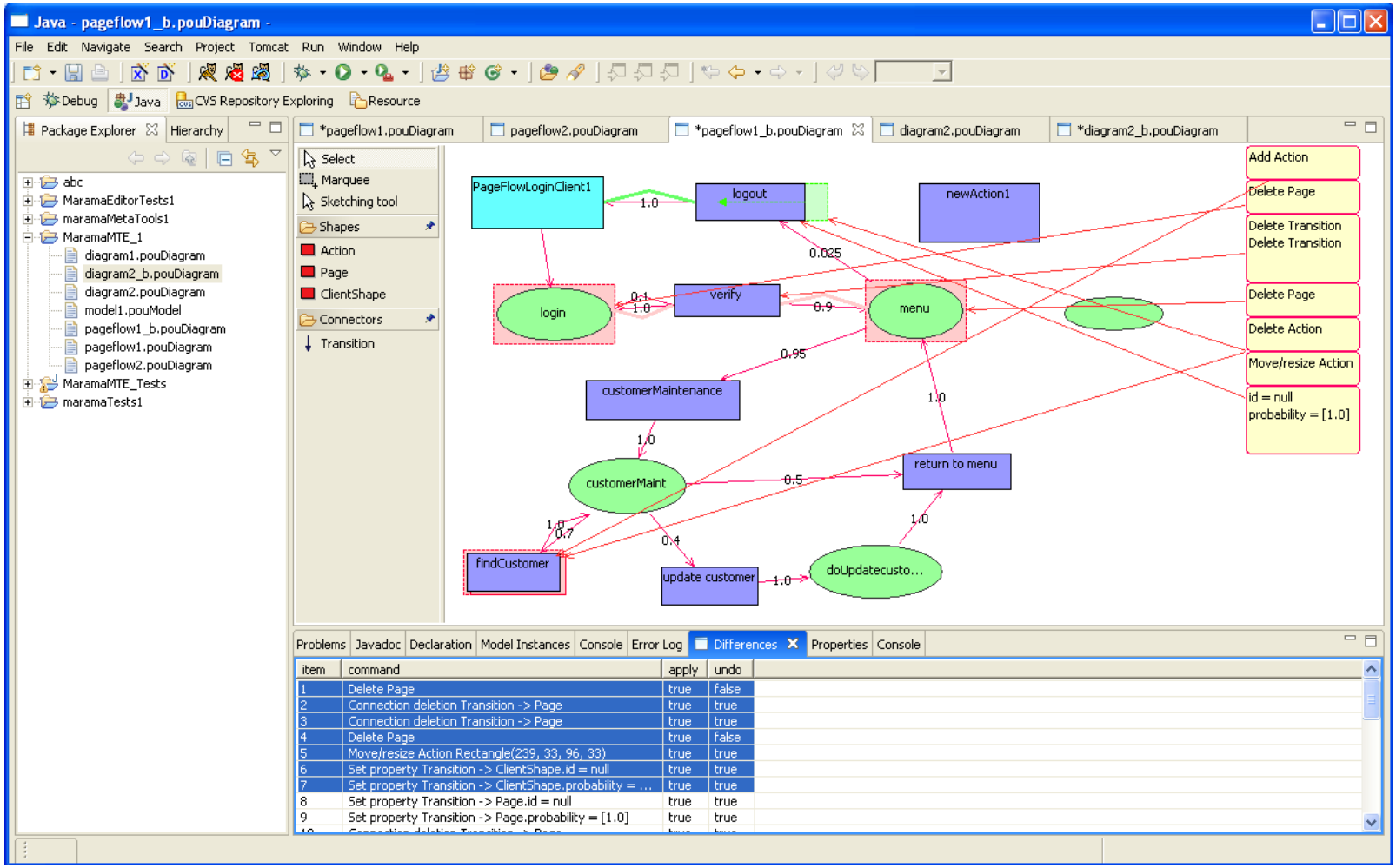
- Need to be able to specify/generate:
  - ✓ Metamodel
  - ✓ Icons and connectors
  - ✓ Views and view to model mappings
  - ✓ Behaviour
  - ✓ Model transformations
  - Tool deployment
    - Scalable
    - Sharable
    - Usable
    - Intelligent
    - …

MaramaThin, MaramaMobile

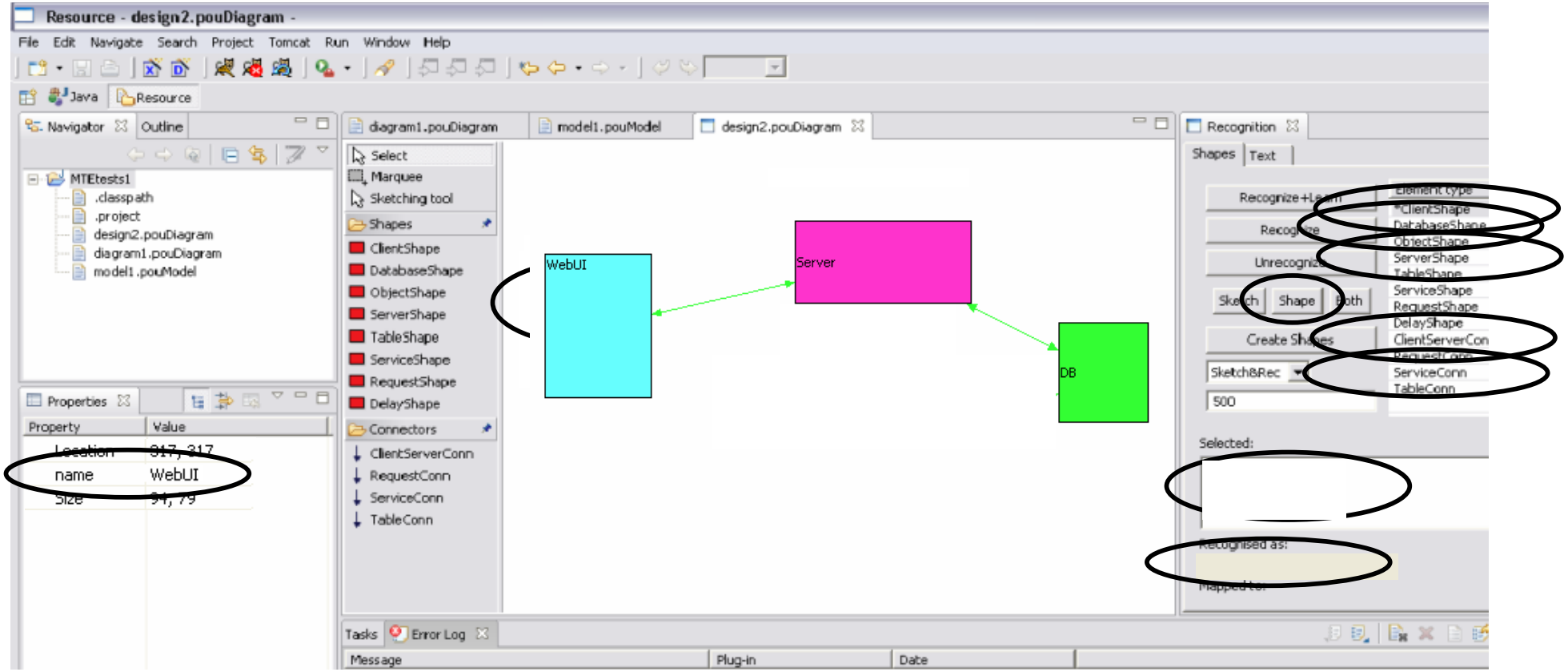




MaramaDiffer

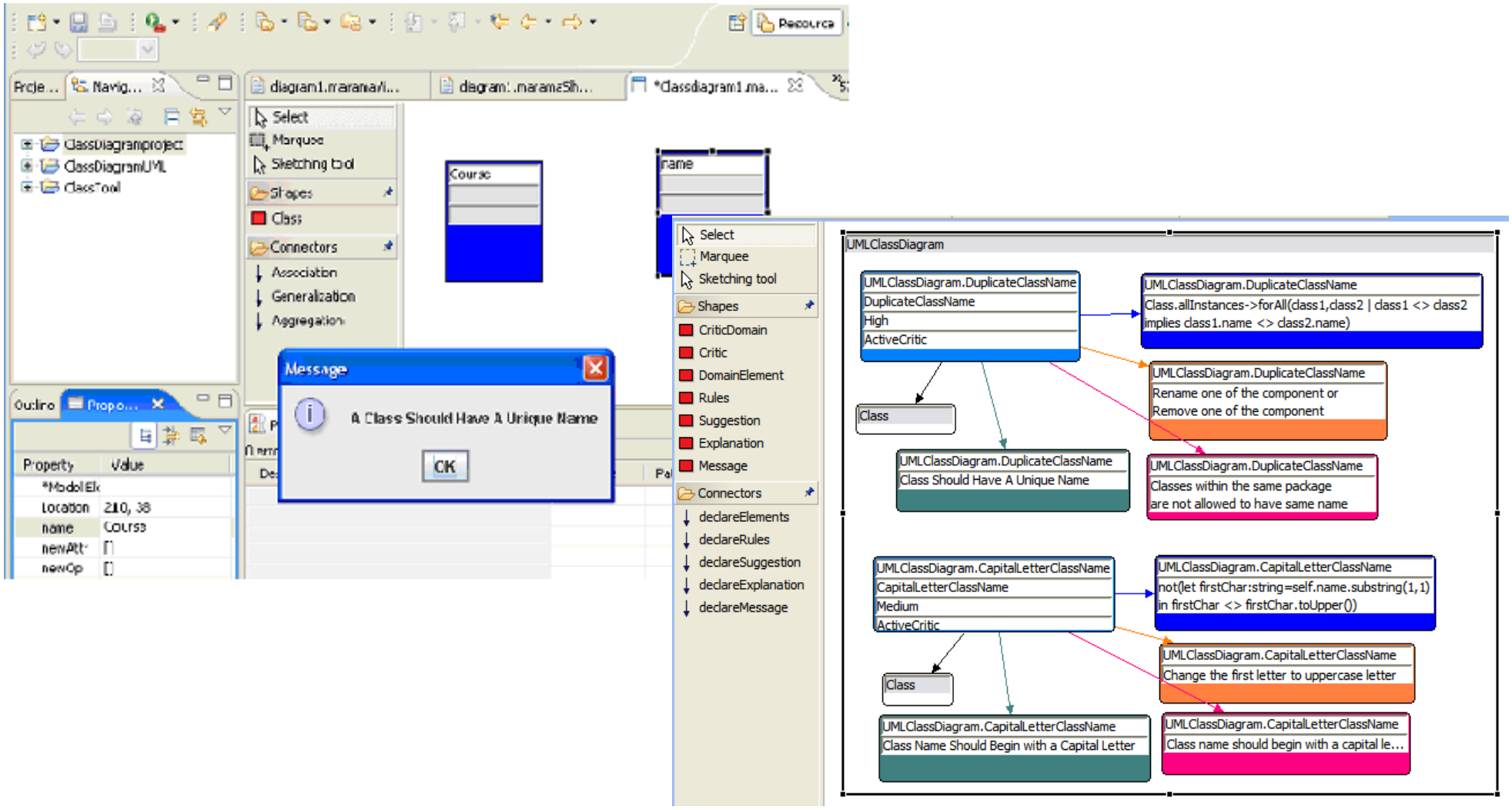


MaramaSketch



(how cool is that?!! ☺)

MaramaCritics

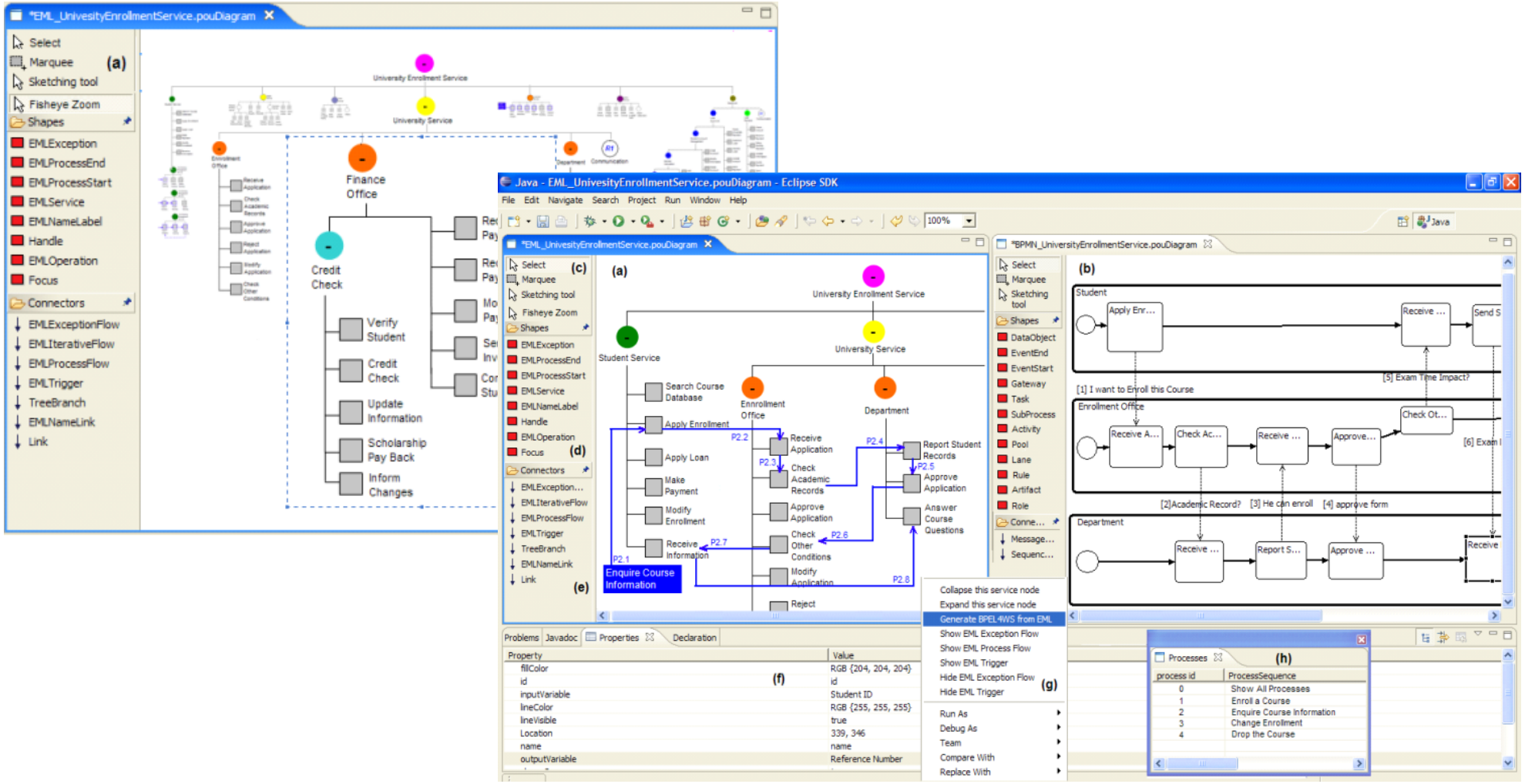


Example tools

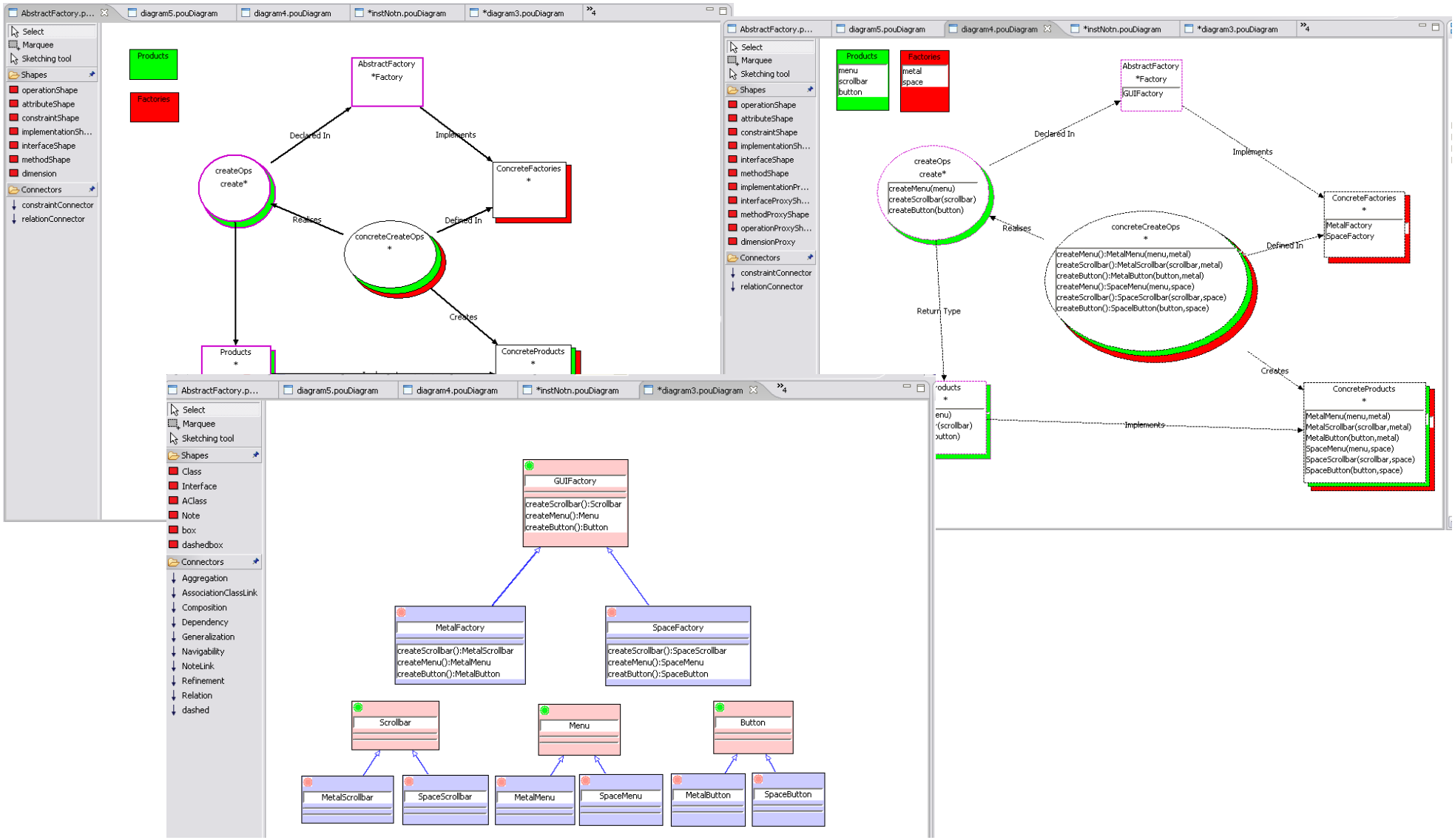
- Marama metatools themselves ☺
- MaramaMTE
- MaramaTorua

- MaramaEML – business process modeller
- MaramaDPML – design pattern tool
- Healthcare plan specification (& mobile deployment)
- Various industry rapid prototypes

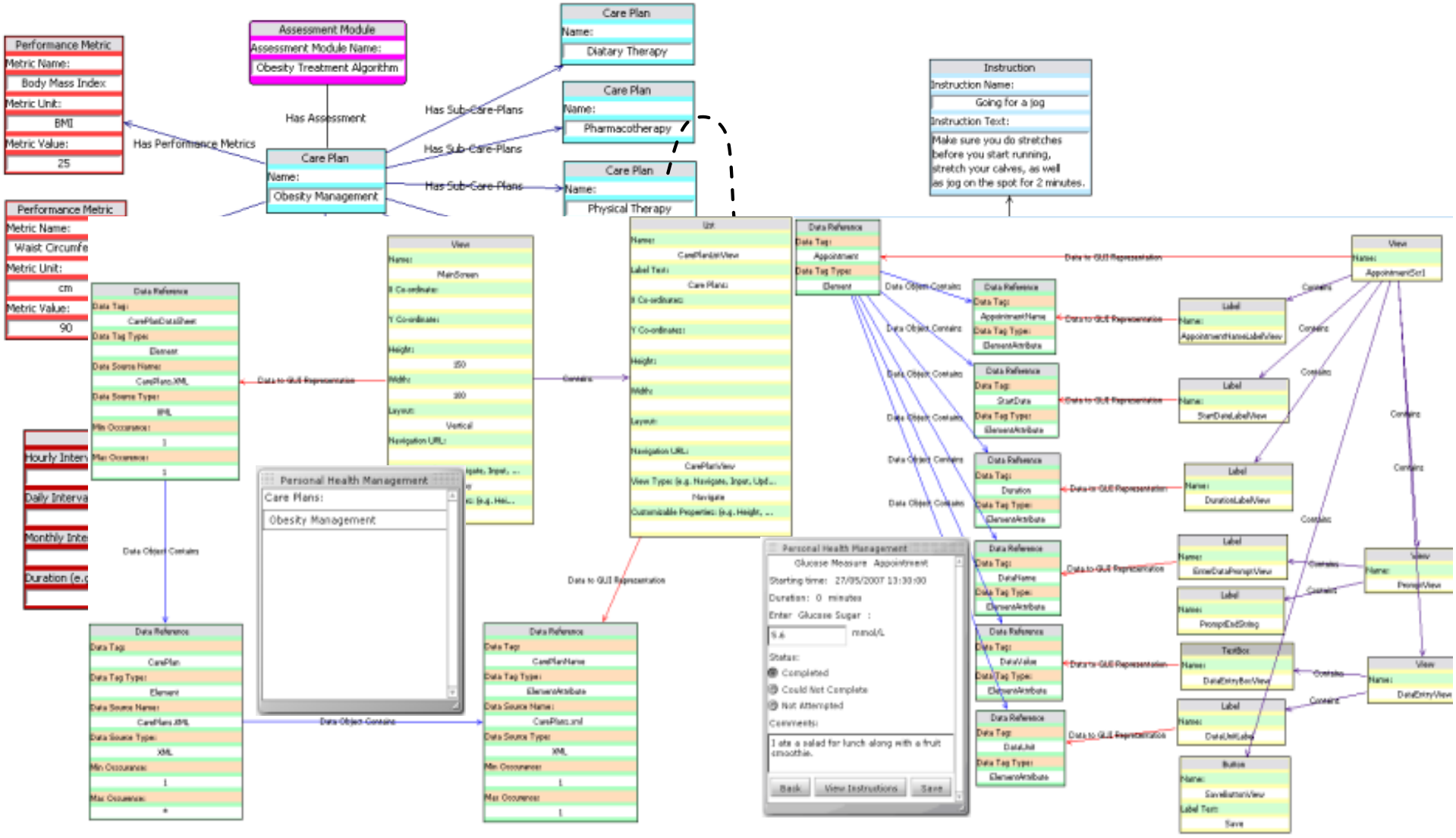
MaramaEML – Enterprise Modelling (best demo paper ASE2008)



MaramaDPML Tool – Design Patterns



VCPML & VPAM – Health Care Plans



Evaluation

- A variety of evaluation approaches
- Use of Cognitive Dimensions to:
  - Inform design and
  - Undertake lightweight evaluation
- Experience of use in designing and implementing systems
- Small group survey based usability evaluations
  - Primarily of generated tools and tool extensions
- Large group use with PG CS/SE students
  - (~130 in 2007; ~80 in 2008 participants)
  - Extended tool development exercise
  - Survey based evaluation of core meta tool
  - Results very good
    - Consistent with similar series of surveys undertaken with Pounamu

Sutcliffe's Design metadomain model

MaramaSketch

Core Marama

MaramaDiffer/CSCW

MaramaThin, Mobile
The "Visual Wiki"

MaramaCritic

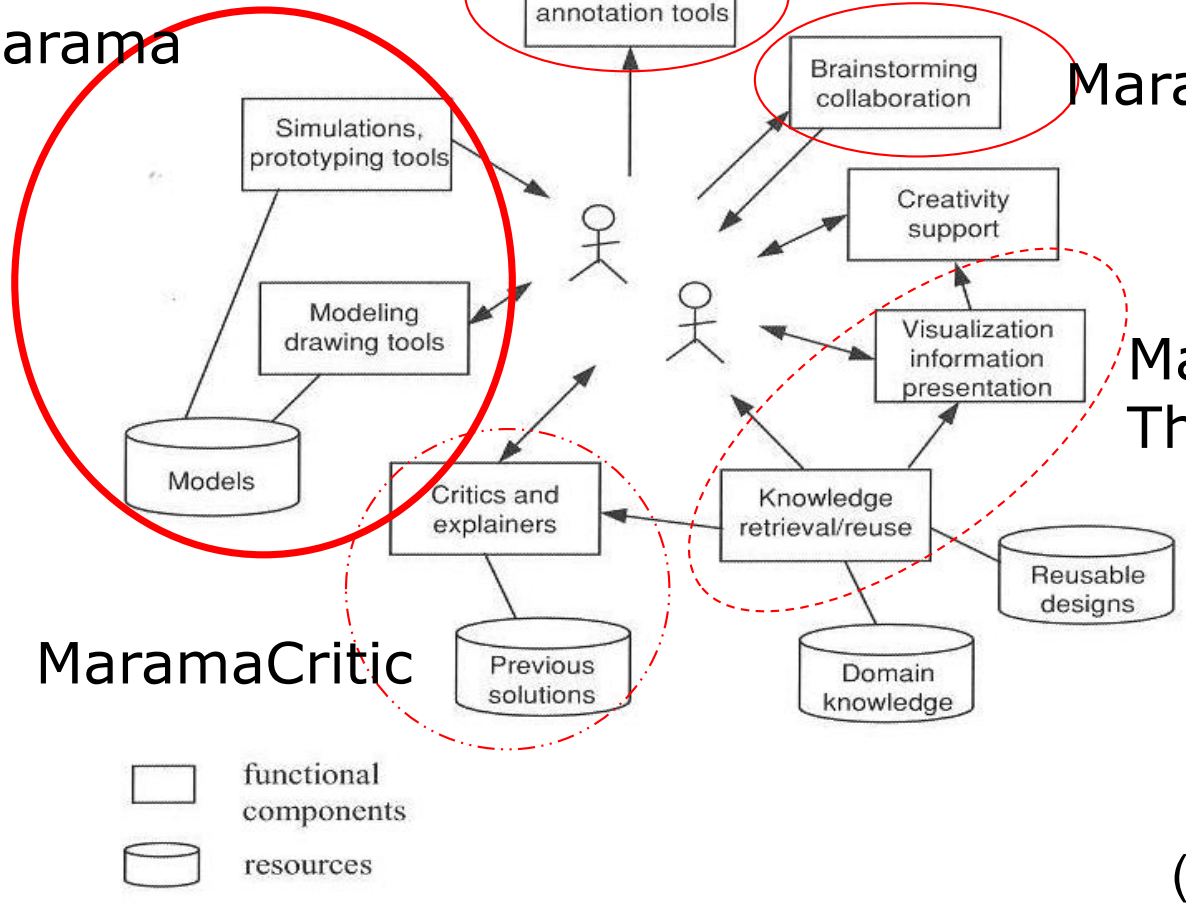


FIG. 5.13. Architecture of the Design metadomain model.

(from Sutcliffe 2002)

Where to next (Marama)?

- *Modelling* vs *visualisation* – explore existing models vs build new ones
- Domain knowledge management e.g. with EU FP7 SUDDEN and SERVE projects; NICTA (Jenny Liu)
- Commercialising and "industry hardening" with Sofismo (Swiss IT company)
- Model-driven development tools using DSVLs – MaramaMTE, VPAM good examples…
- Use to develop tools! E.g. for cloud computing (with Anna Liu @ UNSW); model-to-model mapping, tracability, consistency (with Rainbow Cai @ ANU); visualise various Eclipse projects (and itself ☺); business process modelling; health care DSVL tools; Construction IT tools (back to Kea!); …

Where to next (bigger picture)?

- Better integration with workflow/ process/ knowledge management tools e.g. the "visual wiki" (see: thinkbase.cs.auckland.ac.nz for prototype)
- Handling (well) model evolution; collaborative modelling; cross-domain modelling; model integration
- Reusing others model checking, validation etc work
- Modelling vs visualisation – integration of the concepts via multiple views
- How do we design and validate DSVLs effectively?
- "End-user" DSVLs tools - much wider applications

Summary

- Models are used in huge range of domains
- Need good tools to author, manage, evolve etc models
- Have described Marama – a meta-modelling tool builder:
  - Meta tools for multi-view modelling tool generation
  - Extensions to support:
    - Model transformation
    - Sketching
    - Tool critic authoring
    - Collaboration
- Some Applications:
    - Performance Engineering, design patterns, health care planning, model mapping and transformation, …
- BUT - we still don't know how to design good model representations (DSVLs) vs build tools for them…

Credits

- Assoc Prof Robert Amor
- Dr Rick Mugridge
- Dr Beryl Plimmer
- Dr Gerald Weber
- Dr Karen Li
- Jun Huh
- Richard Li
- Rainbow Cai
- Team @ Sofismo

https://wiki.auckland.ac.nz/display/csidst/

- Funding from New Zealand Foundation for Research Science & Technology – DS Tools & SPPI projects

References

- Li, L, Grundy, J.C., Hosking, J.G. A visual language and environment for enterprise system modelling and automation, Journal of Visual Languages and Computing, vol. 25, no. 4, Elsevier, pp. 253-277
- Ali, N.M., Hosking, J.G., Grundy, J.C., A Taxonomy and Mapping of Computer-based Critiquing Tools, IEEE Transactions on Software Engineering, vol. 39, no. 11, November 2013, pp. 1494-1520
- 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.
- Kamalrudin, M., Hosking, J.G, and Grundy, J.C. Improving requirements quality using essential use case interaction patterns, In Proceedings of the 2011 International Conference on Software engineering (ICSE 2011), Hawaii, USA, May 21-28 2011.
- Pei, Y.S., Hosking, J.G. and Grundy, J.C. Automatic Diagram Layout Support for the Marama Meta-toolset, In Proceedings of the 2011 IEEE Symposium on Visual Languages and Human-Centric Computing, Pittsburgh, USA, Sept 18-22 2011, IEEE Press.
- Li, K., Hosking, J.G., Grundy, J.C. , Ly, T. and Webb, B., Augmenting DSVL Meta-Tools with Pattern Specification, Instantiation and Reuse, 2nd International Workshop on Visual Formalisms for Patterns, Co-located with IEEE Visual Languages & Human-Centric Computing 2010, Madrid, Spain, 24th Sept 2010. Post-workshop revised version in Electronic Communications of the EASST Vol 31.
- Ali, N., Hosking, J.G., Huh, J. and Grundy, J.C. Template-based Critic Authoring for Domain-Specific Visual Language Tools, In Proceedings of the 2009 IEEE Symposium on Visual Languages and Human-Centric Computing, Cornwallis, Oregon, USA, Sept 20-24 2009, IEEE CS Press.
- Huh, J., Grundy, J.C., Hosking, J.G., Li, N., Amor, R., Integrated data mapping for a software meta-tool, In Proceedings of the 2009 Australian Software Engineering Conference, Gold Coast, Australia, April 2009, IEEE CS Press.
- Abizer Khambati, John Grundy, John Hosking, and Jim Warren, Model-driven Development of Mobile Personal Health Care Applications, In Proceedings of the 2008 IEEE/ACM International Conference on Automated Software Engineering, L'Aquilla, Italy, 15-19 September 2008, IEEE CS Press.
- Grundy, J.C., Hosking, J.G., Li, N. and Huh, J. Marama: an Eclipse meta-toolset for generating multi-view environments, Formal demonstration at the 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 2008, ACM Press.