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
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

Marama Meta-tool Application

Specification Tools
- Meta-model Designer
- Shape Designer
- View Designer
- Tool Specifications – XML documents

Eclipse IDE

Tool config. held in DOMs

Marama Plug-in (GEF Editor)

Marama Plug-in (EMF Models)

EMF OCL Plug-in (OCL Interpreter)

Adapter API

Event handler objects

Marama save files - Eclipse workspace files (XMI)

Tool specification projects (XML)
Example tool: MaramamaMTE

- Palette
- Modelling Window
- Property Sheet
- Helpers
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

Web form interaction spec

Architecture 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

```javascript
self.object.name.concat('.').concat(name)
```
Constraint violation

Critic mechanism

self.requests.size() <> 0
Visual constraints in views

- Can add some predefined layout constraints in view specification (e.g., containment)

Encloses(ObjectShape, ServiceShape)
Contains(ServiceShape, RequestShape)
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

Core Marama

MaramaSketch

MaramaDiffer/CSCW

MaramaThin, Mobile
The “Visual Wiki”

MaramaCritic

(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

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

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