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

John Hosking

Department of Computer Science
University of Auckland, New Zealand

john@cs.auckland.ac.nz
History – research platforms

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

Evolving Frameworks Pattern Language
Marama - goals

- Make modelling tool implementation easy for:
  - Experienced modeller
  - Familiar with basic modelling concepts
    - Eg EER, OCL, meta models
  - Construct basic modellers within 1 day
    - Plus time for backend code generators etc

- Leverage strength of Eclipse
  - Standalone Pounamu left us with too much to support
    - Make use of EMF, GEF, JET, etc
  - Eclipse community attractive

- Paper at ASE06 on early version of Marama
  - Used Pounamu metatools
  - Realised tools in Eclipse using Marama runtime plugin
Sutcliffe’s Design metadomain model

(from Sutcliffe 2002)
Marama basic requirements

• Need to be able to specify/generate:
  – Metamodel
  – Icons and connectors
  – Views and view to model mappings
    • View – model consistency
  – Behaviour
    • Constraints, operations
  – Model transformations
    • Backend code generation
    • Tool integration
Marama – basic architecture

Marama Meta-tool Application

Specification Tools
- Shape Designer
- Meta-model Designer
- View Designer

Tool Specifications – XML documents

(1)

Marama save files - Eclipse workspace files (XMI)

Tool projects (XML)

Eclipse IDE

Eclipse IDE resource management

Tool config. held in DOMs

Marama Plug-in (GEF Editor)

Marama Plug-in (EMF Model)

Adapter API

Event handler objects

(2)

(3)

(4)

(5)

(6)

(7)

EMF OCL Plug-in (OCL Interpreter)
Example tool: MaramaMTE
Meta model specification

• EER (KISS)
  - Entities
  - Relationships
  - 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

End user interaction spec

Architecture spec
Marama basic requirements

• Need to be able to specify/generate:
  ✓ Metamodel
  ✓ Icons and connectors
  ✓ Views and view to model mappings
    ✓ View – model consistency
  – Behaviour
    • Constraints, operations
  – Model transformations
    • Backend code generation
    • Tool integration
MaramaTatau – model level constraints

- Specification of behaviour always difficult in meta modellers
  - Initial approach – Java event handlers
  - Clumsy to write, need detailed API knowledge etc

- MaramaTatau allows constraints to be specified as OCL expns over the meta model
  - 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

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

Critic mechanism
Visual constraints in views

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

- Kaitiaki: 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
    ✓ View – model consistency
  ✓ Behaviour
    ✓ Constraints, operations
  – Model transformations
    • Backend code generation
    • Tool integration
Marama Torua – visual mapping/model transformation specn and generation

Mapping specs

Hierarchical schema

Element mappings

Generated XSLT

Mapping formula
MaramaTorua

Semi-automated mapping spec

Form based mapping spec
Installing mapping into a Marama tool
Marama basic requirements

• Need to be able to specify/generate:
  ✓ Metamodel
  ✓ Icons and connectors
  ✓ Views and view to model mappings
    ✓ View – model consistency
  ✓ Behaviour
    ✓ Constraints, operations
  ✓ Model transformations
    ✓ Backend code generation
    ✓ Tool integration
Example tools

- Marama metatools
- MaramaMTE
- MaramaTorua

- MaramaEML – business process modeller
- MaramaDPML – design pattern tool
- Healthcare plan specification (& mobile deployment)
- Various industry rapid prototypes
MaramaDPML Tool
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
  - (~ 80 participants)
  - Extended tool development exercise
  - Survey based evaluation of core meta tool
  - Results currently being processed – look good
  - Consistent with similar series of surveys undertaken with Pounamu
Sutcliffe’s Design metadomain model

(from (Sutcliffe 2002))
Where to?

Modelling->visualisation
Capturing & reusing ip
Better integration with process/KM tools: visual wiki
Plug in to UM domain knowledge development
Work ...
Use to develop tools!

(from (Sutcliffe 2002))
Summary

• Have described Marama
  – Meta tool for multi-view modelling tool
  – Extensions to support
    • Model transformation
    • Sketching
    • Tool critic authoring
    • Collaboration
  – Applications
    • Performance Engineering
    • Model mapping
    • etc
Credits

- Prof John Grundy
- Assoc Prof Robert Amor
- Dr Beryl Plimmer
- Dr Gerald Weber
- Karen Liu
- Jun Huh
- Richard Li
- Rainbow Cai

- Funding from New Zealand Foundation for Research Science & Technology