Marama: an Eclipse Meta-toolset for Generating Multi-view Environments

John Grundy, John Hosking, Jun Huh, Karen Na-Liu Li
Departments of Computer Science and Electrical and Computer Engineering,
University of Auckland,
Private Bag 92019, Auckland, New Zealand
+64-9-3737-599 ext 88761
{john-g, john, jhuh003, karen}@cs.auckland.ac.nz

ABSTRACT
We describe the Marama suite of meta-tools. This Eclipse-based toolset permits rapid specification of notational elements, meta-models, view editors and view-model mappings. It has a novel set of behavioural specification tools for both visual and model level behaviours. An integrated mapping tool provides model transformation and code generation support. The toolset has been applied to several significant application development tasks and has undergone a variety of evaluations.

Categories and Subject Descriptors
D.2.2 [Design Tools and Techniques]: Computer-aided software engineering (CASE)

D.2.6 [Programming Environments]: Graphical environments

General Terms
Design, Human Factors

Keywords
Meta-tools, domain-specific visual languages, code generation, model-driven engineering

1. INTRODUCTION
Meta-tools are sets of software tools that support rapid specification and implementation of other software tools. They are often used to develop domain specific visual languages for tasks such as configuration (e.g. of frameworks and software product lines) and end user oriented activity specification. They are one element in a broader Model Driven Engineering approach to software development.

A wide range of meta-tools have been developed, including MetaEdit+ [6], IPSEN [7], MetaMOOSE [5], DiaGen [15], GME [8] and our own Pounamu [16]. Much recent attention has focussed on the Eclipse GMF [4] and Microsoft's DSL Tools toolsets [14]. Problems with these include difficulty dealing with behaviour specification (all), model transformation (all, but GMF has projects to address this), a compile edit cycle (all except Pounamu), and non visual specification (some parts of DSL Tools and GMF).

ICSE'08, May 10–18, 2008, Leipzig, Germany.
ACM 978-1-60558-079-1/08/05.

In this paper we describe Marama, an Eclipse based meta-toolset that provides a more accessible approach to domain specific visual language specification than do other meta-tools. We begin by describing our high level approach before examining Marama's architecture and core toolset. We then briefly describe a tool developed using Marama, our approach to evaluation and current research directions.

2. OUR APPROACH
Our goal for the Marama toolset was to make the implementation of diagrammatic modelling/MDE tools easy for experienced modellers, familiar with basic modelling concepts, such as Extended Entity Relationship (EER) models, OCL, and the notion of meta-models. The aim was to permit such users to construct basic visual modellers within 1 day, with extra time for specification of backend code generators and complex editing or behavioural constraints in a seamlessly integrated framework managing both interface and semantic consistencies. Thus we wanted a tightly integrated meta-tool environment with quick and easy to use modelling tools. Specific requirements for the toolset include support for:

- Icon and connector and/or containment based visual metaphors
- Specification/generation of: the tool meta-model; icons and connectors (including containment based); views, view editors and view-model mappings and consistency mechanisms
- Behaviour, including model and view level constraints and operations
- Model transformations, including code generation
- Tool integration mechanisms

In addition we aimed to preserve Pounamu's liveness characteristic, i.e. changes made to tool specifications are reflected immediately in the realised models.

3. ARCHITECTURE AND TOOLSET
Figure 1 shows a high level architecture diagram for Marama. Marama is realised as a set of Eclipse plugins. Tools are specified using shape, meta-model and view tools (1, 2) and then implemented by interpretation of the specifications using a set of plug-ins that leverage the GEF and EMF frameworks (3-7). The meta-tools are themselves implemented using these plug-ins.

Figure 2 shows the Marama meta-tools in use. The meta-model tool (left) uses, for simplicity for our target end-users, an EER

representation, supplemented by visually annotated and dynamically interpreted OCL constraints (specifying attribute calculations, invariants, and cardinalities, which, although designed for UML, are effectively applicable to EER models), specified using a novel editor that mitigates many of the usual issues associated with OCL use [11], such as the lack of visual connections of the OCL constraints with their contextual model specification and the lack of semantic checks of OCL at design time. The visual shape designer (centre) allows rapid specification of composite icons and connectors. The view designer (right) specifies which visual elements are in a view type, their relationship to underlying model elements (including attribute mappings) and additional constraints (such as various composite icon containment mechanisms).

Behaviour is specified in several ways. The model and view level constraint mechanisms described above provide a simple declarative approach to behaviour specification. For view level behaviour, such as alignment constraints, or auto-construction of connectors or icons, an event/data flow-based visual specification mechanism (Kaitiaki), originally developed for Pounamu [10], is available.

Figure 3 (top) shows the specification of a Kaitiaki alignment constraint. This is triggered by a shapeAdded event (top). The filter immediately below checks whether the shape is a TableShape. In this case, the new TableShape is vertically aligned with the other TableShapes in the diagram (accessed via the right hand flow). The effect of the alignment is shown in the bottom figure. The combination of these mechanisms allows most required modelling behaviours to be implemented simply and rapidly. For unusual cases, escape to Java event handler code, with API manipulation of the tool data, is possible. Such code can be packaged into a reusable form for use as primitives in the other behavioural mechanisms.

Model transformations are managed using our MaramaTorua mapping specification tool. This tool, originally developed as a standalone schema mapping specification tool, has been adapted for and integrated with our Marama meta-tools. The tool uses a tree based metaphor to describe both the original tool model schema and the target schema to transform the tool to (e.g. for code transformation or for generating models for use by another tool type). Figure 4 shows a partial specification for a mapping from a custom process business modelling tool to BPEL code. Mappings can be hierarchically decomposed (a, b) and the schema are themselves hierarchically arranged (c). Individual element mappings are expressed using constraint formulae (d) and XSLT code is generated to implement the mapping (e). The tool also incorporates heuristic assistance to suggest potential mappings in the case of large schema. The generated XSLT code can be incorporated back into the generated tool in the form of a menu triggered event handler.

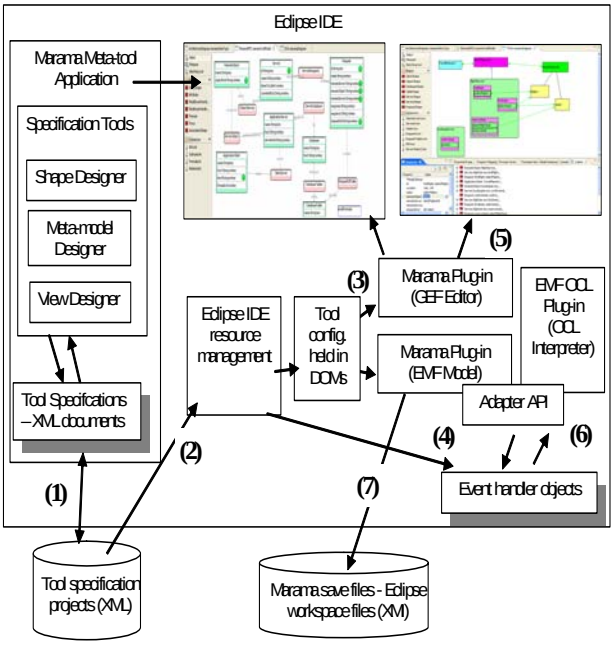

Figure 1: Marama architecture

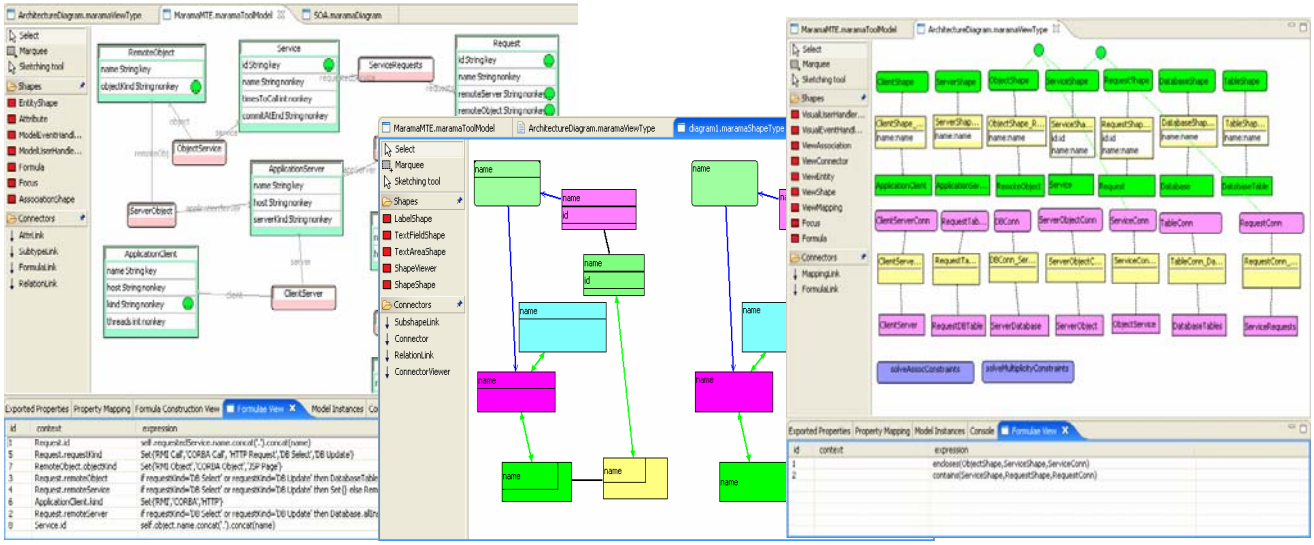

Figure 2: Marama meta-tools in use: meta-model (left) shape designer (centre) and view definer (right)

With the exception of compilation of event handlers (an area we are still addressing), we have retained Pounamu's liveness level across the core Marama toolset. If a tool specification is modified, closing and reopening any diagram constructed using the tool will cause the model to be updated as per the new specification (e.g. icon/connector format changes, additional attributes, additional icons/connectors in the tool palette).

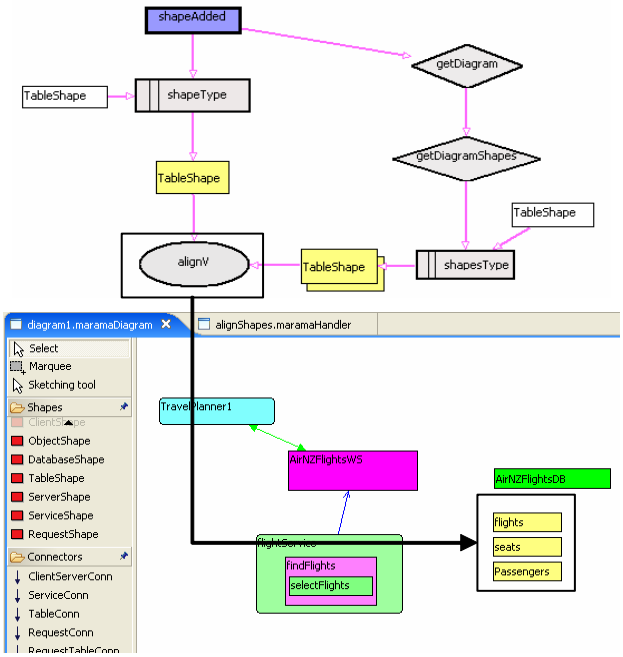

Figure 3: Kaitiaki visual alignment constraint

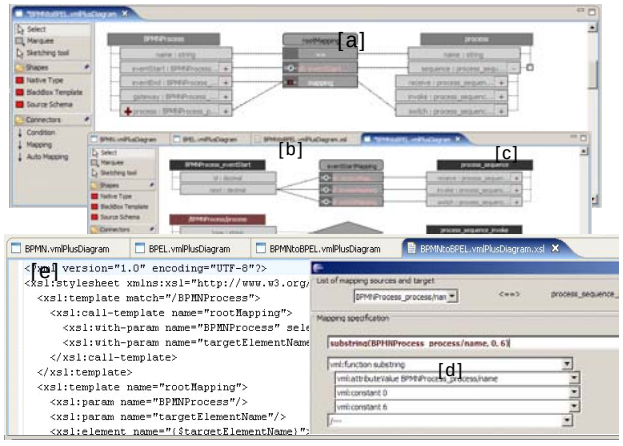

Figure 4: MaramaTorua mapping specification

4. EXAMPLE TOOLS

We have used Marama to construct a wide variety of modelling tools. The Marama meta-tools themselves, including MaramaTorua, are implemented using Marama in a bootstrapped manner and were the first substantial exemplars developed using Marama.

Figure 5 shows two screen dumps from MaramaMTE, a performance engineering tool implemented using Marama, in use. This is a reimplementation (with refactoring) of the ArgoMTE tool we had previously developed [2]. At top a software architecture view describes the high level architecture of a three tier travel planner system. This view permits the various architectural components and their properties (e.g. middleware implementation to be used) in a high level visual manner. At bottom, a form chart view is used to specify a probabilistic model for user interaction with the proposed system. MaramaMTE provides the ability to generate, from these views, a testbed for the proposed system that can be deployed on real hardware and exercised, according to the stochastic formchart models, to obtain an accurate estimate of the system if it were fully constructed [3].

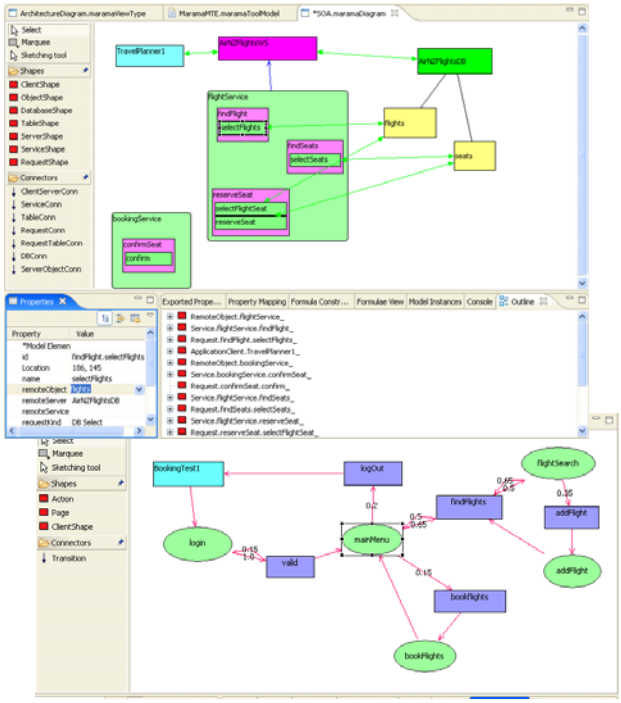

Figure 5: MaramaMTE performance modeller

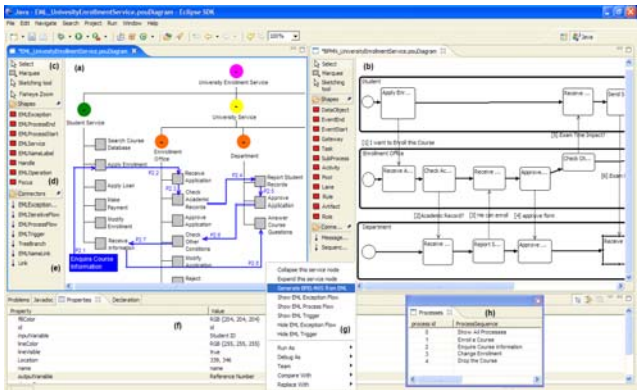

Figure 6: MaramaEML modeller in use

Figure 6 shows a screen dump of the MaramaEML tool in use. This tool permits the specification of business processes using a high level notation, the Enterprise Modelling Language (EML) [9]. This provides a tree based metaphor (right) for describing hierarchical business services, together with overlaid control flow oriented process modelling descriptions. The tool also supports

modelling of individual processes using the BPMN notation (right) and generation of BPEL code that can be deployed on a BPEL engine to execute the specified processes. To provide scalability for large service trees, a fisheye view can be used to focus on specific components of the service tree.

In addition to these examples, other applications we have developed using Marama include tools for:

- Modelling health care plans, with the ability to generate plan descriptions that are deployable on a PDA and that provide reminders and other assistance to patients following the plan.
- Specifying design patterns, their instantiation into a design, and realisation as a UML class diagram [12].
- Modelling goals, goal decomposition and their realisation as coordinated distributed processes for use in an SME based process planning application.

5. EVALUATION

In addition to demonstrating Marama's efficacy through its use in development of substantial applications, we have used a variety of more formal evaluation approaches. We should, however, point out that evaluation of a substantial toolset such as Marama is not a straightforward task. Typically usability study approaches, for example, are only able to focus on a relatively constrained subset of the features available. Accordingly we have adopted a set of overlapping evaluation approaches to prove the utility of our toolset and environment. These include use of:

- Cognitive Dimensions to both inform design and undertake lightweight evaluation.
- Formative small group survey plus open ended interview based usability evaluations. These are primarily of generated tools (hence are an indirect measure of the efficacy of Marama) but have also been applied to individual Marama tool extensions.
- Large group use with more than 120 participants applying the toolset to a tool design and development exercise of the participants' choosing with a survey based approach to understanding the end user experience.

The results have been uniformly positive [3][9][10][11], with the exception of the usual issues of stability concerns that arise with leading edge proof of concept software applications, and are consistent with a similar series of surveys we undertook as part of the Pounamu meta-tool development. The toolset has now been released in an open source form [13] and is being taken up by a number of research groups and industrial partners for software tool prototyping.

6. CONCLUSIONS AND FUTURE WORK

We have described Marama, an Eclipse based meta-toolset for constructing multi-view multi notation visual modelling tools. This builds on experience gained in the previous Pounamu project. Eclipse has been applied to a number of substantial software tool development projects and has been evaluated using a number of evaluation approaches.

In current work, we are extending the toolset to include a number of additional features. These include: a generic critic authoring tool [1]; a wider range of user interface elements (including video) and layout support; enhanced interaction capabilities, including voice interaction; and thin and mobile client support.

7. REFERENCES

[1] Ali, N.M. A generic critic authoring tool, Proc VLHCC'07, IEEE CS Press, 260-2

[2] Cai, Y., Grundy, J.C. and Hosking, J.G. Experiences Integrating and Scaling a Performance Test Bed Generator with an Open Source CASE Tool, In Proceedings of the 2004 IEEE International Conference on Automated Software Engineering, Linz, Austria, September 20-24, IEEE CS Press, pp. 36-45.

[3] Draheim, D., Grundy, J.C., Hosking, J.G., Lutteroth, C. and Weber, G. Realistic Load Testing of Web Applications, In Proceedings of the 10th European Conference on Software Maintenance and Re-engineering, Berlin, 22-24 March 2006.

[4] Eclipse Graphical Modelling Framework http://www.eclipse.org/gmf/

[5] Ferguson R, Parrington N, Dunne P, Archibald J, Thompson J, MetaMOOSE-an object-oriented framework for the construction of CASE tools, Proc. Int Symp on Constructing Soft. Eng. Tools, LA, May 1999

[6] Kelly, S., Lyytinen, K., and Rossi, M., Meta Edit+: A Fully configurable Multi-User and Multi-Tool CASE Environment, Proc. of CAiSE'96, LNCS 1080, 1996.

[7] Klein, P. and Schürr, A. Constructing SDEs with the IPSEN Meta Environment, Proc. 8th Conf. on Software Engineering Environments,1997, pp. 2-10.

[8] Ledeczi A., Bakay A., Maroti M., Volgyesi P., Nordstrom G., Sprinkle J., Karsai G.: Composing Domain-Specific Design Environments, Computer, 44-51, Nov, 2001.

[9] Li, L. Hosking, J.G. and Grundy, J.C. Visual Modelling of Complex Business Processes with Trees, Overlays and Distortion-Based Displays, Proc VLHCC'07, IEEE CS Press, 137-144.

[10] Liu, N., Grundy, J.C. and Hosking, J.G. A visual language and environment for specifying user interface event handling in design tools, Proc AUIC 2007, Ballarat, Australia, CRPIT Press

[11] Liu, N., Hosking, J.G. and Grundy, J.C., MaramaTatau: extending a domain specific visual language meta-tool with a declarative constraint mechanism, Proc VLHCC'07, IEEE CS Press, 95-103.

[12] Maplesden, D., Hosking, J.G. and Grundy, J.C. A Visual Language for Design Pattern Modelling and Instantiation, Chapter 2 in Design Patterns Formalization Techniques, Toufik Taibi (Ed), Idea Group Inc., Hershey, USA, March 2007

[13] Marama https://wiki.auckland.ac.nz/display/csidst/Welcome

[14] Microsoft Domain Specific Language Tools http://msdn.microsoft.com/vstudio/DSLTools/

[15] Minas, M. and Viehstaedt, G. DiaGen: A Generator for Diagram Editors Providing Direct Manipulation and Execution of Diagrams, Proc. VL '95, 203-210 Sept. 1995

[16] 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.