Template-based Critic Authoring for Domain-Specific Visual Language Tools

Norhayati Mohd.Ali, John Hosking, Jun Huh and John Grundy
University of Auckland, New Zealand.

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

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

Outline

- Introduction
- Motivation
- Our Approach
- Critic Authoring Template
- Critic Authoring Process-Example
- Implementation
- Discussions
- Conclusion & Future Work
- Q&A

Introduction

- Many studies have reported that critic tools provide an efficient mechanism for feedbacks.
- However, *critic authoring* continues to be a challenge
  - Little agreement on how critics should be specified
  - Little work on tools to author and realize critics

Introduction

- Our research➔ *Marama Critic Definer*, a critic support-based extension to our Marama metatool.
- Critic authoring extension allow tool designer (not intended to be a skilled programmer) to define his/her own critics and feedback mechanisms specific to their tool.

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

Motivation

- Related work on critics:

Critic tool	Critic's realization approach
LISP-Critic [Fischer]	Rule-based
ArgoUML [Robbins & Redmiles]	Java classes
ABCDE-Critic [Souza et.al]	First-order production system notation
IDEA [Bergenti & Poggi]	Knowledge-based and Prolog
Java Critiquer [Qiu&Riesbeck]	Pattern matching approach

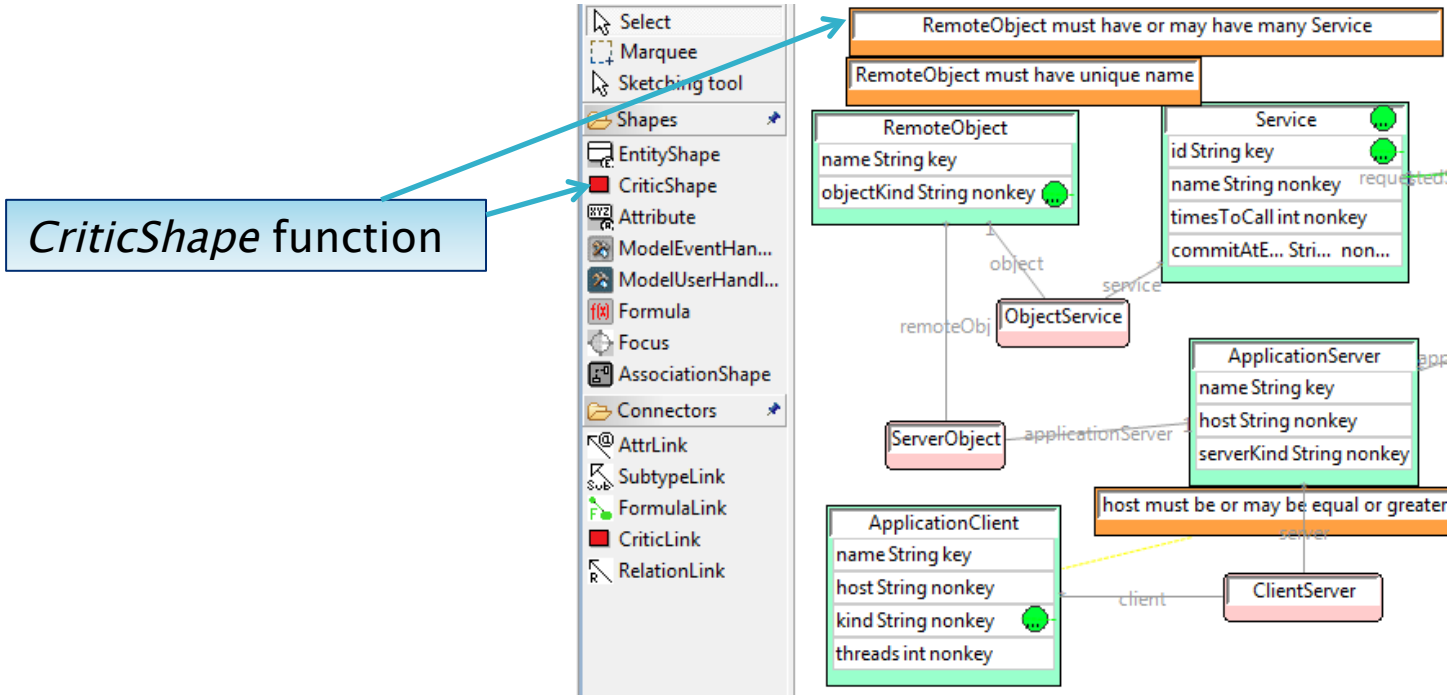
- Those approaches:
  - Require deep understanding of the tool platform
  - Customization of critics would not be easy

Motivation

- Our *Marama Critic* tool similar to tools such as MetaEdit+ and other metamodelling tools
- We imitate the metamodelling approach but our focus is on critics authoring inspired by the critic tools
  - Less discussion on issues of critic authoring, i.e allow end-user and tool designer to customize critic rules

Motivation

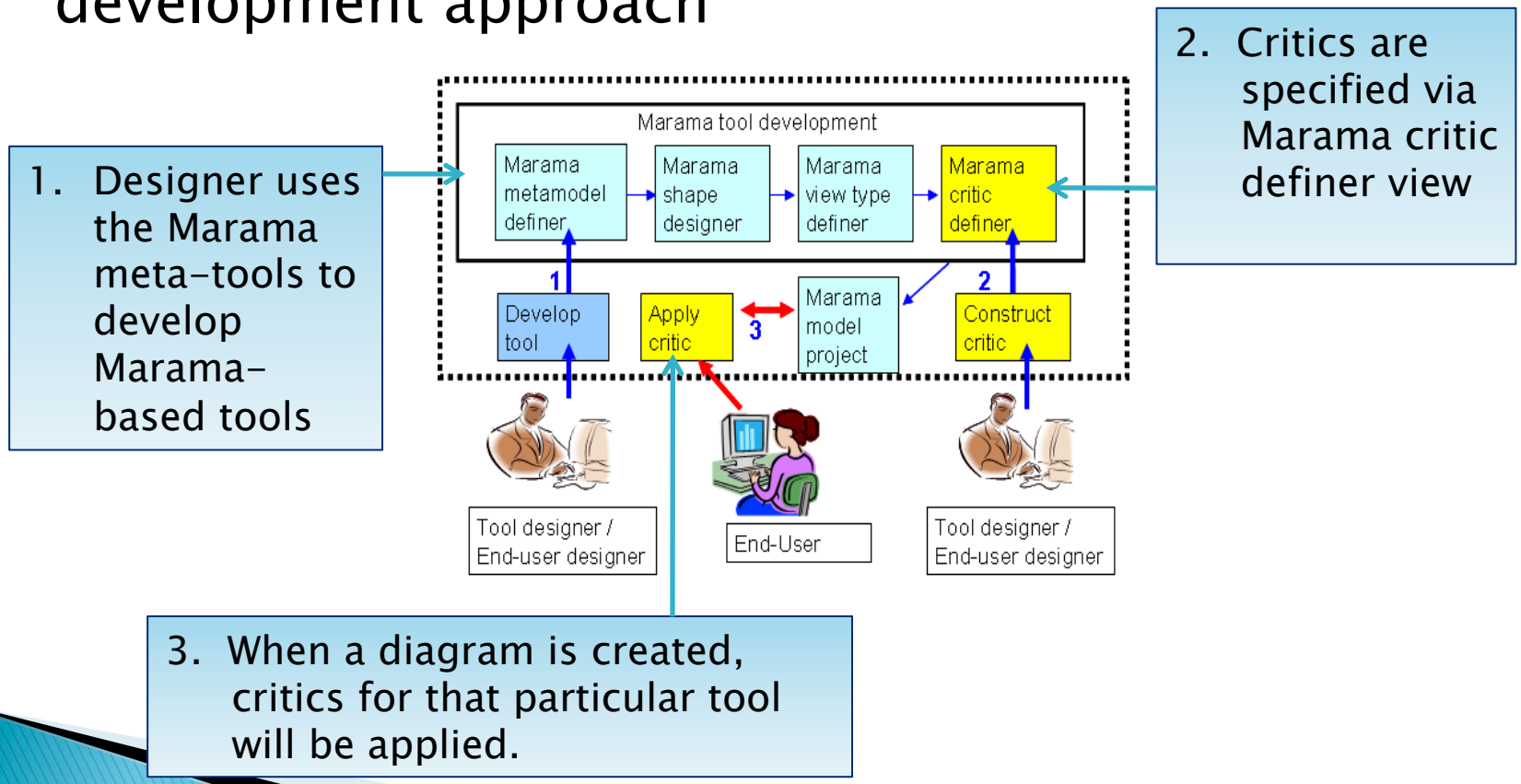
- Motivating Example: MaramaMTE architecture design tool with *Critic* function (initial attempt)



MaramaMTE critics defined using the Marama meta-model editor

Our Approach

- To address some of the problems in the initial attempt, we propose a new Marama Critic development approach



Critic Authoring Template

- The critic-authoring task is adapting the concept of 'business rule templates' [Loucopoulus & Wan Kadir,2008]
- The rule templates are formal sentence patterns that allow the expression of business rules
- Why we choose ?
  - Match properties between metamodel descriptions and 'business rule templates'
  - Support end-users (with limited programming capability) to define and author critics

Critic Authoring Template

- Currently the templates consist of two types: constraint templates and action assertion templates
- Constraint templates:
  - Attribute constraint = specify uniqueness, optionality, and value check of an entity's attributes
  - Relationship constraint = asserts the relationship types, cardinality and roles of each entity in a relationship

Critic Authoring Template

- Constraint and Action Assertion Templates [13]

Constraint	<entity> must have | may have a [unique] <attributeTerm> <attributeTerm1>must be |may be <relationalOperator><value>| <attributeTerm2> [<cardinality>]<entity1> is a/an <role> of [<cardinality>]<entity2> [<cardinality>]<entity1> is associated with [<cardinality>] <entity2> <entity1> must have |may have [<cardinality>]<entity2> <entity1> is a/an <entity2>
Action Assertion	When <event> [If <condition>] then <action>

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

Critic Authoring Template

- We employ the templates for domain-specific tool specification specifically for visual critic authoring.
- Advantages to the critic authors/tool designers:
  - Use formal language definition to define sentence patterns
  - Use of structured natural language
  - Guidance to construct the rules
  - Support the association between tool specification elements and rule statements

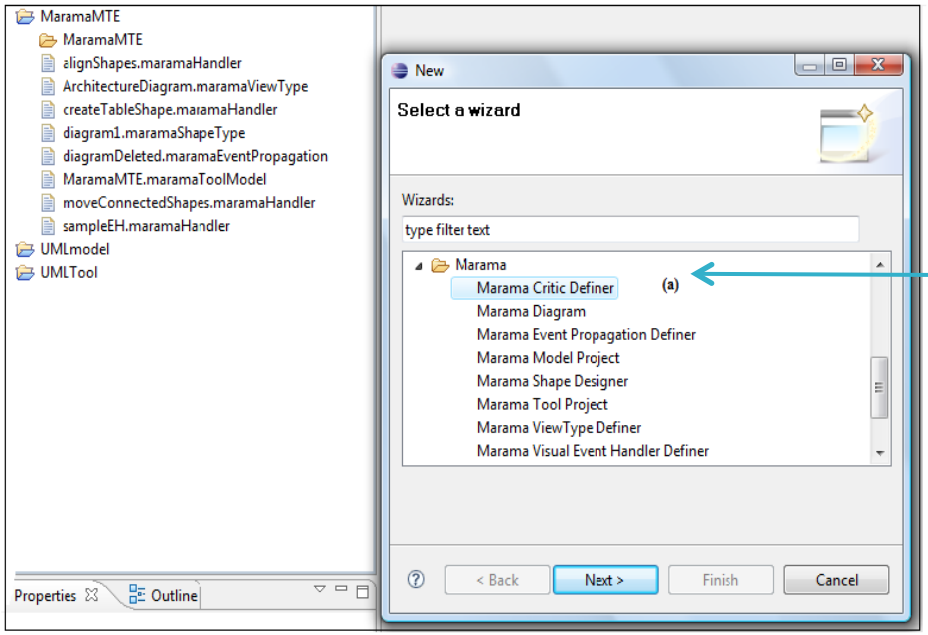
Marama Critic Authoring Process-Example

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

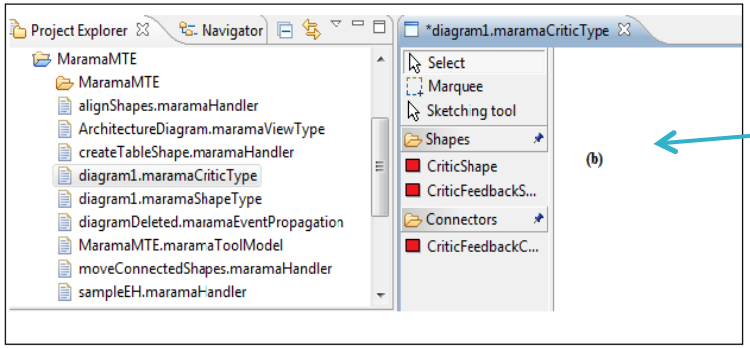
- We illustrate the critic authoring process via three major components:
  - 1) Marama Critic Definer editor
  - 2) Critic Construction editor
  - 3) Critic Feedback editor

- A simplified UML class diagram (MaramaCD) tool- as an example for the critic authoring process.
  - Metamodel for MaramaCD tool
  - Sample of critics and feedbacks
  - Critic and feedback execution

Marama Critic Definer Editor



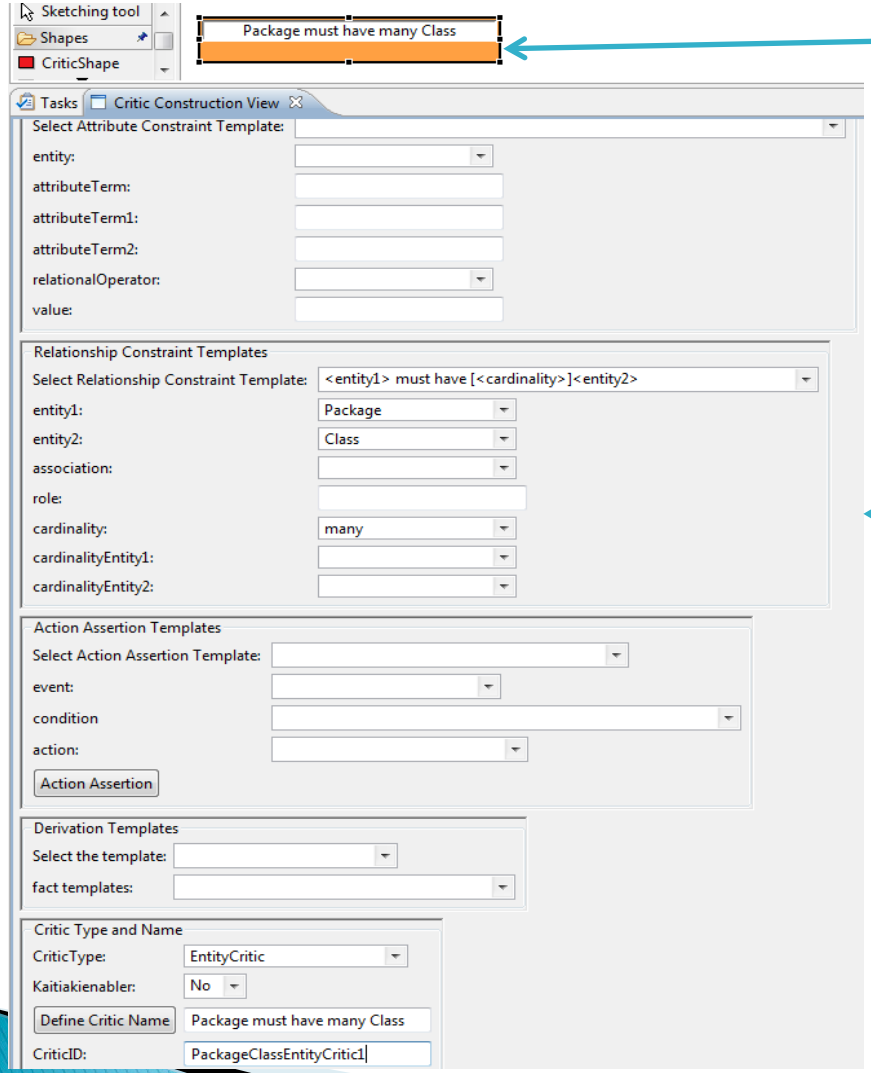
Marama Critic Definer option



Visual critic definer editor

Back

Critic construction editor

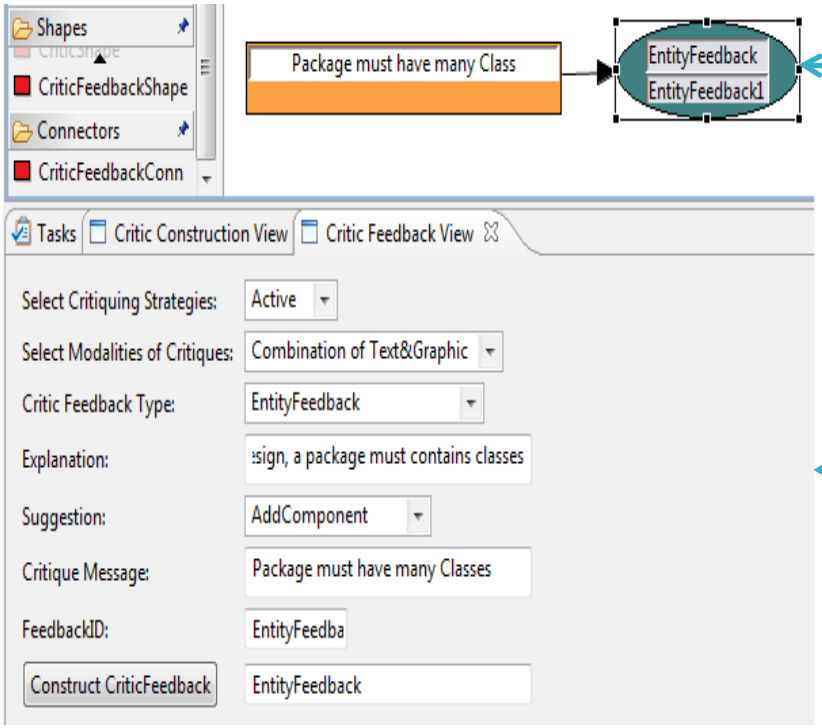


CriticShape function

'Critic Construction View' interface associated with CriticShape function

Back

Critic Feedback editor

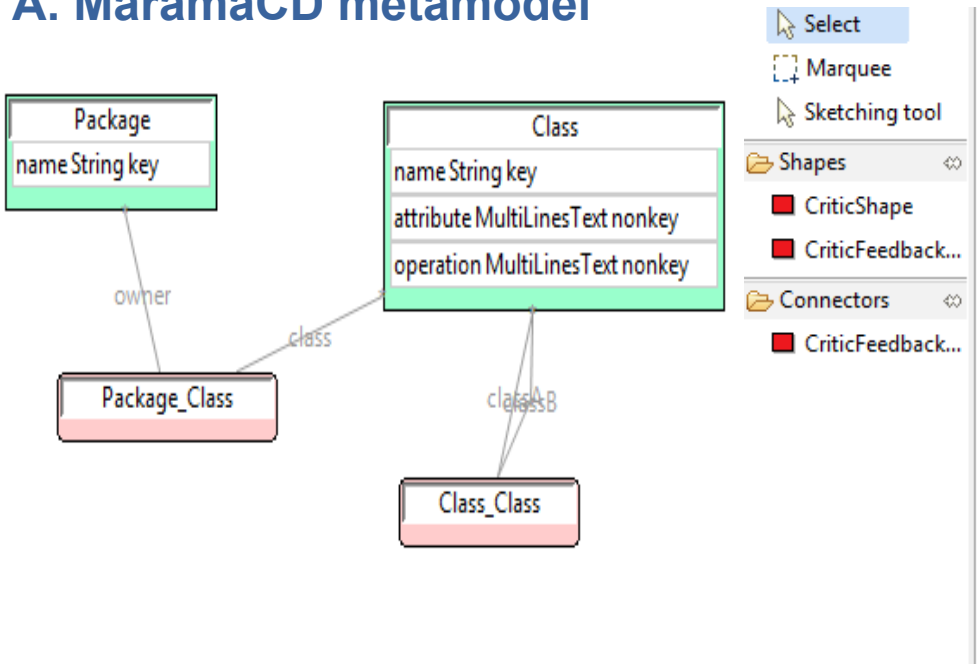


CriticFeedback function

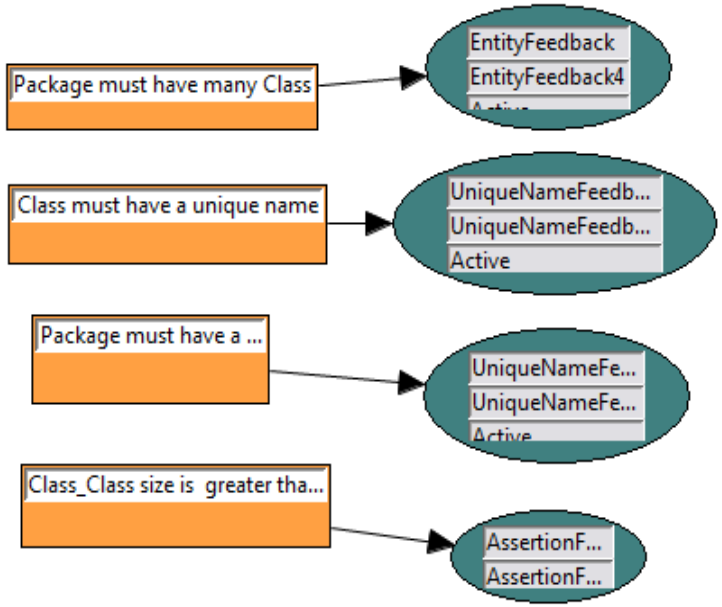
'CriticFeedbackView' interface associated with *CriticFeedback* function

Back

A. MaramaCD metamodel



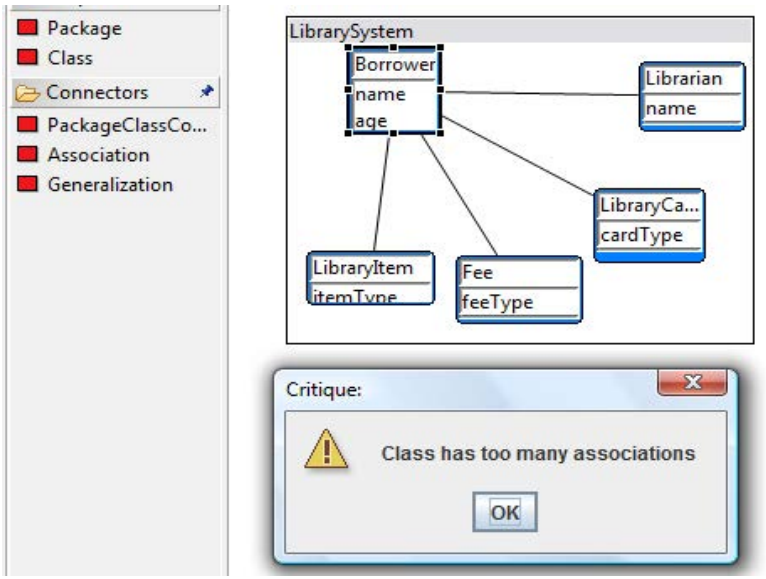
B. Critic Definer editor



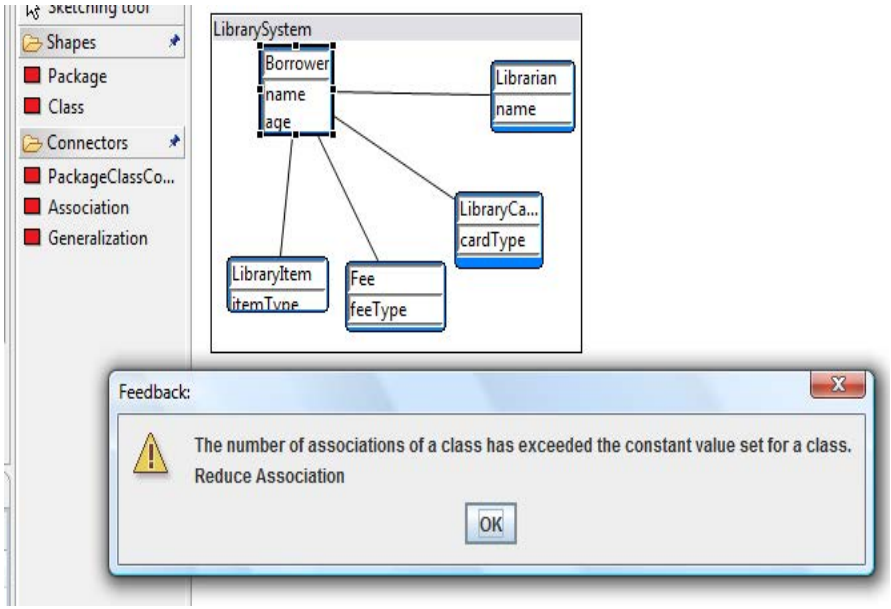
Tool element	Critic rule phrase	Critic template	Type	Feedback
Class	A class must have a unique name	<entity> must have | may have a [unique] <attributeTerm>	Attribute constraint	Remove or rename one of the components
Package	Package must have many classes	<entity1> must have | may have [<cardinality>] <entity2>	Relationship constraint	Add the component
Class	When a class has too many associations then reduce the association	When <event> [if <condition>] then <action>	Action assertion	Reduce the association

C. Example of critics and feedbacks for MaramaCD tool

Critic and Feedback execution:



1) Critic executed at diagram level



2) Critic's feedback with fix action

Implementation

- Create three new editors: *Marama Critic Definer*, *Critic Construction view* and *Critic Feedback view*
- Critics and feedbacks are stored in a repository (XML format)
- A code generator template is used to implement the critic, which is then instantiated into the tool when it is executed

Discussions

- The three new editors contributes several benefits:
  - Provides a simple way to define *critic specifications* and *critic feedback specifications*
  - Novice designer may easily construct the critics and feedbacks
  - The process of customizing critics and their feedback is much easier

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

Discussions

- Main limitations:
  - Limited set of critic authoring and feedback templates and actions
  - Constructing new critic condition and feedback templates is not fully formed yet

- Apply appropriate abstractions:
  - A high-level visual overview of the critics
  - Highly user accessible form-based rule template interfaces
  - Extensibility options for experiences tool users

Discussions

- Evaluation:
  - Used Cognitive Dimensions
  - Reducing *viscosity* and *hard mental operations*
  - Good *closeness of mapping*, low *error proneness*
  - Provide a balance of abstractions

Conclusion & Future Work

- *Marama Critic Definer* provides support to end-user and tool designers for critic authoring and configuration tasks.

- Plans for future work include:
  - Provide a better template specification tool
  - Provide visualization of dependencies between critic and model elements
  - Develops a hierarchical critics to provide more powerful critic reasoning mechanism
  - More extensive case study
  - Conduct a larger end-user evaluation

Thank You…

- BuildIT Travel award, NZ – travel funding support
- Postgraduate Research Student Support account – travel funding support
- Ministry of Malaysia Higher Education & Universiti Putra Malaysia – scholarship support
- VL/HCC 2009 for this opportunity to present

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

Q & A

References

- [1]. Ali, N.M., Hosking, J., Huh, J. And Grundy, J. Critic Authoring Templates for Specifying Domain-Specific Visual Language Tools, Proc. ASWEC09, Gold Coast, Australia, April 14-17 2009.
- [2]. ArgoUML, http://argouml.tigris.org/
- [3]. Barone, R., & Cheng, P. C.-H. (2004). Representations for problem solving: on the benefits of integrated structure. Proc 8th Intnl Conf on Information Visualisation (pp. 575-580). Los Alamitos, CA: IEEE.
- [4]. Bergenti, F. and Poggi. A. Improving UML designs using automatic design pattern detection, Proc SEKE, 2000, pp. 336-343.
- [5]. Costagliola, G., Denfemia, V., Ferrucci, F., and Gravino, C. "A User-centered Methodology to Generate Visual Modeling Enviroments" in *Enterprise Information Systems VI, Springer, Netherlands, 2006.*
- [6]. Eclipse, http://www.eclipse.org/
- [7]. ExtendedBNF, http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf
- [8]. Fischer, G. A critic for LISP, In Proc of the 10th IJCAI (Milan, Aug.1987) pp. 177-184.
- [9]. Florijin, G. RevJava-Design critiques and architectural conformance checking for Java software, Technical Report, White Paper, SERC 2002. http://www.bi-inbusiness.nl/site.nsf/0/A284738E9CD72E0EC1256E43002FE770/$file/Whitepaper_RevJava.pdf
- [10]. Green, T.R.G. & Petre, M. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. Journal of Visual Languages and Computing 1996 (7), pp. 131-174.

- [11]. Grundy, J.C., Hosking, J.G., Huh, J. and Li, N. Marama: an Eclipse meta-toolset for generating multi-view environments, Formal demoICSE'08, Liepzig, Germany, May 2008, ACM Press.
- [12]. Grundy, J.C., Hosking, J.G., Li, L. and Liu, N. Performance engineering of service compositions, ICSE 2006 Workshop on Service-oriented Software Engineering, Shanghai, May 2006.
- [13]. Loucopoulus, P., and Wan Kadir, W.M.N. "BROOD:Business rules-driven object-oriented design", *J Database Management, 19(1), 2008, pp. 41-73.*
- [14]. Oh,Y., Gross, M.D and Do, E.Y.-L, Computer-aided critiquing systems, lessons learned and new research directions. http://code.arc.cmu.edu/lab/upload/caadriaoh.0.pdf
- [15]. Qiu, L., and Riesbeck, C.K., An incremental model for developing educational critiquing systems: experiences with the Java critiquer, *J Interactive Learning Research,* 2008(19), pp.119-145.
- [16]. Robbins, J.E., Redmiles, D.F. Cognitive Support, UML Adherence, and XMI Interchange in Argo/UML. *IST* 2000, 42(2) pp. 71-149.
- [17]. Robbins, J.E., Hilbert, D.M. Redmiles, D.F. Software Architecture Critics in Argo. Intelligent User Interfaces 1998, pp. 141-144.
- [18]. Souza, C.R.B., et al. A Group Critic System for Object-Oriented Analysis and Design, Proc ASE 2000, 313-316.
- [19]. Wan Kadir, W.M.N., and Loucopoulus, P. "Relating evolving business rules to software design", *Journal of Systems Architecture, 50(7), Elsevier, 2004, pp.367-382.*
- [20] Ali, N. Md., Hosking, J.G., Huh, J. Grundy, J.C., Critic Authoring Templates for Specifying Doman-Specific Visual Language Tool Critics, In Proceedings of the 2009 Australian Software Engineering Conference, Gold Coast, Australia, April 2009, IEEE CS Press.
- [21] 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.
- [22] 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.
- [23] Ali, N. Md., Hosking, J.G. and Grundy, J.C. End-User Oriented Critic Specification for Domain-Specific Visual Language Tools, In Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, 20-24 Sept 2010, ACM Press.