A Generic Visual Critic Authoring Tool

Norhayati Mohd Ali
Department of Computer Science, University of Auckland,
Private Bag 92019, Auckland, New Zealand
nmoh044@ec.auckland.ac.nz

Abstract
Critic tools have been used for many domains, including design sketches, education, general engineering, and software design. The focus of this research is to develop a generic visual critic authoring framework embedded within an end user oriented domain specific visual language meta tool. This will allow tool critic support to be rapidly developed in parallel with the tools themselves.

1. Introduction
Developing complex software is costly, time consuming, and error prone. Due to the complexity of software, it is essential to have models and tools that assist end user software designers to manage that software, as designers’ minds cannot grasp all of the particulars. Many integrated support tools have been developed (e.g., ArgoUML, Visible Analyst, Rational Rose) to help designers in developing complex software or systems. One of the components of such integrated support tools are critics. The use of critic tools in software development is the focus of this research.

2. Background and Motivation
Much research has been devoted to critic tools. Argo/UML[1], ArchStudio[2], SoftArch[3], IDEA[4] and ABCDE-Critic[8] are all critic-based tools which focused on assisting software architects and designers with software architecture modelling. The critic support in these tools provides focussed, context dependent knowledge to designers and architects who lack specific understanding of the problem or solution domains. Generally, the critic tools detect potential problems, give advice and alternative solutions, and, possibly, automated or semi-automated design improvements to the end user. Most existing critic tools use representational approaches such as rule-based or object constraint language (OCL) expressions which are themselves complex to express, requiring deep programming understanding in order to design critics, providing a heavy barrier to entry to end user designers. Domain specific visual languages are a common approach to reduce such entry barriers, hence we see an opportunity to design and develop a visual critic authoring tool to support end user software designers to naturally and efficiently specify critics. In the remainder of this paper we expand on our approach, the methodologies we are using, results we have achieved to date and implications of our research.

3. Approach
Our aim is to design and prototype a generic visual critic authoring framework that allows end user (and other) software designers to simply and easily express and construct critics. The critic tool will be embedded within the Eclipse-based Marama [9] meta-tool environment allowing tool designers to concurrently develop visual language environments and critic support for them. Marama is a metatool implemented as a set of Eclipse plugins. Our approach will be based on the “Three examples” pattern of the Evolving Frameworks Pattern Language [7], initially hand crafting critic support for three Marama-implemented tools, and then generalizing from those exemplars to a generic visual language and support environment for critic authoring which we expect to be applicable to other Eclipse meta-tools such as GMF [5].

4. Methodology
The following methodology will be used during the research:

• Conduct a literature review of critic tools in software engineering domain, comparing and analyzing their approaches to critic specification and implementation.
• Identify from this an initial set of requirements for a visual critic tool.
• Develop an initial prototype to explore the problems and issues in designing critic tool.
• Design three examples of visual critic tools with different domains (e.g. software design, business process, and education) using the prototype, but with significant additional hand crafting.
• Identify from this experience a core set of building blocks needed for a generic critic authoring framework and design notation.
• Design and implement the visual critic authoring framework within a metatool.
• Evaluate the visual critic authoring tool by using it to construct prototype systems and examples, testing them carefully against a corpus of design examples developed for the initial prototype systems, and conducting an end user evaluation.

5. Results and Progress

Critic-based tools [1, 2, 3, 4, 8] identified in our literature review provide various levels of research contribution to critic tool development, such as the focus of the critic, the target user, and the approach used in the critic tools. Tools that provide significant contributions regarding the use of critics include Argo/UML and ArchStudio. The other tools (SoftArch [3], IDEA [4] and ABCDE-Critic [8]) are research prototype critic tools and not yet implemented into a realistic development environment.

Experience gained from the reviews has led to the design of an initial critic tool prototype developed using MaramaTatau [6]. MaramaTatau extends the Marama metatool set supporting specification of behavioural extensions to Marama meta-models. The first exemplar, a UML class diagramming tool has been defined using Marama. Critics for UML class design have been identified and formulated into the OCL expressions used by MaramaTatau, and associated with the UML tool meta-model. These critics are then applied in the executing tool (i.e. at the model or Marama diagram level) as shown in figure 1. OCL, as used in MaramaTatau will clearly not be our final critic authoring notation However, this initial prototype is a useful stepping stone towards our understanding of the needed building blocks for the generic framework. The next task is to flesh out the critic rule corpus for the initial exemplar and to begin identifying as well as organizing the needed building blocks for a visual critic authoring tool. The two subsequent exemplars will assist in this process. It is likely that we will take an incremental approach to developing the generic tool, developing it in parallel with the subsequent exemplars and refining the set of building blocks generalised from the initial exemplar as they prove deficient for the subsequent exemplars.

6. Research Implications

Due to the difficulty in defining critics as OCL expression within MaramaTatau, we see an opportunity for higher abstraction level notations to represent the critics. A visual design notation for specifying critics is preferable and will complement the other parts of Marama meta tool. The contribution of this research is that the higher level visual abstraction and its proof of concept implementation in the generic framework will allow end user designers to easily construct critics at the meta level. This will allow direct authoring of critics by end users, both for initial tool development, and also, for local customisation of tools by end users to support their particular business or design processes.

Figure 1: Simple critic (same named classes) violation in MaramaTatau prototype. OCL expression is: `Class.allInstances()->forAll(c1,c2 | c1 <> c2 implies c1.name <> c2.name)`.

7. References