

## Smell Detection in UML Designs which Utilize Pattern Languages

Bahman Zamani

PhD Candidate

Dept. of Computer Science, Concordia University, QC, Canada

September 2007

Model Driven Development (MDD) is a new paradigm in software engineering. One of the challenges in this paradigm, as others, is software quality management. Since the models are the main artifacts which drive software development in MDD, quality assessment of models is an important issue. As merely manual inspection or review is not enough, the tool assistance for quality assurance is necessary.

*Smell detection* is the idea of improving the quality of software by finding and fixing the problems- called bad smells- in the source code. The same idea is applicable at the design level. One of the usages of models in software engineering is to enhance the quality of produced software. Early detection of the problems (bad smells) with UML design models helps designers to produce high quality software.

Recently the usage of patterns in most of the software development phases is encouraged and designers are interested in using patterns while building software. One benefit of using patterns is to help designers to communicate their idea. The name *pattern language* comes from the fact that patterns create a vocabulary about design, if we always use the suggested pattern names. However, applying a pattern needs expertise and novice designers are vulnerable to make mistakes in using patterns.

This research aims to address the issue of detecting and repairing the bad smells in a UML design. So far, the following steps are taken.

1. A critiquing process called Sign/Criteria/Repair (SCR) for detecting and fixing the smells in the application of a pattern language in a UML design has been introduced.
2. As our case study, we have selected some patterns from Martin Fowler's "Patterns of Enterprise Architecture Applications" (Patterns of EAA) book. The book contains interrelated patterns which can be considered as a pattern language for building enterprise applications.
3. An investigation on how the SCR process can be implemented in different environments (tools) and how these tools can help the designer improve a UML model, has been done. The selected environments are state-of-the-art tools: ArgoUML, Epsilon, and OCLE.

As part of our future work, we aim to synchronize ArgoUML and EWL, as well as adding the possibility of doing "update transformations in the large" to the tool, meaning that upon confirmation by the user, all the problems in the models are fixed at a glance. Having the possibility of preview, such as what is available for code refactoring in IDEs would be another nice feature to add to the tool. Our final goal is to build a framework for verification of models, similar to the JUnit framework for testing Java programs. This framework would help users to better see how problematic is their design, how far they are from a sound design, and how much progress they made in fixing the problems.