A Framework for Maintaining Artefact Consistency during Software Development

Ildiko Pete and Dr Dharini Balasubramaniam*
School of Computer Science,
University of St Andrews

1 Introduction and motivation

A software system is composed of a variety of artefacts, which are products of the various activities involved in the development process. In practice, software artefacts evolve at different paces and modifications applied to one artefact may not necessarily get reflected in other related artefacts. This differential evolution of software artefacts may result in synchronisation issues and inconsistency among artefacts. Outdated artefacts also constitute impediments to effective system maintenance and evolution. Although incremental development provides a more flexible solution for handling changes [Larman 2001], artefact consistency is often neglected: different representations of software go through stages of refinement without considering all the dependent artefacts as the process does not enforce artefact links.

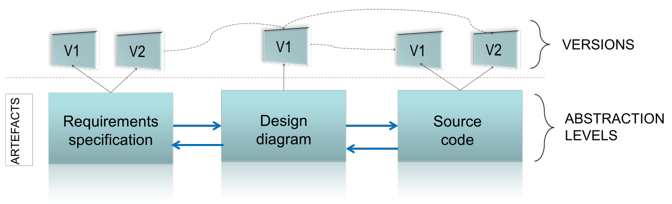

Figure 1: *The evolution of software artefacts.*

2 Evaluation of state of the art solutions

Various solutions have been proposed in the areas of consistency management and traceability, which aids change impact analysis [Ibrahim et al. 2005], to support consistent evolution of software artefacts [Lucia et al. 2010] [Reiss 2006] [Winkler and Pilgrim 2009] [Olsson and Grundy 2002] [Spanoudakis and Zisman 2001]. However, a number of challenges remain as current approaches only provide partial solutions: they are restrictive in terms of supported artefacts and they do not cater for all aspects of artefact consistency.

3 Current and Future Work

We hypothesise that a holistic and incremental framework that enables artefact synchronisation and consistency during the software lifecycle is a viable solution. We have identified the following aspects of artefact consistency management: trace link creation and maintenance, change detection, change impact analysis, consistency checking and change propagation.

The proposed work involves the design, implementation and evaluation of a framework for artefact consistency management. The novelty of the solution stems from its holistic approach to solving the artefact consistency issue. At the outset, three types of artefacts (requirements in natural language, design in UML class diagrams and Java source code) are considered.

*e-mail:ip24, dharini@st-andrews.ac.uk

Currently we are working on extracting fine-grained information on elements and their relationships from the original artefacts, and representing it using a graph structure.

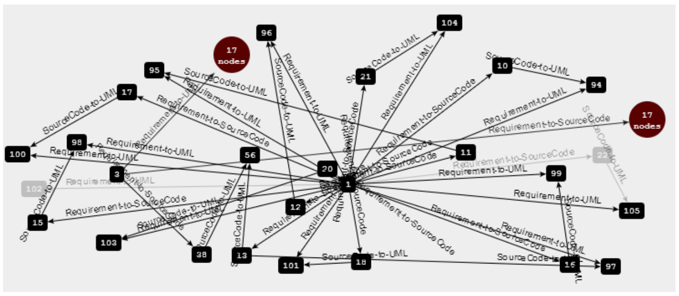

Figure 2: *Graph representation of software artefact elements and their relationships.*

Artefact elements are represented as nodes and relationships among them as edges of the graph. Traversing the graph will allow the analysis of the impact of changes. Work planned for the near future includes the remaining aspects of consistency management for the considered artefacts and the evaluation of the proposed framework using artefacts from open source repositories to assess the feasibility of the approach.

References

Ibrahim, A. D. S., Munro, M., and Deraman, A. 2005. A requirements traceability to support change impact analysis. *Asean J. Inf. Technol.*, 1–12.

Larman, C. 2001. *Applying UML and Patterns*. Prentice Hall.

Lucia, A. D., Fasano, F., Oliveto, R., and Tortora, G. 2010. Fine-grained management of software artefacts: the adams system. *Softw. Pract. Exper.*, 1007–1034.

Olsson, T., and Grundy, J. 2002. Supporting traceability and inconsistency management between software artefacts. In *International Conference on Software Engineering and Applications*.

Reiss, S. P. 2006. Incremental maintenance of software artifacts. *IEEE Trans. Softw. Eng. 32*, 9, 682–697.

Spanoudakis, G., and Zisman, A. 2001. Inconsistency management in software engineering: Survey and open research issues. In *in Handbook of Software Engineering and Knowledge Engineering*, World Scientific, 329–380.

Winkler, S., and Pilgrim, J. 2009. A survey of traceability in requirements engineering and model-driven development. *Softw. Syst. Model. 9*, 4, 529–565.