Using a Rule-Base Approach and Responsibility Modeling for Automatic Software Evolution

Ming-Jen Huang
School of Information Science,
Japan Advanced Institute of Science and Technology

March, 2008

Abstract

The purpose of this research work is to improve software evolution by managing the complex relationships between abstractions of different development stages. To this end, we propose and implement an automation approach for managing these relationships. This approach is based on the idea of capturing and reusing various types of relationships between abstractions.

A program realizes different types of high-level abstractions. As more functions are added to the program, the realization relationship between the program and the high-level abstractions conceived in the development process becomes more complex. To evolve a program without downgrading its quality, managing this complexity is the key point. To this end, in this research work we propose a new development approach, which is based on three theories. (1) First, to eliminate the gap between different worlds in software development process, we use a single-type paradigm for modeling abstractions that are created in different worlds but are also related at the same time. (2) Second, to simplify the evolution of the relationships among abstractions, we propose directly creating a program by reusing previously considered development knowledge of relationships among abstractions. More specifically, a program is constructed from the modules of the relationships of abstractions which are conceived in the development process and are recorded by the single-type paradigm in (1). (3) Third, we propose using rule-based engine for implementing a tool for automating software evolution by reusing and composing the modules mentioned in (2). The automation provided by this approach and its implementation is for the following three evolution scenarios: (a) when the given business processes are evolving, (2) when the realization-development knowledge evolving, and (3) when a different implementation technology is adopted. To evaluate the effectiveness of the proposing approach, a case study with three software systems is conducted.

In this dissertation, we described the construction of the proposing approach. In the first step, the basic framework is constructed. This framework helps developers to capture development knowledge they acquire in the development process. It includes a modeling language and a set of graphical notations. We then described that how the modules of relationships among abstractions can be used to construct/evolve a program. In the second step, the implementation for automated program construction/evolution is

developed. This implementation provides the features of development knowledge modeling and program construction/evolution automation. Finally, a case study is conducted. The results of the case study provide the support for the proposing three theories for software evolution.

The evaluation results show that a single-type paradigm by using responsibility can be effectively used to describe the relationships of abstractions within the four worlds. The modularization of development knowledge by using PRUs can effectively capture how developers design realization of abstractions of different worlds. Finally, a rule-based engine can encode the development knowledge of PRUs for inferring the development of system responsibilities, object responsibilities, and program responsibilities

Key Words: **software evolution, rule-based, automation, model-driven development**