An Input-Driven Approach to Generate Class Diagram and Its Empirical Evaluation

Faridah Hani Mohamed Salleh

Department of Software Engineering,
College of IT, Universiti Tenaga Nasional,
43009 Kajang, Selangor, Malaysia
faridahh@uniten.edu.my

Abstract. This paper presents an approach for generating Unified Modeling Language (UML) class diagram. The objective of the project is to assist users in generating class diagram in a systematic way, with less dependent to the developers' skill. The approach consists of a number of steps or instructions to be executed by users. Executing the steps requires the user to enter inputs that will then generate half-completed class diagram. Step-by-step approaches are presented to show how the candidate of classes, attributes and relationships between classes are captured. The proposed approach is presented using a case study of ATM system. Testing was conducted to assess the effectiveness of the proposed approach. Analysis and discussion of the testing results are discussed in a few last sections of this paper. The development of the system is underway.

Keywords: Object-oriented analysis and design, class diagram.

1 Introduction

Class diagram is categorized as an important diagram in the development of object-oriented application. Class diagram is often starting to be discovered during analysis phase. A process of generating a complete and correct class diagram is something that needs to be put into consideration. For the large and complex system, the process of generating the class diagram can be very time consuming and in many cases, it would result in many problems if it is not being done carefully. Should a class be badly identified by developers, it can complicate the application's logical structure, reduce reusability, and deter the application's maintenance [1].

There is no formal or standardized ways of designing OO (object-oriented) application, where developers define and design the application based on their experiences and skills level. Motivated by the above situation, the project attempts to propose an approach that can facilitate class diagram generation, lower the risk of producing an inaccurate class diagram and produce class diagram that useful for development. The real system that simulates the proposed approach is yet to be developed. Thus, the testing was conducted using Excel to mimic the flow of the proposed approach.

At this stage, the approach is designed to be able to find the following elements of class diagram:

- Classes
- Attributes of classes
- Relationships between classes

This proposed approach works based on the following assumptions:

- A complete and correct use case diagram must be produced beforehand.
- Class diagram produced from this approach must go through several refinement processes before it becomes correct and complete.
- The user knows what they want the system to do and manages to list basic flows of system.
- Have knowledge of the application domain.

Method to identify operations and multiplicity are out of this project scope and will be considered for future enhancements of this project.

2 Motivation

The next generation of software engineering will involve designing systems without using paper-based formats, instead using software to develop software [3]. Although most of the techniques were well-explained, up to the author's knowledge, none of them is computerized. There is a shortage of good tools for supporting OO development efforts. The tools include [3]:

- programs to assist in the design of objects,
- manage libraries of reusable objects,
- design and maintain data-input forms and reports and
- coordinate the development efforts of large teams of programmers.

Most of the past and current research projects in OO give attention to the results after execution of OOAD (object-oriented analysis and design) phase. For example, concentration had been given to the resulting class diagram where the diagram was checked for its quality, consistency and many other aspects. Lack of research is conducted to help developers in performing early stage of OOAD where producing useful models for OO development is needed.

The process of generating OO related work products or specifically class diagram cannot be formalized, but relies on experience, creativity and intuition. This situation causes problem to novice developers as they need much time to learn OO concepts compared to expert. Most of the techniques to produce class diagram requires significant training, practice, intuition, and experience, which usually takes at least 6 months of on-the-job training [4]. Since OO technology requires an entirely different approach to software development, there are significant costs associated with making the transition to OO technology. Education and training for developers can be costly and developers can experience a loss of productivity as they learn to work effectively with the new concepts and techniques.