CHAPTER I

INTRODUCTION

1.1 Problem Statement

Software testing is one of the most important activities in software development life cycle. Software is required to be adequately tested in order to gain confidence from both software providers and software users. A lot of test techniques have been invented to detect fault in software according to various fault assumptions and test levels.

Since the last decade, object-oriented analysis and design has become a dominant paradigm for software development due to its advantage in many areas. Features in object-oriented paradigm allow and encourage software designers and developers to solve complicated problems with modular, reusable, and extensible design structures. Unfortunately, these features introduce new kinds of defects that are not found in traditional procedural software, as stated in a number of research publications [1, 2, 3, 4, 5, 6]. Most research results state that testing techniques designed for traditional procedural software are not effective in detecting fault in object-oriented software. It implies that there is an undeniable need for specific test methods for object-oriented software to effectively detect these defects.

UML [7] becomes more important as the most popular standard for modeling object-oriented systems. Software engineers can effectively use diagrams and notations defined in UML to communicate their ideas in a form of models in various views and abstraction levels. This brings up a need to verify whether the work products, particularly programs, are created according to their models. This is classified as conformance-based testing [8], which focuses on finding discrepancies between software and its specification rather than finding defects according to fault assumption like fault-based testing.

From the necessity of specific test techniques for object-oriented software and the growth of UML as modeling language, an approach for testing object-oriented software based on UML diagrams is in need. UML sequence diagram is selected as test

specification for this research. It is a good choice as test specification for cluster level testing, since it represents an interaction between objects in a cluster. In addition to aiming at checking conformance, the approach must also take into account the features and characteristics of object-oriented software, polymorphism in particular for this research. With a systematic approach, it is possible to automate the test process, at least to some extent, which could ease the task of test designers and testers.

1.2 Objective

The objective of this research is to define an approach for testing object-oriented software based on UML sequence diagrams, as interaction diagrams, focusing on polymorphic interactions. Message sending instrumentation, its analysis, and test case creation method are also included in the approach.

1.3 Scope

1. To define an approach to create test cases for object-oriented software from UML sequence diagrams with some auxiliary information from class inheritance hierarchy.
2. Adequacy criteria are defined based on the target approach for guiding test case generation. The criteria are specific to polymorphic interaction testing.
3. A test case contains the conditions of test inputs, test setup steps, and the expected message sending sequence.
4. The target approach evaluates message sending sequences for test results. Evaluation of outputs of programs under test is not supported by the target approach. The message sending sequence verification procedure supports verification of implementation with refinement.
5. UML sequence diagrams and/or UML class diagrams may be altered, possibly adding some additional information such as markers for polymorphic assignment (e.g. in a form of stereotype) to supply some information, which is not originally available to the test case generation process.

6. Class inheritance hierarchies and other information, e.g. operations and their signatures, are required in addition to sequence diagrams. They can possibly be gathered from a design model such as class diagrams.
7. A tool for testing Java programs based on this approach is implemented to demonstrate the concept of this approach. The tool must take UML sequence diagrams in a definite form, e.g. XMI format [9], declarative textual specification [10] etc., as the primary input for test case generation with some additional sources for additionally required information. Generated test cases are not, by themselves, executable, but they contain the conditions of test inputs and test setup steps which are sufficient for writing a complete test driver. The expected message sending sequence is also generated by the tool. Also the tool is capable of verifying message sending sequences according to the defined approach.
8. This approach is evaluated through the implemented tool. Four different interactions in the form of UML sequence diagrams are selected for test case generation for each of proposed adequacy criteria. The generated test cases are evaluated according to each selected adequacy criterion. Programs for the interactions are developed and tested with the test cases. Some parts of the programs are intentionally altered to be different from diagrams for the purpose of evaluation of message sending sequence analysis procedure.
9. Issues about multiple inheritance are not considered in this research.

1.4 Research Methodology

This research aims at defining an approach for solving the problems of object-oriented software testing, particularly polymorphic related ones, discussed in the previous section. UML sequence diagram is selected as a test specification for this approach due to its popularity. It represents an interaction of a cluster of objects; therefore, it is appropriate for test specification of cluster level testing. The main purpose of this approach is to find defects which are the result of the behavior of the software under test, which is different from what is defined in the design model. As a UML sequence diagram is designed as a message sending sequence between objects in an

interaction, it is expected that the software implemented from this diagram has an equivalent message sending sequence. From a UML sequence diagrams as a design model, an expected message sending sequence is extracted. An actual message sending sequence is captured from the instrumentation of the execution of the software under test, implemented from the design model. Figure 1.1 shows the overview of this approach.

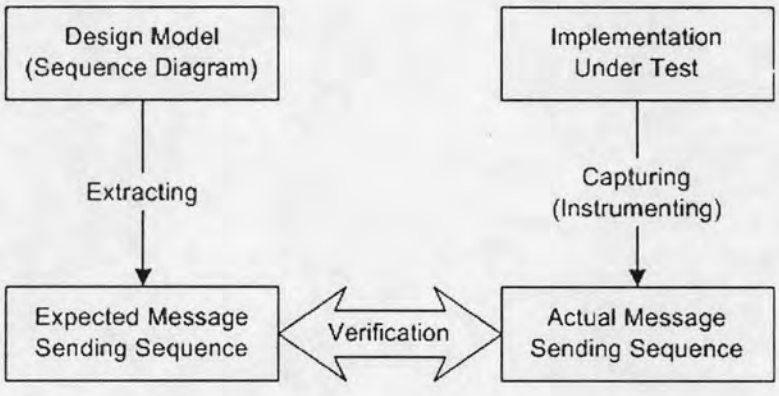

Figure 1.1 Overview of the approach

The analysis of message sending sequence is not just a plain comparison between message sending sequences. Since design models, UML sequence diagrams in particular, may not capture all details, it is possible that their actual implementation is different from the models to some degree. The procedure of message sending sequence analysis must take this issue into account.

Polymorphism is an important issue in this research. It is important that occurrences of "polymorphic assignment" [11] in a UML sequence diagrams are identified so that how to control a particular polymorphic behavior is known. For this purpose, patterns of polymorphic assignments must be defined. The patterns allow polymorphic assignments to be systematically identified.

While how to control polymorphic behaviors is identified by the patterns, it is also important to define adequacy criteria for testing a program which makes use of polymorphism. It is obvious that adequacy criteria related to polymorphism are in need, since there is no counterpart from procedural software testing techniques.

Polymorphism-related defects must be taken into account when designing these criteria. Besides ensuring test coverage, these adequacy criteria are an important tool for test case generation.

A method for test case generation is defined based on the adequacy criteria. In this step, elements in UML sequence diagrams are examined to see how they can help in generating test cases. Extension of basic elements in UML sequence diagrams is possibly necessary for defining properties not readily available from the basic elements.

A tool which is an implementation of the concepts presented in this research is important for demonstrating and evaluating the concepts. A number of UML sequence diagrams and their implemented programs must be used to evaluate the tool. Although the tool is not a fully-automated implementation of the concepts, it should be sufficient for evaluation purpose.

The summary of research activities is illustrated as a UML activity diagram in figure 1.2.

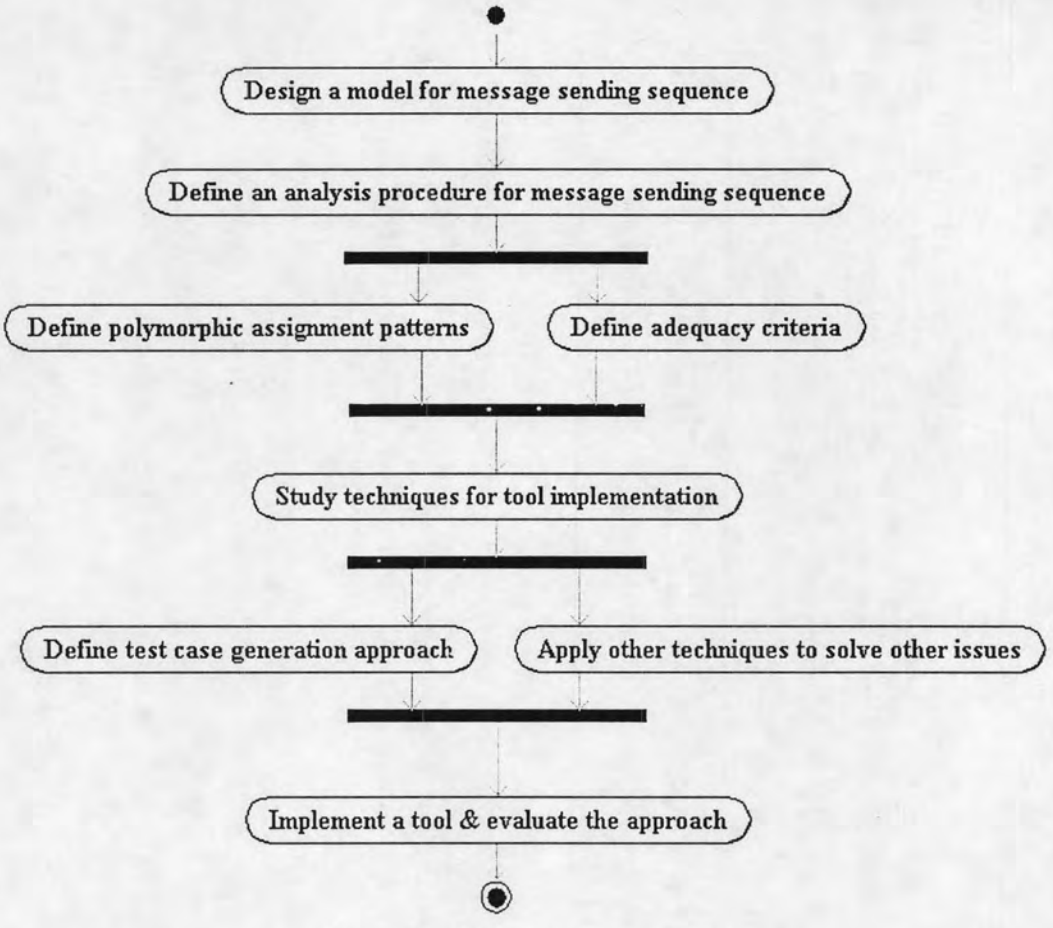

Figure 1.2 Activities in this research

1.5 Contribution

1. In addition to output, test execution can be checked whether message sending sequences in the actual execution are equivalent to the one designed in UML sequence diagrams.
2. With the concern of polymorphism, test cases can be generated automatically from UML sequence diagrams according to adequacy criteria.

1.6 Publications

Several parts of the research are selected to be presented in international conferences. They are also published in the corresponding proceedings as show below. Full papers of the publications are in appendix A, B, and C respectively.

1. "An Instrumentation Model for Supporting Software Testing Based on UML Sequence Diagrams", Proceedings of Information and Computer Engineering Postgraduate Workshop 2004, Phuket, Thailand, January 22-24, 2004.
2. "Adequacy Criteria for Testing Polymorphism in the Context of Interactions", Proceedings of the 2004 International Conference on Software Engineering Research and Practice, Las Vegas, Nevada, USA, June 21 – 24, 2004.
3. "Testing Polymorphic Interactions in UML Sequence Diagrams", Proceedings of the International Conference on Information Technology, Las Vegas, Nevada, USA, April 4-6, 2005.