

Fault Detection Effectiveness of UML Design Model Test Adequacy Criteria

Nilesh Kawane

Computer Science Department, Colorado State University
kawane@cs.colostate.edu

Abstract

UML models are widely used by software developers to model complex software systems. Testing the models can reveal flaws in the early stages of software development. Criteria based on UML class diagrams and interaction diagrams were proposed earlier to specify UML model elements that need to be covered during testing. This paper describes a case study that evaluated the fault detection effectiveness of these criteria.

Keywords: Class diagram, collaboration diagram, software testing, test adequacy criteria, UML.

1. Introduction

In the design phase, software developers model a system using various types of UML [2] diagrams representing different views of the system. Systematic testing of the models can detect existing faults even before the actual implementation of the system, thereby reducing the cost of finding faults in later developmental phases. Dynamic analysis of the models for testing the modeled behavior requires model execution.

Andrews et. al [1] propose several criteria to guide the selection of test cases and measurement of test adequacy. The criteria can be broadly classified as class diagram criteria and interaction diagram criteria. Class diagram criteria include the following:

1. *Association-end Multiplicity (AEM)*: Designed to ensure that configurations containing boundary and non-boundary occurrences of links between objects are tested.
2. *Generalization (GN)*: Designed to ensure that each specialized class is instantiated and used during the tests.
3. *Class Attribute (CA)*: Designed to ensure the testing of behaviors using combinations of representative class attribute values.

Interaction diagram criteria include:

1. *Condition Coverage (Cond)*: Designed to ensure that each condition in the interaction diagram evaluates to true and false during the tests.
2. *Full Predicate Coverage (FP)*: Designed to ensure that every predicate in each condition inside the interaction diagram evaluates to true and false.
3. *Each Message on Link (EML)*: Designed to ensure that each message on a link connecting two objects in the interaction diagram is executed at least once.
4. *All Message Path (AMP)*: Designed to ensure that each possible message path in the interaction diagram is executed at least once.
5. *Collection Coverage (Coll)*: Designed to ensure that tests execute each interaction with collection objects using different representative sizes.

Developers need to plan and allocate resources for testing. The planning process is influenced by the ability of adequate test sets to detect faults and the effort required to create the tests. We set up a couple of case studies to measure (1) the fault detection effectiveness of the criteria and (2) the number of test cases required to satisfy each criterion.

2. Evaluation Procedure

Two systems were used: (1) a web based course management system (WebC) and (2) a poker gaming system (Poker). The WebC system model consists of a set of eight use cases and corresponding collaboration diagrams, and one class diagram containing eight classes. The Poker system model consists of a set of nine use cases and corresponding collaboration diagrams, and a class diagram containing six classes. Each system has one instance of the generalization-specialization relationship. The WebC system defines constraints on the values of some of the class attributes using the Object Constraint Language (OCL). The Poker system defines pre- and post-conditions for some system operations using the OCL. Several collaboration diagrams model conditional constructs. Thus, there is more than one possible execution path in those diagrams. In

WebC, the system operations are independent of each other, whereas in Poker, the system operations must occur in a defined sequence.

A set of faults that commonly occur in UML models was compiled by identifying the faults that designers could introduce in the constructs of a collaboration diagram. Each modeler seeded a fault one by one in each collaboration diagram, thereby generating a number of faulty models. These models were given to the testers. First, the testers randomly generated test cases using a computer program which used information regarding types and constraints on parameters of system operations and the sequence in which system operations can occur. The test cases that increased coverage with respect to some criterion were added to the test set. Since the randomly generated test cases did not result in complete coverage, a few test cases had to be added manually by the tester.

Each test case specifies a sequence of operations and takes the following general form:

$$\{ M_0(p_{00}, p_{01}, \dots, p_{0l}), \\ M_1(p_{10}, p_{11}, \dots, p_{1l}), \\ \dots \\ M_n(p_{n0}, p_{n1}, \dots, p_{nl}) \}$$

where M_i is a system operation and p_{ij} is a system operation parameter corresponding to M_i . A test set contains several such test cases.

Testing is carried out by executing the system operations in the sequence using the specified parameter values. Test failure is indicated by violations of constraints on the system operations, an inconsistent system configuration or a deviation in the system behavior from the specification in the use cases.

3. Evaluation Results

The results of testing the systems are given in Tables 1 and 2. Eight out of ten seeded faults were detected in the WebC system. Five out of nine seeded faults were detected in the Poker system. The WebC system required thirteen test cases to detect the seeded faults while the Poker system required fourteen. Table 3 summarizes our observations about relations between failures and their possible causes.

4. Conclusions and Future Work

We reported two case studies that evaluated the effectiveness of UML model-based test adequacy criteria. The study also gave insights into the test case generation procedure. The systems used in the study were comparatively

Table 1. Test Results for WebC System

Test Criteria	Number of Test Cases	Number of Faults Detected
Cond, EML, AMP	7	3
AEM	3	2
GN	3	3
CA, FP	N/A	N/A
Total	13	8

Table 2. Test Results for Poker System

Test Criteria	Number of Test Cases	Number of Faults Detected
Cond, EML, AMP	12	5
AEM	N/A	N/A
GN	2	0
CA, FP	N/A	N/A
Total	14	5

Table 3. Faults and Probable Cause

Detected Faults	Probable Cause
Message to null object	Order of messages, missing conditions
Use of a variable before initialization	Order of messages, incomplete models
Inconsistency between class and interaction diagrams	Missing create or call message
Incorrect message sequence number or missing messages	Missing messages, missing conditional branch
Missing messages (as compared to use case descriptions)	Missing call message, incomplete models.

smaller and simpler than real-world systems. We are planning similar studies on complex systems. We are also working on building an exhaustive repository of system failures and faults that cause them. We are investigating the types of faults targeted by each criterion.

References

- [1] A. Andrews, R. France, S. Ghosh, and G. Craig. Test Adequacy Criteria for UML Design Models. *Journal of Software Testing, Verification and Reliability*, 13(2):95-127, April-June 2003.
- [2] The Object Management Group. The Unified Modeling Language. Version 1.4, OMG, formal/2001-09-67, 2001.