Chapter 39
Consistency Checking of Specification in UML

P. G. Sapna
Coimbatore Institute of Technology, India

Hrushikesha Mohanty
University of Hyderabad, India

Arunkumar Balakrishnan
Coimbatore Institute of Technology, India

ABSTRACT

The increasing use of software is giving rise to the development of highly complex software systems. Further, software systems are required to be of high quality as a defect can have catastrophic effect on business as well as human life. Testing is defined as the process of executing a program with the intention of finding errors. Software testing is an expensive process of the software development life cycle consuming nearly 50% of development cost. Software testing aims not only to guarantee consistency in software specification but also to validate its implementation meeting user requirements. On the whole, it is observed that in general, errors in software systems set in at the early stages of the software development cycle (i.e. while gathering user requirements and deciding on specification of intended software). Even though formal specification in B and Z assures a provable system, its use has become less popular due to mathematical rigor. The Unified Modeling Language (UML), a semi-formal language with graphical notations consisting of various diagrams has caught software developers' imaginations and, it has become popular in industry. UML, with its several diagrams, helps to develop a model of intended software, and the model behaviour is simulated and tested to the satisfaction of both developer as well as users. As a UML model includes specifications of different aspects of a software system through several diagrams, it is essential to maintain consistency among diagrams so that quality of the model is maintained, and through inconsistency checking and removal, the model moves toward completeness. The works reported in literature on this topic are reviewed here.

DOI: 10.4018/978-1-5225-3923-0.ch039

INTRODUCTION

Specification is the genesis of a software system and maintaining its correctness is of prime concern for ensuring quality of system under development. Software system testing includes both checking specification as well as its implementation. Formulating software specification (requirements gathering) precedes design. Both need a method for concrete as well as unambiguous specifications so that the testing team can trace implementation to requirements. Though formal specification with Z or B leads to provable systems, they are not commonly used due to mathematical rigour.

Responding to the want of a concrete as well as acceptable technique for professionals in industry, the Object Management Group defines the Unified Modelling Language (UML) as a general-purpose visual modeling language that is used to specify, visualize, construct and document artifacts of a software system. UML captures information about the static structure as well as dynamic behaviour of a system. The static structure defines objects as well as the relationship between objects that are part of the system implementation usually represented using use case, class and component diagram. Dynamic behaviour of the system is specified by the activity, sequence and state diagrams.

The semi-formal nature of UML has both advantages and disadvantages: the advantage primarily lies in its ease of use as well as understandability by various stakeholders of the system. Also, different diagrams can be used to model varying aspects of the system. The same leads to difficulties in the form of maintaining completeness and consistency within and between UML diagrams. Specification based testing using UML needs consistent and complete UML diagrams. Again for testing, scenarios representing the working of intended system are extracted and studied for the purpose.

The focus of this chapter is to look at how the Unified Modeling Language aids in exploring the issue of consistency checking which forms the basis for testing. The Unified Modeling Language is discussed, with the use as well as advantages and disadvantages. Next, the issue of checking consistency in UML models is explored followed by a discussion on Model-driven Testing. Comparison of work in the area is presented followed by scope for research.

THE UNIFIED MODELING LANGUAGE

What Is UML?

The Unified Modeling Language (UML) is a general-purpose visual modeling language used to specify, visualize, construct, and document artifacts of a software system. Developed and propagated by the OMG group, UML can be used across all phases of the software development process (requirement, analysis and design, testing, and documentation). One or more diagrams can be used to represent the system. UML models can be classified as static models and dynamic models. Static models represent the structure of the system, whereas dynamic models are used to represent the behaviour of the system. Thus, a combination of the models may be used to suit the type and domain of the software to be developed. A UML diagram is not refined to provide all relevant aspects of an application. The semi-formal nature of UML leads to ambiguities in representation and interpretation of stated requirements. To overcome this, the Object Constraint Language (OCL) is used to write constraints on model elements. OCL expressions are used to specify invariants on classes, define pre- and post conditions on operations and methods,