International Journal of Information System Modeling and Design
Volume 7 • Issue 3 • July-September 2016

An SMT-based Approach for Generating Coverage Oriented Metamodel Instances

Hao Wu, Computer Science Department, National University of Ireland, Maynooth, Ireland

ABSTRACT

An effective technique for generating instances of a metamodel should quickly and automatically generate instances satisfying the metamodel's structural and OCL constraints. Ideally it should also produce quantitatively meaningful instances with respect to certain criteria, that is, instances which meet specified generic coverage criteria that help the modelers test or verify a metamodel at a general level. In this paper, the author presents an approach consisting of two techniques for coverage oriented metamodel instance generation. The first technique realises the standard coverage criteria defined for UML class diagrams, while the second technique focuses on generating instances satisfying graph-based criteria. With the author's approach, both kinds of criteria are translated to SMT formulas which are then investigated by an SMT solver. Each successful assignment is then interpreted as a metamodel instance that provably satisfies a coverage criteria or a graph property. The author has already integrated this approach into his existing tool to demonstrate the feasibility.

KEYWORDS

Coverage Criteria, Graph, Instance Generation, Metamodel, Satisfiability Modulo Theories (SMT)

INTRODUCTION

A model provides a representation of aspects of a system. This can include design models such as UML class or sequence diagrams, or implementation models, such as source code in a programming language. A metamodel is a model that is used to describe the structure of other models, modelling languages or domain specific languages. Each instance of a metamodel is then a model that can be regarded as a test case. These test cases are important not just for validating a metamodel itself, but also useful for testing the tools and frameworks that process the models defined by that metamodel such as model transformation.

For example, given a domain specific language *L*, say, a metamodel would usually define the abstract syntax and static semantics of the language. A typical representation of the metamodel would be as a UML class diagram (using a subset of the constructs) with constraints specified using the Object Constraint Language (OCL). A set of instances of this metamodel would be programs written in language *L*, and would allow language engineers to check that they had specified the relevant constructs correctly.

A number of approaches and tools have already provided a way of generating these instances (Ehriget al., 2009; González Pérez et al., 2012; Cabot et al., 2014). However, these instances are not measured via any criteria. At least, meeting some criteria such as standard coverage criteria for UML class diagram would help users to increase their confidence in designing or validating metamodels.

DOI: 10.4018/IJISMD.2016070102

Furthermore, users may also wish to generate instances that possess certain coverage metrics for other testing purposes such as using depth of inheritance tree for testing inheritance relationships. Thus, naively generating instances from a metamodel without taking account of coverage criteria or other properties is not very adequate.

This paper addresses the issue of generating metamodel instances satisfying coverage criteria. More specifically, this paper makes the following contributions:

- A technique that enables metamodel instances to be generated so that they satisfy partition-based coverage criteria.
- A technique for generating metamodel instances which satisfy graph properties.

Both two techniques that encode coverage criteria and graph properties into a set of SMT formulas. These formulas are then combined with the formulas generated from our previous work, and solved by using an external SMT solver (Wu et al., 2013). Each successful assignment for the formulas is interpreted as an instance. We have already automated this process into a tool to demonstrate the feasibility of this approach.

BACKGROUND

In this section, we briefly review the standard coverage criteria defined for UML class diagram, notations we use for expressing a metamodel as a graph, and basic SMT encodings from our previous work. Formally, we consider all metamodels in this paper as being presented as UML class diagrams, and represented as graphs.

METAMODEL COVERAGE CRITERIA

A metamodel is a structural diagram and can be depicted using the UML class diagram notation. Thus, the coverage criteria defined for UML class diagram can also be borrowed for metamodels. In particular, we focus on the coverage criteria presented in (Andrews et al., 2003) (Ghosh et al., 2003), especially the work focused on testing the structural elements of a UML class diagram. These coverage criteria are standard criteria for testing a UML class diagram and they are defined as follows:

- Generalisation coverage (GN) which describes how to measure inheritance relationships.
- Association-end multiplicity coverage (AEM) which measures association relationships defined between classes.
- Class attribute coverage (CA) which measures the set of representative attribute value combinations in each instance of class.

AEM and CA are partition-based testing criteria which means that testing results depend on the choice of a representative value from each partition (Ostrand and Balcer, 1988). Therefore, the value domain is partitioned into several equivalence classes, and each value from an equivalent class is expected to have the same results. The partitions can also be decided using domain knowledge based partitioning.

For example, to satisfy the CA criterion for the metamodel in Figure 1, we may assume a user could choose a representative value of 18 [1], and this allows the attribute *age* in the abstract class Person to be divided into *3* partitions which are $18<age$, $age = 18$ and $age>18$. The hypothesis is that any single value from one of the three partitions is expected to have the same results for all other values from that partition (Myers and Sandler, 2004). Similarly, to satisfy the AEM criterion, the binary

Open-Source Software Issues

Sofiane Sahraoui (2009). *Software Applications: Concepts, Methodologies, Tools, and Applications (pp. 33-38).*

www.igi-global.com/chapter/open-source-software-issues/29376?camid=4v1a

Integrating Access Control into UML for Secure Software Modeling and Analysis

Thuong Doan, Steven Demurjian, Laurent Michel and Solomon Berhe (2010). *International Journal of Secure Software Engineering (pp. 1-19).*

www.igi-global.com/article/integrating-access-control-into-uml/39006?camid=4v1a