Generating OCL Constraints from Test Case Schemas for Testing Model Behavior

Nisha Desai and Martin Gogolla

University of Bremen, Department of Mathematics and Computer Science,
D-28334 Bremen, Germany
{nisha,gogolla}@informatik.uni-bremen.de

Abstract. This contribution studies testing behavioral aspects of a given UML and OCL model. In our approach, a so-called model validator can automatically generate test cases (object models) by using configurations for the object models and manually formulated OCL invariants. But expressing OCL invariants can be complex and difficult, especially for novel or occasional modelers. In this contribution, we present an approach to automatically transform a diagrammatic test case schema into a corresponding OCL invariant. The schema is a visual representation of a behavioral test scenario constructed by the developer and which is instantiated by the model validator to achieve different concrete test cases. This approach enhances the underlying testing technique in making it developer-friendly and independent of OCL expertise.

1 Introduction

As the size and complexity of models grow, there is an increasing need for testing their correctness. Today, modeling languages such as the UML along with the OCL are used to describe structural and behavioral aspects of a system.

For checking such crucial properties of a UML and OCL model, the tool USE can be employed to transform a given application model into an equivalent so-called filmstrip model [5]. In USE, a model validator (MV) can automatically generate valid object diagrams based on given configurations (determining finite sets of objects, links and attribute values) and external OCL invariants. For the validation process, external OCL invariants are currently manually formulated. However, writing OCL expressions is a difficult and time-consuming task and often results in erroneous constraints. To address this problem, we propose an approach where developers can express a scenario by constructing a so-called *test case schema (TC schema)* which then can automatically be transformed into an OCL invariant. Furthermore, the MV is used to instantiate the abstract TC schema in order to generate multiple concrete test cases.

The rest of the paper is structured as follows. Section 2 provides the background and motivation of our work. Section 3 describes TC schemas with an example and its transformation to an OCL invariant is explained in Sect. 4. In Sect. 5, we show test case generation using the MV. Section 6 presents related work and we conclude our paper with future work in Sect. 7.

2 Background

The MV in USE is specifically designed for structural analysis of models. Therefore, we use our filmstrip approach which transforms invariants (structural properties) and operation pre- and postconditions (behavioral properties) of an application model into a filmstrip model which possesses only invariants.

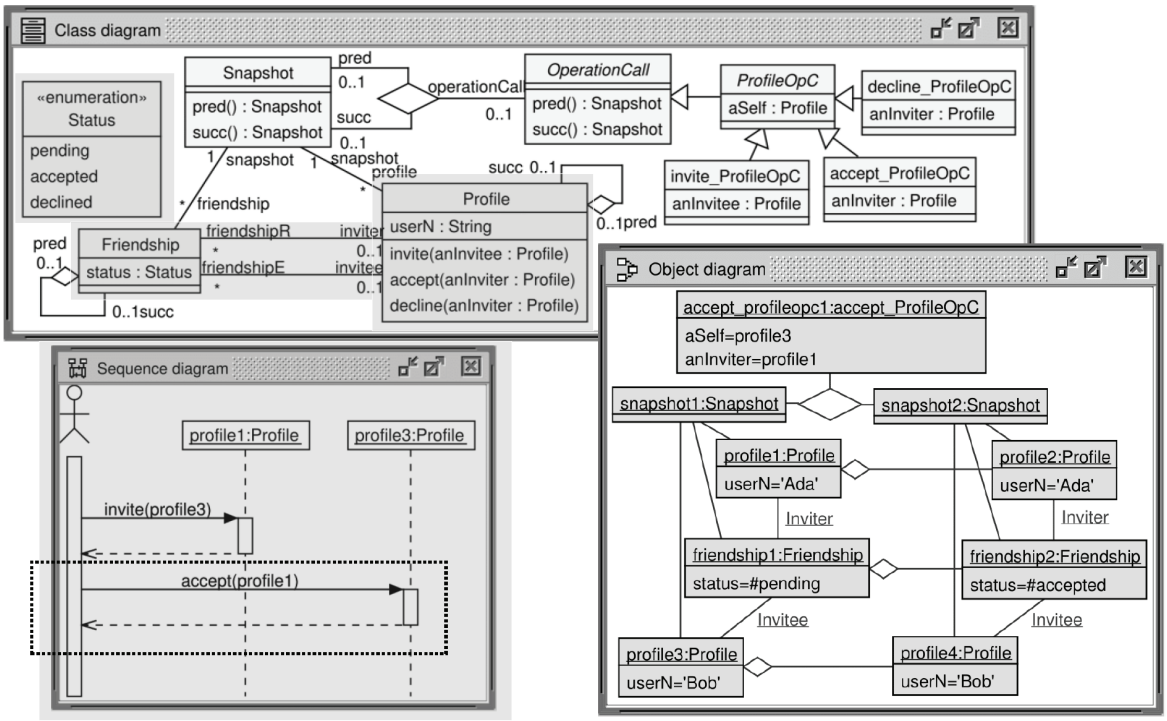

Fig. 1. Application model and filmstrip model.

The filmstripping approach can be explained best in terms of an example. A simple `SocialNetwork` model in which a user can invite, accept and reject a friendship request is chosen as an example. The upper part of Fig. 1 shows the class diagram of the filmstrip model. The original application model, consisting of the classes `Profile` and `Friendship` with the associations `Invite` and `Invitee`, is completely contained in the filmstrip model and indicated in a gray-shaded style. The small sequence diagram also represents elements of the application model. The application model is automatically transformed with a plug-in into the filmstrip model: the non-gray shaded classes and invariants (not shown) are added. In essence, the application model sequence diagram becomes a filmstrip model object diagram. `Snapshot` objects explicitly allow to capture single system states from the application model. Operation call objects (suffix OpC) describe operation calls from the application model. Basically, each operation is transformed into an `OperationCall` class with attributes for the `self` objects and for the operation parameters. Thus, for example, the call `profile3.accept(profile1)` (dotted box) from the sequence diagram is represented by the object `accept_profileopc1` in the filmstrip object diagram. The effect of the operation call is represented by the differences between the left and the right snapshot: The `accept` operation call changes the attribute `status`. The

four `Profile` and the two `Friendship` objects represent different object states before and after the operation call. So one could say that the object `profile2` is a later incarnation of the object `profile1`.

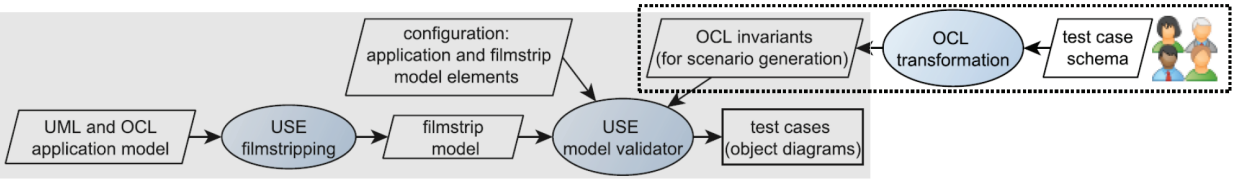

Fig. 2. Overview on filmstrip validation (gray box) with TC schema approach (dotted box).

The gray highlighted part in Fig. 2 gives an overview on the existing filmstrip transformation and model validation process. In the validation process, external OCL invariants are specified to guide the object diagram generation into a particular direction, e.g., for attesting that objects or links with particular properties exist (for example, for the initial or final scenario state) [5]. Up to now, these external OCL invariants had to be written manually. Thus, we propose an approach where a TC schema, which is a diagrammatic representation of a scenario, can automatically be transformed into an OCL invariant for model validation in order to make the validation process free of OCL expertise.

3 Test Case Schema Example

A TC schema is basically a partial filmstrip object diagram consisting of different snapshots which represent system states, and these snapshots can contain application model objects and links. The objects of different snapshots can be connected through filmstrip (pred,succ) links. We now show the construction of a TC schema for an example scenario of the `SocialNetwork` model.

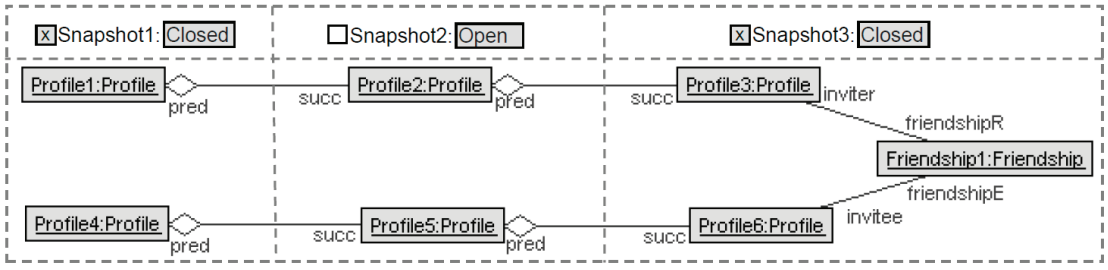

Fig. 3. TC schema of the test scenario.

The description of the user defined test scenario is as follows: There have to exist two user profiles. In the initial state, they are not linked with each other, and in the final state, they are linked with each other through a friendship request. Figure 3 shows the TC schema for this scenario. In `Snapshot1`, there exist two user profiles which are not linked with each other. In `Snapshot3`, the

Profile3 is linked to Profile6 through Invite and Invitee links as well as a Friendship object. For each snapshot, developers have a choice between a so-called *open* snapshot and a *closed* snapshot, and the specification of snapshots will be according to the expected scenario generation. If a snapshot is classified as closed, only the stated links (if any) are allowed between the snapshot objects in a generated test case; if a snapshot is classified as open, other links are allowed between the snapshot objects in a generated test case. More details about the classification of snapshots and their OCL representations are discussed below.

4 Transformation of a TC Schema into an OCL Invariant

As previously stated, a TC schema is comprised of snapshots (application objects and links) and filmstrip links. So during the transformation from a TC schema to an OCL invariant, these elements must be transformed into OCL expressions. The OCL expressions for the filmstrip (pred,succ) links which connect different snapshot objects are directly generated using the succ role name. However, different OCL representations are possible for the snapshots, as they can be classified as an *open* or *closed*. In the *open* case, the mentioned objects and links are fixed by the generated OCL invariant, but it is possible that more links are present in generated test case. However, in the *closed* case, apart from the mentioned objects and links, other possible application links which are not explicitly mentioned, are excluded by the generated OCL invariant. To illustrate the generation of an OCL invariant, we continue with the TC schema shown in Sect. 3. The generated OCL invariant is as follows:

```
context Snapshot inv FirstLastClosedSnapshots:
 Profile.allInstances->exists(p1,p2,...,p6|Set{p1,p2,..,p6}->size()=6 and
  p1.succ=p2 and p2.succ=p3 and p4.succ=p5 and p5.succ=p6 and
  Friendship.allInstances->exists(f1|
   p1.friendshipR.invitee->excludes(p4) and // Snapshot1
   p1.friendshipE.inviter->excludes(p4) and
   p4.friendshipR.invitee->excludes(p1) and
   p4.friendshipE.inviter->excludes(p1) and
   p3.friendshipR->includes(f1) and f1.inviter = p3 and // Snapshot3
   p6.friendshipE->includes(f1) and f1.invitee = p6  ))
```

For the closed snapshots, OCL expressions are generated guaranteeing (a) the absence of links between Profile1 and Profile4 in Snapshot1 and (b) the presence of links between Profile3 and Profile6 in Snapshot3. In the open snapshot, OCL expressions are not generated, as there are no links.

5 Applying the Model Validator for Scenario Generation

The MV uses a given configuration and the generated OCL invariant for setting the sequence of operation calls by fixing (a) attribute values and (b) objects and links that have been left open in the TC schema in order to construct different test cases. We check the feasibility of our approach by transforming a TC schema into an OCL invariant and analyze the generated test cases.

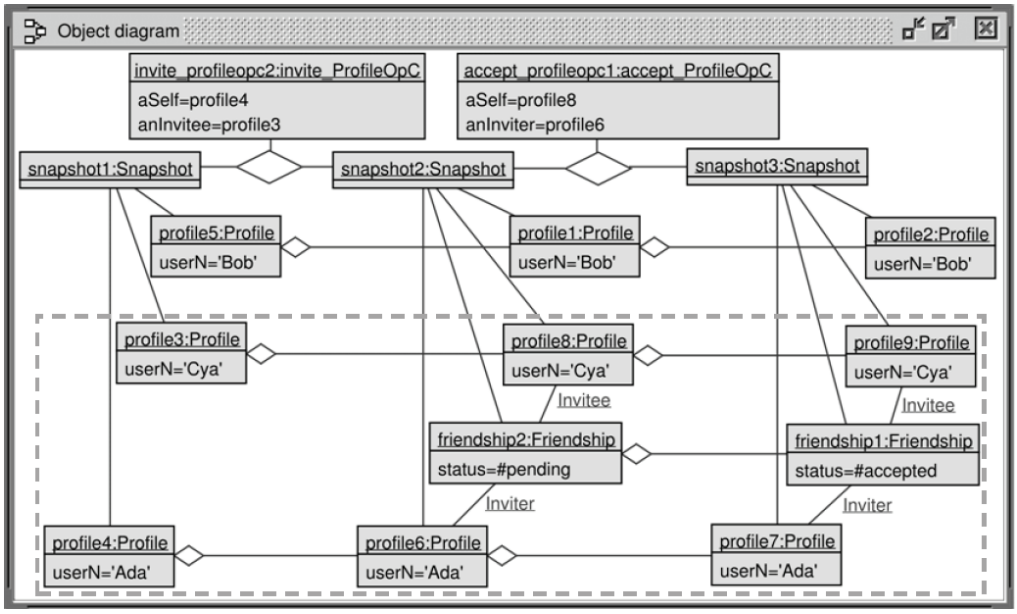

Fig. 4. Automatically generated test case 1

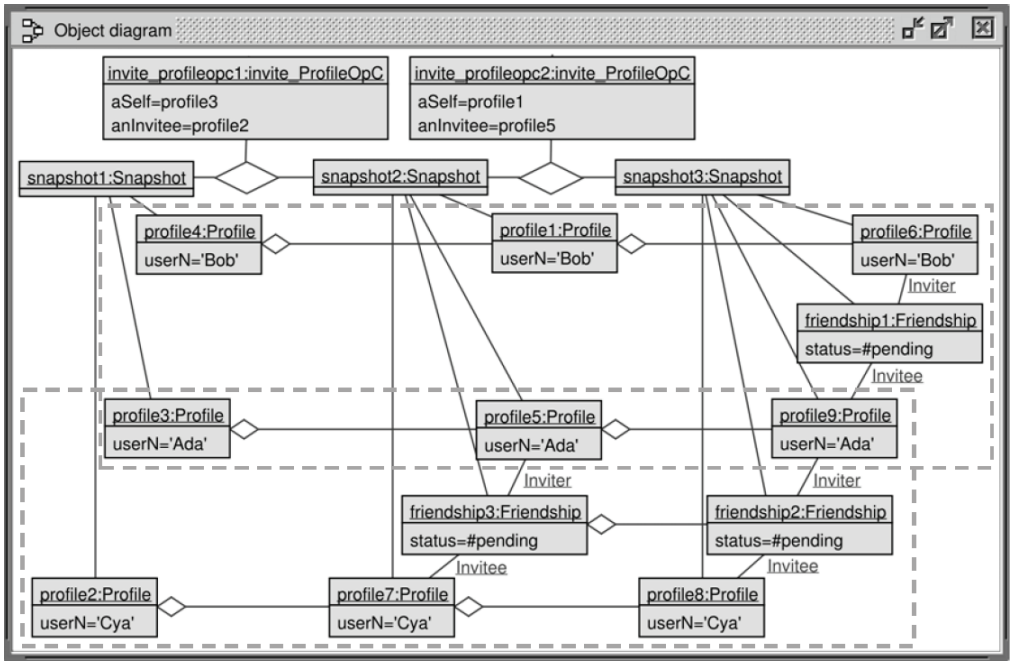

Fig. 5. Automatically generated test case 2

From many generated test cases (filmstrip object diagrams), two are shown in Figs. 4 and 5: one test case is an `invite; accept`; the other one is an `invite; invite`. In the TC schema, one friendship is expected between two profiles. In order to satisfy the scenario at least one `invite` operation call should exist, and another operation call could be an `accept`, `decline` or `invite` as the attribute `status` of the `friendship` object is not specified. In both test cases, the test schema is satisfied (highlighted with dashed rectangles). The objects and links in the open snapshots have been decided by the MV depending on the operation calls.

In both shown test cases, the expected test scenario is precisely generated. These show the successful transformation of an OCL invariant from a given TC

schema, and validate that our concept of distinguishing between open and closed snapshots lead to the desired results. Various other larger models and larger test case schemas have been developed. Due to space limitations we stick to the small demonstration example.

6 Related Work

There are several contributions discussing techniques and approaches for OCL transformation and test case generation. In [1], the authors are using the Semantic Business Vocabulary and Rules (SBVR) to transform constraints written in natural language to OCL statements. In [6], the tool MoMuT::UML is presented to generate fault based test cases for UML state machine models. In [4], the authors describe symbolic scenarios as operation sequences to generate functional test cases. In [2], the authors propose a method to generate test data on a higher-order representation of OCL models. In [3], the tool UMLtoCSP allows a developer to perform verification and validation of a UML/OCL model based on Constraint Logic Programming. In contrast to all these works, our approach is the only one generating OCL constraints automatically from a developer-specified scenario to generate concrete test cases for behavior model validation.

7 Conclusion

This contribution proposed a transformation for automatically generating an OCL invariant from a TC schema. We showed scenario generation using a model validator which constructed valid behavioral scenarios with different sequences of operation calls based on the generated OCL invariant. Future work will consider more options for attribute specification in the transformation, as this should help developers to express a scenario more effectively. More and larger case studies must check the applicability of the approach.

References

1. Bajwa, I.S., Bordbar, B., Lee, M.G.: OCL constraints generation from natural language specification. In: Proc. of the 14th IEEE Int. Enterprise Distributed Object Computing Conf. pp. 204–213. IEEE Computer Society (2010)
2. Brucker, A.D., Krieger, M.P., Longuet, D., Wolff, B.: A specification-based test case generation method for UML/OCL. In: Models in Software Engineering - Workshops at MODELS. pp. 334–348. Springer (2010)
3. Cabot, J., Clarisó, R., Riera, D.: UMLtoCSP: a tool for the formal verification of UML/OCL models using constraint programming. In: 22nd IEEE/ACM International Conference on Automated Software Engineering. pp. 547–548. ACM (2007)
4. Castillos, K.C., Dadeau, F., Julliand, J.: Scenario-based testing from UML/OCL behavioral models - application to POSIX compliance. STTT 13(5), 431–448 (2011)
5. Gogolla, M., Hamann, L., Hilken, F., Kuhlmann, M., France, R.B.: From Application Models to Filmstrip Models: An Approach to Automatic Validation of Model Dynamics. In: Proc. Modellierung. pp. 273–288. GI, LNI 225 (2014)
6. Krenn, W., Schlick, R., Tiran, S., Aichernig, B.K., Jöbstl, E., Brandl, H.: MoMut: : UML model-based mutation testing for UML. In: 8th IEEE Int. Conf. on Software Testing, Verification and Validation. pp. 1–8. IEEE Computer Society (2015)