CoSTest: A tool for Validation of Requirements at Model Level

Maria Fernanda Granda[1]
Department of Computer Science
University of Cuenca
Cuenca, Ecuador
fernanda.granda@ucuenca.edu.ec

Nelly Condori-Fernández
Department of Computer Science
University of A Coruña, Spain
Vrije Universitei Amsterdam, The Netherlands
n.condori.fernandez@udc.es
n.condori-fernandez@vu.nl

Tanja E.J. Vos, Oscar Pastor
[1]PROS Research Center
Universitat Politècnica de València
Valencia, Spain
{tvos, opastor}@dsic.upv.es

***Abstract*—We present CoSTest, a tool that supports the validation of Conceptual Schemas by using testing. The tool implements techniques for transforming instantiations from a Requirements Model into test case implementations by supporting a Model-driven architecture.**

***Index Terms*—Conceptual Schema Validation, Requirements-based testing, Model-driven validation, Conceptual Schema Testing.**

I. Introduction

Requirements Engineering and Testing both aim to support the development of software products that meet stakeholder's expectations regarding functionality and quality at different stages of the software development life cycle. However, connecting requirements and testing processes is still a significant challenge [1], because requires a clear (verifiable) specification of requirements and practices that support this alignment. According to Mac Millan Dictionary's definition, 'align' is when 'activities or systems are organised so that they match or fit well together'. A weak alignment of requirements with the test cases may lead to problems in delivering the software product on time with the right quality. For example, if requirements changes are agreed without involving testers and without updating the requirements specification, the changed functionality is either not verified or incorrectly verified.

In order to overcome these issues, we propose a novel solution that aims at combining the testing process with automated reasoning procedures in a model-driven development environment to generate the test cases from a Requirements model based on Communication Analysis [2]. In [3], we proposed the model-driven approach for testing Conceptual Schemas (CS) based on UML class diagram (CD). In this paper we

The main purpose of the tool is the validation of CS according to stakeholders' requirements. A UML CD-based CS can be tested if the CS is specified in an executable form, for example in the Action Language for Foundational UML (ALF [4]), which is supported as a standard. Out tool aims the validation of two CS quality goals [5]: the correctness (covers both syntactic correctness -right syntax or well-formedness, and semantic correctness -right meaning and relations relative to the knowledge about the domain) and completeness (i.e. all the necessary information is defined in the CS).

Our tool may be used by testers/modellers/analysts in any development phase of a CS based on UML class diagram (CD). For example, as part of the test-last validation (i.e. correctness and completeness are checked by testing after the CS definition) or test-first development, in which the elicitation and definition is driven on a set of test cases.

In the next section, we briefly describe our CoSTest tool. In Appendix, we explain what exactly will be demonstrated as well as the expected participants in the demonstration.

II. The CoSTest Tool

Figure 1 shows an overview of the three main functionalities of the CoSTest: Test Suite Generation, CSUT generation and Test Execution. Each phase depicts the CoSTest artefacts, processes, inputs and outputs and modeller/tester interactions. As the names suggest, CoSTest processes are done automatically by our tool whereas the modeller/tester activities are done manually. The numbered ovals represent activities and directed edges to and from processes represent the consumption and production of artefacts, respectively. In the next, we describe the different steps of the CoSTest tool.

1. ***Generation of the Test Model***. CoSTest analyses the structure of the Requirements Model (RM) by automatically traversing all the RM nodes (event sequences) and extracting all the Test Model elements and their properties.
2. ***Generation of the Test Scenarios Model***. CoSTest computes the possible test scenarios (based on event sequence) and generates a model of test scenarios.
3. ***Concretization of Test Values***. The next step is to concretize the variables of the test cases. The tester can (i) recover a variable list from the test model and generate values automatically way from the example values specified in the RM, or (ii) concretize manually by introducing values for each variable.
4. ***Generation of Abstract Test Cases***. Transform each test scenario into Test cases scripts (ALF script).
5. ***Choosing the tests types.*** The tester can select between two types of test cases such as (i) partial (only positive test cases) ii) complete, which adds test cases with some test items such as values out of range, constraint violations, and, unique value violation for class variables.

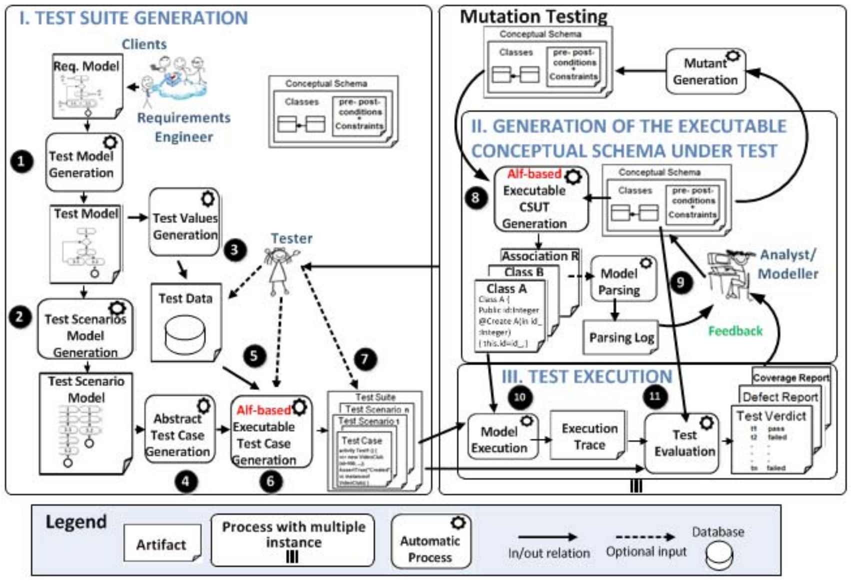


Fig 1. Overview of the CoSTest

6. ***Generation of Executable Test Cases.*** CoSTest automatically transforms each test scenario with abstract test cases into parameterized ALF scripts. CoSTest then generates the executable and concrete test cases, including the test data, test objective and an expected output (oracle based on test case type) that is used to validate the CS requirements.
7. ***Prioritizing and select test cases***. In order to know the test types that should be prioritized, CoSTest allows to mutate CS and to evaluate the effectiveness [6] in killing mutants of the test cases generated by CoSTest
8. ***Generation of Executable CS under test (CSUT)***. CoSTest translates the UML CD-based CS into an executable format (ALF scripts) for its execution.
9. ***Parsing the CSUT.*** The CSUT should be parser before the testing process for verifying that it is well-formed.
10. ***Executing the CSUT.*** Test cases are executed on the executable CS and the output is compared to the oracle.
11. ***Test Evaluation***. CoSTest generates a report in which the executed test cases are classified as passed, failed or inconclusive. A report including times, test cases coverage and detected faults is generated.

III. Conclusions and Future Work

The CoSTest tool is a research prototype that takes a requirements model based on Communicational Analysis as input to generate test cases and execute them against UML CD-based CSs for detecting defects of correctness and completeness. We also introduced briefly that this tool mutates the CSs to evaluate the effectiveness of the tests cases generated by CoSTest and to prioritize them.

In the future, since the requirements model based on Communication Analysis is time consuming now, we need to find some efficient way to solve the problem, so that it can be more feasible to specify the requirements of a large scale of modellers. In addition, the reliability of the tool would be evaluated with large-scale users and CSs.

Acknowledgment

This work is supported by SENESCYT of the Republic of Ecuador, Spanish Ministry of Economy, Industry and Competitiveness with the PGE and FEDER Projects: TIN2016-78011-C4-1-R and TIN2013-46238-C4-3-R.

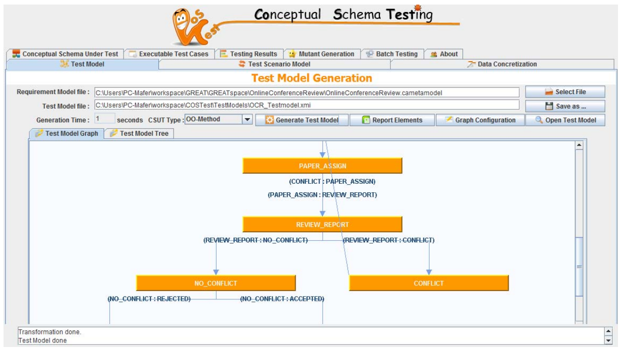
References

1. Bjarnason, E., Runeson, P., Borg, M., Unterkalmsteiner, M., Engström, E., Regnell, B., Sabaliauskaite, G., Loconsole, A., Gorschek, T., Feldt, R.: Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empir. Softw. Eng. 19, 1809–1855 (2014).
2. Granda, M.F., Condori-Fernandez, N., Vos, T.E.J., Pastor, O.: Towards the automated generation of abstract test cases from requirements models. In: 1st International Workshop on Requirements Engineering and Testing. pp. 39–46. IEEE, Karlskrona, Sweden (2014).
3. Granda, M.F.: Testing-Based Conceptual Schema Validation in a Model- Driven Environment. In: CAiSE 2013 Doctoral Consortium. , Valencia (2013).
4. Object Management Group: Action Language for Foundational UML (ALF). (2013).
5. Mohagheghi, P., Dehlen, V., Neple, T.: Definitions and approaches to model quality in model-based software development - A review of literature. Inf. Softw. Technol. 51, 1646–1669 (2009).
6. Granda, M.F., Condori-Fernández, N., Vos, T.E.J., Pastor, Ó.: Effectiveness Assessment of an Early Testing Technique using Model-Level Mutants. In: 21st International Conference on Evaluation and Assessment in Software Engineering. , Karlskrona, Sweden (2017).

APPENDIX

The demo of CoSTest aims at allowing participants to generate test cases from a Requirements model and find some injected faults with the purpose of showing the main tool functionalities. Anyone (e.g. analysts, conceptual modelling researchers, testers, students and practitioners) considering or planning to conduct validation of UML CD-based CS using a tool, as well as anyone interested in taking a systematic sound snapshot the CS validation practice are expected as visitors in this demonstration. Participants will get a general overview about of (1) the test suite generation, (2) the executable CSUT generation, and (3) the test execution. For this demonstration, we will use a simple CS of an online conference review (OCR).

The user interface of the tool is composed by eight tabs (see Fig 2). Each tab corresponds to one of the above main functionalities of CoSTest. As seen in Fig 2, the derivation strategy of the test cases starts with loading the requirement model (see Fig 3) based on the *Test Model* tab.



Test Model Generation

Requirement Model file : C:\Users\PC-Mafer\workspace\GREAT\GREATspace\OnlineConferenceReview\OnlineConferenceReview.cametamodel

Test Model file : C:\Users\PC-Mafer\workspace\COSTest\TestModels\OCR_Testmodel.xmi

Generation Time : 1 seconds CSUT Type : OO-Method

```
Transformation done.
Test Model done
```

Fig 2. Test Model Generation

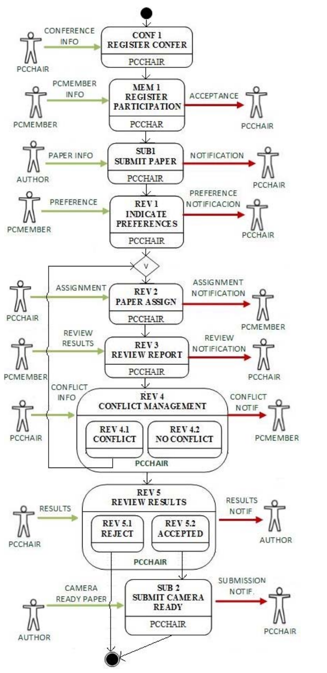
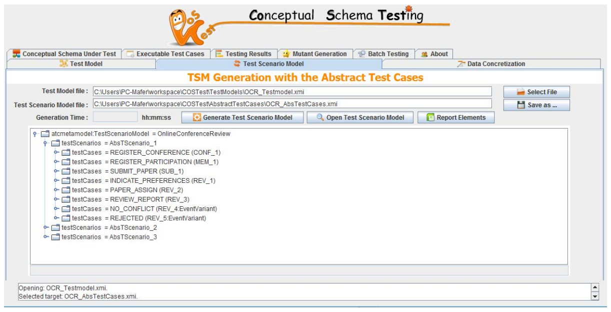


Fig 3.Excerpt of a Requirements model for our OCR example

Once the model has been selected, the requirement model is transformed into the test model (see Fig 2). Then, on the next tab (i.e. *Test Scenario Model)*, the test scenario model is generated with the different abstract test cases (show in Fig 4).



TSM Generation with the Abstract Test Cases

Test Model file : C:\Users\PC-Mafer\workspace\COSTest\TestModels\OCR_Testmodel.xmi

Test Scenario Model file : C:\Users\PC-Mafer\workspace\COSTest\AbstractTestCases\OCR_AbsTestCases.xmi

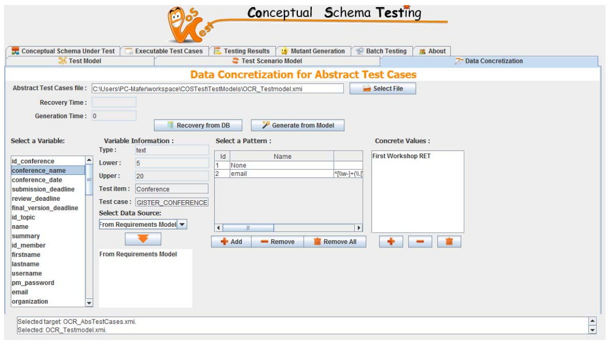
Generation Time : hh:mm:ss

```
atcmetamodel:TestScenarioModel = OnlineConferenceReview
  testScenarios = AbsTScenario_1
    testCases = REGISTER_CONFERENCE (CONF_1)
    testCases = REGISTER_PARTICIPATION (MEM_1)
    testCases = SUBMIT_PAPER (SUB_1)
    testCases = INDICATE_PREFERENCES (REV_1)
    testCases = PAPER_ASSIGN (REV_2)
    testCases = REVIEW_REPORT (REV_3)
    testCases = NO_CONFLICT (REV_4:EventVariant)
    testCases = REJECTED (REV_5:EventVariant)
  testScenarios = AbsTScenario_2
  testScenarios = AbsTScenario_3
```

```
Opening: OCR_Testmodel.xmi.
Selected target: OCR_AbsTestCases.xmi.
```

Fig 4. Generation of the test scenario model

After that, on the *Data Concretization* tab, we are able to setup a data base by creating, reading, updating and deleting test data values for concretize the test cases. A variable may be concretized with values by using (i) the requirement model, (ii) a manual entry, or (iii) a web-based generation. Fig 5 shows the main components (e.g. list of variables, patterns, concrete values) of this screen.



Data Concretization for Abstract Test Cases

Abstract Test Cases file : C:\Users\PC-Mafer\workspace\COSTest\TestModels\OCR_Testmodel.xmi

Recovery Time :

Generation Time : 0

Select a Variable: id_conference, conference_name, conference_date, submission_deadline, review_deadline, final_version_deadline, id_topic, name, summary, id_member, firstname, lastname, username, pm_password, email, organization

Variable Information : Type : text; Lower : 5; Upper : 20; Test item : Conference; Test case : GISTER_CONFERENCE

Select Data Source: From Requirements Model

From Requirements Model

Select a Pattern :

Id	Name	
1	None	
2	email	^[\\w-]+(\\.[

Concrete Values : First Workshop RET

```
Selected target: OCR_AbsTestCases.xmi.
Selected: OCR_Testmodel.xmi.
```

Fig 5. Screenshot for the data concretization in the CoSTest tool

Fig 6 shows the CS of our example with five defects injected for this demonstration: (a) missing association, (b) unnecessary parameter, (c) wrong association, (d) wrong constraint and (e) missing constraint.

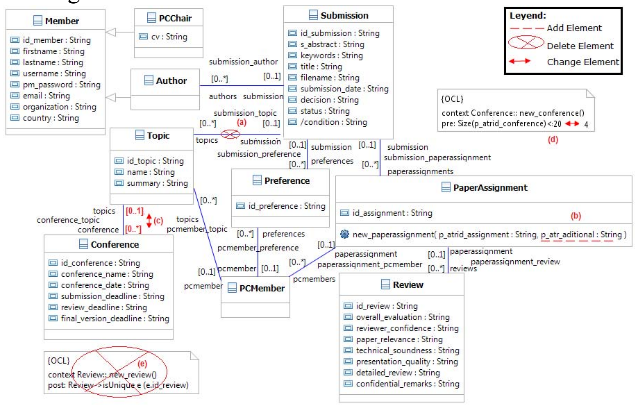
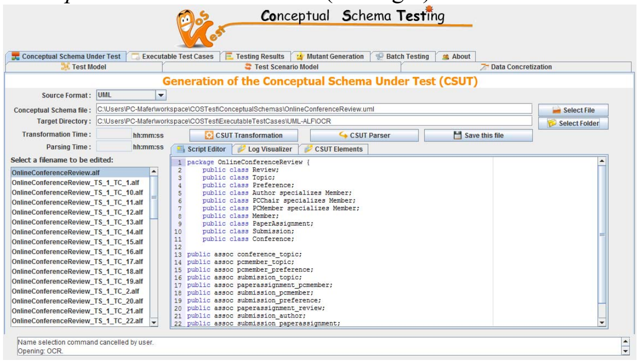


Fig 6. UML class diagram for OCR CS

This model is selected in the next tab of the tool named *Conceptual Schema Under Test* (see Fig 7).



Generation of the Conceptual Schema Under Test (CSUT)

Source Format : UML
Conceptual Schema file : C:\Users\PC-Mafer\workspace\COSTest\ConceptualSchemas\OnlineConferenceReview.uml
Target Directory : C:\Users\PC-Mafer\workspace\COSTest\ExecutableTestCases\UML-ALF\OCR
Transformation Time : hh:mm:ss
Parsing Time : hh:mm:ss
Select a filename to be edited:

OnlineConferenceReview.alf
OnlineConferenceReview_TS_1_TC_1.alf
OnlineConferenceReview_TS_1_TC_10.alf
OnlineConferenceReview_TS_1_TC_11.alf
OnlineConferenceReview_TS_1_TC_12.alf
OnlineConferenceReview_TS_1_TC_13.alf
OnlineConferenceReview_TS_1_TC_14.alf
OnlineConferenceReview_TS_1_TC_15.alf
OnlineConferenceReview_TS_1_TC_16.alf
OnlineConferenceReview_TS_1_TC_17.alf
OnlineConferenceReview_TS_1_TC_18.alf
OnlineConferenceReview_TS_1_TC_19.alf
OnlineConferenceReview_TS_1_TC_2.alf
OnlineConferenceReview_TS_1_TC_20.alf
OnlineConferenceReview_TS_1_TC_21.alf
OnlineConferenceReview_TS_1_TC_22.alf

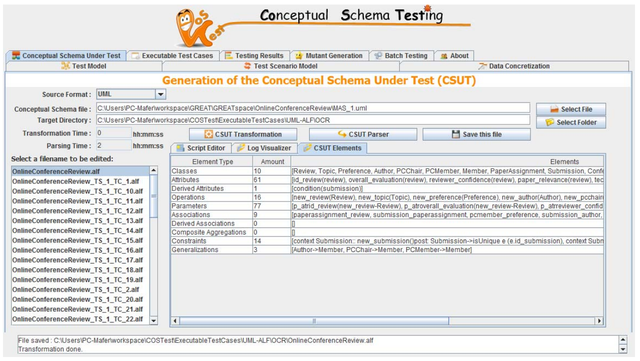
```
package OnlineConferenceReview {
    public class Review;
    public class Topic;
    public class Preference;
    public class Author specializes Member;
    public class PCChair specializes Member;
    public class PCMember specializes Member;
    public class Member;
    public class PaperAssignment;
    public class Submission;
    public class Conference;

public assoc conference_topic;
public assoc pcmember_topic;
public assoc pcmember_preference;
public assoc submission_topic;
public assoc paperassignment_pcmember;
public assoc submission_pcmember;
public assoc submission_preference;
public assoc paperassignment_review;
public assoc submission_author;
public assoc submission_paperassignment;
```

```
Name selection command cancelled by user.
Opening: OCR.
```

Fig 7. Screenshot for generating an executable CSUT in the CoSTest tool

After this translation, the user can parser the CSUT to verify that it is well-formed as well as to report a list of translated elements (see Fig 8). Fig 9 shows a defect founded by the parser in our example.



Generation of the Conceptual Schema Under Test (CSUT)

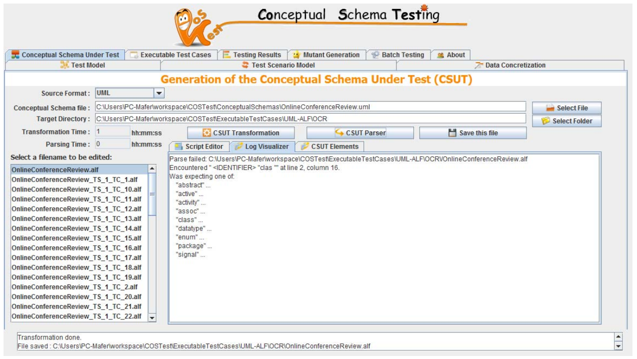
Source Format : UML
Conceptual Schema file : C:\Users\PC-Mafer\workspace\GREAT\GREATspace\OnlineConferenceReview\MAS_1.uml
Target Directory : C:\Users\PC-Mafer\workspace\COSTest\ExecutableTestCases\UML-ALF\OCR
Transformation Time : 0 hh:mm:ss
Parsing Time : 2 hh:mm:ss
Select a filename to be edited:

OnlineConferenceReview.alf
OnlineConferenceReview_TS_1_TC_1.alf
OnlineConferenceReview_TS_1_TC_10.alf
OnlineConferenceReview_TS_1_TC_11.alf
OnlineConferenceReview_TS_1_TC_12.alf
OnlineConferenceReview_TS_1_TC_13.alf
OnlineConferenceReview_TS_1_TC_14.alf
OnlineConferenceReview_TS_1_TC_15.alf
OnlineConferenceReview_TS_1_TC_16.alf
OnlineConferenceReview_TS_1_TC_17.alf
OnlineConferenceReview_TS_1_TC_18.alf
OnlineConferenceReview_TS_1_TC_19.alf
OnlineConferenceReview_TS_1_TC_2.alf
OnlineConferenceReview_TS_1_TC_20.alf
OnlineConferenceReview_TS_1_TC_21.alf
OnlineConferenceReview_TS_1_TC_22.alf

Element Type	Amount	Elements
Classes	10	[Review, Topic, Preference, Author, PCChair, PCMember, Member, PaperAssignment, Submission, Confe
Attributes	61	[id_review(review), overall_evaluation(review), reviewer_confidence(review), paper_relevance(review), tec
Derived Attributes	1	[condition(submission)]
Operations	16	[new_review(Review), new_topic(Topic), new_preference(Preference), new_author(Author), new_pcchair
Parameters	77	[p_atrid_review(new_review-Review), p_atroverall_evaluation(new_review-Review), p_atrreviewer_confid
Associations	9	[paperassignment_review, submission_paperassignment, pcmember_preference, submission_author,
Derived Associations	0	[]
Composite Aggregations	0	[]
Constraints	14	[context Submission:: new_submission()post: Submission->isUnique e (e.id_submission), context Subn
Generalizations	3	[Author->Member, PCChair->Member, PCMember->Member]

```
File saved : C:\Users\PC-Mafer\workspace\COSTest\ExecutableTestCases\UML-ALF\OCR\OnlineConferenceReview.alf
Transformation done.
```

Fig 8. Report of translated elements



Generation of the Conceptual Schema Under Test (CSUT)

Source Format : UML
Conceptual Schema file : C:\Users\PC-Mafer\workspace\COSTest\ConceptualSchemas\OnlineConferenceReview.uml
Target Directory : C:\Users\PC-Mafer\workspace\COSTest\ExecutableTestCases\UML-ALF\OCR
Transformation Time : 1 hh:mm:ss
Parsing Time : 0 hh:mm:ss
Select a filename to be edited:

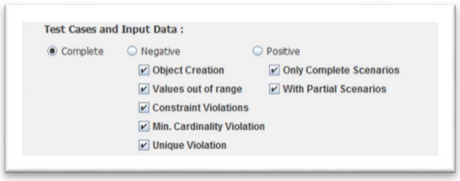
OnlineConferenceReview.alf
OnlineConferenceReview_TS_1_TC_1.alf
OnlineConferenceReview_TS_1_TC_10.alf
OnlineConferenceReview_TS_1_TC_11.alf
OnlineConferenceReview_TS_1_TC_12.alf
OnlineConferenceReview_TS_1_TC_13.alf
OnlineConferenceReview_TS_1_TC_14.alf
OnlineConferenceReview_TS_1_TC_15.alf
OnlineConferenceReview_TS_1_TC_16.alf
OnlineConferenceReview_TS_1_TC_17.alf
OnlineConferenceReview_TS_1_TC_18.alf
OnlineConferenceReview_TS_1_TC_19.alf
OnlineConferenceReview_TS_1_TC_2.alf
OnlineConferenceReview_TS_1_TC_20.alf
OnlineConferenceReview_TS_1_TC_21.alf
OnlineConferenceReview_TS_1_TC_22.alf

```
Parse failed: C:\Users\PC-Mafer\workspace\COSTest\ExecutableTestCases\UML-ALF\OCR\OnlineConferenceReview.alf
Encountered " <IDENTIFIER> "clas "" at line 2, column 16.
Was expecting one of:
    "abstract" ...
    "active" ...
    "activity" ...
    "assoc" ...
    "class" ...
    "datatype" ...
    "enum" ...
    "package" ...
    "signal" ...
```

```
Transformation done.
File saved : C:\Users\PC-Mafer\workspace\COSTest\ExecutableTestCases\UML-ALF\OCR\OnlineConferenceReview.alf
```

Fig 9. An example of the parser results

When the test cases are generated, the user can choose the kind of test cases to be generated (shows in Fig 10).

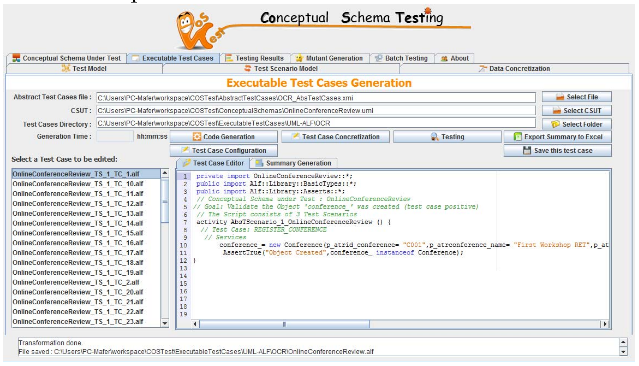


Test Cases and Input Data :

[✓ Complete] Negative Positive
- [✓] Object Creation
- [✓] Values out of range
- [✓] Constraint Violations
- [✓] Min. Cardinality Violation
- [✓] Unique Violation
- [✓] Only Complete Scenarios
- [✓] With Partial Scenarios

Fig 10. Configuration of kind of the test cases

Fig 11 shows an example of test case generated by CoSTest for our example.



Executable Test Cases Generation

Abstract Test Cases file : C:\Users\PC-Mafer\workspace\COSTest\AbstractTestCases\OCR_AbsTestCases.xmi
CSUT : C:\Users\PC-Mafer\workspace\COSTest\ConceptualSchemas\OnlineConferenceReview.uml
Test Cases Directory : C:\Users\PC-Mafer\workspace\COSTest\ExecutableTestCases\UML-ALF\OCR
Generation Time : hh:mm:ss
Select a Test Case to be edited:

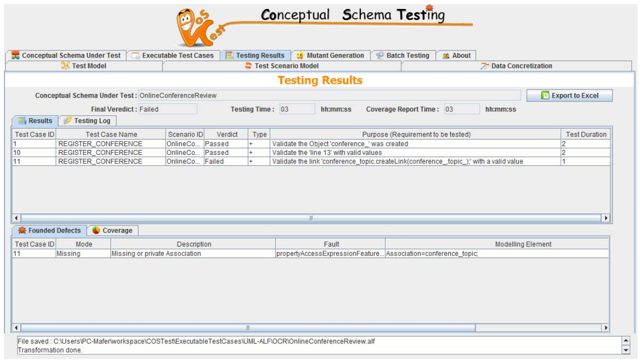
OnlineConferenceReview_TS_1_TC_1.alf
OnlineConferenceReview_TS_1_TC_10.alf
OnlineConferenceReview_TS_1_TC_11.alf
OnlineConferenceReview_TS_1_TC_12.alf
OnlineConferenceReview_TS_1_TC_13.alf
OnlineConferenceReview_TS_1_TC_14.alf
OnlineConferenceReview_TS_1_TC_15.alf
OnlineConferenceReview_TS_1_TC_16.alf
OnlineConferenceReview_TS_1_TC_17.alf
OnlineConferenceReview_TS_1_TC_18.alf
OnlineConferenceReview_TS_1_TC_19.alf
OnlineConferenceReview_TS_1_TC_2.alf
OnlineConferenceReview_TS_1_TC_20.alf
OnlineConferenceReview_TS_1_TC_21.alf
OnlineConferenceReview_TS_1_TC_22.alf
OnlineConferenceReview_TS_1_TC_23.alf

```
private import OnlineConferenceReview::*;
public import Alf::Library::BasicTypes::*;
public import Alf::Library::Asserts::*;
// Conceptual Schema under Test : OnlineConferenceReview
// Goal: Validate the Object 'conference_' was created (test case positive)
// The Script consists of 3 Test Scenarios
activity AbsTScenario_1_OnlineConferenceReview () {
 // Test Case: REGISTER_CONFERENCE
  // Services
       conference_= new Conference(p_atrid_conference= "C001",p_atrconference_name= "First Workshop RET",p_at
        AssertTrue("Object Created",conference_ instanceof Conference);
}
```

```
Transformation done.
File saved : C:\Users\PC-Mafer\workspace\COSTest\ExecutableTestCases\UML-ALF\OCR\OnlineConferenceReview.alf
```

Fig 11. Generation of the executable test cases

Fig 12 shows the result of the execution of an example of CoSTest's test suite on *Testing Results* Tab, in which one test case has had problems in its execution, and therefore the global verdict is Inconclusive.



Testing Results

Conceptual Schema Under Test : OnlineConferenceReview
Final Veredict : Failed
Testing Time : 03 hh:mm:ss
Coverage Report Time : 03 hh:mm:ss

Test Case ID	Test Case Name	Scenario ID	Verdict	Type	Purpose (Requirement to be tested)	Test Duration
1	REGISTER_CONFERENCE	OnlineCo...	Passed	+	Validate the Object 'conference_' was created	2
10	REGISTER_CONFERENCE	OnlineCo...	Passed	+	Validate the 'line 13' with valid values	2
11	REGISTER_CONFERENCE	OnlineCo...	Failed	+	Validate the link 'conference_topic.createLink(conference_,topic_);' with a valid value	1

Test Case ID	Mode	Description	Fault	Modelling Element
11	Missing	Missing or private Association	propertyAccessExpressionFeature...	Association=conference_topic;

```
File saved : C:\Users\PC-Mafer\workspace\COSTest\ExecutableTestCases\UML-ALF\OCR\OnlineConferenceReview.alf
Transformation done.
```

Fig 12. Test execution Report in the CoSTest tool

After that, the user must change the CS or the requirements in order to make it correct and complete. Then, the valiation process should be re-run to ensure that the changes are valid.

Finally, CoSTest tool is publicly available in the project website https://staq.dsic.upv.es/webstaq/costest.html, as well as other CS examples (e.g. Medical Treatment, Sudoku Game, Expense Report, and Photography Agency) and complementary documentation about the tool.

Demo: https://youtu.be/YUDGsm634rQ
Email: fernanda.granda@ucuenca.edu.ec