Messir: A Text-First DSL-Based Approach for UML Requirements Engineering (Tool Demo)

Benoît Ries, Alfredo Capozucca and Nicolas Guelfi

University of Luxembourg, Esch-sur-Alzette, Luxembourg

Context

Due to the need for (and absence of) an integrated requirements engineering tool centered on a *textual specification language*, providing rich coverage of UML, report generation, and formal simulation to be used by our students at University of Luxembourg, in our software engineering project-based lectures, we have started to develop the Excalibur workbench and Messir DSLs.

1. Messir & Excalibur

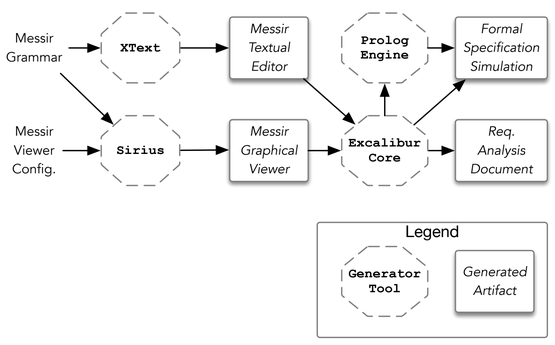

Figure 1:Excalibur Architecture

Messir approach's main characteristics:

- Scientific approach
- Focusing on textual specifications
- Offering a flexible requirements specification language
- UML-based that provides
  - an improved use-case modeling phase
  - environment and conceptual modeling
  - a declarative executable operation language
  - a test specification language

Excalibur [1] has been developed by the authors as an extension to Eclipse combining the 4 tools, as shown in Figure 1:

- *XText* converting an EBNF-like grammar into a full-fledge textual editor, including syntax highlighting, auto-completion and validation rules.
- *Sirius* displaying the textual files written with our DSL in UML-like graphical notations.
- *Excalibur Core* is implemented in Java and XTend, providing: a dedicated *Outline* allowing to navigate through the specification elements in a tree-view style; a *Requirements analysis document* generator; a *Formal specification simulation* code generator.
- a *Simulator*, based on the SICStus prolog engine, interpreting the prolog code generated and displaying the simulation results in Eclipse as tabular-tree view.

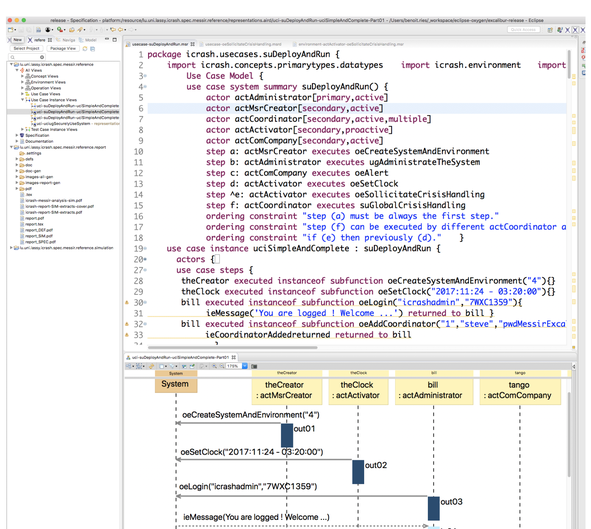

Figure 2:Excalibur Workbench

2. Messir Textual DSLs

Messir Constraint Language

- declarative specification of operations
- syntax inspired from OCL
- semantics defined as a manual translation to prolog
- covered concepts include: navigation, conditional expressions, messages sending

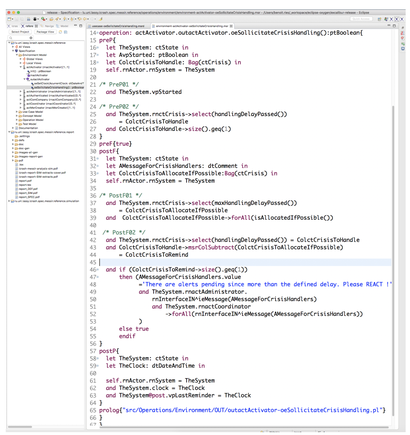

Figure 3:Messir Constraint Language

Messir Documentation Language

- complementary textual language
- natural language descriptions used during report generation
- allow documenting Messir specification elements
- allow documenting Excalibur views

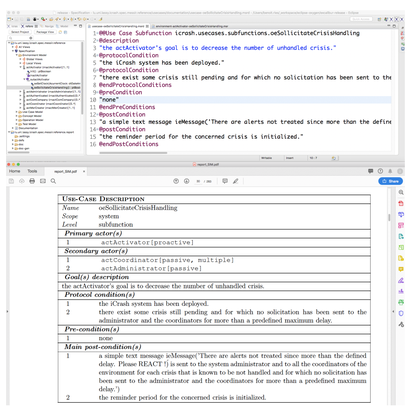

Figure 4:MessirDoc Language

Validation Rules

- A number of syntactical validation rules are generated automatically by the XText framework based on the Messir DSL grammar.
- We have implemented 50 additional runtime validation rules used as educational means:
- *warning rules* are meant to let the end-user know about future steps to be done, or particular aspects of the methodology to not be overlooked.
- *error rules* are meant to block the end-user in his requirements specification process.

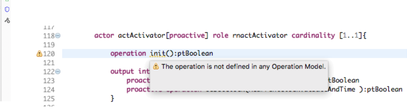

Figure 5:Warning Validation Rule

Students Feedback

Our students surveys on the lectures using Messir/Excalibur resulted, out of 90+ answers, in a majority of students agreeing both on "recommending the lectures to others", and that "the learning resources met their needs". Positive comments were "the integrated hands-on approach" and "the report generation". Negative comments were mostly about "the actual presence of bugs in the tool"...

Conclusion

This poster presents our solution for a requirements engineering tool, named Excalibur, supporting our methodology, that is centered on the Messir *textual* DSLs having typical features of textual editors (thanks to XText) for which we have developped 50 custom validation rules to guide the analyst during the requirements elicitation phase.
Excalibur implements three generative techniques to make the best use of the textual requirements specifications, firstly by generating read-only views in a UML-style (thanks to Sirius), secondly by generating an extensible requirements analysis document compiling all textual and graphical requirements information; lastly by generating a partial prolog implementation supporting the DSL metamodel for simulation purpose.

Work Summary

Messir is a scientific approach, yet flexible, for the specification of UML requirements. It is supported by the Excalibur tool, used for software engineering education, since 2012 in numerous institutions for project-based lectures.
The Excalibur tool provides as an integrated worbench, the possibility to describe rich textual UML requirements & analysis specifications, to generate a structured report in LaTeX, *and* to formally simulate, with a prolog engine, the test cases specified in the requirements.

3. Generative Techniques

View Generation

- read-only views
- *illustrate certain aspects* of the textual requirements
- supported views are : use-case, use-case instance, concept model, environment model, operation scope, test case, test-case instance
- their concrete syntax is based on UML use-case, class and sequence diagrams
- these views are integrated in the requirements analysis document during the report generation phase.

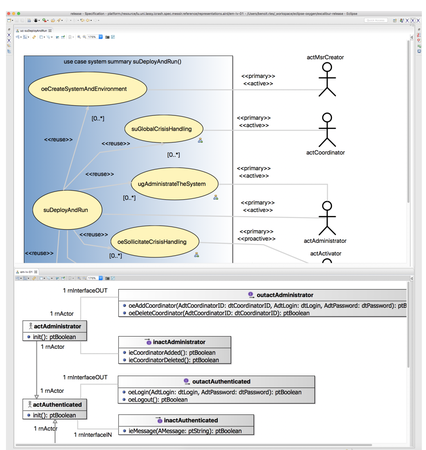

Figure 6:Some Excalibur Views

Report Generation

- Takes as input :
  - requirements elements in Messir
  - documentation in MessirDoc of the elements and views
  - actual views created in the requirements project
- Generates a LaTeX document to be completed, e.g. with introduction, conclusion, etc.
- Flexible process :
  - *definition-level* mainly contains natural language descriptions and documented views.
  - *specification-level* additionally includes declarative operation specifications in MCL.
  - *simulation-level* additionally includes prolog code of the operations and types semantics.

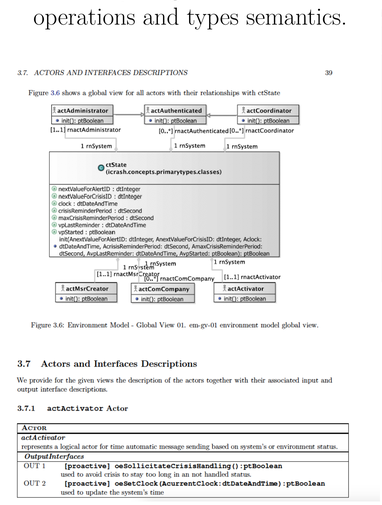

Figure 7:Generated Report Extract

Simulation

- Takes as input
  - test cases and instances of test cases specified in Messir
- Generates a prolog simulation project containing
  - *MESSAM prolog code*: the MESSir Abstract Machine, which is our prolog implementation of the Messir DSL metamodel
  - and *Types specification*: all specified types and actors in prolog compliantly with MESSAM.
- Some other parts are completed manually :
  - the *operation pre/post conditions*
  - and the *test cases specification*

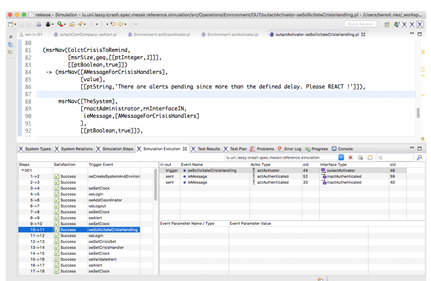

Figure 8:Excalibur Simulation View and Sample Prolog Code

References

[1] Messir and Excalibur website.
`https://messir.uni.lu`.

[2] N. Mahmud, C. Seceleanu, and O. Ljungkrantz.
ReSA Tool: Structured Requirements Specification and SAT-based Consistency-checking.
pages 1737–1746, October 2016.

[3] A. da Silva, S. Vlajic, S. Lazarevic, I. Antovic, V. Stanojevic, and M. Milic.
Preliminary Experience Using JetBrains MPS to Implement a Requirements Specification Language.
In *9th International Conference on the Quality of Information and Communications Technology*, pages 134–137, Guimaraes, Portugal, 2014. IEEE.

[4] D. Savic, S. Vlajić, S. Lazarević, I. Antović, S. Vojislav, M. Milić, and A. Silva.
Use case specification using the SilabReq domain specific language.
Computing and Informatics, 34:877–910, 2015.

[5] V. Hoffmann, H. Lichter, A. Nyßen, and A. Walter.
Towards the Integration of UML- and textual Use Case Modeling.
The Journal of Object Technology, 8(3):85, 2009.

[6] Ed Seidewitz.
A Development Environment for the Alf Language Within the MagicDraw UML Tool (Tool Demo).
In *10th ACM SIGPLAN International Conference on Software Language Engineering*, SLE 2017, pages 217–220, Vancouver, BC, Canada, 2017. ACM.

Acknowledgements

The authors would like to thank all the students from University of Luxembourg (Luxembourg), University of Rosario (Argentina), Innopolis University (Russia) and St Petersburg Polytechnic University (Russia) for their help and support in the usage of the tool and bug reporting that greatly helped reaching more stable releases.

excalibur MESSIR METHOD APPLICATION — OPEN SOURCE

MESSIR METHODOLOGY

UNIVERSITÉ DU LUXEMBOURG