Traceability Support in OpenModelica using Open Services for Lifecycle Collaboration (OSLC)

Alachew Mengist Adrian Pop Adeel Asghar Peter Fritzson

PELAB – Programming Environment Lab, Department of Computer Science, Linköping University, Sweden,
{alachew.mengist, adrian.pop, adeel.asghar, peter.fritzson}@liu.se

Abstract

A common situation in industry is that a system model is composed of several sub-models which may have been developed using different tools. The quality and effectiveness of large scale system modeling heavily depends on the underlying tools used for different phases of the development lifecycle. Available modeling and simulation tools support different operations on models, such as model creation, model simulation, FMU export, model checking, and code generation. Seamless tracing of the requirements and associating them with the models and the simulation results in the context of different modeling tools is becoming increasingly important. This can be used to support several activities such as impact analysis, component reuse, verification, and validation. However, due to the lack of interoperability between tools it is often difficult to use such tools in combination. Recently, the OSLC specification has emerged for integrating different lifecycle tools using linked data. In this paper we present new work on traceability support in OpenModelica where the traceability information is exchanged with other lifecycle tools through a standardized interface and format using OSLC. In particular, OpenModelica supports automatic recording and tracing of modeling activities such as creation, modification, and destruction of models, import model description XML, export of FMUs, and creation of simulation results.

Keywords: OpenModelica, traceability, OSLC, tool interoperability, tool integration, model management, Modelica

1 Introduction

Modeling and simulation tools have become increasingly used for industrial applications. Such tools support different activities in the modeling and simulation lifecycle, like specifying requirements, model creation, model simulation, Functional Mock-up Unit (FMU) export (Blochwitz et al, 2011; FMI-Standard.org, 2014), model checking, and code generation. However, the heterogeneity and complexity of modern industrial products often require special purpose modeling and simulation tools for different phases of the development life cycle. Seamless exchange of models between different modeling tools is needed in order to integrate all the parts of a complex product model throughout the development life cycle.

During the past decade, the Open Services for Lifecycle Collaboration (OSLC) specifications (Open-services.net, 2008) have emerged for integrating development lifecycle tools using Linked Data (Heath and Bizer, 2011). For traceability purposes, in particular the OSLC Change Management specification is relevant. In earlier work (Elaasar and Neal, 2013) OSLC has successfully been demonstrated for integration of modeling tools in general, and traceability in particular.

OpenModelica (Fritzson *et al*, 2006) is an open source modeling, simulation, and optimization tool for Modelica (Modelica Association, 2012; Fritzson, 2014) language. The OpenModelica Connection Editor OMEdit (Asghar *et al*, 2010) is a graphical Modelica model editing and simulation tool. It supports model creation, deletion, FMU export/import, textual and graphical model editing including connections drawing, simulation, plotting, and documentation presentation. In the previous version of OpenModelica (Pop *et al*, 2014) the compiler supports traceability in terms of tracing generated C code back to the originating Modelica source code, but not in the OSLC sense, and mostly used for debugging.

In this paper we present new traceability support in OpenModelica where the traceability information is exchanged with other lifecycle tools through a standardized interface and format using OSLC. In particular, OpenModelica supports automatic recording and tracing of modeling activities such as creation, modification, and destruction of models, import of model description XML, export of FMUs, and creation of simulation results to link models from various tools. OpenModelica supports simple queries (traces to and traces from) to present the traceability information to the user.

The rest of this paper is structured as follows: In Section 2 an overview of OSLC is given. The traceability design and architecture is presented in Section 3. An Example of integrated tools to trace

artifacts created during the system development process is presented in Section 4. Section 5 describes the traceability and model management workflow in OpenModelica. The prototype implementation is described in Section 6. Conclusions and future work are presented in Section 7.

2 Open Services for Lifecycle Collaboration (OSLC)

Open Services for Lifecycle Collaboration (OSLC) (Open-services.net, 2008) is an open source initiative for creating a set of specifications that enables integration of development life cycle tools (e.g., modeling tools, change management tools, requirements management tools, quality management tools, configuration management tools). The goal of OSLC is to make it easier for tools to work together by specifying a minimum amount of protocol without standardizing the behavior of a specific tool.

The OSLC specifications use the Linked Data model to enable integration at the data level via links between tool artifacts defined as Resource Description Framework (RDF) (Manola and Miller, 2004) resources (beside other possible representations such as XML, JavaScript Object Notation (JSON) (json.org, 2016), Atom, and Turtle). The resources are identified by HTTP URIs. A common protocol to perform creation (HTTP POST) and retrieval (HTTP GET), update (HTTP PUT) and delete (HTTP DELETE) operations on resources is also specified.

3 Traceability Design and Architecture

The traceability design and architecture is mainly being developed in the INTO-CPS project (into-cps.au.dk, 2015) which contains a set of tasks. One of these is the design of traceability and model management with the following goals (Lausdahl *et al*, 2016):

- Checking the realization of requirements in models
- Enabling collaborative work by connecting artifacts and knowledge from different users
- Decreasing redundancy by connecting different tools to a single requirements source and allowing a system-wide view that is not only limited to single tools

The Provenance (PROV) (Moreau *et al*, 2013) and OSLC standards presented in (Fitzgerald *et al*, 2015) are used to support traceability activities. PROV is a set of documents built on the notation and relation of entities, activities, and agents.

The design and architecture of the traceability-related tools has recently been developed in (Lausdahl *et al*, 2016) and is shown in Figure 1. Any modeling tool written in any programming language can use these traceability standards to support the traceability of activities performed within the tool and interact with other tools.

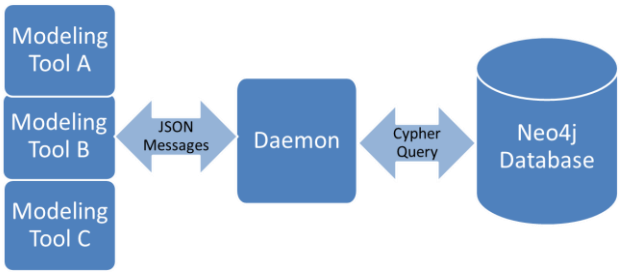

Figure 1. Schematic architecture of the traceability-related tools.

As depicted in Figure 1, the architecture is divided into three parts:

- **Modeling Tools** – The modeling tools send traceability information from activities that are performed within the tools (e.g., model creation, modification, import model description in XML) to the daemon.
- **Daemon** – The daemon provides an OSLC interface compliant with RESTful (Richardson and Ruby, 2007) to store the traceability information into the database and retrieve the traceability data from the database. It is launched and terminated by modeling tools.
- **Neo4j Graph Database** – The Neo4j database (Neo Technology, Inc, 2007) is a graph database to store the OSLC triples that make up the traceability data.

4 An Example of Integrated Tools for Cyber-Physical Model Development

OpenModelica has been successfully integrated with the INTO-CPS tool chain to trace artifacts created during the system development process from high level requirements to simulation results. The tools involved are Overture (Larsen *et al*, 2010), 20-sim (Controllab Products B.V, 2013), Modelio (Favre, 2005) and RT-Tester (Verified Systems International GmbH, 2012). The tool chain as shown in Figure 2 is defined by the connections between the system architecture and the simulation via the model description XML file and the FMU.

The SysML Connection diagram defines the components of the system and their connections. The internals of these block instances are created in the various modeling tools and exported as FMUs. The modeling tools support importing the interface definition (ports) of the blocks in the Connection diagram by importing a modelDescription.xml file containing the block name and its interface definition linked with requirements. All tools are storing information in Git and sending information about existing and created artifacts to the global database.

DOI 10.3384/ecp17132823

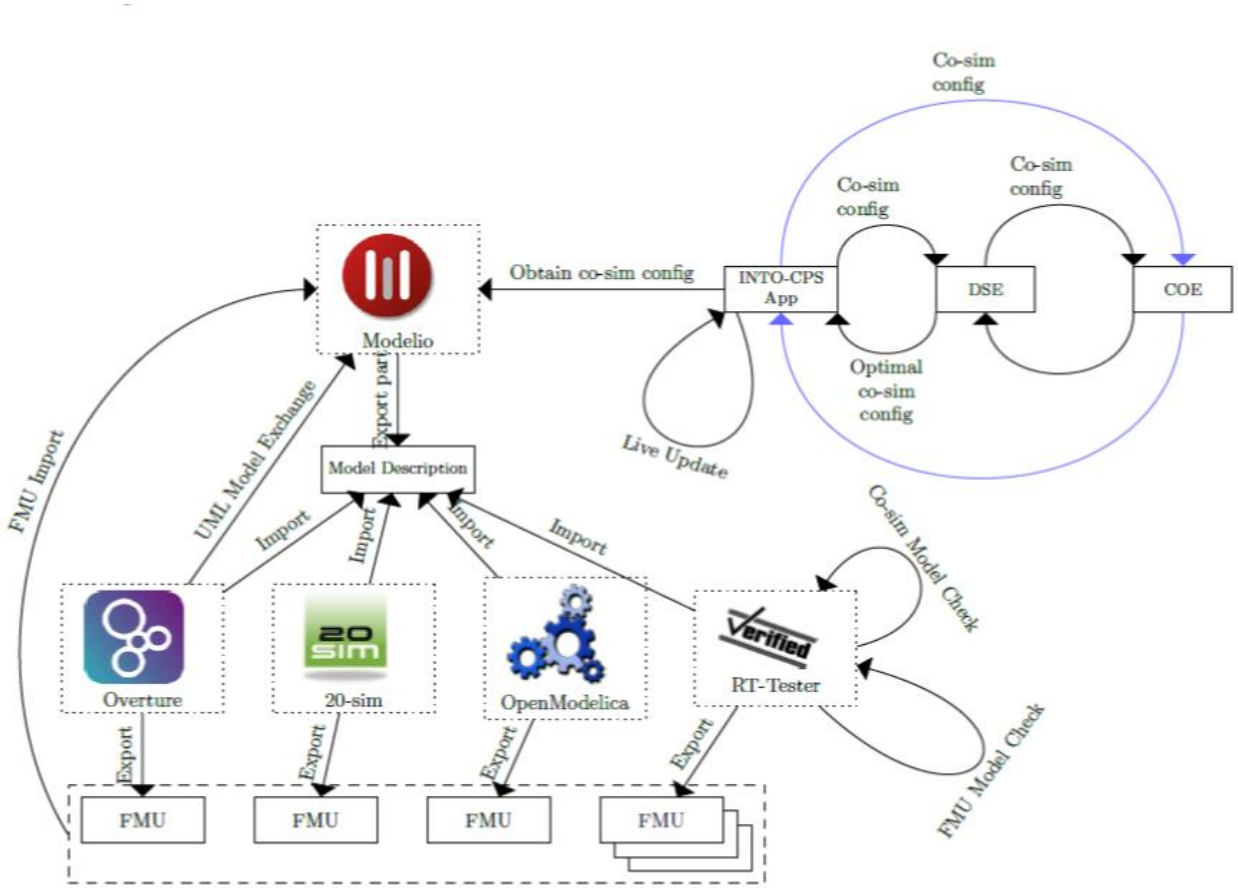

Figure 2. An Example of integrated tools to trace artifacts created during the system development process (Bandur *et al* , 2016).

5 Traceability and Model Management in OpenModelica

In the new work reported in this paper, OpenModelica has been extended with support of traceability in the OSLC sense, where traceability information is exchanged with external tools through a standardized interface and format. The implementation is based on an architecture and a common interface defined in (Lausdahl *et al*, 2016) for exchanging traceability information.

The modeling activities that can be recorded automatically and traced within OpenModelica are:

- Model description XML import (linked with requirements)
- Model creation
- Model modification
- Model destruction
- FMU export
- Simulation result creation

The complete workflow for traceability artefacts within OpenModelica and the different components that rely on are shown in Figure 3.

The following summarizes the main workflow that could be used to create and record traceability information in OpenModelica during cyber-physical model development process.

1. Commit model file entity to Git repository and record the Git-hash
2. Create URIs of the activity based on the Git-hash
3. OSLC triples describing the activity are generated using the URIs
4. OSLC triples are sent to the traceability Daemon
5. Retrieve the traceability information (traces to and traces from)

The traceability information is represented in JSON format. The modeling activities described by OSLC triples represented in JSON format are sent from OpenModelica to the daemon. These traces are then sent through the daemon to the Neo4j database, where they are stored. In order to view and analyze traceability data, this is later retrieved (traces to and traces from) from OpenModelica, through the appropriate queries from the daemon to the database.

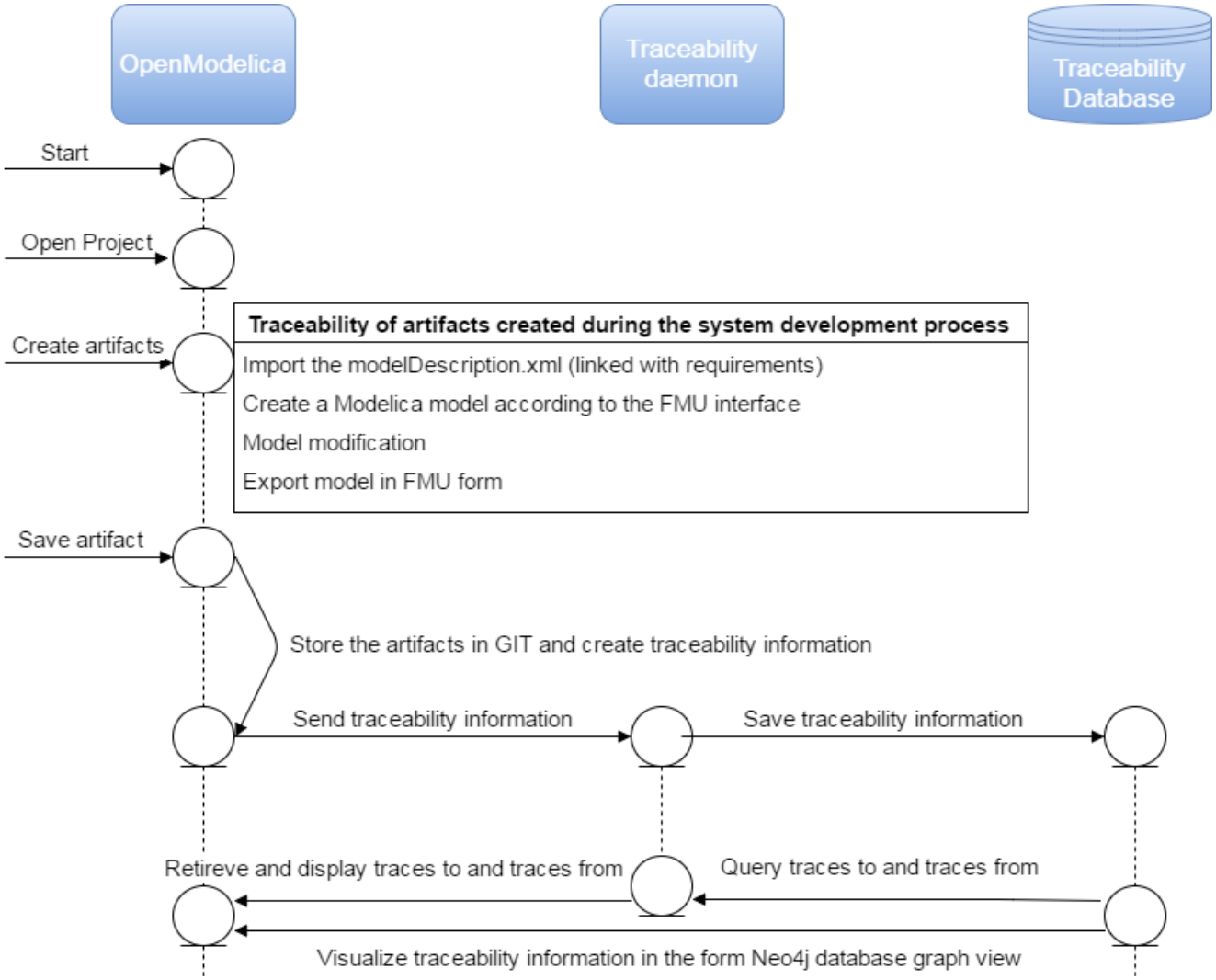

Figure 3. Workflow of traceability of artifacts during the system development process in OpenModelica.

6 Prototype Implementation

We have implemented a prototype to demonstrate the idea of exchanging traceability information for integrating lifecycle modeling tools using OSLC. The prototype is implemented based upon the design and architecture presented in Section 3.

As mentioned, the implementation of this prototype is an extension of OMEdit (Asghar *et al*, 2010) which is implemented in C++ using the Qt Framework (Nokia Corporation, 2011) graphical user interface library. For presentation reasons, we have grouped the prototype functionality into three categories: importing model description XML, model management with Git integration, and traceability support using OSLC, which are described in the following subsections.

6.1 Import Model Description in XML

As a preparation for the extension to support tracing for importing modelDescription.xml interface files, we extended OpenModelica to support importing modelDescription.xml (See Figure 4).

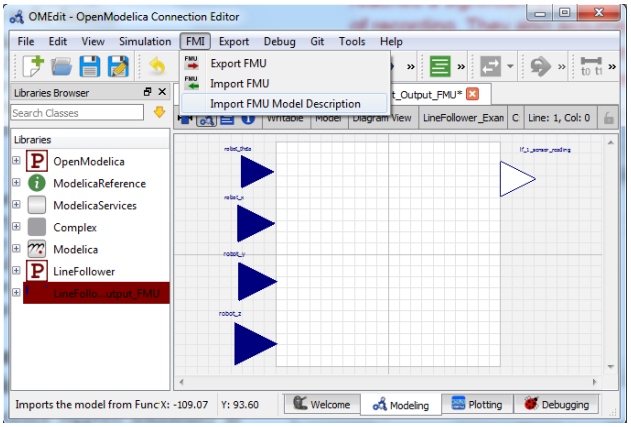

Figure 4. A screen shot of the model description XML import operation.

OpenModelica can import model description XML interface files (linked with requirements) created using other system architectural modeling tools and create

DOI 10.3384/ecp17132823

Modelica models from this information. The result is a generated file with a Modelica model stub containing the inputs and outputs specified in the model Description.xml file. Then the user can create a complete model using the GUI via drag and drop in the editor. Hence, the traceability chain within OpenModelica traces models linked with requirements through model description xml import, model creation, model modification, FMU export and simulation results.

6.2 Model Management with Git Integration

One of the objectives of the traceability tooling is to manage the development process in terms of modeling activities within the modeling tools. In order to achieve this objective access to the version control system is required in OpenModelica. Therefore the OpenModelica Connection Editor OMEdit has been enhanced to support Git version control as shown in Figure 5.

The OMEdit Git integration is currently in an early stage of development but already supports some basic functionality (See Figure 5) such as staging modified tracing operations on files for commit, committing, and reverting changes. It is useful to provide viewing of status and version history which can be used for creating the resource URIs for the modeling activities on each new commit.

The implemented prototype also allows to create a local Git repository by selecting Git -> Create New Repository from the menu bar. Since the URI, as presented in (Fitzgerald et al, 2015) is the combination of the Git-hash and the unique path for every file in the project, creating a Git repository for traceability purposes automatically adds a structure (See the left part of Figure 5) for models, simulation results, FMUs, and model description XML files to the Git repository.

6.3 Traceability Support in OpenModelica

The traceability support in OpenModelica provides a graphical user interface to interact with other lifecycle modeling tools.

As already mentioned in Section 4, OpenModelica supports traceability in the OSLC sense, where traceability information is exchanged with external tools through a standardized interface and format. The implementation is based on the architecture and a common interface defined in (Lausdahl *et al*, 2016) for exchanging traceability information.

OpenModelica imports the modelDescription.xml and creates a Modelica model according to the FMU interface. The generated Modelica model is completed with behavior for the SysML block and the final model is exported in the FMU form. The generated FMU is then used in a whole system simulation connected according to the SysML connection diagram. The

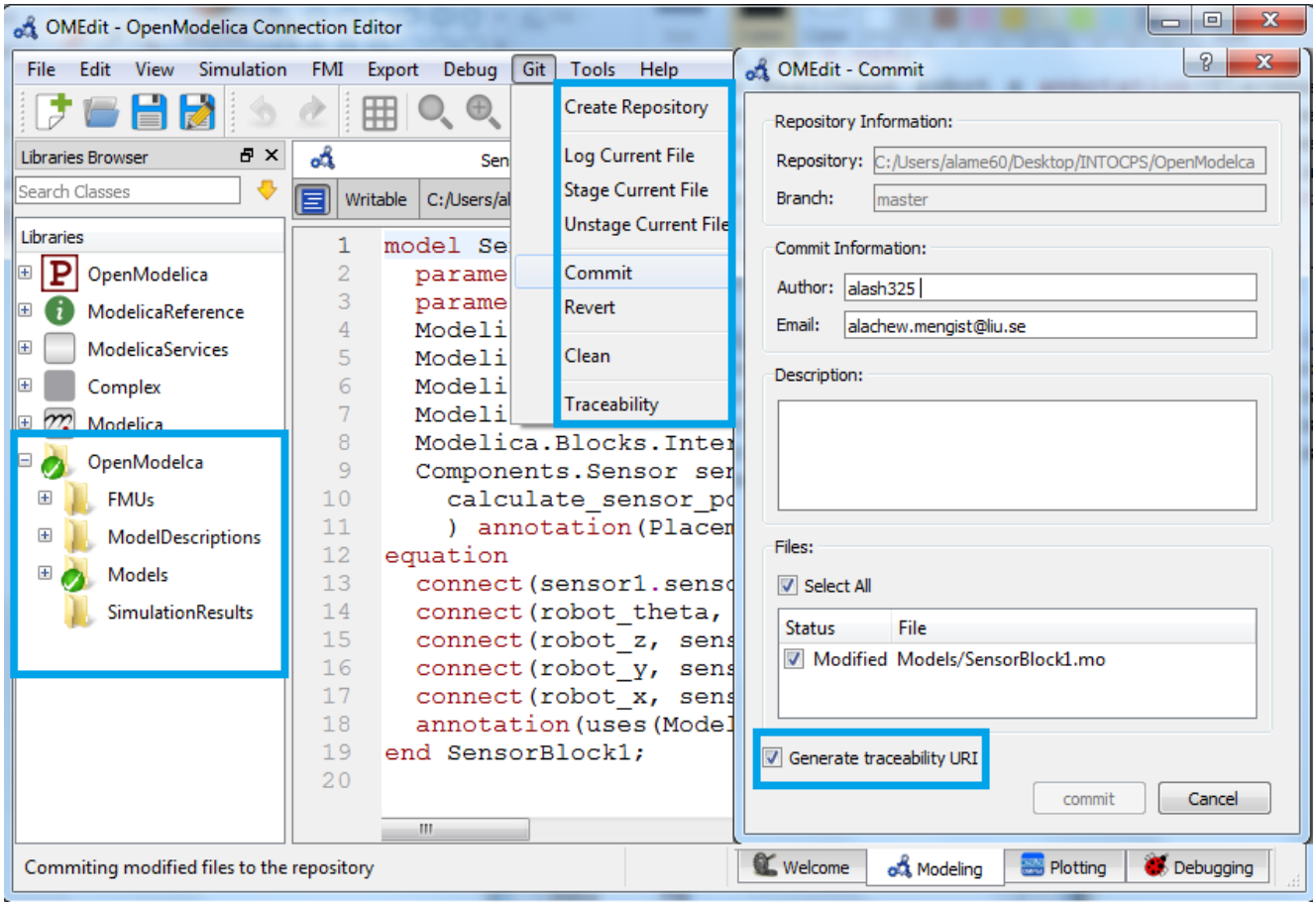

Figure 5. GUI of Git Integration in OpenModelica and functions available to create traceability URI.

FMU master simulation algorithm component performs the simulation via the INTO-CPS App. This whole chain is traced using OSLC.

We have designed a graphical user interface shown in Figure 6 which allows the user to record the traceability information and send to the Daemon (OSLC triples in JSON format), describing the activity using the URIs generated in the GUI shown in Figure 5. The PROV and OSLC relations that are mainly used in this work can be found in (Fitzgerald *et al*, 2015).

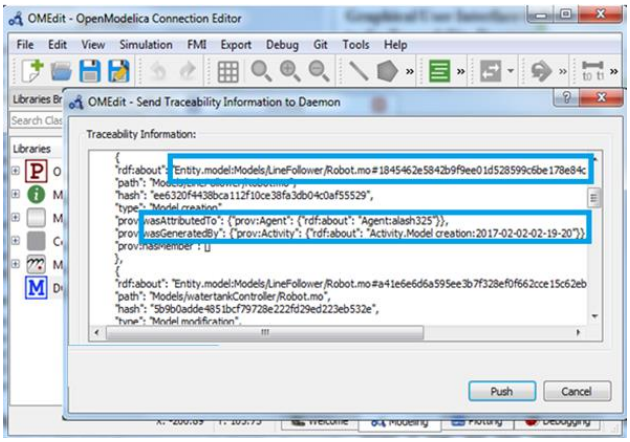

Figure 6. GUI to send traceability information to daemon.

These traces are then sent through the daemon to the database via HTTP POST http://localhost:8080/traces/push/json, where they are stored. Figure 7 shows an example of traceability information sent from OpenModelica to the daemon and visualized in the Neo4j database.

Entities (e.g. *Modelica files, FMUs, modelDescription XML file*) are shown in green, actions (e.g. *model creation, FMU export, modelDescription XML import*) are shown in yellow, agents (e.g. users with the names *"Alachew", "Adrian", "Peter", and "Adeel"*) are shown in blue, and their relationships "*what come from what*" and "*what used what*" (e.g. "*wasGeneratedBy", "wasDerivedFrom", "usedTool"*) are shown with red arrows.

In order to view and analyze traceability data, we have also designed a graphical user interface shown in Figure 8 which allows the user to query traceability information (traces to and traces from) from the daemon to the database (via HTTP GET):

- http://localhost:8080/traces/from/<URI>/json and
- http://localhost:8080/traces/to/<URI>/json

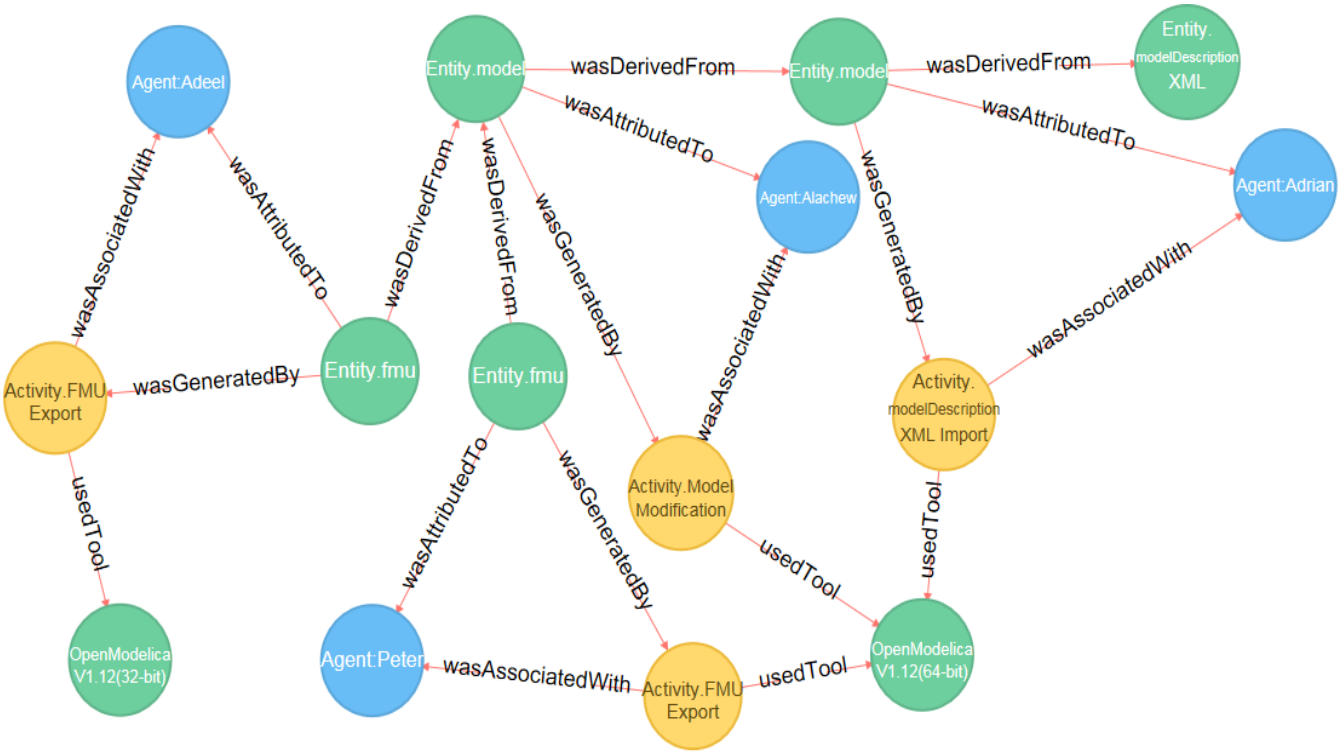

Figure 7. An example of traceability information sent from OpenModelica to the daemon and visualized in the Neo4j database.

DOI 10.3384/ecp17132823

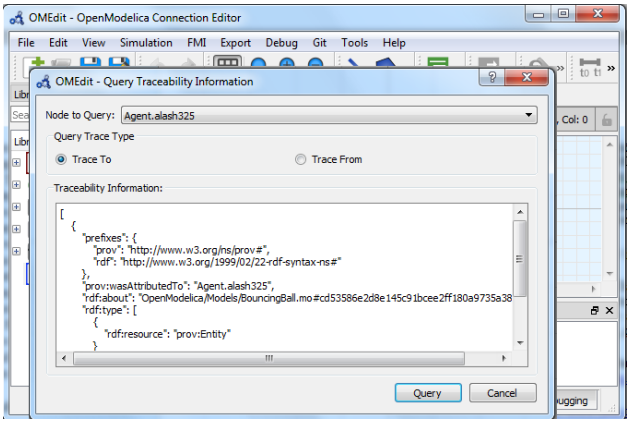

Figure 8. GUI to query traceability information (traces to and traces from) from Neo4j database.

7 Conclusions and Future Work

This paper has presented a framework for traceability and model management in OpenModelica, and its integration with the Git version control system.

The new version of OpenModelica supports traceability in the OSLC sense, where traceability information is exchanged with external tools through a standardized interface and format. The Modeling activities that can be recorded automatically within OpenModelica and traced are import of model description XML linked with requirements, creation of models, modification of models, destruction of Models, export of FMUs, and creation of simulation results.

A first prototype to query traceability information (traces to and traces from models or simulation results) from the database and display to end-users in JSON format is also complete. As future work, we also intend to extend the OpenModelica tool to support visualization and presentation of the traceability data viewed both in the form of graphs and trees.

The OpenModelica model management with Git integration is currently in an early stage of development but is already being able to support end-users to trace back all steps of the modeling process and to revert each step in the development history, and also model collaboration between end-users. Ongoing work is focused on having fully functional Git integration including showing two versions of the same model in parallel.

Future work also involves computing the impact of two different versions of the same model on simulation results and merging the models in way that the resulting model can be valid without modification.

Acknowledgments

This work has been supported by the European Union in the H2020 INTO-CPS project. Support from Vinnova in the ITEA3 OPENCPS project has been received. The OpenModelica development is supported by the Open Source Modelica Consortium. Special thanks to Kenneth Lausdahl, Peter Niermann, Jos Höll, Carl Gamble, Oliver Möller, Etienne Brosse, Tom Bokhove, and Luis Diogo Couto for collaboration and valuable input to traceability related tools design.

References

Adeel Asghar, Sonia Tariq, Mohsen Torabzadeh-Tari, Peter Fritzson, Adrian Pop, Martin Sjölund, Parham Vasaiely, and Wladimir Schamai. An Open Source Modelica Graphic Editor Integrated with Electronic Notebooks and Interactive Simulation. In Proc. of the 8th International Modelica Conference 2011, pp. 739–747. Modelica Association, March 2011.Linköping University, Sweden, 2010.

Victor Bandur, Peter Gorm Larsen, Kenneth Lausdahl, Casper Thule, Anders Franz Terkelsen, Carl Gamble, Adrian Pop, Etienne Brosse, Jrg Brauer, Florian Lapschies, Marcel Groothuis, Christian Kleijn, and Luis Diogo Couto. INTO-CPS Tool Chain User Manual. Technical report, INTO-CPS Deliverable, D4.2a, December 2016.

Torsten Blochwitz et al. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In Proceedings of the 8th International Modelica Conference, Dresden, Mar. 2011. doi: 10.3384/ecp11063105.

Controllab Products B.V. Modelling and simulation software package for mechatronic systems http://www.20sim.com/, January 2013.

Maged Elaasar and Adam Neal. Integrating Modeling Tools in the Development Lifecycle with OSLC: A Case Study, pages 154-169. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

Jean-Marie Favre. Foundations of Model (Driven) (Reverse) Engineering: Models – Episode I: Stories of The Fidus Papyrus and of The Solarus. In Language Engineering for Model-Driven Software Development, March 2005.

John Fitzgerald, Carl Gamble, Richard Payne, and Ken Pierce.Methods Progress Report 1. Technical report, INTO-CPS Deliverable, D3.1b, December 2015.

FMI-Standard.org (2014). Functional Mock-up Interface for ModelExchange and Co-Simulation Version 2.0. https://www.fmi-standard.org/ (accessed: 10th of December 2016).

Peter Fritzson. Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach. 1250 pages. ISBN 9781-118-859124, Wiley IEEE Press, 2014.

Peter Fritzson, Peter Aronsson, Adrian Pop, Hakan Lundvall, Kaj Nyström, Levon Saldamli, David Broman, Anders Sandholm. OpenModelica – A Free Open-Source Environment for System Modeling, Simulation, and Teaching. Proceedings of the 2006 IEEE Conference on Computer Aided Control System Design, Munich, Germany, October 4–6, 2006.

Tom Heath and Christian Bizer (2011) Linked Data: Evolving the Web into a Global Data Space (1st edition). Synthesis Lectures on the Semantic Web: Theory and Technology, 1:1, 1-136. Morgan & Claypool, 2011. doi: 10.2200/S00334ED1V01Y201102WBE001.

into-cps.au.dk (2015). Integrated Tool Chain for Model-based Design of Cyber-Physical Systems. http://into-cps.au.dk/ (accessed: 10th of December 2016).

json.org. JavaScript Object Notation. http://www.json.org/ (accessed: 10th of December 2016).

Peter Gorm Larsen, Nick Battle, Miguel Ferreira, John Fitzgerald, Kenneth Lausdahl, and Marcel Verhoef. The Overture Initiative – Integrating Tools for VDM. SIGSOFT Softw. Eng. Notes, 35(1):1–6, January 2010.

Kenneth Lausdahl, Peter Niermann , Jos Höll , Carl Gamble ,Oliver Möller , Etienne Brosse , Tom Bokhove , Luis Diogo Couto , Adrian Pop , and Christian König. INTO-CPS Traceability Design. Technical report, INTO-CPS Deliverable, D4.2d, December 2016.

Frank Manola and Eric Miller, editors (2004). RDF Primer. W3C Recommendation. World Wide Web Consortium. https://www.w3.org/TR/2004/REC-rdf-primer-20040210/ (accessed: 10th of December 2016).

Modelica Association (2012). Modelica: A Unified Object Oriented Language for Physical Systems Modeling, Language Specification version 3.3. https://modelica.org/ (accessed: 10th of December 2016).

Luc Moreau, Paolo Missier, James Cheney and Stian Soiland-Reyes, editors and contributors (2013): An Overview of the PROV Family of Documents. https://www.w3.org/TR/prov-n/ (accessed: 10th of December 2016).

Neo Technology, Inc (2007). Neo4j Database. https://neo4j.com/ (accessed: 10th of December 2016).

Nokia Corporation (2011). Qt Project. https://www.qt.io/ (accessed: 10th of December 2016).

Open-services.net (2008): Open Services for Lifecycle Collaboration – Lifecycle Integration Inspired by the Web. http://open-services.net/ (accessed: 10th of December 2016).

Adrian Pop, Martin Sjölund, Adeel Ashgar, Peter Fritzson, and Francesco Casella. Integrated Debugging of Modelica Models. Modeling, Identification and Control, 35(2):93{107, 2014.

Leonard Richardson and Sam Ruby. RESTful Web Services (First ed.), O'Reilly, 2007.

Verified Systems International GmbH, Bremen, Germany. RTTester Model-Based Test Case and Test Data Generator – RTTMBT: User Manual, 2015. https://www.verified.de/products/model-based-testing/, Doc. Id. Verified-INT-003-2012.

DOI 10.3384/ecp17132823