Information and Software Technology 000 (2017) 1–18

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A dedicated approach for model composition traceability

Youness Laghouaouta[a,*], Adil Anwar[b], Mahmoud Nassar[a], Bernard Coulette[c]

[a] IMS Team, ADMIR Laboratory, Rabat IT Center, ENSIAS, Mohammed V University in Rabat, Morocco
[b] SIWEB Team, EMI, Mohammed V University in Rabat, Morocco
[c] IRIT Laboratory, University of Toulouse, France

ARTICLE INFO

Article history:
Received 20 March 2016
Revised 2 July 2017
Accepted 4 July 2017
Available online xxx

Keywords:
Model traceability
Model composition
Aspect-oriented modeling
Graph transformations
NFR framework

ABSTRACT

Context: Software systems are often too complex to be expressed by a single model. Recognizing this, the Model Driven Engineering (MDE) proposes multi-modeling approaches to allow developers to describe a system from different perspectives. In this context, model composition has become important since the combination of those partial representations is inevitable. Nevertheless, no approach has been defined for keeping track of the composition effects, and this operation has been overshadowed by model transformations.

Objective: This paper presents a traceability approach dedicated to the composition of models. Two aspects of quality are considered: producing relevant traces; and dealing with scalability.

Method: The composition of softgoal trees has been selected to motivate the need for tracing the composition of models and to illustrate our approach. The base principle is to augment the specification of the composition with the behavior needed to generate the expected composed model accompanied with a trace model. This latter includes traces of the execution details. For that, traceability is considered as a crosscutting concern and encapsulated in an aspect. As part of the proposal, an Eclipse plug-in has been implemented as a tool support. Besides, a comparative experiment has been conducted to assess the traces relevance. We also used the regression method to validate the scalability of the tool support.

Results: Our experiments show that the proposed approach allows generating relevant traces. In addition, the obtained results reveal that tracing a growing number of elements causes an acceptable increase of response time.

Conclusion: This paper presents a traceability approach dedicated to the composition of models and its application to softgoal trees. The experiment results reveal that our proposal considers the composition specificities for producing valuable traceability information while supporting scalability.

1. Introduction

Traceability consists in linking the software development artifacts to each other through specific relationships [1]. Its practice is recognized as an essential activity that enhances quality aspects of the final solution (e.g. efficiency, maintainability, ...). For example, functional coverage analysis can be achieved by exploring traceability relationships between requirements and their realizations [2]. Also, relations between design products and their implementations provide a support for optimizing maintenance tasks and analyzing impacts of changes [3–5].

Moreover, the complexity of current systems makes it essential to use a traceability support. The number of software artifacts is increasing significantly, traceability management helps mastering this complexity by exposing intrinsic relationships and by recording the life cycle of each artifact. However, this legitimate need for traceability practice and the various advantages it offers do not confer a wide scale use. This is due, essentially, to the additional cost of capturing and maintaining traceability links, and the divergence of interests between the links creators and users [6].

The requirements engineering community was the first to invest in traceability integration. Its purpose being to ensure that the requirements are well covered by the final product, traceability is presented as a support to such assessment. Thereafter, the traceability benefits have attracted the interest of Model Driven Engineering (MDE) researches. In this software engineering field, other aspects of traceability management have emerged. Those aspects depend in terms of traceability intentions (the users' expectations by its practice) and the nature of the elements to be traced (model

* Corresponding author.
E-mail address: y.laghouaouta@um5s.net.ma (Y. Laghouaouta).

http://dx.doi.org/10.1016/j.infsof.2017.07.002
0950-5849/

or non model artifacts). Essentially, two classes of traceability approaches were identified [7]: requirements traceability and model transformations traceability.

Indeed, in a Model Driven Development (MDD) process, the final system is obtained by transforming requirements towards solutions that realize them. These solutions are not directly produced through one shot operation, but they result from sequential transformations applied to the primary models. Accordingly, requirements traceability specifically addresses the management of relationships between requirements and the corresponding solutions. While model transformations traceability focuses on capturing the effects of executing model transformations against the managed models. Nevertheless, the model transformation operation does not constitute the only pillar of MDD. The composition operation is also of major importance as it supports multi-modeling approaches. Indeed, to reduce the complexity of the development activity, developers can describe the system by as many models as they want. These partial representations will be less complex than the global model. However, they need to be composed to deal with model validation and synchronization issues, and to better understand the interrelations between them.

Nonetheless, the composition of models remains a complex activity. In order to overcome this issue, traceability practice is presented as a significant solution. Indeed, traces expose the exact effects of executing the composition and help developer to better comprehend interrelations among managed models. Also, they provide a basis for validating the composition (e.g. each element of the composed model has to be connected with a traceability link which justifies its existence) and evolving models when changes occur (e.g. elements that are impacted by removing a given source model element can be retrieved from analyzing traceability links they are connected with). Taking full benefits of the aforementioned possible exploitations of traceability information implies the capture of expressive and interesting traces. However, given that existing transformation traceability supports disregard the composition features; their application to composition scenarios compromises the expressiveness of traces and leads to the production of traceability information almost worthless.

Recognizing this, the paper presents a traceability management approach dedicated to the model composition operation. Our proposal is applied to model compositions specified with a rule based language. The principal is inspired from the work presented by Jouault [8] and it consists in allowing the composition to generate traces like other target elements. Hence, traces that expose the effects of applying the composition specification to the source models are constructed similarly and in parallel to the construction of the composed model elements.

Fig. 1 illustrates this idea. The basic artifact of the traces generation process is a preexisting specification that allows generating the composed model. The principle is to augment this specification with the behavior it needs to generate traceability information. After that, the execution of the resulting specification will produce two outputs: the default composed model, and the corresponding trace model. This latter conforms to a generic traceability metamodel that provides the concepts needed to express the expected traceability information. For implementing such mechanism, we consider traceability as a crosscutting concern that is encapsulated in an aspect. The application of this aspect (by means of a dedicated weaving process) weaves the traces generation patterns into the primary composition specification.

In previous works [9,10], we presented the specialization of this traces generation process for the Epsilon Merging Language EML [11] and the ATLAS Transformation Language ATL [12]. We described the corresponding traces generation patterns as well as mechanisms to perform the weaving of these patterns into the

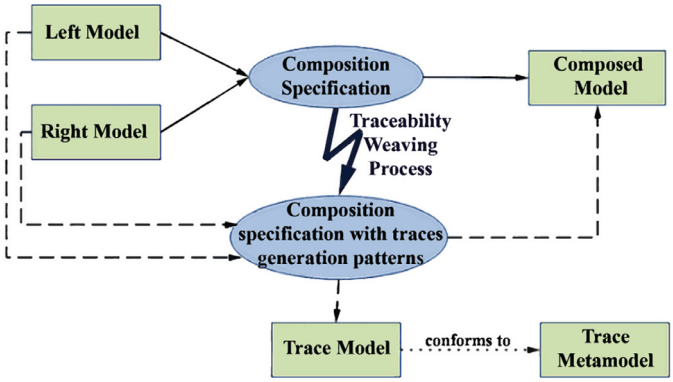

Fig. 1. Global overview of the traces generation process.

composition specifications. The current paper focuses on the validation of the proposed traceability approach. By using the *ComposeTracer* plugin (i.e. the tool support we implemented for the proposed traceability approach) we realized different composition scenarios. After that, we calculated and compared the values taken by some appropriated metrics to validate the relevance of the generated traces. Also, we relied on a statistical method for analysis (i.e. linear regression) to predict the variation of response time depending on the number of elements to be traced. For convenience, we summarize the specific contributions of this paper in three points:

- Composition of softgoal refinement trees which has been selected to demonstrate the need for a traceability approach dedicated to the model composition operation.
- Implementation details of the tool supporting the proposed traceability approach.
- Results of an empirical study that aims to assess the traces relevance and the scalability of the traceability support.

The remainder of this paper is structured as follows. In Section 2, we introduce our traceability approach. Section 3 presents an illustrative example that motivates the need for a dedicated composition traceability approach. Section 4 gives a background to traceability management and lists a set of requirements that have driven our approach. Section 5 provides an overview of our conceptual proposal. In Section 6, we explain implementation details of the *ComposeTracer* plug-in which supports the traces generation. Section 7 presents the validation of the proposal. Section 8 discusses our contributions to overcome some traceability challenges. Section 9 lists the related works. Finally, Section 10 summaries this paper and presents future work.

2. Overview of the proposed traceability approach

The aim of this section is to provide a global overview of our traceability approach dedicated to the composition of models. Basically, such approach for generating traces is applied on the composition of any type of software artifacts models (e.g. requirements models, design models, implementation models). However, given the type of traceability information produced by our approach and especially its granularity level, the proposed approach is mainly intended for analysts and designers. Implementation models are excluded since maintaining their traceability requires fine grained supports.

Recalling from Section 1, the underlying idea is to allow the composition specification to generate and record traces in an extra-output model which accompanies the expected composed model. For that, the composition specification (it consists of a set of