Science of Computer Programming 113 (2015) 51–81

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

A graph-based algorithm for three-way merging of ordered collections in EMF models ☆

Felix Schwägerl *, Sabrina Uhrig, Bernhard Westfechtel

University of Bayreuth, Chair of Applied Computer Science I, Universitätsstr. 30, 95440 Bayreuth, Germany

ARTICLE INFO

Article history:
Received 28 May 2014
Received in revised form 16 February 2015
Accepted 19 February 2015
Available online 16 March 2015

Keywords:
Model-driven software engineering
EMF models
Version control
Three-way merging
Graph algorithm

ABSTRACT

In EMF models, ordered collections appear as the values of multi-valued structural features. Traditional, text-based version control systems do not sufficiently support three-way merging of ordered collections inside EMF models since they cannot guarantee a consistent result. The operation three-way merging is defined as follows: based on a common base version b, two alternative versions a_1 and a_2 were developed by copying and modifying the base version. To reconcile these changes, a merged version m is to be created as a common successor of a_1 and a_2. In this paper, we present a graph algorithm to solve the problem of three-way merging of ordered collections in EMF models. Each version of a collection can be represented by means of a linearly ordered graph. To create the merged version, these graphs are combined to a merged collection graph using set formula. To create the merged collection, a generalized topological sort is performed on the merged collection graph. Conflicts occur in case the order of elements cannot be deduced automatically; these conflicts are resolved either interactively or by default rules. We have implemented the merge algorithm in our tool BTMerge, which performs a consistency-preserving three-way merge of versions of EMF models being instances of arbitrary Ecore models. Our implementation relies on an alternative form of representing multiple versions of a collection, namely a versioned collection graph which forms a superimposition of collection versions. The algorithm presented here is purely state-based. Matching and merging of collections are clearly separated sub-problems. Insertions and deletions performed on the elements of the collection are propagated into the merged version in a consistent way. Our algorithm makes only minimal assumptions with regard to the underlying product model and thus may be applied to ordered collections inside plain text or XML files. By taking arbitrary move operations into account, the algorithm considerably goes beyond the functionality of contemporary merge tools which cannot adequately handle move operations.

☆ This paper is an extended version of [1].

* Corresponding author.
E-mail addresses: felix.schwaegerl@uni-bayreuth.de (F. Schwägerl), sabrina.uhrig@uni-bayreuth.de (S. Uhrig), bernhard.westfechtel@uni-bayreuth.de (B. Westfechtel).

http://dx.doi.org/10.1016/j.scico.2015.02.008
0167-6423/

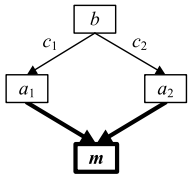

Fig. 1. Three-way merging.

1. Introduction

1.1. Background

Model-driven software engineering (MDSE) [2] denotes a software engineering process which is driven by the development of high-level models being expressed in well-defined modeling languages. MDSE involves a high degree of automation through the use of tools for creating models and automatically or interactively transforming them into an executable form. In this way, MDSE promises to increase the productivity of software engineers, who are relieved of low-level routine coding tasks and may focus on creative and intellectually challenging modeling tasks. Just like source code, developing models in a team over a large period requires sophisticated *version control*, and in particular tool support for three-way merging concurrently developed model versions.

Three-way merging is an important operation for *optimistic version control* as supported by a variety of traditional, text-based version control tools such as Subversion [3] or CVS [4]. Different software engineers may concurrently create different successors of the same base version without being delayed by locks on the base version. Later on, they reconcile their work by three-way merging such that non-conflicting changes are combined automatically and conflicts are detected and resolved. In terms of text-based merging, conflicts are reported in case developers perform different modifications to the content of corresponding lines of text of their respective versions. Three-way merging is part of the daily work of software engineers working in teams over a large period. Merge tools for text files have been used for long; model-based three-way merging is being addressed by current research.

Often, models are represented externally as diagrams rather than as text (consider, e.g., the *Unified Modeling Language* (UML)). Tools operating on models represent them internally as graphs of objects, attributes, and links. For data exchange, models are usually stored within XMI documents (*XML Metadata Interchange*). Although XMI documents are text files, they are not human readable. The application of traditional version control tools to models brings serious limitations with it, particularly concerning the *comparison* of model versions [5]. A textual comparison of these files results in differences on the XMI representations rather than syntactic or semantic, model-level differences. Likewise, textual merging of XMI-represented models may easily deliver a result which is neither consistent nor meaningful. In case an inconsistent result is created, a tool operating on the model may not be able to load the flawed XMI representation, which needs to be fixed manually.

Thus, tools for *comparing* and *merging models* which take the syntax and/or semantics of models into account, are urgently needed. In contrast to text-based tools, *model-based tools* operate on the representation of models as sets of interconnected model elements. For comparing models, a variety of algorithms have been proposed and implemented [6–9], including, e.g., the well-known tool EMF Compare [10]. Likewise, quite a number of model-based merge tools have been developed [11–15]. Unfortunately, the current state of the art does not guarantee that the produced merge result is *consistent*, i.e., that it may be represented in its concrete graphical or textual syntax without any constraint violation.

For three-way merging of models, we have developed the tool *BTMerge* [16] which is based on the theoretical foundations presented in [17,18] and is characterized by the following features: first, BTMerge is a *model-based* tool which internally relies on a *graph representation* of the input models to be merged. Second, the tool may be applied to *any EMF model*, regardless of the underlying Ecore model. Thus, BTMerge covers a large set of models, obviating the need for a specific merge tool for each model type. Third, BTMerge *preserves consistency*. Being supplied with input models which are consistent instances of a common Ecore model, BTMerge constructs a consistent merged model. This is an important advantage over text-based merge tools, which cannot guarantee the well-formedness of their three-way merge results.

1.2. Contributions

In this paper, we focus on *three-way merging* of model versions (Fig. 1). Based on a common *base version b*, two *alternative versions* a_1 and a_2 were developed by copying and modifying the base version. To reconcile the changes, a *merged version m* is to be created as a common successor of a_1 and a_2.

1.2.1. Problem statement: merging linearly ordered collections

The current paper deals with an important sub-problem which frequently occurs in three-way model merging: *merging* of *(linearly) ordered collections*. The problem of merging linear data structures has been studied for different kinds of artifacts such as models (the focus of this paper), text files, or XML documents [11,10,15,19,20]. Ordered collections are contained in virtually any EMF model; consider, e.g., parameters of operations or structural and behavioral features of classes in