Constraint-Aware Model Merging

Alessandro Rossini, Uwe Wolter
University of Bergen
P.O. Box 7803, 5020 Bergen, Norway
{rossini,wolter}@ii.uib.no

Adrian Rutle, Florian Mantz, Yngve Lamo
Bergen Universiy College
P.O. Box 7030, 5020 Bergen, Norway
{aru,fma,yla}@hib.no

Since the beginning of computer science, raising the abstraction level of software systems has been a continuous process. One of the latest steps in this direction has led to the usage of modelling languages in software development processes. Software models are abstract representations of software systems which are used to tackle the complexity of present-day software by enabling developers to reason at a higher level of abstraction. In model-driven engineering (MDE), models are first-class entities of the software development process and undergo a complex evolution during their life-cycles. As a consequence, there is a growing need for techniques and tools to support model management activities such as version control.

In *optimistic* version control, each developer has a local copy of a software artefact. These local copies are modified independently and in parallel and, from time to time, local modifications are merged together. The merge is performed using a *three-way* merging technique [6], which attempts to merge two versions of a software artefact relying on the common ancestor version from which both versions originated. This technique facilitates the detection of conflicts. Roughly speaking, conflicts may arise when the modifications are contradictory. They are resolved either manually or, when applicable, automatically.

Mainstream version control systems, e.g. [1], target text-based artefacts. Hence, underlying techniques such as merging, conflict detection and conflict resolution are based on a per-line textual comparison. Since the structure of models is graph-based rather than text-based, these techniques are not suitable for MDE.

To cope with this problem, a few prototype version control systems have been developed that target graph-based structures, e.g. [2]. However, a uniform formalisation of model merging, conflict detection and conflict resolution in MDE is still debated in the literature. Research has lead to a number of findings in this field [6]. The interested reader may consult [7, 3, 12, 11, 4] for different approaches to model merging, conflict detection and conflict resolution. Unfortunately, these techniques consider only model elements and their conformance to the constraints of the corresponding modelling language, e.g. well-formedness constraints. However, these techniques should also consider constraints added to model elements, e.g. multiplicity constraints. An interesting challenge is then to extend the current techniques by enabling version control of constraints.

In this work, a formal approach to constraint-aware model merging is proposed; i.e. a model merging technique which enables the detection and possibly resolution of *semantic conflicts* on constraints. The proposed approach is based on the Diagram Predicate Framework (DPF) [9, 7, 10, 8] which provides a formalisation of (meta)modelling and model transformation based on graph theory and category theory.

The word “model” has different meanings in different contexts. In software engineering, model denotes “an abstraction of a (real or language-based) system allowing predictions or inferences to be made” [5]. Models in software engineering are typically diagrammatic. The word “diagram” has also different meanings in different contexts. In software engineering, diagram denotes a structure which is based on graphs; i.e. a collection of nodes together with a collection of arrows between nodes. Since graph-based structures can be visualised in a natural way, “visual” and “diagrammatic” modelling are often treated as synonyms. In this work, visualisation and diagrammatic syntax are clearly distinguished. That is, the proposed approach focuses on precise syntax and semantics of diagrammatic models independent of their visualisation.

In DPF, models are represented by *(diagrammatic) specifications*. A specification $\mathfrak{S} = (S, C^{\mathfrak{S}} : \Sigma)$ consists of an underlying graph S together with a set of *atomic constraints* $C^{\mathfrak{S}}$ [9, 8]. The graph

Table 1: A sample signature Σ

p	$\alpha^{\Sigma}(p)$	**Proposed vis.**	**Semantic interpretation**
`[mult(`m,n`)]`	$1 \xrightarrow{a} 2$	X —f [m..n]→ Y	$\forall x \in X : m \leq \lvert f(x)\rvert \leq n$, with $0 \leq m \leq n$ and $n \geq 1$
`[surjective]`	$1 \xrightarrow{a} 2$	X —f [surj]→ Y	$f(X) = Y$
`[inverse]`	$1 \underset{b}{\overset{a}{\rightleftarrows}} 2$	X —f [inv] g→ Y	$\forall x \in X, \forall y \in Y : y \in f(x)$ iff $x \in g(y)$

represents the structure of the model while atomic constraints add restrictions to this structure. Atomic constraints are formulated by predicates from *(diagrammatic predicate) signature*s. A signature $\Sigma = (P^{\Sigma}, \alpha^{\Sigma})$ consists of a collection of predicates, each having a name, a shape graph, a visualisation and a semantic interpretation [9, 8].

The proposed approach to constraint-aware model merging is presented in the following by means of a running example. This example is kept intentionally simple, retaining only the details which are relevant for the discussion. Suppose that Alice and Bob are working on the same model $\mathfrak{V}_1$ (see Fig. 1a). Alice modifies the model $\mathfrak{V}_1$ to $\mathfrak{A}_1$ by adding a multiplicity constraint $(\texttt{[mult(}0,3\texttt{)]}, \delta_1 : (1 \xrightarrow{a} 2) \to (X \xrightarrow{f} Y))$ on the arrow f (see Fig. 1b). Then, Bob modifies the model $\mathfrak{V}_1$ to $\mathfrak{B}_1$ by adding a surjectivity constraint $(\texttt{[surjective]}, \delta_3)$ on the arrow g and an inverse constraint $(\texttt{[inverse]}, \delta_2)$ on the arrows f and g (see Fig. 1c). Finally, Alice tries to merge her own modifications with Bob's modifications. Usually, a version control system would naively merge the changes.

However, in the model $\mathfrak{B}_1$ the arrows f and g are inverse of each other, thus a surjectivity constraint $(\texttt{[surjective]}, \delta_3)$ on the arrow g entails a totality constraint $(\texttt{[mult(}1,\infty\texttt{)]}, \delta_1)$ on the arrow f [8]. The entailed constraint $(\texttt{[mult(}1,\infty\texttt{)]}, \delta_1)$ introduced by Bob's modification and the constraint $(\texttt{[mult(}0,3\texttt{)]}, \delta_1)$ added by Alice on the same arrow f would lead to a conflict (see Fig. 1d). This is because in the model $\mathfrak{V}_2$ these multiplicity constraints would overlap on the same arrow f. In fact, this would not be satisfiable according to the semantic interpretation of the predicate `[mult(`m,n`)]`. A constraint-aware model merging approach should provide automatic resolution of this conflict. This

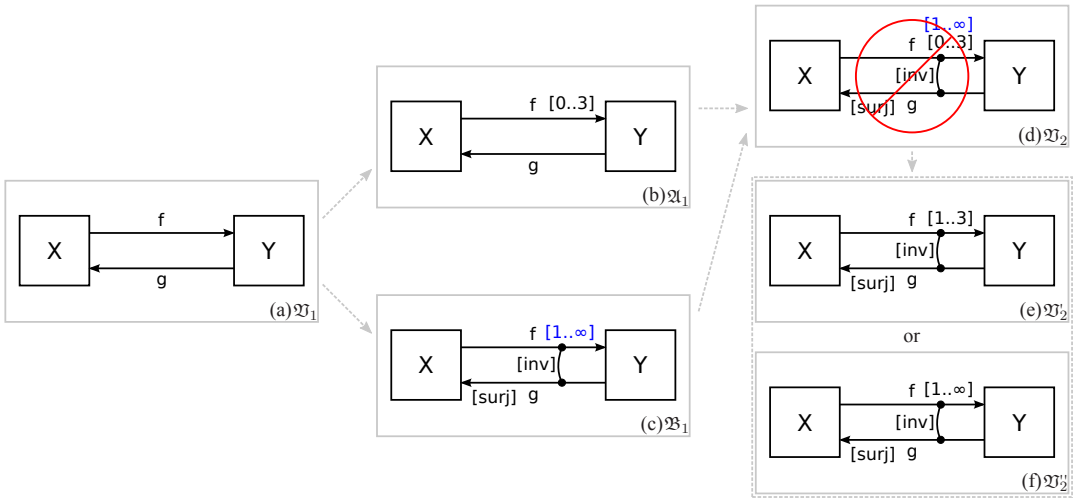

Figure 1: A conflict on multiplicity and surjectivity constraints and possible resolutions

is possible by adopting one of the following *resolution patterns* [3]. The first (conservative) resolution pattern is to merge the multiplicity constraints to a constraint which is the intersection of the two; i.e. $(\texttt{[mult(}1,\infty\texttt{)]},\delta_1) \wedge (\texttt{[mult(}0,3\texttt{)]},\delta_1) \equiv (\texttt{[mult(}1,3\texttt{)]},\delta_1)$. The second (liberal) resolution pattern is to merge the multiplicity constraints to a constraint which is the union of the two; i.e. $(\texttt{[mult(}1,\infty\texttt{)]},\delta_1) \vee (\texttt{[mult(}0,3\texttt{)]},\delta_1) \equiv (\texttt{[mult(}0,\infty\texttt{)]},\delta_1)$. However, since the surjectivity constraint $(\texttt{[surjective]},\delta_3)$ on the arrow g entails a totality constraint $(\texttt{[mult(}1,\infty\texttt{)]},\delta_1)$ on the arrow f, we can conclude $(\texttt{[surjective]},\delta_3) \wedge (\texttt{[mult(}0,\infty\texttt{)]},\delta_1) \equiv ((\texttt{[surjective]},\delta_3) \wedge (\texttt{[mult(}1,\infty\texttt{)]},\delta_1)) \wedge ((\texttt{[surjective]},\delta_3) \wedge (\texttt{[mult(}0,\infty\texttt{)]},\delta_1)) \equiv (\texttt{[surjective]},\delta_3) \wedge (\texttt{[mult(}1,\infty\texttt{)]},\delta_1)$. The application of the first and the second resolution patterns would lead to the models $\mathfrak{V}'_2$ and $\mathfrak{V}''_2$, respectively (see Fig. 1e and Fig. 1f).

In this work, the formal foundation of DPF is extended with a logic and reasoning systems which facilitate constraint-aware model merging. A fully fledged logic will enable developers to reason about properties of specifications and to detect faults in specifications such as inconsistencies, contradictions and unsatisfiability. In this regard, the definition of further deduction rules and logical connectives between constraints may be considered.

References

[1] Apache Subversion. *Project Web Site*. http://subversion.apache.org/.

[2] P. Brosch, G. Kappel, M. Seidl, K. Wieland, M. Wimmer, H. Kargl, and P. Langer. Adaptable Model Versioning in Action. In *Modellierung 2010*, volume 161 of *LNI*, pages 221–236. GI, 2010.

[3] A. Cicchetti, D. Di Ruscio, and A. Pierantonio. Managing Model Conflicts in Distributed Development. In *MoDELS 2008*, volume 5301 of *LNCS*, pages 311–325. Springer, 2008.

[4] M. A. A. da Silva, A. Mougenot, X. Blanc, and R. Bendraou. Towards Automated Inconsistency Handling in Design Models. In *CAiSE 2010*, volume 6051 of *LNCS*, pages 348–362. Springer, 2010.

[5] T. Kühne. Matters of (meta-)modeling. *SoSyM*, 5(4):369–385, 2006.

[6] T. Mens. A State-of-the-Art Survey on Software Merging. *IEEE Trans. on Software Engineering*, 28(5):449–462, 2002.

[7] A. Rossini, A. Rutle, Y. Lamo, and U. Wolter. A Formalisation of the Copy-Modify-Merge Approach to Version Control in MDE. *JLAP*, 79(7):636–658, 2010.

[8] A. Rutle. *Diagram Predicate Framework: A Formal Approach to MDE*. PhD thesis, Department of Informatics, University of Bergen, Norway, 2010.

[9] A. Rutle, A. Rossini, Y. Lamo, and U. Wolter. A Diagrammatic Formalisation of MOF-Based Modelling Languages. In *TOOLS 2009*, volume 33 of *LNBIP*, pages 37–56. Springer, 2009.

[10] A. Rutle, A. Rossini, Y. Lamo, and U. Wolter. A Formalisation of Constraint-Aware Model Transformations. In *FASE 2010*, volume 6013 of *LNCS*, pages 13–28. Springer, 2010.

[11] G. Taentzer, C. Ermel, P. Langer, and M. Wimmer. Conflict Detection for Model Versioning Based on Graph Modifications. In *ICGT 2010*, volume 6372 of *LNCS*, pages 171–186. Springer, 2010.

[12] B. Westfechtel. A Formal Approach to Three-Way Merging of EMF Models. In *IWMCP 1010*, pages 31–41. ACM, 2010.