Approaching Collaborative Modeling as an Uncertainty Reduction Process

Romina Eramo
Università degli Studi dell'Aquila, Italy
romina.eramo@univaq.it

Alfonso Pierantonio
Università degli Studi dell'Aquila, Italy
alfonso.pierantonio@univaq.it

Gianni Rosa
Università degli Studi dell'Aquila, Italy
gianni.rosa@univaq.it

ABSTRACT

Model-Driven Engineering (MDE) technologies aim to support the growing complexity of software systems. Models are increasingly becoming large and unmanageable, and hence difficult to be understood by humans and processed by machines. As a consequence, multi-user environments are necessary to enable designers to create and refine large models in a collaborative manner enabling the engineering, modularization and reuse.

In this paper, we propose a model-driven approach to represent, manage and manipulate models edited in a collaborative manner. In particular, we propose to represent the solutions space (i.e, model versions) in an intensional manner by adopting a model with uncertainty. We define a plan to manage the uncertainty by selecting the desired design, to manipulate their collaborative models in manually or automatic way, and to exploit a collaborative environment for real time multi-user editing. The approach is showed by means of a motivating example that involves business models demonstrating the advantages of the proposed approach.

Keywords

Model-Driven Engineering, Collaborative Modeling, Uncertainty

1. INTRODUCTION

Complex software systems demand for both effective and specific approaches to keep under control the increasing software functionalities, heterogeneity, team sizes and geographical distribution of the developers. The growing maturity of Model-Driven Engineering (MDE) [27] technologies is leading industry to take advantage of their benefits in terms of productivity, quality and reuse [22, 7]. However, applying MDE in this setting requires research effort in order to achieve industry-scale tools.

As observed in [19], scalability is a critical concern for the industrial adoption of MDE. Large-scale tools have to cope with different topics; among many, creating and refining large models in a collaborative manner, working with model fragments obtained from partitions of large models, enabling the engineering, modularization and reuse of models and fragments, having transformation tools to cope with versioned models, and having infrastructures to storage and manage large models.

When projects and teams are large, modeling activity requires designers to divide their models in smaller partitions and work on them in a collaborative manner. In a typical scenario, multiple developers work on the same model in multiple branches, where each developer (or sub team) may initialize a repository branch and make modifications to this model; thus, versions grow rapidly. Traditional text based version control systems (e.g., SVN, Git) allow programmers to write code by locking a portion of work or to edit code collaboratively; in this case, programmers tend to frequently perform merge operations involving different versions of the same textual artifacts. Conversely, in software design, that is a complex and non-linear process, decisions are made at different development stages and obtaining the merged version early could not be desired. In fact, sometimes designers have not the complete, consistent and accurate information required to make a decision at design-time (or when the conflict occurs), that can be regarded as a source of *uncertainty* [24]. For instance, they could perform some analysis or wait for different view models before to be able to choose the best strategy. In these cases, they could be interested in postponing the version merging and continuing to operate on their model versions separately.

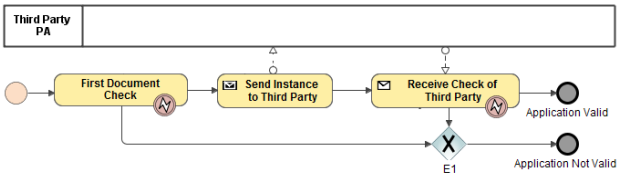


Figure 1: The *Check Application (CA)* sub-process

In this paper, we propose a model-driven approach to represent and manage models edited in a collaborative manner by leveraging uncertainty to a first-class status. To this end, we propose a metamodel-independent approach to represent the solutions space (i.e, multiple versions) in a compact and intensional manner by adopting a model with uncertainty. We define a plan to manage the uncertainty by selecting the desired design, to manipulate their collaborative models in manually or automatic way, and to exploit a collaborative environment for real time multi-user editing and handling of models. The technique is applied to a running example that involves business process models in the domain of Public Administrations (PAs) demonstrating the advantages of the proposed approach.

Structure of the paper. The paper is organized as follows. Section 2 sets the context of the paper through a motivating example that is used throughout the paper to demonstrate the approach. In

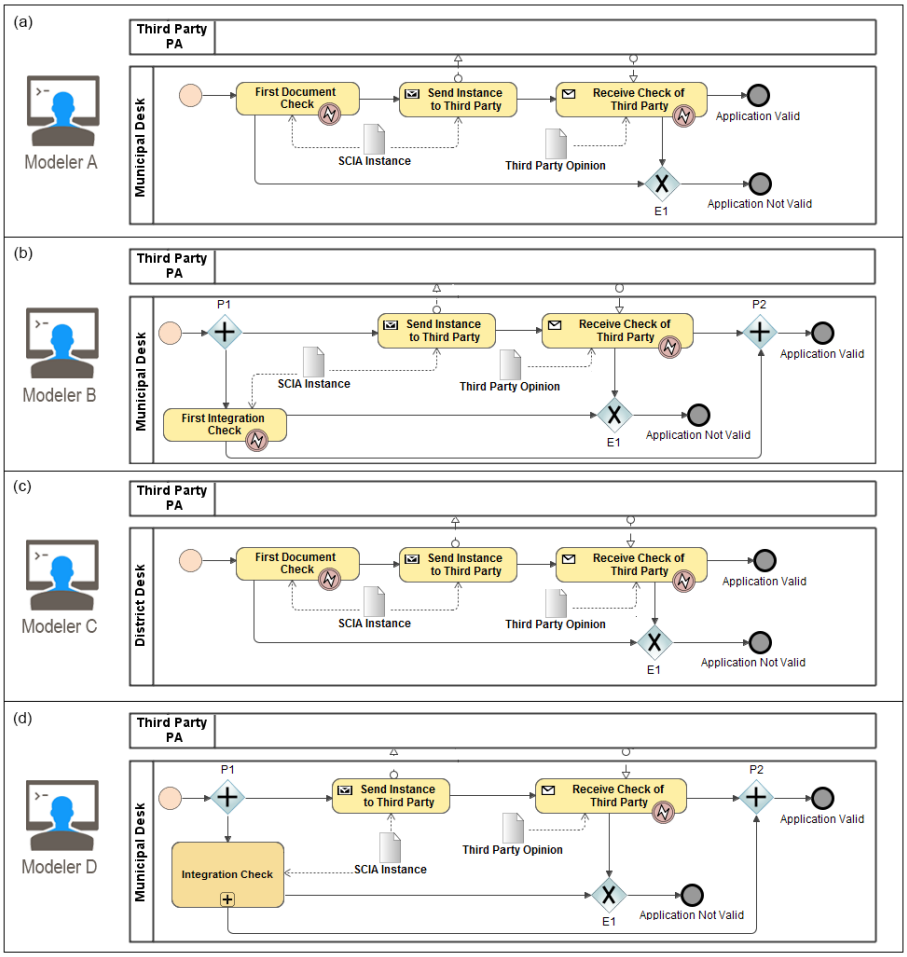


Figure 2: Parallel dependent changes in *CA* sub-process

Sect. 3 the model-driven approach to represent and manage collaborative models with uncertainty is presented and related challenges are discussed. In Sect. 4, related works are presented and finally Sect. 5 draws some conclusions.

2. MOTIVATING EXAMPLE

As aforesaid, the management of large models raises not obvious issues mainly related to the collaborative use of models. In order to illustrate our proposal, we consider a motivating example that takes place in the public administrations (PAs) sector.

In modern society, the role of PAs is undergoing a transformation from controller to proactive service provider. In most cases, the provisioning of services is a collaborative activity shared among different, possibly many, organizations that are in general quite interrelated. In order to provide efficient services to citizens and companies, public servants have to manage extremely complex processes and a large amount of information due to changes in law and regulations, societal globalization and fast technology evolution.

In this context, the Learn PAd[1] project [8] aims at developing a social, collaborative and holistic e-learning platform that enables process-driven learning and improvement of the process on a user-friendly basis of wiki pages enriched with additional documentation for a clearer understanding of the process together with guidance based on formalized models. The specification of business processes is usually done by means of standard notations like BPMN 2.0 [23] and concerns a sequence of activities that the administration executes in order to produce a service for the end user. Typically, it starts with *i)* receiving of some input (i.e. request, documentation), then *ii)* performing activities that add value (i.e., checks) using resources (i.e., humans, structures), and finally *iii)* producing an output.

For instance, Fig. 1 depicts a sub-process of a standard process, called *Titolo Unico*, where a citizen makes a request to municipality and third parties in order to obtain the permission to start a business activity according to the Italian law. Specifically, in the sub-process *Check Application (CA)* the employee in charge checks the received documents and verifies their validity. After that, the valid requests are sent to third party in PA in order to undergo the final verification and eventually to be accepted.

Models in PA are constantly updated and shared among different organizations, thus public servants daily deal with a large amount of data and a multitude of model versions, that is tedious and error-prone. In order to better understand the context, let us to consider the following scenario of collaborative modeling in PAs. Figure 2 depicts a scenario in which some modelers work collaboratively on the model in Fig. 1, that is considered as initial model. Then, it

[1]Model-Based Social Learning for Public Administrations is part of the program FP7-ICT-2013.8.2 Technology-enhanced learning. The project started on Feb 1, 2014 and terminate on Jul 31, 2016 with a cost of €3,535,000. For further detail please refer to http://www.learnpad.eu

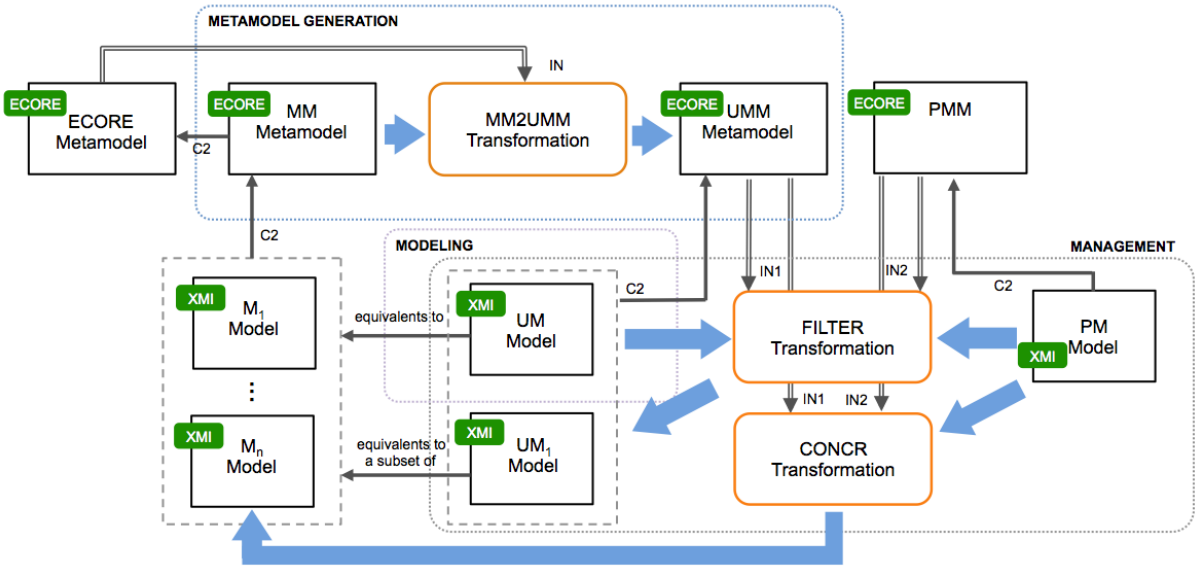


Figure 3: The overall architecture

is extended, refactored and splitted in a set of versions, so that the changes are *parallel dependent* causing conflict issues.

Let us to suppose that, `Modeler A` and `Modeler B` are two employees of the same organization and they are working together in a collaborative manner. Both of them consider the sub-process *Check Application (CA)* depicted in Fig. 1 from the Learn PAd repository. The `Modeler A` is an expert of PAs procedures and adds to the initial model a BPMN element of type `Lane`, named `Municipal Desk`, in order to establish the unit responsible for the activity and improve the process execution (see Fig. 2(a)). Then, an expert of the BPMN language, called `Modeler B`, modifies the model and adds a pair of `Inclusive Gateway` to logically separate the existing message flows between `Application valid` and `Application not valid`, as showed in Fig. 2(b).

In meanwhile, an other organization decides to re-use models stored in the Learn PAd repository. In particular, as depicted in Fig. 2(c), the `Modeler C` acquires the model in Fig. 2(a) and renames the lane `Municipal desk` into `District Desk`; in this way, the responsibility of the activities is given to a different organizational unit.

Finally, as shown in Fig. 2(d), in a third organization, the `Modeler D` performs same changes on the model in Fig. 2(b) in order to improve the quality of the business process. In particular, the element `First Integration Check` of type `Task` is modified in `Integration Check` of type `SubProcess` in order to modeling a more complex process that includes a set of activities.

Let us to suppose that, modelers continue to work on the model and at a later time each organization requires to merge the modelers versions and obtain the proper organization version.

In this scenario, the same model is modified by different organizations and for each of them one or more modelers worked on it. Furthermore, for the scope of the Learn PAd repository, models and their versions has to be maintained and traced, so that further organizations can access and re-use it. For these reasons, it is crucial to have a comprehensive model representation that enable the information tracing and the management of different model versions in an automatic manner. The scope of this paper is to represent the multitude of generated models into a model with uncertainty capable of encoding all the alternatives in one instance and allowing the management.

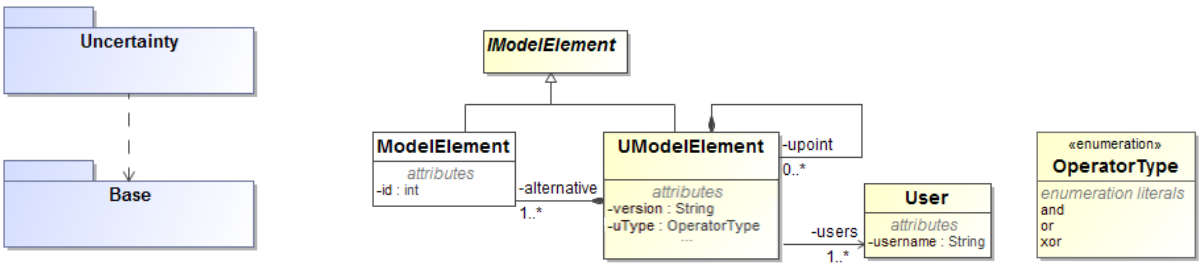
3. A MODEL-DRIVEN APPROACH TO COLLABORATIVE MODELING

In collaborative modeling, decisions may be made at different development stages requiring the designers to work with their model versions as long as uncertainty can be solved and a merged version can be obtained. In order to reduce the burden of managing a multitude of model alternatives, we present a metamodel-independent approach able to represent and manage *uncertainty in collaborative modeling* based on a revised basis in [24].

During the modeling, the same model is edited by multiple users. Changes introduced by modelers are maintained as alternative elements in the model producing uncertainty. During the management, modelers increasingly reduce (until resolve) such uncertainty by choosing among the alternatives. In particular, at each step, the modeler selects the desired alternative (i.e., making a model element in the solution space as certain), so that the uncertainty is gradually resolved by considering choices and dependencies. Note that, the two phases are not necessarily consecutive, in fact, at any time a modeler may decide to manipulate or manage the model or a part of it.

The overall architecture of the approach is depicted in Fig. 3. The *Metamodel generation* is realized by means of a model-to-model transformation written in ATL[2], called $MM2UMM$. It takes as input a metamodel MM conforms to Ecore[3] and generate the correspondent metamodel with uncertainty UMM. At this point, modelers may specify their collaborative models with uncertainty conform to UMM (see the UM model in the *Modeling* box). An UM model is semantically equivalent to a set of models $\{M_1..M_n\}$ conform to MM (i.e., all the candidate solution models represented in UM). The component *Management* allows modelers to select the desired solution models by means of two operations: $Filter$ and $Concr$. The $Filter$ operation allows to select elements in the solution space parametrically. It is based on an ATL model-to-model transformation; it takes as input the UM

[2]ATL Transformation Language: https://eclipse.org/atl/
[3]EMF Ecore: http://www.eclipse.org/modeling/emf/



(a) (b)

Figure 4: The Uncertainty Metamodel (UMM) Structure

model and a parameter (represented by a model PM conforms to the metamodel PMM), and outcomes another model conforms to UMM. In particular, the model UM is filtered by the transformation, so that the generated model UM_1 is semantically equivalent to a subset of the models $\{M_1..M_n\}$ according to the parameter expressed in PM. The $Concr$ operation allows to select the desired final model conforms to MM; it takes as input the UM model and outcomes all the solution space, i.e. a set of models conforms to MM. The overall architecture has been implemented within the Eclipse Modelling Framework (EMF)[4] [5].

3.1 Uncertainty metamodel

The uncertainty metamodel UMM is obtained by extending a *base* metamodel MM with specific connectives able to represent multiple alternatives asisen from collaborative modeling. These connectives denote the points of uncertainty where model elements belonging to a certain version are attached. Moreover, such points of uncertainty are traceable in order to permit the identification of specific merge strategies. The approach is metamodel independent, in fact its construction is realized by means of a model transformation and applies to any Ecore metamodel [14].

The uncertainty metamodel UMM structure is depicted in Fig. 4. As in the left-hand side of the figure, the metamodel is composed of two packages, the *Base* package that contains the elements of the base metamodel and the *Uncertainty* package that contains the elements that extend the base metamodel in order to allow the specification of points of uncertainty.

In particular, the uncertainty metamodel UMM is generated extending the base metamodel as following:

1) the metaclass `User` with attribute `username` that specifies the author making the editing is added;

2) the metaclass `OperatorType` that enumerates the logical operator literals `AND`, `OR` and `XOR` is added;

3) for each parent class *C* in base metamodel, such that it does not specialize other metaclasses:

   4.1) a corresponding abstract metaclass *IC* is created such that *C* specializes *IC*. Such interface allows to choose among a certain element or an uncertain one.

   4.2) a metaclass *UC* that extends *IC* is added. It specifies an association `alternatives` of type `C` and multiplicity `1..*` that relates the alternative elements to the uncertainty point of type `C`; an association `upoints` of type `UC` and multiplicity `0..*` enabling to nest uncertainty points; the attribute `uType` of type `OperatorType` for specifying the logical operator connecting the alternatives; the association `user` of type `UUser` with multiplicity `1..*` for specifying the authors;

The aforementioned procedure is implemented as an endogenous model-to-model transformation implemented in ATL.

The main advantage of such representation technique is that the uncertainty represented by a set of alternative models is leveraged to a first-class status. Hence, a complete set of models can therefore be manipulated as whole, for instance with an automated transformation (as done for instance in [16]), without iterating over each individual in the set.

3.2 Models with uncertainty

According to the scenario described Sect. 2, we assume that in PAs, modelers are interested to specify complex processes by means of the standard notation BPMN and to exploit the Learn PAd repository where models are constantly updated and shared among different organizations.

Let us considering the BPMN 2.0 [6] language as mentioned in the motivating example in Sect. 2. The language specification supports different modeling conformance levels (e.g., process modeling, process execution, etc.) that makes it as rich and complex. For the sake of understanding and brevity, we provide a restricted version of the BPMN 2.0 metamodel that concerns the process modeling conformance level and allows us to specify processes at a certain abstraction level and to model the running example discussed in Sect. 2. Starting from the simplified version of the BPMN metamodel, the correspondent uncertainty metamodel UBPMN is obtained as illustrated in the previous section [7].

At this point, modelers may specify their models with uncertainty. In particualr, by using the proposed approach, the model in Fig. 2 can be represented by means of the model with uncertainty depicted in Fig. 5. The sub process `Check Application` is composed of a common part shared among all the modelers (that is the lane `Third Party PA`) and an uncertain part in which model elements are independently modified by the four modelers. In particular, the point of uncertainty of type `ULane` includes the nested uncertainty point `Municipal Desk` edited by `Modeler A`, `Modeler B` and `Modeler D`, and the nested uncertainty point `District Desk` edited by the `Modeler C`. Each uncertainty point is composed of a set of elements that are in common to the modelers and nested uncertainty points that group the modeler's modifi-

[4]EMF Modelling Framework: http://www.eclipse.org/modeling/emf/
[5]The implementation is available at http://jtl.di.univaq.it/

[6]For reason of readability, we omit the description of the language. The interested reader may refer to the OMG specification [23].
[7]The implementation is available at http://jtl.di.univaq.it/

cations. For instance, the uncertainty point edited by the `Modeler B` and `Modeler D` is composed of common elements such as `Start Event`, `Send Instance to Thirdy Part`, etc., and a set of nested uncertainty points, for instance the nested uncertainty point edited by the `Modeler B` include the sequence flow `P12FirstIntegrationCheck`, the task `First Integration Check`, etc.

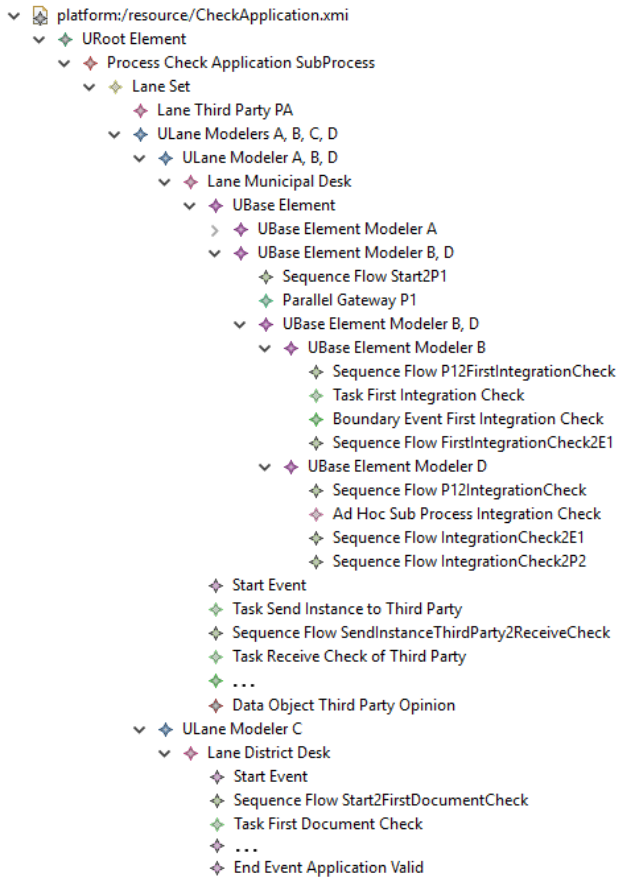


Figure 5: The *CA* sub-process in Fig. 2

Models with uncertainty may over-approximate the solution space because of their combinatorial nature [24]. For instance, the scenario in Fig. 2 suggests that only one lane for the same activity can exist in the final model. However, the generated model with uncertainty could admit also models with both the lanes giving place to more solutions than those expected. Therefore, this outcome is avoided by using logical operators that allowing modelers to constrain and restrict the solutions to the ones depicted in the Fig. 2.

Listing 1 shows a fragment of the model with uncertainty representing the alternative solution models generated by the collaborative modeling in Fig. 2. It is given in its XMI [8] format. In particular, the `LaneSet` is composed of a model element of type `Lane` named `Third Party PA` (line 8) and an uncertainty point of type `ULane` (line 9). The latter contains two nested uncertainty points: the lane `Municipal Desk` (line 11) edited by the users `Modeler A`, `Modeler B` and `Modeler D`, and the lane `District Desk` (line 22) edited by the users `Modeler C`. Each uncertainty point is composed of its alternative elements and/or other nested uncertainty points. For instance, `Municipal Desk` contains the element of type `StartEvent` (line 12) and the uncertainty point edited from `Modeler A` with a set of alternative elements (such as the element named `First Document Check` of type `Task`, in line 16).

```
1 [...]
2 <users username="Modeler_A"/>
3 <users username="Modeler_B"/>
4 <users username="Modeler_C"/>
5 <users username="Modeler_D"/>
6 [...]
7 <laneSets xsi:type="LaneSet">
8   <lanes xsi:type="Lane" name="Third␣Party␣PA"/>
9   <lanes xsi:type="ULane" users="//@users.0//@users.1//
        @users.2//@users.3" />
10    <upoints users="//@users.0//@users.1//@users.3" uType
          ="AND">
11      <alternatives name="Municipal␣Desk">
12        <partitionElement xsi:type="StartEvent" />
13        [...]
14        <partitionElement xsi:type="UBaseElement">
15          <upoints users="//@users.0" uType="AND">
16            <alternatives xsi:type="Task" name="First␣
                  Document␣Check"/>
17            [...]
18          </upoints>
19        </partitionElement>
20        [...]
21    <upoints users="//@users.2" uType="AND">
22      <alternatives name="District␣Desk">
23      [...]
24  </lanes>
25 </laneset>
26 [...]
```

Listing 1: A fragment of a model conforms to UBPMN

3.3 Interoperability between models

The proposed approach allows to use models with uncertainty in order to represent design alternatives in collaborative models. The approach aims to respond to the need for techniques indeed to the engineering, modularization and reuse of large models and complex modelling enabling their flexible combination.

As said, modelers edits their modification on the model with uncertainty that semantically correspond to a set of models conform to the base matamodel (solution space); then alternatives has to be selected and/or merged in order to obtain the desired (final) model version. Dependencies and conflicts in models cause issues related to consistency checking that has to be tackled.

However, a number of operation may be needed to give the designer a more effective support and to achieve a complete interoperability between the base and the uncertainty metamodels, as shown in the next section.

3.4 Management of models with uncertainty

Once the uncertainty metamodel UMM is generated and modelers start to edit the model UM conforms to it in a collaborative manner, it is important to achieve the mean to manage the arising uncertainty. As said, multiple editing generates multiple alternatives; modelers may iteratively reduce the uncertainty by selecting the desired alternatives in a parametric manner and finally, when the uncertainty is totally resolved, the final model representing the desired design model is obtained. Note that the modeler may intentionally manage a sub-set of uncertainty points and postpone the complete resolution.

To this end, we provide an operation called $filter$ that permits to the modelers to reduce the uncertainty according to a given property. In particular, the $filter$ operation is a mapping defined as following:

$$\mathit{filter} : P \times UMM \rightarrow UMM$$

that for any model with uncertainty $UM \in UMM$ and a parameter p defined over the metamodel MM returns a model $UM' \in$

[8] http://www.omg.org/spec/XMI/

UMM where the uncertainty is reduced according to p (i.e., the solution models that not satisfying a given property are removed).

The operation is implemented by means of a model-to-model transformation called $Filter$ written in ATL. As said, it takes as input the UM model and a parameter (represented by a model PM conforms to the metamodel PMM), and outcomes a model UM_1 conforms to UMM, that is filtered according to PM. In particular, the generated model UM_1 is included in UM, that is semantically equivalent to a subset of the models $\{M_1..M_n\}$ according to the parameter expressed in the model PM.

When the uncertainty is solved, the outcome is a final model without uncertainty (i.e., concretization model). Along with an incremental management, we provide to the designer the possibility to directly select one or more (or all as well) concretization models.

The concretization operator is defined in order to return the whole set of the concretizations represented in a model with uncertainty. More formally, such operator *concr* is a mapping defined for any base metamodel MM:

$$concr : UMM \rightarrow MM$$

which takes an arbitrary model with uncertainty $UM \in UMM$ and returns its k concretizations $\langle M_1 \cdots M_k \rangle \in MM$. By entangling the uncertainty with the specialized metamodel construction UMM, it is of crucial relevance to be able to retrieve the concretizations as instances of their base metamodel. As one can expect, the reasons why this is important can be numerous including the necessity to keep on exploiting, utilizing, and capitalizing on existing tools.

For instance, the uncertainty of the model in Fig. 5 can be solved in this way: *i)* the versions of the `Modeler A` and the `Modeler B` are merged together in order to obtain a final model of the first organization, *ii)* the versions of the `Modeler C` is selected to obtain a final model of the second organization, *iii)* the versions of the `Modeler D` is selected to obtain a final model of the third organization, and *iv)* a new version is selected by a modeler of a different organization.

Let us suppose that the modeler's of the first organization need to solve the uncertainty and obtain a final model, thus the $Filter$ transformation is applied as in the following:

- the modeler selects the task `First Integration Check` belongs to the uncertainty point edited by the `Modeler B`. Such parameter is given as input of the transformation; since the uncertainty point is involved in a logical operation `AND` all the elements in the uncertainty point are taken as certain part of the final model;
- the modeler selects the parallel gateway `P1` belongs to the uncertainty point edited by `Modeler B` and `Modeler D`. Since the uncertainty point is involved in a logical operation `AND` both the elements `Start2P1` and `P1` are taken as certain and the second point of uncertainty is solved; finally
- the lane `Municipal Desk` is selected and the last uncertainty point is solved.

The final model is depicted in Fig. 6.

3.5 Manipulation of models with uncertainty

In this work, we aim to support designers in specifying collaborative models and perform operation on them; for instance, designers may need to performs operations of model transformation on versioned models.

Model transformation techniques typically operate under the assumption that models do not contain uncertainty. Actually, when a modeler is still working on a versioned artifact, she may delay the application of model transformations until the information needed to perform the merge becomes available, or make premature resolutions of the existing versions in order to apply the model transformations, thus creating a risk that these resolutions are incorrect. With our approach, modelers may resolve the uncertainty by considering only one version of model, otherwise they may decide to postpone the resolution, so they they need to be supported in their operations (e.g., model transformations should be applied on models with uncertainty [16]). To this reason, we aim to provide a technique to adapt existing model transformations so that they can be applied to models even if they contain uncertainty [11].

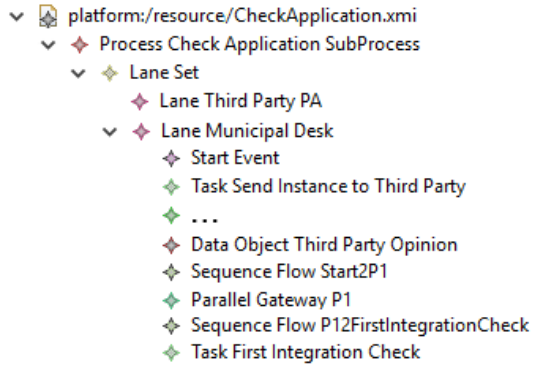


Figure 6: A possible resolution of the model in Fig. 5

Furthermore, by means of our approach modelers can therefore easily grasp the differences among candidate models and consistently make their decision without manually inspecting each model individually.

3.6 Modelling system

Our approach requires to be integrated on a collaborative modeling platform that supports multi user in real time and large models (of the scale of millions of model elements) paying attention to efficiency and performance.

The existing model versioning systems are mainly designed as version control systems (VCS), such as CVS or SVN. They support the long transaction model that assume that designers perform commit of large part of the work with respect to a certain previous version [19]. These systems are asynchronous and provide merging features and mechanisms able to automatically detect and solve conflicts by means of primitives like push, pull, commit and merge.

A modeling system supporting our approach has to be able to deal with models with uncertainty and to allow multi-users to collaboratively work with their models in a way that the uncertainty is transparent. The architecture is designed to support both offline and online collaboration in a multi-user and multi-device environment, and to provide a model access layer (transaction management, queries, views and manipulation), and an adaptation layer for the integration of access control and authentication (these may be provided by additional middleware such as the web server that hosts the actual communication between server and client). The environment will be implemented as an Eclipse Plugin embedded in the EMF framework[9].

Let us to consider that modelers have access to a repository that provide different organizations to store and manage their models (as in the example in Sect. 2). Thus, modelers need to: *i)* visualize the repository and select the desired model by exploit a standard no-

[9]https://eclipse.org/modeling/emf/

tation of feature diagrams [10], and *ii)* edit their artifact by means of an XMI editor and/or a visual editor embedded in the EMF environment. Going in more details, when the designer select a model, the *concr* operation extract the specific model from the model with uncertainty. Whereas, when modifications are made from a modelers and the *push* operation is performed, the system as to consequently update the model with uncertainty (i.e., changes are added as alternatives within uncertainty points).

4. RELATED WORK

Uncertainty is ubiquitous within contexts such as requirements engineering [12], software processes [18] and adaptive systems [26]. Uncertainty management has been studied in many works, often with the intention to express and represent it in models. In [15], *partial models* are introduced in order to let the designer specify uncertain information by means of a base model enriched with annotations and first-order logic. In [25] a formal approach called MAVO is proposed and applied to design models in order to express and allow automated reasoning in presence of uncertainty. In [24], an approach to manage the uncertainty generated as the outcome of a non-deterministic model transformation is proposed. The JTL bidirectional transformation engine [9] is revised to accommodate a intensional semantics, that permits a transformation to natively generate a model with uncertainty instead of a myriad of models. In particular, a model with uncertainty provides a compact representation of the solution space for the sake of usability and effectiveness. Model transformation techniques typically operate under the assumption that models do not contain uncertainty. Nevertheless, the work in [16] proposes a technique for adapting existing model transformations in order to deal with models containing uncertainty. In [11], the authors propose a bidirectional model transformation framework to cover incomplete transformations producing a multitude of possible solutions to consistency restoration. This multitude is managed in an intentional manner via models with built-in uncertainty (as proposed in [24]) and allows to these to be included as input in a model transformation process.

Feature models are a popular formalism for managing variability in software product lines. In this context, the Familiar project [4] provides an executable language that supports manipulating and reasoning about feature models and fully integrated modeling tool.

Despite a number of works have been proposed to cope with large and complex software systems, a new line of research is imperative in order to achieve scalability across the MDE technical space and to enable MDE to remain relevant [20]. Among the existing approaches that aim to cope with different topics concerning scalability, such as modeling languages, transformations, collaborative modeling, persistence and so on, we consider the followings as most related to our work. In [17] the authors aim at providing compositional technology and techniques for a language-independent model modularization. In particular, they proposed a way of extending modeling languages with component capabilities. The implementation is based on EMF Eclipse. However, the work does not consider issues related to consistency preserving when models are interconnected.

The state of the art in collaborative modeling includes several prototype attempts of collaborative systems that are more closely aligned with version control systems (VCS), such as CVS or SVN [21, 6, 1, 2]. Recently, the Eclipse collaborative modeling community proposed some convergent approaches. The Eclipse Modeling Framework Connected Data Objects (CDO) [5] is a model repository for EMF models supporting version management. EMF Compare [13] is a model comparison, differencing and merging tool often used in combination with traditional VCS. EMFStore [3] is a model versioning framework for EMF able to deal with conflict management.

They provides a still immature means for collaborative work and locking and conflict management [20]; they are more closely aligned with version control system such as CSV and SVN and implemented within EMF Eclipse. Moreover, they do not fulfill the need to have a flexible and scalable framework able to work independently from the specific domain-specific language or ad-hoc architecture.

5. CONCLUSION

MDE is now practiced in industry, and on very large complex systems engineering projects and problems. From this arises the need to enable designers to create and refine large models in a collaborative manner enabling the engineering, modularization and reuse.

In this paper, we proposed to deal with the uncertainty arisen from the collaborative editing of models by means of a metamodel-independent approach able to represent models with uncertainty. Furthermore, we proposed to manage the uncertainty by means of two kinds of operations to reduce or resolve the uncertainty. We plan to realize a model transformation approach able to deal with model with uncertainty (i.e., model with uncertainty are input and/or output of the transformation engine). Finally, we plan to integrate our approach in a collaborative environment for real time multi-user editing. We aim to extend the framework to help the modelers to make decisions among proposed design versions. The versions are initially partitioned, constrained, abstracted, and graphically visualized to the user. Then, when decisions are made, they are stored and used to drive subsequent decisions.

6. ACKNOWLEDGMENT

This research was supported by the EU through the Model-Based Social Learning for Public Administrations (Learn Pad) FP7 project (619583).

7. REFERENCES

[1] The AMOR project. Adaptible model versioning project website. 2009. http://modelversioning.org.

[2] Eclipse Modeling Team rFamework proposal. 2011. http://www.eclipse.org/proposals/mtf/.

[3] EMFStore project. 2011. http://eclipse.org/emfstore.

[4] FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning). 2011. http://familiar-project.github.io/.

[5] Connected Data Objects model repository (CDO) project. 2012. http://eclipse.org/cdo.

[6] K. Altmanninger, G. Kappel, A. Kusel, W. Retschitzegger, M. Seidl, W. Schwinger, and M. Wimmer. Amor–towards adaptable model versioning. In *1st International Workshop on Model Co-Evolution and Consistency Management, in conjunction with MODELS*, volume 8, pages 4–50, 2008.

[7] P. Baker, S. Loh, and F. Weil. *Procs of MoDELS 2005*, chapter Model-Driven Engineering in a Large Industrial Context – Motorola Case Study, pages 476–491. 2005.

[8] A. Bertolino. Model-Based Social Learning for Public Administrations (Learn PAd). EU-FP7 project. Description of Work, 2014.

[9] A. Cicchetti, D. Di Ruscio, R. Eramo, and A. Pierantonio. JTL: a bidirectional and change propagating transformation language. In *SLE10*, pages 183–202, 2010.

[10] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration using feature models. In *International Conference on Software Product Lines*, pages 266–283. Springer, 2004.

[11] Z. Diskin, R. Eramo, A. Pierantonio, and K. Czarnecki. Incorporating uncertainty into bidirectional model transformations and their delta-lens formalization. In *Procs of the Bx 2016 Workshop, co-located with ETAPS 2016*, pages 15–31, 2016.

[12] C. Ebert and J. D. Man. Requirements uncertainty: influencing factors and concrete improvements. In *Procs. of ICSE*, pages 553–560. ACM Press, 2005.

[13] Eclipse Foundation. EMF Compare. 2010. http://www.eclipse.org/modeling/emft/?project=compare.

[14] R. Eramo, A. Pierantonio, and G. Rosa. Uncertainty in bidirectional transformations. In *Procs. of MiSE 2014*, 2014.

[15] M. Famelis, R. Salay, and M. Chechik. Partial models: Towards modeling and reasoning with uncertainty. In *ICSE*, pages 573–583, 2012.

[16] M. Famelis, R. Salay, A. D. Sandro, and M. Chechik. Transformation of models containing uncertainty. In *MoDELS'13*, pages 673–689, 2013.

[17] F. Heidenreich, J. Henriksson, J. Johannes, and S. Zschaler. *Transactions on Aspect-Oriented Software Development VI*, chapter On Language-Independent Model Modularisation, pages 39–82. 2009.

[18] H. Ibrahim, B. H. Far, A. Eberlein, and Y. Daradkeh. Uncertainty management in software engineering: Past, present, and future. In *CCECE*, pages 7–12. IEEE, 2009.

[19] D. S. Kolovos, R. F. Paige, and F. Polack. The grand challenge of scalability for model driven engineering. In *Models in Software Engineering, Workshops and Symposia at MODELS 2008*, pages 48–53, 2008.

[20] D. S. Kolovos, L. M. Rose, N. D. Matragkas, R. F. Paige, E. Guerra, J. S. Cuadrado, J. de Lara, I. Ráth, D. Varró, M. Tisi, and J. Cabot. A research roadmap towards achieving scalability in model driven engineering. In *Procs of BigMDE 2013*, page 2, 2013.

[21] G. Kramler, G. Kappel, T. Reiter, E. Kapsammer, W. Retschitzegger, and W. Schwinger. Towards a semantic infrastructure supporting model-based tool integration. In *Procs of GaMMa 2006*, pages 43–46. ACM, 2006.

[22] P. Mohagheghi, M. A. Fernandez, J. A. Martell, M. Fritzsche, and W. Gilani. *Models in Software Engineering: Workshops and Symposia at MODELS 2008*, chapter MDE Adoption in Industry: Challenges and Success Criteria, pages 54–59. 2009.

[23] B. P. M. OMG. Notation (BPMN) 2.0. *Object Management Group: Needham, MA*, 2494:34, 2011.

[24] G. R. Romina Eramo, Alfonso Pierantonio. Managing uncertainty in bidirectional model transformations. In *SLE 2015*, 2015.

[25] R. Salay, M. Chechik, J. Horkoff, and A. D. Sandro. Managing requirements uncertainty with partial models. *Requir. Eng.*, 18(2):107–128, 2013.

[26] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein. Requirements-aware systems: A research agenda for re for self-adaptive systems. In *RE*, pages 95–103. IEEE, 2010.

[27] D. Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *Computer*, 39(2):25–31, 2006.