Comparative Analysis of Collaborative Approaches for UsiXML Meta-Models Evolution

Boukhebouze, Mohamed; Koshima, Amanuel; Englebert, Vincent; Thiran, Philippe

Published in:
Proceedings of the UsiXML-EICS workshop

Publication date:
2010

Document Version
Early version, also known as pre-print

Comparative Analysis of Collaborative Approaches for UsiXML Meta-Models Evolution

Mohamed Boukhebouze, Amanuel Koshima, Philippe Thiran, Vincent Englebert
PReCISE research center, University of Namur, Belgium,
{mohamed.boukhebouze, amanuel. koshima, philippe.thiran, vincent.englebert}@fundp.ac.be

ABSTRACT
The UsiXML project is dedicated to define a user interface description language called UsiXML. This language specifies the different abstract levels of a user interface by using set of model according to Model-driven engineering (MDE). In the UsiXML project, several industrial and academic members work in collaborative manner to evolve meta-models of UsiXML models. Several collaborative approaches can be applicable. For instance, the control of meta-models modifications can be centralized or not. In the paper, we aim at proposing the most appropriate approach for the UsiXML meta-models evolution. To achieve this, we present a comparative study of the different collaborative design approaches.

Keywords
UsiXML, Meta-Model evolution, Collaborative design, Centralized approach, Decentralized approach, control of modification.

INTRODUCTION

UsiXML (USer Interface eXtensible Markup Language) offers a common way to specify a User Interface (UI) independently of any target programming language that is intended to implement it [1]. This language allows describing an UI in four main levels of abstraction: task & domain, abstract UI, concrete UI, and final UI. On the basis of these four levels, UsiXML proposes a set of models. For each model, UsiXML defines their meta-model by using the UML Class diagram notation. Each meta-model can then be expressed in XML (XSD [2], XMI [3]), or in OWL [4]). All these notations are required by standardization organizations like OMG or W3C.

The evolution of the UsiXML meta-models is carried out by an ITEA project involving industrial and academic members [5]. Evolving UsiXML meta-models requires a collaborative system that supports interactive work between members of the ITEA project. This collaborative system is intended to manage the UsiXML meta-models evolution by:

- Offering a repository of the UsiXML meta-models,
- Managing the members and their roles,
- Supporting the evolution of UsiXML meta-models in collaborative way; which involves managing concurrent modifications.

Such a system can be designed in different ways. First, the meta-models can be stored in a *centralized repository* (a unique repository for all the members) or in *decentralized repositories* (a local repository per member). Secondly, the modifications can be either *controlled* by a central authority (called the owner) or *delegated* to every member. In the first case, the modifications issued by the members are sent to the owner of a meta-model who can commit or reject them. In the second case, each member has the control of his own version and can integrate the modifications of the peer-designers in his version. The convergence of these meta-models to a unique consolidated version is the result of a workgroup communication. In both cases, a reconciliation process is always required since syntactic or semantic discrepancies have to be resolved [6].

Each of these approaches has its strengths and its weaknesses. Let us mention the most relevant ones. On one hand, the centralized approach allows controlling the reconciliation process by an owner but this exclusive role can lead to slow down the modifications frequency. On the other hand, the decentralized approach can support a higher rate of modifications but it requires a more complex reconciliation process.

In this paper, we propose a comparative study of the different collaborative design approaches. The study is aimed at proposing the most appropriate approach for the UsiXML meta-models evolution.

The rest of this paper is organized as follows. In Section 2, we describe the different approaches for the collaborative design of meta-models. In Section 3, we conduct a comparative study on these approaches based on a different comparison criterion. In Section 4, we apply our comparative study on the collaborative design of UsiXML meta-models. We conclude this paper by discussing the most appropriate approach for UsiXML.

COLLABORATIVE META-MODEL EVOLUTION APPROACHES

According to Schmidt et al. [6], *collaborative work* is characterized by a mutual interdependence of work among multiple actors to produce a specific product or service. *Interdependence work* means that actors share the same field of work (i.e. document, codes of a program). *Mutual work* means that the work of one actor relies on the quality and timeliness of other actors.

By analogy with the Schmidt's definition, *collaborative meta-model evolution within a project* is characterized by a mutual interdependence of meta-model evolution among multiple project members. All members share a common meta-model by using a central repository or a local one and they interact each other by modifying the version of their meta-model.

For this reason, four approaches can be considered:

- Centralized approach with modification control;
- Centralized approach without modification control;
- Decentralized approach with modification control;
- Decentralized approach without modification control.

Note that these approaches define how to design, store, and evolve the meta-model. In the following sections, we detail these approaches.

Centralized approach with modification control

In this approach, we consider that there is one central meta-model repository. The meta-model has its owner who is the only member that can modify it. The other members can participate in the design of a meta-model by issuing a modification request to its owner. It is under the responsibility of the owner to commit or not the modification requests. As such, the owner controls the modification of the meta-model repository (Figure 1).

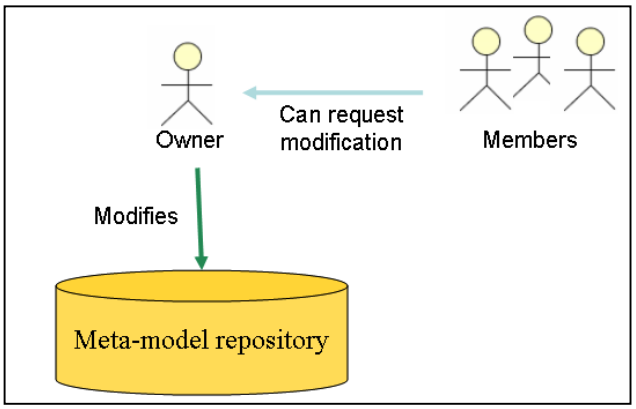

Figure 1: Architecture of centralized approach with modification control

Centralized approach with modification control implements an administrative hierarchy, like any classical design, project by using a central authority, which is called owner. This owner can be the leader of project or the expert designer. However, when the design project is large, the dynamism of modifications can become slow. This is due to the fact that a large project implies a high number of members. Each member can request a modification. This can imply a considerable time to handle all the request modifications by the owner. Finally, note that this approach can be implemented on the top on CVS system as proposed in [7]. In CVS system, the changes are committed to a master repository (a owner in our case) before they are propagated to other users (members in our case).

Centralized approach without modification control

This approach considers that there is one central meta-model repository that can be modified by any project member at any time. This assumes that a modification can overwrite another modification without any control (Figure 2). This means that any member can freely modify meta-model elements without informing the other members. Moreover, the repository stores only the recent modifications as they always overwrite the previous modifications on a same element.

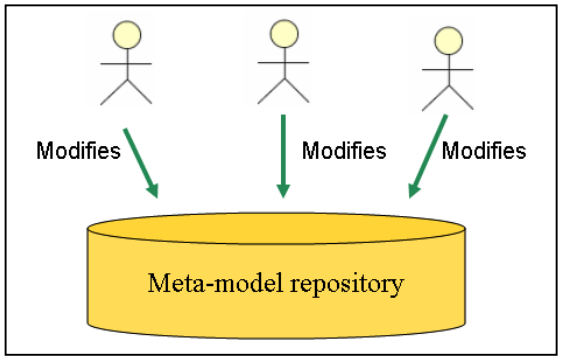

Figure 2: Architecture of centralized approach without modification control

The absence of modification control lets designers modify meta-models faster in comparison with the above approach. However, they are continuously being modified so that it is difficult to get a stable version. Indeed, obtaining a stable version is problematic even if there exists a methodological agreement between members on how to cooperate. A way to circumvent this problem is to partition the meta-model into distinct areas of interest. As such, only one member can modify one area. Consequently, conflicts are avoided during concurrent accesses of views. However, partitioning of meta-model into different distinct areas of interest is a challenging issue.

Finally, note that this approach is quite suitable for the collaborative text editing where user can create and edit documents while collaborating in real-time with other users (e.g., Google document[1]).

Decentralized approach with modification control

In this approach, we consider that all the members have their own local repository. Each member is free to access or modify his local repository. As such, a member can create, read, update or delete (CRUD) an element of local meta-model. In the decentralized approach, we suppose that all the CRUD actions on a local repository are logged and stored in a local journal. This journal keeps traces of the local meta-model modifications [8]. The local journals are

[1] http://googledocs.blogspot.com/

exchanged between members via a coordinator (see Figure 3). The coordinator has the responsibility to control the interactions among members based on different policies such as:

- Centralized consolidation: the coordinator aggregates and consolidates the modifications sent by the members. The consolidated modifications are then sent to all the members so that they can modify their local copies. As such, their local meta-model repository can be consistent each other.
- Decentralized consolidation: the coordinator propagates each modification (one by one) to members (without aggregation and consolidation). This means that modifications are propagated in FIFO order to members. It is under the responsibility of each member to consolidate his meta-model wrt the propagated modifications. Local members cannot reconsider them. Hence, their current modifications can be overwritten and invalidated by the propagated modifications.

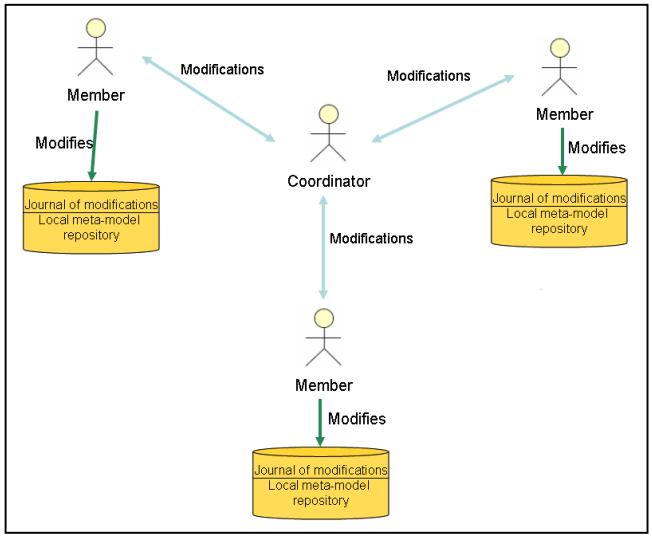

Figure 3: Architecture of decentralized approach with modification control

This approach gives a freedom to members in terms of time (work at any time) and space (work in a local repository).

In the second policy, a member who firstly sends modifications could be determined how the meta-model is evolving. As result, there might be competition among members to become the first one to send modification. As such, this could hamper the collaborative work.

Decentralized approach without modification control

In this approach, we consider that all the members can freely modify the meta-model repository. As the previous approach, the CRUD actions are stored in a local journal. Since there is no coordinator to control meta-model modifications, a member sends his local journal (modifications) to all the other members (see Figure 4). Afterwards, each member votes yes/no for the proposed change; if the majority of members accept the modification, it is implemented in all local repository of each member. If not, the modification is rejected.

Like the third approach, this approach ensures freedom in time and space for members. However, the reconciliation process is more difficult than all the above approaches. This is due to the fact that it is not easy to get consensus about the suggested changes among all the members. Finally, note that this approach can be implemented by using a peer-to-peer architecture as it is proposed by Mougenot et al. [9]. However, unlike the decentralized approach without modification control, Mougenot et al. propose a reconciliation strategy based on an order computed by Lamport clocks.

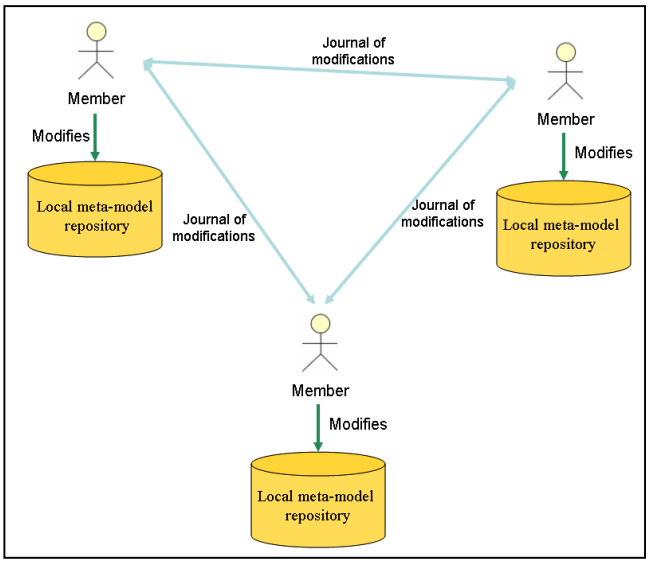

Figure 4: Architecture of decentralized approach without modification control

COMPARATIVE ANALYSES OF COLLABORATIVE META-MODEL EVOLUTION APPROACHES

In the previous section, we presented several approaches for supporting the collaborative editing of meta-models. Each of them addresses specific needs that we now compare according to three categories.

Project

Modification dynamism: a project may accept a slow rate of modifications (in terms of days), or may require a faster rate (in terms of minutes).

Project size: a project can involve a small working group (e.g. open space or department of an enterprise) or a medium working group (e.g. enterprise or organization) or a large working group (e.g. multi-enterprise collaboration).

		Project criterion		Meta-model criterion	Implementation criterion		
		Dynamism of modifications	Size of project	Expression of meta-model	Request of modification	Role of member	Reconciliation
Centralized approaches	With control	Slow	Large	Formal or informal expression	Need requests	Role based	A priori
	Without control	Fast	Small	Formal or informal expression	Optional	Not role based	No reconciliation
Decentralized approaches	With control	slow/medium	Large	Formal expression	Optional	Role based	A priori
	Without control	Fast	Medium	Formal expression	Optional	Not role based	Posteriori

Table1: Comparative analyses of collaborative meta-model evolution approaches

Meta-model

Expression of meta-model: a meta-model can be expressed formally by using a formal language (e.g. XMI, XSD) or informally by using a bitmap image or the natural language.

Links between meta-models: the meta-models can be linked each other by sharing some elements or by defining binding between meta-models. In this paper, we suppose that the meta-models are not related together. This hypothesis is restrictive, as it is not confirmed in the UsiXML realm. Nevertheless, as the links between the meta-models have not been so far formally elicited, we can consider that problem as out of scope of this paper.

Implementation criterion

Request for modification: requests for modifications can be expressed in a formal or informal way. This depends on the language used to define the meta-models. A formal request could be expressed as "patch file", OCL statements, etc. Informal requests are just textual descriptions. The first form enables an automation of the reconciliation process while the second one cannot be automated.

Role of member: some implementations require assigning roles (owner or not) to members to let the cooperative scenario go off.

Reconciliation: the reconciliation process can be applied *a priori* to avoid divergent versions. It can also occur *a posteriori* to solve the problem once it is observed. These approaches are also called resp. pessimistic and optimistic [10]. Note that it is also possible to have no *reconciliation* if it is not required to have a common stable version. In this case, we assume that each member maintains his own version of the meta-model in a consistent way. The member is free to integrate or not the modifications of the other members [11].

Table 1 depicts a summary of this criterion. This table can be used as a decision table based on the nature of the project. As explain above, collaborative approaches with control cannot support a fast dynamism of modifications. This is due to the fact that these approaches require a central point that handles the reconciliation process. On the contrary, the collaborative approaches without control support a fast dynamism of modifications. However, they cannot be applicable to a large project because it is difficult to achieve a consensus among members about modification.

The collaborative approaches with control can support a large project. Indeed, these approaches implement a central authority that can be the leader of project or the expert designer who manages modifications. Hence, large projects could be easily managed.

According to Table 1, the decentralized approaches require a formalized meta-model since these approaches use it to exchange journals of modifications among members. Moreover, formal expression of meta-models could avoid conflicts and confusions. On the top of that, the formalization of meta-model helps to automatically generate the CRUD journal when a meta-model is modified. Whereas centralized approaches could express meta-model into formally or informally.

Finally, the aforementioned controlled approaches implement a priori reconciliation to ensure that the modification is consistent with the central point. On the other hand, the above approaches without control use posteriori reconciliation. This is due to the fact that there is no central point that evaluates the proposed modifications.

USIXML PROJECT

In this section, we apply our comparative study on UsiXML project. UsiXML is a User Interface Description Languages (UIDL) that uses a Model-Driven Engineering (MDE) for specifying a UI at an implementation-independent level [12]. The UI specifications are usually specified in different models: each one denotes a facet of the interface characteristics. The complete set of UsiXML models comprises [13]: (1) Task model: it describes the interactive tasks as viewed by the end user interacting with the system; (2) Domain model: it is a description of the classes of objects manipulated by a user while interacting with a system; (3) Abstract User Interface (AUI) model: it is a model that represents a canonical expression of the renderings and manipulations of the domain concepts and functions in a way that is independent of any interaction modality and computing platform; (4) Concrete User Interface (CUI) model: it is a model that allows the specification of the presentation and behavior of a UI with elements that can be perceived by the users; (5) Mapping model: it contains a series of related mappings between models or elements of models; (6) Transformation model: it contains a set of rules enabling the transformation of one specification (at a certain level of abstraction) into another or to adapt a specification for a new context of use.

For each model, UsiXML defines their meta-model. These meta-models are upgraded in a collaborative way by all members. In the next section, we will detail the characteristic of the UsiXML project and UsiXML meta-model.

UsiXML project characteristic

UsiXML project is a large project (Section 2.1), because it involves more then twenty-eight industrial partners (e.g., DefiMedia[2], Thales Research & Technology[3], Telefonica[4]) and academic partners (e.g. University of Namur[5], University of Louvain[6], University of Grenoble[7], University of Valencia[8]) [5].

One of the goals of the UsiXML project is to propose the first stable versions of all the UsiXML meta-models. This requires the all the UsiXML meta-models must be reviewed, updated, and improved by the members. As such, the project requires supporting a medium/fast rate of modifications.

[2] http://www.defimedia.be/

[3] http://www.thalesgroup.com/

[4] http://www.telefonica.com/

[5] http://www.software-engineering.be/

[6] http://www.uclouvain.be/

[7] http://www.ujf-grenoble.fr

[8] http://www.uv.es/

UsiXML Meta-models

UsiXML meta-models are expressed with the UML class diagram notation [14] by using CASE tools that can export these models in a formal standard exchange format (XMI). The Figure 5 shows an example of the meta-model of task expressed in UML.

Besides the UML class diagram notation, each UsiXML meta-model comes with free textual annotations aimed at eliciting the element semantics (for instance, the semantic definition of the task class is: "*task is the basic structure that composes the task model. Tasks are activities that have to be performed to reach a goal*" [13]).

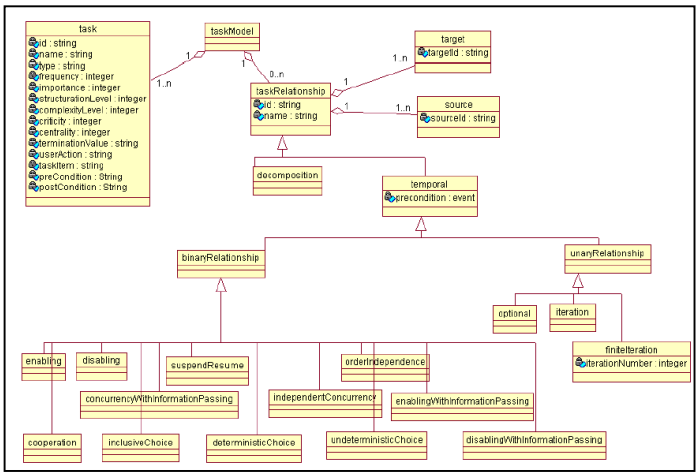

Figure 5: Meta-model of task

DISCUSSION

According to Table 1, the collaborative UsiXML meta-model evolution can be designed using a centralized approach with modification control. This is motivated by the fact that this approach supports a large project. In addition, the UsiXML project requires that a scientific leader is nominated for the control of the meta-model definition. In a centralized approach with modification control, an administrative hierarchy can be implemented by considering the leaders as owners. So that, the divergent versions is avoided by ensuring the reconciliation process is done a priori way. However, such an approach can lead to a high rate of modifications. Moreover, the large size of the UsiXML implies many frequent requests of modifications. This could lead up the owner to be overloaded.

The second approach that can be applied to UsiXML is the decentralized approach with modification control. Indeed, this approach is compliant with projects of large size. In addition, it offers a fast rate of change especially if we consider that the coordinator can be automated [15]. This automated coordinator helps to exchange the journal of modifications between members project. However, the reconciliation process can be complex in comparison with the centralized approach with modification control. This is due to the fact that each member can modify his own local meta-model. This requires a reconciliation process in which

a coordinator controls the different interactions between members. This process can be complex when the number of members is high.

Finally, note that the two approaches without modification control cannot be applicable to UsiXML project because these approaches cannot support a large project.

REFERENCES

1. UsiXML, User Interface eXtensible Markup Language, Available online: http://www.usixml.org, 2010.
2. W3C, XML Schema Part 0: Primer Second Edition, in REC-xmlschema-0-20041028, 2004.
3. OMG, XMI Mapping Specification, v2.1.1, formal/07-12-0, 2007.
4. W3C, OWL Web Ontology Language 2.0, in REC-owl-features-20040210/2004.
5. ITEA2, UsiXML Full Project Proposal, Decembre 23, 2009.
6. Schmidt, K., Simone, C. Coordination Mechanisms: Towards a Conceptual Foundation of CSCW System Design. Computer Supported Cooperative Work. In the Journal of Collaborative Computing 5: pp. 155-200, 1996.
7. Altmanninger,K.,Seidl,M.,Wimmer,M.:Asurveyonmodel versioningapproaches. In IJWIS 5(3) (2009) 271–304
8. Constantin, G., Englebert, V., Thiran, P., A Reconciliation Framework to Support Cooperative Work with DSML, in Proceedings of the First International Workshop on Domain Engineering held in conjunction with CAiSE'09 Conference, collection CEUR-WS.org , volume 457.
9. Mougenot, A., Blanc, X., Gervais, M.P.:D-Praxis : A Peer-to-Peer Collaborative Model Editing Framework. In Distributed Applications and Interoperable Systems 2009, Volume 5523/2009, 16-29 P.
10. Uwe, M., Johann, S. Computer-Supported Cooperative Work, In Book isbn 9783540669845, Oxford University Press, 2000.
11. Grebici, K., Yee, M., Goh, S., Zhao, B., McMahon, C. Information Maturity Approach for the Handling of Uncertainty within a Collaborative Design Team. In IEEE CSCWD 2007.
12. Stanciulescu, A. A Methodology for Developing Multimodal User Interfaces of Information System, Ph.D. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 25 June 2008.
13. UCL, UsiXML V1.8 Reference Manual, February 2007.
14. Object Management Group, Unified Modeling Language 2.0. In formal/2007-02-03, 2007.
15. Saeki, M., Oda, T. A Conceptual Model of Version Control in Method Engineering Environment. CAiSE Short Paper Proceedings 2005.