Towards Personalized Change Propagation for Collaborative Modeling

Mohammadreza Sharbaf*,†, Bahman Zamani*, and Gerson Sunyé†
*MDSE Research Group, University of Isfahan, Isfahan, Iran
†LS2N, University of Nantes, Nantes, France
Email: m.sharbaf@eng.ui.ac.ir, zamani@eng.ui.ac.ir, gerson.sunye@univ-nantes.fr

***Abstract*—Building complex software-intensive systems requires effective collaboration between stakeholders from diverse domains. Model-driven engineering (MDE) aims to reduce the complexity of software systems using separation of concerns and continuous evolution of systems at high-level abstraction. When dealing with complex systems, teams of stakeholders with complementary expertise work concurrently on models to produce a coherent and complete system. In such cases, multi-user modeling environments would help designers create and refine models from multiple points of view. Such environments must be adaptable to the needs and interests of stakeholders. To address this requirement, we propose an approach for personalized change propagation in collaborative multi-view modeling. We capture our ideas in the EMF.cloud environment and demonstrate our approach by introducing a multi-user architecture for multi-view collaborative modeling. We express personalized change propagation as a high-level requirement for a collaborative modeling environment and argue that no existing tool satisfies such requirement. Finally, we discuss research challenges for the proposed approach and outline directions for future research.**

***Index Terms*—Collaborative Modeling, Multi-View Modeling, Personalized Change Propagation, EMF.cloud**

I. Introduction

Over the last decades, the complexity of software-intensive systems has drastically increased. Hence, the development of such systems is no longer an individual task, i.e., it requires an effective team working by developers with different backgrounds [1]. One of the promising approaches for coping with the complexity of software is Model-Driven Engineering (MDE) [2]. In MDE, models are the primary artifacts in software development and are used in different contexts (e.g., code generation, documentation, and verification) to provide automation and hence higher productivity [3]. MDE advocates the use of Domain-Specific Modeling Languages (DSMLs), which abstract the software construction from the underlying implementation technology. The MDE foundation alleviates the growing gap between heterogeneous domains by increasing the level of abstraction and separation of concerns. However, to make the collaboration possible, there is a need for modeling frameworks with multi-user communication support.

In collaborative projects, a team is composed of technical and non-technical stakeholders, together with external collaborators. Having diverse expertise, they work together on models to produce a coherent and complete system [4]. In such projects, stakeholders describe the system at different levels of abstraction from multiple points of view. They may work on distinct views or different parts of the same view. Moreover, multiple stakeholders that work on different aspects of a system have diverse habits, interests, and priorities. Therefore, providing a collaborative multi-view modeling framework that allows stakeholders to select their desired functionalities and personalize their environment according to their own needs is essential [5].

Change propagation, either synchronous (instantaneous) or asynchronous (delayed), is an essential part of collaborative modeling and has a high impact on stakeholders' cooperation. For instance, consider two stakeholders that work concurrently on the same project. If they work on distinct views, they do not need to receive each other's changes immediately. In contrast, if they work on the same view or manipulate interdependent views, immediate change propagation is a key concern. The need for change propagation in collaborative projects contrasts with the lack of support in current collaborative environments. We reviewed 10 existing popular collaborative tools to investigate how the modeling environments address change propagation features. Despite the advances in the MDE world, the result shows that there is no proper support for personalized change propagation in the current collaborative multi-view modeling environments.

In this paper, we introduce a new research roadmap in the collaborative multi-view modeling domain to support personalized change propagation. We first introduce the collaborative modeling features that are fundamental for change propagation. Then we explain how our approach handles each feature. Moreover, we propose a multi-user architecture to support collaborative modeling in the EMF.cloud multi-view modeling environment and express main points to enable support of personalized change propagation. Finally, we outline the challenges and directions that we have identified as significant to address this concern.

The rest of this paper is organized as follows. Section II introduces a motivating example and identifies envisioned idea. Section III presents the overview of our approach for personalized change propagation in collaborative modeling environments. Section IV introduces the proposed architecture for collaborative modeling in EMF.cloud and expresses the personalized change propagation idea for this architecture. Section V examines the support for personalized change propagation in some popular collaborative modeling tools. Finally, Section VI concludes the paper.

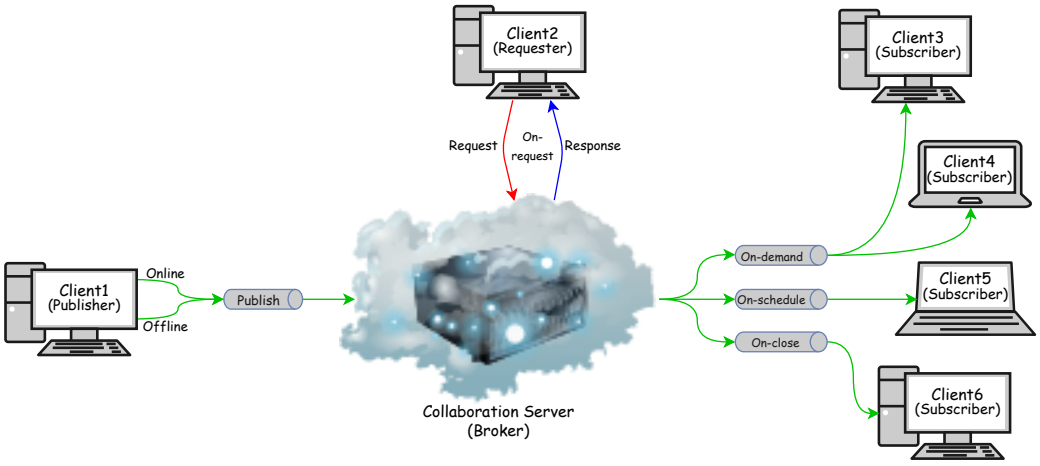

Fig. 1. Overview of Personalized Change Propagation Approach

II. Motivating Example

To motivate our work, we use simple model views of the wind turbine case study [6] that describes the cooling of the *Generator* subsystem. This subsystem includes *input* temperature sensors, *output* fans for cooling the wind turbine generator, and *system parameters* to specify temperature limits for starting the cooling system. The design of control units for cooling the generator relates to the specifications by three stakeholders from different domains, including *WT Manager*, *IO Manager*, and *System Manager*. The *WT Manager* defines all the objects of the model view that specifies the wind turbine. In the model view that relates to *IO Manager*, the attributes of the wind turbine system are obfuscated, and only the input and output objects of the generator system appear. The *System Manager* can only specify system parameters, while she/he has the same view as *IO Manager*. In such cases, all stakeholders do not need to immediately send their changes and receive those applied by others. Therefore, we should allow the stakeholders to personalize the propagation of change operations according to their needs and resources that are common among the stakeholders.

To achieve this, stakeholders must have the authority to select how to receive changes from others. For the *IO Manager*, it is sufficient to be able to see the *System Manager's* changes when the latter has finished his work. But it may be necessary for the *IO Manager* to receive the changes made by the *WT Manager* immediately and on-demand. While the *WT Manager* can become aware of the latest changes that the *IO Manager* applies at a scheduled time. Furthermore, the stakeholders should have the right to request the latest common version of a model view at any time. For instance, the *IO Manager* can request the *System Manager* model even before the end of his/her work. Moreover, a stakeholder can instantly send his/her changes to others or delay sending them until he/she reached a certainty. In our case, *WT Manager* and *IO Manager* decide to send their changes with online and synchronous transactions, but *System Manager* selects the offline and asynchronous submission.

III. Overview of The Approach

In this section, we present an overview of our approach that has been conceived to provide personalized change propagation in a collaborative modeling environment. Personalized change propagation refers to the solutions in which each user is able to customize the fundamental features of change propagation according to his/her needs. In this context, *Collaboration Scenario*, *Collaboration Type*, *Push Method*, and *Undo/Redo* are features that are concerned with propagating change operations when multiple users work on models [7].

The collaboration scenario specifies the ability of the environment to support the change propagation for working on different views. A view may be a projection of a single model while another presents a projection of several models. In such cases, a change on a model must be propagated to all views on that model. Several collaboration scenarios are possible: *Multi-User Single-View* (MUSV), *Multi-View Single-Model* (MVSM), *Multi-View Multi-Model* (MVMM), and *Single-View Multi-Model* (SVMM) [5]. The collaboration type is divided into *Online* and *Offline*. Online collaboration modifies models and propagates change operations to others immediately, while offline collaboration allows publishing change operations at a later time [4]. The push method defines the mechanism to inform the users of change operations, including *Any Modification* and *Bulk Notification*. Any modification is a continuous approach in which users receive all changes in real-time. But in the bulk notification, when a specific condition is reached, users receive a group of changes with an approach, such as *Event-Driven*, *Periodic*, or *Change Threshold*. The Undo/Redo mechansim may be *User-Specific* or *Global*. The user-specific undo/redo simply reverts the last change operation performed by the user on a view. While the global undo/redo considers the operations that are applied by all users [7].

Figure 1 illustrates a snapshot of the proposed approach consisting of a collaborative server and a number of clients, which underpin the development of different modeling artifacts. In order to optimize the change propagation, the collaborative server works as a broker to store and transmit models and their

changes by means of Application Program Interfaces (APIs). Each client who joins the collaboration has a user identity and particular access to modify model views. Clients may adapt one or more of the following roles at the same time:

- *Publisher* in which the client sends changes that are applied in its model view to the collaboration server,
- *Subscriber* in which the client listens to the particular channel to receive changes performed on a model view,
- *Requester* in which the client sends a request and receives the latest version of a model view as the response from the server.

The mentioned roles allow clients to send and receive model view modifications to/from the collaboration server using two patterns: *Publish-Subscribe* and *Request-Response*. Following the *Publish-Subscribe* pattern, the collaboration server does not broadcast every change operation to other clients. The server enables the definition of particular channels to personalize sending and receiving modifications for each client, given the change propagation features. To this end, the client (*publisher*) can select a check-in method for sending its modifications to the collaboration server. According to the *collaboration type*, we consider the following check-in methods:

- *Online check-in* that sends each change operation to the server immediately,
- *Offline check-in* that sends a group of change operations to the server at a later time.

In the collaboration server, the change propagation is considered as updating and synchronizing resources. Each resource can be a single model or a distinct view that the server admin can specify according to the selected *collaboration scenario* for supporting multi-view collaboration. When the server receives new change operations for a resource, it first updates the global instance of the resource. Then it forwards all change operations to the check-out channels that are specified for the resource. Clients should subscribe to one check-out channel for receiving changes that are publishing in a resource. We propose three check-out channels for each resource based on the *push methods* as follows:

- *On-demand* in which subscribers immediately receive any change operation of the resource when they are connected to the collaboration server. This method is similar to the *any modification* approach of the push method.
- *On-schedule* in which subscribers receive existing change operations of the resouce in a *Specific Time* (e.g., at 2:00 A.M of every day), in a *Period* (e.g., every 10 minutes), or based on a *Numerical Threshold* (e.g. for every 10 changes). We can map the *On-schedule* check-out method to the *periodic* and *change threshold* approaches of the push method.
- *On-close* in which subscribers receive all change operations that are submitted to the server by users that are working on the resource when all users close the resource. This method is a severe case that propagates changes on specific conditions, which is concerned with *Event-Driven* approach of the push method.

We also allow clients (*requester*) to receive a resource from the collaboration server following *Request-Response* pattern. Using this method that we called *On-request*, clients can receive the latest version of a resource and its *position* from the server when requesting it. The position is a number, which specifies the last change operation that the collaboration server applied to the resource. Moreover, since the local and global lists of the propagated change operations are present, both alternatives for the Undo/Redo feature can be handled in our approach. In the simplest case, we only must apply the reverse of the last change operation in the local list of a client for *user-specific* undo and in the global list of the collaboration server for the *global* undo.

IV. Collaborative Modeling in EMF.cloud

In this section, we demonstrate how the personalized change propagation approach can empower the EMF.cloud modeling environment to support multi-view collaborative modeling. EMF.cloud is a generic, open-source, and application-independent framework for editing, processing, and managing EMF-based models in a browser-based client. EMF.cloud with a client-server scenario is focused on enabling the use of many existing EMF-based technologies in a cloud context. The model server is a central component of EMF.cloud that is responsible for storing and updating model instances that are hosted on its local workspaces. The model server provides command-based editing and undo/redo support for the client and allows multiple views on the same model instance based on the Language Server Protocol (LSP). However, the current implementation of EMF.cloud does not support multi-user and collaborative modeling. In the following, we first propose a collaborative architecture for multi-user modeling in EMF.cloud. Then, we explain the personalized change propagation in the EMF.cloud multi-user architecture.

A. Multi-User Modeling Architecture

We propose a collaborative modeling architecture that uses a client-server approach to support the communication between multiple users. Each user represents an instance of EMF.cloud that consists of a browser-based *Client* and a local *Model Server*. Figure 2 shows an overview of the proposed architecture, where multiple instances of EMF.cloud are connected exactly to one *Collaboration Server*. The central collaboration server is used to coordinate the change propagation among all connected users following an advanced RESTful API protocol that uses the JSON serialization to transfer model data. We briefly review the main attributes of each component in the following.

1) Collaboration Server: The collaboration server is responsible for transmitting change operations among users and storing the global version for each resource. To this end, the collaboration server consists of an *Incoming Manager* for receiving changes and requests from users, a *Synchronizer* to

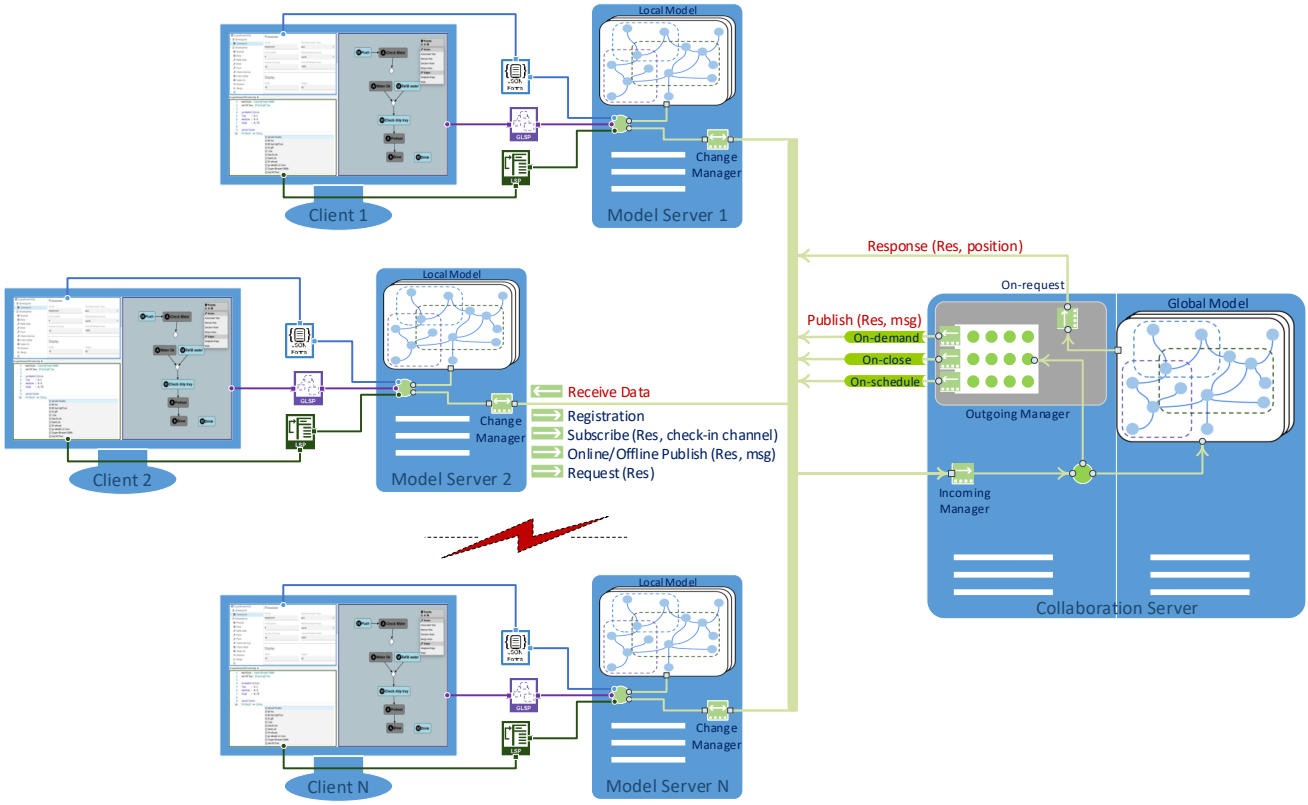

Fig. 2. Overview of the Proposed Architecture for Multi-user Modeling with Emf.Cloud

update resources and related change queues, and an *Outgoing Manager* for sending resources and changes to the users.

2) Model Server: The model server is responsible to support loading views, manipulating them, and propagating changes. We extended the original version of the EMF.cloud model server by adding a *Change Manager* for managing communication with the collaboration server.

3) Client: The client component is mainly the same as the client of the original EMF.cloud. We only added related forms (e.g., Registration form) and *Resource Change Awareness* for specifying incoming changes.

B. Personalized Change Propagation

Supporting both *Publish-Subscribe* and *Request-Response* communications are critical to perform the personalized change propagation in multi-user EMF.cloud architecture. To achieve this, the following prerequisites must be considered.

- *User registration* and *unique identity* are required to authenticate users
- Users should receive a *list of resources* when connect to the collaboration server
- Each *resource* should be identified by a *unique identity*
- The collaboration server must *create global copy* for each resource
- Each *change operation* has a *publisher* and modifies a *resource* that should be specified for the *check-in*
- The collaboration server must *create check-out channels* and *notify users* for new resources
- Each user can *subscribe* to only one *check-out* method for each resource
- Each *model server* must have a *stack* of changes for user-specific undo/redo
- The collaboration server must have a *stack* of changes for each resource to support global undo/redo

V. Discussion

In the last decade, several modeling environments are proposed to support multi-user collaboration. Each environment uses a specific mechanism to send and receive change operations between users. However, there is no a single tool that allows users to personalize change propagation. We selected important features of change propagation in collaborative modeling environments based on the feature model presented by Masson et al. [7]. Table I summarizes the support for features of change propagation in 10 existing popular environments in the last decade that could be used for collaborative modeling in academia or industry. Note that we only investigated features that the environment addresses, and those provided by external extension or user's effort are not the focus of this paper.

Table I shows that most of popular collaborative modeling environments focus on *Multi-User Single-View* scenario. Only one environment provides both *Online* and *Offline* collaboration. In most of the environments, undo/redo is provided

TABLE I
SUPPORT FOR CHANGE PROPAGATION FEATURES IN POPULAR ENVIRONMENTS FOR COLLABORATIVE MODELING

Tool	Collaboraion Scenario	Collaboration Type	Undo/Redo	Push Method
CDO [8]	MUSV	Online, Offline	Not Supported	Any Modification, Event-Driven
OBEO [9]	MUSV, MVSM	Offline	User-Specific	Event-Driven
AToMPM [10]	MUSV	Online	User-Specific	Any Modification
WebGME [11]	MUSV	Online	Not Supported	Any Modification
MetaEdit+ [12]	MVMM	Offline	User-Specific	Event-Driven
EMFStore [13]	MUSV	Offline	User-Specific	Event-Driven
EMFCollab [14]	MUSV	Online	User-Specific	Any Modification
MDEForge [15]	MUSV	Offline	Not Supported	Event-Driven
GenMyModel [16]	MUSV, SVMM	Online	User-Specific, Global	Any Modification
Visual Paradigm [17]	MUSV	Offline	User-Specific	Event-Driven

in *User-Specific* level, while one environment supports both levels, and three environments do not support it. Moreover, *Any Modification* and *Event-Driven* are only two mechanisms that are used for the push method. Given the limited capabilities in the existing modeling environments that have not allowed experts of different domains to customize propagating changes in their collaboration, we propose research on collaborative modeling techniques to support personalized change propagation. The following challenges and directions should be addressed:

- ***Multi-View Branching:*** Users with diverse domains may work on views of the same or different models. We feel an environment should concern view branching and work on older versions.
- ***Conflict Management:*** Several approaches are proposed for conflict detection and resolution, but this seems intended only for a single view, and internal relationships among views are not considered for that functionality.
- ***Tracability:*** Tracing change dependencies helps understand their impact on different views. But no tool is identified that allows tracking those relationships.

VI. CONCLUSION

Building complex software-intensive systems is no longer an individual task; it is a collaborative endeavor between stakeholders from diverse domains. Different stakeholders coordinate with each other and share their modifications to achieve their project goals. Customizable tools are increasingly demanded in today's software industry. We have reviewed ten popular modeling tools, but no tool allows stakeholders to customize its environment based on their needs. In this article, we introduced personalized change propagation as a new research roadmap in the area of collaborative multi-view modeling. We provided a general solution to support personalized change propagation in collaborative modeling environments. Moreover, we contributed an architecture for multi-user modeling with EMF.cloud. The proposed architecture argues the prerequisites to develop a customizable environment for change propagation. We also outline some significant challenges and future directions that must be addressed in this regard.

REFERENCES

[1] J. Whitehead, "Collaboration in software engineering: A roadmap," in Proceedings of the Future of Software Engineering, 2007, pp. 214–225.

[2] M. Brambilla, J. Cabot, and M. Wimmer, Model-driven software engineering in practice, 2nd ed. Morgan & Claypool Publishers, 2017.

[3] J. Whittle, J. Hutchinson, and M. Rouncefield, "The state of practice in model-driven engineering," IEEE software, vol. 31, pp. 79–85, 2013.

[4] M. Franzago, D. Di Ruscio, I. Malavolta, and H. Muccini, "Collaborative model-driven software engineering: a classification framework and a research map," IEEE Transactions on Software Engineering, vol. 44, pp. 1146–1175, 2017.

[5] J. Corley, E. Syriani, H. Ergin, and S. Van Mierlo, "Cloud-based multi-view modeling environments," in Modern Software Engineering Methodologies for Mobile and Cloud Environments, 2016, pp. 120–139.

[6] A. Bagnato, E. Brosse, A. Sadovykh, P. Maló, S. Trujillo, X. Mendialdua, and X. De Carlos, "Flexible and Scalable Modelling in the MONDO Project: Industrial Case Studies," in Proceedings of Workshop on Extreme Modeling @MoDELS, 2014, pp. 42–51.

[7] C. Masson, J. Corley, and E. Syriani, "Feature Model for Collaborative Modeling Environments," in Proceedings of Workshop on Collaborative Modelling in MDE at MODELS, 2017, pp. 164–173.

[8] E. Stepper, "CDO Model Repository Overview," (Last accessed: 2021 July). https://www.eclipse.org/cdo.

[9] Obeo Designer, "Collaborate with Your Team," (Last accessed: 2021 July). https://www.obeodesigner.com/collaborative-features.

[10] E. Syriani, H. Vangheluwe, R. Mannadiar, C. Hansen, S. Van Mierlo, and H. Ergin, "AToMPM: A Web-based Modeling Environment," in MoDELS 2013 Demonstration, 2013, pp. 21–25.

[11] M. Maróti, T. Kecskés, R. Kereskényi, B. Broll, P. Völgyesi, L. Jurácz, T. Levendovszky, and Á. Lédeczi, "Next generation (meta) modeling: web-and cloud-based collaborative tool infrastructure," in Proceedings of Workshop on Multi-Paradigm Modeling @MoDELS, 2014, pp. 41–60.

[12] S. Kelly, "Collaborative modelling with version control," in Proceedings of the Federation of International Conferences on Software Technologies: Applications and Foundations, 2017, pp. 20–29.

[13] J. Helming and M. Koegel, "EMFStore Project," (Last accessed: 2021 July). http://eclipse.org/emfstore.

[14] EmfCollab, "Collaborative Editing for EMF models," (Last accessed: 2021 July). http://qgears.com/products/emfcollab.

[15] J. Di Rocco, D. Di Ruscio, A. Pierantonio, J.S. Cuadrado, J. De Lara, and E. Guerra, "Using ATL transformation services in the MDEForge collaborative modeling platform," in International Conference on Theory and Practice of Model Transformations, 2016, pp. 70–78.

[16] M. Dirix, A. Michel, and V. Aranega, "Genmymodel: an online uml case tool," in ECOOP. 2013.

[17] Visual Paradigm, "Team Collaboration Toolset," (Last accessed: 2021 July). https://www.visual-paradigm.com.