Collaborative Modeling and Visualization of Software Systems Using Multidimensional UML

Matej Ferenc, Ivan Polasek and Juraj Vincur
Faculty of Informatics and Information Technologies
Institute of Informatics and Software Engineering,
Slovak University of Technology in Bratislava
Ilkovičova 2, SK-84216 Bratislava 4
{xferencm, ivan.polasek, juraj.vincur}@stuba.sk

***Abstract*— This paper introduces our approach to a real-time synchronous collaborative modeling of software systems using 3D UML in a way similar to shared *Google Document* online with the aim to reduce the complexity of UML models and to increase work efficiency. In our approach, we decided to visualize the system with 2D UML diagrams on interconnected layers containing components (in class diagrams) or use case scenarios of the system (in sequence or activity diagrams) in 3D space. The goal of our method is to improve user's awareness of other developers in a multi-user workspace, adjust redundant components and visualize the history of user's actions in the UML class diagrams.**

***Index Terms*—3D UML, real-time synchronous collaboration, modeling, visualization, evolution**

I. Introduction

Developing complex and large-scale software systems is a difficult and complicated process in which many people are involved. Multiple experts with various specializations need to collaborate concurrently in order to analyze and design the system. Therefore, the main motivation of this paper is to introduce and propose our approach (and coupled tool) which supports:

- working on a model of the software system similarly to draw.io (https://www.draw.io) or *GenMyModel* (http://www.genmymodel.com) online, increasing the productivity and work efficiency in a collaborative and parallel modeling compared to the standard offline and uninformed non-collaborative work,
- visualizing parallel layers (with parallel versions or with the other collaborating components or with particular structures, supporting other use cases) on which other developers are working (collaborators can work on the same layer too),
- visualizing similar parts on the other component layers to not reinvent the wheel and reduce the vague and redundant elements,
- reducing of the complexity of exact and large UML models using the layers decomposing the extensive model to the real components,
- visualizing evolution in the model and collaboration, controlled with timeline widget as the gource system [10] visualizing source code evolution.

II. Related Work

Ellis, Gibbs, and Rein [1], describe collaboration using the 3C Collaboration Model which defines collaboration as an interlay between coordination, communication and cooperation. Fuks et. al. [2] later adopt this model and state that awareness mediates and fosters all three aspects of collaboration. This model can be used as a base for analyzing and designing groupware. Dourish and Belloti [3] define awareness as "an understanding of the activities of others, which provides a context for one's own activities." Many papers discuss the importance of user's awareness of other's in a shared workspace [4][15][16]. Gutwin and Greenberg [4], state that every collaborator should be intuitively aware of present related aspects such as who is in the workspace, where they are located, what they are working on, as well as past related aspects such as how this artefact came to be in this state or who made this change and when. They state that good awareness provides the following benefits:

- collaborators will not miss a chance to collaborate or oppositely, will not interrupt others at inappropriate time,
- collaborator has a better contextual understanding of where assistance is required,
- unnecessary need for communication is eliminated,
- collaborator can predict the others' actions and therefore make an easier decision on choosing their next task,
- work redundancy is eliminated and division of labor is simplified.

Based on the above, we can state that good awareness of others and their activities increase work efficiency and productivity in multi-user workspace. However, this does not solve the problem of readability of complex and large-scale UML models of software systems. Many research papers propose that modeling and visualizing a system in 3D space can eliminate this problem and introduce many improvements. The idea of system modeling in three-dimensional space was first published in 1991 as a Doctoral Dissertation by Koike [5]. He proposes that the increase in dimension enables to visualize large number of objects and relations among them. Since then many other papers have been published covering this topic. For example, Casey et al. [6], propose an approach to visualizing

UML class diagrams as geon diagrams. He states that it is easier for users to remember 3D geometrical shapes than text. Another example is Dwyer's [7] 3D UML visualization of a class diagram, where he used force-directed algorithm to layout UML class diagram in 3D space. In his visualization, he represented standard 2D UML classes as 3D blocks, 2D relationships as 3D connectors and enclosed UML classes within the same UML package inside a sphere.

Another approach to 3D system visualization is by placing standard 2D UML diagrams on multiple layers in 3D space. Von Pilgrim and Duske [8] provide an example of this approach in their research paper and present a 3D framework called Gef3D. The framework enables to transform any existing GEF-based 2D editors into 3D editors and enables to visualize connections between 2D diagrams on layers in 3D space. This topic is also being researched in our institute with the aim to reduce complexity of UML models and to propose other improvements of UML and use case modeling and visualization [13][14]. For example, in their paper, Gregorovic and Polasek [9][12] present an approach for automatic generation of object and class UML diagrams from sequence UML diagrams. In addition, they introduce automatic layout of UML class diagrams in 3D space based on the class semantics.

Based on the previous examples and especially with the rise of virtual and augmented reality, modeling systems in 3D space seems to be the trend of how systems will be modelled in the future.

III. Our Approach

In our approach, we decided to continue visualizing the system with 2D UML diagrams on interconnected layers (close to the concept of 2.5D and 3D UML [8]) containing components (in class diagrams) or use case scenarios (in sequence or activity diagrams) of a system in 3D space. We aimed to improve the process of UML modeling with real-time synchronous collaboration and we have designed and implemented a real-time collaborative 3D UML application with many awareness features (Fig. 1).

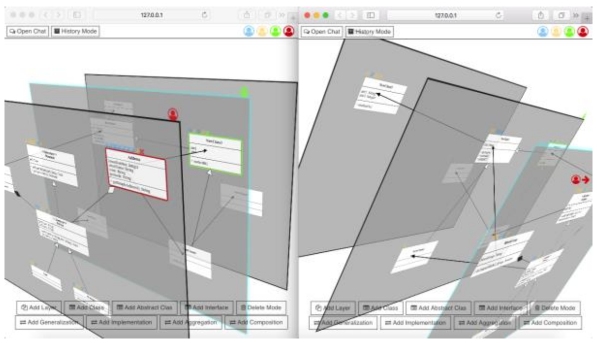


Fig. 1. Collaborative 3D UML Application

We have also implemented an Enterprise Architect Add-in (Fig. 2), to enable integration with EA and to enhance EA with real-time synchronous collaboration.

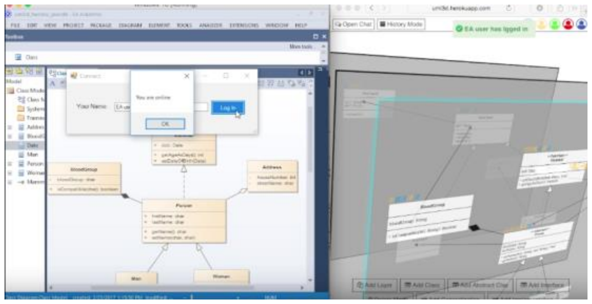


Fig. 2. Enterprise Architect Add-in

Figure 3 shows the architecture of our proposed method for collaborative 3D UML modeling. Following, the architecture's components are briefly described.

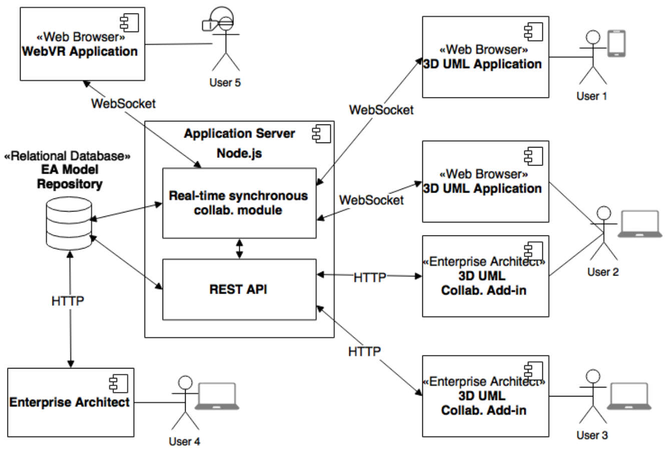


Fig. 3. Collaborative 3D UML Application

Relational Database – dedicated server based repository for storing shared system models as well as other relevant data. EA's existing physical data model was used for storing the UML models. The models are therefore standardized and could be easily imported or shared.

Application Server – centralized server for managing communication and synchronization among multiple clients. Where fast real-time synchronous communication is required, WebSockets are used and REST API was used for communication between EA and the application server.

3D UML Application – our main tool used for real-time synchronous collaborative system modeling and visualization using 3D UML. In this tool, multiple users should be able to collaborate in real-time, therefore creating UI and features with high awareness factor were our main goal.

Enterprise Architect Add-in – integrates EA with our 3D UML application and enhances EA with real-time collaborative features. This also enables users to utilize EA's built-in features for complex system modeling that they have already adopted.

Our 3D UML application enables users to create UML class diagrams in 3D space collaboratively. All of the awareness features are implemented as real-time synchronous functionality. Therefore, any change made by one collaborator can be instantly seen by all other users. For example, if one collaborator is moving a UML element, all other collaborators can see who is moving the element in real time. An example of this can be seen on the following sequence diagram (Fig. 4).

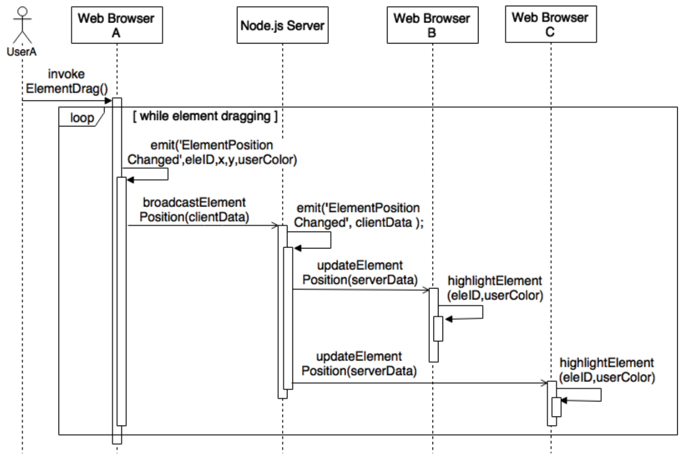

UserA
Web Browser A
Node.js Server
Web Browser B
Web Browser C
invoke ElementDrag()
loop
[while element dragging]
emit('ElementPosition Changed',eleID,x,y,userColor)
broadcastElement Position(clientData)
emit('ElementPosition Changed', clientData);
updateElement Position(serverData)
highlightElement (eleID,userColor)
updateElement Position(serverData)
highlightElement (eleID,userColor)

Fig. 4. Real-time Synchronous Element Dragging Functionality

The following is a summary of all implemented awareness features. The first implemented awareness feature is the login notification. Consequently, after an existing or a new user logs in, a notification is broadcasted to all other collaborators and they are instantly notified about who joined the workspace In multi-user workspace it is important to let others know if a collaborator is online and prepared to work. (Fig. 5).

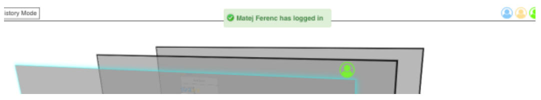


Fig. 5. Login Notification

Presence Awareness Features enable a user to instantly understand who is in the workspace, where others are working and on what objects they are currently working. There are four presence awareness features that were implemented and can be seen in the figure 6. The collaborators are always aware of all project participants (Fig. 6 - 1), on what layer they are working (Fig. 6 - 2), on what specific element they are working (Fig. 6 - 3), end even of the actions of others if they are working outside of their view (Fig. 6 - 4).

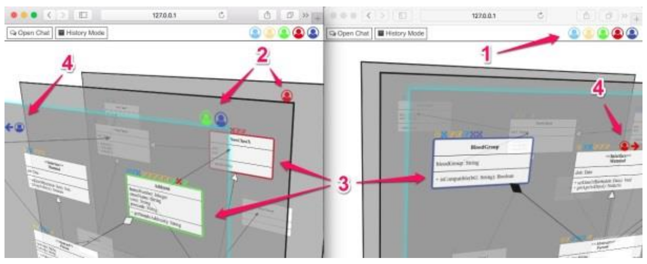


Fig. 6. Presence Awareness Features

In a multi-user workspace, it is also very important to understand not only where others are working in the present, but also to understand the history of their actions. We present the following feature as one possible approach to visualise the history of user's actions. A small flag is placed beside the element that has been changed. The flag has the same color as the user who modified it and fades out as it is further in history. This enables the collaborators to see five most recent actions of other collaborators before the flag disappears. Furthermore, the collaborators can see what exactly has been changed by moving the mouse over a flag (Fig. 7).

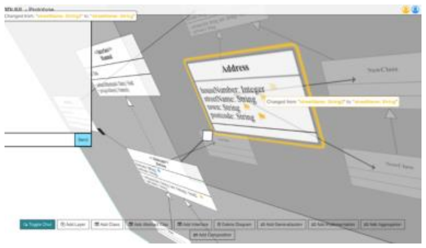


Fig. 7. User Action History

However, this feature later proved not to be as effective, since it was very complicated to see the fading of the small flags if they were distributed around the whole project. Therefore, we proposed another approach how the history of user's actions can be visualised simultaneously with the history of each UML class element. In this approach, a different icon is placed on top of a UML class element for each type of change. This enables a collaborator to find the change he was looking for more efficiently. The icons are also the same color as the user who made the change, which identifies the user. A collaborator can also immediately visualise details about the change, by placing his cursor above the element. For example, the figure 8 (left) shows a user viewing the most recent change of a UML class. He can immediately see that the change has been made by the green user and that it was made 2 minutes ago. We have also provided a visual assistance to quickly find what exactly has been changed. We stroked out and used red color to visualise what has been removed and used green color to visualize what has been added. Similarly, the figure 8 (right) shows how a user can see a different type of change. In this example, a user can see who has moved a UML class and when it was moved. The red and green colors were also used to illustrate the old and new positon of the UML class.

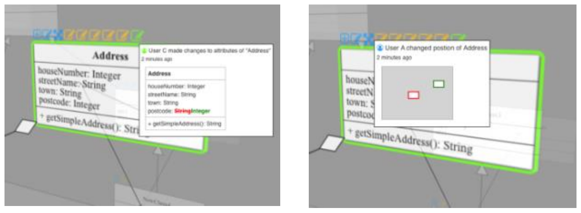
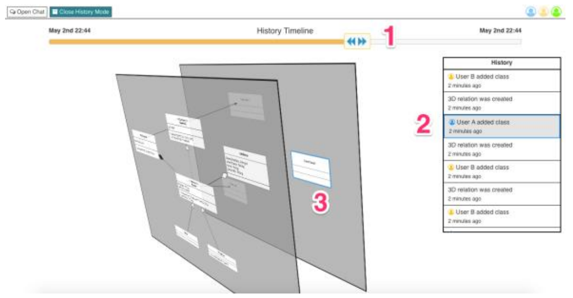

Address
houseNumber: Integer
streetName: String
town: String
postcode: Integer
+ getSimpleAddress(): String
User C made changes to attributes of "Address"
2 minutes ago
User A changed position of Address
2 minutes ago

Fig. 8. UML Class and User Action History

The previous feature provides visual elements to show the most recent changes of one UML class element only. The next feature enables a user to see the history of all collaborators' actions and simultaneously visualise the state of the whole project in a specific time in history. We called the next feature the project history timeline. A collaborator can simply visualise the entire project from the initial state to the last collaborator's contribution. A collaborator can navigate back and forth in history by moving the history timeline slider (Fig. 9 - 1). Each step of the slider represents one user's action in history. The collaborator can see more details about the action in the history window (Fig. 9 - 2). He can see who has made the change and when the change was made. The actual change is highlighted directly in the UML diagram (Fig. 9 - 3).



User B added class
2 minutes ago
3D relation was created
2 minutes ago
User A added class
2 minutes ago
3D relation was created
2 minutes ago
User B added class
2 minutes ago
3D relation was created
2 minutes ago
User B added class
2 minutes ago

Fig. 9. Project History Timeline (History Mode)

Figure 10 shows an example of this functionality. As the user moves the timeline slider from the initial user's contribution towards the last project modification, the collaborators' actions are being executed and the project is dynamically growing. New changes, performed by others, are also being added in real-time.

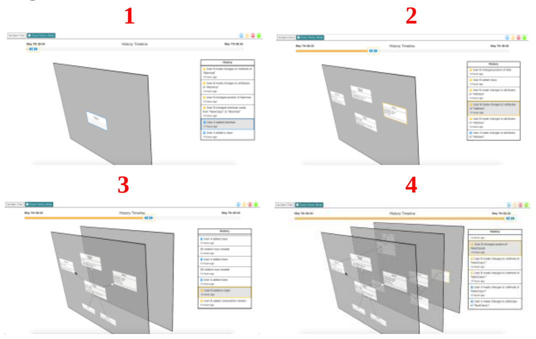
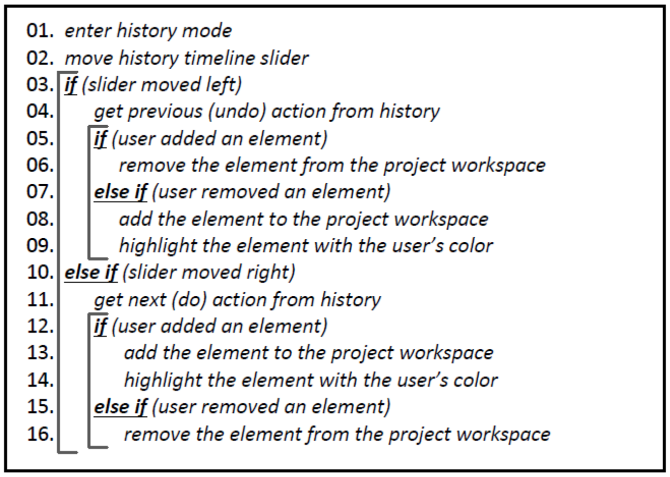


Fig. 10. Project History Timeline Functionality Example

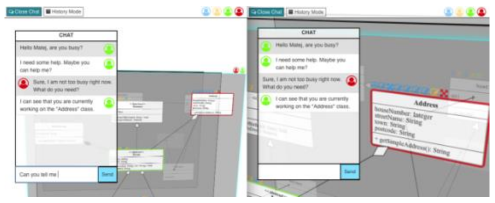
We have implemented this functionality by logging and creating an undo action for every action a user made (an action is any CRUD operation, such as adding a new layer, adding a relation between two elements, adding, removing or updating element's name, attribute or method). Therefore, each action saved in history is composed of a "do" and "undo" action. If the slider is being moved forward the "do" actions are executed and if the slider is being moved backwards the "undo" actions are executed. These "undo" actions are created on the server and then broadcasted to all connected clients with every standard user's action. Therefore, every client always has a local and up to date copy of the entire history of the project. In the future we can visualise developer interaction or use this information to predict bad smells in the model. The local execution of selected history actions (i.e., add, remove) can be seen in the figure 11 (moreover, modifications are also supported similarly).



```
enter history mode
move history timeline slider
if (slider moved left)
    get previous (undo) action from history
    if (user added an element)
        remove the element from the project workspace
    else if (user removed an element)
        add the element to the project workspace
        highlight the element with the user's color
else if (slider moved right)
    get next (do) action from history
    if (user added an element)
        add the element to the project workspace
        highlight the element with the user's color
    else if (user removed an element)
        remove the element from the project workspace
```

Fig. 11. Execution of History Actions (pseudocode)

One of the benefits of above features is that they minimize the need for communication. However, there are situations when a collaborator needs to ask for assistance or quickly inform others about something. In these situations, a simple chat is an efficient solution. Figure 12 shows our implementation of chat and an example of a chat communication between two collaborators. Other forms of communication were also considered, such as the exchange of comments on a specific UML element, as it is possible in EA or audio/video chat using WebRTC. These features can be implemented in the future, as they enable other benefits in communication.



Hello Matej, are you busy?
I need some help. Maybe you can help me?
Sure, I am not too busy right now. What do you need?
I can see that you are currently working on the "Address" class.
Can you tell me

Fig. 12. Communication Example

By default, the chat window is not visible. This is due to the fact, that chat is not a primary collaborative feature and it can also cover a lot of space of the working area. However, we have also intentionally made the chat window slightly transparent, thus enabling the collaborator to be aware of any actions hidden by the chat window. The user can also move the chat to any location on screen. If the chat is closed and a new message is received, a user is notified by a sound alert and also a small icon with the number of new messages appears beside the "Open chat" button (Figure 12).

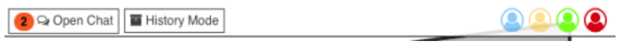


Fig. 13. New Message Notification

IV. Open Questions and Hypotheses for Future Work and Evaluation

We have proposed a method and tool prototype for collaborative 3D system modeling. The real-time synchronous collaboration enables collaborators to work on one centralized model in real-time. This eliminates the need for sharing or merging of multiple versions of UML models. We have proposed various visual artefacts and features which improve the user's present and past awareness of others and their actions in a multi-user workspace. These aspects are used as conversational artefacts or visual evidence to replace or complete possible verbal communication and therefore minimize the need for communication. In addition, by always knowing what others are currently working on, a collaborator can make a faster decision on choosing his next step based on his prediction or expectation of what others will do next. This also eliminates redundant or duplicate work. The question if these features could help to provide faster and more efficient system modeling has to be answered and evaluated. We have the opportunity to do it in a new industrial insurance software project with research background proposed by the software company Gratex International (gratex.com or gratexinsurance.com).

We can use this method in similar areas (not only for UML models, but also for ontologies and domain specific language models, etc.) or as a practice for modeling of specific industrial standards as a *pair modeling*: trainer and novice in parallel layers.

It is possible to run the collaborative 3D UML application simply by navigating to http://uml3d.herokuapp.com and running the prototype in multiple web browsers. A video, which describes the functionality of the collaborative 3D UML application, can be accessed on https://youtu.be/ehx6HI8B_fQ.

V. Acknowledgment

This work was supported by the Scientific Grant Agency of Slovak Republic (VEGA) under the grant No. VG 1/0646/15 and No. VG 1/0752/14. This contribution was created with the support of the Ministry of Education, Science, Research and Sport of the Slovak Republic within the Research and Development Operational Programme for the project "University Science Park of STU Bratislava", ITMS 26240220084, co-funded by the ERDF.

References

[1] A. C. Ellis, J. S. Gibbs and G. Rein, "Groupware: some issues and experiences," in Communications of the ACM, 1991, pp. 39-58.

[2] H. Fuks et al., "The 3c collaboration model," in The Encyclopedia of e-collaboration. Ned Kock (org), 2007, pp. 637-644.

[3] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in Proceedings of the 1992 ACM conference on Computer-supported cooperative work. New York: ACM, 1992, pp. 107-114.

[4] C. Gutwin and S. Greenberg, "A descriptive framework of workspace awareness for real-time groupware," in Computer Supported Cooperative Work (CSCW), 2002, 11.3-4, pp. 411-446.

[5] H. Koike, "Three-dimensional software visualization: a framework and its applications," in Visual Computing. Springer, 1992, pp. 151-170.

[6] K. Casey and C. Exton, "A Java 3D implementation of a geon based visualisation tool for UML," in Proceedings of the 2nd international conference on principles and practice of programming in Java. Computer Science Press, 2003, pp. 63-65.

[7] T. Dwyer, "Three dimensional UML using force directed layout," in Proceedings of the 2001 Asia-Pacific symposium on information visualisation, vol. 9. Australian Computer Society, 2001, pp. 77-85.

[8] J. von Pilgrim and K. Duske, "Gef3D: a framework for two-, two-and-a-half-, and three-dimensional graphical editors," in Proceedings of the 4th ACM symposium on software visualization. New York: ACM, 2008, pp. 95-104.

[9] L. Gregorovic, I. Polasek and B. Sobota, "Software model creation with multidimensional UML," in Confenis, WCC 2015, Daejeon, South Korea, LNCS 9357. Springer, 2015, pp. 343-352.

[10] A. Caudwell, "Gource: visualizing software version control history," in OOPSLA '10. New York: ACM, 2010, pp. 73-74.

[11] R. Minelli et al., "Visualizing Developer Interactions," in VISSOFT 2014, IEEE working conference on software visualization. IEEE, 2014, pp. 147-156.

[12] L. Gregorovic and I. Polasek, "Analysis and design of object-oriented software using multidimensional UML," in International Conference on knowledge technologies and data-driven business, I-KNOW '15, Graz, Austria. New York: ACM, 2015, article no. 47.

[13] M. Bystrický and V. Vranic, "Preserving use case flows in source code: approach, context, and challenges," in Computer science and information systems journal, vol. 14, no. 2, 2017, pp. 423–445.

[14] M. Bystricky and V. Vranic, "Development Environment for Literal Inter-Language Use Case Driven Modularization," in Modularity Companion 2016, Companion Proceedings of the 15th International Conference on Modularity, Demos & Posters, March 2016, Málaga, Spain. New York: ACM, 2016, pp. 12-15.

[15] S. Liu et al., "Real-time Collaborative Software Modeling Using UML with Rational Software Architect," in Proceedings of IEEE international conference on collaborative computing: networking, applications and worksharing. IEEE, 2006, pp. 1-9.

[16] M. Arciniegas-Mendez, A. Zagalsky, M. Storey and A. F. Hadwin, "Using the Model of Regulation to Understand Software Development Collaboration Practices and Tool Support," in Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17). New York: ACM, 2017, pp. 1049-1065.