SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
Faculty of Informatics and Information Technologies

Bc. Matej Ferenc

Method for Collaborative Modelling and Visualisation of Software Systems Using Multidimensional UML

Master's Thesis

FIIT-5220-46204

Degree Course
Software Engineering

Field of Study
9.2.5 Software Engineering

Study Department
Institute of Informatics, Information Systems and Software Engineering. FIIT STU

Supervisor
Ing. Ivan Polášek, PhD.

2017, May

Declaration of Originality

With my signature I confirm that:

This thesis is my own work and I have documented all sources of information, including figures and tables which were not created or written by me.
I have committed none of the forms of plagiarism.
I have implemented all the prototypes by myself, using my knowledge and technical skills.

…….

Bc. Matej Ferenc

Acknowledgement

I would first like to give my greatest appreciation and gratitude to my advisor Ing. Ivan Polasek, PhD. I would like to thank him for having his office doors always open for me, for always motivating me and mainly for all his helpful advice and ideas. Next, I would like to give special thanks to my sister for taking her time to read this thesis over, for correcting grammar mistakes and for making this thesis more readable. Lastly, I would like to thank my family and friends for their love and support.

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Autor:	Bc. Matej Ferenc
Študijný program:	Softvérové inžinierstvo
Diplomová práca:	Metóda pre kolaboratívne modelovanie a vizualizáciu softvérového systému pomocou viacrozmerného UML
Vedenie diplomového projektu:	Ing. Ivan Polášek, PhD.
Dátum:	máj, 2017

Cieľom diplomovej práce je navrhnúť nový a inovatívny prístup, ktorý by zjednodušil komplexnosť UML modelov a zároveň zvýšil produktivitu a efektívnosť práce pri kolaboratívnom modelovaní systému. Práca sa dotýka všetkých aspektov týkajúcich sa kolaborácie vrátane uvedomenia si prítomnosti v softvéri určenom na spoluprácu, rôznych druhov kolaborácie ako aj spôsobov riešenia konfliktov. Okrem vyššie uvedeného sa práca ďalej zaoberá rôznymi spôsobmi vizualizácie systémov pomocou 3D UML a vymenúva niektoré obmedzenia modelovania pomocou 3D UML s cieľom ich eliminácie. Na základe analýzy je navrhnuté používateľské rozhranie spolu s rôznymi vizuálnymi prvkami a funkciami, ktoré zvyšujú uvedomenie si prítomnosti. Práca taktiež navrhuje rôzne kritéria využitia force-directed algoritmov za účelom automatického rozmiestnenia UML diagramov s cieľom zvýšiť uvedomenie si prítomnosti. Následne je vytvorený návrh architektúry systému a sú implementované tri prototypy. Prvým je webová aplikácia, ktorá umožňuje synchrónnu kolaboráciu modelovania systému v reálnom čase pomocou 3D UML a prináša nové prvky uvedomenia si prítomnosti. Druhým implementovaným prototypom je Enterprise Architect Add-in, ktorý umožňuje integráciu medzi EA a implementovanou webovou aplikáciou a ktorý dopĺňa EA o synchrónnu kolaboráciu v reálnom čase. Posledným prototypom je WebVR aplikácia, ktorá bola implementovaná za účelom experimentovania s modelovaním pomocou UML vo virtuálnej realite.

ANNOTATION

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Author:	Bc. Matej Ferenc
Degree Course:	Software Engineering
Master's Thesis:	Method for Collaborative Modelling and Visualisation of Software Systems Using Multidimensional UML
Supervisor:	Ing. Ivan Polášek, PhD.
Date:	2017, May

The aim of this Master's thesis is to introduce and propose a new and innovative approach which would reduce the complexity of UML models and increase the productivity and work efficiency in collaborative system modeling. The thesis covers all aspects related to collaboration, such as a user's awareness of others and their actions, different types of collaboration or conflict resolution techniques. Moreover, various 3D UML visualisation methods are analyzed and some limitations of 3D UML are listed with the goal of their elimination. Based on the analysis, a UI is designed containing various visual elements and features which improve the user's present and past awareness of others in a multi-user workspace. The thesis also proposes various approaches to how weighted force-directed algorithms could be applied to automatically layout UML diagrams to improve awareness in 3D multi-user workspace. Furthermore, system architecture is designed and three prototypes are implemented. First one is a 3D UML web application for real-time synchronous collaborative 3D UML modeling. Second prototype is an Enterprise Architect Add-in which enables integration with EA as well as enhances EA with real-time synchronous collaboration. The last prototype, a WebVR application, was implemented to experiment with collaborative UML modeling in virtual reality.

Table of Contents

List of Figures

Terms and Abbreviations

UML – Unified Modeling Language
XMI – XML Metadata Interchange
OMG – Object Management Group
OT – Operational Transformation
2D – Two-dimensional
3D – Three-dimensional
HTML – Hypertext Markup Language
CSS – Hypertext
JS – JavaScript
JSON – JavaScript Object Notation
XML – Extensible Markup Language
HTTP – Hypertext Transfer Protocol
TCP – Transmission Control Protocol
EA – Enterprise Architect
RSA – Rational Software Architect
REST – Representational State Transfer
HTTP – Hypertext Transfer Protocol
DBMS – Database Management System
VR – Virtual Reality
AR – Augmented Reality
UI – User Interface

1. A Brief Thesis Introduction

Developing complex and large-scale software systems is a difficult and complicated process in which many people are involved. Software analysis and design are the first and very important phases of the process. Just like when developing a building; first a model is created and all aspects are analyzed, specified and designed in detail before the development phase begins. It requires different types of tools that are used to visualize, specify and design the systems architecture, functionality, representation of data, integration and communication of specific components from different perspectives and layers of abstraction.

The most widely and commonly used graphical language for software modeling is the UML (Unified Modeling Language) which is a world-wide standard defined and managed by the Object Management Group[1]. When designing a complex and large-scale system, the models, represented by UML diagrams, can acquire large dimensions and therefore the models are becoming chaotic and hard to read and as a result the work efficiency and productivity of software analytics, architects and developers are decreasing. Adding a third dimension to model visualization opens up new opportunities and ways of viewing the diagrams and system models. In the near future software analytics, architects and developers might be modeling systems together in three-dimensional space in virtual reality using technologies like Oculus Rift[2].

Simplifying and eliminating the problem of complex models is not the only aspect that is decreasing productivity and work efficiency in system modeling. When modeling a complicated system multiple experts with various specializations need to concurrently collaborate in order to analyze and design the system. Therefore, collaboration is another important aspect which must be considered and implemented. However, if several people are modeling a system simultaneously, it is necessary to display what each person is currently working on, what is finished, remembering and displaying history of changes or resolving any conflicts with simultaneous component modification, but the most important aspect is efficient, organized and intuitive team coordination and communication.

Therefore, the main motivation of this thesis is to create and introduce a new method for collaborative modelling and visualisation of software systems using multidimensional UML which would result in a modern and more intuitive system modeling interface with collaborative functions.

[1] http://www.omg.org
[2] https://www.oculus.com

2. Foundational Knowledge Requirements

This chapter covers required aspects that are needed to successfully design and implement a method for collaborative modeling and visualisation of software systems using 3D UML. The first subchapter introduces collaboration and describes how it should be used in groupware design. It also covers different types of collaborations and their advantages and disadvantages are presented. Another subchapter is dedicated to 3D UML system modeling and visualisation. In this subchapter UML is briefly introduced, system modeling using 3D UML is analyzed and some 3D UML visualisation methods are presented. Furthermore, approaches to how force-directed algorithms can be applied to automatically layout UML diagrams are presented. Lastly, some recent commercial tools for system modeling are analyzed with the focus on their collaborative features.

2.1. Collaboration and Its Presence in Groupware

In 1991 Ellis, et. al. [1] defined groupware as: "computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment" and this definition is still accurate today. To create a new method for collaborative system modeling it is important to understand the key aspects of collaboration and how they are applied to groupware.

2.1.1. The 3C Collaboration Model

Ellis, Gibbs, and Rein [1], are also the first to describe collaboration using the 3C Collaboration Model. Fuks et. al. [2] later adopted this model and enhanced the 3C Collaboration Model (Figure 1) as a guide for analysing groupware application domain or as a base for modelling and developing collaboration software. The three C's stand for Communication, Coordination and Cooperation and collaboration can be described as an interplay between them [2].

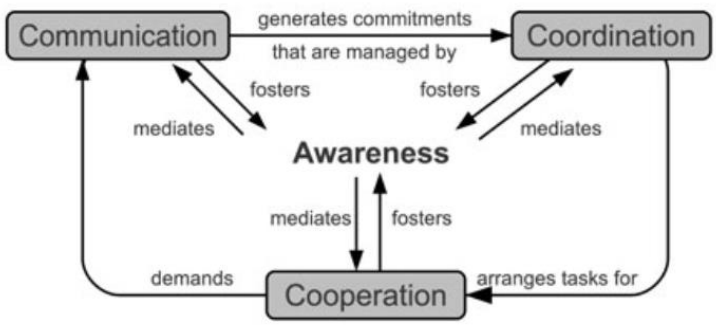

Figure 1: The 3C Collaboration Model proposed by [1] and adapted by [2]

Communication represents how information or messages are exchanged among people and how commitments are negotiated. Coordination represents how people, their activities and resources are managed as well as how conflicts are resolved. Cooperation represents how group members work together in a shared workspace and how they cooperatively generate and manipulate with objects with the common goal of completing a task. [2]

When designing a new method for collaboration it is important not to separate the three C's. There should be a constant connection between communication, coordination and cooperation. For example, chat is a common collaboration feature used for communication. However, its implementation should consist of all C's, the exchange of messages for communication, access policies for coordination and registration and sharing for coordination [3]. Therefore, from the 3C Collaboration Model perspective, the following aspects should be taken into consideration when designing a new collaboration method for system modeling:

- Improvement in how the messages are transferred
- Simplification of the way to start a conversation
- Improvement in the shared workspace and how people interact with shared objects
- Improvement in one's awareness of the other users' activities and the impact it has on collaboration
- Improvement in the support which people get for self-management

2.1.2. Awareness in Groupware

In figure 1, we could see that all three C's from the 3C Collaboration Model contribute to a person's awareness of his surroundings. Awareness is the key aspect of collaboration. Dourish and Belloti [4] defined awareness as "an understanding of the activities of others, which provides a context for one's own activities." In other words, awareness defines how aware one person is of the other's activities and how relevant his contributions to the group activities are.

Gutwin and Greenberg [5] created in their research paper workspace awareness framework which provides useful information for groupware designers. They have created two useful tables (Table 1, Table 2), where they have categorized and described elements that every designer should take into consideration when designing awareness-oriented multi-user workspace. The first table contains present related awareness elements and the second table contains past related awareness elements. They have also provided useful examples of how the elements could be implemented.

Table 1. Elements of workspace awareness relating to the present [5].

Category	Element	Specific questions
Who	Presence	Is anyone in the workspace?
	Identity	Who is participating? Who is that?
	Authorship	Who is doing that?
What	Action	What are they doing?
	Intention	What goal is that action part of?
	Artifact	What object are they working on?
Where	Location	Where are they working?
	Gaze	Where are they looking?
	View	Where can they see?
	Reach	Where can they reach?

Table 2. Elements of workspace awareness relating to the past [5].

Category	Element	Specific questions
How	Action history	How did that operation happen?
	Artifact history	How did this artifact come to be in this state?
When	Event history	When did that event happen?
Who (past)	Presence history	Who was here, and when?
Where (past)	Location history	Where has a person been?
What (past)	Action history	What has a person been doing?

The above mentioned research paper [5] also describes five common situations in shared workspace with the necessity of awareness. As the result of the analysis of the situations there are following five activities that should not be overlooked when designing a workspace for collaboration.

Management of Coupling - In collaborative environment people tend to switch between their own and shared work. With poor awareness of others, a user could miss a chance to collaborate or oppositely, could interrupt others at inappropriate time.

Simplification of Communication – If it is not necessary, the need for verbal or written communication between people should be eliminated or minimized to the maximum extent. If a person is not aware of the other's activities, he is forced to ask someone, which could be avoided and time could be saved with higher awareness. The research suggests using workspace objects as conversational artefacts or visual evidence to replace verbal communication.

Coordination of Action – People need to be aware of individual or the other's actions and their boundaries from different levels of abstraction. For example, if a person finishes a task and has a good awareness of what tasks are finished, what still has to be done, which tasks

are currently in progress, and which tasks are assigned to others, he can choose his next task more easily. Improving awareness in coordination of action can prevent work redundancy and coordination and division of labor.

Anticipation – When people collaborate they also tend to predict the others' actions and make a decision on choosing their next step based on their prediction or expectation of what others will do next. Improving awareness in this collaboration behaviour will improve peoples' predictions for their next activity.

Assistance – assisting another person with their task is a common part of collaboration. For example, one can indicate by a specific notification or visualisation that he needs assistance with a problem. When someone is not busy and knows the solution, he can provide assistance immediately. Improving awareness in this aspect of collaboration would result in better contextual understanding to where assistance is required.

2.1.3. Types of Collaboration

Ellis, et. al. [1] divided collaboration into four categories based on time and space taxonomy. He stated that groupware can either be used by a face-to-face group or by groups distributed over multiple locations and simultaneously, the communication and collaboration could take place in real-time or at different times. This resulted in four categories represented by the following time-space matrix (Figure 2).

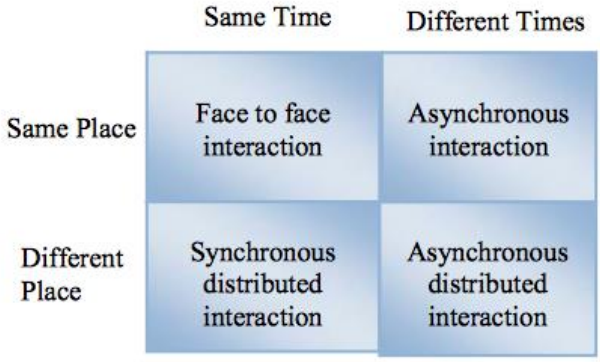

Figure 2: Groupware categorization (Time-Space Matrix) proposed by [1] adapted by [6]

However, at present, the collaboration software is commonly divided along two main dimensions:

Synchronous or Real-time Collaboration - When people work together simultaneously on one version of a project. The changes made by all people are visualized instantly.

Asynchronous or Non-real-time Collaboration - When people work together, but on different project versions. Working offline is possible and changes are merged to one common repository.

Both approaches to collaboration have advantages as well as disadvantages when modeling a system. It is important to understand them and to find balance between both approaches to determine correct requirements for a new method of collaboration. The following points describe some advantages and disadvantages of both approaches to be taken into consideration in collaborative system modeling.

- When modeling a complex system, deep thought is often required and thus real-time rapid collaboration is not ideal.
- Sometimes, an expert needs more time to think about a problem and his string of thought could be interrupted as other co-workers make changes to the model.
- In real-time collaboration immediate response and feedback can be given by other coworkers. This is a great advantage since it usually saves a lot of time with communication. A co-worker can immediately ask for help or discuss a specific problem with other co-workers.
- In asynchronous collaboration conflict resolution can be time consuming, whereas in synchronous collaboration conflicts are resolved instantly.

2.1.4. Conflicts Resolution in Collaboration

In asynchronous collaboration conflict resolution is usually done manually, since the computer is unable to tell which version is correct. In real-time collaboration conflict resolution is done immediately after each modification is made by the users. It can be challenging to implement it, but later it saves time since it does not have to be done manually as in asynchronous collaboration. The following are common ways of how conflicts are currently resolved in collaborative tools for modeling systems.

Manual

As mentioned above the conflicts that occur during asynchronous collaboration have to be resolved manually. It usually requires multiple participants to discuss their modifications, preserve only the newest or the correct changes and create one new version from the old ones. However, recently a study was conducted, in which an automatic conflict resolution at composite level in UML model versioning systems was presented [7].

Locking

This is a technique of rather preventing conflicts than resolving them, but it could also be potentially used as one possible solution to overcome conflicts in our method of

collaboration. This method can be implemented in many ways, but the idea is to temporarily prevent other users to modify an object if it is currently being modified by someone else and unlocking it only after the user has finished editing it and other users have received the updated version.

Real-time (Operational Transformation)

This method of resolving conflicts is used in real-time synchronous tools for collaboration. In real-time collaboration multiple users can modify the same object at the same time which can lead to a conflict. This conflict has to be resolved immediately, since it can result in inconsistent outcome. Implementing this type of conflict resolution is not a simple task, but there is one solution called Operational Transformation (OT) and many existing libraries that use this technique for real-time conflict resolution. The principle of OT consists of recalculating concurrent operations (representations of changes) that have been applied to the same object by different clients and computing a new operation that can be applied after the second operation, which preserves the first operation's intended change. This process of resolving conflict with OT is illustrated in the following figure and it is compared to an unresolved real-time conflict.

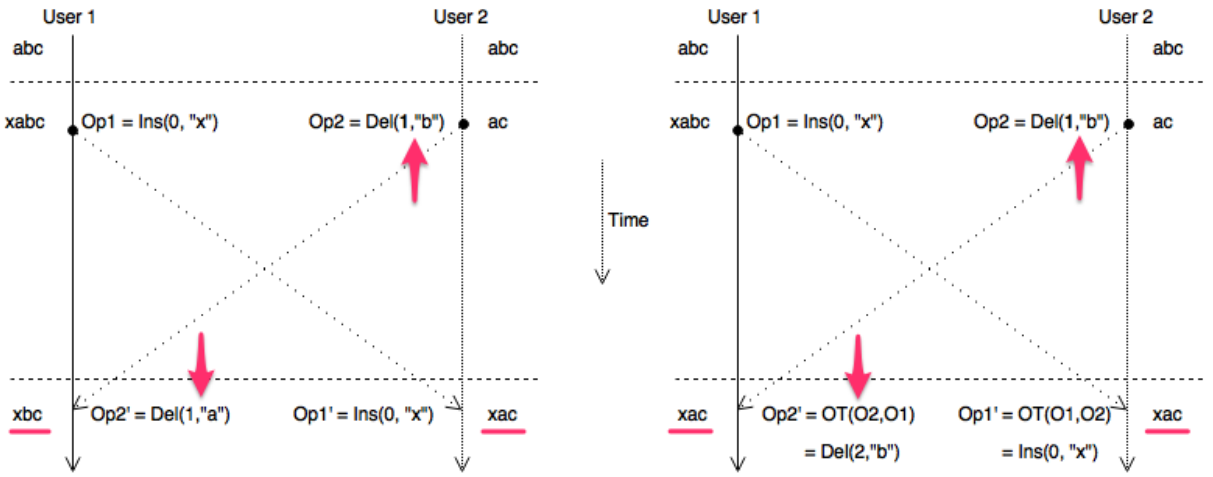

Figure 3: Unresolved conflict (left) compared to a resolved conflict by OT (right)

Both diagrams illustrate a use case where two users make concurrent changes to a simple string "abc". The first user inserts the character "x" and at the same time the second user deletes the character "b". Without OT conflict resolution both users would end up with an inconsistent result as illustrated by red underline and arrows in the figure. At present, this technique is mainly used in collaborative tools for real-time document modification. However, Chengzheng Sun shows in his research paper how operational transformation can also be used for dependency conflict resolution in real-time collaborative 3D design systems [8].

2.2. System Modelling and Visualisation Using 3D UML

2.2.1. A Brief Introduction to UML

The most common and commercially accepted graphical language for software modeling is UML (Unified Modeling Language). UML is a world-wide standard defined and managed by the OMG (Object Management Group)[3]. Various types of structural or behavioral UML diagrams are used to model a system from different levels of abstraction to better understand its requirements, specifications, information representation or dynamics. Semantics, syntax, notation and other limitations or rules for creating UML diagrams, are defined in meta-models and can be found in official UML Specifications [10,11].

2.2.2. System Modeling Using 3D UML

When designing complex and large-scale systems, the models, represented by UML diagrams, can acquire enormous dimensions, and therefore the models are becoming chaotic and hard to read. As a result, the work efficiency and productivity of software engineers is decreasing. Adding a third dimension to model visualization opens up new opportunities and ways of viewing the diagrams and system models. For example, there are not only relationships among the elements of one UML diagram, but there are also relationships among different types of diagrams as well. Viewing these relationships in 2D space would be too chaotic. Visualising these relationships in the third dimension could be beneficial for system engineers.

The idea of modeling system in three-dimensional space was first published in 1991 as a Doctoral Dissertation by Koike [12]. However, despite the fact that many research papers about 3D UML modeling have been published, 3D UML modeling still has not adapted for commercial use. In 1992, McIntosh et. al. [13] described factors or limitations why 3D system modeling had not been adopted by software engineers.

- **Computer graphics** – 3D graphics are limited by computer hardware.
- **Evolution** – Simply, Software engineering still has not evolved from adopted practises and tools.
- **Open standards** – Standardisation of 3D software visualisation is missing. 3D models cannot be easily produced and shared.
- **Perceived Return on Investment –** The benefits of creating a 3D software are not worth the invested time or money as too much effort is required.

[3] http://www.omg.org

- **Software changes –** The process of software development is changing and too change 3D software visualisation with it, would be too complex.
- **Desktop Computer Limitations** – 3D visualisation requires specialised graphics capabilities and thus it does not integrate well with standard software engineers' work flow or environment.

At present, hardware limitations are significantly eliminated. Also, with the elimination of hardware limitations, many 3D frameworks and tools emerged that simplify the creation of 3D software, thus software changes, received return on investment or desktop computer limitations are currently also irrelevant. However, open standards are still a major problem. Therefore, creating a new method for collaborative modeling and visualisation of system models in 3D according to UML standards will be essential.

2.2.3. 3D System Visualization Methods

There are many research papers [15, 16, 17, 14, 22] proposing different approaches for system visualisation in three dimensional space. In general, these approaches can be divided into two categories:

Category 1: System Modeling Using 3D Objects

These approaches transform standard 2D diagram elements and relationships between them to 3D objects visualized in 3D space. For example, Casey [16], created a Java tool, for visualizing UML class diagrams as geon diagrams. He states that it is easier for users to remember geometrical shapes then text. The following figures show an example of geon diagram compared to its equivalent UML class diagram (Figure 4) and how the relationships were visualized in a geon diagram (Figure 5).

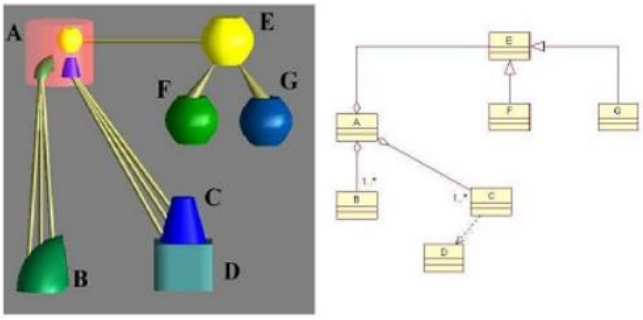

Figure 4: Geon diagram (left) compared to UML Class diagram(right) [16]

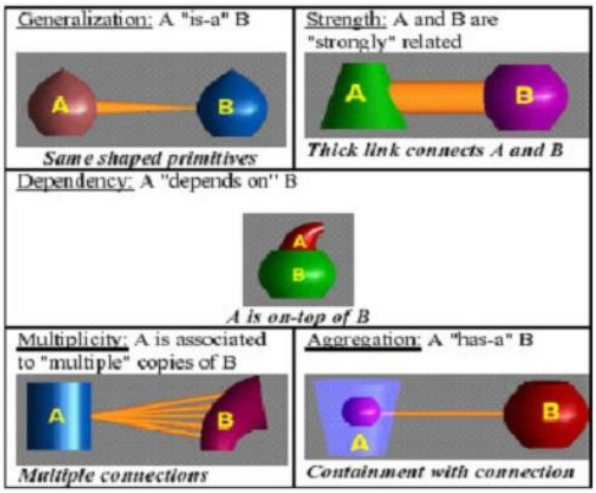

Figure 5: Representation of UML class relationships in geon diagram [16].

A different type of example is Dwyer's [15] 3D UML visualisation of a class diagram, where he used force-directed algorithm for positioning UML class in 3D space. In his visualisation, 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 (Figure 6).

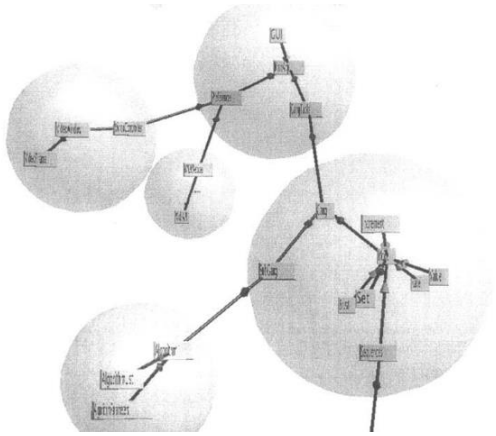

Figure 6: Dwyers' 3D visualisation of UML class diagram [15].

Category 2: System Modeling Using 2D Diagrams and Layers in 3D Space

This approach visualizes system with standard 2D UML diagrams which are located on multiple layers in 3D space, which can be arranged in various ways. The idea behind this is to preserve the UML diagram standards and use the third dimension for eliminating the problem with distribution of diagrams and visualizing relations among them. Krolovitsch and Nilsson [14] provide an example of this approach in their research paper by presenting a 3D

framework Gef3D (Figure 7), which is based on commonly used 2D framework Eclipse GEF. The following figure shows 3D visualization of series of models and relationships among them created by model-to-model transformations.

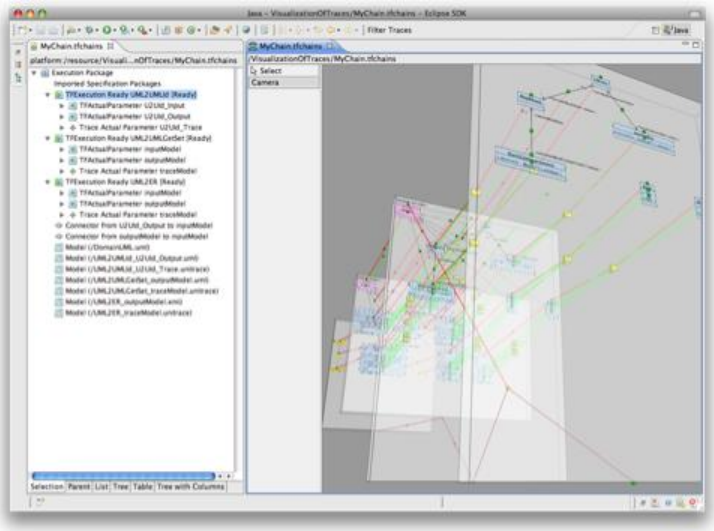

Figure 7: 3D visualization of models connected by their transformation traces [14].

Another example is a prototype that is being created at our Faculty of Informatics and Information Technologies, where the goal is to create an advanced modern environment of interconnected 3D UML diagrams in layers for modeling large software systems. The base of the prototype was created by Bc. Matej Škoda in his master's thesis [7]. The prototype was implemented as a desktop application in C++ using a framework and graphical engine Ogre3D. At present, the faculty prototype supports 3D visualisation of UML diagrams in layers, as can be seen in the following figure.

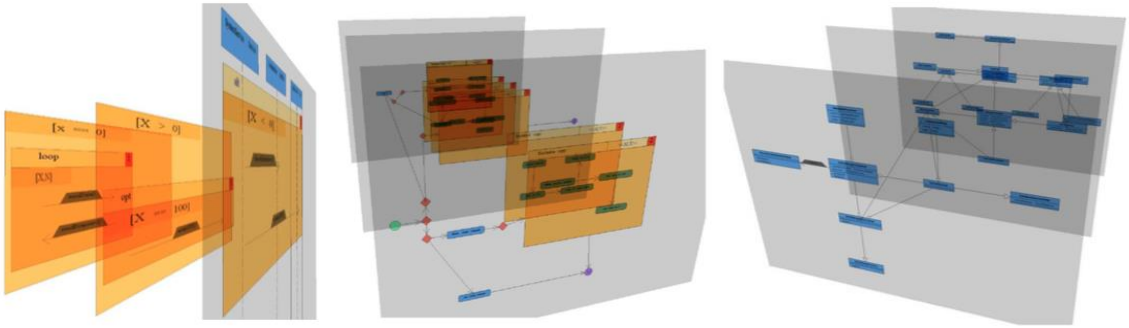

Figure 8: Faculty prototype showing sequence, activity and class diagram in 3D layers [19].

University students have been constantly improving the prototype and adding new features, such as diagram transformation among different types of diagrams, XMI exporting, automatic diagram generation from text or improving the visualisation of diagrams using various algorithms.

2.3. Force Directed Layout of UML Elements

Part of this thesis, also aims at exploring some possible methods of how UML elements could be placed or rearranged automatically, so that software engineers would benefit from it. Traditionally, all UML modeling tools require a user to place a UML element manually by hand and therefore, finding the best arrangement for UML elements is unnecessarily time consuming for all software engineers.

For automatic layout of vertices in two dimensional or three dimensional space force directed graph drawing algorithms can be used. A basic force directed algorithm is the Fruchterman-Reingold algorithm, which is used to distribute and layout vertices of a graph G = (V,E), which consists of a set of vertices V and a set of edges E, where each edge always connects two vertices [24]. The position of vertices is firstly randomly calculated and later it is recalculated and improved by a force-directed algorithm. Typically, repulsive forces are applied to the vertices of a graph, so they are repelled from each other and oppositely attractive forces are applied to the edges, which act as springs and pull the vertices towards each other.

2.3.1. Related Approaches

These force-directed algorithms can be easily used to layout UML elements, in which case UML elements could represent the graph vertices and the relations between the UML elements could represent the edges between the vertices. There are some research papers that propose effective methods of applying force-directed algorithms to layout not only two dimensional UML diagrams, but also multidimensional UML diagrams [25,26]. For example, Gregorovič [26] proposed a solution for improving the layout of a UML Class diagram based on the class semantics. He modified the Fruchterman-Reingold algorithm by adding weight to the edges, based on different types of relations, when calculating the attractive forces. He assumed that inheritance was more important than association between two classes, so he added a higher weight to generalization compared to the association relationship. Therefore, inherited classes were placed closer together, as can be seen in the following figure.

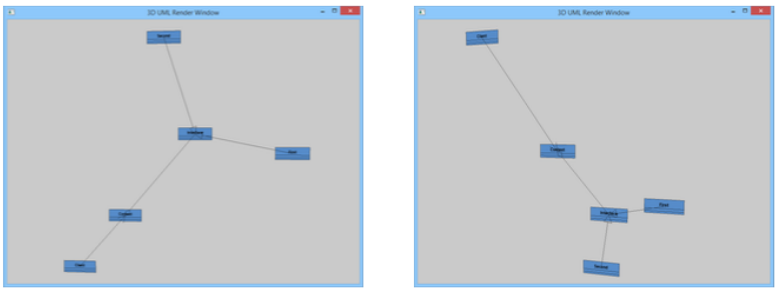

Figure 9: Fruchterman-Reingold layout (left) compared to Weighted FR layout (right) [19].

Gregorovič, in his master's thesis [19], also proposed another possible example of how his modification of the Fruchterman-Reingold algorithm could be used to layout a UML class diagram. He proposed that the UML classes did not have to be only attracted to each other based on the semantics of the relationships between them, but that they could also be attracted to each other based on their semantic similarity of names. This way the relations between classes would not act as weighted edges that attract the classes, but virtual or invisible edges would have to be created. However, the principle of adding weight to the edges would not change, and therefore the classes with similar names would be positioned closer to each other.

2.3.2. Possible Considerations for our Approach

As mentioned in the previous subchapter, there are various approaches to how UML elements could be automatically laid out using force directed algorithms and how a system can be visualized. However, since this thesis deals mainly with the problematics of collaboration in three dimensional UML modeling, it is important to analyze and consider approaches to how force directed algorithms could be used to visualize collaborative features in system modeling. A good example is a tool called Gource, introduced by Caudwell [28] which is used to visualize the development history of software systems. Software version control history is visualized as follows:

- The directories and files are represented as vertices. The directories are visualized as tree nodes with files grouped around the node as leaves.
- The edges between directories represent their hierarchical structure.
- A force directed algorithm is used to place directories and files around the root directory which is always in the center.
- The files are colored according to their type.
- A force directed algorithm is used to attract the developers to close proximity around files they have just modified, added or deleted.
- Beams of light connect the developer to the file and the color of the beam indicates the type of change the developer has just made. Red color for deletion, green for creation and orange for modification of the file.

This type of software history visualization enables a user to see the collaborators' contribution to the project, what type of change a collaborator made and when it was made. This can be seen in the following (Figure 10).

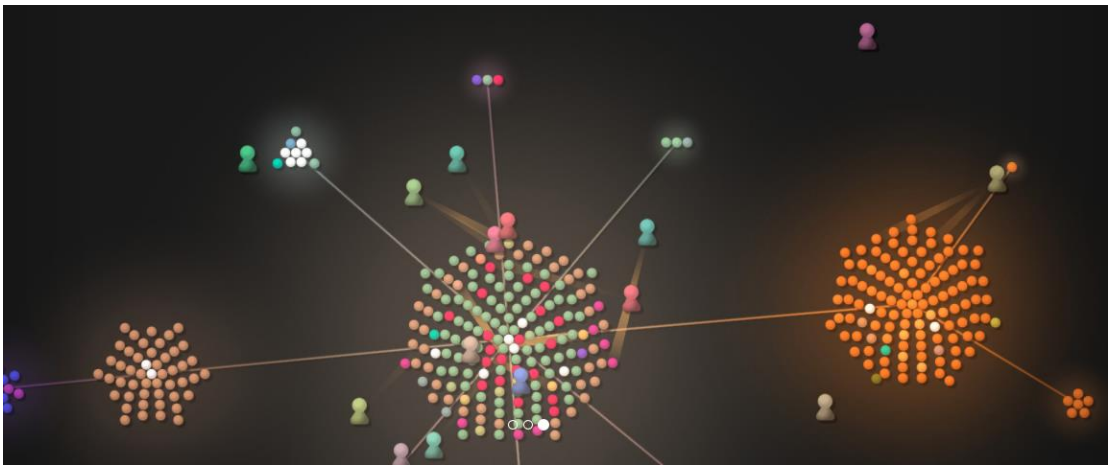

Figure 10: Gource - a software version control visualization tool [27].

In collaborative UML modeling, being able to see what was changed, when it was changed and who made the change would increase collaborators' awareness and could lead to more efficient and faster software modeling. Furthermore, based on the previous example, it is also possible to consider using the force directed algorithm for grouping UML elements by criteria such as the following:

- By the time of the changes – the most recently changed UML elements would be grouped closer together. This would allow the collaborators to quickly find the most recently modified elements.
- By the collaborator – the UML elements, that a collaborator worked on would be grouped closer together which would allow other collaborators to quickly visualize all UML elements a specific collaborator modified.
- By the type of change – the UML elements would be grouped by the specific type of change. For example, collaborators could quickly visualize only UML elements where something was deleted if deletion was the type of change they were looking for.

Using a force directed algorithm to group UML elements by the above criteria or their combination would enable any collaborator to find answers to the following questions more quickly. Who made the change? When was the changed made? What was changed? The above are only a few examples from which the collaborators could benefit and should be considered when designing a collaborative three dimension UML modeling method.

2.4. Collaboration Features in Commercial UML Modeling Tools

If our desire is to design and implement a new method for collaboration and visualization using 3D UML Modeling, it is essential to also analyze commercial software used on day-to-day basis by software engineers. The public adopted this software for a reason and by understanding the features used for collaboration and advantages of this software we can enhance and apply them in our own method. At present, there are many tools used for collaboration, however, each tool should offer features such as connectedness, awareness, sharing and communication in order to be considered as one [9].

2.4.1. **Enterprise Architect (EA)[4] and IBM Rational Software Architect (RSA)** [5]

Both of these tools are leading professional applications used worldwide by major software companies for large and complex system modeling. These tools were developed by professionals to meet every requirement for a system modeling tool. Both of them provide a full set of features for modeling systems in 2D UML. When it comes to collaboration, these tools provide only features for mostly asynchronous collaboration. For example, EA allows many concurrent users to model UML models by providing the following[6]:

- **Support for sharing model data in dedicated server based repositories**
  Rather than storing UML Models locally in standard .EAP files, EA allows the models to be stored in database management systems. EA also provides initialization SQL scripts for creating database schemas and for loading the initial base data into the database.

- **Role-based user security**
  EA provides access control functionality such as restricting the modification of models to users with certain privileges only or per-user and per-group locking of elements. This prevents the unintentional modification of elements by multiple users at the same time.

- **Export and Import of models into XMI**
  XMI format is an OMG standard for exchanging metadata information via XML. This means that EA models can be easily shared, imported into other models or tools or maintained in version control systems.

[4] http://www.sparxsystems.com/products/ea/
[5] https://www.ibm.com/developerworks/downloads/r/architect/
[6] http://www.sparxsystems.com.au/enterprise-architect/distributed-teams-collaboration/distributed-teams-collaboration.html

- **The Team Review facility**
  The Team Review facility is EA's interface for collaboration. It enables team members to communicate through messages. A user can read a message, reply to a message or link a message to a specific element in a model.

Both EA and RSA, also allow the creation of custom plug-ins to extend their functionality, thus users can build their own solutions for collaboration. However, there are still not official widely used real-time collaboration solutions. We have found only one research paper in which a prototype for Real-Time Collaborative Software Modeling Using UML with RSA is introduced [20].

2.4.2. Draw.io

Draw.io[7] is a tool for real-time synchronous collaboration. It enables diagram drawing such as flowcharts or UML diagrams. It was developed by Google using the Google Drive Realtime API which uses OT (Operational Transforms) for real-time conflict resolution. Every user can instantly see the updates from other collaborators. The updates are visualized by colored indicators showing who has changed the diagram and where the change has occurred. This can be seen in the following figure and it is illustrated with red arrows.

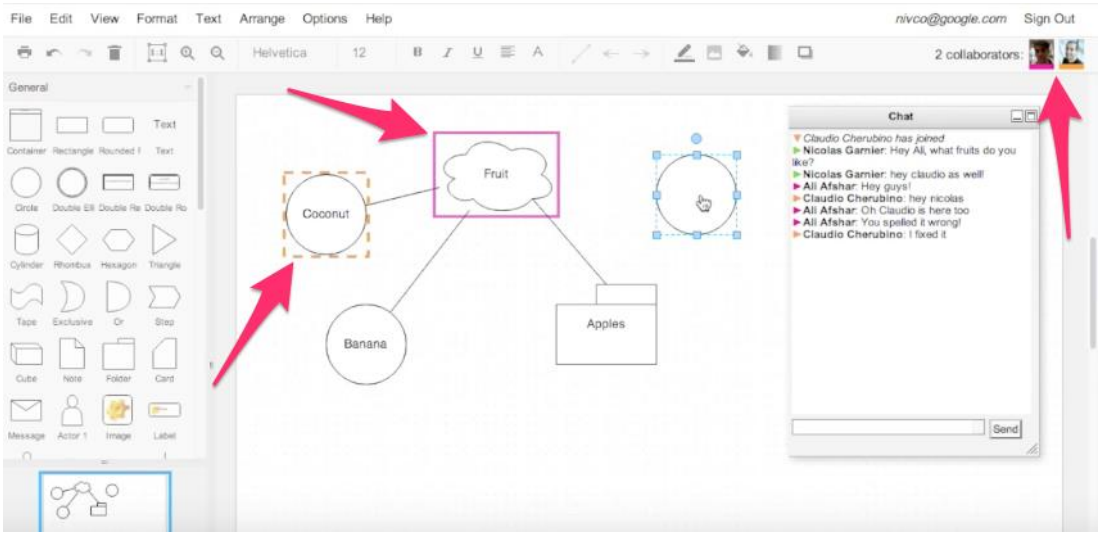

Figure 11: Multiple users collaborating on a diagram in real-time using rt.draw.io.

The advantage of this tool is a great real-time visualization and awareness of other collaborators' contributions, but on the other hand, this tool is not designed for modeling large complex systems with many interconnected diagrams. It only supports basic UML diagrams.

[7] https://rt.draw.io

2.4.3. LucidChart

LucidChart[8] is another online real-time diagraming and visual communication solution. It is similar to draw.io, but a great advantage of LucidChart is the revision history feature. LucidChart saves full history of every document. Each entry in the timeline represents a save point regardless of how much a document changed between saves. It shows the date, time and collaborator of the save. By clicking on the history entry, a preview of how the document looked at a given time is shown. This is a great example of who, when and what awareness (see subchapter 2.1.2). It is possible to revert the document to a chosen revision or create a new document from that point in history. The following figure shows two people collaborating in LucidChart and the one on the right shows previewing the document at a specific point in time.

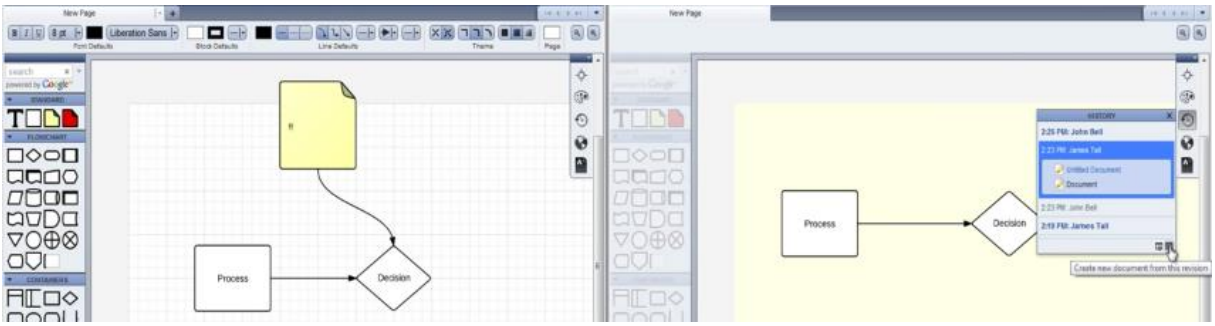

Figure 12: Revision history feature in LucidChart

Similar to draw.io, LucidChart is not an ideal tool for large complex system modeling but the revision history is a great awareness feature that many of the collaborative tools are missing.

2.4.4. Git

One of the best examples of asynchronous collaborative development is Git[9]. Git is an open source distributed version control system. It allows a team of people to collaborate using the same files and eliminates the confusion when multiple people edit them. Since this is an asynchronous collaboration approach, it means that every team member can pull recent changes from one common project repository and have their own local version of the project files. Each member can modify the files and synchronize the changes back to one main repository. Git automatically records who changed the file, when the file was changed and what was changed. Therefore, each file has a revision history so it is possible to go back to a specific version in case of inconsistencies. Git also allows merging multiple sets of changes into one file and if any conflicts occur during the merge, they are highlighted for

[8] https://www.lucidchart.com
[9] https://git-scm.com

easier resolution. Git is also used by major organizations like Microsoft, Google or Facebook on daily basis.

Main advantages of using Git are:

- It is possible to work offline, since all files are local.
- It is not possible to overwrite the files and lose older version.
- People can simultaneously collaborate on the same files without real-time conflicts.
- It eliminates single point of failure. If there is a problem with the main repository on the central server, everybody usually has an up-to-date copy of the project as a backup.
- In case of a problem, it is possible to simply revert back to an older version of the file or project.
- It is possible to add, modify, test or experiment with new features of the project without modifying the one in production. When everything is tested, the changes are synchronized to the main project in production.

Implementing Git-like versioning and merging system to our new method of collaboration can be taken into consideration, since it has various advantages over the real-time synchronous collaboration tools.

All of the tools mentioned above are great examples of collaborative functions and features that we can consider using or get inspired by when designing our new method of collaborative system modeling.

3. Designing the Method for Collaborative 3D System Modelling

3.1. System Requirements and Considerations for Our Approach

Based on analyses from previous chapters we have learnt important factors that we need to consider for creating a new method for collaborative modeling and visualization of systems using 3D UML.

We have learnt that good collaboration in groupware is an interlay between coordination, communication and cooperation among users and that users' awareness plays an important role in collaboration. A user needs to be aware of his surroundings as well as of his own contributions to group work.

We have also summarized some limitations why 3D system modeling is still not commonly used among software engineers. Most of these problems are nowadays irrelevant, but two of them still need to be taken into consideration. The first limitation proposes that software engineers have their adopted practices and tools from which it is difficult to evolve. The second limitation states that most of the 3D software is not standardized and therefore producing and sharing 3D models is more complicated.

Based on the above factors we created the following high level system requirements for our solution:

- Rather than creating a new tool and forcing users to use it, allow them to use tools they have already adopted, by only enhancing them with collaboration and 3D system visualisation features.

- Design and development of the new tool or extension should be done according to world-wide standards. Therefore, 3D diagrams should be valid against meta-models so they can be easily imported or shared between other modeling tools as well.

- The users' awareness factor in collaboration should be as high as possible. To achieve this, real-time synchronous collaboration should be possible as in asynchronous groupware it is harder to stay aware, at all times, of the others' activities. However, sometimes real-time collaboration is not ideal. Many times a deep thought is required when the user should not be distracted or disturbed by other users. Therefore, it is important to also allow asynchronous collaboration and find balance between them.

To meet these requirements, we have proposed an approach for designing a tool, where the method for collaborative modeling and visualization of systems using 3D UML could be applied. We decided to design a tool in which users could collaboratively model and visualize systems using 3D UML but also utilize existing powerful tools for UML modeling such as EA or RSA.

3.2. High Level Overview of System Architecture

This following design of system architecture constructs a method for collaborative system modeling and visualisation using 3D UML. The setup is flexible enough to meet every user's requirement. Multiple people can communicate and easily and simultaneously interact with the application from anywhere in the world. Web Application can be accessed anywhere and by anyone since the only requirement is a web browser. Users can utilize environments that they have adopted. Both real-time synchronous and asynchronous types of collaboration are enabled. Users can work offline, if needed, or if they do not want to be disturbed or distracted by others. Room for increasing the awareness of users in collaboration is created and 3D UML diagrams are standardized.

This architecture setup is designed to allow five use cases how a user can collaboratively model and visualize a system:

1. A user can use only our 3D UML Application for modeling and visualizing a system in 3D, where real-time synchronous collaboration will be possible (see User 1 in Figure 13).

2. A user can use our 3D UML Collaboration EA Add-in for normal 2D system modeling in EA, but with real-time synchronous collaboration features provided by our Add-in (see User 3 in Figure 13).

3. A user can use EA normally with EA's supported powerful collaborative features (see User 4 in Figure 13).

4. By standardizing UML models, the integration of other applications is easier. For example, an application for UML modeling in virtual reality could be additionally created. (see User 5 in Figure 13).

5. A user can use any combination of the above methods. For example, he can use our 3D UML Collaboration EA Add-in only for collaboration and 2D system modeling and simultaneously use our 3D UML Application for real-time 3D visualization of the same model (see User 2 in Figure 13).

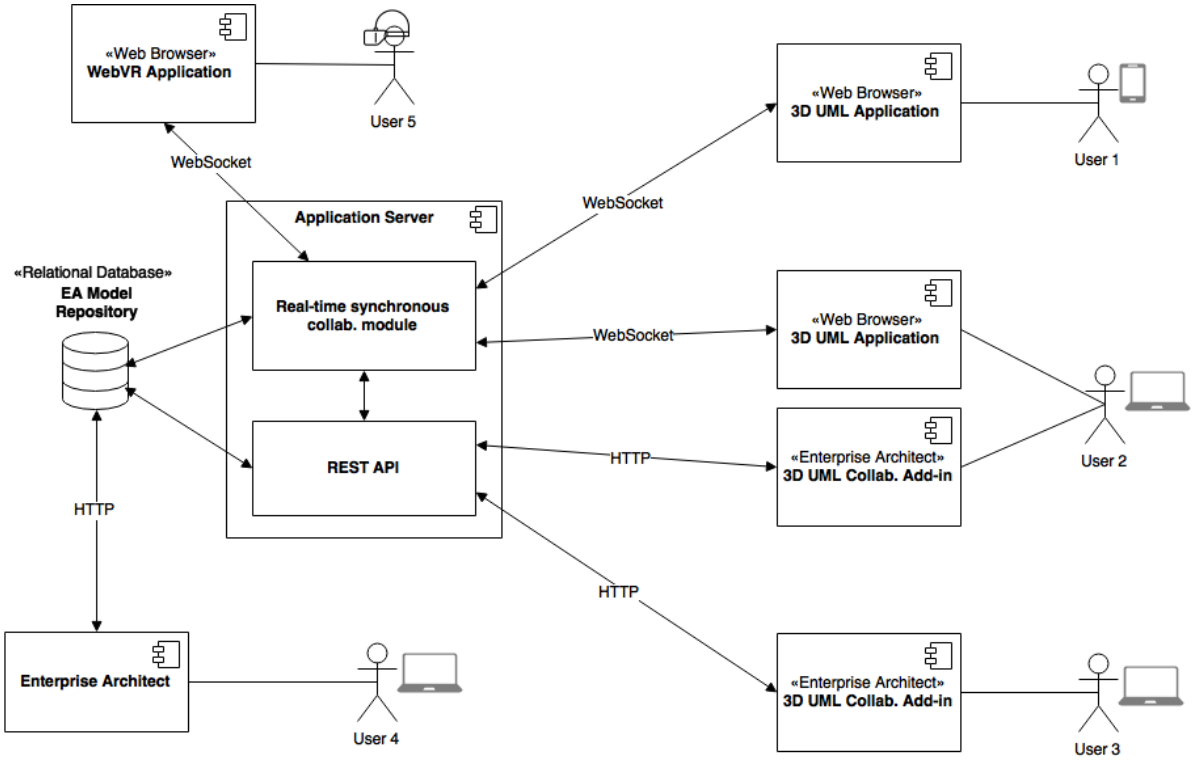

Figure 13: High level system architecture design

3.2.1. Relational Database

This will be our dedicated server based repository for storing shared system models as well as other relevant data. It can be any DBMS, as long as it is supported by EA to enable the users to use standard collaboration feature provided by EA. By using EA's existing physical data model, the UML models would be standardized and valid against meta-models and they could be easily imported or shared among other modeling tools as well.

3.2.2. Application Server

This is the central server for managing communication among multiple clients. Its main responsibilities will be the exchange of data, synchronization, versioning or conflict resolution. It can consist of many components or modules that are in charge of these functions. Where fast real-time synchronous communication is required, WebSockets will be used (see subchapter 3.3 for reasoning) and for normal requests, that are not required very often, standard REST API can be implemented.

3.2.3. 3D UML Application

This component represents our main tool used for real-time synchronous collaborative modeling and visualizing systems 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 will be the main goal. Here users will be able to create and view UML diagrams in 3D space; therefore, it should be powerful enough to render many objects in 3D space. We considered enhancing the original faculty prototype with collaborative functionality as one of possible solutions for implementing this tool. However, we decided to design this tool as a web application, since the original prototype was developed as a heavy graphical desktop application, which is not ideal for real time collaboration (see subchapter 3.3 for reasoning).

3.2.4. Enterprise Architect and 3D UML Collaboration Add-in

The main idea for integrating EA into our collaborative method is to allow users to also normally work in a powerful widely used tool for system modeling they have already adopted as well as to utilize EA's built-in features for complex system modeling. However, to enable real-time 3D visualization of EA's 2D models a custom add-in is required. The 3D UML Collaboration Add-in would also enhance EA with real-time collaborative features that would increase users' awareness.

3.2.5. WebVR Application

This component was added to prove that by standardizing UML models the integration of other applications is easier and also to experiment with system visualization in virtual reality. This opens another dimension how systems could be visualized and modeled in the future.

3.3. Choosing the Right Technology

As mentions in the previous chapter, our primary goal was to enhance the original faculty prototype with collaborative functionality to create our collaborative 3D UML application. The original faculty prototype was implemented as a desktop application in C++ using the framework and graphical engine Ogre3D. At the time when the original prototype was being developed, using this technology was the best choice for implementing a 3D modeling application. Ogre3D graphical engine development was well supported, and widely popular for developing computer games, which required high graphical performance. However, time showed that development in this technology was too time consuming and overly complex for modeling diagrams that consisted of simple geometry shapes. For this reason, we first

decided to migrate the original prototype into a newer and more suitable technology that would also make the implementation of collaboration easier.

Since the original prototype was created, web technologies have rapidly gained popularity. Graphical engines and frameworks were created that support development of hardware accelerated 3D applications directly in the web browser. We have also compared the popularity of Ogre3D with one of its web-based alternatives called WebGL using Google Trends (Figure 14). The results prove that popularity of Ogre3D is decreasing compared to the rapid increase of WebGL popularity.

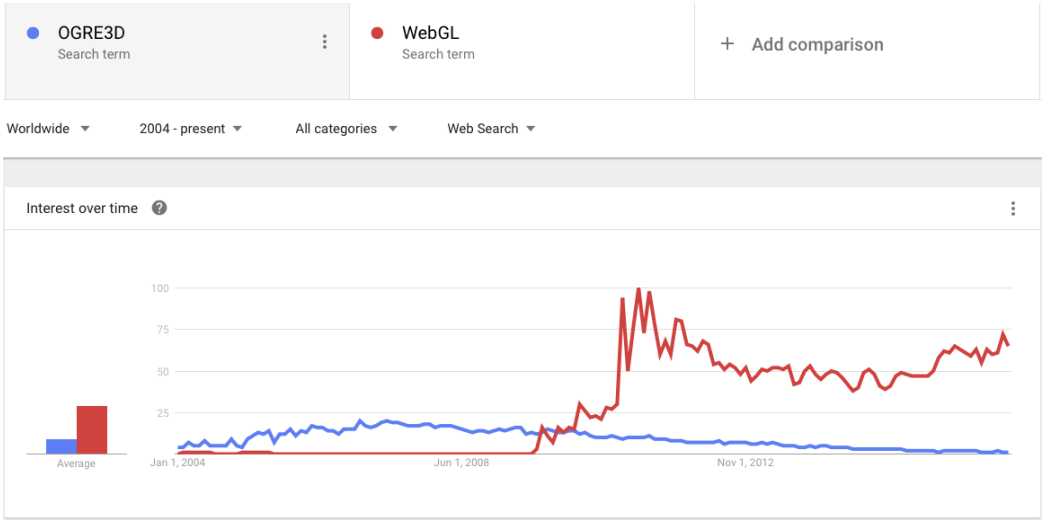

Figure 14: Comparison of Ogre3D and WebGL Search Terms on Google Trends[10]

Migrating the heavy graphical desktop client into a web application would also bring many other benefits than just the advantage of a large development community. Taking collaboration into consideration, the following are just a few examples: web-based application would be platform independent; it would be easily accessible by anyone with a web browser and from anywhere; no installation would be required and the integration of other systems would be easier. Knowing the fact, that web-based technology is ideal for collaboration and that the creation of 3D graphical applications is possible in the browser, we decided to design and implement the prototype as a web application.

3.3.1. Technology for Synchronous Real-time Collaboration

Supporting real-time collaboration functionality in a 3D graphical application means that the rate of messages transmitted over the Internet between clients and server will be very high at all times so everyone can stay up to date. This is why we have to take into consideration

[10] https://www.google.com/trends/

many aspects such as the lowest possible latency, high transmission rate, small overhead of messages sent, scalability of the application and many other factors. A study [21] carried out in several network settings to test the performance of different technologies that support real-time collaborative communication shows (Figure 15) that WebSockets are the best choice of technology, which provides fast and consistent performance in web-based applications [21].

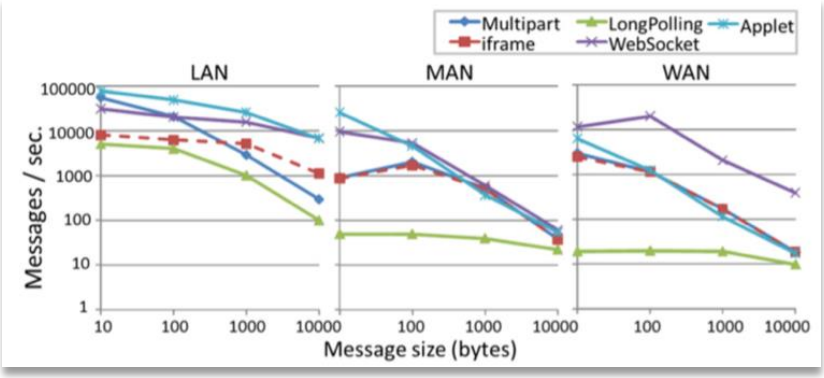

Figure 15: Comparison of web-based technologies for collaboration [21].

WebSockets provide an asynchronous, bidirectional, full-duplex communication with minimum massage overhead where a standard three-way handshake TCP connection is established between client and server. Afterwards event-based messages are streamed. Compared to standard HTTP communication, when the request and the response need to be sent every time and typically the entire page has to be refreshed, it limits the update rate to less than one frame per second [21]. WebSockets are part of HTML5 standard and are supported by all major web browsers. Data can be sent in two standards JSON or XML and also binary streaming is supported. Therefore, even files, audio or video can be transmitted back and forth at a very high rate which makes WebSocktes an ideal technology for all kinds of collaborative functionality.

3.3.2. Technology for 3D Graphical Rendering

Due to our decision, based on the previous chapters, to develop our 3D UML Collaboration component as web application, it was essential for us to analyze web best graphical solutions for rendering 3D UML diagrams. Currently, there are three commonly used technologies for creating 3D graphical applications [23]:

- **WebGL –** Enables hardware-accelerated 3D rendering with JavaScript. It is a standard, based on OpenGL graphics API supported by nearly all web browsers.

- **CSS 3D Transformations –** This technology is the most recent. It evolved from basic CSS3 3D transforms, transitions or custom-filters used for advanced page effects. It now also supports hardware-accelerated 3D rendering and animation features through CSS and JavaScript.

- **The HTML5 Canvas Element –** HTML DOM element with 2D drawing context API. It is used for drawing arbitrary 2D graphics. However, the 3D effects can be added manually with JavaScript. It could be used as an alternative to CSS 3D Transformations or WebGL when in specific and rare cases they are not supported.

We further researched and analyzed which one of these three technologies would be the best suitable solution for our web application. We eliminated the HTML Canvas element, due to the reason that the 3D graphical effects need to be created manually. Our further research, focusing on our needs and requirements, together with hands on experimentation with the two technologies, resulted in the following observations:

- Creating and working with a 3D object using WebGL is much easier compared to CSS 3D Transformations, where the 3D objects have to be created and styled manually using HTML/CSS/JS and they can later be rendered as CSS 3D Objects. WebGL was developed for creating and working with complex geometry shapes. Textures or materials can be easily added to 3D object as well as shading or lighting effects.

- Connecting the 3D objects (UML associations) is also much easier in WebGL. WebGL supports functionality for drawing lines between two vertexes. When using CSS 3D Transformations, this functionality would have to be implemented manually, which could present a complex problem.

- In addition, working with text in WebGL is problematic, since it is used mainly for working with complex geometrical shapes. On the other hand, since 3D objects are in CSS 3D Transformations created by using basic HTML/CSS/JS, text is natively supported.

- For creating 3D UML diagrams, the decision which technology is better, depends on the method of their visualization. If the UML diagrams as well as their elements would be represented by 3D objects in 3D space, then WebGL would be a better choice, since the creation of a 3D object is easier compared to CSS 3D Transformations (see first bullet). However, if the UML diagrams would be 2D and visualized on layers in 3D space, then the development in CSS 3D Transformations would be beneficial. This approach enables us to separate the implementation of the 3D world (layers) from the implantation of the UML diagrams. Therefore, it is even

possible to use any standard HTML/CSS/JS functions or libraries to create or interact with the UML diagrams.

- Both WebGL and CSS 3D Transformations work well with WebSockets for adding collaborative functionality.

To summarize our list and to make a decision we created the following table to compare and evaluate the two technologies (Table 3). We have also assigned weight to each functionality between 1-5 (5 being the highest evaluation). However, we assigned the weight taking into account the fact that for the 3D visualization the layers method would be used.

Table 3. WebGL and CSS 3D Transformations comparison

	WebGL		**CSS 3D Transformations**	
Functionality	**Description**	**Weight**	**Description**	**Weight**
Web Browser support	all major browsers	5	all major browsers	5
Hardware accelerated 3D Rendering	yes	5	yes	5
Creation of 3D objects	programmatically, but simple	5	declaratively, but complicated	1
Styling the 3D objects	textures/materials	0 (N/A)	CSS	4
Connecting the 3D Objects	natively supported	5	needs implementation	0
Working with text	limited	1	natively supported	5
Collaborative functionality (WebSockets)	supported	5	supported	5
Creation of UML diagrams	complicated	1	simple	5
Libraries for UML	none	0	only for few types of UML diagrams	3
User interaction with UML diagrams and UI/UX	complicated	1	simple	5
Total		**28**		**38**

The Decision

WebGL, compared to CSS 3D Transformations, is very powerful for creating complex 3D objects and scenes. However, by using the layers method for visualizing 3D systems, the UML diagrams will consist of text and only simple geometrical shapes like squares or lines therefore, using WebGL as a graphical framework for complex geometry shapes is contra-productive. Also, a great advantage of CSS 3D Transformations, is that it is possible to utilize

existing tools or libraries for UML modeling, that provide extensive functionality or even validation against UML metamodels. The only major disadvantage is the problem with connecting objects in 3D space. This will need to be implemented manually, however the other advantages of using CSS 3D Transformations far out-weigh this only major disadvantage. Therefore, CSS 3D Transformations will be used to implement our 3D UML Application component.

3.4. Designing the UI for Maximum Awareness

Arising from the analysis of collaboration and UML modeling, the users' awareness factor in collaboration should be as high as possible, so collaborators could work faster and more efficiently. Collaborators need to understand the activities of others to require a context for their own activities and therefore, the design of awareness-oriented multi-user workspace forms an important part of this thesis.

3.4.1. The UI Design

To verify and evaluate our designed method, we have decided to create a prototype for modeling UML class diagrams, in which multiple UML class diagrams could be simultaneously created by multiple collaborators in real-time, and each UML class diagram would be located on a separate layer in 3D space. Firstly, a high level UI design was created, which can be seen in the following figure. The UI is designed to meet the criteria Gutwin and Greenberg [5] proposed in their research paper in which they categorized and described elements that every designer should take into consideration when designing awareness-oriented multi-user workspace.

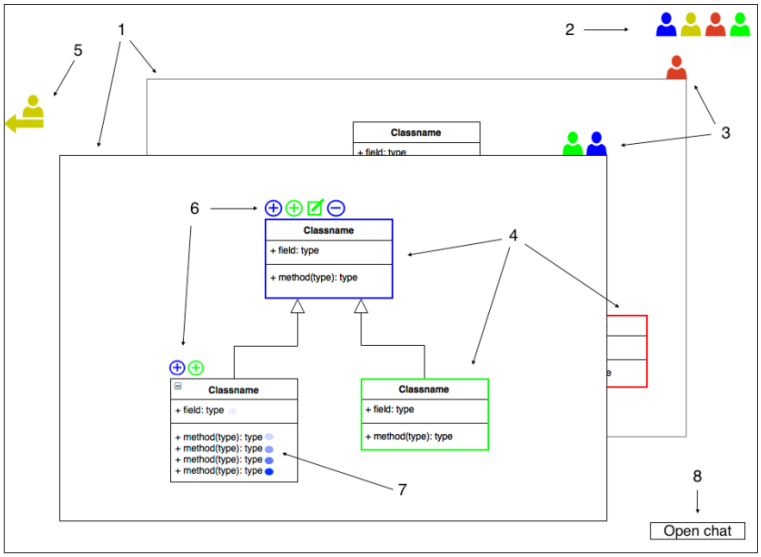

Figure 16: UI design with awareness elements and features.

The following table was created to explain the designed awareness features from the above figure and at the same time to provide answers to the questions from the two tables from chapter 2.1.2 which Gutwin and Greenberg [5] have created.

Table 4: Description of the designed collaborative user awareness features.

#	Description and Significance	Category	Specific Question
1	UML class diagrams on two layers in three dimensional space. The goal is to simplify the complexity of 2D UML models.	N/A	N/A
2	All online collaborators. A unique color is assigned to each collaborator. When the user's icon is clicked his name can be shown. Also, user's icons can be replaced by user's photo with a colored border for easier identification.	Who (present)	Is anyone in the workspace? Who is that? Who is doing that?
3	Collaborator's icon located on top of the layer indicates on what layer the collaborator is currently working.	What Where (present)	What object are they working on? What are they doing? Where are they working?
4	The object where the collaborator is currently working is highlighted with collaborator's color.		
5	When working in 3D environment not all objects can be seen by a collaborator at all times. Sometimes a layer can be located outside of the collaborator's view. Therefore, we designed a small user's icon with an arrow pointing to the direction of user's current action if it is happening out of view.		
6	Action history of each class is located on top of each class. The specific icon indicates what type of change happened to the object and the color of the icon indicates who made the change. The plus icon indicates that a method or attribute has been added to the class, the minus icon indicates that a method or attribute has been deleted and the edit icon indicates that something has been modified. If the icon is clicked more detail can be shown, such as the time of the change or what exactly has been changed.	How When Who Where What (past)	How did that operation happen? How did this artefact come to be in this state? When did that event happen? Who was here and when? Where has a person been?

7	Whenever a user makes a change to a class attribute or method a small dot is added beside the element he has modified. The dot has the same color as the user and it fades out as it is further in history. This way collaborators can quickly see what exactly the collaborator changed and in what order.		What has a person been doing?
8	Basic chat provides a simple way for a user to ask others for assistance.	All of the above	All of the above

3.4.2. Solutions to Common Activities in Collaboration

The UI design also provides solutions to many activities that are commonly overlooked when designing a workspace for collaboration.

Simplification of Communication
The always visible user action icons minimize the need for communication. They are used as conversational artefacts or visual evidence to replace verbal communication and therefore the collaborators are not forced to ask someone about others activities or changes made to the UML model.

Coordination of Action
All collaborators are always aware of others actions, where they are working and what they are working on. A collaborator can easily see which tasks are in progress, and which tasks are currently being done by others. Therefore, a collaborator can choose his next task more easily. This improves the awareness in coordination of action and prevents work redundancy, coordination and division of labor.

Anticipation
Also, 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.

Assistance
A quick access to basic chat provides a simple way for a collaborator to ask others for assistance. If someone is not busy and knows the solution, he can instantly take over the user's task and provide assistance immediately.

3.4.3. Layout of UML Diagrams

During the analysis phase of this thesis, a few approaches were proposed of how force directed algorithms could be applied to automatically layout UML diagrams. The UML elements could be automatically rearranged in a way that would allow collaborators to get a better understanding of the actions of others or to find the desired change or element more efficiently. The following are the proposed criteria that could be applied as weights for the force directed algorithm:

- **The time of change** – the most recently changed UML elements would have the highest weight which would result in grouping the most recently changed elements closely together. This would allow the collaborators to quickly find the most recently modified elements.

- **The specific collaborator** – an additional weight would be only added to the UML elements, on which a collaborator worked. This would allow other collaborators to quickly visualize all UML elements that a specific collaborator has modified.

- **The type of change** – an additional weight would be only added to the specific type of change. UML elements where a specific change occurred would be grouped closer together. For example, collaborators could quickly visualize only UML elements in which something has been deleted if deletion was the desired type of change.

- **Any combination of the above** – the final weight would be a sum of the specific combination of the above. Collaborators could toggle between any combination of the above criteria. For example, one common use case in collaborative UML modeling is that a collaborator needs to quickly find what has been most recently changed by a specific collaborator. This example is visualized in the following figure which demonstrates the layout of class diagram before and after the application of force directed algorithm.

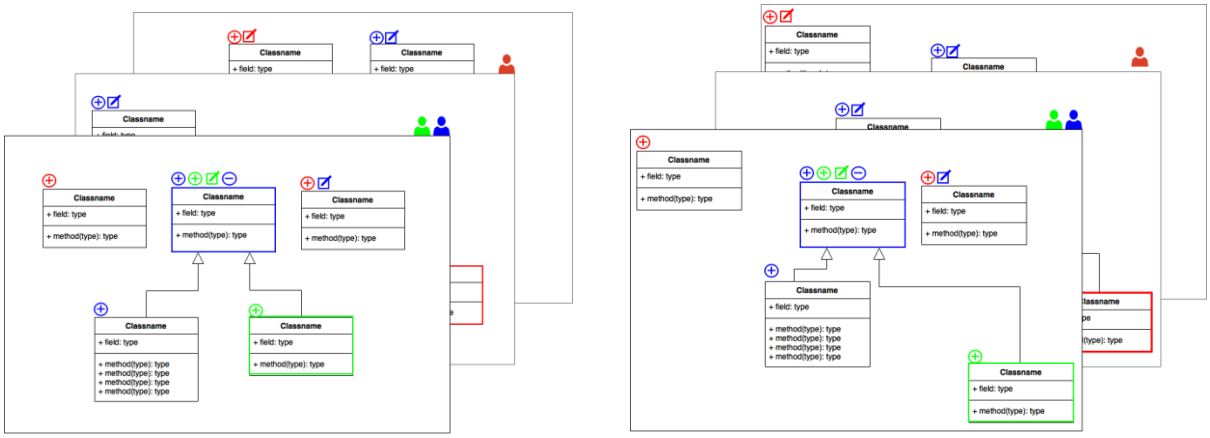

Figure 17: Automatic layout of UML class diagram based on user's actions.

We can see that after applying a force directed algorithm the most recently modified classes by the blue user are positioned closer together in the center and classes that were not modified by the blue user are moved to the sides. If the system was more complex with many UML elements, this could be a great feature, since the collaborators would not have to look in the history log, search the workspace or even ask the blue user verbally.

3.5. Physical Data Model

When designing a tool for system modeling using UML diagrams it is crucial to follow worldwide standards. The syntax, semantics and other rules and constraints for creating UML diagrams are defined in metamodels defined and managed by the ORM. From our perspective, to follow the standards is of key importance, so the integration of other systems would be easier. We designed the physical data model (Figure 18) for storing UML class diagrams based on the Enterprise Architect's XMI export of a UML class diagram. This data model was designed to serve only for our prototype need. However, the original data model provided by EA could be used in the future.

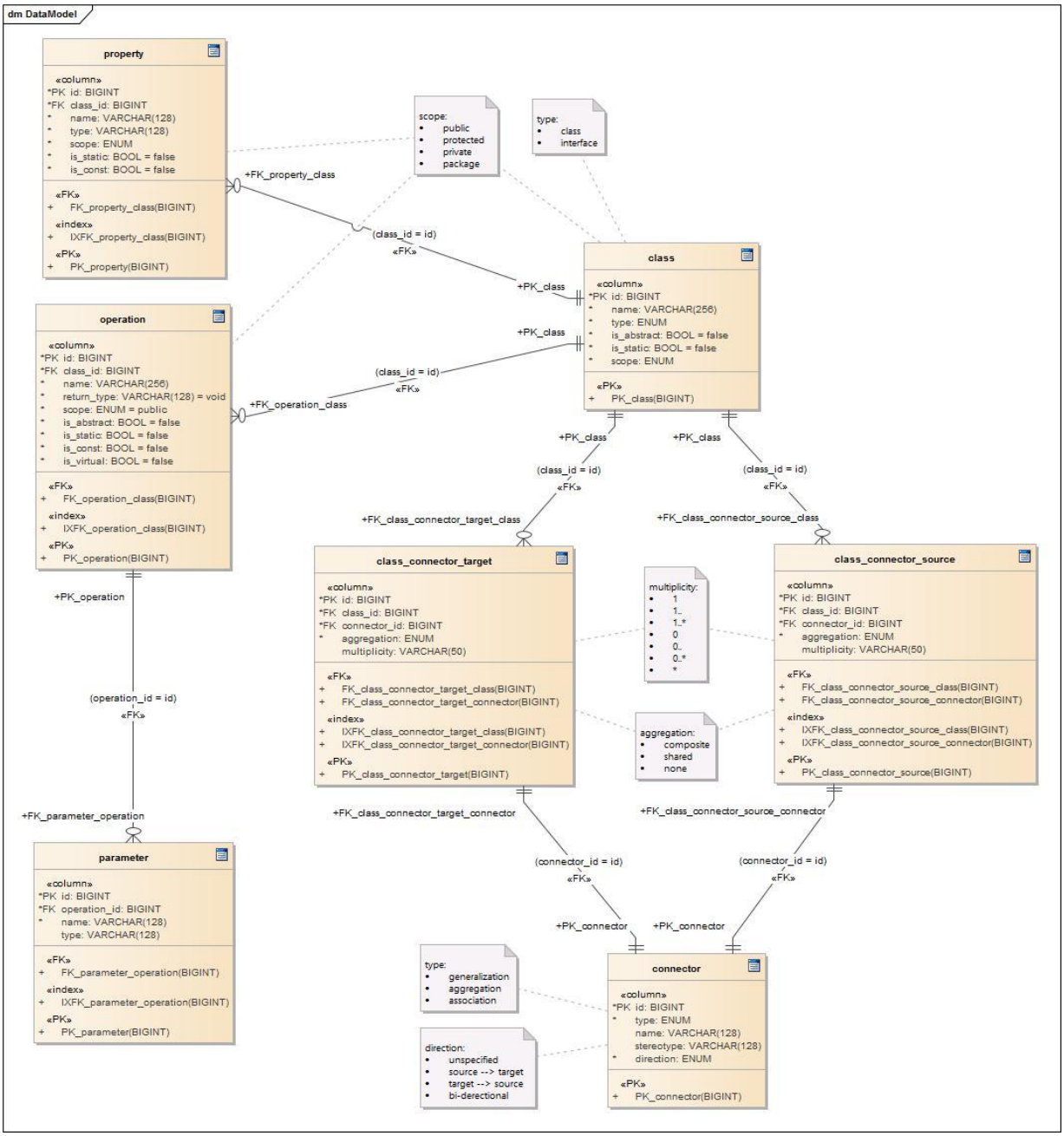

Figure 18: Physical data model design based on XMI export of UML class diagram from EA.

4. Implementing the Collaborative 3D Modeling Method

4.1. First Phase: Implementing Real-time Collaborative Functionality

Our prototype runs in Heroku, a Platform as a Service (PaaS) providing a scalable environment in which developers can build and run applications in the cloud. We chose Heroku as a dedicated server to test the performance of our prototype in production environment. One of the main reasons why Node.js was used for the server-side environment was that it enables server-side JavaScript language. Therefore, communication via WebSockets can be easily implemented. For handling HTTP requests, we used experess.js framework built on top of Node.js. Socket.io API enables real-time communication between web clients and server. In the following figure we can see our cloud schema design with an example of a chat message broadcasted from client A to all other connected clients (Figure 19).

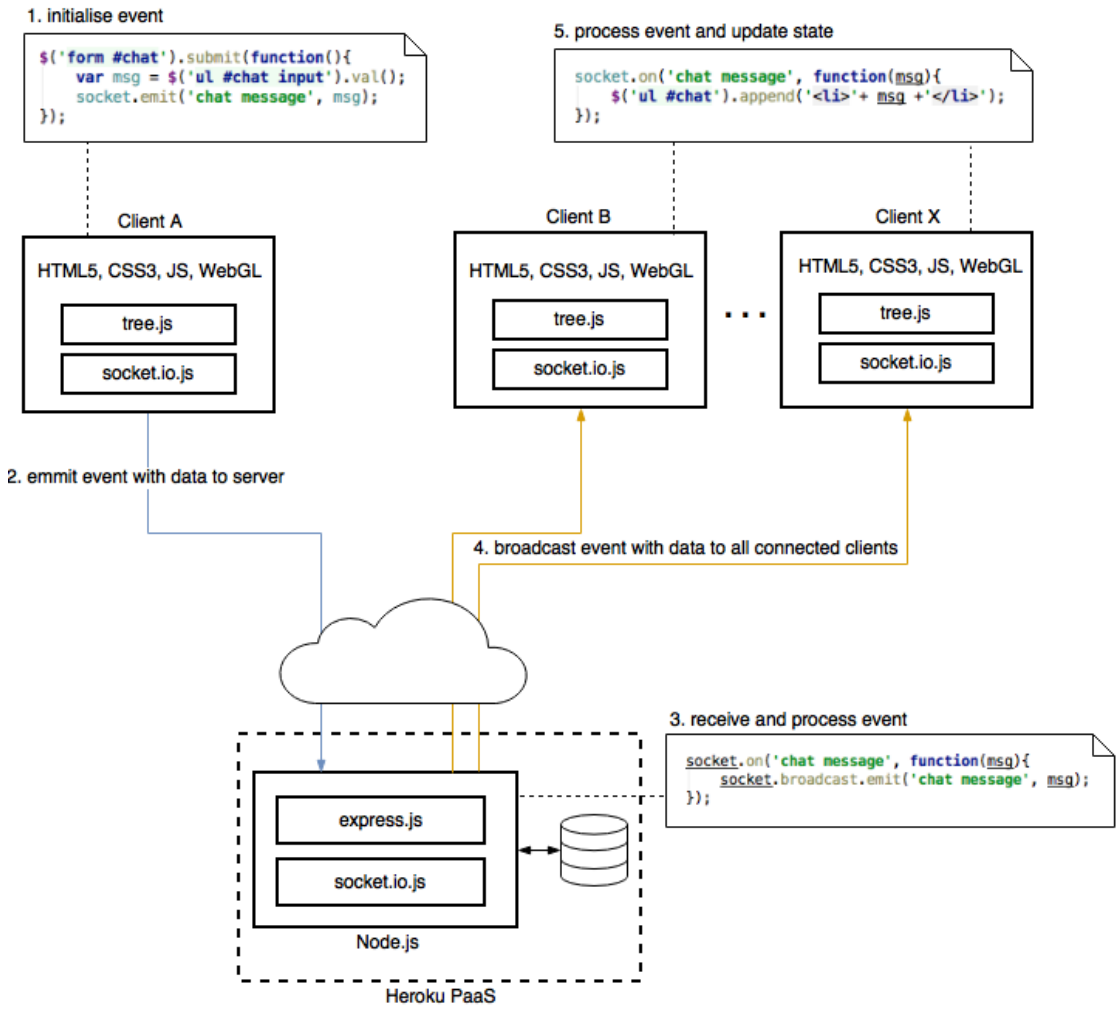

Figure 19: Chat message transmission in cloud schema using socket.io.

Each time a specific event occurs a message is transmitted to server and broadcasted to all connected clients in real-time. The events can concern anything such as notifying everyone that a new user has logged in, sending a chat message or updating the position of 3D objects. Each transmitted message consists of two parts: the event trigger identifier and the specific data to be transferred in JSON format. The following sequence diagram shows a use case when a user updates the position of an UML element (Figure 20).

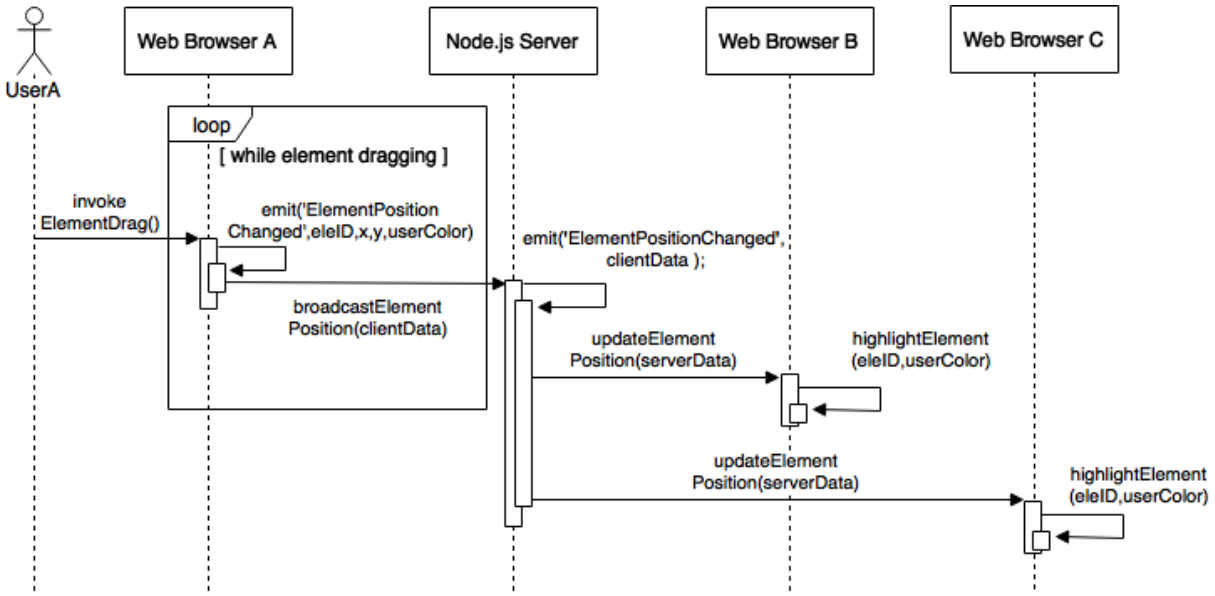

Figure 20: Sequence diagram showing diagram dragging functionality in our prototype.

To enable the dragging functionality of an UML element, JavaScript library jQuery-UI was used. When a user invokes element dragging, the event '*ElementPositionChanged' with* the element's ID, element's new position and the user's color are emitted to the server every 20ms. When the user stops dragging the element the interval timer is cleared and the calling of the function stops. The implementation of this functionality can be seen in the following figure.

```
var socket = io(), sendInterval, user;
...
$( ".element.draggable" ).draggable( {
    start: function( ) {
        var element = $(this);
        sendInterval = setInterval( function () {
            var x = element.css('left'); var y = element.css('top');
            socket.emit('ElementPositionChanged', {diagramID: element.attr("id"), x: x, y: y, userColor:  } );
        }, 20);
    },
    stop: function( ) { clearTimeout( sendInterval ); }
} );
...
```

Figure 21: Implemention of the UML class element drag function.

Each time the server receives the *'ElementPositionChanged'* event from the client socket, the server broadcasts the same event with client's data to all other connected client sockets. This implementation is shown in the following figure.

```
...
socket.on('ElementPositionChanged', broadcastElementPosition(clientData ));

function broadcastElementPosition( clientData ) {
    socket.broadcast.emit('ElementPositionChanged', clientData );
}
...
```

Figure 22: Implementation of the 'ElementPositionChanged' event broadcast.

Each client's socket reacts to *'ElementPositionChanged'* event when received from the server and calls the *updateElementPosition* function which resets the element's position in the user's web browser. Consequently, a function *highlightElement* is called, which highlights the element with the user's color (Figure 23).

```
...
socket.on('ElementPositionChanged', updateElementPosition( serverData) );

function updateElementPosition ( serverData ){
    $("#"+ serverData.eleID).css('left', serverData.x).css('top', serverData.y);
    highlightElement( serverData.eleID, serverData.userColor );
}
...
```

Figure 23: Implementation of the UML class element position update function.

The following figure shows the result of our implemented prototype in which three users are collaborating (Figure 24). We can see two side-by-side screens of a red user on the left and a blue user on the right. Each connected user has a specific color and his mouse pointer with the same color can be seen moving around in real-time. This way every user can see what the other users are currently working on. A blue user is currently dragging an element and a red user has just written a chat message. We have also implemented a technique of showing history of users' actions. Users' actions, such as adding a new method to a class element, are represented by a small user's color circle besides the element he added or edited. The color is fading out as it is further in history.

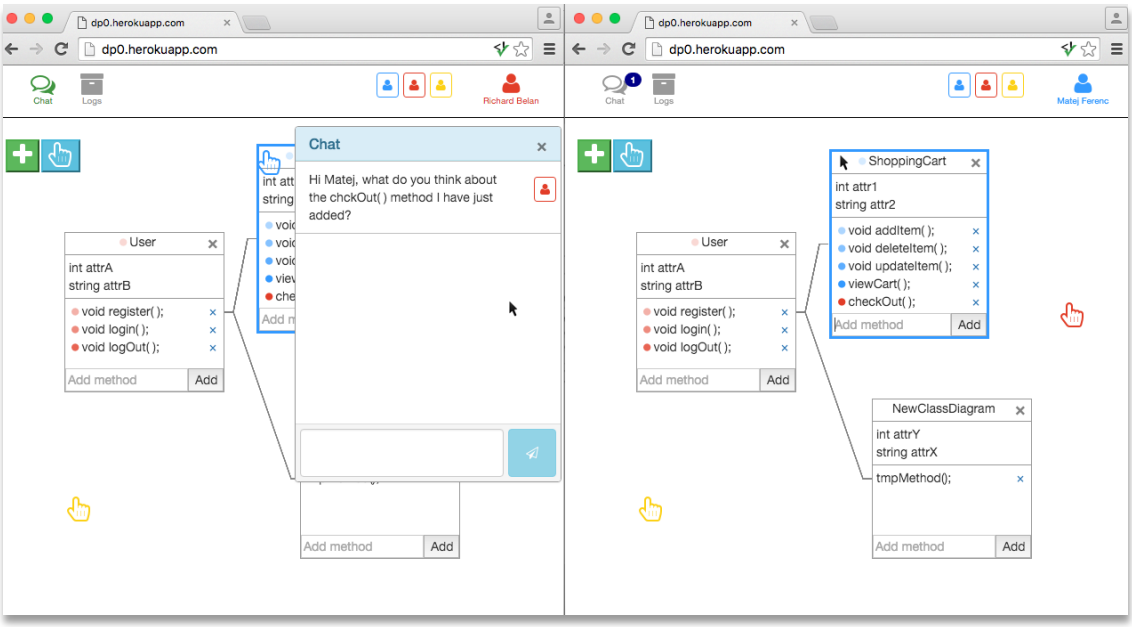

Figure 24: Real-time synchronous collaboration shown in our implemented prototype.

4.2. Second Phase: Implementing the 3D UML Application

Due to relevant reasons (see sub chapter 3.3), we have decided to stop the development of the original 3D UML prototype implemented in C++ and Ogre3D and to implement the 3D UML prototype as a web application instead. Based on the analysis and intensive discussion about the two main web technologies for 3D graphical rendering, we decided to implement our 3D UML prototype using the CSS 3D Transformations. Since, we are only working with 2D UML diagrams visualized on layers in 3D space, we used the CSS 3D Transformations to create the 3D scene and implemented only the layers as CSS 3D objects. This allowed us to use any standard HTML/JS libraries for creating, editing and working with UML diagrams. Specifically, we used Joint.js library, which comes with build in functionality for creating UML class diagrams. This saved us a lot of time and we could focus more on the main topic of the thesis. It is possible to divide the frontend of our 3D UML web application into 3 main modules:

- **3D Operations Module** – This component is responsible for all 3D functionality. Such as initializing and rendering the 3D scene, rotating the camera or the transformation of CSS 3D objects, which in our case are the 3D layers.

- **UML Diagram Operations Module** – This module is responsible for rendering and working with 2D UML class diagrams on 3D layers. It consists of Joint.js and Backbone.js JS libraries which represent the Model and View parts of a MVC architecture.

- **Application Logic and Collaboration Module** – This module acts as a Controller in a MVC architecture. It is responsible for updating the Model and communication with the server where UML models are stored in a database.

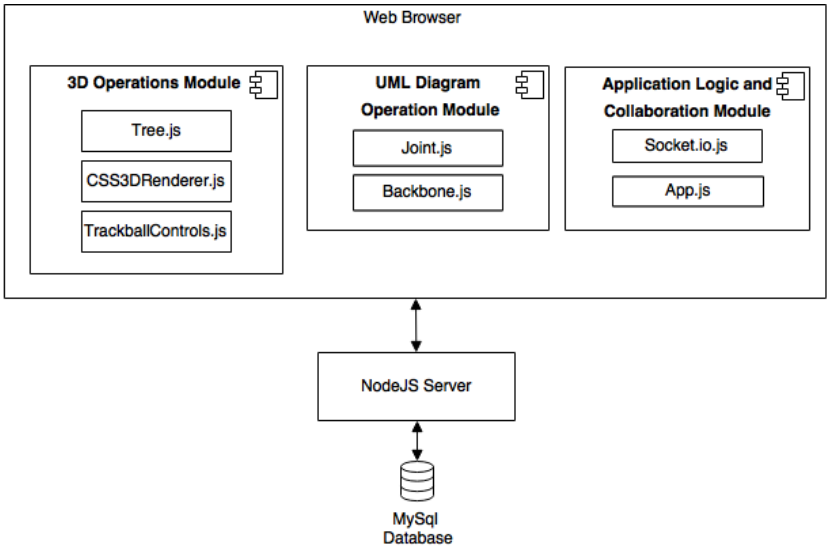

Figure 25: 3D UML application components

The following figure shows the implemented frontend of our 3D UML web application. First, we implemented the 3D scene end enabled a user to add many layers behind each other in 3D space. Next, we implemented basic functions for creating UML class diagram elements such as classes, abstract classes and interfaces, as well as relationships between them such as, association, generalization, implementation or aggregation. This could be done on any selected layer. The user could also rotate the layers and zoom in or out to any specific element to visualize the system from any perspective. In conclusion, it was possible to create and visualize a basic UML class diagram in 3D space. Therefore, we created the base for our designed 3D UML web application component.

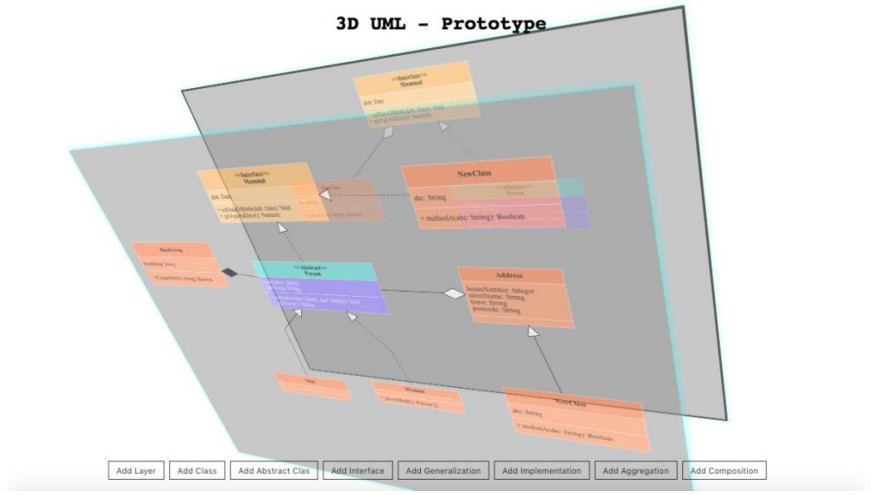

Figure 26: Prototype of 3D UML Web Application component

4.3. Third Phase: Solving the Problem with 3D Arrow

From time perspective, choosing CSS 3D Transformations over WebGL for our 3D UML web application saved us significant amount of time. However, now, the only complex problem we needed to solve was connecting the elements in 3D space. This could be easily implemented using WebGL, since it natively supports this functionality, but other advantages of using CSS3D Transformations far out-weighted this only disadvantage. Therefore, we decided to implement our own solution.

The following pseudo algorithm with a graphical explanation (Figure 27) describes the process of how the position and length of the 3D arrow is being re-calculated during the movement of the HTML element (UML class element). The blue arrow shows the movement direction of *ElementA* and red line represents the 3D arrow. Red line with the label *a'* represents the initial position of the 3D arrow and the red line with label *c''* represents the final position of the 3D arrow.

while element is moving **do**
- **set** a' **to** distance between layers
- **set** b' **to** y-axis distance between ElementA and ElementB
- **set** c' **to** $\sqrt{a'^2 + b'^2}$
- **set** a'' **to** c'
- **set** b'' **to** x-axis distance between ElementA and ElementB
- **set** c'' **to** $\sqrt{a''^2 + b''^2}$
- **set** β' **to** $\sin^{-1} b'/c'$
- **set** β'' **to** $\sin^{-1} b''/c''$
- **set** line's length **to** c''
- rotate line around x-axis β' degrees
- rotate line around y-axis β'' degrees

endwhile

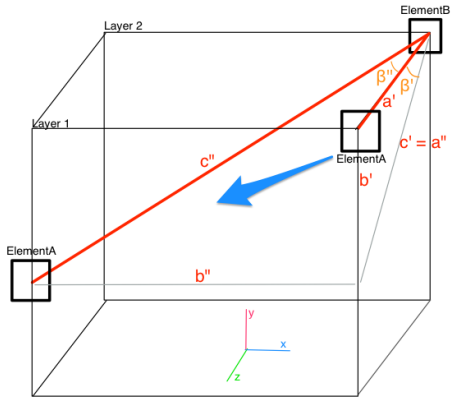

Figure 27: 3D arrow pseudo algorithm with graphical explanation

Using simple goniometric functions, we successfully implemented an algorithm in JavaScript for connecting two standard HTML elements located on two 3D layers using a 2D line (Figure 28). The designed algorithm is simple and fast enough to allow real-time line re-rendering while user simultaneously moves both elements in any direction within a layer.

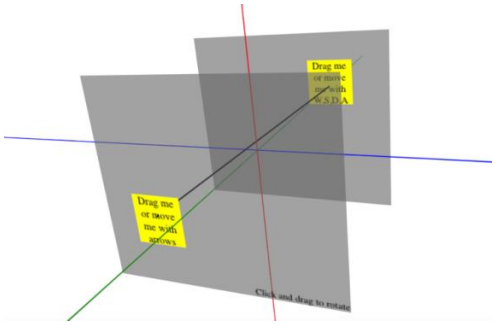

Figure 28: The implementation of 3D arrow connecting two HTML elements

However, there was still one minor problem. Since the line was only a 2D HTML element and was only 0 pixels thick from one side, it would not be visible, when looking at it from certain camera perspectives. To fix this problem, we created another 2D line, but rotated it over y-axis by 90 degrees which created a 3D effect. Later, besides optimizing the algorithm, we have also wrapped it within a function, that could be called with two arbitrary elements as parameters. The 3D arrow is now automatically created between the source and target element and re-rendered whenever an element is moved.

4.4. Fourth Phase: Combining the Previous Phases and Adding Awareness

By completing the three previous phases, all was prepared for completing the designed 3D UML web application. We used the first 3D UML prototype and integrated the collaborative real-time synchronous functionality from the first phase. Next, we added the 3D arrow to enable connecting UML class elements between layers and finally, we have enhanced the prototype with many additional features to increase users' awareness or to simplify the process of UML modeling in 3D. All of these features are summarized in the sub chapter 4.7. The final implementation of the prototype can be seen in the following figure.

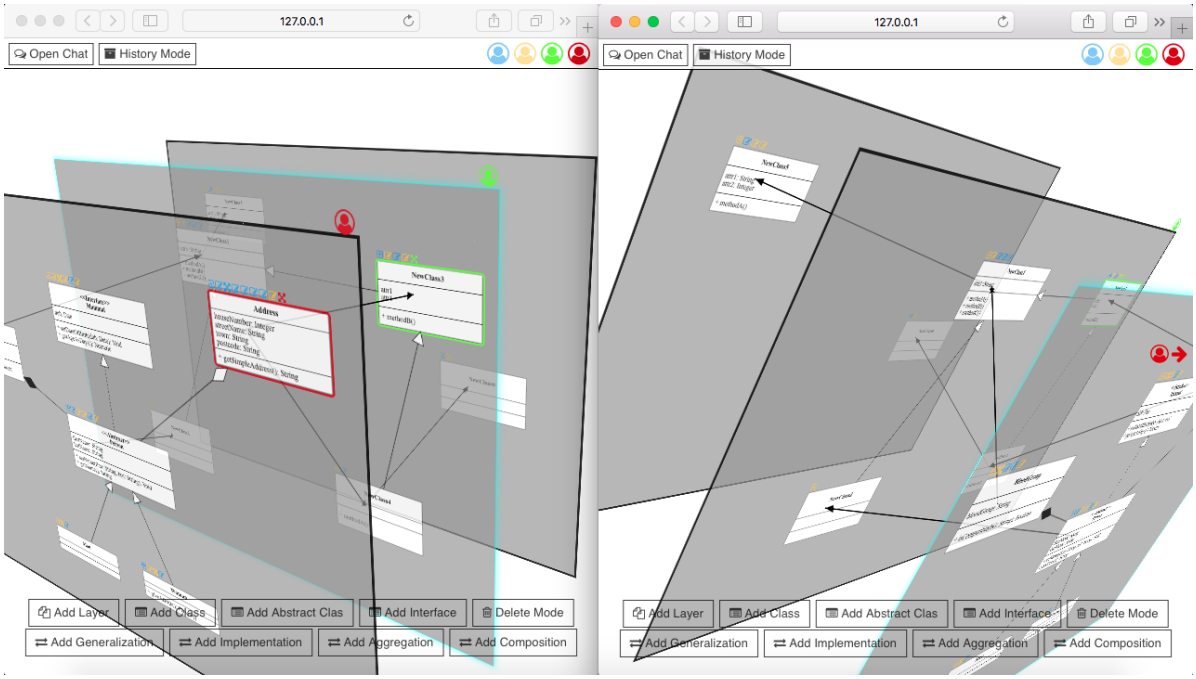

Figure 29. 3D UML web application with collaborative functionality and 3D arrow

4.5. Fifth Phase: Integration with Enterprise Architect

3D UML Collaboration EA Add-in is the last component of our designed architecture that constructs our method for collaborative system modeling and visualisation using 3D UML. The main idea for integrating EA into our collaborative method was to allow users to also normally work in a powerful widely used tool for system modeling they have already adopted and also utilize EA's features for complex system modeling. Add-in would also enhance EA with real-time collaborative features that would increase users' awareness. However, to enable real-time 3D visualization of EA's 2D models a custom add-in was required. The EA add-in was implemented using C# and .NET framework. To enable communication with the server, the designed REST API module was implemented on the server. It serves as a RESTful web service for the EA add-in. To make RESTful request from the EA add-in to our application server we used a .NET REST API, called RestSharp[11].

Currently, the EA add-in enables a user to establish a connection with our server by logging in. All other users using the 3D UML web applications are instantly notified. Once the user connects to our server, he can collaboratively model the same system as he would in our 3D UML web application. However, so far we have implemented the functionality for creating UML class diagrams as well as modifying their title. We have proved, that real-time synchronous collaboration and synchronization with EA was possible, but in the future it would be necessary to finish the implementation of other events. Currently, whenever a user creates or modifies an UML class in EA the change is instantly synchronized with our server and the change is broadcasted to all other connected clients. The following figure shows our 3D UML Collaboration EA Add-in and how a user would connect to our server.

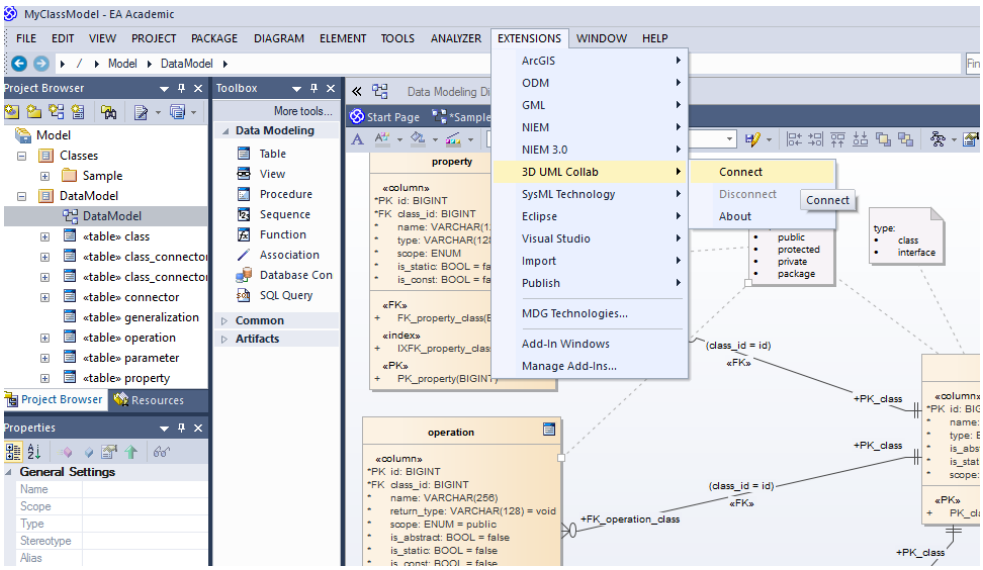

Figure 30: EA 3D UML Collaboration Add-in

[11] http://restsharp.org

4.6. Sixth Phase: WebVR Experiment

In this phase a prototype for system visualization in virtual reality was implemented as an addition to our designed method. The prototype was also implemented as a web application. It was implemented mainly in JavaScript, thus enabling the use of Socket.io for real-time synchronous collaboration. This enabled the application to connect to the same Node.js server as our 3D UML application and listen for and emit the same events as the 3D UML application. Therefore, no additional changes had to be made to either the application server or the 3D UML application. It was only required to react to the events server broadcasted to all users automatically and implement their visualization in virtual reality.

Enabling VR experience directly in a web browser was possible thanks to an open standard called WebVR. However, WebVR currently enables to only view web applications in VR that are implemented in WebGL. Therefore, this prototype had to be implemented in WebGL and not by using CSS 3D transformations which we used for our 3D UML application. However, the purpose of implementing this prototype was also to experiment with a different approach to 3D system visualization, so we decided to change the user's perspective. We positioned the layers around the user in a circle, thus positioning the user directly in the center of the scene. The user would now have to look around to view the whole UML model, but on the other hand, the layers were not stacked behind each other, so all UML elements were visible and easily accessible from all perspectives. The following figure shows the implemented WebVR prototype. The first two images show the layers positioned around the user while he is looking around and the third image shows the web application in VR mode.

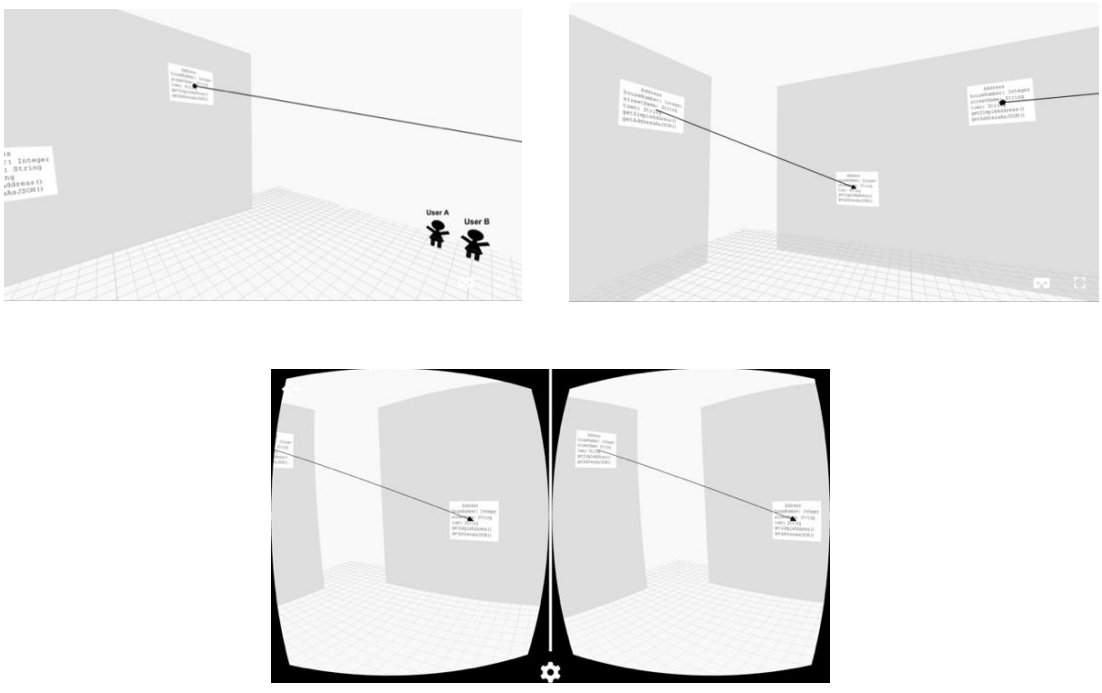

Figure 31: Implemented WebVR prototype

Due to the fact that this prototype was implemented only as an additional experiment, we have only implemented the functionality for visualizing all active users and the functionality for the UML element position update. This was only implemented to prove that the real-time synchronous communication is fully compatible with the 3D UML application and that collaborative system modeling is also possible in virtual reality.

4.7. Summary of All Implemented Collaborative Awareness Features

Before all collaborative awareness features are summarized, it is important to mention again that, all of the 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 an UML element all other collaborators can see what element is being moved and by who in real time. The following sub chapters briefly summarize all of the implemented features.

4.7.1. Login Notification

The first implemented awareness feature was 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 (Figure 32). It is important in multi-user workspace to let others know if a collaborator is online and is prepared to work.

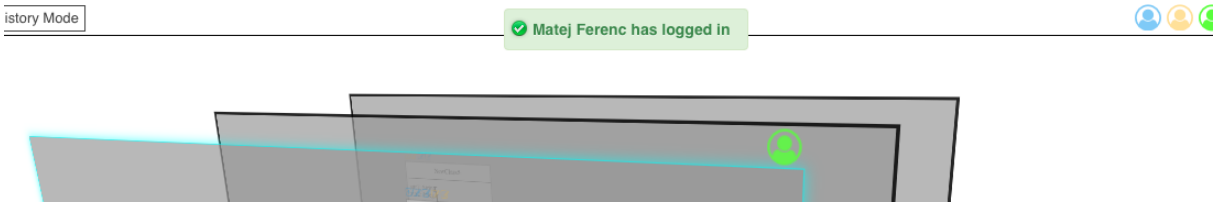

Figure 32: Login notification

4.7.2. Presence Awareness Features

These 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 following figure. The collaborators are always aware of all project participants (Figure 33 - 1), on what layer they are working (Figure 33 - 2), on what specific element they are working (Figure 33 - 3), end even of the actions of others if they are working outside of their view (Figure 33 - 4).

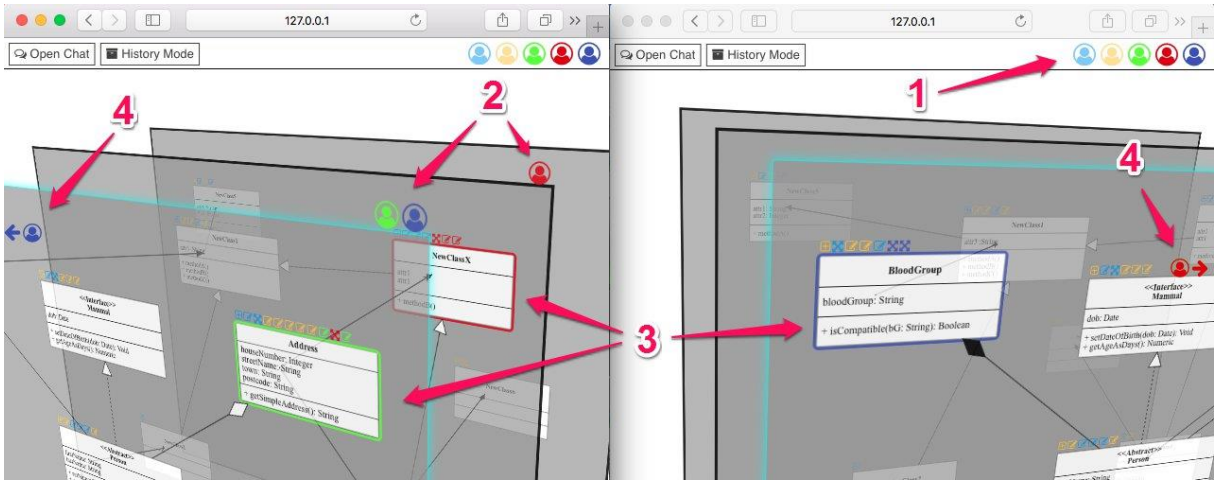

Figure 33: Implemented presence awareness features.

4.7.3. User Action History

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 have firstly implemented this feature by placing a small flag beside the element that has been changed. The flag has the same color as the user who modified it. The flag also 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 (Figure 34).

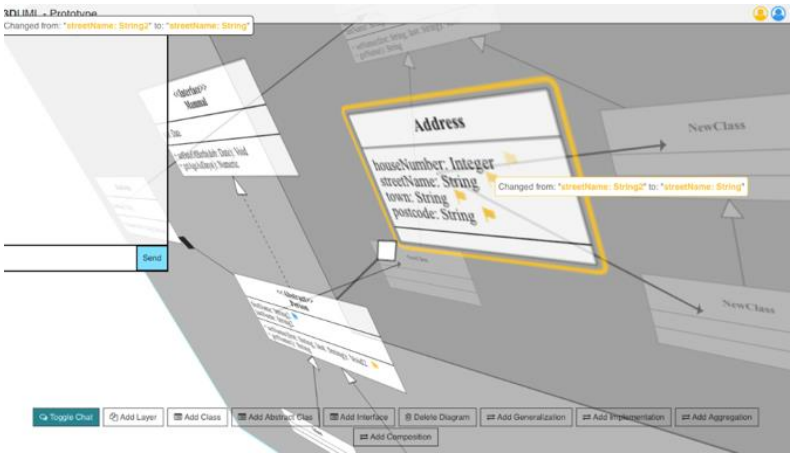

Figure 34: Impemented user action history feature.

However, this later proved to be inefficient, since it was very complicated to see the small flags if they were distributed around the whole project.

4.7.4. UML Class Action History

The previous feature focuses on the history of the users' actions. On the other hand, this feature rather focuses on the history of each UML class element. It is still possible to see who has made the change, when the change was made and what has been changed, but it visualized differently. The icons are also the same color as the user who made the change, but in this approach, a different icon is used for each type of change. The icons are placed on top of the UML element. This enables a collaborator to find the change he was looking for more efficiently. A collaborator can also immediately visualize more detail about the change, by placing his cursor above the element.

For example, the following figure 35 shows a user viewing the most recent change of an *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 visual assistance to quickly find what exactly has been changed. We stroked out and used red color to visualize what has been removed and used green color to visualize what has been added.

Similarly, the figure 36 shows a user viewing a different type of change. In this example a user can see who has moved a UML class and when it has been moved. The red and green colors were also used to illustrate the old and new positon of the UML class.

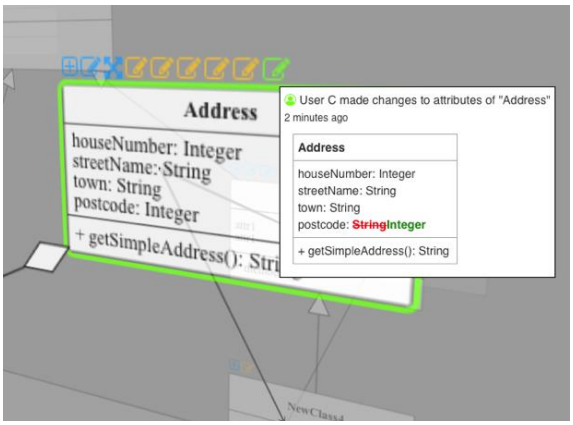

Figure 35: UML class action history – attribute modfication.

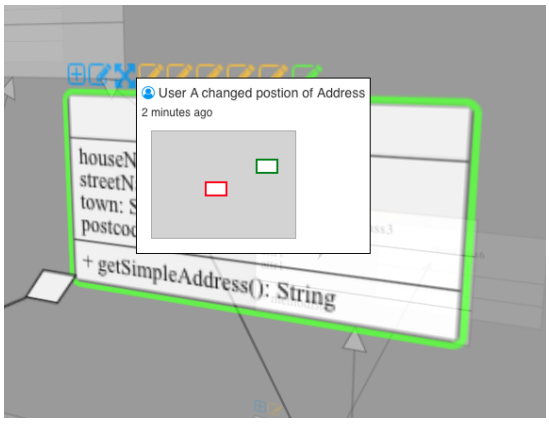

Figure 36: UML class action history – position change.

4.7.5. Project Action History

This feature enables the users to see every action of all collaborators and simultaneously see the state of the whole project in a specific time in history. A collaborator can simply visualize 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 (Figure 37 - 1). Each step of the slider represents one user's action in history. The collaborator can see more

detail about the action in the history window (Figure 37 - 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 (Figure 37 - 3).

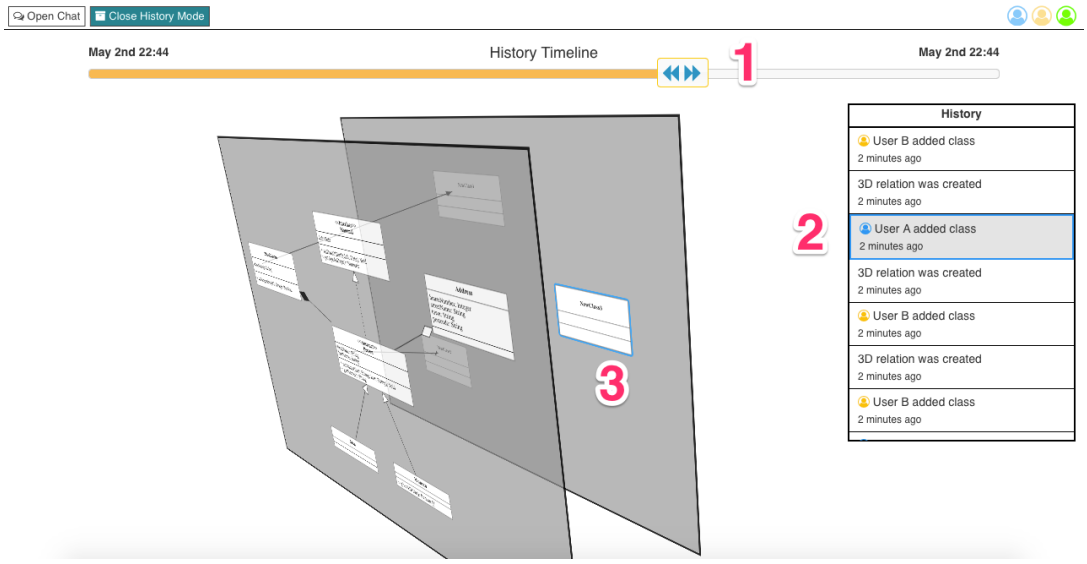

Figure 37: Implemented project history timeline feature.

The following figure 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.

Figure 38: History timline functionalty example.

We have implemented this functionality by logging and creating an undo action for every action a user made. 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 action. Therefore, every client always has a local and up to date copy of the entire history of the project. When a user enters history mode, simply the collaborative functionality is disabled and the local history actions are executed as standard actions except the events are not emitted to other clients. The local execution of history actions can be seen on the following diagram, together with the implementation of the *executeDoAction* and *executeUndoAction* functions.

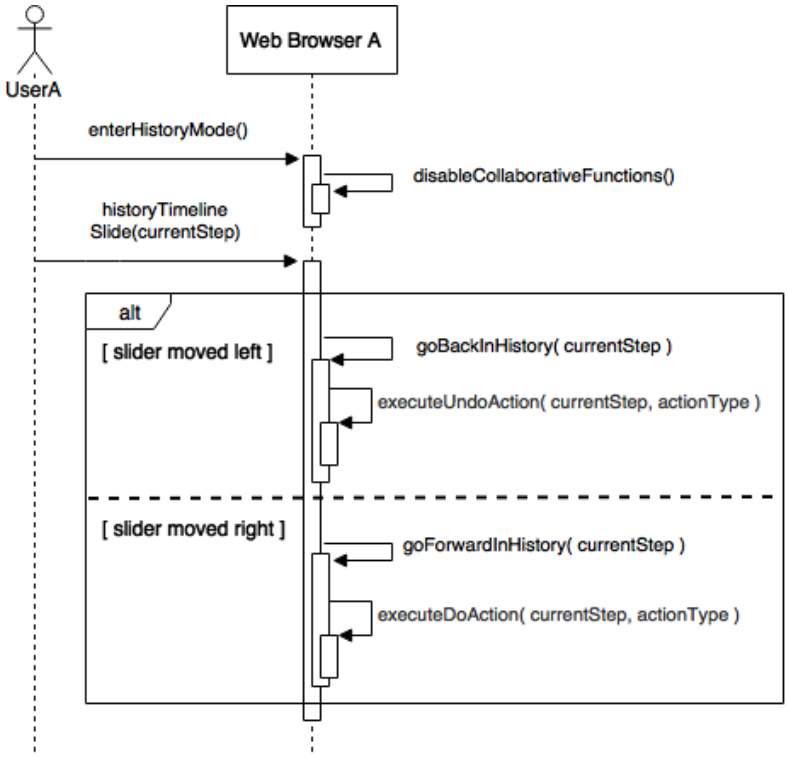

Figure 39: Sequence diagram showing the process of history timeline functionality.

```
function executeUndoAction ( currentStep, actionType ){
    ...
    if(actionType == 'elementNameChanged'){

        // Update UML class element with old name
        var elementID = historyActions[i].undo.data.elementID;
        var newName = historyActions[i].undo.data.name;
        updateElementName(elementID, newName);
        // Higlight the element with user's color
        var user = historyActions[i].user;
        setCellHiglight( elementID, user.color );

    }
    ...
}
```

Figure 40: Implementation of the “executeUndoAction” function.

```
function executeDoAction ( currentStep, actionType ){
    ...
    if(actionType == 'elementNameChanged'){

        // Update UML class element with new name
        var elementID = historyActions[i].do.data.elementID;
        var newName = historyActions[i].do.data.name;
        updateElementName(elementID, newName);
        // Higlight the element with user's color
        var user = historyActions[i].user;
        setCellHiglight( elementID, user.color );

    }
    ...
}
```

Figure 41 Implementation of the "executeDoAction" function.

4.7.6. Chat

Even though, communication is an important part in collaboration, we have implemented all of the above features with the purpose to minimize the need for it. 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. The following figure shows our implementation of chat and an example of a chat communication between two collaborators.

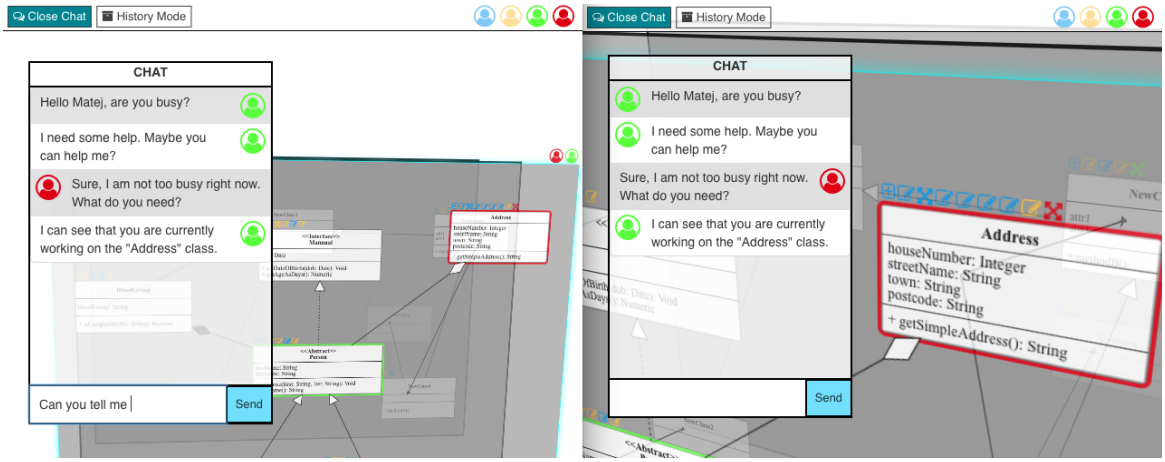

Figure 42: Chat communication example.

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 43).

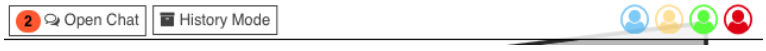

Figure 43: New chat message notification.

4.8. Solving the Element Accessibility Problem in 3D UML Modeling

The first problem in 3D UML modeling we came across was how to visualize other collaborators' actions and their locations if they are working out of the collaborator's view. The solution to this problem was already presented while designing the UI for maximum awareness (Subchapter 3.43.4.1). However, another problem occurred when a collaborator needs to work on a layer that is located behind another layer. Accessing the UML elements on that layer was only possible from the sides which made the UML modeling complicated. The problem was solved by the implementation of edit mode. Firstly, a collaborator double clicks the desired layer (Figure 44 - Step 1). Consequently, the desired layer separates from other layers (Figure 44 - Step 2), the camera is moved in front of the desired layer and zoomed so the layer becomes full screen (Figure 44 - Step 3). This allows the user to work the same way as he would in standard 2D environment. When the layer is double clicked again the camera and the other layers are moved back to their original position and it is again possible to visualize the system in 3D (Figure 44 - Step 4). This is a simple and efficient way of how this problem could be eliminated in any 3D UML modeling workspace.

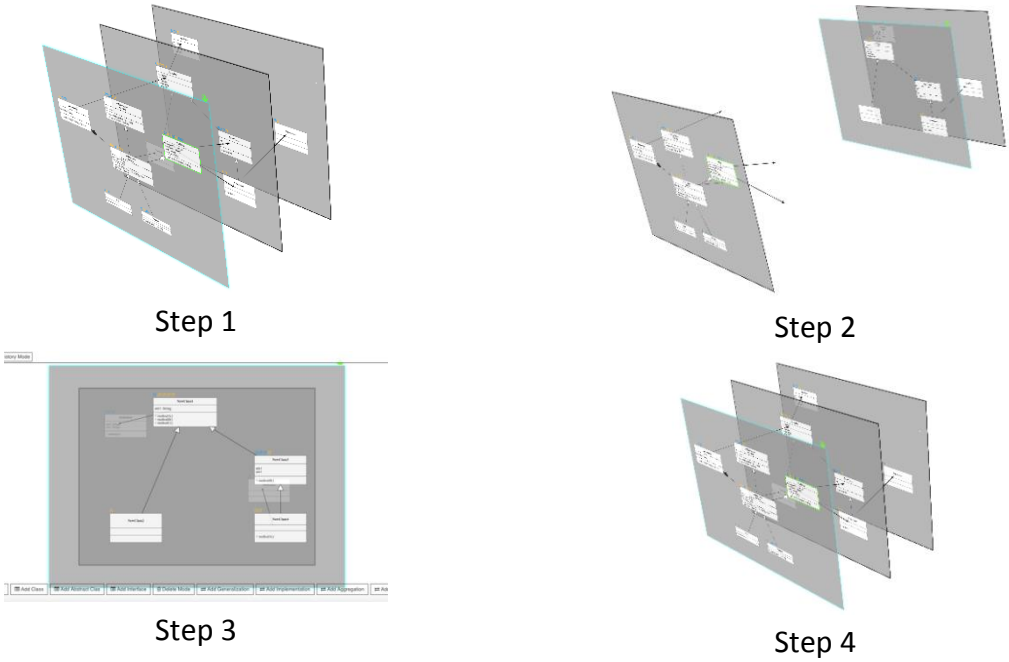

Figure 44: Layer edit mode.

5. Conclusion

5.1. General Thesis Summary

Firstly, the analysis was created covering required aspects that were needed to successfully design and implement a method for collaborative modeling and visualisation of software systems using 3D UML.

The first subchapter introduced collaboration and described how it should be used in groupware design. Next, different types of collaborations and their advantages and disadvantages were presented. As a result we have come to conclusion that users' awareness of others and his own actions is very important in collaboration and that synchronous real-time collaboration has many advantages compared to asynchronous collaboration.

Another subchapter was dedicated to 3D UML system modeling and visualisation. In this subchapter UML was briefly introduced, system modeling using 3D UML was analyzed and some 3D UML visualisation methods were presented. Based on this subchapter, we summarized some limitations of 3D system modeling and how they could be eliminated. Next, we have also analyzed and considered the use of force directed algorithms to automatically rearrange and layout UML diagrams. We proposed a few approaches how the force directed algorithms could be applied in collaborative environment.

Lastly, some recent commercial tools for system modeling were analyzed with the focus on their collaborative features. We considered implementing many collaborative features that have been adopted by the public and that could be improved by our method.

Based on the analysis we created a list of high level system requirements. To meet these requirements, we designed a high level system architecture that proposed a new approach to collaborative 3D UML system modeling and visualisation. We decided to design and implement a solution where users could collaboratively model and visualize systems using 3D UML but also utilize already powerful tools for UML modeling such as EA. We analyzed and decided on the best suitable technology, in which our tool could be implemented. Consequently, we designed the UI together with many awareness features that could be used in collaborative 3D system modeling.

We have successfully implemented a prototype for every component, that our designed system architecture was composed of. The implementation consisted of several phases. Firstly, we designed and implemented a simple prototype for real-time synchronous collaboration where multiple users could connect and collaborate in real-time to create a simple UML class diagram. This was a required first phase to test the chosen real-time

collaboration technology. The next phase was to create a web application for 3D UML system modeling and visualisation. Next, we needed to solve the problem of connecting the UML diagrams in 3D space so we implemented our own solution. Additionally, we have implemented more user awareness features, such as the UML class action history, the history timeline or the out of view user presence awareness icon. This concluded the implementation of our main tool for collaborative 3D UML system modeling. The last phase, was to integrate EA into our 3D UML Application, to complete our designed method for collaborative system modeling and visualisation using 3D UML. We implemented a simple EA add-in, which would support real-time collaborative functionality between EA and the 3D UML web application. This concluded the implementation of every component of our designed method. However, we have additionally implemented another prototype to experience with UML modeling in virtual reality and also proposed some advantages and new approaches to collaborative 3D UML software system modeling.

5.2. The Summary of Proposed Benefits

Based on the analysis, design and implementation phases, it is possible to conclude and propose the following benefits of our method for collaborative system modeling using 3D UML.

Faster and More Efficient 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. The collaborators are always aware of each other's actions. The need for communication is minimalized and redundant or duplicate work is eliminated. All of these benefits save a lot of time, and therefore provide faster and more efficient system modeling.

Simplification of Communication
The always visible user action icons minimize the need for communication. They are used as conversational artefacts or visual evidence to replace verbal communication and therefore the collaborators are not forced to ask someone about activities or changes made to the UML model by other users.

Coordination of Action
All collaborators are always aware of each other's actions, where they are working and what they are working on. A collaborator can easily see which tasks are currently in progress, and which tasks are currently being done by others. Therefore, a collaborator can choose his next task more easily. This improves the awareness in coordination of action and prevents work redundancy, coordination and division of labor.

Anticipation

Also, 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.

Assistance

A quick access to basic chat provides a simple way for a collaborator to ask others for assistance. If someone is not busy and knows the solution, he can instantly take over user's task and provide assistance immediately.

Lucidity

To visualize UML diagrams in standard 2D environment requires a significantly larger working area compared to 3D environment. By dividing the UML diagrams into layers and by stacking them behind each other, the space needed to visualize the whole system is reduced. Virtual reality also enables to place the layers around the collaborator, and therefore provides better visibility and accessibility to the UML diagrams from all perspectives. These benefits deliver better lucidity and readability of system models.

Enjoyable and Healthier Labor

Collaborative modeling and visualization of systems in multidimensional workspace or in virtual reality proposes a new and innovative approach to system modeling. Collaborators can find this approach more enjoyable. In virtual reality the collaborators can stand, move and look around, which is a lot healthier than just sitting in front of the computer.

5.3. Evaluation and Feedback

The above proposed benefits are only our assumption based on the research and results of this thesis. These benefits still need to be evaluated and we need to see if they could be adopted by the public. We have designed and implemented various user awareness features and it is important to evaluate them and see if they can be intuitively used and adopted by software engineers in collaborative system modeling. However, to objectively evaluate our method and the proposed benefits, an evaluation by professional software engineers is required. Due to the complexity and the amount of implementation that was required to complete all of the prototypes and features, we did not have enough time to acquire such conditions for evaluation. Currently, my supervisor, Ing. Ivan Polášek, PhD., is looking for an industrial project which would provide a relevant evaluation and possibility for future development of our collaborative 3D UML modeling method.

However, we asked several software engineering students for their feedback anyway, thus providing a simple evaluation. We have asked them to focus on the most common tasks in collaborative software modeling:

- Create a complete UML model from scratch
- Modify an existing model
- Search for a specific change in an existing model

We have also asked them to compare our method of 3D UML modeling with UML modeling tools they use on a regular basis and support collaboration, such as Enterprise Architect, LucidChart or draw.io. We have asked them to evaluate and focus on the above mentioned benefits and mainly on the following criteria.

- The overall time needed to complete a task
- The amount of communication needed with others
- The awareness of others
- The simplicity and efficiency of work
- Their overall feeling

Based on the feedback, we acquired various interesting opinions. Firstly, we have to conclusion that all of the participants found 3D modeling more enjoyable and that they would consider using it for real projects in the future. However, they still mentioned various disadvantages or missing functionality.

The participants stated that to complete a simple UML class diagram was much faster than in other tools they regularly used. However, this is not an objectively evaluated conclusion. For example, EA is a professional tool for complex UML systems and to create an UML class with attributes and methods, it is required to set many constraints, data types, method return types, method parameters, access levels, visibility and so on. In our prototype, validation against UML class meta-model was considered, but it was not fully implemented. Therefore, a method or an attribute is added only by simple input field without validation, thus saving a lot of time. However, this would have to be changed in the future.

The next main conclusion is that most of the participants found real-time synchronous evaluation very helpful. They were aware of other actions and they intuitively chose their next task. However, on the other hand, some participants told us, that they were really distracted by the activities of other users and in more complex system, where deep thought is required, this would not be ideal. They proposed that it could be beneficial, but it should be possible to disable this function when not needed.

5.4. Future Work

Based on the previous subchapter, the first and important next step is further evaluation of the proposed benefits of this thesis. The evaluation could prove many benefits and our method for collaborative software modeling using 3D UML could provide more efficient and faster software modeling and visualization approach. Furthermore, the validation against UML metamodel was not fully implemented, and therefore implementing this functionality is another important step towards standardization of UML models.

In this thesis we have also proposed few approaches to how force directed algorithms could be used to automatically rearrange and layout UML class diagrams, in order to provide various benefits in collaborative system modeling. Implementing and evaluating these approaches could be another interesting addition to this method. In general, many more collaborative features could be implemented, such as the automatic revision to specify version in case of unwanted change or the asynchronous collaboration with automatic merging and synchronization of UML models.

Lastly, experimenting with software modeling in virtual reality or even augmented reality could provide many other benefits to software modeling in the future. It is possible to completely transform our prototype into augmented or virtual reality and evaluate its benefits.

6. Resume in Slovak Language

6.1. Úvod a motivácia

Vývoj zložitých a rozsiahlych softvérových systémov predstavuje náročný a komplikovaný proces, ktorý si vyžaduje spoluprácu mnohých špecialistov. Vzhľadom k tomu bolo hlavnou motiváciou tejto práce navrhnúť nový a inovatívny spôsob, ktorý by zjednodušil komplexnosť UML modelov a zároveň by zvýšil produktivitu práce pri kolaboratívnom modelovaní systému.

6.2. Analýza kolaborácie a 3D UML

Kolaboráciu je možné definovať ako súhru medzi koordináciou, komunikáciou a kooperáciou [1]. Dôležitým faktorom pri kolaborácii je uvedomenie si prítomnosti kolaborantov a ich činnosti. Tento faktor ovplyvňuje a je ovplyvňovaný všetkými tromi aspektmi kolaborácie [2]. Kolaboranti si pri práci musia byť vedomí činností iných kolaborantov, za účelom lepšieho porozumenia kontextu vlastných aktivít [3]. Musia vedieť kto je momentálne aktívny, na čom pracuje, kde pracuje, aké zmeny boli vykonané, kto ich vykonal a kedy [4]. Tento fakt je potrebné brať do úvahy pri analýze a návrhu softvéru určeného na spoluprácu. Uvedomenie si prítomnosti prináša mnoho výhod okrem iného aj elimináciu zbytočnej komunikácie, duplicitnej práce a efektívnejšie prerozdelenie úloh.

Vo všeobecnosti poznáme dva typy kolaborácie:

- Synchrónna kolaborácia v reálnom čase – pracuje sa na jednej spoločnej verzii projektu a všetky zmeny sú okamžite viditeľné všetkými kolaborantmi.
- Asynchrónna kolaborácia – pracuje sa na viacerých verziách spoločného projektu a zmeny je potrebné zlúčiť do jednej finálnej verzie.

Pri synchrónnej kolaborácii môžu nastať konflikty, keď kolaboranti editujú rovnaký element v rovnakom čase a pri asynchrónnej kolaborácii môže nastať konflikt pri zlučovaní verzií, v ktorých boli rôzne upravené rovnaké časti projektu. Tieto konflikty je možné riešiť nasledovnými spôsobmi:

- manuálne,
- pomocou zablokovania aktuálne upravovanej časti projektu pre ostatných kolaborantov,
- okamžite v reálnom čase pomocou algoritmu (napr. Operational Transformation) [8].

Kolaborácia je dôležitá pre efektívnejšiu prácu, avšak nerieši problém s komplexnosťou zložitých a rozsiahlych softvérových systémov. Jedným z možných spôsobov riešenia tohto

problému je modelovanie systémov v trojrozmernom priestore [15, 16, 17, 14, 22]. Vo všeobecnosti je možné vizualizovať systém v 3D dvomi spôsobmi:

- vizualizácia pomocou 3D objektov
- vizualizácia 2D UML diagramov na viacerých vrstvách v 3D

UML diagramy zobrazené v 2D zaberajú omnoho väčšiu plochu ako diagramy zobrazené v 3D. Umiestnenie diagramov do vrstiev značne znižuje potrebnú plochu na zobrazenie systému. Zároveň umiestnenie diagramov do vrstiev umožní zobraziť aj vzťahy nielen medzi jednotlivými prvkami UML diagramu, ale aj medzi rôznymi typmi UML diagramov. Vizualizovanie týchto vzťahov je omnoho jednoduchšie v 3D. V 3D priestore je taktiež možné zobraziť väčšie množstvo objektov ako aj vzťahov medzi nimi [12].

6.3. Návrh riešenia

V prvej fáze sme navrhli architektúru systému na základe systémových požiadaviek, ktoré vyplývajú z analýzy. Vychádzali sme z predpokladu, že uvedomenie si prítomnosti predstavuje najdôležitejší aspekt pri kolaborácii. Vzhľadom na to sme navrhli architektúru, ktorá umožňuje synchrónne kolaboratívne modelovanie systému v 3D v reálnom čase, zároveň umožňuje používateľom využívať existujúce profesionálne nástroje ako napríklad Enterprise Architect a zachováva štandardizáciu UML modelov, takže ich zdieľanie s inými modelovacími nástrojmi je jednoduché.

Najjednoduchším a najefektívnejším riešením bolo vytvoriť webovú aplikáciu. Zistili sme, že pre umožnenie synchrónnej kolaborácie je najideálnejšia technológia WebSocket-ov.

Na prácu s 3D objektami sme sa rozhodovali medzi dvomi technológiami: WebGL a CSS 3D Transformáciami. Keďže v našom prípade sme sa rozhodli pracovať iba s jednoduchými 2D objektami (UML diagramami) v 3D vrstvách, CSS 3D Transformácie bola postačujúca voľba technológie. Táto voľba technológie nám taktiež umožnila využiť už existujúce HTML/JS knižnice na prácu s UML diagramami, čo nám ušetrilo veľa času.

Ďalej sme navrhli rôzne prvky uvedomenia si prítomnosti, ako napríklad spôsob vizualizácie histórie zmien nad každou triedou, spôsob zobrazovania histórie činností kolaborantov, prvky, ktoré vizualizujú kde sa kolaborant práve nachádza a zároveň na akom konkrétnom objekte momentálne pracuje.

Okrem iného sme navrhli kritéria ako pomocou force-directed algoritmov automaticky preusporiadať prvky UML diagramov: podľa času vykonania zmeny, podľa kolaboranta, podľa typu zmeny alebo ich kombináciou.

6.4. Implementácia prototypov

V prvej fáze sme implementovali prototyp s jediným cieľom a to overiť vybranú technológiu na synchrónnu kolaboráciu. Tento prototyp sme implementovali ako webovú aplikáciu v JavaScripte. Na zabezpečenie synchrónnej komunikácie medzi klientom a serverom bola využitá technológia WebSockets s využitím knižnice Socket.io. Server bol implementovaný pomocou Node.js.

V druhej fáze sme implementovali samotnú webovú aplikáciu na modelovanie systému v 3D. Tento prototyp vizualizuje systém pomocou vrstiev v trojrozmernom priestore, na ktorých sú umiestnené 2D UML diagramy. Keďže pracujeme iba s dvojrozmernými diagramami, prototyp sme implementovali pomocou CSS 3D Transformácií. Keďže technológie ako WebGL sú zamerané skôr na prácu s komplexnými 3D objektami. Tento fakt nám umožnil využiť hociktorú štandardizovanú HTML/JS knižnicu na prácu s 2D UML diagramami. Konkrétne sme použili knižnicu Joint.js, ktorá umožňuje modelovanie UML class diagramov.

Výber technológie CSS 3D Transformácií nám ušetril podstatne veľa času, avšak vyskytol sa problém spájania elementov v 3D priestore. Technológia WebGL túto funkcionalitu umožňuje, ale výhody CSS 3D Transformácií ďaleko prevýšili WebGL. V dôsledku toho sme sa v tretej fáze rozhodli implementovať vlastné riešenie. Pomocou vlastného algoritmu s využitím goniometrických funkcií sa nám podarilo pomocou JavaScriptu implementovať vlastné riešenie na spojenie dvoch HTML elementov nachádzajúcich sa na dvoch rôznych vrstvách v 3D. Navrhnutý algoritmus je dosť jednoduchý a rýchly na to, aby umožnil prekresľovanie čiar v reálnom čase zatiaľ čo užívateľ simultánne pohybuje oboma elementmi rôznymi smermi v rámci vrstvy.

V ďalšej fáze sme zlúčili všetky tri prototypy a vznikla tak samotná webová aplikácia, ktorá umožňuje synchrónnu kolaboratívne modelovanie systému v 3D. Následne sme implementovali rôzne dodatočné prvky a funkcie, ktoré zvyšujú uvedomenie si prítomnosti a zefektívňujú proces kolaboratívneho modelovania v 3D, ako napríklad vizualizácia celej histórie projektu pomocou posúvania sa v čase na časovej osi.

V poslednej fáze sme implementovali aj EA Add-in, ktorý rozširuje EA o synchrónnu kolaboráciu v reálnom čase a zároveň umožňuje vizualizovať UML model v implementovanej webovej aplikácii. EA Add-in sme implementovali v C# s použitím .NET framework. Na serveri bolo implementované REST API, ktoré odpovedá na RESTové volania z EA.

Navyše sme implementovali WebVR prototyp za účelom experimentovania s vizualizáciou a modelovaním systému vo virtuálnej realite. Táto aplikácia bola implementovaná pomocou WebGL ako webová aplikácia, vďaka čomu sme mohli taktiež využiť knižnicu Socket.io na synchrónnu komunikáciu. Jednoducho bolo možné napojiť sa na existujúci Node.js server a bolo potrebné iba reagovať na udalosti rozposielané serverom všetkým užívateľom a implementovať

vizualizáciu týchto udalostí vo virtuálnej realite. Vrstvy sme rozmiestnili do kruhu okolo užívateľa, ktorý sa síce musí pozerať okolo seba, aby videl celý UML diagram, ale na druhej strane vrstvy nie sú umiestnené za sebou, takže UML prvky sú viditeľné a ľahko dostupné.

6.5. Záver

Navrhli sme spôsob kolaboratívneho modelovania systému v 3D. Synchrónna kolaborácia umožňuje kolaborantom pracovať na jednej spoločnej verzii UML modelu v reálnom čase, čím sa eliminuje potreba zlučovania jednotlivých verzií UML modelov. Navrhli sme rôzne vizuálne prvky a funkcie, ktoré zvyšujú uvedomenie si prítomnosti kolaborantov a ich činností. Tieto prvky majú konverzačný charakter alebo poskytujú vizuálne pomôcky, ktorými sa minimalizuje potreba nadbytočnej komunikácie. Zároveň sa kolaborant môže rýchlejšie rozhodnúť pre jeho nasledujúcu činnosť, keďže vidí na čom pracujú ostatní kolaboranti a vie predvídať ich kroky. Vyššie uvedené prvky uvedomenia ako aj synchrónna kolaborácia taktiež eliminujú nadbytočnú alebo duplicitnú prácu, šetria čas, čo umožňuje kolaborantom rýchlejšie a efektívnejšie modelovať systém. V budúcnosti je potrebná podrobná evaluácia implementovaného spôsobu modelovania, za účelom overenia intuitívnej využiteľnosti profesionálnymi softvérovými inžiniermi.

7. Bibliography

[1] ELLIS, Clarence A.; GIBBS, Simon J.; REIN, Gail. Groupware: some issues and experiences. Communications of the ACM, 1991, 34.1: 39-58.

[2] FUKS, Hugo, et al. The 3c collaboration model. The Encyclopedia of E-Collaboration, Ned Kock (org), 2007, 637-644.

[3] STEINMACHER, Igor; CHAVES, Ana Paula; GEROSA, Marco Aurelio. Awareness support in global software development: a systematic review based on the 3C collaboration model. In: International Conference on Collaboration and Technology. Springer Berlin Heidelberg, 2010. p. 185-201.

[4] DOURISH, Paul; BELLOTTI, Victoria. Awareness and coordination in shared workspaces. In: Proceedings of the 1992 ACM conference on Computer-supported cooperative work. ACM, 1992. p. 107-114.

[5] GUTWIN, Carl; GREENBERG, Saul. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)*, 2002, 11.3-4: 411-446.

[6] ARORA, Ritu; GOEL, Sanjay. Collaboration in software development: a spotlight. In: Proceedings of the CUBE International Information Technology Conference. ACM, 2012. p. 391-396.

[7] CHONG, Hao; ZHANG, Renwei; QIN, Zheng. Composite-based conflict resolution in merging versions of UML models. In: Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2016 17th IEEE/ACIS International Conference on. IEEE, 2016. p. 127-132.

[8] SUN, Chengzheng, et al. Operational transformation for dependency conflict resolution in real-time collaborative 3D design systems. In: *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 2012. p. 1401-1410.

[9] IONESCU, Bogdan, et al. A chat-centric collaborative environment for web-based real-time collaboration. In: *Applied Computational Intelligence and Informatics (SACI), 2015 IEEE 10th Jubilee International Symposium on*. IEEE, 2015. p. 105-110.

[10] Grady Booch, Ivar Jacobson, and Jim Rumbaugh, editors. OMG Unified Modeling Language Specification Version 2.4.1 - Infrastructure. 2.4.1 edition, 2011.

[11] Grady Booch, Ivar Jacobson, and Jim Rumbaugh, editors. OMG Unified Modeling Language Specification Version 2.4.1 - Superstructure. 2.4.1 edition, 2011.

[12] KOIKE, Hideki. Three-dimensional software visualization: A framework and its applications. In: Visual Computing. Springer Japan, 1992. p. 151-170.

[13] MCINTOSH, Paul; HAMILTON, Margaret; VAN SCHYNDEL, Ron. X3D-UML: enabling advanced UML visualisation through X3D. In: Proceedings of the tenth international conference on 3D Web technology. ACM, 2005. p. 135-142.

[14] VON PILGRIM, Jens; DUSKE, Kristian. Gef3D: a framework for two-, two-and-a-half-, and three-dimensional graphical editors. In: Proceedings of the 4th ACM symposium on Software visualization. ACM, 2008. p. 95-104.

[15] DWYER, Tim. Three dimensional UML using force directed layout. In: Proceedings of the 2001 Asia-Pacific symposium on Information visualisation-Volume 9. Australian Computer Society, Inc., 2001. p. 77-85.

[16] CASEY, Ken; EXTON, Chris. 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, Inc., 2003. p. 63-65.

[17] KROLOVITSCH, Anne-Katrin; NILSSON, Linda. Support of Embedded Hardware Equipment Facilitated by a Smartphone Application. 2011.

[18] Matej Škoda. Three-dimensional visualization of uml diagrams. Diploma Thesis, Slovak University of Technology Bratislava, Faculty of informatics and information technologies, May 2014.

[19] Lukáš Gregorovič. Advanced methods of analysis and design using multidimensional UML. Diploma project, Slovak University of Technology Bratislava, Faculty of informatics and information technologies, May 2015.

[20] LIU, Siyuan, et al. Real-time collaborative software modeling using UML with rational software architect. In: 2006 International Conference on Collaborative Computing: Networking, Applications and Worksharing. IEEE, 2006. p. 1-9.

[21] GUTWIN, Carl A.; LIPPOLD, Michael; GRAHAM, T. C. Real-time groupware in the browser: testing the performance of web-based networking. In: *Proceedings of the ACM 2011 conference on Computer supported cooperative work*. ACM, 2011. p. 167-176.

[22] Gregorovic L. and Polasek I. Analysis and Design of Object-oriented Software using Multidimensional UML. I-Know 2015. 15th International Conference on Knowledge Technologies and Data-driven Business, Graz, Austria. ACM 2015

[23] PARISI, Tony. Programming 3D Applications with HTML5 and WebGL: 3D Animation and Visualization for Web Pages. " O'Reilly Media, Inc.", 2014.

[24] FRUCHTERMAN, Thomas MJ; REINGOLD, Edward M. Graph Drawing by Force-directed Placement. Software: Practice and Experience, 1991, 21.11: 1129-1164.

[25] DWYER, Tim. Three Dimensional UML Using Force Directed Layout. In: Proceedings of the 2001 Asia-Pacific Symposium on Information Visualisation-Volume 9. Australian Computer Society, Inc., 2001. p. 77-85.

[26] GREGOROVIČ, Lukáš; POLASEK, Ivan; SOBOTA, Branislav. Software model creation with multidimensional UML. In: Information and Communication Technology-EurAsia Conference. Springer International Publishing, 2015. p. 343-352.

[27] CAUDWELL, Andrew H. 1996. Gource - a software version control visualization tool http://gource.io

[28] CAUDWELL, Andrew H. Gource: Visualizing Software Version Control History. In: Proceedings of the ACM international Conference Companion on Object Oriented Programming Systems Languages and Applications Companion. ACM, 2010. p. 73-74.

Appendix A - Contents of the Attached Electronic Media

/masters_thesis.pdf	- master's thesis full document
/annotation.txt	- annotation in English
/anotacia.txt	- annotation in Slovak
/src	- folder containing all source code
/src/3duml-prototyp	- 3D UML web application + WebVR prototype
/src/3duml-prototyp/server.js	- 3D UML web application backend implementation
/src/3duml-prototyp/public/js/script.js	- 3D UML web application frontend implementation
/src/3duml-prototyp/public/vr	- WebVR prototype
/src/ea-addin	- EA Add-in prototype
/src/3d-arrow	- the 3D arrow prototype

Appendix B - Installation Guide

B.1. Collaborative 3D UML Modeling Web Application

1. Any web browser is required to run the application. However, the application was developed and tested in Safari v10.1.1 and in Google Chrome v58.0.3029.81 (64-bit). Users experienced some rendering glitches, while using Google Chrome, therefore Safari is recommended.
2. Node.js[12] installation is required.
3. Execute the command "*node server.js*" inside the "*/src/3duml-prototyp*" directory to start the server.
4. Open any web browser and navigate to address "*http://127.0.0.1:8888*".
5. Run the application in multiple web browsers to experience the collaborative functionality.

B.2. EA Add-in Extension

To setup the EA Add-in, follow the official SparxSystems UML, Enterprise Architect documentation[13] on how to create EA Add-in. Look for instructions on how to create EA Add-in with C# and Visual Studio. The folder "*/src/ea-addin*" contains all of the required EA Add-in project files.

B.3. WebVR Prototype

To run the WebVR Prototype as a standard web application all steps can be repeated from section B.1. The only difference is in step four, where it is required to add "*/vr*" to the end of the url, and therefore navigating to "*http://127.0.0.1:8888/vr*". However, to actually experience the virtual reality a VR headset is required. The steps for the installation and setup differ based on the type of the VR headset.

[12] https://nodejs.org/en/

[13] http://www.sparxsystems.com/enterprise_architect_user_guide/9.3/automation/creating addins.html

B.3.1. Smartphone VR Headsets

1. It is required to host the application on any shared hosting or remote server so it can be accessed on a smartphone.
2. Access the application through "*http://yourdomainaddress/vr*" (replace the "*yourdomainaddress*" with your actual domain address).
3. Press the VR mode button located in the bottom right corner of the application.
4. Insert the phone into a VR headset.

B.3.2. PC connected VR Headsets

To run the prototype in a PC connected VR headset, two things are required, a VR headset and a compatible web browser. The installation and setup instruction can differ depending on the web browser and the VR headset. For more information, visit https://webvr.info/.

Appendix C - User Manual

C.1. Controls and Login Page

After opening the web application in the browser a login page is shown to the user. User name authenticates the user. For prototype purposes, password authentication is not required. The login page also explains some basic features and controls of the application.

The controls:

- Scroll to zoom in and out
- Click and drag in space to rotate the scene
- Click and drag a UML class to move it
- Click a layer to select it
- Double click a layer to toggle edit mode
- Click a UML class to select it for edit mode
- Double click a UML class attribute, method or title to modify it

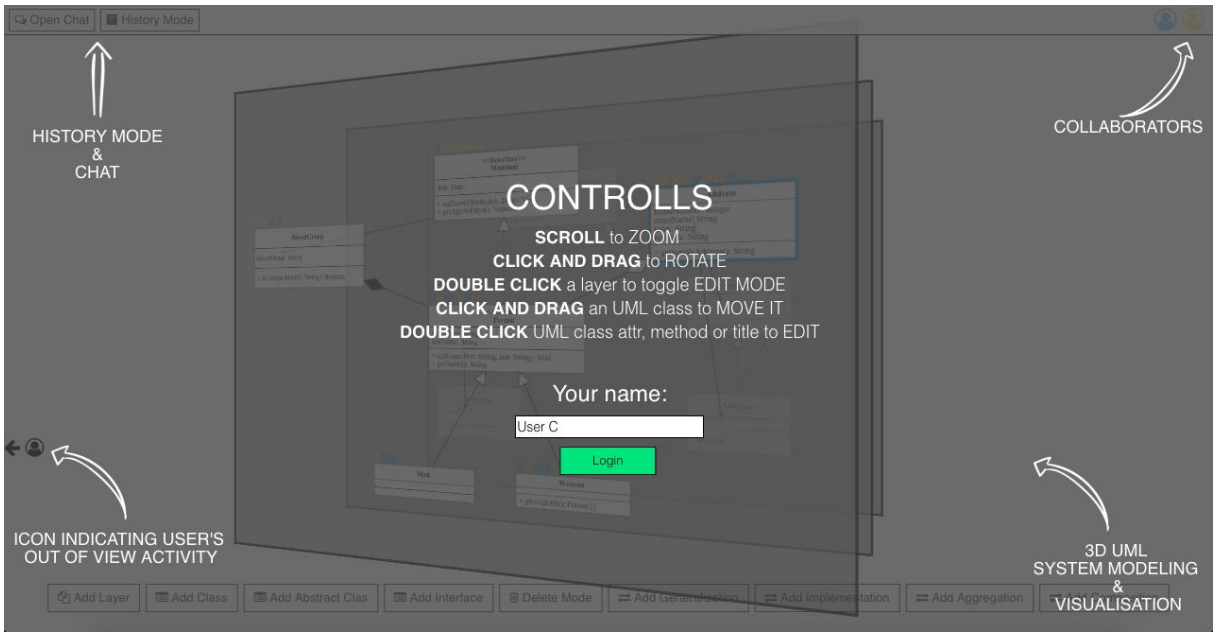

Figure 45: 3D UML aplication login page

C.2. Main Page

After user logs in, he enters the main application (Figure 46) and he immediately joins other collaborators and is presented with the current state of the UML model.

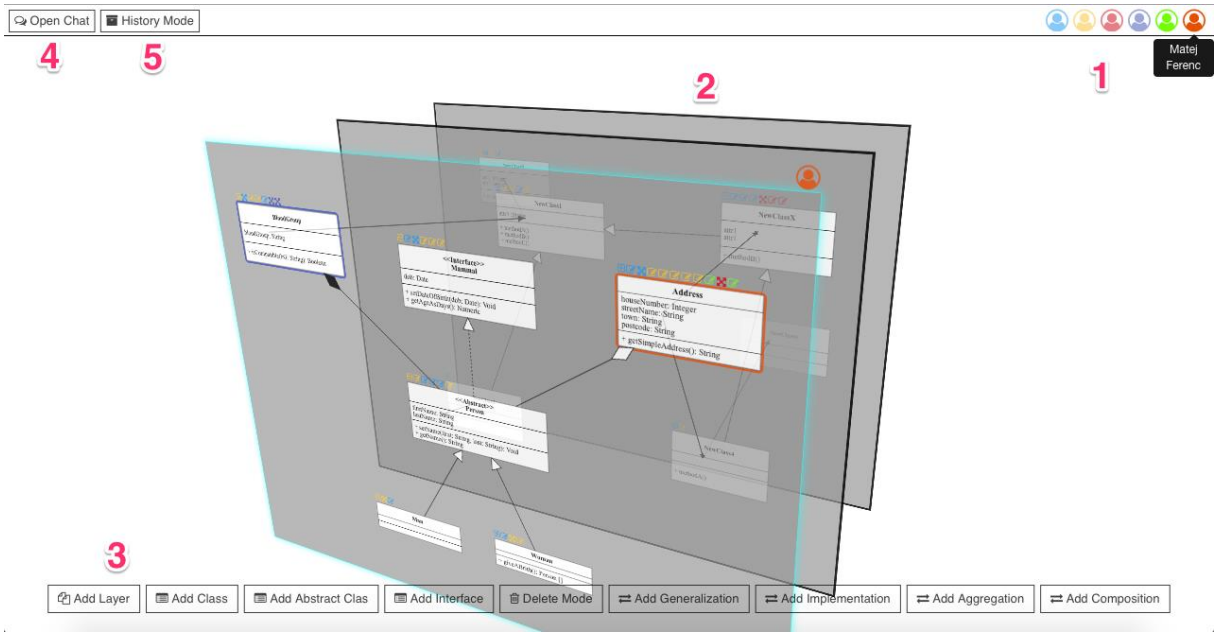

Figure 46: Main 3D UML applications screen

1. All collaborators that are working on the project. User's name is shown if user hovers over the user's icon.
2. The current project.
3. Main menu:
    a. **Add Layer button** – adds a new layer behind the last layer.
    b. **Add Class button** – adds a new class element to the center of the selected layer. If no layer is selected, the user is prompted to select a layer first.
    c. **Add Abstract Class button** – adds a new abstract class element to the center of the selected layer. If no layer is selected, the user is prompted to select a layer first.
    d. **Add Interface button** – adds a new interface element to the center of the selected layer. If no layer is selected, the user is prompted to select a layer first.
    e. **Delete mode button** – when pressed, user enters delete mode. It is then possible to delete any element, including the relations from the layer by clicking on them. To disable delete mode, it is required to press the button again.
    f. **Add Generalization, Add Implementation, Add Aggregation, Add Composition buttons** – after pressing any of these buttons, it is required to

firstly click on the source element and secondly on the target element to add the desired relation between them.

4. **Open chat** – when pressed a chat window is toggled. (see subchapter 4.7.6)
5. **History mode** – when pressed the history mode is toggled. (see subchapter 4.7.5)

C.3. Collaborative Features

All collaborative features that maximize users' awareness are covered and explained in subchapters 4.7 of this thesis.

C.4. Edit Mode

If a user needs to work on a layer that is located behind another layer, he can double click the desired layer for better accessibility to the UML elements located on that layer (Figure 47 - Step 1). Subsequently, the desired layer separates from other layers (Figure 47 - Step 2), the camera is moved in front of the desired layer and zoomed so the layer becomes full screen (Figure 47 - Step 3). This allows the user to work the same way as he would in standard 2D environment. When the layer is double clicked again, the camera and the other layers are moved back to their origin position (Figure 47 - Step 4).

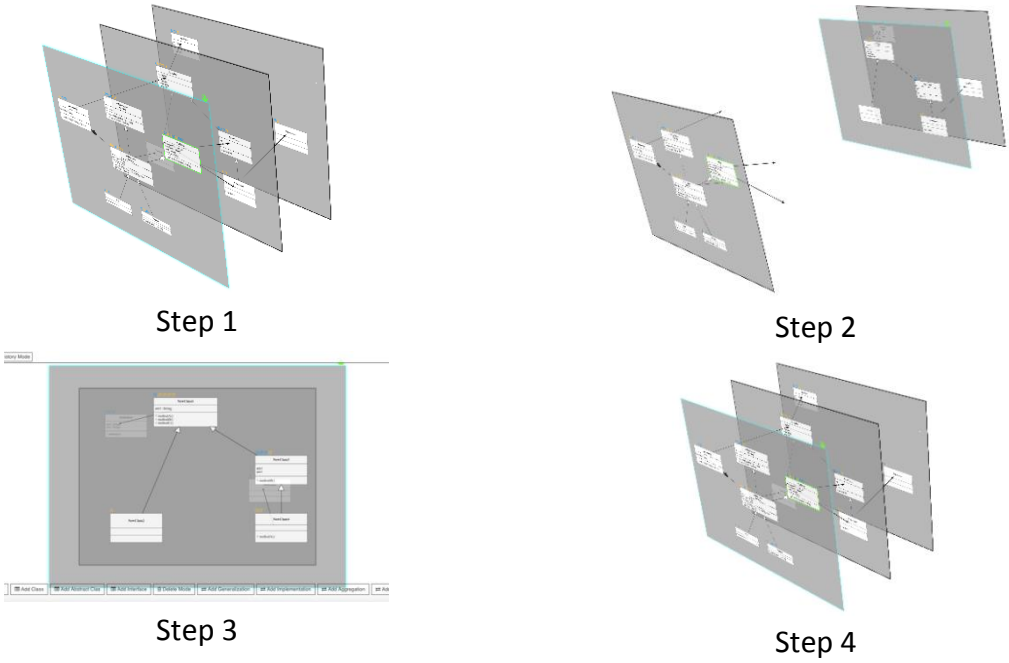

Figure 47: Layer edit mode

C.5. UML Class Modification

To make changes to an UML class, the user first needs to select the desired class by clicking on it. The class is then highlighted with the user's color. When the class is selected it is possible to make changes to it.

Adding an attribute or a method

When a user's mouse enters the attribute's or method's area a small green plus icon appears on the bottom of the area. When clicked, a new attribute or method is added (Figure 48).

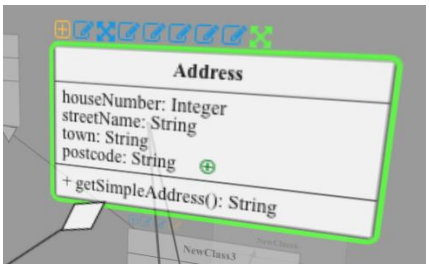

Figure 48: Adding an attribute or a method to an UML class.

Deleting an attribute or a method

When a user's mouse is located over an attribute or a method a small red delete icon appears next to it (Figure 49). When clicked, the attribute or the method is deleted.

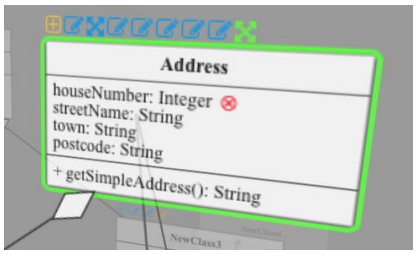

Figure 49: Deleting an attribute or a method from an UML class.

Updating the class name, an attribute or a method

When a user double clicks an attribute or a method, the text can be modified (Figure 50).

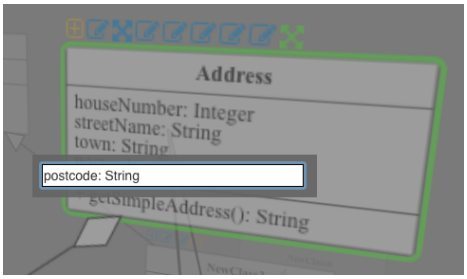

Figure 50: Editing an attribute or a method on an UML class.

Appendix D - Research Paper Draft for VISSOFT 2017 Conference

Collaborative Modelling and Visualisation of Software Systems Using Multidimensional UML

Matej Ferenc and Ivan Polasek

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}@stuba.sk

Abstract

This paper introduces a new and innovative approach for real-time synchronous collaborative system modelling using multidimensional UML, with the aim to reduce the complexity of UML models and increase work efficiency in collaborative system modelling. We propose various visual elements and features with the goal to improve user's awareness of others in a multi-user workspace. We also propose an approach for visualising the history of UML class diagrams simultaneously with the history of user's actions.

Keywords

3D UML, Real-time Synchronous Collaboration, Collaborative System Modelling, Awareness in Groupware

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 concurrently collaborate in order to analyze and design the system. Therefore, the main motivation of this paper is to introduce and propose a new and innovative approach, which would reduce the complexity of UML models and increase the productivity and work efficiency in collaborative system modelling.

Analysis and Related Work

Ellis, Gibbs, and Rein [1], described collaboration using the 3C Collaboration Model which defines collaboration as an interlay between coordination, communication and cooperation. Fuks et. al. [2] later adopts this model and states 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] defined awareness as "an understanding of the activities of others, which provides a context for one's own activities." 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 are they working on, as well as past related aspects such as how this artifact 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 increases 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 modelling and visualising a system in 3D space

can eliminate this problem and introduce many improvements. The idea of system modelling 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 [6], proposes 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 visualisation of a class diagram, where he used force-directed algorithm to layout UML class diagram in 3D space. In his visualisation, 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. Krolovitsch and Nilsson [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 modelling. For example, in their paper, Gregorovic and Polasek [9] 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, modelling systems with 3D UML seems to be the trend of how systems will be modelled in the future.

Our Approach

In our approach, we decided to continue to visualize the system with 2D UML diagrams on 3D layers and we aimed to improve the process of UML modeling with real-time synchronous collaboration. We have designed and implemented a real-time collaborative 3D UML application with many awareness features (Figure 1).

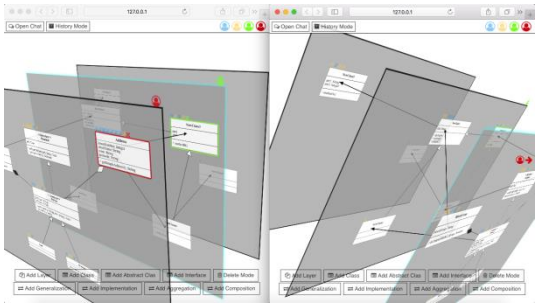

Figure 1. Final 3D UML Application

Our 3D UML application enables users to collaboratively create UML class diagrams in 3D space. 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. The following is a brief summary of the 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 (Figure 2). It is important in multi-user workspace to let others know if a collaborator is online and prepared to work.

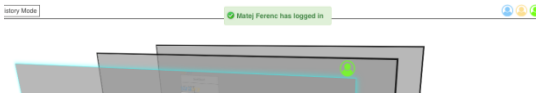

Figure 2. Login Notification

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 following figure 3. The collaborators are always aware of all project participants (Figure 3 - 1), on what layer they are working (Figure 3 - 2), on what specific element they are working (Figure 3 - 3), end even of the actions of others if they are working outside of their view (Figure 3 - 4).

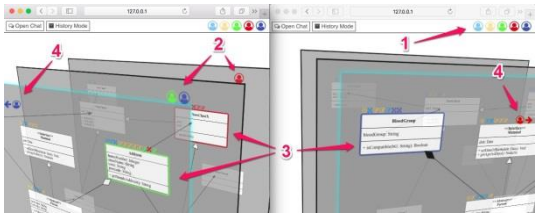

Figure 3. Presence Awareness Features

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 visualize 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 (Figure 4Figure 34).

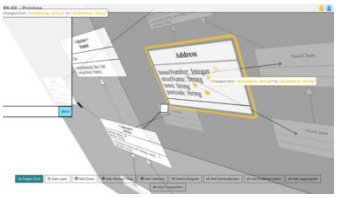

Figure 4. User Action History

However, this feature later proved to be not 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 visualized 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 visualize more detail about the change, by placing his cursor above the element. For example, the figure 5 (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 visual assistance to quickly find what exactly has been changed. We stroked out and used red color to visualize what has been removed and used green color to visualize what has been added. Similarly, the figure 5 (right) shows a user viewing 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.

Figure 5. Final 3D UML application

The previous feature provides visual elements to only show the most recent changes of one UML class element. The next feature enables a user to see the history of all collaborators' actions and simultaneously visualize 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 visualize 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 (Figure 6 - 1). Each step of the slider represents one user's action in history. The collaborator can see more detail about the action in the history window (Figure 6 - 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 (Figure 6 - 3).

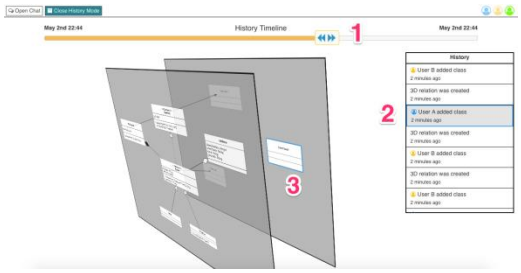

Figure 6. Project History Timeline (History Mode)

The following figure 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.

Figure 7. Project History Timeline Functionality Example

One of the benefits of the 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. The following figure shows our implementation of chat and an example of a chat communication between two collaborators.

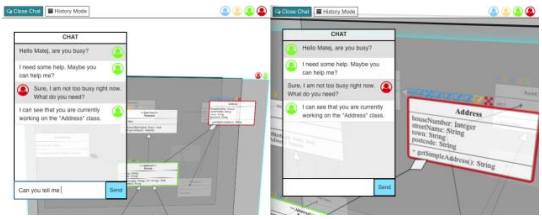

Figure 8. Chat

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 9).

Figure 9. New Message Notification

Besides the above mentioned awareness features, we have also implemented and proposed a solution to a UML elements accessibility problem if they are located on a layer that is located behind another layer. Accessing the UML elements on that layer was only possible from the sides which made the UML modeling complicated. We proposed the following solution. Firstly, a collaborator double clicks the desired layer (Figure 10 - Step 1). Consequently, the desired layer separates from other layers (Figure 10 - Step 2), the camera is moved in front of the desired layer and zoomed so the layer becomes full screen (Figure 10 - Step 3). This allows the user to work the same way as he would in standard 2D environment. When the layer is double clicked again the camera and the other layers are moved back to their original position and it is again possible to visualize the system in 3D (Figure 10 - Step 4). This is a simple and efficient way of how this problem could be eliminated in any 3D UML modeling workspace.

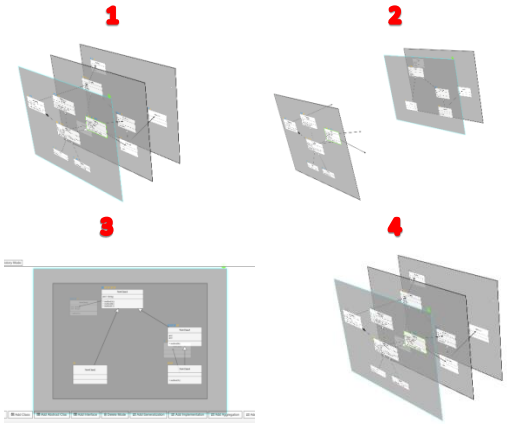

Figure 10. Accessing Hidden UML Elements

Conclusion and Future Work

We have proposed a method for collaborative 3D system modelling. 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 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. Overall, these benefits save a lot of time, and therefore provide faster and more efficient system modelling. However, the above proposed benefits are only our assumption based on the research and results of this paper. These benefits still need to be evaluated and see if they can be intuitively used and adopted by software engineers in collaborative system modelling.

Acknowledgment

The work reported here was supported by the Scientific Grant Agency of Slovak Republic (VEGA) under grants No. VG 1/0808/17 and VG 1/0752/14. This contribution is also a partial result of the Research & Development Operational Programme for the project Research of Methods for Acquisition, Analysis and Personalized Conveying of Information and Knowledge, ITMS 26240220039, co-funded by the ERDF.

References

1. ELLIS, Clarence A.; GIBBS, Simon J.; REIN, Gail. Groupware: some issues and experiences. Communications of the ACM, 1991, 34.1: 39-58.
2. FUKS, Hugo, et al. The 3c collaboration model. The Encyclopedia of E-Collaboration, Ned Kock (org), 2007, 637-644.
3. DOURISH, Paul; BELLOTTI, Victoria. Awareness and coordination in shared workspaces. In: Proceedings of the 1992 ACM conference on Computer-supported cooperative work. ACM, 1992. p. 107-114.
4. GUTWIN, Carl; GREENBERG, Saul. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)*, 2002, 11.3-4: 411-446.
5. KOIKE, Hideki. Three-dimensional software visualization: A framework and its applications. In: Visual Computing. Springer Japan, 1992. p. 151-170.
6. CASEY, Ken; EXTON, Chris. 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, Inc., 2003. p. 63-65.
7. DWYER, Tim. Three dimensional UML using force directed layout. In: Proceedings of the 2001 Asia-Pacific symposium on Information visualisation-Volume 9. Australian Computer Society, Inc., 2001. p. 77-85.
8. VON PILGRIM, Jens; DUSKE, Kristian. Gef3D: a framework for two-, two-and-a-half-, and three-dimensional graphical editors. In: Proceedings of the 4th ACM symposium on Software visualization. ACM, 2008. p. 95-104.
9. GREGOROVIČ, Lukáš; POLASEK, Ivan; SOBOTA, Branislav. Software model creation with multidimensional UML. In: Information and Communication Technology-EurAsia Conference. Springer International Publishing, 2015. p. 343-352.

Appendix E - The Thesis Time Schedule

1st Semester

Week	Task
1-2	Analysis of all aspects related to collaboration, such as user's awareness of others, different types of collaboration or conflict resolution techniques.
3-4	Analysis of all aspects related to 3D UML modelling, such as related work, 3D UML visualisation methods or limitations of 3D UML.
5-6	Analysis of the existing faculty prototype and consideration of the pros and cons of its migration into a web application.
7-8	Analysis of the best suitable technology for real-time synchronous collaboration.
9-10	Design and implementation of the first prototype to experiment with real-time synchronous collaboration.
11-12	Finalization of the DPI document and draft creation of a paper for IITSRC 2016.

2nd Semester

Week	Task
1-2	Analysis and design of approaches to how weighted force-directed algorithms could be applied to automatically layout UML diagrams to improve awareness in 3D multi-user workspace.
3-4	Analysis of two technologies for web based graphical rendering: CSS3D Transformations and WebGL. Discussion with other students about which technology is more suitable for 3D UML.
5-6	Implementation of the web application for 3D UML modeling.
7-8	Implementation of the 3D arrow to connect UML elements between layers.
9-10	Enterprise Architect Add-in implementation to enable integration with EA and to enhance EA with real-time synchronous collaboration
11-12	Finalization of the DPII document

3rd Semester

Week	Task
1-2	Implementation of WebVR prototype to experiment with UML modeling in virtual reality.
3-4	Implementation of UML class history feature.
5-6	Implementation of timeline history feature.
7-8	Finalization of all prototypes and bug fixes.
9-12	Finalization of the DPIII document and the creation of research paper draft for VISSOFT 2017 conference.