Activity Analysis in Software Visualization Systems

V.L. Averbukh[1,2], N.V. Averbukh[2], I.L. Gvozdarev[2], G.I. Levchuk[2]
averbukh@imm.uran.ru|natalya_averbukh@mail.ru|ilyagvz@gmail.com|levchukgeorgy1@gmail.com
[1]IMM UB RAS, Yekaterinburg, Russia;
[2]Ural Federal University, Yekaterinburg, Russia.

The goal of this paper is to identify the task for researching the activity of software engineers using software visualization systems based on virtual reality. It discusses the activity theory statements developed in our country as far back as in the past century. The paper describes possible tasks to analyse the activity of software engineers using such systems. It provides examples illustrating the use of the systems based on virtual reality for the purposes of software complexes representation and visual programming. The activity of the user of such systems is analysed.

Keywords: *software visualization, visual programming, activity theory, virtual reality.*

1. Introduction

Over the past several years a whole range of interesting examples have emerged of using virtual reality in software visualization systems. Of great importance is the evaluation of the efficiency of such systems due to the possibility of using them in software development. Before we proceed to efficiency evaluation, we need to analyze software development as a process. Such analysis should be carried out taking into account the conditions in which this process takes place and, in particular, the influence of virtual reality, its usefulness and applicability in the activity of various categories of software engineers.

The activity theory, which has been developed in the Russian psychology since the 1930s, can be offered as a tool for the analysis. This theory makes it possible to study the working process from the viewpoint of goals and goal-setting and, at the same time, to structure it by defining cognitive constituents – actions – which, in turn, consist of operations. This provides the opportunity to analyze various elements of the development process using the activity theory tools. This type of analysis prioritizes the process of goal-setting by the software engineer and choosing the actions that will lead to accomplishment, which makes it possible to assess the efficiency of using a certain development tool.

Describing a software engineer's activity is a complex task, even in the context of technical development in the 1950s – 60s. A computer program describes processes that are launched on a computer. The task of an engineer is to correctly describe this process, and then to test and debug the program. Nowadays, the activity of a software engineer has become much more complex. Software developers are divided into a whole range of specialties; for instance, a software developer as such, a coder, a tester etc. Each of these specialties presupposes its own type of activity, with its own goals, tasks, including certain actions and operations.

There have been attempts in the IT industry of creating a strict description of each type of work in certain companies. In other companies no software engineering work formalization took place.

The activity of various types of software developers combines versatile work with code, the creative factor, interaction with colleagues and many other things. That is why it is extremely hard to describe a software engineer's activity in full. This paper sets a narrower task of describing the work of a user of software visualization systems based on virtual reality systems. It is noteworthy that currently such systems cannot provide fully-featured development of all the aspects of software. That is why we will describe several types of activities for software engineers, testers and, probably, engineers of a relatively lower level in the system of software visualization with the use of virtual reality. Our goal is to elicit common opportunities of such systems, their bottle necks, problems which users (that is the software engineers themselves) face, and, of course, those advantages that can be enjoyed when using virtual reality for software development.

2. Activity theory

Before proceeding to the use of the activity theory in the fields of software visualization and visual programming, let us provide a general idea of this psychological paradigm. The activity theory was developed in the first half of the XX century by A.N. Leontyev and S.L. Rubinshtein [5, 8].

The central notion determining a person's activity is a conscious goal that a person establishes for themselves. This activity is also defined by the person's motives, their personal qualities and conditions in which the activity occurs. The analysis of the activity should be carried out in two directions. On the one hand, it is necessary to set a goal, motives and conditions for the activity. Let us provide a real-life illustration: the same activity will differ when carried out in the freezing cold, in the heat or in comfortable temperature. Also, when it comes to a specific person, it is preferable to take into account their personal qualities. Some types of activities, on the contrary, require certain personal qualities from the operator. Thus, for instance, the work of a corrector requires attentiveness, and any creative work, obviously, requires creativity.

In the context of this paper's topic, the activity carried out fully or partially in virtual reality will differ from the activity carried out by a software engineer at their usual desk.

The second direction involves dividing activity into reportable actions and dividing the latter, in turn, into operations. It is understood that the same actions may include different operations and, on the contrary, the same operations may be part of different actions. It is important to take into account the fact that carrying out an action is conscious and purposeful, as well as all activity in general.

A person establishes an intermediary goal, which defines the action itself. When the action is fulfilled, the goal is shifted forward, and the previous action becomes the means of carrying out another action aimed at a more general goal. This way, the action aimed at smaller goals is deleted from the conscious and transits to the unconscious. Thus, a hierarchical structure is established: *activity – conscious actions – operations*. In other words, activity breaks down into a number of conscious and motivated actions, which are realized by means of a combination of operations. (Rubinshtein, 2005)

Focusing on activity paradigm in developing a software system could help the developer in creating such conditions, so that the user's tasks (the user in this case being a software engineer of some kind, as we are talking about developing software visualization systems) are connected only with direct goals in a certain programming field. For this, a rather precise description of the activity of various professionals (for instance, software engineers, developers, encoders etc.) is necessary,

including the type of goals and tasks set within the lines of their direction, the type of actions and operations they carry out. Of interest would also be research into motivation, professionally critical qualities (for example, what directions require diligence, criticism, creativity etc.) and other psychological characteristics.

3. The task of analyzing the activity of software engineers

The task of analyzing software engineers' activity has been established in previous papers, [1]. In this paper, we want to provide a case study for such analysis. The systems to be presented in this paper represent the conditions for the activity, as well as the environment and the tool set. It is important for all the components to conform to the goal. It is noteworthy that a goal perceived by the individual determines the whole activity. This is why environment developers need to envisage both the goal and the tasks, the accomplishment of which leads to achieving the goal; they need to think through the actions which the user will carry out and the operations which compose these actions.

When working in the real world a person can choose actions and operations themselves (although the range is somewhat limited by the capabilities and features of the tools), whereas when working in virtual reality, everything should be thought through by the developer. Their job will be much more efficient if they keep in mind the diagram of the activity analysis: the goals established by the software engineer using their system; the tasks established for them; the actions needed to fulfill the tasks; the operations composing the activity.

A software engineer's activity is vast and versatile; however, this paper focuses on the actions carried out in virtual reality, and it considers them as a separate activity. We will describe the goals established by a developer and a tester of a visual software system, as well as by a software engineer working in visual programming. We will describe the tasks that can be established within virtual reality and the operations that may be included into it. The specific feature of activity in virtual reality is the fact that some actions are carried out on several levels simultaneously. On the one hand, a software engineer carries out an action inside virtual reality, for instance, moving around the city or approaching a planet. On the other hand, he or she makes physical movements using certain controllers. The goal here is not enjoying the attraction or believing the events are real (as in entertainment environments), but understanding the software code for further work with it. That is why operations carried out when interacting with the system can be done by means of usual computer devices such as a keyboard and a mouse; although specific VR controllers can also be used. The third level of operations includes those changes that occur on the level of software code. Detailed development of operations should be carried out with due regard for both the specific software system and the supposed user (software engineer).

4. Software visualization based on virtual reality

About a decade ago papers describing the opportunities of virtual reality in the field of software objects representation on the basis of city and landscape metaphors started to emerge [3, 4]. One of the most popular metaphors for software visualization systems using the means of virtual reality is a city metaphor. For example, this metaphor is used by two systems with similar names worth mentioning: VR City [10] and CityVR [7]. The systems based on virtual reality also use other metaphors. In 2018 City Metaphor was used for visualizing software [9].

From the viewpoint of a software engineer's work process description, the developed systems of software product visualization and visual programming based on virtual reality media are of interest for their brief descriptions of user's impressions while working with such systems. The article [7], discusses an approach towards software development called gamification. This approach presupposes the creation of tools, which give software developers an interface similar to computer games. An analysis has been carried out of the way software developers interact with software visualization systems. The developers were excited, they warmed to their work, they felt a certain challenge, experienced immersion into the virtual world with retaining control over the system. See also paper [6]. During the interaction they spent a considerable amount of time on navigation in the virtual world and on choosing the right elements of the program. The users realized that time had gone by quicker than it does in reality, that is why they were willing to spend more time using software media to solve the problem of understanding the task. At the same time, we have not been able to find a full description of a software engineer's activity in such systems yet. Further, we will describe the prototypes of software complexes visualization systems and visual programming systems, and we will attempt to describe the actions, operations, as well as goals and motivations of the users of such systems.

We use a prototype in our research based on an extended city metaphor.

This prototype displays a system visualizing three-dimensional presentation of the code structure in virtual reality based on an extended city metaphor (a city with active agents). Part of the system responsible for visualization, shaping and working in virtual reality is implemented via Webgl and three.js libraries. The system (connecting the software structure visualization in virtual reality with file representation of the code) is implemented on 'js' and C#. Also implemented is the user mode with different levels of access and different types of user accounts (an administrator, who can change access authorization for projects and add users; a developer, who works with the code; a tester, who 'catches bugs', i.e. finds mistakes and provides information about them to the developer). This paper will demonstrate the activity of a developer and a tester.

Let us describe the conditions in which the activity occurs. A user in the role of a developer opens the project represented as a city (see Fig. 1). The user's movements in the virtual environment are carried out by means of a camera moving around the virtual city, its buildings and rooms inside the buildings. The movements are carried out by means of a keyboard or a mouse. The class is represented as a building. it has several floors, which is connected with the fact that for each class there are two representations for the descriptions. In a simple representation, each single method is described in a room or on a specific floor. There is also a logical description of the types of data the class works with, the type of methods it uses, and possible applications of this class.

The user may have the role of a tester. Let us consider a simple example – the task of finding the root of a number. The tester enters a number; the active agent enters the corresponding method in the examined class. This is displayed as the agent entering the building which represents the class (see Fig. 1, 2), going along the corridor and choosing the right room (see Fig. 3). The agent enters the room (method) through the door and exits it in the same way. All this is displayed as the agent entering with a number written on him and exiting with the root of this number. And the wall of this building (Fig. 4) illustrates the way this method influences memory: what load there is, a brief characteristic of the weight of this number (as there can be very big numbers, which put a heavy load on the processor); and a brief description of the way this method finds the root from this number.

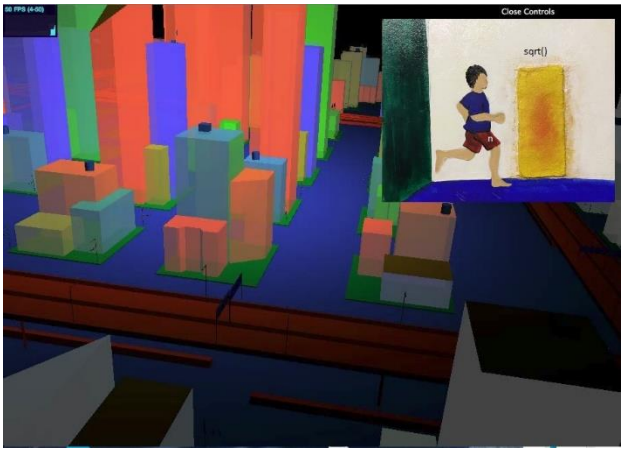

Fig. 1. Presentation of a software project as a city in virtual reality and the inner part of the structure of the code with the figure of the active agent

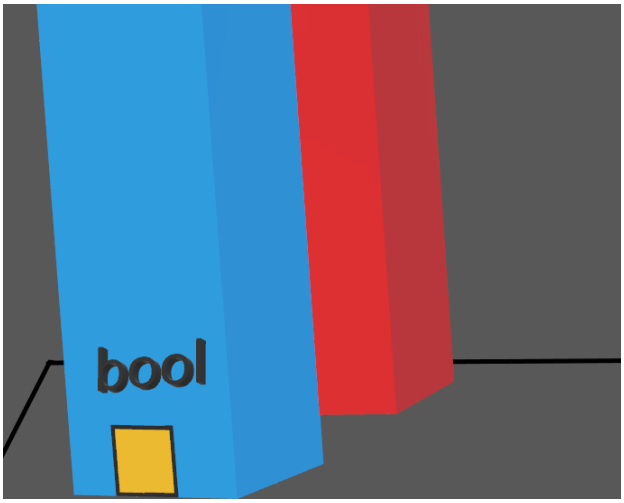

Fig. 2. An entrance into the building describing the 'boolean' class

As a result, the tester understands the way this method works, the way the program works. Visual manifestation makes it possible to quickly understand whether the program works correctly.

The city metaphor makes it possible to visualize larger volumes of data in a much denser way. Virtual reality allows information perception on several levels. One can inspect all the procedures step-by-step due to agent movements. There is an opportunity of working within the method, watching the elements go through all the procedures and evolve.

In the work of a usual program, the system takes the code text and transcodes it into computer code. In our system, each method has its own name and place in the code, and the rooms in the buildings of the city are named accordingly. A software developer can watch the program function in order to work out the code and its structure, which offers an advantage over reading the code to decide on further development.

At this point, there is no possibility to interfere with the program from virtual reality. These possibilities are to be further extended.

Each type of software engineer faces similar tasks in their line of work, which have no critical differences.

A software engineer's activity is generally (not only in virtual reality) carried out in the following way: an engineer uses two screens, one of which is connected with VR glasses and shows the city, while the other one shows the code. The engineer switches between the systems. He or she can launch the program and watch it work in virtual reality. Upon discovering the unrealized methods, an engineer switches to the second screen and searches the web for the possibility of using ready-made methods. Then he or she returns into virtual reality and watches the way they work to make a decision on whether they need changes or enhancements.

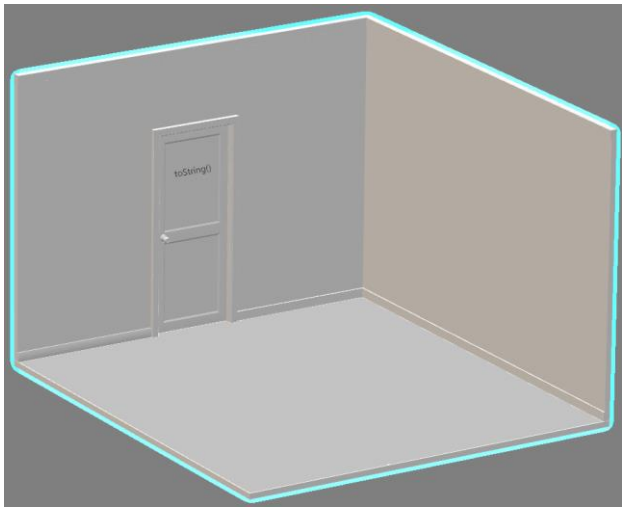

Fig. 3. A corridor with a view of the entrance into the 'toString()' method from the 'boolean' class in the building sectional view

Fig. 4. A view of the wall in a room with a visual and textual description of the program elements

5. Visual Programming

The developed system of visual programming can be used for creating object-oriented programs. Such programs have a certain inner structure and hierarchy, a link between the objects, an ordered set of operations. The goal of this project is to reduce the number of intermediaries between the software engineer and the program idea. Text is not the most obvious and natural means of translating the idea into the program. It appears that coding is better through manipulations with graphic objects. Nowadays, coding looks the following way: a software engineer enters the programming environment and starts creating files, writing texts, creates packages, classes, studies the syntax. This can be avoided if classes are represented not by a sequence of lines with content names, not by just by a succession of structures and operators, but by a graphic object. In order to work in the system of visual programming a person needs to understand the process of carrying out a certain action. This understanding should be as intuitive as possible.

The cosmic metaphor with a heliocentric world view is chosen as the basic metaphor (Fig. 5). Notably, the programs' essences are represented by planets, their satellites and rings (like Saturn's rings). User classes are represented by planets. Each

planet (class) has two types of views: the open one, which looks like a planet with satellites and rings (see Fig. 6), and the active one, where the class is chosen. The active view represents a circle with a section of rings on the left and structured satellites on the right. The section of rings represents class methods. Each ring displays a separate method. Outer rings are public methods, while inner ones are private and protected. Satellites represent class fields. From top to bottom, first go the satellites furthermost from the planet – public fields, then go closer satellites – private and protected ones. Inside the planet there is also a section of rings, and an ordered set of satellites (static methods and fields), belonging to the class itself (Fig. 7). In the center there is a nucleus containing all the class constructors. [1]

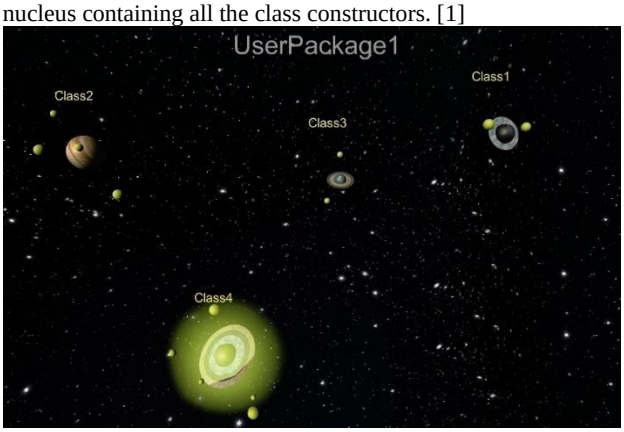

Fig. 5. View of the visual programming environment

If a planet represents a class, then creating a method means creating a ring for this planet. If a software engineers wants to create several methods he or she does not need to code similar elements; he or she can simply click on the planet and then click on a ring in a dropdown menu for the 'create a ring' option. After that, the engineer sets the types and names of input data, which is also implemented graphically. The goal of creating this system is to make coding a simpler and comfortable process. The goal of a software engineer (when working inside virtual reality) is to code and enhance the program.

Virtual reality provides visibility and the opportunity to keep all the created classes, interfaces and their methods in sight, with the option of in-depth analysis of their structure by means of zooming the camera in or out, or by clicking on them with a mouse. Virtual reality provides navigation benefits. One can travel around the class structure (the cosmic space). The effect of presence is imitated via the camera. The interaction can be carried out with VR controllers but, for now, it is performed on a keyboard (when cording identifiers and creating objects).

The system is implemented in such a way that a writing into the source file (.java) is made when the user changes the structure of the program (for example, creates a variable – a satellite or a planet of a certain type); when the variable is deleted, the writing from the source file is erased. At present, the file responsible for the visualization system state and the corresponding source file should be kept paired up and should not be changed separately from each other, because there is no interpreter to translate .java files into graphic objects.

A software engineer performs actions creating planets (classes) by carrying out operations of choosing interfaces and typing. One can also choose a project nucleus through the operation of choosing fields (satellites) and create static methods (rings).

When analyzing a software engineer's activity, identifying the level of operations is quite difficult. Obviously, one cannot reduce the operations to hand movements, mouse clicks, the use of keyboard or VR controllers. An operation should be described on three levels: movements in the material world, changes in virtual reality, and changes in the code. In the material world, a software engineer moves the mouse or the controller, clicks the button. In virtual reality, a cursor, a pointer or any other type of indicator moves, planets are selected, text boxes appear etc. On the level of code, a file is created, a name is assigned to it etc.

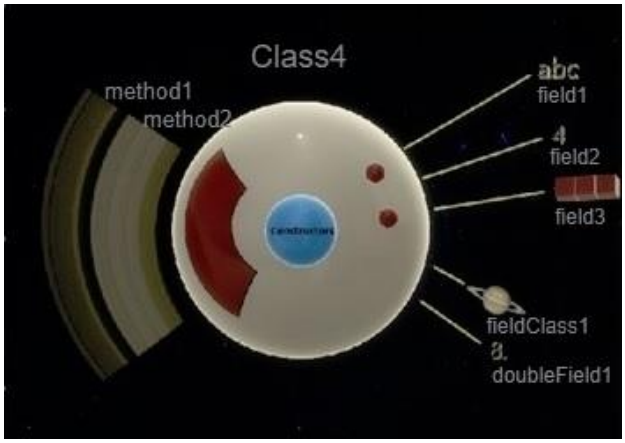

Fig. 6. View of the class

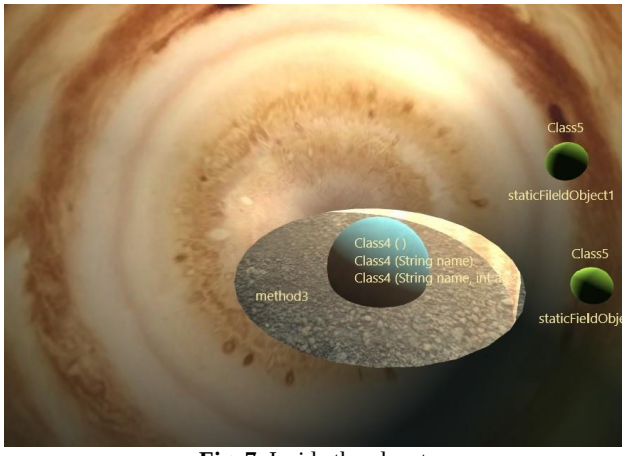

Fig. 7. Inside the planet

6. Conclusion

This paper describes the activity of software engineers in software visualization and visual programming systems. Our primary focus has been on the systems themselves and on the actions performed there. The goal of an activity is typically quite obvious; it comes down to achieving the required result (analysis of the program's work, understanding, coding a new program etc.). A crucial part of the activity analysis is elaborating the motivation. However, when describing an activity of a category of people (instead of a single person), we face difficulties talking about motivation, as it is, in effect, a matter of personal business. That is why we can only talk about the motivation for a certain aspect of activity. In this paper, we are interested in the motivation of choosing virtual reality as an environment for executing an activity. It can be connected with the content of the activity (comfort, new advantages) or not ('out of curiosity', 'to brag about using cutting edge technologies', 'to try out new features', 'boss's orders' etc.).

The developed prototypes use traditional interfaces, but the use of VR controllers, as well as gesture interfaces, is contemplated, which may offer considerable advantages in carrying out operations in virtual reality environments. [2], [11], [12]. This may reduce the gap between the two levels of operations: physical actions of the user (software engineer) and changes in the virtual environment.

Our research is at the initial stage now. Based on the developed prototypes, experiments should be carried out to study

the activity of future users, different categories of software engineers.

The work was supported by Act 211 Government of the Russian Federation, contract No 02.A03.21.0006

7. References

[1] Averbukh V. L., Gvozdarev I. L., Levchuk G. I. Software Visualization based on Virtual Reality // GraphiCon 2018: Proceedings of the 28th International conference on computer graphics and machine vision. Tomsk, 24-27 Sept., 2018 . P. 77-81

[2] Averbukh V. L. Averbukh N. V., Starodubsev I. S., Tabolin, D. J. the Use of gestural interfaces in the interaction with objects // Scientific Perspective – 2014. – No 56(10). – P. 57-66.

[3] Glander, T., Dôllner, J.: Abstract representations for interactive visualization of virtual 3d city models. Computers. Environment and Urban Systems 33(5), 375–387 (2009)

[4] Glander, T., Dôllner, J.: Automated cell-based generalization of virtual 3d city models with dynamic landmark highlighting. In: The 11th ICA workshop on generalization and multiple representation, June 20–21 , Montpellier, France. The International Cartographic Association (2018)

[5] Leontiev A. N. Activity. Consciousness. Personality. M., Political. 1975.

[6] Merino L., Bergel A., Nierstrasz O. Overcoming issues of 3d software visualization through immersive augmented reality. Proceeding of 2018 IEEE Working Conference on Software Visualization (VISSOFT). 2018. Pp. 54-64.

[7] Merino L., Ghafari M., Anslow C., Nierstrasz O. CityVR: Gameful Software Visualization // IEEE International Conference on Software Maintenance and Evolution (ICSME TD Track). 2017, pp. 633-637.

[8] Rubinstein S. L. Fundamentals of General psychology. - SPb. Peter, 2005.

[9] Rüdel M.-O., Ganser J., Koschke R. A Controlled Experiment on Spatial Orientation in VR-Based Software Cities // 2018 IEEE Working Conference on Software Visualization (VISSOFT), 2018. Pp.21-31.

[10] Vincur, J., Navrat, P., Polasek, I.: VR city: Software analysis in virtual reality environment. In: 2017 IEEE International Conference on Software Quality. Reliability and Security Companion. pp. 509–516 (2017)

[11] Zhai S., Hunter M., Smith B. A. The Metropolis Keyboard – An Exploration of Quantitative Techniques for Virtual Keyboard Design // UIST '00 Proceedings of the 13th annual ACM symposium on User interface software and technology, 2000. Pp. 119-128.

[12] Zirkelbach C., Krause A., Hasselbring W. Hands-On: Experiencing Software Architecture in Virtual Reality // Report number: TR_1809. Christian-Albrechts-Universität zu Kiel. January 2019. DOI: 10.13140/RG.2.2.23096.39680.