Informatics in Education, 2006, Vol. 5, No. 2, 255–264

JeCo: Combining Program Visualization and Story Weaving

Niko MYLLER Jussi NUUTINEN
Department of Computer Science, University of Joensuu
P.O. Box 111, FI-80101 Joensuu, Finland
e-mail: niko.myller@cs.joensuu.fi, jussi.nuutinen@cs.joensuu.fi

Received: May 2006

Abstract. We present a collaborative learning tool for programming, Jeliot Collaboratively or JeCo. Jeliot Collaboratively is a combination of a program visualization tool for Java programs, called Jeliot 3, and a collaborative authoring tool, Woven Stories. We introduce these systems and explain how they can be used in learning. Furthermore, we present future directions in order to support a wider range of use cases with JeCo.

Key words: CSCL, program visualization, programming learning, collaborative authoring, Java.

1. Introduction

Currently, a large number of students are struggling with their programming courses (McCracken *et al.*, 2001) and especially in the distance education courses in programming (V. Meisalo *et al.*, 2003). Pair programming is one of the processes suggested by extreme programming (XP) methodology. Pair programming has been found successful in learning situations (Kuppuswami and Vivekanandan; McDowell *et al.*, 2003; Williams *et al.*, 2003; Nagappan *et al.*, 2003; Stotts *et al.*, 2003). Studies have shown that students enjoy programming in pairs, produce better quality programs and perform better in examinations. Furthermore, students should also learn team work skills because team work is one of the key elements of software development. Pair or group programming could be introduced to distance education programming courses as well but that would require new tools to support the interaction between students when they are learning and working at a distance. However, there have been only few attempts to develop appropriate tools (Ratcliffe and Thomas, 2004; Jehng and Chan, 1998; Hanks, 2005).

Hundhausen (Hundhausen, 2002) and Hübscher-Younger (Hübscher-Younger and Narayanan, 2003) have claimed that when communication and collaboration are accompanied by visualizations, students' learning of algorithms is enhanced. According to Hundhausen (Hundhausen, 2002), bi-directional communication between instructor and students as well as between students supported with relevant visualizations helps students to learn algorithmics because visualization supports relevant communication of the topic. Hübscher-Younger (Hübscher-Younger and Narayanan, 2003) states that construc-

tion, sharing and discussion of the students' representations of a certain algorithm help students to learn the algorithm better.

We have developed a program visualization tool, *Jeliot 3*, that can be used during learning and teaching of programming (Moreno *et al.*, 2004c). It is designed to help novice level students by visualizing the Java program's state and source code during the execution of the program. In this way, students are able to build a reference model that they can use to solve problems they have not encountered before and they acquire vocabulary to discuss programs with each other and their teacher (Ben-Bassat Levy *et al.*, 2003).

As a first attempt to support distributed collaborative programming and learning to program, we have combined a program visualization system, Jeliot 3, and a co-authoring tool, Woven Stories, into an application called *JeCo* (*Jeliot Collaboratively*) (Moreno *et al.*, 2004b; Moreno *et al.*, 2004a). The version of JeCo described in earlier papers was based on the previous version of Woven Stories which had limited capabilities. In this paper, we explain a new version of JeCo and the underlying systems: Jeliot 3 and the new version of *Woven Stories* (Nuutinen, 2006). Moreover, we present some future directions which could benefit all these systems in the contexts of distance education.

2. Previous Work

As stated in the previous section, new tools are needed in order to support the interaction between students when they are learning and working at a distance. There are several tools that support just the minimum requirements (e.g., collaborative editor or desktop sharing) to support distributed pair/group-programming as reviewed by (Hanks, 2005). However, there are less tools that try to elaborate on this and add other features to increase the usability and productivity of the tools such as visualizations or gestures (Hanks, 2005).

An example of a tool that uses visualization to support collaboration is *VorteX* (Ratcliffe and Thomas, 2004). It supports the design of programs with a modifiable class diagram that is shared by the group. This tool can be primarily used only during the design phase of programming, however, other parts of the programming process should also be supported. Jehng and Chan (Jehng and Chan, 1998) have developed a collaborative programming environment for LISP-LOGO. They also showed that their environment had a positive influence on learning results when students were learning recursion in collaboration. This is a good indication how visualization can support programming when students are not co-present. However, LISP-LOGO or LOGO based languages in general are not commonly used in programming courses and thus this approach is not widely applicable as such.

3. Jeliot

Jeliot 3 (Moreno *et al.*, 2004c) is a program visualization tool supporting teaching and learning of programming in introductory Java programming courses. It supports dif-

ferent kinds of approaches in teaching of introductory programming such as objects-first or fundamentals-first. It is released under the GPL and is freely available at (`http://cs.joensuu.fi/jeliot/`).

Fig. 1 illustrates the user interface of Jeliot 3. Most of the Java language concepts are currently supported and visualized by Jeliot 3. It displays the operation of a virtual machine during program execution. However, the animation takes place on the Java language level and not the level of bytecode to make it relevant for students. It shows all the details of the program execution by visualizing expression evaluations, method calls, and object- and array allocations. Thus, the visualization can be used to teach and learn programming concepts. Another view (not shown in the figure) shows the method call tree of the executed program. Furthermore, Jeliot provides a possibility to explore the history of the execution through static snapshots taken before and after a possible interesting event in the visualization.

In a classroom study, Ben-Bassat Levy *et al.* (Ben-Bassat Levy *et al.*, 2003) found that Jeliot 2000 supported the learning of mediocre students. Students created viable mental models of the program execution based on the Jeliot's visual display. Jeliot also provided them with vocabulary to describe the execution. Because of that students were able to answer questions related to unseen situations by drawing a Jeliot-like display and describing the situation through the diagram. Because Jeliot 3 uses the same visualization scheme as Jeliot 2000 and only extends it, we can assume that similar results are achieved when Jeliot 3 is used in a classroom.

One reason to develop Jeliot 3 was to enhance the modularity and extensibility of the previous versions. We have developed an extension that allows Jeliot 3 to interact with BlueJ, a widely used educational programming environment. It is possible to start animations in Jeliot directly from BlueJ's object bench. Another plugin has been developed for Editing Java Easily (EJE), a development environment containing special features that simplify the usage of the tool for novices and allows the use of the tool directly from a web page. Furthermore, in this paper we present how Jeliot 3 has been combined with a co-authoring tool, Woven Stories, to enable use of Jeliot collaboratively on the web.

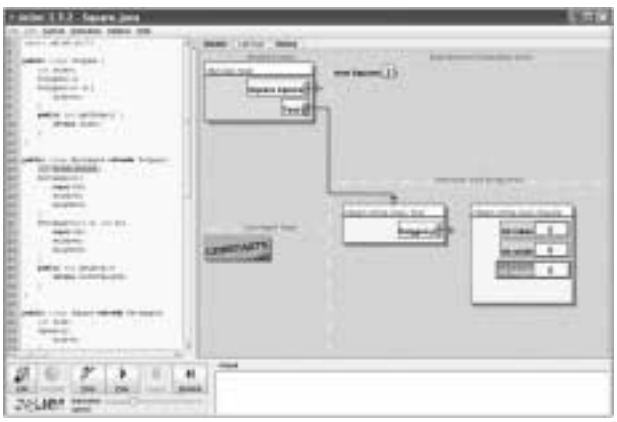

Fig. 1. User interface of Jeliot 3.

4. Woven Stories

Woven Stories (WS) (Gerdt *et al.*, 2001; Nuutinen, 2006) is a concept that allows persons to write stories with a new approach. The stories are built of small blocks that are called *sections*. Each section can contain an unlimited amount of text. The sections are linked together with arrows that are called *edges*. Using this approach the story can be visualized as a directed graph where each path forms an individual story.

Fig. 2 represents a simple woven story that has four different *storypaths*. These storypaths have been achieved by introducing three new sections to the original story of six sections (gray in Fig. 2). With this approach, the users are able to change the existing story by introducing new sections to it. It means that the users are able to introduce their ideas without removing anything from the original story. The result of the writing process with Woven Stories can be a versatile document with several optional storypaths.

The concept of Woven Stories is a mixture of *concept mapping* (Novak and Gowin, 1984), *flow charts*, *collaborative writing*, *graphs* and *finite automata*. It forces the users to think what they write, to divide the texts into sections and finally, to see the relationships between the sections. Since people learn what they process (Bereiter, 2002; pp. 274), this approach can be efficient for learners, especially in collaborative situations.

Based on the concept of the Woven Stories a client-server application has been developed (Nuutinen, 2006). This tool consist of a client, *the Loom*, and a server called *WS-Server*. Loom implements the concept of the Woven Stories into a collaborative environment, where synchronous and asynchronous users can work with the same story. Fig. 3 represents the user interface of the Loom.

While using the Loom the users can see at all times the structure of the document from the *Story Space* (1. in Fig. 3). The contents of the sections can be seen by selecting the section from the Story Space. The contents are displayed on the *Content Viewer* (2. in Fig. 3) at the top right corner. The Loom also supports small scale communication with *Chat* (4. in Fig. 3) and awareness with *Action Info* (5. in Fig. 3) that displays recent activities of the current document.

The Story Space has a relaxed WYSIWIS (What You See Is What I See) approach which means that the users have the same data, but do not necessarily see the same portion of it at all times. All the actions with the objects (sections and edges) are first sent to the server and then back to the clients. This guarantees that all the clients really have the same data at all times.

The Loom has been tested in a couple of situations with a small number of users, during which it was found that the concept is easy to understand and that the software is easy to use (Nuutinen, 2006). However, users found it problematic to reuse the sections that other people had written. Due to this the stories tend to have a sequential nature.

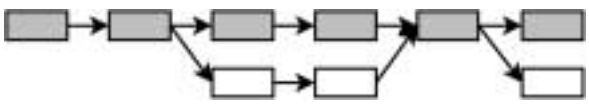

Fig. 2. Visualization of a simple woven story.

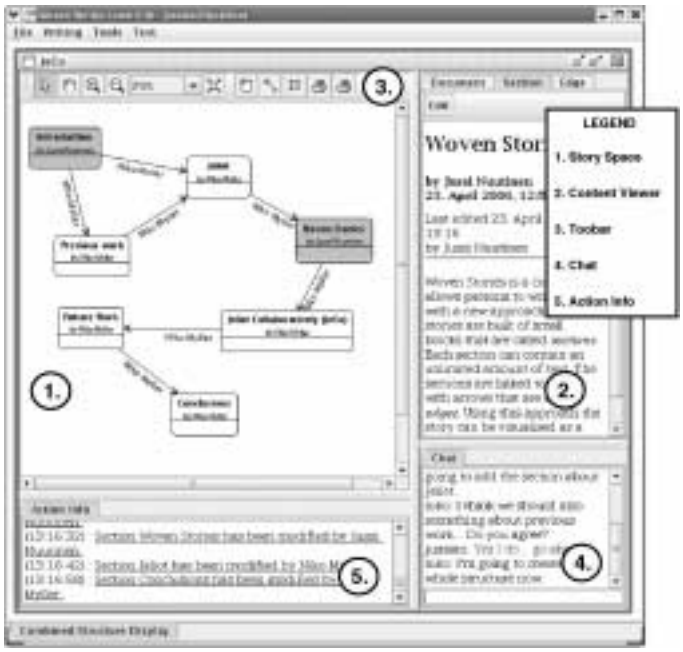

Fig. 3. User Interface of the Loom.

We believe that this approach can be efficient in various subject domains. If the communication and the awareness features of the Loom are improved, it can be used even more efficiently within distributed group.

5. Jeliot Collaboratively (JeCo)

Program visualization tools normally support only individual learning and engage students cognitively but do not necessarily support social activities, even though these social practices can be crucial to learning. In a classroom, the social activities can be handled with group activities that are guided by the teacher without any additional support from the software. However, in distance education this is not possible because the interaction and social aspects need to be supported by software. We want to support both cognitive and social processes in a distance education through *JeCo* (Moreno *et al.*, 2004b; Moreno *et al.*, 2004a), which is a combination of Jeliot 3 and Woven Stories.

JeCo gives students a possibility to collaborate with each other with tools that are made for the purpose. Students can acquire vocabulary related to the programming concepts from Jeliot and thus it creates a context for the discussions (Ben-Bassat Levy *et al.*, 2003). In the case of JeCo, this vocabulary will grow into an inter-subjective set of shared concepts.

Jeliot and Woven Stories are combined by modifying the Loom client program whereas the WS-Server can be kept as it is because of its flexible design. Fig. 4 illustrates

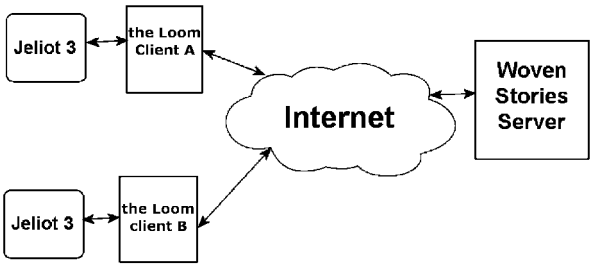

Fig. 4. The combination of Woven Stories and Jeliot in JeCo with multiple users.

how these systems are related to each other. The Loom client was modified to recognize whether a section contains program code that can be animated in Jeliot. If it does, the user is informed of this possibility by showing a text telling that Jeliot is available on that section. When a user clicks the right mouse button on that kind of section, there is an option in the popup menu to animate the program with Jeliot. This creates a link between these two systems and the animation allows both asynchronous and synchronous communication among students about a program or its visualization. The program code can be sent to the forum by editing a section and clicking on a button with Jeliot logo on it and paste the code into the appearing window. This will make the code attached to the section and recognizable to the Loom client.

For example, any user (e.g., *client A*) can send to the server a section that consists of a message and program source code. If another user (e.g., *client B*) wants to see the animation of the program code sent by *client A* and discuss that with *client A* about it, s/he can request the section and animate it. While Jeliot is animating the program, *clients A* and *B* can discuss the phases of the program's execution. This kind of scenario is illustrated also in Fig. 4 and the users would see a similar view as shown in Fig. 5. In the Fig. 5, the currently viewed section is bordered with a rectangle and sections containing the source code that can be animated with Jeliot 3 are indicated with the *Jeliot available* text in the bottom of the section rectangle. If the discussion leads into new developments, *Client B* could add another section to the document and connect it to the section(s) that inspi red this work. Thus, it would keep the history of the development visible to users that are not currently logged in into the system.

Because Woven Stories supports HTML documents, it is possible to bring the whole course materials inside multiple Woven Stories documents. For example, each topic of the course can be a separate document. These documents can contain regular text but also source code that can be visualized with Jeliot. Several students can then see the same visualization at the same time and discuss it, but it is also possible that students comment on the programs and give their own examples and let the other students try them out as well. Thus, the tool is supporting both synchronous and asynchronous modes of communication and learning. Moreover, JeCo can support programming courses as a platform in collaborative exercises, assignments and projects.

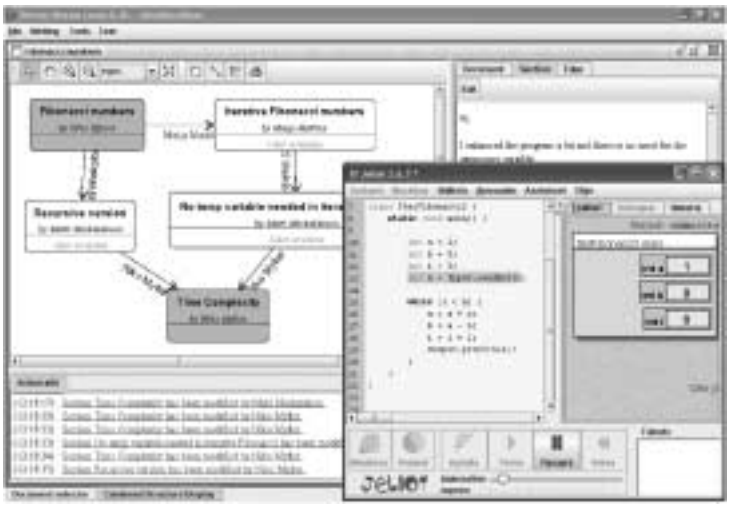

Fig. 5. User Interface of JeCo.

6. Future Work

As stated in Section 4, the users of the Loom have had problems to create stories with optional storypaths. While it is an important research question to find solutions to this problem within the development of Loom, it is not such an issue when considering JeCo. This is due to the fact that program versions tend to branch naturally. Furthermore, the storypaths can be used for other purposes than version management. For example, we could consider the story as a program and view the graph of sections as a class diagram of a program in which each section would be a class.

Currently, there are also some features missing from the Loom in order to support distributed use of the tool: Users should be more aware of other users and the communication capabilities should be enhanced.

Awareness (Gutwin and Greenberg, 1996) means the knowledge of other people and their actions. This is an important aspect of collaboration, since the users of the system must know what the others are doing at the same time, since this can affect their work. For example the users have to be aware who are present at the system in order to engage into conversations.

Communication is important part of collaboration. In systems where the users collaborate on-line, the communication should be carefully planned. Currently, the Loom supports communication only by chatting. While this can be used when the users are on-line, there is no way to communicate in other situations. Due to this, features supporting the asynchronous communication has to be created. For example features such as annotations (Nokelainen *et al.*, 2003) could be useful in context of Woven Stories. In synchronous communication, the users should be able to refer to the artifacts they are dealing with. For example, in a chat session a user might want to refer to a just written section and want to get opinions from others. In order to make things simple, users should be

able to point at that section from the Story Space. Such features are called *conversational props* (Brinck and Gomez, 1992).

By implementing the features to improve the support of awareness and communication in the Loom, we can improve the usefulness of the Loom in distance use cases. When paying attention also to asynchronous collaboration aspects, the Loom can be used also in situations where the users are not able to be on-line at the same time. Since the JeCo is based on the Loom these same benefits apply also to it. For example, the annotation could be used to comment the source code or similar code segments in multiple files could be annotated in order to find similarities.

Currently, the visualizations of two distributed users are not synchronized, and thus it can be difficult to discuss a very specific point in a visualization. There should be a possibility to stream one visualization to multiple users so that one user would be the host of the visualization and other users could see the visualization but not necessarily control it. This could be used also for guided session where a teacher controls the visualization and send explanations to the chat and students can see the visualization from their screens and post questions and comments to the chat.

7. Conclusion

In this paper, we have introduced a program visualization tool, Jeliot 3, a collaborative authoring tool, Woven Stories, and their combination JeCo, a collaborative learning tool for programming. The combination was possible due to their flexible and modular design. We also discussed the possible scenarios where these systems can be used and how. We have also presented future directions in order to enhance the learning experience with JeCo. We have already published Jeliot 3 under GPL license and plan to do so for JeCo as well. We hope this will encourage others to use the tools and contribute to them in order to form a learning community of users and developers.

References

Ben-Bassat Levy, R., M. Ben-Ari, P.A. Uronen (2003). The Jeliot 2000 program animation system. *Computers & Education*, **40**(1), 15–21.

Bereiter, C. (2002). *Education and Mind in the Knowledge Age*. Lawrence Erlbaum Associates, Mahwah, NJ.

Brinck, T., and L.M. Gomez (1992). A collaborative medium for the support of conversational props. In *CSCW '92: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work*. ACM Press, New York, NY, USA. pp. 171–178.

Gerdt, P., P. Kommers, C.-K. Looi, E. Sutinen (2001). Woven stories as a cognitive tool. In *Cognitive Technology: Instruments of Mind, Lecture Notes in Computer Science*, **2117**, 233–247.

Gutwin, C., and S. Greenberg. Workspace awareness for groupware. In *CHI '96: Conference Companion on Human Factors in Computing Systems*. pp. 208–209.

Hanks, B. (2005). *Tool Support for Distributed Pair Programming – Empirical Studies of Distributed Pair Programming*. PhD dissertation, Computer Science, University of California, Santa Cruz. `http://faculty.fortlewis.edu/hanks_b/publications/dissertation.pdf` (Accessed 8.8.2006).

Hübscher-Younger, T., and N.H. Narayanan (2003). Constructive and collaborative learning of algorithms. *SIGCSE Bulletin*, **35**(1), 6–10.

Hundhausen, C.D. (2002). Integrating algorithm visualization technology into an undergraduate algorithms course: ethnographic studies of a social constructivist approach. *Computers & Education*, **39**(3), 237–260.

Jehng, J.-C.J., and T.-W. Chan (1998). Designing computer support for collaborative visual learning in the domain of computer programming. *Computers in Human Behavior*, **14**(3), 429–448.

Kuppuswami S., and K. Vivekanandan (2004). The effects of pair programming on learning efficiency in short programming assignments. *Informatics in Education*, **3**(2), 251–266.

McCracken, M., V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. Ben-David Kolikant, C. Laxer, L. Thomas, I. Utting, T. Wilusz (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, *SIGCSE Bulletin*, **33**(4), 125–180.

McDowell, C., L. Werner, H.E. Bullock, J. Fernald (2003). The impact of pair programming on student performance, perception and persistence. In *Proceedings of the 25th International Conference on Software Engineering*. IEEE Computer Society. pp. 602–607.

Meisalo, V., E. Sutinen, S. Torvinen (2003). Choosing appropriate methods for evaluating and improving the learning process in distance programming courses. In *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference (FIE2003)*. Boulder, CO, USA. pp. T2B-11–16.

Moreno, A., N. Myller, E. Sutinen (2004a). Collaborative program visualization with woven stories and Jeliot 3. In *Proceedings of the IADIS International Conference on Web Based Communities 2004*. Lisbon, Portugal. pp. 482–485.

Moreno, A., N. Myller, E. Sutinen (2004b). JeCo: a collaborative learning tool for programming. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. Rome, Italy. pp. 26–29.

Moreno, A., N. Myller, E. Sutinen, M. Ben-Ari (2004c). Visualizing programs with Jeliot 3. In *Proceedings of the International Working Conference on Advanced Visual Interfaces AVI 2004*. Gallipoli (Lecce), Italy. pp. 373–376.

Moreno, A., N. Myller, E. Sutinen (2004b). JeCo: a collaborative learning tool for programming. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. Rome, Italy. pp. 26–29.

Nagappan, N., L.A. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, S. Balik (2003). Improving the CS1 experience with pair programming. *SIGCSE*, 359–362.

Nokelainen, P., J. Kurhila, M. Miettinen, P. Floren, H. Tirri (2003). Evaluating the role of a shared document-based annotation tool in learner-centered collaborative learning. In *Advanced Learning Technologies, Proceedings of the 3rd IEEE International Conference*. pp. 200–203.

Novak, J.D., and B.B. Gowin (1984). *Learning How to Learn*. Cambridge University Press, New York.

Nuutinen, J. (2006). Designing a computer supported collaborative mindtool: woven stories. *Licentiate Thesis* (Manuscript in preparation).

Ratcliffe, M.B., and L.A. Thomas (2004). Understanding our students: incorporating the results of several experiments into a student learning environment. In *16th Workshop of the Psychology of Programming Interest Group*. Carlow, Ireland. pp. 10–20.

Stotts, P.D., L.A. Williams, N. Nagappan, P. Baheti, D.Jen, A. Jackson (2003). Virtual teaming: experiments and experiences with distributed pair programming. In F. Maurer and D. Wells (Eds.) *XP/Agile Universe, Lecture Notes in Computer Science*, **2753**, pp. 129–141.

Williams, L.A., C. McDowell, N. Nagappan, J. Fernald, L.L. Werner (2003). Building pair programming knowledge through a family of experiments. *ISESE*, 143–153.

N. Myller has a master's degree in mathematics and he has been working with educational technology since 2001. He has worked in number of interdisciplinary international research projects. He is currently employed as a senior assistant at the Deparment of Computer Science at the University of Joensuu and studying for his PhD under supervision of professor Erkki Sutinen (University of Joensuu, Finland) and Dr. Piet Kommers (University of Twente, the Netherlands). His research topic is collaborative editors and mindtools and especially a software called Woven Stories.

J. Nuutinen received his BSc in 2003 and MSc in 2004 both from the Department of Computer Science at University of Joensuu in a record time of 2.5 years from starting. Currently, he is studying for his PhD under supervision of prof. Erkki Sutinen (University of Joensuu) and prof. Mordechai Ben-Ari (Weizmann Institute, Israel) and the expected graduation is in 2007. His research interests lie in the fields of visualization and concretization technologies, CSCL, information retrieval, computer ethics and adaptive systems. He has published more than 30 papers in international journals and conferences.

Priemonė „JeCo“: programų vizualizavimo ir fabulos dėstymo jungtis

Niko MYLLER, Jussi NUUTINEN

Straipsnyje supažindinama su programavimo mokymo priemone „Jeliot Collaboratively“ arba „JeCo“. „Jeliot Collaboratively“ apima programų vizualizavimo priemonę „Java“ programoms „Jeliot 3“ ir interaktyvią priemonę „Woven Stories“, skirtą kuriantiems rašinius autoriams. Straipsnyje aprašomos šių sistemų galimybės ir supažindinama su jų panaudojimo mokymesi būdais. Be to, brėžiamos kryptys ateičiai – siekiama išplėsti „JeCo“ galimo panaudojimo galimybes.