

COLLABORATIVE PROGRAM VISUALIZATION WITH WOVEN STORIES AND JELIOT 3

Andrés Moreno, Niko Myller and Erkki Sutinen

University of Joensuu

P.O. BOX 111

FIN-80101

Finland

{amoreno,nmyller,sutinen}@cs.joensuu.fi

ABSTRACT

The novel concept of collaborative program visualization combines individually oriented cognitive tools with a collaborative environment. The concept is based on the socio-cultural constructivism in which collective actions are in the key role. In this context the learning process is acculturation into an established community of practice where the subjective feelings and knowledge of the participants is transformed into commonly accepted knowledge of the community. This process is supported by the environment called JeCo that supports both synchronous and asynchronous collaborative tools. Users of this new platform will collaborate in developing programs and solving together different programming tasks. The platform structure is the result of the union of two already developed tools: Woven Stories, which provides the collaborative environment, and Jeliot 3, which animates the programs.

KEYWORDS

Software visualization, program visualization, woven stories, collaborative tools

1. INTRODUCTION

Algorithm animation and program visualization are active fields of study but the focus of the research is on the learning of individuals (Diehl, 2002). A large number of distance education courses are organized but the tools do not support the collaboration between students in courses of programming. Furthermore, the novice programmers seem to have a number of problems during the distance programming courses (Meisalo et al., 2003). We want to address these issues and introduce a novel concept of collaborative program visualization. In addition, we give a description of the system that could support this type of collaborations, called JeCo (Jeliot Collaboratively).

The paper is organized as follows. Section 2 introduces the concept of collaborative program visualization and motivations behind it. Sections 3 and 4 introduce the systems that are combined to form the new system, JeCo that is described in Section 5. Section 6 contains the concluding remarks and future directions.

2. COLLABORATIVE PROGRAM VISUALIZATION

The novel idea is to combine program visualization and collaboration tools such as collaborative authoring applications to form software that can support both collaboration and visualization of programs, *collaborative program visualization*. In general these kinds of tools can support discussion about programs and learning of programming because they create a context for the peer-to-peer interpretation. This field of study is new and there are few, if any, contributions (Diehl, 2002).

In the first program and algorithm visualization systems the user was made to watch the visualization and he/she had not the possibility to interact with the visualization. These kinds of systems supported learning in a *behavioristic* (Burton et al., 1996) way where the student tries to acquire as much knowledge as possible (response) from the expert made animations (stimulus).

The visualization systems developed during the past ten years have been cognitive tools with a constructivist orientation (Duffy and Cunningham, 1996). The systems support individual learning in which the learner tries to make sense of the algorithms by creating animations by herself and constructs new knowledge by interacting with the system. This means that the instructors and peer learners only serve as stimuli for the learner and arouse her to learn.

According to the recent theories of *socio-cultural constructivism* (Duffy and Cunningham, 1996) a learner should have a possibility to interact with other learners and to be an active member in a learning community. This led us to the idea of collaborative program visualization because the learning should not be individual but inter-subjective where the subjective ideas and knowledge of the learners are shared and developed further so that they together can create less subjective or commonly accepted knowledge.

From this point of view collaborative program visualization is the means to support socio-cultural constructivist learning in communities of programming learners. The knowledge is first contributed and shared by sending the program codes and visualization of them to the forum. From this initial information new knowledge is formed through collaboration and group work. The collaboration is supported with the tools designed especially for the purpose, for instance by comments, chats and visualizations.

3. JELIOT 3

Jeliot 3 (Moreno & Myller, 2003) is a new version of the program visualization tools family called Jeliot (Ben-Ari et al., 2002). They are intended to be used by novices learning to program. Jeliot 3 visualizes the execution of a Java program by showing the current state of the program (e.g. methods, variables and objects) and animations of the expression evaluations. The animation window is divided into two panels. The animations are displayed on the left panel called theater and the source code on the right panel. Methods and their variables are shown on the left side of theater. Expressions (e.g. $a+b$) are evaluated next to the method frames and the variable values are moved there to show the evaluation. Assignments move the values from the expression area back to the variables. Objects and arrays are displayed at the bottom of the theater; however, the references connect the variables and the reference types to explain the reference semantics.

Jeliot 3 uses two separate systems to produce the animation. One is the user interface and visualization engine of Jeliot 3 and the other is Java interpreter called DynamicJava. The communication between these two separate systems is achieved with an intermediate language that is plain ASCII and can be saved as such. In this way the design is modular and separable.

Currently Jeliot 3 animates a large subset of Java programs with features like values and variables of the primitive data type (e.g. int, boolean, char) and strings, one dimensional arrays of primitive type, expressions such as operations and assignments, control structures, error reporting, object-oriented programming and I/O operations.

Its uses as a teaching tool are multiple:

- The *lecturer* can use Jeliot 3 as a part of the lecture material. She can explain different concepts of programming through Jeliot animations. Thus *students* can create the correct relationship between the animation and the concept, and apply it later with reduced possibility of misunderstanding (Ben-Bassat Levy et al., 2003).
- *Students* may use Jeliot 3 by themselves after the lectures and do the assignments with it.
- Jeliot 3 can be used in an interactive *laboratory session* where students may utilize their recently acquired knowledge by writing programs and debugging them through Jeliot.
- Finally, Jeliot 3 provides a tool that can aid in courses when external help is not available (e.g. in distance education). Its visualization paradigm creates a *reference model* that can be used to explain problems and thus it eases the communication and creates vocabulary between students and teacher when difficulties come up (Ben-Bassat Levy et al., 2003).

Especially, the latter usage is going to be supported with the new tool introduced in this paper. The key features that make the integration to other systems possible are the separation between the visualization engine and the interpreter and the intermediate code that can be saved and the program can be thus run even without a new interpretation of the original program.

4. WOVEN STORIES

Woven Stories (Gerdt et al., 2001) is a co-authoring tool that represents the document history as a graph where documents or sections of documents are the nodes and the connections or derivations between the documents are the edges. Users can create new nodes and new edges between the existing nodes and his/her nodes and view nodes authored by other users. Moreover, there is a chat program integrated to the platform where students can discuss the documents with each other. Thus the platform supports both synchronous and asynchronous group work.

Users connect to a Woven Stories server with a client program and they see a user interface similar to that illustrated in Figure 2 without the Jeliot window in the left bottom corner of the picture. There is the structured view of the document history on the left hand side of the screen. The right panel contains the document view and below it is the chat.

The structure of Woven Stories is also modular and permits new integrations to the system. Especially, the server is generic when handling the user documents and that permits basically any kind of document to be loaded to the server. This means that the client then must be modified to understand a new kind of content. As an example of the modularity a business strategy planner was produced with slight modification to the original system (Nuutinen et al, 2003).

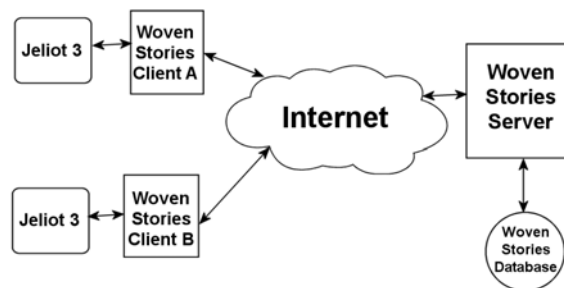


Figure 1: The functional structure of the JeCo system.

5. COMBINING THE SYSTEMS: JECO

The new system called JeCo, Jeliot Collaboratively, is based on the systems presented above. Both of these systems are designed to be flexible for new integrations and modifications. The new system lets the user to visualize programs, send these visualizations to other users of JeCo, comment other users' programs and visualizations and chat with other users. To sum up, JeCo supports the collaborative program visualization as described in Section 2.

Description of the JeCo system and its usage: The system is based on the framework of Woven Stories where the clients are connected to the server through the Internet (see Figure 1). All the documents and their connections to each other are stored to the database and then broadcasted to all the clients that visualize the changes in the structure view. Any user (e.g. client A) can send to the server a document that consists of a message, program's source and the corresponding intermediate code produced by running the program through Jeliot 3 that is integrated into the client software. If another user (e.g. client B) wants to see the same animation and to comment on it then she can request the document. After loading the document the animation is shown in a new window. The user sees a similar view as shown in Figure 2. In Figure 2 the currently viewed document is bordered with red rectangle and documents containing the intermediate code for Jeliot animation are indicated with the icon on the right side of the document rectangle.

The JeCo system can be especially useful in distance education courses of programming. It gives the students the possibility to collaborate with each other and at the same time use the tools that are made for the purpose. As found by Ben-Bassat Levy et al. (2003) Jeliot creates a vocabulary related to the programming concepts and thus creates the context for the discussion. In the case of JeCo, this vocabulary will grow into a truly inter-subjective set of shared concepts. However, this kind of forum can also support regular lecture courses as a platform for exercises, assignments and group work.

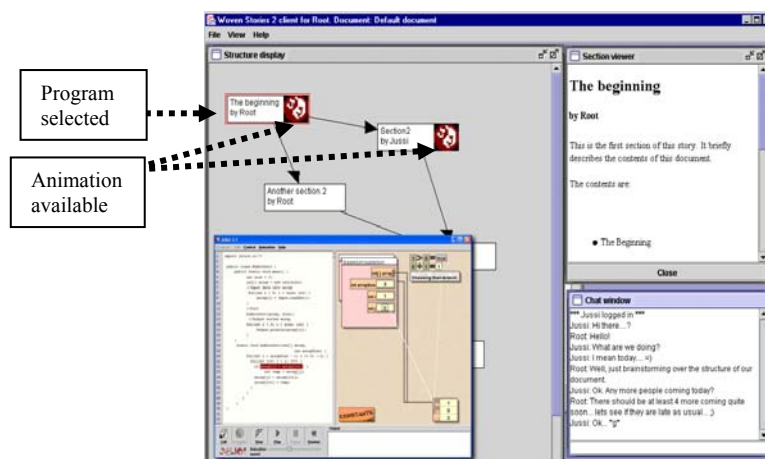


Figure 2: The prototype user interface of the JeCo.

6. CONCLUSION

In this paper we introduced the novel concept of collaborative program visualization. In collaborative program visualization the key point is that not only individual learners can see the same animation but the whole community of learners has the possibility to see and develop together the same animation even with the same input data. Furthermore, they can communicate and share their ideas with each other easily and in connection with programs and the corresponding animations. This transforms the learning process into a socio-culturally flavored journey of constructing inter-subjective ideas and concepts.

We also described an architecture of the system that will support the collaborative program visualization. The applications used for the new system are flexible and modular making the development of the system quite simple. The tool will be implemented during 2004 spring term and an evaluation of it can be carried out during next autumn semester.

REFERENCES

- Ben-Ari, M. et al., 2002. Perspectives on Program Animation with Jeliot. *Software Visualization: International Seminar, Lecture Notes in Computer Science 2269*, Dagstuhl Castle, Germany, pp 31-45.
- Ben-Bassat Levy, R. et al., 2003. The Jeliot 2000 program animation system. *Computers & Education* Vol. 40 No. 1, pp. 15-21.
- Burton, J.K. et al., 1996. Behaviorism and Instructional Technology. (Jonassen, D.H. Ed.) *Handbook of Research for Educational Communications and Technology*. Macmillan, N. York.
- Diehl, S. (Ed.), 2002, *Software Visualization: International Seminar, Lecture Notes in Computer Science 2269*, Springer-Verlag, Germany.
- Duffy, T.M. and Cunningham, D.J., 1996. Constructivism: Implications for the design and delivery of instruction. (Jonassen, D.H. Ed.) *Handbook of Research for Educational Communications and Technology*. Macmillan, N. York.
- Gerdt, P. et al., 2001. Woven Stories as a Cognitive Tool. *Cognitive Technology, Lecture Notes in Artificial Intelligence 2117*, pp. 233-247.
- Meisalo, V. et al., 2003. Choosing appropriate methods for evaluating and improving the learning process in distance programming courses. In the *Proceedings of the 33rd ASE/IEEE Frontiers in Education Conference (FIE2003)*, pp. T2B-11-16.
- Moreno, A. and Myller, N., 2003. Producing an Educationally Effective and Usable Tool for Learning, The Case Of Jeliot Family. In the *Proceedings of International Conference on Networked e-learning for European Universities*. Granada, Spain.
- Nuutinen, J. et al., 2003. Strategists' Learning Space. To appear in the *Proceedings of the IASTED International Conference on Web-Based Education*, Innsbruck, Austria.