

# JeCo, a Collaborative Learning Tool for Programming

Andrés Moreno, Niko Myller, and Erkki Sutinen  
University of Joensuu  
Computer Science Department  
P.O. Box 111, FI-80101 Joensuu, Finland  
{amoreno, nmyller, sutinen}@cs.joensuu.fi

## Abstract

*The pair-programming activities in a classroom are found to support learning of the novice students during the first programming courses. However, there are very few tools that could support pair or group programming in distance education courses. We present a concept of collaborative program visualization and a tool to realize it, called JeCo that can help the students to work together on a platform that supports both collaborative authoring and program visualization.*

## 1. Introduction

There is a growing interest in pair-programming in classroom settings. Furthermore, a large base of positive results in empirical experiments supports the use of pair-programming in programming courses [9].

At the same time, the students are still struggling in the programming course and especially in the distance education courses of programming [13]. Similar kind of pair- or group-programming could be introduced to distance education programming courses. However, new tools are needed to accomplish this task as the interaction between the students should be supported.

Algorithm and program visualization have a potential to aid students learning to program. The studies have given contradictory results. The key feature for success seems to be the level of engagement of the students during the visualization activities [2, 6].

With our approach we aim to increase the added value of program visualization tool in an educational context by merging it with a collaborative tool. This means that the visualization is used to support the collaboration and to create a context in which the collaboration takes place. In our case we have combined *Jeliot 3* [11], a program animation tool, intended for novices, visualization object-oriented Java programs, and *Woven Stories* [5], a co-authoring and collabora-

tion tool for collaborative writing. The integration of these systems has resulted into an environment called *JeCo* (Jeliot Collaborative) [10].

## 2. Related Work

Currently, we are aware of two similar systems. The first one is made to support assembly programming where students can send the program codes the each other and to the teacher through email [7]. Our system is aimed to be more explicit in the collaboration in order to show the whole group and is visible also after the collaboration and have a structure.

Another tool focused on group work and programming is *Gild* [12], a plug-in for the full featured IDE, Eclipse. This tool modifies Eclipse in order to make it more convenient for novices. Eclipse, with the *Gild* plug-in installed, contains several items a learner may need during the programming process (e.g. a discussion board, a chat and a debugger).

## 3. Collaborative Program Visualization

The collaborative program visualization combines program visualization and collaboration tools such as collaborative authoring applications to form a piece of software that can support both collaboration and visualization of programs. This kind of combination of tools can create a context where students learning to program can work together.

According to the theories of *socio-cultural constructivism* [1, 4] a learner should have a possibility to interact with other peers and to be an active member in a learning community or community of practice. The collaborative program visualization helps a learner of programming to interact with other peer learners and form a learning community. In this community, the intra-subjective knowledge becomes inter-subjective where the subjective ideas and knowledge of the learners are shared and developed fur-

ther through collaboration so that the knowledge inside the community is created and learning happens.

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

These features can also support the pair-programming in distance education but in a different manner than that of *eXtreme Programming* (XP). In this context, the participants can be both 'drivers' and 'observers' by writing own code and reviewing the code of the other participant. The program visualization helps developing as well as reviewing process by debugging support but also by creating the context and vocabulary for the conversation and collaboration [3].

## 4. JeCo

### 4.1. Description of the System

The system supporting collaborative program visualization called Jeco is an integration of two existing systems, Jeliot 3 and Woven Stories.

Jeliot 3 animates the execution of the program, showing how expressions are evaluated and how new objects and arrays are created. There are several reasons why this system was suitable for the use in this project: the proved usability and the provided vocabulary for the students [2], the automatic visualization generation and the modularity of the system for easy integration. The design of Jeliot 3 facilitates its integration also with other visualization tools, and gives a user a possibility to create new views of the same program.

Woven Stories 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.

Woven Stories is modular which permits new integrations to the system. Especially, the server is generic and can handle different kinds of user documents. This means that only the client must be modified to understand a new kind of content. Figure 1 explains the structure of JeCo system.

As was already discussed and can be seen from the figure only the client is needed to be modified to make the Woven Stories client to run Jeliot 3 when needed with relative ease.

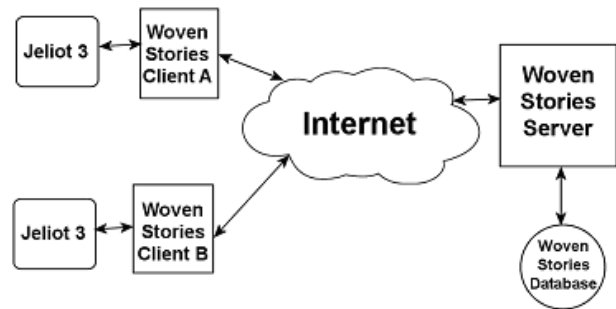


Figure 1. The structure of the JeCo system

### 4.2. Utilization of the System

When a user connects to a Woven Stories server with a client program and she sees 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. Furthermore, the new client application lets the user to visualize programs, send the source code of the program to other users of JeCo and comment other users' programs and visualizations.

The system is based on a framework build for Woven Stories where the clients are connected to the server through the Internet (see Figure 1). All documents and their connections to each other are stored to a database and broadcasted to all clients that visualize the changes in the structure view. Any user (e.g. client A) can send to the server a section that consists of a message and a program's source code. If another user (e.g. client B) wants to see the animation of the program code sent by client A and to comment on it then she can request the section. After loading the section 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 section is bordered with red rectangle and sections containing the source code for Jeliot 3 animation are indicated with the icon on the right side of the section 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. Jeliot creates a vocabulary related to the programming concepts and thus creates the context for the discussion [2]. In the case of JeCo, this vocabulary will grow into an 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.

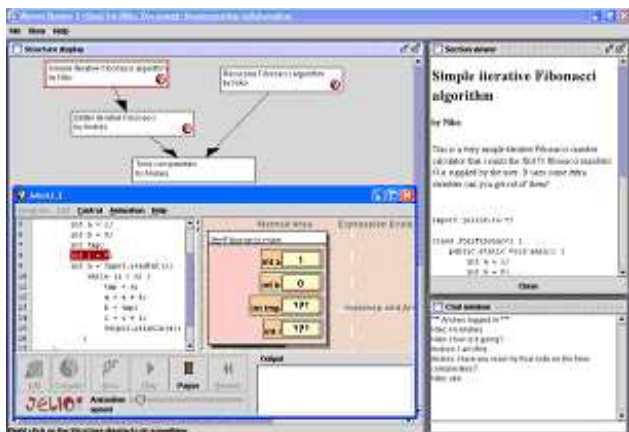


Figure 2. The user interface of JeCo

## 5. Future Work and Conclusions

We have presented a concept of collaborative program visualization that can especially help the novice students to learn programming through collaboration in the distance education courses. We have also presented a tool for the purpose called JeCo that combines two existing systems. However, there is still much work to be done to make the collaboration as flexible as possible and support the group processes in programming.

The group work can be supported with other groupware tools just as annotation possibilities. The programming can be supported, for example, with syntax highlighting. The visualization and code transmission should be made easier and streaming of the visualization should be made possible. Furthermore, there should be a support for editing a source code in a group.

To support more advanced courses in OOP, the integration of a tool like BlueJ [8] into JeCo, could help the design of the create their own class interfaces and propose suitable connections with already defined classes or group mate's one. This way the application could be built up.

Finally, we are planning an experiment during which the students in a distance programming course use the tool to do the assignments in pairs or groups and the utilization of the tool will be analyzed.

## References

[1] M. Ben-Ari. Situated Learning in Computer Science Education. *Computer Science Education*, 2004. to appear.

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

[3] E. F. Churchill, J. Trevor, S. Bly, L. Nelson, and D. Cubranic. Anchored conversations: chatting in the context of a document. In *CHI 2000 Conference on Human Factors in Computing Systems*, pages 454–461, The Hague, The Netherlands, 2000.

[4] T. M. Duffy and D. J. Cunningham. Constructivism: Implications for the design and delivery of instruction. In *Handbook of Research for Educational Communications and Technology*. Macmillan, New York, USA, 2001.

[5] P. Gerdt, P. Kommers, C. K. Looi, and E. Sutinen. Woven stories as a cognitive tool. In *Cognitive Technology, LNAI 2117*, pages 233–247, 2001.

[6] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko. A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, 2002.

[7] Y. Imai and S. Tomita. A web-based education tool for collaborative learning of assembly programming. In *IADIS International Conference WWW/Internet 2003*, pages 703–710, 2003.

[8] M. Kölling, B. Quig, A. Patterson, and J. Rosenberg. The bluej system and its pedagogy. *Journal of Computer Science Education*, 13(4), 2003.

[9] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald. The impact of pair programming on student performance, perception and persistence. In *Proceedings of the 25th international conference on Software engineering*, pages 602–607. IEEE Computer Society, 2003.

[10] A. Moreno, N. Myller, and E. Sutinen. Collaborative program visualization with woven stories and jeliot 3. In *International Conference on Web Based Communities*, pages 482–485, Lisbon, Portugal, March 2004.

[11] A. Moreno, N. Myller, E. Sutinen, and M. Ben-Ari. Visualizing programs with jeliot 3. In *International Working Conference on Advanced Visual Interfaces AVI 2004*, Gallipoli (Lecce), Italy, May 2004.

[12] M.-A. Storey, D. Damian, J. Michaud, D. Myers, M. Mindel, D. German, M. Sanseverino, and E. Hargreaves. Improving the usability of Eclipse for novice programmers. In *Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange*, pages 35–39. ACM Press, 2003.

[13] E. Sutinen and S. Torvinen. The candle scheme for creating an on-line computer science program: experiences and vision. *Informatics in education*, 2(1):93–102, January 2003. [http://www.vtex.lt/informatics\\_in\\_education/htm/INFE009.htm](http://www.vtex.lt/informatics_in_education/htm/INFE009.htm) (accessed 10.6.2004).