# MULTI-AGENT EDUCATIONAL SYSTEM FOR PROGRAM VISUALIZATION

R. Bednarik[1], M. Joy[2], A. Moreno[1], N. Myller[1], S. Sun[2], E. Sutinen[1]
[1]Department of Computer Science, University of Joensuu, Finland
PO Box 111, FI-80101, Joensuu, Finland
[2]Department of Computer Science, University of Warwick, UK
Coventry CV4 7AL
*bednarik@cs.joensuu.fi*

*Abstract*- **We propose a multi-agent educational system to support programming learning with program visualizations. Current program visualization tools lack sensitivity to learners' growth of expertise and other important contextual factors influencing the learning process. Combining an agent-based intelligent system with a program visualization environment is a potential solution to the problem. We propose the architecture of such a system and discuss its potentials and drawbacks.**

## I. INTRODUCTION

Program visualization (PV) systems have been used to support the understanding of programming concepts in novice-level courses. Often using animations, these systems aim to support students in understanding the control and/or data flows of an algorithm or a program. However, empirical evaluations of these systems have not uniformly shown a positive effect on students' understanding [14]. One of the reasons is that visualizations were designed to fit the mental models of experts instead of those of novices. Another reason is that they do not offer any support for the learner's learning style.

In our previous work, we have performed a critical analysis of current PV systems [1] and have developed a multi-agent educational system [27][28]. In this paper, we review the present state of the research into agent-based educational systems and discuss current PV tools. We argue that the lack of contextual sensitivity can be a reason for a particular failure of previous PV systems. To overcome this problem, we present the design of an intelligent, agent-based tool to support program comprehension.

The rest of this paper is organized as follows: in the next section we introduce the idea of using agent-based systems to support adaptivity in education, and then we discuss some aspects of current PV tools. A multi-agent system is described, and we then discuss how the agent-based systems are linked with PV tools. Finally, we discuss the directions of future research of smart educational tools and present final conclusions.

## II. AGENT-BASED EDUCATIONAL ADAPTIVE SYSTEMS

### *A. Agent Technology*

Depending on the roles that agents take in their deployed environments, their abilities may vary significantly. However, we can still identify essential and commonly agreed properties of agents, namely: autonomy, proactiveness, responsivity, and adaptivity. Additionally, agents should also know users' preferences and tailor their interactions to reflect these [15]. It is generally accepted that an agent is an entity that is capable of carrying out flexible autonomous activities in an intelligent manner to accomplish tasks that meet its design objectives, without direct and constant intervention and guidance of humans.

Multi-agent systems contain many agents that communicate with each other. Each agent has control over certain parts of the environment, so they are designed and implemented as a collection of individual interacting agents. Luck *et al.* [18] remark that "Multi-agent systems provide a natural basis for training decision makers in complex decision making domains [in education and training]". Furthermore, multi-agent systems can substantially contain the "spread of uncertainty", since agents typically process information locally [12]. In the context of our education system architecture, agents provide means to manage the complexity and uncertainty of the domain.

### *B. Agent-based Educational Adaptive Systems*

In the context of adaptive education, agent technology can provide a dynamic adaptation not only of the domain knowledge but also of the behaviour of individual learners, and has already been used in a number of educational tools. However, most systems incorporating agent technology, such as [3][6][22][24][26], have decoupled the agents from the pedagogical foundations of the system. Existing systems tend to emphasise a particular aspect, such as training, group work, or human resources requirements. Beer and Whatley [3] report the initial design of agents to support students undertaking group projects in health care education. For each group of students, they provide a local agent to monitor the project, and enhance the communication between members of the group. The use of agents is emphasized as providing dynamic support for synchronous collaboration.

Each of the current approaches has its individual ways of organizing the learning materials, and few have considered the effect of different learning styles. For example, in Shang *et al.*'s system [26], the student's learning styles are stored in personal agents at the beginning of a student's use of the system, and are not changed dynamically during the learning process.

However, learning styles will change during the time students are using the system. In our proposed multi-agent system, students' learning styles are updated during the learning process. Shang *et al.* [26] organize agents according to different courses, while Boicu *et al.* [6] use agents that are implemented according to specific learning topics. In our system, however, the agents are decomposed by their function in the teaching and learning process. The use of learning objects in such systems is rare, although the technology has begun to be used in non-adaptive training software. Garro and Palopoli's [11] system is designed to assist finding appropriate employees and measuring the skill gaps between the employee and the requirements of the organisation from a human resources perspective.

## III. Program Visualization

Programming is certainly a cognitively demanding problem-solving activity requiring knowledge and skills. Central to programming is program comprehension, the ability to understand programs written by oneself and by others. Novice programmers often have no idea about what are important aspects of programs; they lack the domain knowledge and skills. As in other fields, to become a successful programmer, the student has to gain competence by deliberate practice [9]. However, as the programming domain involves complex relations and dependencies often unseen by novice programmers, teaching and learning programming need to be supported by proper tools, which assist students in creating viable mental models of programs and programming concepts.

Many PV tools have been developed to support the teaching and learning of programming, however evaluations of such tools are not uniformly positive [14]. Beside methodological problems of empirical evaluations [20], the reasons for these findings may derive from the facts that (i) the tools and visualizations are constructed in a uniform fashion and (ii) the visualization systems do not allow for enough interaction between the learners and the visualization. It has been claimed [4] that visualization design has to be appropriate to the learner's experiences and learning style. Furthermore, learners have to be taught to *use* the visualization. A recent analysis of the current PV tools performed by Bednarik *et al.* [1] suggests that some of the mentioned problems may be overcome by integrating PV tools with intelligent adaptive frameworks. To demonstrate how the possibilities of a smart learning environment could be exploited using a current system in use, we selected an existing PV tool, Jeliot 3.

### A. Smart PV systems

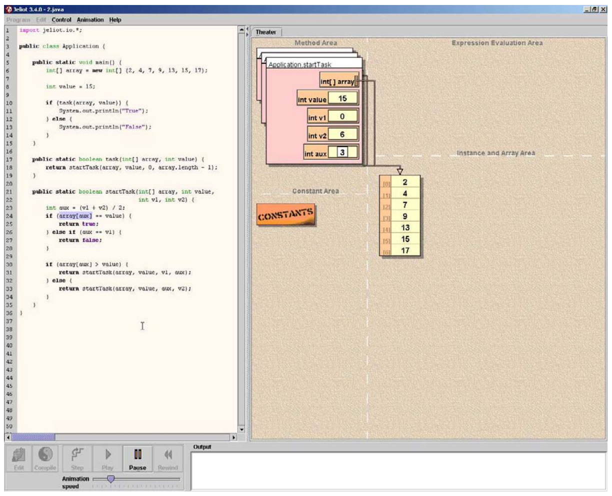
As PV tools have been researched for several decades, many tools have been developed. For example, Javavis [23], Jive [13], and DDD [30], are tools that automatically visualize the programs data flow and part of the control flow during program execution, while the visualizations in Dynalab [7] and PlanAni [25] concentrate on variables.

In the context of PV, there have been a few attempts to support the learner with agents and adaptivity. Seal [19] makes use of social agents to support the understanding of the PVs. Social agents discuss with the user how the visualization proceeds and whether the results of the visualization are as expected. Moreover, in problem situations the agent tries to explain what went wrong. WADEIn [8] adapts the level of details in the visualization to the understanding of the learner. The learner's knowledge in terms of elementary units of the programming language is monitored and the information is stored into a centralized user model repository. The adaptive system adjusts the steps in the animation of an algorithm to focus on elements that are not yet understood as well as the other parts. However, both of these systems are very limited in the scope of use. WADEIn can be used only to visualize expression evaluation in C and Seal to visualize a very limited pseudo-code language.

A different approach is taken in Jeliot I [29] where the visualization is made adaptable and users can modify the visualizations to make it better fit into their mental model of the program. This can be achieved by changing the visualization parameters of the visual objects (i.e. variables). For example, the shape, color, and style used in the visualization can be changed. This is definitely one way to support learners, but may be found too difficult to grasp by the novices in programming [17].

### B. Jeliot 3

Jeliot 3 (see Fig. 1) is a typical PV tool developed to aid novices in creating of viable mental models of computation [19]. The tool creates dynamic visualizations of user-written Java programs, supports object-oriented programs, and allows users to pause and step through the animation or to set break points. Jeliot 3 and its predecessors have been evaluated in several empirical experiments in teaching [17][5] and in laboratory settings [2]. Due to the open architecture, Jeliot 3 is a suitable up-to-date tool that easily allows for integration with other platforms and environments.


```
import jeliot.io.*;

public class Application {

    public static void main() {
        int[] array = new int[] {2, 4, 7, 9, 13, 15, 17};

        int value = 15;

        if (task(array, value)) {
            System.out.println("True");
        } else {
            System.out.println("False");
        }
    }

    public static boolean task(int[] array, int value) {
        return startTask(array, value, 0, array.length - 1);
    }

    public static boolean startTask(int[] array, int value,
                                    int v1, int v2) {
        int aux = (v1 + v2) / 2;
        if (array[aux] == value) {
            return true;
        } else if (aux == v1) {
            return false;
        }

        if (array[aux] > value) {
            return startTask(array, value, v1, aux);
        } else {
            return startTask(array, value, aux, v2);
        }
    }
}
```

Fig. 1. Interface of Jeliot 3.

## IV. THE MULTI-AGENT SYSTEM

Agent-based technology has been adopted by few PV systems, and the use of such technology is a means of delivering PV systems within an intelligent adaptive framework.

A multi-agent system is a system that contains several software agents that communicate with each other. Each agent has control over certain parts of the environment, and they are designed and implemented as a collection of individual interacting agents. Our system is functionally constructed by five agents; the Student Agent, Record Agent, Modelling Agent, Learning Object Agent, and Evaluation Agent. Each agent is designed to satisfy a certain functional requirement that contributes to the purpose of the overall learning system that is to provide dynamic and adaptive learning materials to individual users.

The Student Agent is responsible for communicating with students; the Record Agent maintains information about each student; the Modelling Agent creates models of students' skills and learning objectives; the Learning Object Agent manages the set of learning objects; and the Evaluation Agent ensures that learning objects are presented in individual and adaptive learning paths to each student. In the remainder of this section, we describe some of these agents in more detail, and discuss how students interact with the system.

Among the five functioned agents in the system, the Student Agent and the Learning Object Agent are the direct interface to PV system.

### A. *The Student Agent*

The Student Agent is responsible for communicating with students, and provides the interface between the system and human users. The function of the agent is to fulfil the communication and data collection requirements, and to provide information from the user to other agents in the system.

When a student first logs in to the system, the Student Agent provides an initial questionnaire to ascertain the student's knowledge level, and to obtain information about their learning requirements (such as module details or the specific subject that the student wishes to study). During the time that the student is logged in to the system, the Student Agent records all of their actions, including the time they spend engaged in each activity presented to them, clicking times, whether they are active or not, etc. The data describing the interaction between the student and the system are of great importance to allow for an accurate modelling. For that reason, an advanced interaction device – such as an eye tracker – should be used, so it can provide the Student Agent with rich data.

The Student Agent has a clear functional division, and the BDI notions of beliefs, desires, and intentions can naturally represent its functional operation. Its knowledge includes students' preferences, the available learning materials in the system, and students' knowledge levels – provided by the students themselves and updated by the system. This knowledge can be mapped to a set of beliefs, and the agent's interactions with students and the other system components can be mapped to its desires. According to its beliefs certain desires will be triggered, and plans for these are adopted as intentions in the BDI-based implementation. For example, according to the knowledge level provided by a student, appropriate learning materials can be sent. Beliefs are partially based on the information provided by students, and so they may not be completely accurate (e.g. students might exaggerate their abilities). Thus, not all of the desires can necessarily be achieved, since the choice of intention may be based on inaccurate information.

### B. *The Learning Object Agent*

The Learning Object Agent manages the learning objects, which are organized according to the learning style scheme. In response to instructions from the Modelling Agent, the Learning Object Agent provides different learning style students with relevant learning objects. The function of the Learning Object Agent is to organise the adaptive learning materials for users based on the information that the system has collected.

The Learning Object Agent is a hybrid agent, and has an architecture in which its subsystems are arranged into a hierarchy of layers as shown in Fig. 2. The Learning Object Agent communicates with the other agents through its communication layer. Decisions are sent to the learning object's management layer, which is in charge of managing all of the learning objects in its repository. The learning objects repository is organized into different levels, according to the learning style scheme. Finally, the learning objects management layer selects a series of learning objects, which are transmitted to the Evaluation Agent through the communication layer.

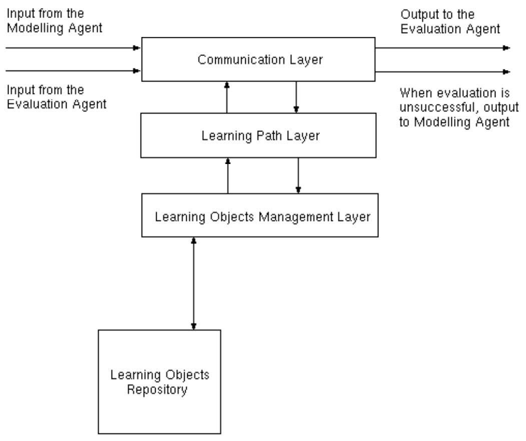

Input from the Modelling Agent
Output to the Evaluation Agent
Communication Layer
Input from the Evaluation Agent
When evaluation is unsuccessful, output to Modelling Agent
Learning Path Layer
Learning Objects Management Layer
Learning Objects Repository

Fig. 2. Learning Object Agent.

The PV system is considered as a system to transform the learning materials (i.e. programs) stored in the repository into learning objects, since the purpose of the PV tool within the whole system is to create new learning objects. In this case, we use Jeliot 3 as the PV tool, but our agent architecture is not dependent on Jeliot 3. A collection of learning objects, namely program visualizations, would be constructed by selecting an appropriate set of programs or code fragments which could be used as data for the PV system. Each such piece of code would then yield a learning object – the PV system given the code as data – that would be managed by the Learning Object Agent.

The learning path layer adopts a learning style model to organise learning objects to fulfil different students' requirements, and which categorizes the learning objects in the repository. In our current architecture, the learning style model is the Felder-Silverman Learning Style Model [10], which categorizes learning schemes according to four distinct dimensions, each of which is represented on a scale of five points:

- "*sensing* (concrete thinker, practical, oriented toward facts and procedures) or *intuitive* (abstract thinker, innovative, oriented toward theories and underlying meanings);
- *visual* (prefer visual representations of presented material, such as pictures, diagrams, flow charts) *or verbal* (prefer written and spoken explanations);
- *active* (learn by trying things out, enjoying working in groups) or *reflective* (learn by thinking things through, prefer working alone or with a single familiar partner);
- *sequential* (linear thinking process, learn in small incremental steps) or *global* (holistic thinking process, learn in large leaps). "[10]

Organization of the learning materials as learning objects, based on a pedagogic learning style scheme in an agent environment, is a characteristic of this architecture that distinguishes it from existing pedagogic agent-based systems.

## V. PROPOSED SOLUTION

Fig. 3 depicts the overall structure of the proposed system with examples of data exchanged between the modules.

### *A. Learning Objects and Learning Object Metadata*

The system maintains a repository of learning objects, including instances of programs suitable for animation by the PV tool in addition to other learning objects, such as quizzes, explanations, and other resources that would not be suitable for presentation by the PV tool.

In addition to basic information such as author, date, etc., the learning object metadata incorporates a *dimension description*, suggesting for each of the four Felder-Silverman learning style dimensions the extent of each object's suitability on a five-point scale, and also incorporates the key concepts taught by the learning object itself, as well as the prerequisites (if applicable). In the case of visualizations presented to the user by the PV tool, we may expect that the "visual" dimension would normally be strong, so that visualizations would be presented to users only if that was appropriate for their individual learning styles.

Learning objects from the PV system are categorized in advance, and stored in the repository, according to the learning style space. The Learning Object Agent handles presentation of these to the user. The PV system uses an ACL – Agent Communication Language – to communicate with the agent, and vice versa.

### *B. Using the System*

In the following, we describe a typical scenario of interactions between a student and the system. When a student first logs in to the system, the Student Agent enters into a dialogue with the student to ascertain the student's learning requirements. At the same time, the interaction device – e.g. an eye-tracker – starts tracking the interaction of the student with the system until the student logs out the system.

After initially analyzing the results of the dialogue by the Student Agent, the Record Agent is informed of the student's learning requirements together with a suggested knowledge level for the student. These items of information are recorded and then passed to the Modelling Agent, which then sends results and instructions to the Learning Object Agent. The Learning Object Agent in turn arranges the first batch of learning objects. For instance, if the learning requirement is some basic flow control concepts, then the learning objects will consist of material containing an *if* statement, *while* loop, etc. These learning objects are then sent to the Student Agent according to the results of the learning style analysis (which occurs in the learning path layer). The learning objects, as mentioned above, are organized according to the learning style scheme.

Considering the visualizations as instances of learning objects, Jeliot 3 provides a series of learning objects to the Learning Object agent. These learning objects are first sent to the Evaluation Agent, which checks the student's data from the Record Agent to evaluate whether the learning objects appear to be suitable for this student. If the evaluation is successful, the series of learning objects is sent to the Student Agent (and then to the student) and recorded by the Record Agent. Otherwise, the Evaluation Agent asks the Learning Object Agent to provide alternative learning objects. The purpose of using the Evaluation Agent is that, since the system is dynamically adaptive to individual students, so each learning cycle should be recorded and the data should be used to facilitate providing students with adaptive learning materials over the time.

After the student has used the learning objects, response data, containing student satisfaction, time spent, whether the student is comfortable with the learning material, are returned to the Student Agent, which transmits them to the Record Agent. Furthermore, during the use of the PV system, interaction data can be collected from the user and transferred to the Modelling Agent.

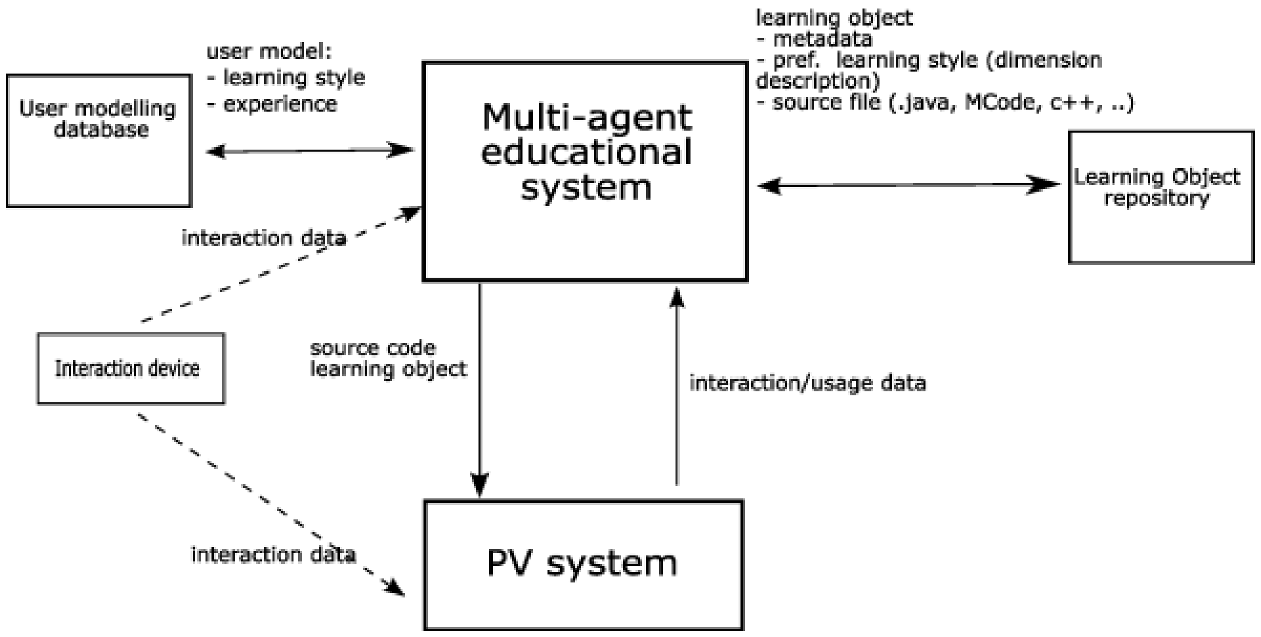


Fig. 3. Proposed framework.

This data consists of times viewing and points of pausing etc.

## VI. Future Work

The implementation of the proposed system requires defining interfaces to allow for communication between the parts of system. Once implemented, we plan to conduct series of experiments to evaluate the system. We also intend to conduct experiments where we compare the preferred learning styles and the use of visualization tool, in terms of interactions and gaze behaviour measured with an eye-tracker.

### A. *Interaction Patterns and Learning Style Preferences*

As it has been discussed above, the learning objects contain information about learning styles. To fully exploit the potentials of learning styles, the system should be able to provide such materials that best fit a current learner. A first step is to find out how the patterns of interaction are related to students with different learning style preferences.

After the initial differences are identified, the information will be used in the Evaluation Agent, which is in charge of ensuring that the learning materials sent to students are suitable for their learning style preferences.

### B. *Multi-agent adaptive system as part of learning-management systems*

Learning management systems (LMS) can provide a basic infrastructure for complete online education. Visualization tools have already been used together with LMS [16] as parts of programming courses. However, interaction between the systems has been only one way, the LMS sending programs to be visualized without feedback from the PV system. As further development the PV tool could update the LMS information with the interaction data gathered from the PV tool. We believe that a multi-agent educational system will provide the required functionality. Results of the analysis would update the contents of the online course for a particular user.

Making the multi-agent educational system for a PV a part of an LMS will help with the adoption of the proposed solution by educational institutions, where the use of powerful, but not smart LMS, such as Moodle, is common.

## VII. Conclusions

We have described our perspective on an intelligent tool for teaching programming by means of an agent-based adaptive PV tool. The suggested approach solves one of the problems identified in previous systems that have not used any intelligent techniques: proper educational materials are delivered to learners according to their needs. Our system will perform an analysis of learner's needs and interaction to provide an appropriate visualization to match with the learner's cognitive style, growth and abilities.

## References

[1] R. Bednarik, A. Moreno, N. Myller, E. Sutinen, "Smart Program Visualization Technologies: Planning a Next Step", *Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies (ICALT 2005)*, pp. 717-721, (2005).

[2] R. Bednarik, N. Myller, E. Sutinen, M. Tukiainen, "Effects of Experience on Gaze Behaviour during Program Animation", *Proceedings of the 17th Annual Psychology of Programming*

IEEE COMPUTER SOCIETY

*Interest Group Workshop (PPIG'05)*, Brighton, UK pp. 49-61, (2005).

[3] M. Beer and J. Whatley, "A Multi-agent Architecture to Support Synchronous Collaborative Learning in an International Environment", *Proceedings of the 1st International Conference on Autonomous Agents*, pp. 505–506, (2002).

[4] M. Ben-Ari, "Program Visualization in Theory and Practice", *Informatik. Informatique*, 2, pp. 8–11, (2001).

[5] R. Ben-Bassat Levy, M. Ben-Ari, P. A. Uronen, "The Jeliot 2000 Program Animation System", *Computers & Education*, 40 (1), pp. 1–15, (2003).

[6] M. Boicu, G. Tecuci, B. Stanescu, D. Marcu, M. Barbulescu, and C. Boicu, "Design Principles for Learning Agents", *Proceedings of AAAI-2004 Workshop on Intelligent Agent Architectures: Combining the Strengths of Software Engineering and Cognitive Systems*, (2004).

[7] C. M. Boroni, T. J. Eneboe, F. W. Goosey, J. A. Ross, R. J. Ross, "Dancing with Dynalab - Endearing the Science of Computing to Students," in *Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education*, pp. 135–139, (1996).

[8] P. Brusilovsky, and H. Su, "Adaptive Visualization Component of a Distributed Web-Based Adaptive Educational System", S.A. Cerri, G. Gouardères and F. Paraguaçu (Eds.), *Intelligent Tutoring Systems*, Vol. 2363 of LNCS, Springer-Verlag, 2002, pp. 229–238.

[9] K. A. Ericsson, "The acquisition of expert performance as problem solving: Construction and modification of mediating mechanisms through deliberate practice", In Davidson J. E. & Sternberg R. J. (Eds) *Problem solving*, pp. 31–83. New York, Cambridge University Press, (2003).

[10] R. M. Felder and J. E. Spurlin, "Applications, Reliability, and Validity of the Index of Learning Styles", *International Journal of Engineering Education*, 21(1), pp.103–112, (2005).

[11] A. Garro and L. Palopoli, "An XML Multi-agent System for e-Learning and Skill Management", 2002. url:http://www.netobjectdays.org/pdf/02/papers/ malceb/-0623.pdf (accessed Feb. 2003).

[12] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge, "The Belief-Desire-Intention Model of Agency", *Proceedings of the 5th International Workshop on IntelligentAgents V: Agent Theories, Architectures, and Languages*, pp. 1–10, (1999).

[13] P. Gestwicki, B. Jayaraman, "Interactive visualization of Java programs", in *Proceedings of IEEE Symposia on Human Centric Computing Languages and Environments* 2002, pp. 226–235, (2002).

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

[15] N. R. Jennings and M. Wooldridge, "Applications of Intelligent Agents," *Agent Technology Foundations, Applications, and Markets*, Springer-Verlag, 1998.

[16] R. Küstermann, D. Ratz, D. Seese, "Effektive Java-Grundausbildung unter Einsatz eines Learning Management Systems und spezieller Werkzeuge," in proceedings of *INFOS' 05*, pp. 10, (2005). (In German)

[17] M. Lattu, J. Tarhio, V. Meisalo, "How a Visualization Tool Can be Used - Evaluating a Tool in a Research & Development Project", *Proceedings of the 12th Psychology of Programming Interest Group (PPIG) Workshop*, (2000).

[18] M. Luck, P. McBurney, and C. Preist, "Agent Technology: Enabling Next Generation Computing A Roadmap for Agent Based Computing", *AgentLink*, (2003).

[19] R. Miraftabi, "Intelligent Agents in Program Visualizations: A Case Study with Seal", in *Proceedings of the First International Program Visualization Workshop*, Porvoo, Finland, July (2001), pp. 53–58.

[20] A. Moreno, N. Myller, E. Sutinen, M. Ben-Ari, "Visualizing Programs with Jeliot 3", in *Proceedings of Advanced Visual Interfaces, AVI 2004*, pp. 373–376, (2004).

[21] P. Mulholland, "Using a Fine-Grained Comparative Evaluation Technique to Understand and Design Software Visualization Tools", in S. Wiedenbeck and J. Scholtz (Eds.), *Empirical Studies of Programmers: Seventh Workshop*. New York: ACM Press (1997).

[22] T. J. Norman and N. R. Jennings, "Constructing a Virtual Training Laboratory using Intelligent Agents", *International Journal of Continuous Engineering Education and Life-Long Learning*, 12(1–4), pp. 201–213, (2002).

[23] R. Oechsle, T. Schmitt, "JAVAVIS: Automatic Program Visualization with Object and Sequence Diagrams Using the Java Debug Interface (JDI)," in S. Diehl (Ed.), *Software Visualization*, Vol. 2269 of LNCS, pp. 176–190, (2002).

[24] M. A. Razek, C. Frasson, and M. Kaltenbach, "Toward More Cooperative Intelligent Distance Learning Environments", *Software Agents Cooperation Human Activity*, 2002. url: http://www-perso.iro.umontreal.ca/~abdelram (accessed Feb. 2003).

[25] J. Sajaniemi, M. Kuittinen, "Program animation based on the roles of variables", *Proceedings of the 2003 ACM symposium on Software visualization*, pp. 7–16, (2003).

[26] Y. Shang, H. Shi, and S. Chen, "An Intelligent Distributed Environment for Active Learning", *Proceedings of the 10th International Conference on World Wide Web*, pp. 308–315, (2001).

[27] S. Sun, M. Joy, and N. Griffiths, "An Agent-Based Approach to Dynamic Adaptive Learning", *Proceedings of Agent Based Systems for Human Learning (ABSHL) Workshop, the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pp. 48-58, (2005).

[28] S. Sun, M. Joy, and N. Griffiths, "The Use of Learning Objects and Learning Styles in a Multi-Agent Education System", *Proceedings of ED-MEDIA 2005 - World Conference on Educational Multimedia, Hypermedia & Telecommunications*, pp. 3403-3410, (2005).

[29] E. Sutinen, J. Tarhio, T. Teräsvirta, "Easy Algorithm Animation on the Web", *Multimedia Tools and Applications*, 19 (2), 2003, pp. 179–184.

[30] A. Zeller and D. Lütkehaus, "DDD, A Free Graphical Front-End for UNIX Debuggers", *ACM SIGPLAN Notices*, 31(1), pp. 22–27, (1996).

IEEE COMPUTER SOCIETY