AN AGENT-BASED CONCEPTUAL MODEL FOR COMPUTATIONAL PROBLEM SOLVING

Mohd Aris, T.N.
Faculty of Computer Science and Information Technology
University Putra Malaysia,
43400 UPM Serdang,
Selangor Darul Ehsan, Malaysia
nuranis@fsktm.upm.edu.my

ABSTRACT

Novice students face difficulties to understand problems and transform the problems using problem solving techniques. A novel model based on agents has been designed to assist novices in computational problem solving. This paper focus from the very first step starting from a problem given in text form and producing the Problem Analysis Chart (PAC), Hierarchy Input Process Output (HIPO) chart, Input Process Output (IPO) chart, flowchart and algorithm. This model consists of two modules namely the Problem Solving Comprehension Module and the Assignment Module. The main role of agents in the Problem Solving Comprehension Module is to extract information, transform to problem solving documents and generate module number. This module consists of six agents – GUI, PAC, HIPO, IPO, flowchart and algorithm agents. The main role of agents in the Assignment Module is interaction with the user. This module consists of four agents – GUI, Novice, Assessment and Guidance agents. The model is tested with three different problems and proved that the proposed extraction algorithm worked accurately.

KEYWORDS

problem solving, novice, agents, extraction, visualization

1 INTRODUCTION

Problem solving techniques is one of the earliest topics taught to novice students in the computer programming course. In University Putra Malaysia, extra time is allocated to teach the problem solving topic. Understanding a problem clearly is very important for novices before writing the actual program source codes. However, based on experience, most of these students find it very difficult to understand the problem given and transform them using problem solving techniques. Problems that are misinterpreted by novices may cause an incorrect result and misconceptions of the programming concepts. This may also lead to programming errors and bugs. Previous researches have focused on various techniques to enhance the comprehension of novices in problem solving [1], [2], [3].

Software agents have the potential to facilitate in the learning and teaching process. Agents have been widely applied in various applications such as interactive tutoring [4], medical diagnosis system [5] and image analysis [6]. Featuring four properties namely autonomy, pro-activeness, reactivity and social ability, agents have assisted humans in searching, providing information and decision making.

Due to the difficulties of novices to map a problem given to problem solving techniques, an agent based model is proposed to help them. Instructors are also expected to gain advantage from the proposed model to assist them in teaching. We introduced a novel intelligent agent paradigm based on the Belief, Desire and Intention (BDI) architecture to perform a mapping of a computational problem to produce problem solving documents. These problem solving documents will be elaborated in further sections.

This paper consists of six sections. The second section explains about the previous works that are related to our research. Then, the proposed agent conceptual model is described in section three. Next, section four, present the experimental results which test the proposed model with three different problems. This is followed by section five the gives a discussion of the proposed model. Lastly, section five concludes the paper and provides briefs suggestions for future works.

2 RELATED RESEARCH

Conventional computational problem solving techniques consists of flow chart, algorithm, structure chart, Nassi-Shneiderman chart and Warnier-Orr diagram. Researchers have designed visualization tools that integrate problem solving techniques to help novices in program comprehension. Several of them are VIP for C++ [7], Ville [8] and Planani [9] for multiple different languages. These tools are based on source code level that assumed students already have programming knowledge. Another tool which is close to the source code level is RAPTOR [10] that enables students to construct flow charts and the tool will visualize them by showing the content of variables and arrays.

A system that adopts kick-start activation was introduced in the beginning of an introductory programming course to provide the deep structure of programming before the surface structure by making the students solve different kinds of problem [11]. Simulation game is another visualization tool that assists students in computational problem solving [12].

Authors proposed Zoom Visual Flow (ZViF) Technique that represents source code in graphical views based on different shapes and colors of Action Icons Control Constructs (AICOS) notations [13].

A 3De synergetic program visualization tool was designed to help novice students to view programming stages starting from designing problem solving, developing code and validating logical flow of the program through visualizing multi level of program abstraction [14]. The mentioned visualization tools do not utilize multi agent interactions in their design. Multi agent interactions are software systems that contain more than one agent which communicate with each other. Software agents are reactive systems and able to work in inaccessible, non-deterministic, dynamic and continuous environments compared to functional systems (e.g. a compiler). Therefore, they are more flexible than functional systems where they can make decisions autonomously in the mentioned environments.

Several works support visualization with agents to visualize pseudo-code language [15], visualize expression in C [16] and visualize object parameters [17]. Social agents are used to support the understanding of program visualization through discussion with user and explanation from the agent [15]. [16] adapt the level of details in the visualizations based on the learner's knowledge and focus on topic that are not understood by the user. An approach to change visualization parameters for example shape, colors and style is used to help novices in programming [17]. Jeliot 3 [18] integrates multi agents namely Student Agent, Record Agent, Modelling Agent, Learning Object Agent and Evaluation Agent to provide dynamic and adaptive learning materials to individual users.

Designing multi-agent system requires knowledge on the role of agents. Besides autonomous features, agents are designed to interact with human or systems, synthesize information, monitor, negotiate and coordinate. Agents and Artificial Intelligence (AI) are associated to each other. Several research have integrated agents with fuzzy logic for information extraction [19], controller design [20], healthcare [21], manipulator systems [22], decision making [23].

The mentioned tools differ from the proposed model in terms of the architectural design. To date, no other visualization problem solving tools have modeled agents as what has been proposed. The uniqueness of the design will be explained further in the next section. The feature of agents as problem solvers and capability to act independently is appropriate to aid novices in computational problem solving.

3 PROPOSED CONCEPTUAL MODEL

The proposed tool focuses on novice students that do not have any background in programming. The scenario is as follows. It starts from a simple problem or exercise given to students in text form. The students are required to understand the problem and transform the problem to problem solving documents consisting of a PAC, HIPO Chart, IPO Chart, Flowchart and Algorithm. These problem solving documents are related to one another. The PAC is based on the problem, the HIPO is based on the PAC, the IPO is based PAC and HIPO, the flowchart and algorithm is based on the IPO. The data that are related will be explained in detail shortly. Therefore, a clear understanding on the problem given is vital. This is to avoid or reduce any programming errors or bugs during program development.

Our proposed model consists of two modules namely Problem Solving Comprehension Module and Assignment Module as shown in Figure 1. The agents are modeled using Prometheus notation [24], [25].

The Problem Solving Comprehension Module provides a database containing various example problems in computer programming based on the most basic course syllabus: fundamental programming, selections, loops, methods, arrays, objects and classes and strings and text input/output. The role of agent in the Problem Solving Comprehension Module is to extract current information starting from the problem given in text form, transform the information to problem solving documents and generate module number. Six agents are needed to perform these tasks namely, GUI Agent, PAC Agent, HIPO Agent, IPO Agent, Flowchart Agent and Algorithm Agent. The extraction algorithm proposed in [26] and fuzzy logic will be integrated in these agent actions. The GUI Agent is the interface that will be used by the user to interact with other agents. There is a communication from agents marked as * to the GUI Agent in Figure 1. The arrows are not shown in Figure 1 due to limited spaces. These agents communicate with each other by sending data/messages - keywords, input/Process/Output (I/P/O), module number and process.

The Assignment Module provides a database containing exercise to be solved by novices. The exercise covers all the topics mentioned in the Problem Solving Comprehension Module. The roles of agents in the Assignment Module are mainly for interaction with the user starting from obtaining the students background, guiding the students and evaluate the students' level. The agents designed for this module consists of GUI Agent, Guidance Agent, Novice Agent and Assessment Agent. The function of the GUI Agent in this module is the same as the function of GUI Agent in the Problem Solving Comprehension Module. These agents also communicate with each other by sending data/messages – student information, student current status/background, student level/performance and guide students in certain topic.

The agents in the proposed system are practical reasoning agents based on the standard BDI architecture. The Jason AgentSpeak interpreter is used to implement the agents [27].

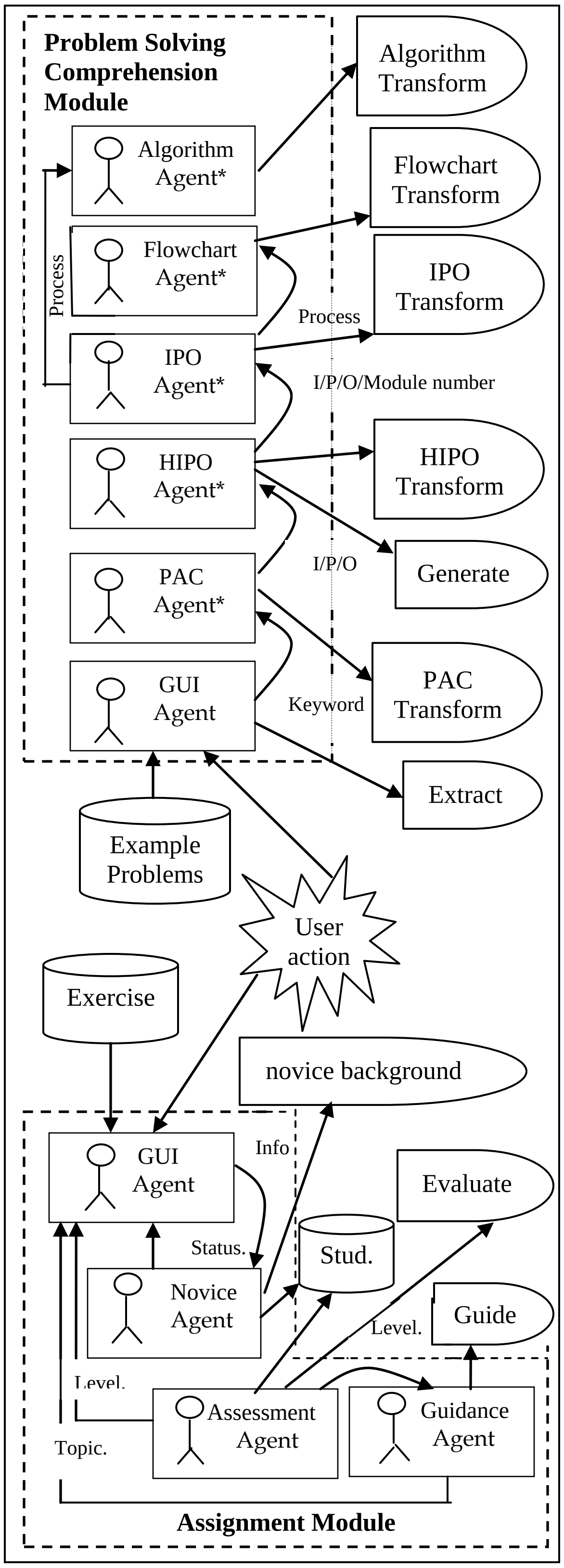


Figure 1. Proposed Computational Problem Solving Conceptual Model

Currently, we are in the initial stage of our research. We started designing and coding the Problem Solving Comprehension Module consisting of the GUI Agent and the PAC Agent. Figure 2 shows the class diagrams of these agents. The ProblemSolvingAgentGUI extends the AgArch class and communicates with the guiAgent and pacAgent. The ProblemSolvingAgentGUI constructor displays the interface containing a button that is used by the novice to start the problem understanding process. The act method executes an action and the stopAg is called by the infrastructure tier when the agent is about to be killed.

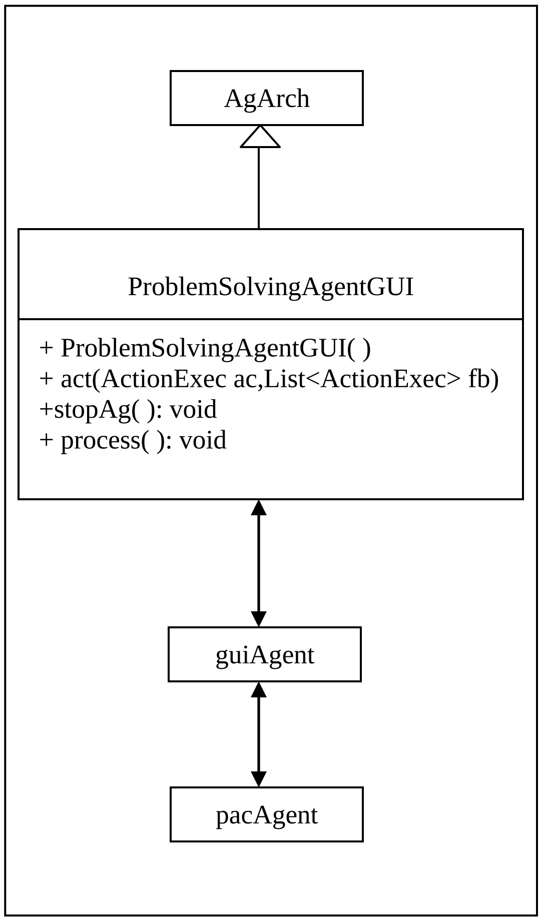


Figure 2. Agent Class Diagram

The process method in Figure 2 is explained further in Figure 3 which is the proposed extraction algorithm.

```
process( ) algorithm
1. read problem from example problems
   database
2. read the problem token by token and store
   in an array, a
3. initialize j to 1
4. loop (j < array length a)
  4.1 if (a[j] = = ”keyA”)
     4.1.1 extract and display wordA → dataA
  4.2 if (a[j] = = “keyB”)
     4.2.1 extract and display wordB → dataB
  4.3 if (a[j] = = “keyC”)
     4.3.1 extract and display wordC → dataC
  4.4 if (a[j] = = ”keyD”)
     4.4.1 extract and display wordD → procD
  4.5 if (a[j] = = “keyE”)
     4.5.1 extract and display wordE → procE
  4.6 if (a[j] = = “keyF”)
     4.6.1 extract and display wordF → procF
  4.7 if (a[j] = = ”keyG”)
     4.7.1 extract and display wordG → outG
  4.8 if (a[j] = = “keyH”)
     4.8.1 extract and display word2 → outH
  4.9 if (a[j] = = “keyI”)
     4.9.1 extract and display wordI → out
      j++
   end loop
5. print the PAC
```

Figure 3. Extraction Algorithm

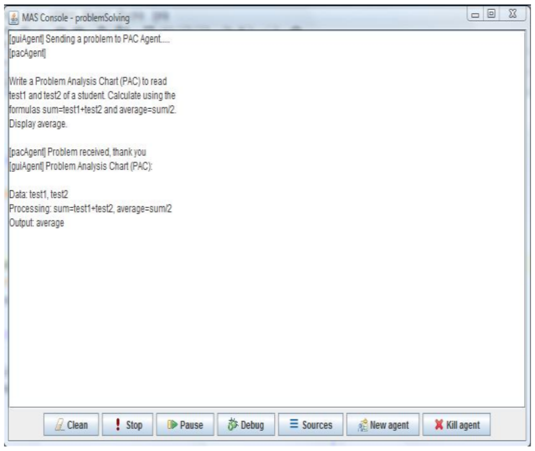
4 EXPERIMENTAL RESULTS

The proposed model is tested with three different problem statements. These problems are written in text form. The problems are given as follows and the results are illustrated in Figures 4, 5 and 6 respectively. Figure 7 shows the user interface controlled by the novice to understand the problems that are stored in the Example Problems database.

Problem 1: Write a Problem Analysis Chart (PAC) to read test1 and test2 of a student. Calculate using the formulas sum=test1+test2 and average=sum/2. Display average.

Problem 2: Write a PAC to get length and width of a rectangle. Calculate the area of the rectangle using the equation area=lengthXwidth. Print area.

Problem 3: Write a PAC to input totalbill and payment by a customer in a supermarket. Calculate changedue=payment-totalbill of the customer. Output totalbill and changedue.



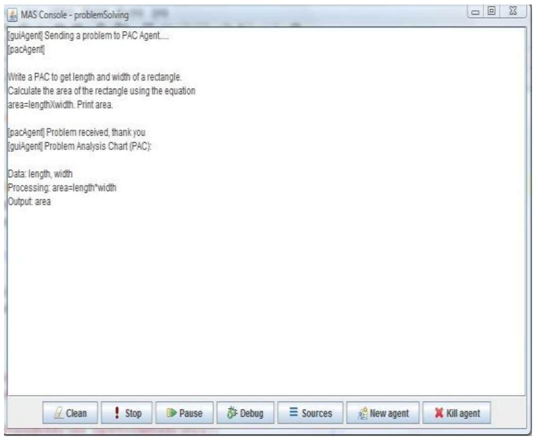
```
[guiAgent] Sending a problem to PAC Agent.....
[pacAgent]

Write a Problem Analysis Chart (PAC) to read
test1 and test2 of a student. Calculate using the
formulas sum=test1+test2 and average=sum/2.
Display average.

[pacAgent] Problem received, thank you
[guiAgent] Problem Analysis Chart (PAC):

Data: test1, test2
Processing: sum=test1+test2, average=sum/2
Output: average
```

Figure 4. PAC Problem 1



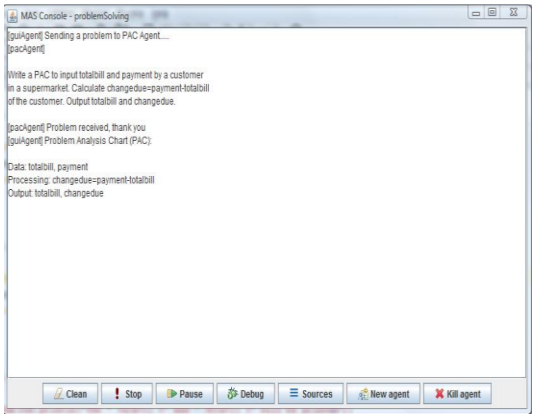
```
[guiAgent] Sending a problem to PAC Agent.....
[pacAgent]

Write a PAC to get length and width of a rectangle.
Calculate the area of the rectangle using the equation
area=lengthXwidth. Print area.

[pacAgent] Problem received, thank you
[guiAgent] Problem Analysis Chart (PAC):

Data: length, width
Processing: area=length*width
Output: area
```

Figure 5. PAC Problem 2



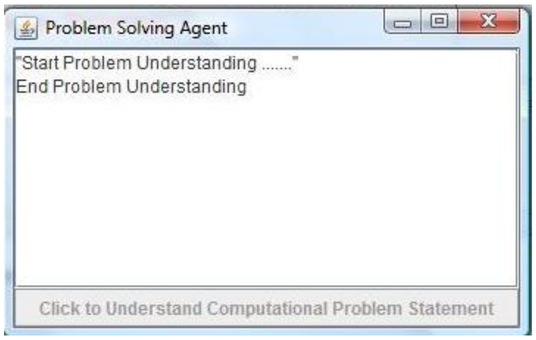
```
[guiAgent] Sending a problem to PAC Agent.....
[pacAgent]

Write a PAC to input totalbill and payment by a customer
in a supermarket. Calculate changedue=payment-totalbill
of the customer. Output totalbill and changedue.

[pacAgent] Problem received, thank you
[guiAgent] Problem Analysis Chart (PAC):

Data: totalbill, payment
Processing: changedue=payment-totalbill
Output: totalbill, changedue
```

Figure 6. PAC Problem 3



```
"Start Problem Understanding ......."
End Problem Understanding
```

Figure 7. User Interface

5 DISCUSSION

Our proposed model has been tested using the three different problems mentioned and succeeded to produce the accurate results. This is shown by the results in Figures 4, 5 and 6 indicating Data, Processing and Output where the keywords in the problems given in text form are extracted correctly. The results also illustrate communication between guiAgent and pacAgent that demonstrate knowledge representation by these agents which shows visualization features.
In our research work, we are using the existing conventional problem solving techniques. Our contribution is mainly on the novel model that is different from other visualization problem solving model. In addition, the extract, transform

and generate algorithms in the Problem Solving Comprehension Module; and capture novice background, evaluate and guide in the Assignment Module are also the contributions in this research work. Furthermore, we intend to incorporate fuzzy logic in our model. We are currently in the process of extending the design and coding of the algorithms mentioned using Jason AgentSpeak [27], an agent programming language which is built on top of the Java programming language.

This model represents agents that exhibit the reactive, proactive and social ability features. These features can aid novices that face difficulties in computational problem solving, as problem solving is very important not only in the computer science field but also other fields such as medical, engineering and business. Moreover, this model can enhance the students' computational problem solving skills.

6 CONCLUSION AND FUTURE WORKS

A model based on agents has been created to assist students in computational problem solving. The Problem Solving Comprehension Module function is for extraction, transformation and module number generation. The Assignment Module is for interaction with the user. The agent in the Problem Solving Comprehension Module consists of GUI, PAC, HIPO, IPO, flowchart and algorithm agents. The agent in the Assignment Module consists of the GUI, Novice, Assessment and Guidance agents. The initial model is evaluated using three different problem statements and produced accurate results. In addition, the proposed model illustrates visualization, communication and knowledge representation features by software agents.

For future works, the design and coding of the whole model that has been built will be implemented. In addition, the proposed extraction algorithm will be extended to provide more keywords to cover a wide problem statement which is associated to the data stored in the Example Problems database. Moreover, the proposed problem solving visualization tool will be integrated with a program development tool based on agents.

7 REFERENCES

1. Guo, P., Qi, H.:(2010). Study on Interactive System for Teaching Programming (ISTP) based on Intelligent Agent. In Proc. 2010 6th International Conference on Wireless Communications WiCOM 2010, Networking and Mobile Computing, pp. 23--25, , Article number 5600855, ISBN 978-142443709-2, Chengdu (2010).
2. Boyer, K.E., Lahti, W., Phillips, R., Wallis, M.D., Vouk, M.A., Lester, J.C.: Principles of Asking Efective Questions during Student Problem Solving. SIGCSE'10 - Proceedings of the 41st ACM Technical Symposium on Computer Science Education, pp. 460--464, ISBN 978-160558885-8, Milwaukee, WI (2010).
3. Pedrycz, W., Rai, P.: A Multifaceted Perspective at Data Analysis: A Study in Collaborative Intelligent Agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 38, no. 4, pp. 1062--1072, ISSN 10834419 (2008).
4. Yaskawa, S., Sakata, A.: The Application of Intelligent Agent Technology to Simulation. Mathematical and Computer Modeling, vol. 37, no. 9-10, pp. 1083--1092, ISSN 0895-7177 (2003).
5. Iantovics, B. L.: Agent-based Medical Diagnosis Systems. Computing and Informatics, vol. 27, no. 2, pp. 593--625, ISSN 1335-9150 (2008).

6. Bell, D. A., Beck, A., Miller, P., Wu, Q. X., Herrera, A.: Video Mining –Learning Patterns of Behaviour via an Intelligent Image Analysis System. In Proceedings of the 7th International Conference on Intelligent Systems Design and Applications, pp. 460--464, ISBN 978-0-7695-2976-9 (2007).
7. Virtanen, A.T., Lahtinen, E., Järvinen H.-M.: VIP, A Visual Interpreter for Learning Introductory Programming with C++. In Proceedings of the Fifth Finnish/Baltic Sea Conference on Computer Science Education, pp. 129--134 (2005).
8. Rajala, T., Laakso, M.-J., Kaila, E., Salakoski, T.: VILLE – A Language-Independent Program Visualization Tool. In Proceedings of the Seventh Koli Calling Conference on Computer Science Education, (2007).
9. Sajaniemi, J., Kuittinen, M.: Visualizing Roles of Variables in Program Animation. Information Visualization 3, pp. 137--153 (2004).
10. Giordano, J.C., Carlisle, M.: Toward a More Effective Visualization Tool to Teach Novice Programmers. In SIGITE ’06: Proceedings of the 7th Conference on Information Technology Education, pp. 115--122.
11. Lahtinen, E., Ahoniemi, T.: Kick-Start Activation to Novice Programming – A Visualization-Based Approach. Electronic Notes in Theoretical Computer Science, vol. 224, pp. 125—132 (2009).
12. Liu, C-C., Cheng, Y-B., Huang, C-W.: The Effect of Simulation Games on the Learning of Computation Problem Solving. Computers & Education, vol. 57, pp. 1907--1918 (2011).
13. Kadar, R., Sulaiman, S.: The Effectiveness of Zoom Visual Flow (ZViF) Technique in Program Comprehension Activities. International Symposium in Information Technology (ITSim), pp. 1--6 (2010).
14. Affandy, Suryana, N., Salam, S., Azmi, M.S., Noersasongko, E.: 3De – Synergetic Program Visualization: A Visual Learning Tool for Novice Students. International Conference on E-Education, Entertainment and E-Management (ICEEE), pp. 133—137 (2011).
15. Miraftabi, R.: Intelligent Agents in Program Visualizations: A Case Study with Seal. In Proceedings of the First International Program Visualization Workshop, pp.53-58, Porvoo, Finland (2001).
16. Brusilovsky, P.: Adaptive Visualization Component of a Distributed Web-Based Adaptive Educational System, Cerri, S.A., Gourdères, G., Paraguaocu (Eds.), Intelligent Tutoring System, LNCS, vol. 2363, pp. 229--238, Springer-Verlag (2002).
17. Lattu, M., Tarhio, J., Meisalo, V.: How a Visualization Tool Can be Used – Evaluating a Tool in A Research & Development Project. In Proceedings of the 12th Psychology of Programming Interest Group (PPIG) Workshop (2000).
18. Ben-Ari, M., Bednarik, R., Levy, R.B-B. Ebel, G., Moreno, A., Myller, N., Sutinen, E.: A Decade of Research and Development on Program Animation: The Jeliot Experience. Journal of Visual Languages and Computing, vol. 22, pp. 375--384 (2011).
19. Ropero, J., Gómez, A., Carrasco, A., León, C.: A Fuzzy Logic Intelligent Agent for Information Extraction: Introducing a New Fuzzy Logic-based Term Weighting Scheme. Expert Systems with Applications, vol. 39, pp. 4567--4581 (2012).
20. Olajubu, E.A., Ajayi, O.A., Aderounmu, G.A.: A Fuzzy Logic Based Multi-Agents Controller. Expert Systems with Applications, vol. 38, pp. 4860--4865 (2011).
21. Fenza, G. Furno, D., Loia, V.: Hybrid Approach for Context-aware Service Discovery in Healthcare Domain. Journal of Computer and System Sciences, vol. 78, pp. 1232--1247 (2012).
22. Bohner, P.: A Multi-Agent Approach with Distributed Fuzzy Logic Control. Computers in Industry, vol. 26, pp.219--227 Elsevier Science B.V. (1995).
23. Li, S., Zheng Li, J.: AgentsInternational: Integration of multiple agents, simulation, knowledge bases and fuzzy logic for international marketing decision making. Expert Systems with Applications, vol. 37, pp. 2580--2587 (2010).

24. Padgham, L., Winikoff, M.: Developing Intelligent Agent Systems: A Practical Guide, ISBN 0-470-86120-7, Wiley, Chichester (2004).
25. Woorldridge, M.: An Introduction to MultiAgent Systems. Wiley United Kingdom (2009).
26. Mohd Aris, T.N.: Object-Oriented Programming Semantics Representation Utilizing Agents. Journal of Theoretical and Applied Information Technology, vol. 31, pp. 10--20 (2011).
27. Jason A Java-based Interpreter for an Extended Version of AgentSpeak http://jason.sourceforge.net/Jason/Jason.html