Dublin Institute of Technology
ARROW@DIT

Conference Papers | Computer Science Education

2007-12-01

Evaluation of Robocode as a Teaching Tool for Computer Programming

Arnold Hensman
Institute of Technology, Blanchardstown, arnold.hensman@itb.ie

Follow this and additional works at: http://arrow.dit.ie/itbinfocsecon

 Part of the Computer Sciences Commons, Engineering Education Commons, and the Robotics Commons

Recommended Citation

Hensman, A.: Evaluation of Robocode as a Teaching Tool for Computer Programming. Proceedings of the National Digital Learning Repository (NDLR) Symposium, 2007, Trinity College Dublin, Ireland. December 14, 2007.

This Conference Paper is brought to you for free and open access by the Computer Science Education at ARROW@DIT. It has been accepted for inclusion in Conference Papers by an authorized administrator of ARROW@DIT. For more information, please contact yvonne.desmond@dit.ie, arrow.admin@dit.ie.

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 License

Evaluation of Robocode as a Teaching Tool for Computer Programming

Arnold Hensman
Department of Informatics, School of Informatics and Engineering
Institute of Technology Blanchardstown, Dublin 15, Ireland
Email: Arnold.Hensman@itb.ie

***Abstract*—Robocode began as an educational tool to aid in learning Java programming. It has since evolved into something of a phenomenon, as the prospect of creating simple to complex virtual tanks appears to pose an attractive challenge to both novice and expert programmers alike. What started out as a teaching tool has grown into a worldwide network of competitors, all keen to prove that their 'bot' stands out from the crowd. Competitions are well organised and many Robocode events are a PR dream for the computing companies that sponsor them. Without a doubt, this easy to use application has sparked the imagination of the world of programming. This is especially evident in the number of higher education institutes that regularly hold competitions for their computing and engineering students, often inviting participants from other colleges. In Ireland alone, a major national event for third level students is held annually at the Tipperary Institute. Sponsors have included the likes of Microsoft and Lenovo and students from most Irish universities and colleges have taken part. This is merely a scenario that has been mimicked across the globe.**

A cursory browse through a typical computing faculty website will likely reveal a reference to Robocode. This paper attempts to look back to the roots of Robocode, and evaluate its merits as a teaching tool whether for use inside or outside the classroom. The detailed results of a survey are presented, showing the responses of students who have used the tool in a number of capacities, more specifically, an evaluation by those who have participated in the national competition or merely used the tool as part of their programming course work. Lecturers have also been asked for an evaluation to gauge its effect on programming students. With so many willing to dedicate extra curricular time to participate, it is worth investigating what ignited this spark in the first place. What motivates a student or indeed any programmer to want to develop a robot tank that fires bullets, and, attempts to dodge the bullets of other tanks?

***Index Terms*—Robocde, Teaching Tool, Object-Oriented Programming, Tank, Bot.**

I. What is Robocode?

THE game is designed to help people learn how to program in Java in an enjoyable way. Programmers use software to control a miniature battle tank on screen. Once designed and coded a robot can be uploaded to take part in a military style battle. Although a basic battle tank can be created in minutes, the most sophisticated models may take months of refinement. Tanks move around an arena and essentially have two functions: to fire bullets at other tanks as precisely as possible and avoid being hit by bullets as much as possible. This involves clever use of scanning, driving closely around the walls of the arena and general manipulation of the physics of battle. There is a large set of specialized robot tanks included in the application which users can pit their own robot up against for practice. Some are good at tracking walls, others at spinning away quickly or aggressively firing at targeted objects. Battles can be one on one, may involve several tanks, or you might even be up against a melee of non-competing tanks just to stir things up.

After a battle, statistical results can easily be viewed such as ranking order, number of hits, energy used etc. Programmers are thus encouraged to bring their own strategies to the game, all of which seems to have captivated the imagination. Fig. 1 below shows a typical multi tank battle scenario in progress.

Robocode was created by Mathew Nelson as an endeavor to use Java to build the game "he always wanted to play". While employed by IBM, he uploaded Robocode to their emerging technologies portal, *alphaWorks* [1], a web community for early adapters to preview prototype technologies. In 2001,

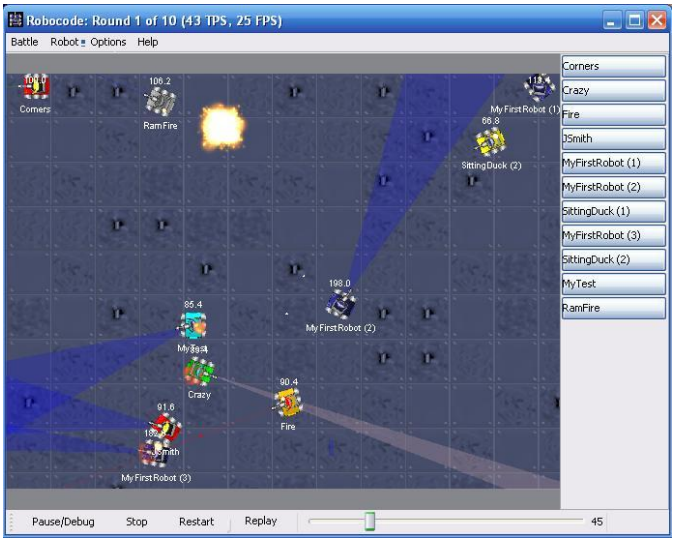

Fig. 1. A Robocode tank battle in progress

after only one year on alphaWorks, Robocode had been downloaded over 120,000 times. Nelson's intention was to create a tool for competitive programmers which would, as he puts it himself, be "Like chess, simple to learn, difficult to master" [2]. Six years on, this Open Source educational game has been highly refined by both its creator and a series of contributors. At this point, some of the more successful bots – computer controlled entities simulating a multiplayer knock-out – have gained reputations in their own right[1]. This more or less sums up the attraction for students, who can almost immediately get a robot up and running with only a basic knowledge of Java. The experimentation and testing phases of robot creation add a fun element to programming.

In Ireland, smaller competitions are held locally in many colleges and universities and a national event is held once a year at the Tipperary Institute, Thurles, where participants are invited from around the country[2]. NUI Maynooth, for example, has a competition outlined on its computing department website to be used as a stepping stone towards the national finals. The problem based learning (PBL) research group at NUI Maynooth has also proposed using Robocode to teach computer programming in a PBL setting. O'Kelly and Gibson [3] assert that a combination of using Robocode both in the classroom and in competitions meets Duch's five requirements [4] for a good *problem* appropriate to PBL. Computer programming is generally considered a subject area where good problems are difficult to come across.

The national competition held annually at the Tipperary Institute is described on its website as an opportunity for talented first year computing students from around the country to showcase their work [5]. This has drawn the attention of local industries, with many spectators on the lookout for such talent. These events have a proven track record for finding the student with a natural flair for programming and giving confidence to the high achiever. The original intention of Robocode however, was not as a medium for programming competitions, but as a learning tool for computer programming. This paper attempts to evaluate its merits in this regard and determine if it could in fact be successfully used in the classroom to help students, both weak and strong to improve their coding skills.

II. Robocode and Object Oriented Programming

At first glance, Robocode may appear to require intricate knowledge of object-oriented programming, a concept many first-years would barely have touched upon. Every basic tank for example is an extension of a parent class. These elements however, can easily be avoided at the initial stages of robot design, leaving the student to concentrate solely on the details of their bot.

It should be noted at this point that there is two schools of thought regarding the best way to teach computer programming. At one end of the spectrum, educators prefer to introduce standard procedural programming first, grounding the student in these ideas, and eventually introducing objects at a later stage. At the opposite end, they strongly recommend introducing object-oriented concepts from the beginning – the so-called *Objects First* method. For the former, Robocode enables novices to get involved at an early stage in their course, even if they must just accept certain elements of the package without fully appreciating them at the time. For the latter, Robocode is an ideal tool, a way to present the complex theories about objects in an imaginative way.

III. The Survey

This paper attempts to evaluate Robocode as a learning aid by posing a survey to students and lecturers who have used it. Many of the lecturers have been involved in organizing local and national Robocode competitions The survey contains questions about the best ways Robocode could be used in a higher-education environment. The aim is also to determine if it is suitable for beginners or if it is merely a platform for the more experienced student to showcase their work. Students have been asked to rate how useful it is in learning programming given the hindsight of spending one or more years in college. All students surveyed are currently in their second year of a computing degree. They are also asked about their own learning preferences and to objectively rate their interest and abilities in computer programming. Since Robocode is mainly targeted at the kinesthetic learner – the 'doers' – some interesting results are evident. Lecturers are asked to rate the package with a view to what stage in a course it could best be used, more specifically, should it be used before or after introducing objects. Lecturers have also been asked how they feel about using it in the classroom and if they have, what kind of results it yielded.

In total, the responses of 25 lecturers and 65 students from various universities and institutes of technologies around the country are presented. The students have been divided into two categories, those who have participated in the national competition or otherwise and those who have not. Students have also been given a slightly different set of questions regarding their profile and how much they feel they learned from the tool. While lecturers have been asked to select a variety of reasons why Robocode has become so popular, students are forced to select the one thing they liked most about it in order to gauge their primary motivation.

IV. Lecturer Evaluation

From the lecturers surveyed, 83.3% agreed that students can still use the tool quite easily without any real knowledge of objects. It is revealing, that only 21.4% of lecturers thought students should spend a considerable time - one year or more - receiving a solid grounding in procedural programming methods before looking into objects. The remainder either thought it best to introduce object-orientated programming very early, after the first semester of first year – 42.9%, or even from the very beginning – 35.7%.

[1] A league table provided by the RoboRumble@Home competition is the main active ranking structure for current bots

[2] The National Robcode Competition is an annual event inaugurated by Philip Burke, The Tipperary Institute, Thurles, Ireland

Several 'Objects first' lecturers even commented that teaching that teaching procedural methods first and then switching to objects can actually confuse weaker students. A few participants commented that they have used Robocode to introduce objects in the second semester of first year. This is interesting to note, since many university and institute of technology syllabi appear to require at almost one full academic year of procedural programming before seriously tackling objects. In some cases the transition is made from C programming in first year - inherently procedural - and progressing to C++ in subsequent years.

Authors of computing text books have been chanting the 'objects first' mantra for years. While in theory it may seem a logical approach, the heavy syntax imposed by Java can make this difficult for new students. Even in their first Java program the, *public static void main* etc, requires some level of mental abstraction.

Barns and Kolling's introductory programming book uses the popular BlueJ application to argue that early objects need not be difficult for novice programmers [6], [7]. BlueJ is an IDE that provides an easy means of representing concepts such as inheritance, aggregation and polymorphism in a visual way. BlueJ can even generate initialization code from UML type diagrams drawn up by users. Once students understand the ideas graphically, the basic coding becomes very accessible. Compared to other UML modeling tools such as IBM's Rational Rose, BlueJ provides a low-budget alternative that works well in an educational setting.

It has been argued that with so much Robocode web content available, the proliferation of Java code would encourage plagiarism. Phelps, Egert, and Bierre[8] have developed a similar gaming system called MUPPETS - multi-user programming pedagogy for enhancing traditional study. It is still in its initial release phase but claims to support both Java and C#. However, when objects are tackled seriously by lecturers, tools like BlueJ and Robocode are highly suited to demonstrating them.

Almost all were lecturers surveyed were optimistic about its merits as an aid to learning as well as its use for competitions. Fig. 2 illustrates the responses of lecturers when asked to evaluate Robocode as a teaching tool.

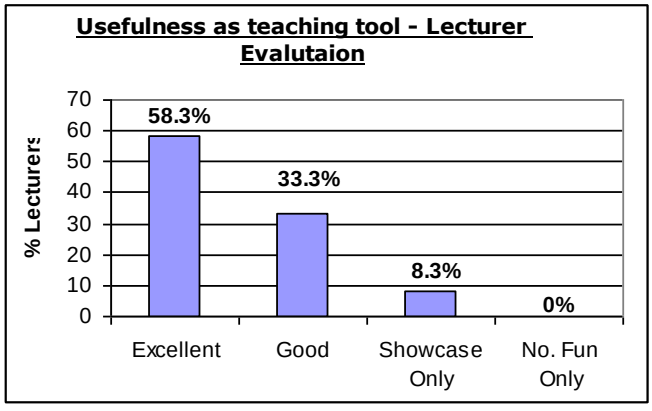

Fig. 2. Lecturer evaluation of Robocode's usefulness as a teaching tool

Regarding using it use in the classroom, 66.7% said they have or would use it in tutorials, while 25% said that they would be willing to provide a demonstration to students and then encourage them to use it as a compliment to their studies. Comments included that Robocode is good for teaching difficult mathematical concepts that student normally struggle with. It was also noted that it is an apt demonstration tool for further development of agents as the implementation of agents is well explained in the accompanying documentation. In fact, within the field of AI, the development of genetically evolved rather than manually coded tanks has recently been tested to great effect. Shichel, Ziserman and Sipper's [9] genetically programmed bot came third place of twenty-seven in a competition where it was the only one not written by a human.

However, for lecturers, it was mainly the social aspects that seemed to be its selling point rather than its use per se, as a learning tool. Learning tools for programming are nothing new, yet most have failed to inspire the imagination in the same way. A simple answer as to what makes Robocode different would be that its multiplayer nature introduces students to a new network of like minded programmers. The good students can release their competitive instincts and the weaker ones can learn from their peers in a fun, sociable way. The application is after all a game, and every gamer loves competition. Fig. 3 illustrates the other that factors lectures felt have contributed to its popularity and the percentage of lecturers that selected that reason.

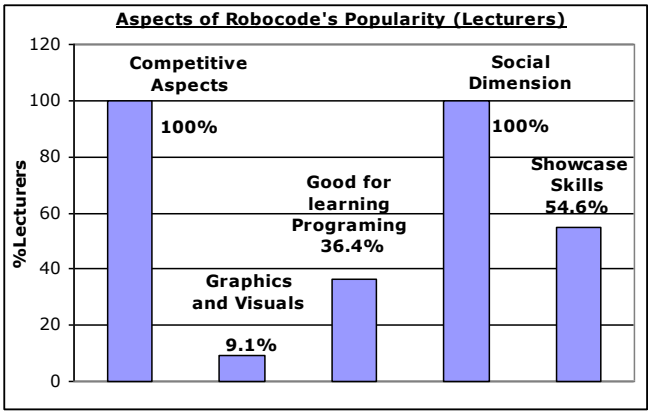

Fig. 3. Reasons chosen by lecturers for Robocode's popularity

It can clearly be seen that the competitive and social dimensions are perceived by lecturers as being the primary motivators for students. Even if the motivation of a student is not necessarily to enhance their programming skills, it would seem that enhanced coding skills are a natural by-product of taking part. Since the national competition requires a team effort from each institution, students are forced to work together and therefore exchange ideas. The social dimension to programming in this way should not be underestimated, and seems a most welcome variation within a pursuit that can often be a solitary one.

Although the original intention outlined by Mathew Nelson was to create a tool catering to any level of programmer, the vast majority of lecturers surveyed, 66.7%, thought that it is mainly suited to novices or first year programmers.

V. Student Evaluation

Practically all students agreed that it was useful as a learning tool and similar findings to the lecturers were determined concerning this question. (Fig. 4 below)

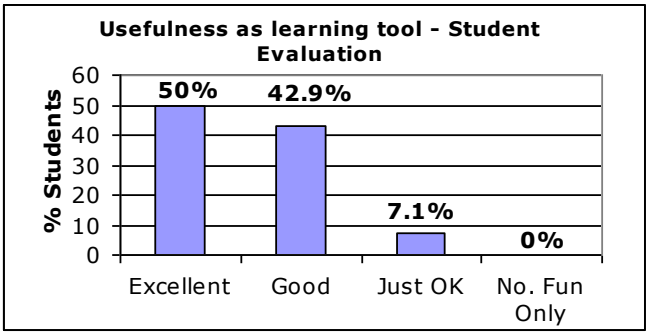

Fig. 4. Student evaluation of Robocode's usefulness as a learning tool

Interestingly, when students were asked about what they enjoyed most, the social dimension did not feature as highly as the lecturers had concluded. However, unlike lecturers, students were forced to choose the one thing that they liked the most and several comments were made by students that the social dimension was a close second. This makes sense since the success of Robocode over other teaching tools seems to be that the learning is made possible due to its interactive context. In fact the difference between those who took part in the national competition and those who simply used it to better their programming was quite telling.

The various aspects of Robocode that students found the most enjoyable is indicated in Fig. 5 below.

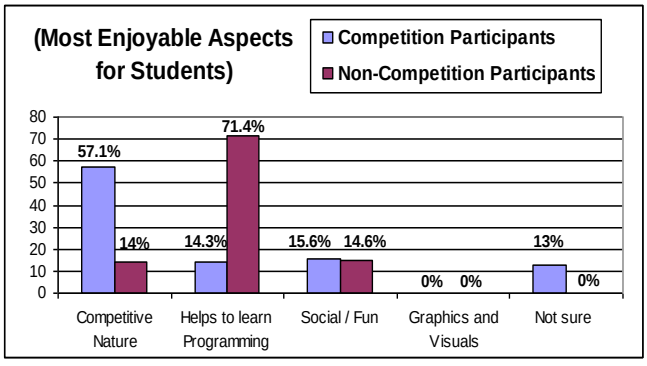

Fig. 5. Student choices of Robocode's most enjoyable aspects

Note the very different pictures emerging from those who took part in competitions and those who did not. The competition participants almost overwhelmingly preferred the competitive aspects over everything else, while those who only used Robocode in tutorials cited the fact that it helps them to learn programming as its most enjoyable feature.

This is an interesting statistic and seems to reinforce the fact that there are two definite advantages to the tool; One within the classroom as an aid to learning and the other within competitions as a platform for students to showcase their work in a socially interactive environment.

Learning Merits of Robocode – Students

It can clearly be seen from Fig. 6 below that students who took part in competitions did not feel as strongly about the learning merits as much as those who did not compete. Note that 0% of the competing students described the tool as '*Excellent'* for learning compared to 12.5% for the non-competing. It also appears that fewer of the non-competing students described Robocode as '*Fun only*'.

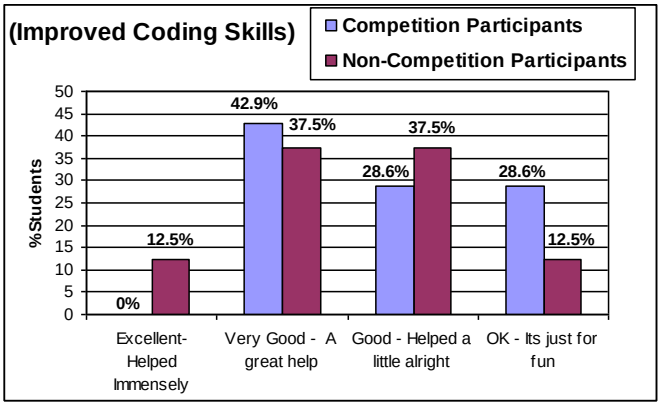

Fig. 6. Student claims regarding improvements in their coding skills as a result of using Robocode

Student Profiles – Self Evaluation of Programming Ability

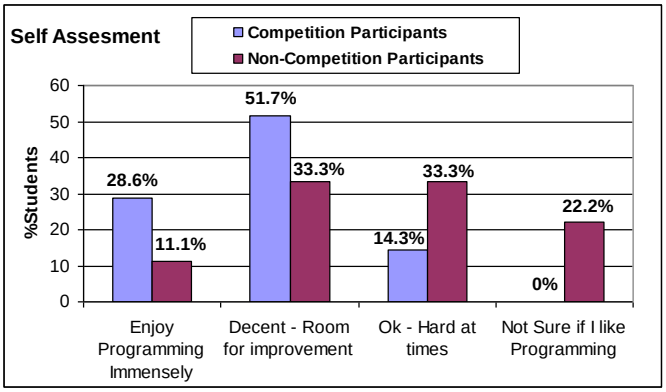

Fig. 7. Student self-assessment of programming abilities

All students were asked to rate themselves objectively about their interest and current ability as a programmer. Within both groups of students there was a direct correlation between how high a student rated themselves as a programmer and whether they preferred the competitive aspects or learning aspects more. Those that rated themselves higher tended to prefer the competitive dimension. Fig. 7 above shows how students evaluated their own abilities as a programmer on a scale from 1 to 4. Students were asked to select option 1 if they enjoyed

programming immensely and had a desire to eventually become a top level computer programmer.

Many of those who thought they were presently weak at programming commented that they were grateful to find something that makes sense of it all. When asked how much they felt they improved by using Robocode, it was actually those who did not participate in competitions at all who reaped the most rewards in terms of increasing their current knowledge. These results clearly indicate that Robocode appears to have a place for both weak and strong students alike.

VI. Learning Preferences

Neither students nor lecturers cited the graphical or visual side as one of the major advantages of using Robocode. This seems somewhat ironic considering it is quite a glossy and colourful computer game.

All students surveyed from IT Blanchardstown (ITB) were given a general screening during their induction into first year by the section of the National Learning Network[3] on the ITB campus. Included in this assessment is a section to gauge the student's personal learning preference and provide feedback on how best to proceed once this preference is known. Fig. 8 shows the learning preferences of the constituent students.

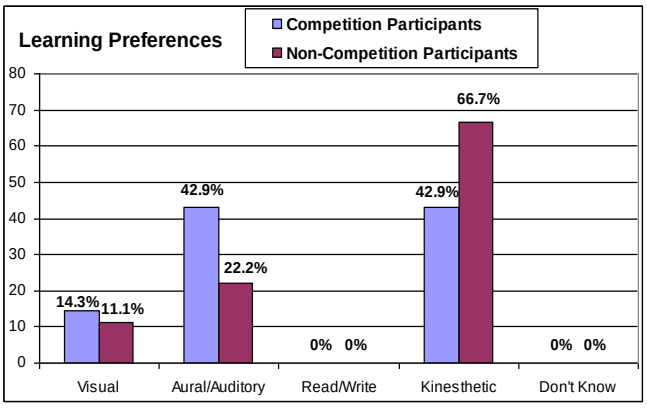

Fig. 8. Student Learning Preferences

While ITB students would have known in advance what their learning preference was, the remaining students surveyed were given the option to be directed to a secondary link in order to do a VARK test online to determine if they have a visual, aural/auditory, read/write or kinesthetic preference in obtaining information. However this was optional and while each preference type was explained in detail within the survey, students ultimately had the power to select any preference they felt was most suitable for them. The secondary link was merely provided to assist them in making this decision.

G. Lyons [10] did a study in 2001 based on student performances in exams and tutorials addressing the question of whether or not computer programming in itself is more suited to a specific kind of learning style. He also considers that it could be the traditional methods by which programming has been taught that caters to certain learning preferences rather than others.

Almost all competition students were either auditory or kinesthetic learners in contrast with the non-competing students who were mainly kinesthetic learners. The auditory learners among competing students would naturally benefit from discussions and social interactions with others. This would seem to correlate with what the lecturers said about the social dimension being a major attraction.

VII. Conclusions

Computer programming is perceived as a difficult subject to learn. Currently, the Irish school system offers no serious provision at second level for students to gain a solid background in computing before presenting themselves to a third level computing course. This can result in students being introduced to an abstract and often confusing experience. Programming requires much experimentation and hands on practical work on the part of the student to gain any significant level of skill. Students are required to think laterally and use a number of faculties such as mathematical ability and algorithmic skill.

From those surveyed about Robocode, competition was the foremost motivator for capable students as was the social/interactive aspects for all who took part. As a teaching tool, Robocode is currently being used in regular courses to incorporate standard pedagogical methods, particularly as an effective way to introduce objects and problem based learning. It seems to have had a good effect on those who described themselves as average or weak programmers. The good programmers enjoyed the opportunity to show off their skills while the less capable improved from peer interaction.

The national competition in Tipperary is now in its fifth year and seems set to continue. Robocode has its place for both junior and senior levels, particularly in providing a context for complicated mathematical theories of graphics and programming intelligent agents within the field of Artificial Intelligence. However, as a learning tool, it seems best suited to first and second year students. Further enhancements to the survey presented in this paper would be to extend the profile of students over number of years and monitor exam results in the months and years after taking part in the national Robocode competition to ascertain if it has had any visible effect.

What Robocode does suggest, is that when an eclectic mix of students is brought together for a specific purpose, it is possible for capable, average and moderate programmers to openly exchange insights. This brings programming to life for a variety of learning styles. Visual elements combined with the social dimension of team building gives rise to a very rich environment for learning.

[3] The National Learning Network Assessment Service was established in 2003 and provides assessment and learning support for adolescents and adults.

REFERENCES

[1] Alphaworks. *IBM's emerging technologies portal*. 2007. Available http://www.alphaworks.ibm.com

[2] D. Triplett, "*AlphaBot, An Interview with Robocode creator Mat Nelson".* DeveloperWorks, IBM's Resource for Developers. May 2002. Available: http://ww.ibm.com/developerworks/java/library/j-nelson

[3] J. O'Kelly, J. P. Gibson, "*RoboCode & problem-based learning: a non-prescriptive approach to teaching programming".* ACM SIGCSE Bulletin, v.38 n.3, September 2006

[4] B.Duch, *Writing Problems for Deeper Understanding*, pp. 47-53, Stylus Publishing, Sterling, Virginia, 2001.

[5] Robocode Ireland Organising Committee. 2008. Tipperary Institute, Thurles Tipperary, Ireland. Available: http://www.robocode.ie

[6] BlueJ, *The Interactive Java Environment*. 2007. Available: www.bluej.org

[7] D. J. Barnes, M. Kölling, *Objects First with Java. A Practical Introduction using BlueJ*. Third Edition, Prentice Hall / Pearson Education, 2006

[8] A. M. Phelps, C. A. Egert, K. J. Bierre. "*MUPPETS: multi-user programming pedagogy for enhancing traditional study: an environment for both upper and lower division students*". In Proceedings 35th Annual Conference of Frontiers in Education, 2005 (FIE '05).

[9] Y. Shichel, E. Ziserman, M. Sipper. "*GP-Robocode: Using Genetic Programming to Evolve Robocode Players".* In Proceedings of 8th European Conference on Genetic Programming, 2005 (EuroGP2005), Vol. 3447 of Lecture Notes in Computer Science, pp. 143-154, Springer.

[10] G. Lyons, *"Learning styles and performance in the introductory programming sequence"*. ACM SIGCSE Bulletin v.34 n.1, March 2002.