An Interactive Programming Assistance Tool (iPAT) for Instructors and Novice Programmers

Milan Amaratunga
and Gayan Wickramasinghe
Sri Lanka Institute
of Information Technology
Malabe, Sri Lanka
Email: milan.amaratunga@gmail.com

Milinda Deepal
and Oshani Perera
Sri Lanka Institute
of Information Technology
Malabe, Sri Lanka
Email: mdeepal@yahoo.com

Dilshan De Silva
and Samantha Rajapakse
Sri Lanka Institute
of Information Technology
Malabe, Sri Lanka
Email: dilshan.i@sliit.lk

Abstract—**This paper describes an Interactive Programming Assistance tool (iPAT) which is designed to assist students in solving introductory programming problems and help instructors in conducting programming lab sessions effectively. In a large computer lab setting with over 30 students, communication can be very limited between the students and the lab instructors. To address this problem iPAT was developed. It allows students to undertake programming exercises and receive interactive guidance in getting their programs to compile and run. This is the only tool developed to assist students and instructors in solving their practical lab session problems, which comes with features such as error handling, remote access, handling PC inventory and a solution archive to solve common errors in C# programming. To ensure the accuracy of the system, it was deployed in a LAN network of twenty computers which gave the idea of virtual lab environment. Then eighteen undergraduates were asked to rank the usefulness of the system based on a five point scale. According to the provided rankings an average rating of 4.3 was obtained.**

Index Terms - **Error handling,introductory programming, programming labs, online learning.**

I. Introduction

Computer programming requires correct understanding of programming languages, concepts and problem solving skills. Unfortunately, many novice programmers often face difficulties in their basic courses already. Butler and Morgan (2007) highlight that students in first year Information Technology find themselves in unfamiliar territory and the computing fundamentals can prove to be a learning challenge, particularly introductory programming courses [1]. According to a survey done by Lahtinen et al. (2005), most difficult issues that the programming students face are due to the lack of understanding of how to design a program to solve a certain task and to find bugs from their own programs [2]. Therefore, beginners need greater practice and more learning materials to develop their programming skills. The popular method of providing greater practice to students is to conduct practical programming labs.

When we consider a large computer lab setting with over 30 students, communication can be very limited between students and the lab instructors. As the number of students in the lab increases, the amount of time that the instructor can devote to each student decreases. Alammary, Carbonne and Sheard (2012) point out that within the limited time allocated to a practical session, the lab instructor might not be able to attend all the problems of each student [3]. Thus, students spend a long time trying to fix simple errors. On the other hand, instructors find themselves explaining the same mistakes over and over again.

All tools found in our literature survey were designed to provide feedback about students activities and performance in a lab session or to improve their programming knowledge using various tutoring systems. Tools were designed to evaluate the students progress during a lab session which can be useful for lab instructors. But no tools have been developed to assist students to correct the errors they encounter when writing programs during a lab session. This is where our tool becomes most essential. Our tool helps students to understand the error messages to correct mistakes on their own and it also allows the instructors correct the students code directly from their working machines from one central point. iPAT is essentially a tool to use in a computer lab which provides solutions to students to handle their programming errors and also assists instructors to conduct their lab sessions productively.

This paper describes an Interactive Programming Assistance Tool (iPAT) for introductory programming in C# computer labs. Section 2 details the related work and background studies. Section 3 gives an overview of the system, technology involvement and the evaluation approach that was used. Research results are provided on section 4, followed by the conclusion and suggestions for future work.

II. Related Work

A considerable number of tools have been developed by several universities to assist programming difficulties faced by beginners. Alammary, Carbonne and Sheard (2012) in Monash University have developed SmartLab which provides instructors timely and detailed feedback about their students' performance during lab sessions [3]. This tool helps tutors to better understand their students' problems and their progress in programming. However, it does not assist students to correct the errors in their codes. Another similar system is the datlab system developed at the University of Western Australia for monitoring student progress in computer science laboratories and providing timely feedback on their work [4].

Yoo et al. in Middle Tennessee State University have come up with a web based tutoring system which contains a question tutor, program tutor and a course management system [5]. Teachers who have access to the course management system can collaborate on managing lab materials. The question tutor and the program tutor address the aspect of learning concepts and programming skills. Unlike our tool, they do not concentrate on providing solutions for the students' errors.

Another body of work has focused on providing automated help to students. An example is 'WebToTeach' developed by Arnow and Barshay (2002) which is a web-based interactive programming exercise system that enables automated checking of students assignments [6].

To reduce the incidence of common programming errors, a certain tool named Arjen has implemented to identify common errors in programs submitted by novice programmers and returns descriptive error information and advice to them [7]. But Arjen is written to target one specific programming language, Java. A certain tool called 'Expresso' which is targeted for the novice Java programmers, simplifies the Java compiler messages into more detailed and easy to read error messages [8]. However, still no tool has implemented to offer suggestions on how to fix errors in a C# programming environment. So there is a need of a tool to assist novice programmers who work with C# language in Visual Studio IDE.

By considering all the studies done so far 'iPAT' is the only tool developed which comes with features such as error handling, screen sharing, a lab resource management system and a solution archive to solve common errors in C# programming. It improves the interactivity of the students and the instructors in a C# programming lab class by allowing the instructor to correct any student error from one central point in the class. The plug-in we have built for the visual studio that extracts the error list from the IDE is another unique feature in the proposed system. With our enhanced lab assistance tool, the error handling feature and the online knowledge sharing environment, we hope to bring a new approach on tutoring C# programming in a much easier and more effective way.

III. Research Methodology

This research was carried out with the intension of building an Interactive Programming Assistance Tool which is designed to teach and learn programming in an effective way. The tool does not require any special hardware support and it can be easily installed in a networked computer lab. The research was mainly focused on the difficulties that arise when writing programs in Visual Studio which is the most popular IDE for C# programming. To provide a new experience to the students, the system was developed by including a lab assistance tool which consists of a PC inventory, an error handling module and a vast collection of learning materials for basic programming.

A. *Handling the PC Inventory*

For the administrative purposes, the lab administrator has to know hardware and software details of each and every workstation in the lab. Therefore, without asking a user to insert data for each and every pc, there is a component in iPAT to handle the PC inventory automatically. The aim of the PC Inventory component is to retrieve the hardware and the software details of the each and every PC to the database. When a user executes this component in a workstation with the use of Windows Management Instrumentation (WMI) queries, iPAT will retrieve the relevant details of the workstation and save it to the database. iPAT will grab the information about the operating system, hard disks, memory and the network details such as the IP address and the machine name.

B. *Error Handling Module*

The Error handling module consists of two sub modules, the add-on for the Visual Studio 2010 and the error handling module in the iPAT application.

C. *iPAT Add-on*

The iPAT add-on has to be installed separately to the Visual Studio 2010 IDE. It is specifically designed for the Visual Studio 2010 environment in order to pass the error list to the iPAT application which is running as a separate process. The main use of this add-on is to get the location of the current working solution in the visual studio environment. When a student gets an error when trying out a lab exercise, he/she can use the iPAT add-on under the tools menu in Visual Studio. When a student clicks on the add-on, it will grab the location of the current working project in the Visual Studio and write it to a file which is monitored by a FileSystemWatcher in order to capture the project path. (In C#, FileSystemWatcher tracks file system change notifications and raises events when a directory, or file in a directory, changes). Then the iPAT application will compile and run the project using Microsoft Build Engine and Microsoft Build Framework. Next the application will display the error list and the error code of the current working project. Screenshots of these interfaces are displayed under the Research Results heading. Using this interface students are able to get assistance by referring to the error solution archive which consists of the following three categories;

1) Solutions by directly correcting the source code which is having the error. In this category, code comparison mechanism is used in order to identify the corrections done to the source code.
2) Solutions through a web link. There are plenty of freely available resources in the web for basic programming

errors. iPAT has an inbuilt web browser to view these hyperlinks.

3) Solutions through a video, preferably Youtube. iPAT application has an inbuilt flash player to play the videos. In addition, there is an inbuilt screen capturing facility to create videos.

All the solutions have a rating mechanism to identify the best solution for a given error. Students can rate the solution to indicate whether its useful or not. If a student is not satisfied with the given solutions, he/she can get help from the instructors using the next option available in the interface mentioned above. If a student has selected this option, first it will write the error details and the error list to the database and then it will create a zip file of the current working project, which will be uploaded to the ftp server with a unique id. Finally it will send a notification to all the users who has the permission to correct the errors. As the projects are uploaded to ftp server, it makes it possible for the users who are going to give a solution to a particular error to download the project and run it. This will be very useful when handling the runtime errors. There is also a facility to upload the corrected project again to the ftp server which will be automatically redirected to the student who has asked for a solution.

D. Error handling in iPAT

Posted errors will be displayed in the home interfaces of the instructors. The Instructors can view the error list and the posted user's comment in order to identify the requirement of the posted user. Solutions can be given under three categories;

1) Quickest way to give a solution is directly editing the posted code using the code editor provided by the iPAT.
2) User can download the complete project from the ftp server and edit it through the Visual Studio and re upload it to the ftp server.
3) Submit a web link or a youtube video link which will be helpful to correct the errors.

E. User Roles

According to the four major user categories that interact with our tool, we have created four core components in our system; the lab instructor, the lab student, the web instructor and the web student.

Lab Instructor: The instructor or the lecturer using the system during a lab session. He can upload necessary lab materials to our system and provide solutions for the student errors as a source code, as a source project, as a web link or as a video link. However, as the time duration in a lab session is limited, our system provides the instructor an easy way to access the students screen remotely and fix the errors in the students code directly.

Lab Student: The student who is using the system during a lab session. When a student encounters errors in the code, there are two ways to get assistance. He/she can view solutions from our solution archive or else he/she can send the error list to the instructor via the iPAT plug-in which we have embedded to the Visual Studio IDE. If he/she is unable to view the projector screen from a distance, there is always the option to access the instructors PC remotely and view the screen.

Web Instructor: The web instructor can be of two types;

1) Lab instructor using the web instructor features outside the lab.
2) Volunteer lecturers using the web instructor features.

The volunteer lecturers who are registered in the system have the opportunity to share their expert knowledge. They provide answers to the errors that the web students have submitted. They can submit MCQs, quizzes and articles related to the most difficult programming concepts. Provided that they have permission from the registered institutes, they can upload certain tutorials and course materials with public access.

Web Student: The web student can be of two types;

1) Lab student using the web student features outside the lab.
2) External students using the web student features.

F. Technology Involvement

The technologies deployed to develop this tool include Microsoft Visual Studio 2010 and SQL Server Management Studio 2008. The .NET extension which was built to extract the error list from the Visual Studio IDE is one of our successful implementations. The latest version of the Virtual Network Computing (RealVNC 4.1) was used to establish remote access to every student machine in a lab. RealVNC, the inventor of VNC remote access technology, was used to provide remote access software for desktop and mobile platforms. To accomplish file synchronization, file download, upload and delete, FTP server was used.

G. System Evaluation

We have adopted some decisive measures to comply with our targeted usage requirements. To ensure the accuracy of the system, the evaluation of the iPAT was carried out in a virtual lab environment which was created using 20 computers with the involvement of 2 senior lecturers and 18 undergraduates. The lecturers are with more than 10 years of experience in teaching programming units and the 18 information technology undergraduates are currently in their 1st year. With the inclusion of these participants a one hour lab session was conducted. During the lab session both lecturers and the students had the opportunity to take on the role as lab instructor and lab student. At the end of the lab session participants were interviewed individually in order to find out their attitude towards the iPAT application. When obtaining the feedback our main concern was to assess the usefulness and the effectiveness of the functionalities used in iPAT. Feedback session was divided into 2 sections; the lecturers evaluation and the students evaluation.

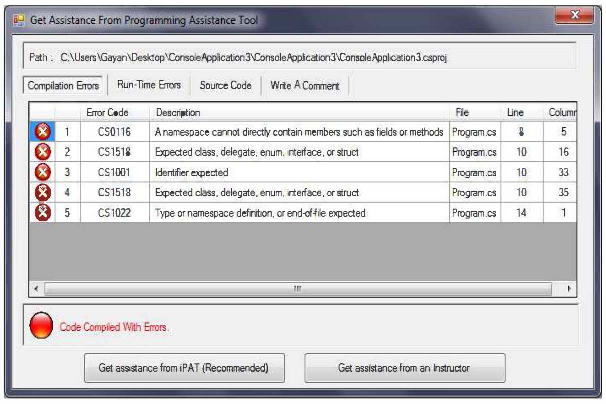
In the lecturers evaluation, they were asked to rank the usefulness of the system on a five point scale giving one to indicate that the system is not at all useful and five to indicate that the system is extremely useful. According to rankings provided by the lecturers, an average rating of 4 was obtained. The lecturers commented that they recommend this tool to be installed in the programming labs of the institute. They also informed that without directly redirecting the error list from the Visual Studio environment to iPAT, the students should try to solve the errors by their own and only if they were unable to correct the errors, they can get the help from iPAT. It is a good technique from the students perspective as students will have opportunity to try and handle their own errors before getting help from the instructors. Also the lecturers highlighted that the error solution archive which includes similar errors and solutions is very helpful for students. It will help to understand their errors and correct them by their own without asking the help from the instructor. When considering the remote access feature, they pointed out that the opportunity to connect to multiple student machines at once is a very useful functionality. It will help the instructors to correct similar kind of errors very easily.

In the students evaluation, the undergraduates gave an average of 4.3 rating. They positively commented on the solution achieve mentioning that the mechanism used to display the code comparison is a very useful mechanism because even a person without programming knowledge can easily understand the modifications that has to be done to correct errors. One student commented that according to his knowledge the facility available to upload the project and get the errors corrected is not available in the most common programming assistance sites available in the internet. He said that the feature is not only useful for the 1st year students but also for students in all years as well. Almost all stated that the ability to view the lecturers screen using the screen sharing facility is highly valuable. They admitted that normally in large lab classes even though the instructors computer is provided with the projector, instructors have to use large font size in order to make the code visible to all the students in the lab.

IV. Research Results

The proposed system was successfully built with unique features and functions which serve different needs of different users of the system.

The add-in we have built in the Visual Studio IDE successfully extracts the error list and the error code of the current working project, as shown in Fig 1. It gives the student two solutions; to get assistance from our solution archive or get help from the instructor.

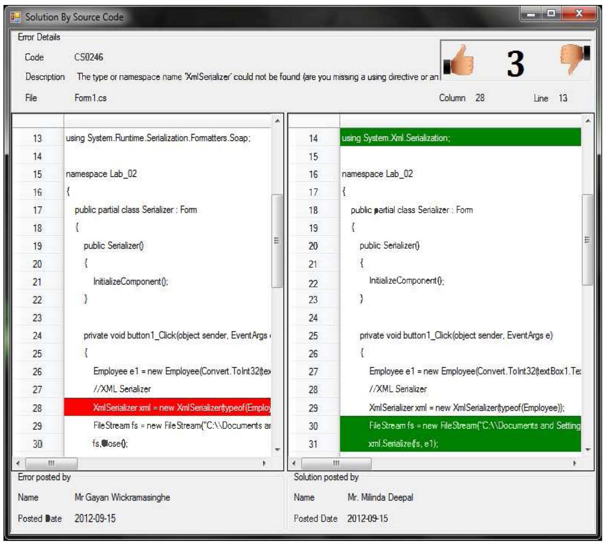

Get Assistance From Programming Assistance Tool
Compilation Errors
Run-Time Errors
Source Code
Write A Comment
Get assistance from iPAT (Recommended)
Get assistance from an Instructor

Path : C:\Users\Gayan\Desktop\ConsoleApplication3\ConsoleApplication3\ConsoleApplication3.csproj

		Error Code	Description	File	Line	Column
⊗	1	CS0116	A namespace cannot directly contain members such as fields or methods	Program.cs	8	5
⊗	2	CS1518	Expected class, delegate, enum, interface, or struct	Program.cs	10	16
⊗	3	CS1001	Identifier expected	Program.cs	10	33
⊗	4	CS1518	Expected class, delegate, enum, interface, or struct	Program.cs	10	35
⊗	5	CS1022	Type or namespace definition, or end-of-file expected	Program.cs	14	1

Code Compiled With Errors.

Fig. 1. Snapshot of the Add-in



Error Details
Code CS0246
Description The type or namespace name 'XmlSerializer' could not be found (are you missing a using directive or an
File Form1.cs
Column 28 Line 13

```
using System.Runtime.Serialization.Formatters.Soap;

namespace Lab_02
{
  public partial class Serializer : Form
  {
    public Serializer()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs
    {
      Employee e1 = new Employee(Convert.ToInt32(tex
      //XML Serializer
      XmlSerializer xml = new XmlSerializer(typeof(Employ
      FileStream fs = new FileStream("C:\\Documents ar
      fs.Close();
```

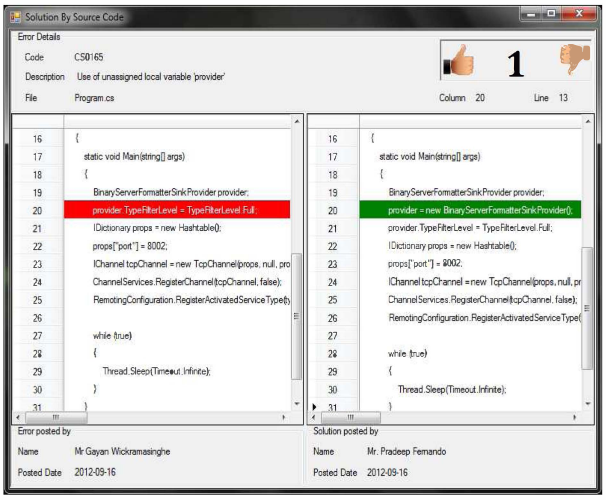
```
using System.Xml.Serialization;

namespace Lab_02
{
  public partial class Serializer : Form
  {
    public Serializer()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
      Employee e1 = new Employee(Convert.ToInt32(textBox1.Te
      //XML Serializer
      XmlSerializer xml = new XmlSerializer(typeof(Employee));
      FileStream fs = new FileStream("C:\\Documents and Setting
      xml.Serialize(fs, e1);
```

Error posted by
Name Mr Gayan Wickramasinghe
Posted Date 2012-09-15

Solution posted by
Name Mr. Milinda Deepal
Posted Date 2012-09-15

Fig. 2. Snapshot of the solution by source code

Our solution archive provides possible remedies to a common set of errors. The solutions are provided as a source code as shown in Fig 2 and Fig 3, as a web link, as a video tutorial or as a Visual Studio project.



Error Details
Code CS0165
Description Use of unassigned local variable 'provider'
File Program.cs
Column 20 Line 13

```
{
  static void Main(string[] args)
  {
    BinaryServerFormatterSinkProvider provider;
    provider.TypeFilterLevel = TypeFilterLevel.Full;
    IDictionary props = new Hashtable();
    props["port"] = 8002;
    IChannel tcpChannel = new TcpChannel(props, null, pro
    ChannelServices.RegisterChannel(tcpChannel, false);
    RemotingConfiguration.RegisterActivatedServiceType(ty

    while (true)
    {
      Thread.Sleep(Timeout.Infinite);
    }
  }
```

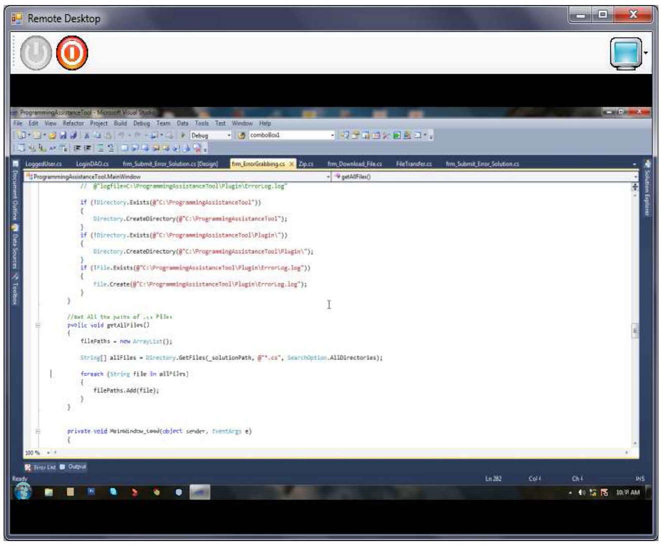
```
{
  static void Main(string[] args)
  {
    BinaryServerFormatterSinkProvider provider;
    provider = new BinaryServerFormatterSinkProvider();
    provider.TypeFilterLevel = TypeFilterLevel.Full;
    IDictionary props = new Hashtable();
    props["port"] = 8002;
    IChannel tcpChannel = new TcpChannel(props, null, pr
    ChannelServices.RegisterChannel(tcpChannel, false);
    RemotingConfiguration.RegisterActivatedServiceType(

    while (true)
    {
      Thread.Sleep(Timeout.Infinite);
    }
```

Error posted by
Name Mr Gayan Wickramasinghe
Posted Date 2012-09-16

Solution posted by
Name Mr. Pradeep Fernando
Posted Date 2012-09-16

Fig. 3. Snapshot of the solution by source code

The screen sharing feature is available if the student needs assistance from the instructor. The student submits the errors to the lab instructor so that he can view the students screen and correct the errors via remote access as shown in Fig 4.



```
        //  @"logfile=C:\ProgrammingAssistanceTool\Plugin\ErrorLog.log"

        if (!Directory.Exists(@"C:\ProgrammingAssistanceTool"))
        {
            Directory.CreateDirectory(@"C:\ProgrammingAssistanceTool");
        }
        if (!Directory.Exists(@"C:\ProgrammingAssistanceTool\Plugin\"))
        {
            Directory.CreateDirectory(@"C:\ProgrammingAssistanceTool\Plugin\");
        }
        if (!File.Exists(@"C:\ProgrammingAssistanceTool\Plugin\ErrorLog.log"))
        {
            File.Create(@"C:\ProgrammingAssistanceTool\Plugin\ErrorLog.log");
        }
    }

    //Get All the paths of .cs Files
    public void getAllFiles()
    {
        filePaths = new ArrayList();

        String[] allFiles = Directory.GetFiles(_solutionPath, @"*.cs", SearchOption.AllDirectories);

        foreach (String file in allFiles)
        {
            filePaths.Add(file);
        }
    }


    private void MainWindow_Load(object sender, EventArgs e)
    {
```

Fig. 4. Snapshot of the Remote Desktop

V. Conclusion

According to Gomes and Mendes, problems start in the early stages of learning programming when students are trying to understand and apply basic programming concepts [9]. One way to assist students is by providing them with programming tasks to solve during lab session with the support of an instructor. However, within the limited time allocated for a lab session, it is practically impossible for a lab instructor to look into each mistake that each student makes. To make things worse most of the time they find themselves explaining the same mistakes over and over again. This is the exact problem that we are addressing from our project through a fully integrated desktop application which allows students to undertake programming exercises and to receive guidance in getting their programs to compile and run.

The recommended users of this tool are the people who are involved in teaching and learning programming such as novice programming students, lab instructors, programming lecturers, volunteer lecturers etc. In addition to that, anyone who is interested in learning and teaching basic programming can benefit from this software tool.

Concluding the whole idea behind this research project, it can be said that what we have developed is a well structured programming tool which enables the student programmers and the lab instructors to perform their tasks more efficiently and effectively. Our intention is not just to improve the overall effectiveness of the lab sessions but also to facilitate the communication between the students and the tutors by providing them with an opportunity to share their expertise knowledge and experience in programming among them and thereby build a better and a sound programming society.

VI. Future Work

This study has opened up new research areas for further improvements. The proposed programming assistance tool could be extended in following ways:

- Create online exams for lab students and implement a method to grade the student answers automatically.
- Graphically represent the summary of each students programming activity. This may show the history of compiling attempts, implemented actions and the amount of time spent on each exercise.
- Provide a detailed analysis on each students common programming mistakes and provide automated guidelines and hints on the skills that he/she needs to be improved.

Acknowledgment

We take this opportunity to express our deep sense of gratitude to those who contributed in many ways to make our undergraduate research a success. Our special thanks go out to all the staff members of the SLIIT Malabe campus for their valuable suggestions, opinions and guidance to complete the project successfully. The team extends their sincere gratitude to all colleagues, friends and families who continued to give their insights, patience, support and co-operation which motivated us in reaching greater heights.

References

[1] M. Butler and M. Morgan, "Learning challenges faced by novice programming students studying high level and low feedback concepts", in ICT: *Providing choices for learners and learning. Proc. ascilite Singapore 2007.* [Online]. Available: http://www.ascilite.org.au/conferences/singapore07/procs/ butler.pdf. [Accessed: 16 Nov 2012].

[2] E. Lahtinen, K. Ala-Mutka, and H. M Jrvinen, "A study of the difficulties of novice programmers", in *Proc. 10th annual SIGCSE conference on Innovation and Technology in Computer Science Education* (ITiCSE05), New York, USA, pp.14-18.

[3] A. Alamary, A. Carbone and J. Sheard, Implementation of a Smart Lab for Teachers of Novice Programmers, in *Proc. 14th Australasian Computing Education Conference*(ACE2012), Melbourne, Australia.

[4] C. MacNish, "Evolutionary programming techniques for testing students code", in *Proc. Australasian conference on Computing education (ACSE 00)*, Ainslie E. Ellis (Ed.). New York, USA, pp.170-173.

[5] J. Yoo, C. Pettey, S. Yoo, J. Hankins, C. Li, and S. Seo, 2006, "Intelligent tutoring system for CS-I and II laboratory", in *Proc. 44th Annual South East Regional Conference* (ACM-SE 44), New York, USA, pp.146-151.

[6] D. Arnow and O. Barshey, "WebToTeach: An interactive focused programming exercise system", in *Proc. 29th ASEE/IEEE Frontiers in Education Conference*(FIE 1999), San Juan, Puerto Rico.

[7] W. Toomey, "Quantifying the Incidence of Novice Programmers Errors", School of IT, Bond University.

[8] M. Hristova, A. Misra, M. Rutter, and R. Mercuri, "Identifying and correcting Java programming errors for introductory computer science students", in *Proc. SIGCSE technical symposium on Computer Science Education* (SIGCSE '03), New York, USA, pp.153-156.

[9] A. Gomes and J. Mendes, "Problem Solving in Programming", in *19th Annual Workshop of the Psychology of Programming Interest Group* (PPIG2007), pp.216-228.