Journal of Information Technology Education: Research

*An Official Publication of the Informing Science Institute InformingScience.org*

*JITEResearch.org*

**Volume 16, 2017**

# COMPARISON OF TRADITIONAL AND ADRI BASED TEACHING APPROACHES IN AN INTRODUCTORY PROGRAMMING COURSE

| | | |
|---|---|---|
| Sohail Iqbal Malik * | Buraimi University College, Al-Buraimi, Oman | sohail@buc.edu.om |
| Jo Coldwell-Neilson | Deakin University, Geelong, Australia | jo.neilson@deakin.edu.au |

* Corresponding author

## ABSTRACT

| | |
|---|---|
| Aim/Purpose | This study introduced a new teaching and learning approach based on an ADRI (Approach, Deployment, Result, Improvement) model in an introductory programming (IP) course. The effectiveness of the new teaching and learning process was determined by collecting feedback from the IP instructors and by analyzing the final exam grades of the course. |
| Background | Learning to program is considered a difficult and challenging task for a considerable number of novice programmers. As a result, high failure and dropout rates are often reported in IP courses. Different studies have been conducted to investigate the issue. One of the reasons for this challenge is the multiple skills that students have to master in order to be able to build programs. These skills include programming knowledge and problem-solving strategies and being able to pay equal attention to these required skills in the IP course. |
| Methodology | A focus group was conducted to obtain feedback from the IP instructors about the ADRI approach. The performance of the students who had completed the IP course before ADRI was compared with those who used the ADRI approach by undertaking a comparative analysis of their final exam grades. |
| Contribution | The study demonstrates that the new teaching and learning approach based on the ADRI model encourages students to pay equal attention to programming knowledge and problem-solving strategies, discouraging programming shortcuts and reducing high attrition rates (failure and dropout) in the IP course. |
| Findings | The results of the focus group show that the instructors preferred the ADRI approach compared to the traditional approach. The final exam grades show that the students performed better in semesters which offered the ADRI approach as compared to those semesters without this approach. |

Accepted by Editor Benson Soong | Received: March 14, 2017 | Revised: May 29, June 15, July 5, 2017 | Accepted: July 26, 2017.

Cite as: Malik, S. I., & Coldwell-Neilson, J. (2017). Comparison of traditional and ADRI based teaching approaches in an introductory programming course. *Journal of Information Technology Education: Research, 16,* 267-283. Retrieved from http://www.informingscience.org/Publications/3793

(CC BY-NC 4.0) This article is licensed it to you under a Creative Commons Attribution-NonCommercial 4.0 International License. When you copy and redistribute this paper in full or in part, you need to provide proper attribution to it to ensure that others can later locate this work (and to ensure that others do not accuse you of plagiarism). You may (and we encourage you to) adapt, remix, transform, and build upon the material for any non-commercial purposes. This license does not permit you to use this material for commercial purposes.

| | |
|---|---|
| Future Research | Future research will explore the ADRI approach in other fields of computer science studies, such as database and data structure, to determine if its impact has a wider application than just teaching introductory programming. |
| Keywords | introductory programming course, ADRI approach, teaching approach, students learning outcomes, failure and dropout rates |

# INTRODUCTION

The introductory programming (IP) course is one of the first courses novices take in the computer science discipline (Matthiasdottir 2006). Novices have to take it to progress through their study plan. Many face difficulties, particularly in the first few weeks of their programming course. They are often unsuccessful when attempting to write computer code that meets the stated objectives (Carbone, Hurst, Mitchell, & Gunstone, 2009). The whole situation de-motivates them and causes frustration which leads to disengagement with the curriculum. As a consequence, high failure and dropout rates are reported (Watson & Li, 2014; Zingaro, 2015). Papp-Varga, Szlávi, and Zsakó (2008) argue that ICT teaching is a comparatively new 'problem domain' compared to other already established subjects like physics or mathematics. So its teaching methodology is not well established or formulated and, as a consequence, different teachers are using their own 'blend of methods' (p. 163).

Buraimi University College, Oman was the context of learning for this study. The IP course is a core course offered at level 1 for all students in the IT department. Most of the students do not have any prior experience in programming and, to add to the difficulties, English, the language of instruction at Buraimi, is the second language for them. A teacher-centred pedagogy (referred to here as 'the traditional approach') is used in teaching the introductory programming course. Lecturers deliver the classes and conduct lab sessions. The programming exercises and examples presented during the classes and lab sessions promote programming shortcuts, as described by Dale and Weems (2010), where students attempt to convert a problem statement directly into a computer code.

*Problem Statement* → *Code*

This study introduced a new teaching and learning approach, based on the ADRI model, in the introductory programming course specifically to discourage programming shortcuts. Further, the new teaching and learning approach supported novices to pay equal attention to both programming knowledge (syntax and semantics) and problem-solving skills. The new approach was compared with the traditional approach used previously in the course to determine its effectiveness in the teaching and learning process.

This paper is organised into a number of sections. It starts with a review of relevant literature, followed by an introduction to the ADRI model. The methodology used in this study is then described and results are reported and discussed. The paper concludes with a summary of the study outcomes and suggestions for further work.

# BACKGROUND

## *PROGRAMMING TEACHING METHODS*

A teaching method involves the methods and principles used to instruct students in achieving the desired learning as it is implemented by teachers. An overview of some of the existing programming teaching methods is as follows:

### Problem-based learning

Nuutila, Törmä, and Malmi (2005) explained that problem-based learning (PBL) engaged students in problem-solving. In PBL, the students face real world problems which help them to enhance their "disciplinary knowledge, higher order thinking and practical skills" (Mohorovicic & Strcic, 2011; p. 49).

Seven steps are involved in the implementation of PBL (Nuutila et al., 2005) where students work in groups. They analyze the problem and determine what they already know about it. They sketch an initial model for the problem to identify important concepts and relations. They determine what they need to learn to solve the problem fully. After that, the students work independently to gain the required knowledge. The students then convene and discuss what they have learned in order to solve the problem. Finally, they elaborate on their solutions.

Researchers concluded that PBL enhances the retention time of students' knowledge up to several years and students' performance in follow-up courses is better than those who attended traditional IP courses. It also helps to enhance students' creative thinking and motivation (Nuutila et al., 2005).

### Puzzle-based learning

The main aim of puzzle-based learning (PZBL) is to teach students problem-solving and critical thinking techniques in a fun way (Merrick, 2010). In programming courses, PZBL is used to encourage students to think about framing and solving unstructured problems. The problem solution is divided into a number of puzzle pieces which students use to reconstruct the program by putting the puzzle pieces in the correct order (Yoneyama et al., 2008).

A research study conducted by Merrick (2010) showed that PZBL increased students' interest and active participation in the programming course.

### Pair Programming

In pair programming, two programmers work on the same program (code) side by side at the same computer. Both programmers are involved in planning, designing, and testing the program. Each programmer has a different role in pair programming. One programmer works as a driver and other as a navigator. The driver is typing actual code and while the navigator is observing his/her work for errors, offering suggestions and alternative solutions. The role of the driver and navigator is swapped at regular intervals to ensure that both students are equally and actively engaged in the development process.

Researchers suggested that pair programming motivated both programmers. They devised solutions in less time and with fewer errors than more traditional approaches to coding, and they approached collaboration in a positive way (Zacharis, 2011). However, some studies revealed that pair programming could be exhausting and irritating. So appropriate care should be taken when organizing the pairs as different skill levels between pair programmers also affects collaboration (Chaparro, Yuksel, Romero, & Bryant, 2005).

Huet, Pacheco, Tavares, and Weir (2004) discussed that some lecturers feel that pair programming is less efficient than individual work for introductory programming courses. They observed that one student actually programs while the other student becomes an observer. On the other hand, some lecturers like pair programming because it promotes learning, and one of the students can serve as a tutor for the other.

### Game-themed Programming

Game-themed Programming (GTP) is a teaching approach in which abstract programming concepts are taught to the students by exploring small game applications. The main aim of GTP is not to teach game programming to the students, but help them to understand programming concepts through simple game assignments (Sung et al., 2011).

A study revealed that success rates in GTP classes were higher than other traditional classes. Students were engaged in devising solutions for game assignments which increased their motivation and enthusiasm (Sung et al., 2011).

Miljanovic (2015) used game based learning in teaching debugging to novices. He prepared and introduced a game, RObоBUG, to novices. He discussed that debugging is a critical skill that novices should acquire early in their programming career otherwise they spend hours attempting to fix errors in their programs. The results showed a positive impact on the learning process of novices.

The above discussion on some existing programming methods shows that most of them focus on either problem-solving skills or programming knowledge (syntax and semantics). On the other hand, novices need to focus equally on problem-solving skills and programming knowledge. The current study, which introduced the ADRI approach, encourages students to focus equally on both sets of skills (problem-solving and programming knowledge).

### *RESEARCH QUESTIONS*

It is evident from the previous discussion that IP courses have a long history of interventions to improve students' learning outcomes but high failure and dropout rates from them are consistently reported in different studies (Guzdial & Soloway, 2002; Lahtinen, Ala-Mutka, & Järvinen, 2005; Sykes, 2007; Watson & Li, 2014; Yadin, 2011; Zingaro, 2015). Therefore, it is important to address the issues of high failure and dropout rates because it affects the retention and recruitment of the students in the computer science discipline. This research project proposes the ADRI approach in the teaching and learning process of the IP course to address the issues of high failure and dropout rates. Moreover, this research proposes a number of research questions to determine the impact of the ADRI approach compared to the traditional approach, two of which are the focus of the study being reported here.

The two research questions being addressed in this research study are:

> *RQ1. What are the perceptions of introductory programming instructors regarding the ADRI approach in their teaching process?*
>
> *RQ2. What is the impact of the ADRI approach compared to the traditional approach on the students' performance in the introductory programming course?*

## INTRODUCTION TO THE ADRI MODEL

The ADRI (Approach, Deployment, Result, and Improvement) model is a well-known quality assurance model. It is used extensively in the education and business sectors (Razvi, Trevor-Roper, Goodliffe, Al-Habsi, & Al-Rawahi, 2012). Australian and New Zealand universities used it for quality audit processes. Moreover, the Australian Business Excellence Framework and New Zealand Business Excellence Foundation used it to evaluate quality in Business Excellence Awards (Carroll & Razvi, 2006). It is based on the Plan-Do-Check-Act (PDCA) model developed by Walter Shewhart (Moen & Norman, 2010).

Malik and Coldwell-Neilson (2016) proposed the four stages of the ADRI approach to enhance students learning outcomes in the IP course. They explained that the first stage (Approach) covers problem solving skills (pseudocode and flowchart), the second stage (Deployment) emphasizes programming knowledge (syntax and semantics), the third stage (Result) explains input, output, and process used to solve a problem statement, and the fourth stage (Improvement) emphasizes different programming constructs. Figure 1 shows the four stages of the ADRI approach.

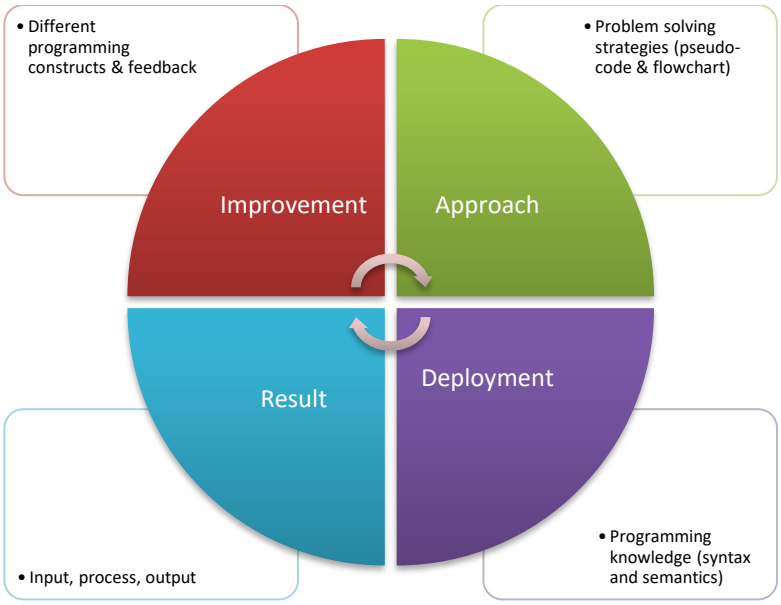



**Figure 1: Four stages of the ADRI model**
(Iqbal & Harsh, 2013; Malik & Coldwell-Neilson, 2016)

## RESEARCH METHODOLOGY AND DESIGN

The first research question was investigated by conducting a focus group with the instructors of the IP course offered with the ADRI approach. The second research question was explored by comparing the final grades of students enrolled in four semesters of the offering of the IP course from 2013 to 2015, both before and after the ADRI approach was implemented.

### *ETHICAL CONSIDERATIONS*

In this study, the IP students and instructors were involved for data collection. So it was important to take care of their privacy, consent, and confidentiality. The IP students' grades for semesters 1 and 2, 2013-14 and semesters 1 and 2, 2014-15 were accessed and analyzed. A focus group session was conducted with the instructors after introducing the ADRI approach in the IP course. This focus group was conducted at the end of the semester 1, 2014-15, with the instructors who offered the IP course with the ADRI approach as participants in the focus group.

To address the ethical issues in this study, approval has been granted by Buraimi University College to collect the data. Data relating to students' grades was de-identified by an independent party (Registration Department at Buraimi University College) to maintain anonymity. During the focus group, notes were taken by the facilitators and participants' responses were collected anonymously.

### *INTRODUCTORY PROGRAMMING COURSE MATERIALS BASED ON THE ADRI APPROACH*

Learning resources used in the IP course were redesigned based on the four stages of the ADRI approach. All the programming examples used in lectures and the programming problems set as laboratory exercises were prepared based on the four stages of the ADRI approach. An ADRI based editor was developed to support the ADRI approach and to assist students in the preparation of programming solutions using the ADRI approach.

All the programming examples and problems based on the ADRI approach have five parts as shown in Table 1. The first part contains a problem statement. The second part (Approach) deals with problem-solving strategies such as pseudocode and flowchart diagrams. The third part (Deployment)

covers the syntax and semantics of the programming language. The fourth part (Result) deals with the program inputs, the process used to solve a problem statement, expected outputs and examples of common syntax and semantic errors. The fifth part (Improvement) provides more practice with different programming language constructs (Malik & Coldwell-Neilson, 2017).

**Table 1. Programming example based on four stages of ADRI approach**

**Write a program that will read the radius of a circle then calculate the area and circumference of the circle and print area and circumference.**

**First Part**

**Area = PI * Radius * Radius**

**Circumference = 2 * PI * Radius**

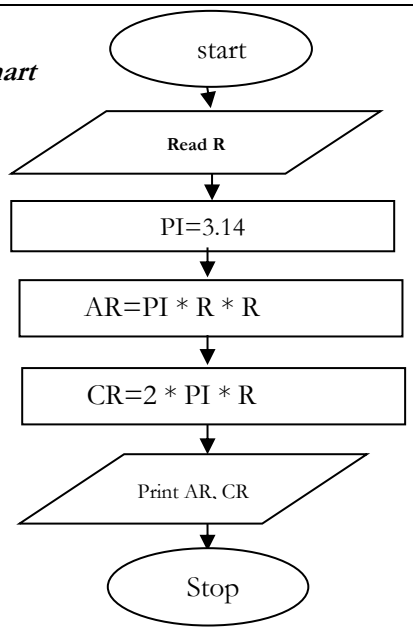
***Step 1: Approach – Problem-solving strategies***

***Pseudo-code***

1. *Start*
2. *Read* radius R
3. PI = 3.14
4. *Calculate* area (AR) = PI * R * R
5. *Calculate* circumference (CR) = 2 * PI * R
6. *Print* AR, CR
7. *Stop*

Second Part

***Flowchart***




**Step 2: Programming Knowledge**

Third Part

```
#include<iostream>
using namespace std;
int main()
{
float R, AR, CR;
const float  PI  = 3.14;
cout<<"Enter Radius: ";      cin  >> R;
AR = PI * R * R;
CR = 2 * PI * R;
cout<<"Area="<<AR<<endl<<"Circumference=" <<CR;
return 0;
}
```

| **Step 3: Result** | **Syntax Error: Incompatible data types / Initialization** |
|---|---|
| **Expected output:** Enter Radius: 5<br>Area = 78.5<br>Circumference= 31.4<br>**Process:** AR = 3.14 * 5 * 5<br>CR = 2 * 3.14 * 5<br>**Achieved output:**<br>**Goals:** Achieved: ☐ Not Achieved: ☐ | **Example**:<br>3 int num = "Hello World";<br>**Reason**: integer value is expected in variable num on line 3. The string value (Hello World) is assigned to an integer variable (num). |
| **Step 4: Improvement**<br>**Update above program so that it also calculates and prints diameter of the circle.**<br>**Diameter = 2 * Radius**<br>**Expected Output:** Enter Radius: 4<br>Diameter = 8 | |

Fourth Part

Fifth Part

The ADRI based editor was designed and developed to specifically emphasize the four stages of the ADRI approach with separate interfaces for each stage. Figure 2 shows one of the user interfaces of the ADRI editor. It consists of six menus (File, Edit, ADRI, Question, Topic and Execute) and each menu has further sub-menus which provide additional functionality.

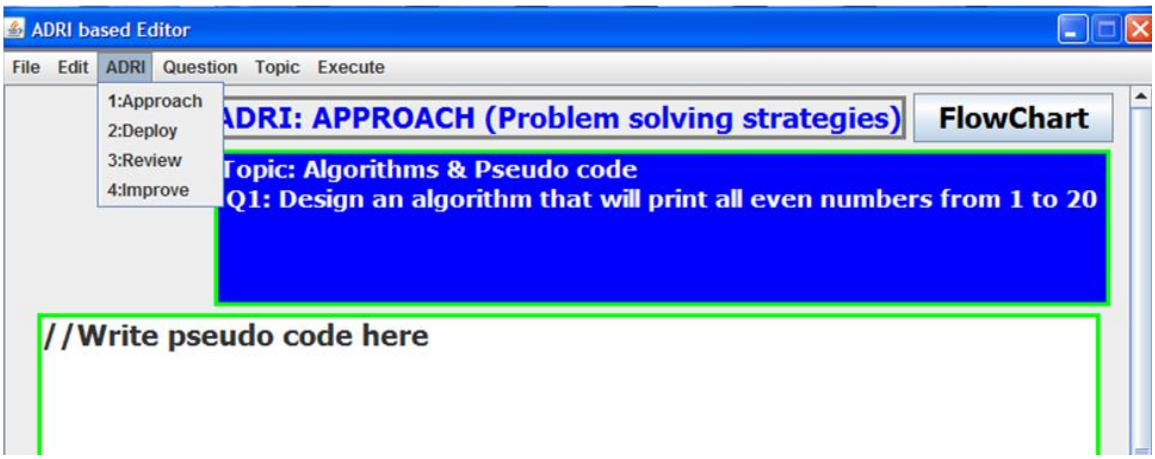



**Figure 2. Interface of ADRI based Editor** (Malik, 2016)

## FOCUS GROUP WITH INTRODUCTORY PROGRAMMING INSTRUCTORS

In this section, we address the first research question which is:

> *RQ1. What are the perceptions of introductory programming instructors regarding the ADRI approach in their teaching process?*

A focus group was conducted with the IP course instructors to seek in-depth feedback on the affordances and barriers of the ADRI approach for students. The focus group consists of five participants including the moderator and assistant moderator. The focus group lasted for 90 minutes and

was not audio-recorded. The moderator and assistant moderator took notes, anonymously and independently, of the discussion during the session. They compared their notes after the session and produced a joint session report. This report was sent to the participants to comment on the accuracy of the reporting. Since participants' responses were noted anonymously, it was not possible for participants to withdraw their statements after the focus group had finished although they could suggest amendments to the record of the focus group in the report.

### *FOCUS GROUP OBJECTIVES*

This research was based on the didactic triangle and instructors were one of the three entities in it. They taught the curriculum to the students who were the other two entities in the didactic triangle (curriculum and student). In our context, instructors had a close interaction with the ADRI approach and the students. So it was beneficial to obtain their feedback on the affordances and barriers of the approach for students to improve the whole learning process. This determined the objectives of the focus group. Specifically, the objectives were to explore instructors' perceptions of:

1. Students' experiences with the ADRI approach in the introductory programming course;
2. The impact of the ADRI approach on students learning in introductory programming course;

and to explore:

3. instructors' views on the strengths and weaknesses of the ADRI approach compared to traditional teaching approaches;

and to suggest:

4. any further enhancements to the ADRI approach in the context of the introductory programming courses.

### *CONDUCTING THE FOCUS GROUP*

The moderator and assistant moderator welcomed the participants. At the beginning of the session, the moderator reminded participants of the purpose of the focus group and set the ground rules for participant conduct during the session, including respecting the ideas and opinions of others, listening and responding to positive and negative remarks, and reminding participants that discussions during the session should remain confidential. The moderator started by asking the participants for their feedback on students' experiences using the ADRI approach in the classroom and then for their views on the approach. The moderator encouraged all participants to participate actively and ensured that each had a chance to speak. During the session, the moderator promoted debate by asking open-ended questions. The moderator and assistant moderator took notes independently of the discussions which took place during the session. At the end of the focus group session, the moderator thanked all the participants. The moderator and assistant moderator compared their notes after the session and produced a session report. The report was sent to the participants for their feedback and approval.

## Experience in teaching introductory programming course

All the participants have a significant amount of teaching experience in introductory programming ranging from 3 to 14 years. They also have considerable teaching experience at Buraimi University College (the site of the study) of between 3 to 8 years. The participants have used C++, Java, and Visual basic languages to teach IP courses during their academic careers. Currently, the curriculum requires them to teach C++ in the IP course. They have agreed that introductory programming is a challenging course for novices in their first semester of study. They pointed out that course material, teaching styles, English language barriers, and transitioning from school to university all contribute to the challenge of learning programming and are the main factors for the high failure rates experienced

in such courses at Buraimi University College. Traditional teaching methods are not successful in overcoming the barriers for students studying introductory programming.

### First impression about ADRI based approach

The participants' first impression of learning to code was relatively unanimous. They agreed that programming appears very weird for the naïve students. It is always harder for an instructor to teach programming to students who are completely unaware of the programming world.

The focus group participants indicated that the students found it difficult to digest the overload of the ADRI approach. The participants used different methods to make students aware of the ADRI. For example, some of them first discussed the importance of all four phases of ADRI approach to the students; this method was successful to a large extent.

The participants found ADRI a more sophisticated way of injecting the programming knowledge into the students understanding. At first, it looked complex for the participants, and the implementation was perceived as a challenging task for them. Accepting the implementation challenges, the participants found it very interesting, comprehensive, and a professional methodology for teaching.

Overall the feedback from the participants concluded that, as instructors, the ADRI approach looked challenging, but later on it appeared to be one of the best teaching methodologies. However, from the students' perspective, the instructors thought it was relatively difficult to get past the initial stages, but once the students fully understood it, they kept going very smoothly.

### Strengths of the ADRI approach

The participants compared the ADRI approach with the traditional approach to summarize the strengths of ADRI approach.

The participants all agreed that ADRI is a better teaching approach than the traditional approach. The traditional approaches focus on all the phases separately, while ADRI integrates all the phases to make it easier for the students to understand the different aspects of problem, solution, testing, and improvement together. It is easier for a student to study all the phases collectively. The student understands the problem first with the help of pseudocode and flowcharts. Once the students understands the problem, they can move toward the solution of the problem in the form of developing program code. The participants agreed that the last two phases of the ADRI approach further enhance the students' level of logic understanding. The student provides the stated input to the program and then observes the output. Furthermore, it shows the process involved in solving the problem statement. It improves the student's understanding, realizing, in reality, the reasons behind writing the program. The improvement phase helps to increase students' understanding of the solved problem. It helps them in developing the logic and, in fact, helps students to understand the exact meaning and purpose of logic.

Further, regarding the structure of ADRI, the participants discussed and agreed that the ADRI approach diverts students' intellect toward program testing. The participants observed that the students tested their programs with values other than those provided in the ADRI exercises. The participants, in such cases, drew their attention to test special values in their program and then see the results. As an example, the students were asked to test the result of a program which divides an integer by another by keeping one integer value to zero. The students found themselves involved in testing, and it made it easy for the instructor to explain to them the program testing process.

The participants debated and agreed that the students stopped taking shortcuts (problem statement →coding) in programming when the ADRI was introduced to them. In the majority of cases, the students wrote one program and then modified it to solve further related problems. The participants had a mixed response as to whether the shortcut problem had indeed been solved with the ADRI approach.

Overall, the participants agreed that the ADRI approach provided a number of strong aspects and is helpful for both the students and the instructors.

### ADRI approach weakness

The focus group participants pointed out some of the major weakness of the ADRI approach. The participants also agreed that these weaknesses are just time based and evaporate gradually with the passage of time and increased understanding of the students.

The participants agreed that it was relatively difficult for the students to start with the ADRI approach. The students have the nature of being bored quickly if they are asked to solve long exercises. The same problem appeared in the adaptation of ADRI; the students felt bored when they were studying the four phases as one. However, the participants agreed that they educated the students about the relationship between the four phases. The majority of the participants agreed upon the fact that students gradually adopted the ADRI approach fully.

Similarly, the participants also observed that their students were initially struggling to handle the complexity of ADRI. Here the complexity reduced as students completed more exercises and developed familiarity with the model.

### Impact of ADRI approach on student learning

The overall impact of the ADRI approach on the students' learning appears positive. The students accepted the notion of implementing the ADRI approach in the programming course which followed the introductory IP course.

The participants discussed the impact of ADRI over different categories of students. In the programming classes, the participants had two cohorts of students: the new students who were studying the subject for the first time; and the students who had failed it earlier and were now repeating the course. Once fully understood and adopted by all the students, the fresh students felt very comfortable and impressed with the ADRI approach. Feedback was collected from the students who were repeating the course. The participants agreed that these students felt comfortable with the new approach too and appreciated the new approach.

Similarly, the student's behavior towards doing the exercises and assignments was also improved in the most recent semester. They were comfortable and interested in carrying on with their assignments and exercises. This is because they had a better understanding of the problem solution and were more confident with following the ADRI approach.

The participants all agreed that they had gained the impression that the students obtained relatively higher grades with the ADRI approach as compared to the previous semester results. This perception was later supported when the student results were analyzed.

Summarizing the general views of the participants, it was concluded that ADRI approach had a positive impact on the students learning and the course learning objectives were better achieved as compared to the previous semester. The students' problem-solving strategies, understanding power, and making logic capabilities were all improved. They were comfortable with solving a programming problem using the ADRI approach phases.

### Comparison of ADRI approach with traditional teaching approaches

The participants agreed that the traditional approach for teaching programming courses had some deficiencies. One of the major drawbacks of the traditional approach was the problem of integrating all the tools together. The students were taught pseudocode, flowcharts, and then programming in a sequential manner. In such cases, the students had to revise the previous work before going into coding. ADRI has made it simple and easy for students to transition from one phase to the next. The students can go through all the steps on a single page.

It was also a part of the debate among the participants that, unlike the traditional approach, the ADRI approach put together all the problem-solving methodologies. In other words, the ADRI approach provides a broader view of programming as compared to the traditional approach.

Most of the participants also pointed out that it was simpler to convey the improvement phase knowledge to students. They used to consider it as an extra burden. However, the smoothness in the ADRI approach has made it simpler and easier for the students.

### Suggestions for further improvement in ADRI approach

The focus group participants had a long discussion over possible improvements to the ADRI approach. The suggestions can be put into three categories: the first category includes possible improvements to the ADRI documentation, the second category speaks about syllabus changes in the institution, and the last category explains the addition of extra functionalities in the ADRI approach.

The participants discussed and then agreed that the ADRI document must be updated and an extra section of glossary should be added at the end. This glossary should explain all the terms used in the various stages, particularly around pseudo code. The glossary should be written and compiled according to the year level of the students in the institution. It should be written in plain English and be in an easy to understand format.

Some participants addressed the problem of credit hours allotted to the basic programming course. It was discussed and agreed that the credit hours must be increased so that students have more time to understand the basics of the programming language.

In the result category, the participants suggested that an extra section related to error handling should be included. The students consider programming code like normal English text. It takes a long time for them to realize that the text of programming code is extremely sensitive and the removal of a single semicolon can lead to many errors in the program for example. A teacher spends extra time in every class explaining the common errors such as missing semicolons, case sensitivity, spelling mistakes, and the impact of such errors on the overall program. The participants suggested that it would be a good idea to add an extra section in the ADRI model that addresses the common errors made by the students. Participants suggested that there should be an extra section for error-handling activities. The activity will ask the students to make a mistake, for example, "remove semi-colon at the end of line#8" or "change the word main to Main" and then record its impact on the program. The instructor's responsibility is then to explain the impact of the error, the technique of finding and fixing the error, and to make students aware of taking care to avoid similar mistakes in the future.

### Other remarks and feedback

The participants appreciated the ADRI-based editor and agreed that this has a positive impact on the students' learning. The editor provides all the necessary design features (pseudocode, flowchart) and programming tools in a single application. The students can easily navigate from phase to phase. It is easy to use, and the students do not need a lot of training or tutorials. It is also interesting to use, and the students enjoy when they are working in the editor. The menus are self-explanatory and easy to use and navigate. In the presence of this editor, the students' attitude towards programming and problem solving is more professional and practical.

One of the major properties of the editor is that it is platform-independent and can be used in any system and any operating system. It does not need complex installation guides, and the students can easily install and run it.

Focus group participants suggested a number of improvements to the editor that would assist students. The steps how to save and open a program should be included in the Help menu, facilitating users to work and interact with the editor. It will also support new users of the editor.

It was also suggested to add more supporting features in the editor, and a number of bugs were identified. For example, the Next and Previous sub-menus were not working properly under some circumstances. If you first use the Next sub-menu, then the Previous sub-menu did not work and vice versa.

Overall the participants' overview about the ADRI approach was positive, and they suggested it for further programming courses. Similarly, the instructors suggested that the students felt comfortable and satisfied with the approach and the feedback from their classes indicated that its use should be continued in future.

## IMPACT OF THE ADRI APPROACH COMPARED TO THE TRADITIONAL APPROACH ON THE STUDENTS' PERFORMANCE

In this section, we describe the second research question which is:

> *RQ2: What is the impact of the ADRI approach compared to the traditional approach on the students' performance in the introductory programming course?*

The ultimate goal of this study is to enhance the students' learning outcomes in the IP course by applying the ADRI approach. Therefore, the final grades of the course over four semesters were compared against failure and dropout rates. This criterion was used in previous studies (Guzdial & Soloway, 2002; Lahtinen et al., 2005; Sykes, 2007; Yadin, 2011; Watson & Li, 2014; Zingaro, 2015) to report the students' performance. Figure 3 shows the failure rate for the last four semesters. The semesters with the ADRI approach show better results compared to the traditional approach. The failure rates in those semesters offered with the ADRI approach were less compared to those semesters offered with the traditional approach. This trend shows the students progression in the course. The ADRI approach helps the students to gain a better understanding of the programming domain.

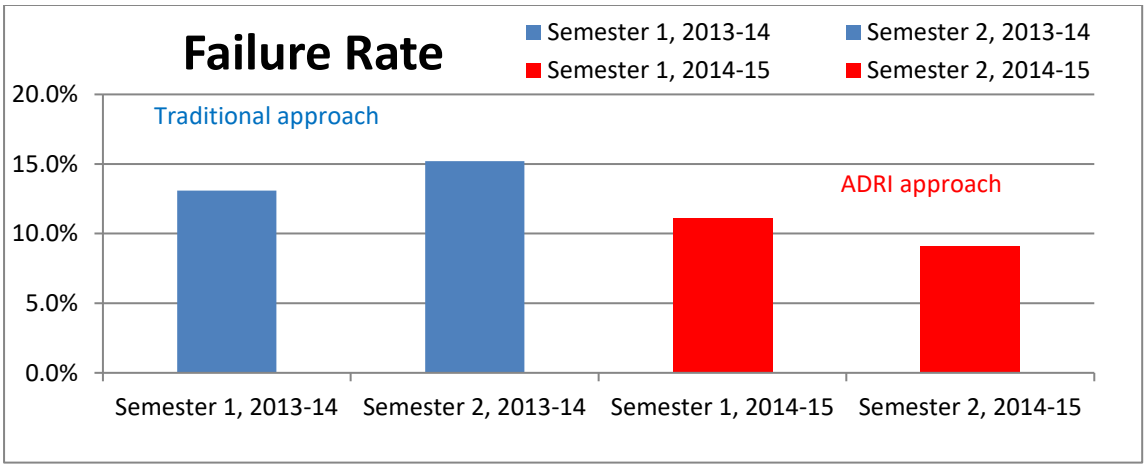



**Figure 3. Failure rates in the IP course**

Figure 4 depicts the dropout rates for the last four semesters. The ADRI approach impacted significantly compared to the traditional approach. The dropout rates in the semesters offered with the ADRI approach were very less compared to the traditional approach. This outcome has an impact on the students' engagement and enrolment in computer science discipline more generally.

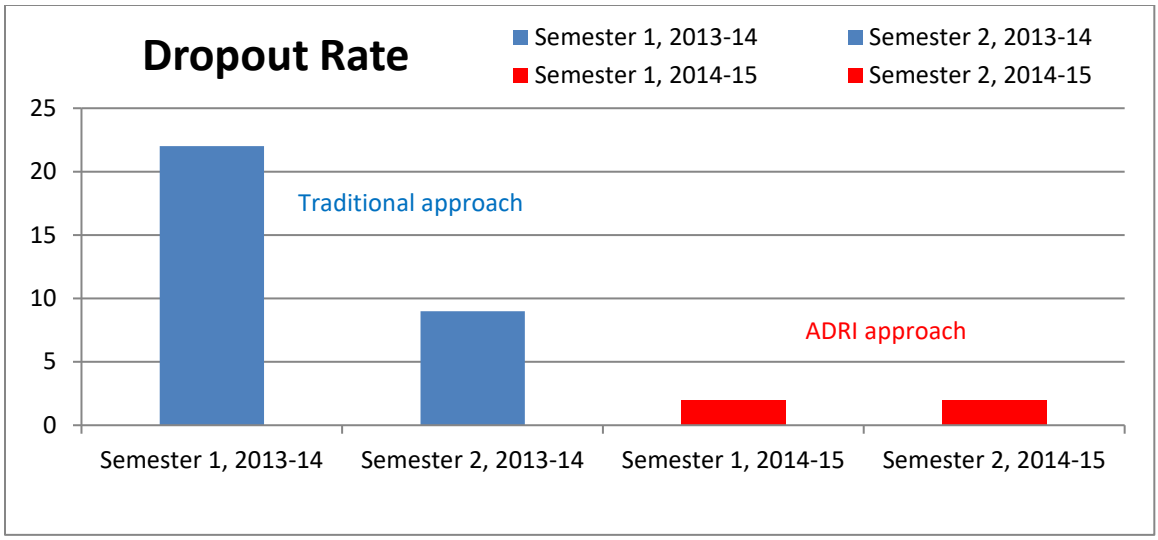



**Figure 4. Dropout rates in the introductory programming course**

Figure 5 illustrates the attrition rate (failure + dropout) for the last four semesters of the course. The ADRI approach shows better results compared to the traditional approach. It reduces the attrition rate to almost half compared to the traditional approach. The overall trend shows that the ADRI approach impacts significantly on the students learning. They feel more comfortable with the new approach which helps them in achieving their objectives. This result is consistent with previous findings of Yadin (2011) and Barak, Harward, Kocur, and Lerman (2007). Yadin (2011) carried out an action research for four semesters to address the high failure rate in an IP course. Barak et al. (2007) introduced 'Studio 1.00' that integrates lectures with in-class demonstrations, active learning sessions, and on-task feedback, through the use of wireless laptop computers' in an IP course to enhance students learning outcomes and reduced failure rates.

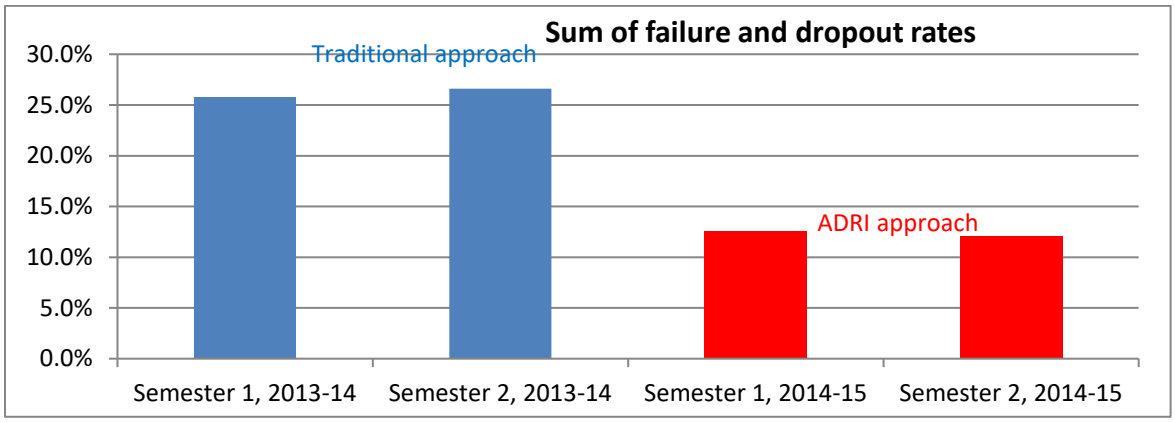



**Figure 5. Overall impact of failure and dropout rates on the introductory programming course**

It is evident from the above discussion that the ADRI approach impacts positively on the students' learning in the course. The students' grades in the semesters offered with the ADRI approach were better compared to the traditional approach.

## DISCUSSION

This study suggests that incorporating the ADRI approach in the teaching and learning process of the IP course is beneficial as the four stages of the ADRI approach provide all the basic skills (problem solving strategies and programming knowledge) required by the novices in comprehending the programming concepts. On the other hand, the traditional approach covers all these skills at different stages of their learning process separately. Consequently, it is difficult for the novices to utilize these skills together in a programming context. The ADRI approach is compatible with that suggested by Ala-Mutka (2004) that learning to program consists of several activities including "learning the language features, program design and program comprehension" (p. 3).

The instructors of the IP course pointed out that students have the nature of being bored quickly if they are asked to solve long exercises. The same problem appeared in the adaptation of ADRI; the students lost interest when they were studying the four phases as one. The graded lab sheets can be introduced to develop the students' interest and motivation for completing the longer exercises. Moreover, the contact hours per week for the IP course can be increased from 3 to 5 which will help the novices in completing the more challenging exercises. This finding is consistent with that suggested by Malik, Mathew, Chowdhury, and Coldwell-Neilson (2014).

The instructors also agreed that, in the traditional approach, it is difficult and confusing for the students to understand the connection between pseudocode, flowchart, and coding because in most of the cases it is discussed separately or chronologically. In contrast, the four stages of the ADRI approach develop a clear connection between these steps of the programming domain.

The instructors of the IP course appreciated the ADRI editor because it is simple to use, ADRI oriented, and easy to run. The students can easily access all the exercises' topics and phases. The editor encourages students to follow an appropriate programming process:

*Problem Statement* →*Problem-solving strategies* →*Programming knowledge*

instead of taking coding shortcuts:

*Problem Statement* →*Codes*

This programming process is consistent with that suggested by Dale & Weems (2010).

The ADRI approach promotes deep learning of the programming domain to the students by introducing the three-step approach (Problem Statement→Solution Plans→Codes) which develops program design, language features, and program comprehension skills. It was also reflected in their assignments and exercises in that semester. The students understood the different ways (pseudocode, flowchart, and program) to articulate the solution for the assignments and exercises questions. The programming process offered by the ADRI approach in promoting deep learning is consistent with that suggested by Jenkins (2001).

## CONCLUSIONS

A teaching approach plays an important role in the learning process of an IP course. In this study, we introduced the ADRI approach in the IP course. The teaching materials were prepared based on the four stages of the ADRI approach. The editor was developed to facilitate the learning process. The ADRI approach was compared with the traditional approach used in the IP course.

The focus group was conducted with the IP instructors to get an in-depth feedback on the ADRI approach. All the participants agreed that the ADRI approach is better than the traditional approach, suggesting that the ADRI approach four stages provides all the basic skills (problems solving strategies and programming knowledge) required by the novices in comprehending the programming concepts. They also appreciated that the ADRI approach discourages students taking programming shortcuts (problem statement →coding). They agreed that the ADRI approach encourages students

to undertake (or think about) program testing. They discussed that overall the students were satisfied with the ADRI approach. They gave some suggestions to improve the ADRI approach further.

The final grades for the students who were offered the IP course with the traditional and ADRI approaches were compared. The comparison shows that the ADRI approach gives a positive impact on the learning process. The students who finished the course offered with the ADRI approach performed better compared to the students who finished the course offered with the traditional approach.

We plan to introduce the ADRI approach in other fields of computer science studies, such as database and data structure, to determine if its impact has a wider application than just teaching introductory programming. Investigating whether using the approach impacts on higher-level programming courses is also worthy of investigation.

## REFERENCES


Ala-Mutka, K. (2004). *Problems in learning and teaching programming – A literature study for developing visualizations in the Codewitz-Minerva project*. Codewitz needs analysis. Retrieved 20 Dec, 2013 from http://www.cs.tut.fi/~edge/literature_study.pdf

Barak, M., Harward, J., Kocur, G., & Lerman S. (2007) Transforming an introductory programming course: From *Lectures to Active Learning via Wireless Laptops*. *Journal of Science Education and Technology*, *16*(4), 325-336. doi: 10.1007/s10956-007-9055-5

Carbone, A., Hurst, J., Mitchell, I., & Gunstone, D. (2009). An exploration of internal factors influencing student learning of programming. *Proceedings of the Eleventh Australasian Conference on Computing Education ACE '09*, Australian Computer Society, Australia, 25-34.

Carroll, M., & Razvi, S. (2006). *ADRI: A quality assurance model for self-reviews and external-reviews*. MOHE and OAC, Workshop handout, Traning Module 01 v6. Retrieved from http://www.oaaa.gov.om/QualityTraining/Handout/01v6_handout.pdf

Chaparro, E.A., Yuksel, A., Romero, P., & Bryant, S. (2005). Factors affecting the perceived effectiveness of pair programming in higher education. *Proceedings of 17th Workshop of the psychology of programming interest group*, Sussex University, UK, 5-18.

Dale, N., & Weems, C. (2010). *Programming and problem solving with C++*. (5th ed.). USA: Jones & Bartlett Publishers.

Guzdial, M., & Soloway, E. (2002) Log on education: Teaching the Nintendo generation to program. *Communications of the ACM*, *45*(4), 17-21.

Huet, I., Pacheco, O.R., Tavares, J., & Weir, G. (2004). New challenges in teaching introductory programming courses: A case study. *Proceedings of 34th ASEE/IEEE Frontiers in Education Conference, IEEE*, Savannah, GA.

Iqbal, S., & Harsh, O.K. (2013). A self-review and external review model for teaching and assessing novice programmers. *International Journal of Information and Education Technology*, *3*(2), 120-123.

Jenkins, T. (2001). Teaching programming – A journey from teacher to motivator. *Proceedings of 2nd Annual LTSN-ICS Conference*, London, 65-71.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14-18.

Malik, I. S. (2016). Enhancing practice and achievement in an introductory programming course using an ADRI editor. *Proceedings of the IEEE International Conference on Teaching, Assessment and Learning for Engineering, IEEE,* Thailand, 7-9 December, pp. 32-39. doi: 10.1109/TALE.2016.7851766

Malik, I. S., & Coldwell-Neilson, J. (2016). A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*. doi: 10.1007/s10639-016-9474-0

Malik, I. S., & Coldwell-Neilson, J. (2017). Impact of a new teaching and learning approach in an introductory programming course. *Journal of Educational Computing Research*. SAGE. doi: 10.1177/0735633116685852

Malik, I. S., Mathew, R., Chowdhury, M. U., & Coldwell-Neilson, J. (2014). Impact of assurance of learning (AOL) in programming course for novices. *Proceedings of the 2nd BCS International IT Conference*, UAE. doi: 10.14236/ewic/bcsme2014.17

Matthiasdottir, A. (2006). How to teach programming languages to novice students? Lecturing or not? *Proceedings of International Conference on Computer Systems and Technologies – CompSysTech'06,* University of Veliko Tarnovo, Bulgaria, 13-1 –13-7

Merrick, K. E. (2010). An empirical evaluation of puzzle-based learning as an interest approach for teaching introductory computer science. *IEEE Transactions on Education*, *53*(4), 677-680.

Miljanovic, M. A. (2015). RoboBUG: A game based approach to learning debugging techniques. Master thesis, accessed on October, 07, 2015, from https://ir.library.dc-uoit.ca/handle/10155/536

Moen, R., & Norman, C. (2010) *Evolution of the PDCA Cycle*. Retrieved 05 July, 2013 from http://www.uoc.cw/financesite/images/stories/NA01_Moen_Norman_fullpaper.pdf

Mohorovicic, S., & Strcic, V. (2011). An overview of computer programming teaching methods. *Proceedings of Central European Conference on Information and Intelligent Systems, CECIIS*, Croatia, 47-52

Nuutila, E., Törmä, S., &Malmi, L. (2005). PBL and computer programming — The seven steps method with adaptations. *Computer Science Education*, *15*(2), 123-142.

Papp-Varga, Z., Szlávi, P., & Zsakó, L. (2008). ICT teaching methods – Programming languages. *Annales Mathematicae et Informaticae*, *35*, 163–172.

Razvi, S., Trevor-Roper, S., Goodliffe, T., Al-Habsi, F., & Al-Rawahi, A. (2012). Evolution of OAAA strategic planning: Using ADRI as an analytical tool to review its activities and strategic planning. *Proceedings of Seventh Annual International Conference on Strategic Planning for Quality Assurance and Accreditation of Universities and Educational Arab Institutions*, Cairo, Egypt.

Sung, K., Hillyard, C., Angotti, R. L., Panitz, M. W., Goldstein, D. S., & Nordlinger, J. (2011). Game-themed programming assignment modules: A pathway for gradual integration of gaming context into existing introductory programming courses. *IEEE Transactions on Education*, *54*(3), 416-427.

Sykes, E. R. (2007). Determining the effectiveness of the 3D Alice programming environment at the computer science I level. *Journal of Educational Computing Research*, *36*(2), 223-244.

Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. *Proceedings of ITiCSE*, ACM, Uppsala, Sweden

Yadin, A. (2011). Reducing the dropout rates in an introductory programming course. *ACM Inroads*, *2*(4), 71-76.

Yoneyama, Y., Matsushitab, K., Mackinb, K. J., Ohshirob, M., Yamasakib, K., & Nunohiro, E. (2008). Puzzle based programming learning support system with learning history management. *Proceedings of the 16th International Conference on Computer in Education*, Taipei, Taiwan, 623-627.

Zacharis, N. Z. (2011). Measuring the effects of virtual pair programming in an introductory programming Java course. *IEEE Transaction on Education*, *54*(1), 168-170

Zingaro, D. (2015). Examing interest and grades in computer science 1: A study of pedagogy and achievement goals. *ACM Transactions of Computer Education*, *15*(3), Article 14.

## BIOGRAPHIES

**Sohail Iqbal Malik** obtained his Ph.D. in Computer Science from Deakin University, Australia in 2016. He has been working as an Assistant Professor at Information Technology Department, Buraimi University College, Oman (Academic collaboration with California State University, Northridge, USA) since February 2007. His research interest includes Computer Education, Technology in Education, Knowledge Management and Mobile Learning.

After 13 years in the ICT industry, **Jo Coldwell-Neilson** has been and academic for over 30 years. Jo has built a strong research and teaching profile engaging students in and with technology. She has been involved in many projects implementing educational technologies in her teaching and the learning and teaching activities at Faculty and University level. Her current research reflects interests, including investigations into gender issues in IT, digital technology uptake in schools and higher education, and preparing students for careers in a digital environment. Jo is a current Australian Learning and Teaching Fellow.