Integration of ICT in Concurrent and Parallel Programming Lectures

Antonio J. Tomeu-Hardasmal[1], Alberto G. Salguero[1(✉)], and Manuel I. Capel[2]

[1] University of Cádiz, Cádiz, Spain
alberto.salguero@uca.es
[2] University of Granada, Granada, Spain

Abstract. An effective teaching and learning in concurrent and parallel programming needs the presentation of short excerpts of code to students in a selected programming language during lectures. This is sometimes necessary to make understandable complicate syntactical constructs. Traditionally, these codes have been presented by the teacher to the students on a blackboard, with slides or by means any projection facilities, together with an oral description of their operation and significance. Teachers try to explain those syntactical constructs with more or less success and students do not usually test program code during lectures, nor can do any modifications to the programs, which are helpful in order to master difficult concepts of parallel programming and Concurrency. In the best possible scenario, students will carry out any tests on the real code of an example presented in lectures during practical lessons at the lab. Nevertheless, without a clear illustration of their actual behavior in a real computer program, any new concepts of concurrent constructs taught in lectures will remain fuzzy and prone to be forgotten. The presented approach consists of changing the traditional teaching and learning model of parallel programming into another where students will be equipped with multicore processors inside their laptops, which in addition to Virtual Campus services will serve to put into practice any programming code that illustrates a programming concept the minute it is presented during any lecture by the teacher to the class.

Keywords: Teaching innovation · Teaching improvement · Virtual campus · ICT integration · Lecturing model · Concurrent programming · Parallel programming · Code · Performance improvement · Interactive theoretical teaching · Students

1 Introduction

Any computer that we purchase nowadays is equipped with a multicore processor that allows the programmer to carry out real concurrent computations or implement "parallelism" in a program if we prefer to call it in this way. Present curricula in Computational Science and Engineering (CSE) have addressed this

S. Hunold et al. (Eds.): Euro-Par 2015 Workshops, LNCS 9523, pp. 114–124, 2015.
DOI: 10.1007/978-3-319-27308-2_10

new reality, and have paid attention to the importance for future graduates of excellence training, which must include mastering well-known Concurrent and Parallel Programming techniques that would enable them to exploit the parallel potential, in terms of speedup, that current multicores are offering nowadays.

In the past, concurrent and parallel programming topics were part of elective courses with a focus on computer architecture or concurrency. Now, they are compulsory subjects to be taken by students in the core CSE curriculum. Nevertheless, this change has not come along with a new way to teach concurrency nor programming, in general [1–10]. In spite of the new teaching models that European Higher Education Space (EHES) promotes, to teach Programming to undergraduate students still needs presenting and discussing pieces of code to the students. This learning model is not likely to be modified in the future. What teachers can modify is the way that students face a concurrent or parallel programming problem.

We propose the students to adopt a protagonist role - instead of the traditional passive role. We present in this paper the results of a teaching innovation project that has been carried out at the University of Cádiz. Thereof, we discuss how to make that didactical change can happen in concurrent and parallel programming courses, and analyze the results obtained from our new teaching model applied to a real study case.

1.1 Environment

The following items describe the conceptual framework within which the experience described here has been developed:

- The mentioned experience has taken place during the past academic year 2013/2014 in a course that has been taught to an audience of 149 enrolled students divided in two classes of 75 students each one for lectures. This course is taken in the first semester of the sophomore year of the CSE degree, to earn 3 credits of lectures and 3 more credits of practical assignments if students are successful.
- This teaching innovation project has been developed and followed by the two tenured teachers assigned to this course, who belong to the same Department but to three different areas of knowledge, Computer Science (CS), Artificial Intelligence (AI) and Software Engineering (SE).
- In order to provide the necessary infrastructure to carry out this new way of giving lectures, a classroom full equipped with laptops (to be used on a deposit–basis) was solicited to the College of Engineering Director, as well as tables with power sockets available to plug in the student owned laptops. The classroom had access to Campus WIFI. A beamer and whiteboard were also available together with a "regular multimedia table" as the ones normally used for traditional teaching.
- The course's lectures run according to the temporal schedule planned by the College beforehand, through a series of theoretical lecture sessions of 2 h each one per week, according to what had been planned by the Vicedean for Academic Affairs.

1.2 Objectives

The objectives to be attained by the experience described here were the following ones:

- Improvement of theoretical content lectures, in particular those on Concurrent and Paralell Programming, and lectures on general programming techniques.
- To mutate the prior passive character of students during lectures into a much more active one.
- To attain the practical integration of Information and communications technology (ICT) in the classical teaching of theoretical contents. We argue that the only use of multimedia, such as the beamer or the whiteboard, do not change the passive nature of students during the presentations carried out by the teacher.
- The work carried out interactively by the class during lectures becomes highly increased.

1.3 Time Schedule

The experience has been developed according to the time schedule shown in Fig. 1, and took place from June 2013 to June 2014. All the semester's theoretical lessons were given according to the methodology which is to be introduced in the following sections. These lectures were given from October 2013 to February 2014.

Previously to carry out the above activities, a preparatory work was necessary in order to:

- Write, debug and test the set of codes that will support the entire experience.
- Configure a virtual platform, fully supported by Moodle, which contains files with the selected set of programming codes, together with the rest of documents that support the course, i.e., readings, slides, examination samples, user manuals, study recommendations, etc.

In the last week of teaching, the final impressions about the new model were collected by using a survey that was previously handed out to the students. Afterwards, the final examinations were taken. Finally, by taking into account the grades that the students obtained in the exams and survey results, we were able to produce the results and conclusions shown in sections three and four of this paper.

2 What Has Been Innovated?

The computers are rarely used as learning tools. Teachers mainly use computers as delivery tools to present instructional content or to captive students by the use of computer-assisted learning applications such as drill and practice, tutorials, and simulations [11]. The teaching of general programming techniques and

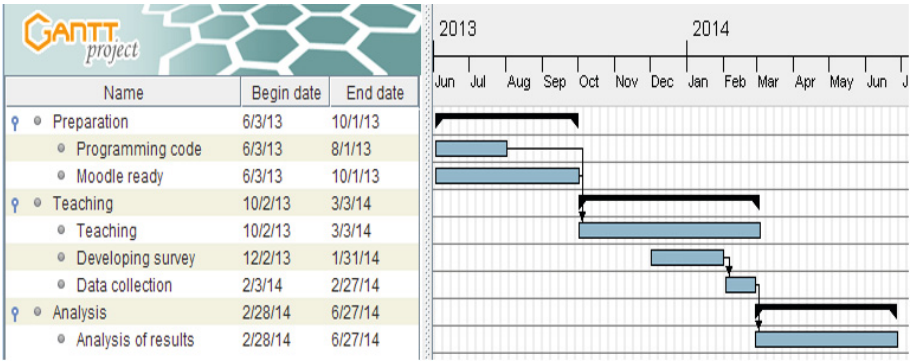

Fig. 1. Project time schedule

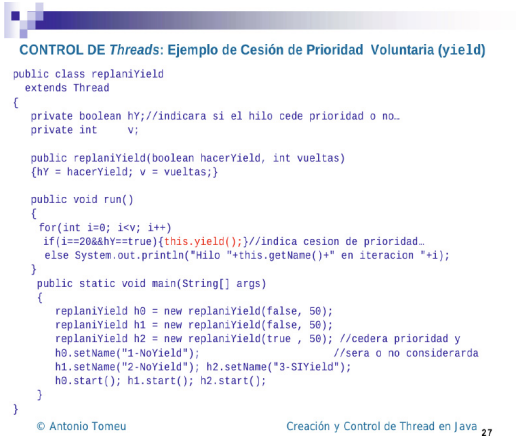

Fig. 2. Slide case example

particularly Concurrent Programming have been traditionally conducted until now by discussing pieces of code from computer programs, whether these are written on the blackboard, or shown with a slide projector (in Fig. 2, a case example of one slide, similar to the ones that we used to show to the students, in theoretical lessons, with a beamer before conducting this experience). Usually, teachers explain any relevant concepts and operations on these slides before going through and... immediately pass on to showing and analyzing the next slide with another program code. Thereby, students usually ended up with a partial view of the semantics of every syntactical construction explained in that way, since they could not immediately test the proposed code by running or debugging it.

The passive role of the students during a lecture was in the best possible way only when they could interrupt and ask the teacher to clarify a specific code in the program being discussed.

Fig. 3. A new model for theoretical teaching

The proposal presented here is a radical rethinking of the student role during a theoretical lesson on Programming. In order to make this new approach feasible, the student must have a laptop or desktop computer with access to "Virtual Campus" (see Fig. 3), as well as one Campus–WIFI driven teaching platform, thoroughly available during the entire session. Within "Virtual Campus", teachers of the different courses have designed material as well as the necessary programs to support the new model. Students can therefore have access to the code on a slide that the Professor is discussing during the lesson, so that they can observe any effects due to download, debug, run and test performed on that code.

In order to support the new teaching model, each "Virtual Campus" block, which correspond to a specific course–subject content, will include a folder with a relevant example of program code, as the one showed in Fig. 4. After downloading that code, the student will start developing the new cycle of his/her assignment as it is proposed during the corresponding theoretical teaching lesson. The sequence of tasks to be done to complete that assignment is shown in Fig. 6. As it can be seen there, the students finally change their behavior from being a "passive information receiver", which is mainly transmitted by their teacher discussing a program code, to become the absolute protagonis of the learning process now. The students have to perform the required empirical analysis of downloaded code from "Virtual Campus", as well as to carry out debugging, execution and analysis of outputs that this code may yield.

On finishing the above stage in the new theoretical teaching process model, if the class has mastered the construct at hand in the code example case, then a series of conceptual reinforcement exercises are proposed, which must be carried out individually. Then, result codes of these exercises are sent to an specific purpose "Virtual Campus" forum. Finally, possible solutions to the exercise are presented to the class and discussed in common to find the best one.

Fig. 4. A subject block in "Virtual Campus" platform

2.1 Development Methodology

During the elaboration phase of didactic contents, which are necessary for implementation of the innovation project, the methodological work guideline followed by the course teachers has gone through the following stages:

- For each course's subject block, teachers have developed a set of program codes that illustrate the theoretical concepts which are the selected reinforcement objectives.
- The proposed codes samples, after being cross-checked between the involved teachers, have been available to students the week before teaching them. On a weekly basis, the codes will be available to students inside a specific folder of "Virtual Campus" repository, as we have already mentioned (see Figs. 4 and 5).
  Two versions of each code have been pre-arranged, one is composed of plain text files, and the other one contains a zip file with the set of files for an easier download.
- During the theoretical lesson, and once the teacher has presented the example code case and discussed its functionality, the student is asked to follow the work guideline depicted in the flow diagram in Fig. 6.

3 Results

In order to carry out the experiment's results analysis, it has been chosen a prospective double stage, then a set of analysis relevant actions have been conducted, previously and posteriorly to the curse assessment:

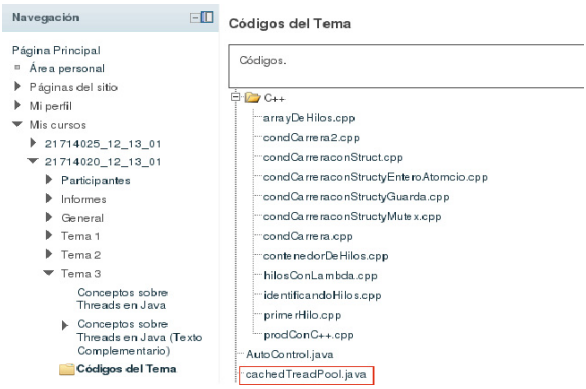

Fig. 5. The "Code" folder

3.1 Pre-assessment

After three complete months of lectures and in order to meet any subjective impression that the new theoretical teaching model was showing among the enrolled students to which it was applied, we also designed a brief survey, aimed at quantitatively measuring the following variables:

- Understanding improvement of concepts presented during theoretical teaching lessons.
- Adjustment of proposed exercises number to reinforce important concepts presented in lessons.
- Adjustment of time devoted to proposed exercises resolution.
- Compliance level with the new model of theoretic teaching.

Each one of the above items could be assessed with a score from one (totally disagree) to 5 (completely agree) and including an additional sixth item to give response to the case in which the student does not want to answer. The following bar charts[1] shows the results. The sampling size was fixed to n =78.

Histogram of Fig. 7 contains the opinion of students regarding the comprehension improvement of the programming concepts, which were introduced, during the theoretical teaching lessons, by using the new model. We see how almost all of the students agree or strongly agree to the benefit that they obtained with the new model compared to the classical passive teaching theoretical scheme.

Histogram of Fig. 8 contains students views regarding the number of exercises proposed during the phase of exercises shown in diagram of Fig. 6. In particular, to know whether the number of exercises is appropriate to reinforce a concept comprehension after its presentation in a lecture. Again, the vast majority of

[1] The histogram legend represents absence of response as 0, and classifying the answers according to the already suggested range: 1 for "disagrees" up to 5 for "completely agrees".

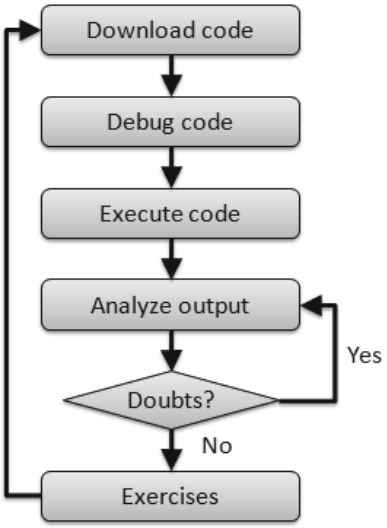

Fig. 6. New model's work cycle

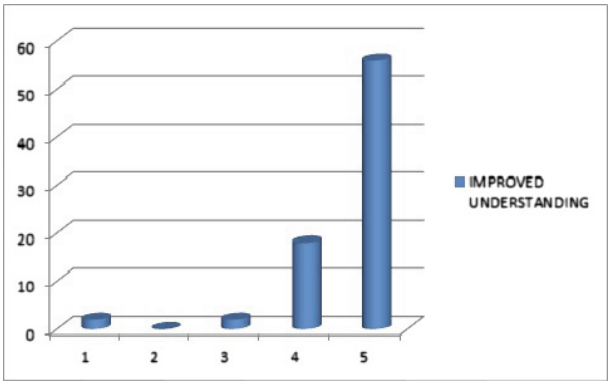

Fig. 7. Understanding amendment

students considered to be so, although this time, we can observe data somewhat more dispersed. A small group of students considered the number of exercises inappropriate, although we can not say if it was by excess or by defect.

Histogram of Fig. 9 shows students opinions about the amount time they were given to solve the proposed exercises. We can appreciate here that approximately one-third of students considered that time to be inappropriate. Again, we do not know whether it was by excess or by defect. We will need to carry out further research on this aspect to take possible actions for attaining an improvement. Histogram of Fig. 10 gathers students opinions regarding their level compliance with respect to the new model of theoretic teaching experienced. Again, the

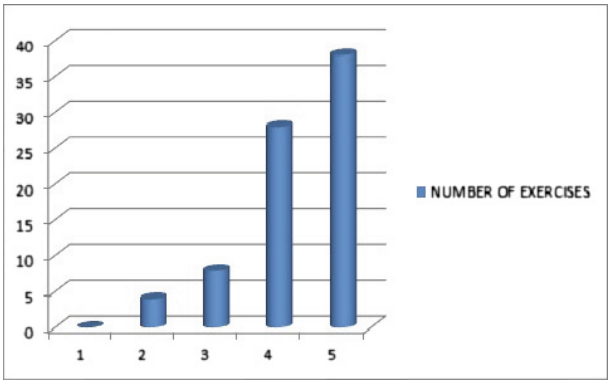

Fig. 8. Exercises number adjustment

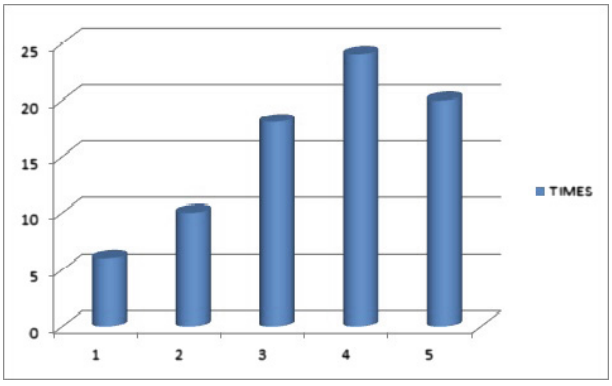

Fig. 9. Exercise resolution time assessment

vast majority of students chose between "agree" or "strongly agree" with it. Additionally, we can notice that with respect to the two items of greatest interest in order to validate the feasibility of the new model(understanding improvement of theoretical contents and agreement level with the new model), the results obtained during the experience were very good in general.

3.2 Post-assessment

In this case, we conducted a comparative analysis of the performance obtained between the academic results of year 2014/2015 and the academic year 2012/13. Thereof, we tackle with an objective measurement of the new model of theoretical teaching goodness degree w.r.t the classical teaching method.

To perform this analysis, we compared the number of students succeeding the course against those who failed. The results of the analysis showed that if during the academic year 2012/2013 only succeeded 35 % of the students, during

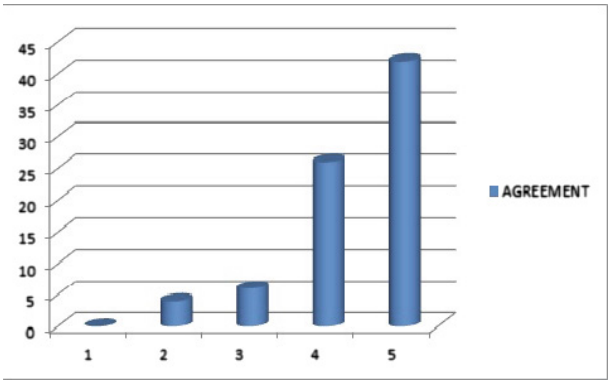

Fig. 10. Agreement level with the model

the academic year 2014/15, by using the proposed methodology in this paper, this percentage showed an improvement of 20.6 % points, then reaching the rate of 55.6 % of students that have succeeded. All these figures refer to the first examination taken by the students.

4 Conclusions and Future Work

After conducting the analysis of the experiment results, which has been carried in Sect. 3.2, and due to the excellent results obtained during the phases of screening and post-assessment, we have come to the conclusion to keep the new model of theoretic teaching in the academic years to come,

- The aimed project objectives have been attained.
- The understanding of programming concepts presented to students in theoretical lectures has been improved.
- We have migrated from a reality where the student behaves passively during theoretical lessons into another in which he or she has to have a more proactive attitude.
- Indirectly, we have achieved a favorable change in students customs, since they now understand the material taught in lectures on a daily basis, thus propitiating a more continuous involvement with the course contents presented in the series of lectures. Since, a good learning cannot be acquired by only attending to listen the teacher's lesson and - eventually - taking some notes. Students need a updated domain of the contents, in order to make an effective use of the new model of theoretical teaching.
- Additionally, in an indirect way, the student becomes more prepared to deal with practical material given during the course; indeed, the new approach has allowed us to increase the workload during practical sessions in labs, as well as increasing the complexity of this material, thus improving the education of students.

- Students are now involved in a more personal and rigorous way with the daily work and course assignments.
- ICT deployment and their effective use have been radically integrated with theoretical teaching of Programming, according to a more effective and less traumatic way of work for the student in order to get success with that course.
- The success rate of students in the course has remarkably improved w.r.t. the previous course, where the traditional model was followed.
- And finally, we should bear it in mind that students tell us that they "like your theoretical lessons as they are given now".

Acknowledgement. We would thank to the Academic Affairs Vicedean of the College of Engineering, who provided us with facilities to put in place the theoretical teaching, which has been the innovation teaching objective in the classroom carried out in this project.

References

1. Area Moreira, M.: Enseñar y aprender con TIC: más allá de las viejas pedagogías. Aprender a educar con tecnología, n2, pp. 4–7, diciembre 2012
2. Area Moreira, M.: Una breve historia de las políticas de incorporación de las tecnologías digitales al sistema escolar en España. Quaderns digitals: Revista de Nuevas Tecnologías y Sociedad, N. 51 (2008). ISSN-e 1575–9393
3. Ben-Ari, M.: A suite of tools for teaching concurrency. In: Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2004, Leeds, UK, 28–30 June 2004
4. Bloom, B.S., et al.: Taxonomy of Educational Objetives: Handbook I, Cognitive Domain. David McKay, New York (1956)
5. Carro, M., Herranz, A., Mario, J.: A model-driven approach to teaching concurrency. ACM Trans. Comput. Educ. **13**(1), 48–67 (2013)
6. Ko, Y., Burgstaller, B. Scholz B.: Parallel from the beginning: the case for multicore programming in the computer science undergraduate curriculum. In: Proceedings of the 44th ACM Technical Symposium on Computer Science Education (2013)
7. Marowka, A., Shenkar, R.: Think parallel: teaching parallel programming today. IEEE Distrib. Syst. Online **9**(8), 1–8 (2008)
8. Paprzycki, M.: Integrating parallel and distributed computing in computer science curricula. IEEE Distrib. Syst. Online **7**(2), 43–49 (2006)
9. Schechter, E.I.: Internet resources for higher education outcomes assessment. JFECS **8**(2), 105–107 (2009)
10. Vilela, A.: Moodle 2 Para Profesores. Ed Rom (2009)
11. Inan, F.A., Lowther, D.L., Ross, S.M., Strahl, D.: Pattern of classroom activities during students' use of computers: relations between instructional strategies and computer applications. Teach. Teach. Educ. **26**(3), 540–546 (2010)