SIMCAN: A Simulator to Improve Learning of Distributed and HPC Systems in Engineering Degrees

Alberto Núñez[1*], Carolina Mañoso[2], Ángel P. de Madrid[2] and Simon Pickin[1]

[1] Software Systems and Computation department, Complutense University of Madrid, Spain

[2] Control and Communication Systems Department, Universidad Nacional de Educación a Distancia – UNED, Spain

***Correspondence**: Alberto Núñez, Software Systems and Computation department, Computer Science Faculty, Complutense University of Madrid, Spain.
E-mail: alberto.nunez@pdi.ucm.es. Phone: +34 91 394 7531.

ABSTRACT

Distributed Systems Programming (DSP) is an important subject in the Computer Engineering undergraduate degree. The use of a version of the SIMCAM simulator adapted to the educational context enabled our DSP students to exercise important facets of DSP that are otherwise difficult or impossible to incorporate in student activities. Analysing and quantifying the assumed benefits of this educational intervention on student learning enables us to better adapt such interventions to student needs. To investigate the impact on student learning of this novel use of a simulator we analyse both the course-assessment results and the constructors of the Technology Acceptance Model (TAM), the latter via an initial survey of student perceptions carried out at the beginning of the course and another carried out after completing the simulator-based assignment. We observe, in particular, an improvement in the overall grades between the target year and those of the year previous to the simulator introduction. Moreover, other statistical findings are also of interest.

Keywords: Educational software, European Higher Education Area (EHEA), performance, simulation, student satisfaction, survey.

Authorship

Alberto Núñez and Simon Pickin conceived the idea of this work. Alberto Núñez developed the SIMCAN simulator and designed the SIMCAN-based assignment of the DSP course (Assignment3). Carolina Mañoso and Ángel P. de Madrid processed the collected data and performed the computations.

All authors discussed the results, have been involved in writing the manuscript and participated in the required revisions. All authors agree on the order in which their names will be listed in the manuscript.

Acknowledgments

This work has been supported by the Spanish MINECO-FEDER (grant numbers DArDOS, TIN2015-65845-C3-1-R and FAME, RTI2018-093608-B-C31) and the Region of Madrid (grant number FORTE-CM, S2018/TCS-4314).

Compliance with Ethical Standards

Conflict of Interest: Alberto Núñez declares that he has no conflict of interest. Carolina Mañoso declares that she has no conflict of interest. Ángel P. de Madrid declares that he has no conflict of interest. Simon Pickin declares that he has no conflict of interest.

SIMCAN: A Simulator to Improve Learning of Distributed and HPC Systems in Engineering Degrees

Abstract— Distributed Systems Programming (DSP) is an important subject in the Computer Engineering undergraduate degree. The use of a version of the SIMCAM simulator adapted to the educational context enabled our DSP students to exercise important facets of DSP that are otherwise difficult or impossible to incorporate in student activities. Analysing and quantifying the assumed benefits of this educational intervention on student learning enables us to better adapt such interventions to student needs. To investigate the impact on student learning of this novel use of a simulator we analyse both the course-assessment results and the constructors of the Technology Acceptance Model (TAM), the latter via an initial survey of student perceptions carried out at the beginning of the course and another carried out after completing the simulator-based assignment. We observe, in particular, an improvement in the overall grades between the target year and those of the year previous to the simulator introduction. Moreover, other statistical findings are also of interest.

Index Terms— Educational software, European Higher Education Area (EHEA), simulation, student satisfaction, survey.

1 Introduction

Distributed and High Performance Computing (HPC) Systems play a fundamental role in the Information and Communication Technology –ICT– sector, in particular in the largest ICT companies such as Google, Amazon, Facebook, Whatsapp and LinkedIn. However, companies and organisations large and small, from all sectors of the economy, now also make extensive use of many types of distributed systems, particularly distributed data centres. Furthermore, distributed systems have now come to form part of the daily life of much of the world's population via social networks, online multiplayer games or cloud storage. The explosive growth in the use of distributed systems in society seen in the last decade is also leading, albeit slowly, to a growth in the prominence of this topic in undergraduate computer science, computer engineering and information science degrees. This increasing importance was reflected in the creation of a new "knowledge area" called "Parallel and Distributed Systems" in the penultimate edition of the ACM Computer Science Curricula Recommendations [1]. These joint ACM/IEEE recommendations are widely used in curricula preparation by universities throughout the world. The authors of [1] refer to "the vastly increased importance of parallel and distributed computing" stating that "parallel and distributed computing has moved from a largely elective topic to become more of a core component of undergraduate computing curricula". The above-mentioned knowledge area no longer appears in the latest edition of the recommendations, due to the number of knowledge areas used being reduced from 18 to 12, but it is explicitly present in several of the remaining 12.

The central role of student centred learning (SCL) in the Bologna process and the European Higher Education Area (EHEA) was highlighted in the ministerial declaration of Leuven in 2009 and reiterated in the subsequent declarations of Bucharest in 2012 and Yerevan in 2015. One of the most salient aspects of SCL, in the opinion of the higher-education student and staff unions surveyed by Geven and Santa [2], is "activity-based learning". It is also stated in this document that SCL "requires a paradigm shift […] to a more interactive and practical approach to teaching and learning." In computer science and computer engineering, computing labs play an important part in implementing a practical approach and in the use of activity-based learning. However, undergraduate-level teaching of distributed-systems can run into certain difficulties with regard to helping the students put their knowledge into practice in university computing labs.

Firstly, the activities that the lecturer wishes to carry out in the lab may come into conflict with the institution's computer-infrastructure security policy. It is common, for example, for severe restrictions to be placed on students' use of network communications, such as user login on multiple nodes, peer-to-peer applications or mobile code. This may result in students being obliged to execute the different components of the distributed – or parallel - applications they develop on a single physical node. Even in the case where logical nodes / virtual machines are used, this restriction has serious drawbacks. For example, it hinders any study of performance considerations and thereby also of those design considerations that are influenced by, or can affect, performance. Experience also shows that students who are forced to develop a distributed application on a single physical node are wont to create applications that do not, in fact, execute correctly *unless* executed on a single physical node [3], [4]. This can be for reasons as simple as inadvertently using "localhost" for the domain name. Finally, if students are not able to experience, or visualise in some way, the distributed execution of the application they are building, they may suffer a loss of motivation and have difficulty believing that their application can indeed work in a distributed fashion.

Secondly, in an era when large-scale distributed systems are becoming increasingly common, distributed and HPC systems courses should also deal with the main aspects of such systems. However, though some of these aspects can be

exercised on low-cost solutions such as Raspberry Pi clusters, in general, it is unrealistic to expect higher education institutions to have sufficient resources to acquire and administer the infrastructure on which students can develop large-scale distributed applications. The costs of a distributed infrastructure for use by students could be shared by groups of educational institutions, similar to the way in which test beds such as PlanetLab [5] have been jointly built by research institutes. However, such a solution is costly, in terms of finance, organisation and administration, which is one of the reasons why, to our knowledge, such common infrastructures do not currently exist. In this article, we discuss a more realistic alternative solution, namely simulation.

The use of virtual environments [6] and simulation in education is not new. The survey article [7] shows simulation to be a valuable tool in science education in general. Recent work that uses simulators for teaching distributed and HPC systems focuses on postgraduate students [3], [4] and, to the best of our knowledge, there is no study that uses simulators for teaching distributed and HPC system to undergraduate students. The advantages of using simulation in tertiary computer science education, in particular, to mitigate the absence of costly infrastructure, is discussed by Alnoukari et al. in [8], the context being the Syrian Virtual University, where the provision of educational infrastructure is currently challenging, to say the least. Other challenges to the successful use of simulation to teach distributed and HPC systems in Engineering degrees are: managing the student's learning curve for the new tool, overcoming difficulties students have in applying previously-acquired knowledge of distributed and HPC systems and architectures to simulation, and helping them to interpret simulation results, in particular, to check the correctness of the systems being evaluated.

The objective of the work presented in this article is two-fold: first, to alleviate the existing challenges for teaching distributed and HPC systems to undergraduate students. A detailed description of how we deal with the previously-described challenges is provided in Section 2. Second, to study the value of simulation in the learning of distributed systems programming using the SIMCAN simulator [9]. To this end, we analyse the results obtained using two different metrics: the students' final mark and the students' perceptions. The users' attitude towards a system is a major determinant of whether the user will actually use or reject it and this attitude originates in their perceptions. Venkatesh et al., in [10] review user acceptance literature and discuss eight prominent models to predicting a person's behaviour based, most of them, on the perceptions.

In particular, to measure the students' perceptions we adapt the Technology Acceptance Model (TAM) to our purposes. TAM, introduced by Davis ([11], [12]) and based on the theory of reasoned action [13], is a powerful tool for predicting the acceptance, adoption and actual use of new technologies. It makes use of three so-called "constructors", namely *perceived usefulness*, *perceived ease of use*, and *behavioral intention to use*, as major elements in the decision to use a given technological element (Fig. 1, continuous line). Perceived usefulness is the degree to which users think that using the technology will increase their performance, effectiveness and efficiency. Perceived ease-of-use is defined as the degree to which users expect the use of the technology to help them optimise their efforts. Finally, behavioural intention of use is defined as the degree to which users have formulated plans to use the technology in the future.
[Insert Figure1]

The TAM model has received extensive empirical support [14] and has been validated in meta-analyses involving dozens of studies [15], [16]. It has been used to study the future use of a wide variety of new innovations in information technology, in particular, in educational environments [17]–[20]. In many of these studies, the TAM model is modified to incorporate new variables or to eliminate others, looking for other elements that influence the acceptance of the technology. For instance, Liu et al. include variables such as the design of the tool or technology and the user's previous experience [21].

2 SIMCAN SIMULATION PLATFORM

Simulations of highly distributed systems, which usually involve a very large number of heterogeneous resources, require precise, efficient and flexible modelling techniques. It is crucial that the system under study be precisely modelled in order to accurately represent its behaviour. Hence, simulation tools must provide enough flexibility to model a wide-range of system configurations. The SIMCAN simulator fulfils these requirements. SIMCAN [9] is a simulation platform for modelling highly distributed systems and applications, which is currently available, e.g. from http://www.simcansimulator.com, as open source software.

The SIMCAN simulator has been developed and validated by one of the authors of the present study [9], [22]. This simulator has three main objectives. The first is to provide a high level of flexibility and scalability, allowing users to model a wide range of highly-distributed system configurations. The second is to ease the development of distributed applications by providing intuitive APIs. The third is to investigate the performance of distributed and HPC applications executed over distributed systems with different architectural configurations.

SIMCAN is written in C++ using OMNeT++ [23], a discrete event simulation (DES) framework that has become very popular in the research community in recent years, and INET [24], a framework to simulate communication networks using real protocols, like TCP and UDP. It is important to remark that many other well-known simulators for modelling distributed systems, such as OverSim [25] and RINAsim [26], are also based on OMNeT++, which has been

used in more than 2,000 scientific publications in the last five years (based on Google Scholar search results).

Since this new version of SIMCAN has been designed to analyse the tendency of the system when different architectural configurations are provided, the execution of a simulated environment is deterministic, that is, executing the same scenario different times always produces the same result. However, it is possible to include new stochastic modules, e.g. for modelling file systems and disk drives, which use random variables [27].

The simulation core of SIMCAN relies on its repository. Basically, this repository contains a collection of models that represents the most relevant components of a distributed system, such as CPUs, disks and communication networks. These models are hierarchically classified into four basic systems: storage, CPU, memory and network. Using this structure, users are able to model large distributed systems, like HPC clusters and data-centres for supporting cloud computing environments [28], [29]. In this case, several computer models are interconnected through a communication network. Each computer model contains an API module that connects the applications with the four basic systems. Thus, user applications are able to request hardware resources by invoking the functions provided by the API module. SIMCAN provides different APIs to develop distributed applications, which are inspired by real APIs like POSIX and MPI [30], [31].

In the last few years, several simulation tools have been used in education for teaching distributed systems and analyzing performance of HPC systems. Zarza et al. [32] propose an innovative teaching strategy for teaching performance evaluation of HPC systems, in which the OPNET Modeler [33] (now known as Riverbed Modeler) is used to implement models of HPC systems, these models being provided to the students, with a view to easing the assimilation of HPC concepts. This approach requires real traces to be replayed in a simulation environment. Moreover, an MPI API is not provided and, consequently, MPI applications cannot be executed in the simulator. Finally, the OPNET simulator is licensed software, which hampers its use in the educational setting.

Another interesting approach is the one presented by Degomme et al. [3]. In this work, the simGrid simulator [34], [35] is used to simulate and analyse the performance of MPI applications. This approach is based on SMPI [36], an MPI simulator provided as part of the SimGrid simulation framework. Since SMPI provides an MPI API, both on-line and off-line simulation are supported, that is, MPI applications can be executed either in real time or using traces gathered from previous executions. SMPI Courseware [4], [37] a very recent approach found in the current literature, proposes a set of assignments that can be incorporated into any HPC course. SMPI Courseware uses SMPI [36] to accurately simulate the execution of MPI applications on customised HPC systems. Although the authors do not specify the target course for this approach, they state that they are developing a large pedagogic corpus providing a wealth of reusable assignments that can be integrated across the HPC curriculum for both undergraduate and postgraduate courses.

While these approaches are certainly of interest for teaching distributed and HPC systems, they also present some drawbacks. Firstly, they are applied in postgraduate courses, in which students are expected to already possess basic distributed systems skills. Secondly, the simulation tools used lack an intuitive GUI, a feature that undoubtedly eases the configuration of parallel and distributed architectures in a simulation tool. In its absence, students are obliged to manually configure the simulation environment (e.g editing XML files to configure a distributed model in simGrid). Moreover, in order to execute MPI applications, an MPI distribution must be installed on the computer where the simulator is used, requiring additional software and libraries. Finally, these simulators do not encourage students to build HPC systems by sharing component models (e.g. CPUs, disks and memories).

[Insert Figure 2]

The current version of SIMCAN makes this simulator a very suitable and practical tool for teaching/learning distributed systems in undergraduate degrees. Moreover, SIMCAN is used to model both distributed and parallel environments, that is, students are able to model a wide range of scenarios, from basic distributed scenarios consisting of a client-server model to large and complex HPC architectures. The following is a summary of the main contributions of this simulator:

1. **Ease of modelling and configuring distributed and HPC systems.** A new user-friendly GUI, which has been written in Java, is provided with the simulator (Fig. 2). All the low-level details, including the language used for configuring hardware components, are hidden to students. Hence, students can easily model large and complex distributed systems without learning additional languages.
2. **Facility to share and distribute models further eases the modelling and configuring of complex systems.** The models created can easily be shared between students just by copying them into the repository, providing two significant benefits. First, the teacher can easily distribute examples and exercises among the students and thereby save them spending time on writing and configuring models. Second, the students can compare different solutions, e.g. provided by other students, by simply loading the corresponding models in the simulator.
3. **Easily-understandable results**. The log system has been adapted to be able to show only those messages that are required to explain a specific system in class. While the research-oriented version of SIMCAN prioritised executing the simulation and providing the results as quickly as possible, the education-oriented version includes a detailed and categorised log, enabling output messages to be fully customised to show only the required information.
4. **Portability**. This simulator is licenced under the GPL and can be executed on the currently most common plat-

forms, notably Windows, Linux and MacOS.

3 COURSE STRUCTURE & METHODOLOGY

The study reported on here concerns the Distributed System Programming (DSP) course, a classroom-based obligatory fourth-year course of the four-year Computer Engineering Degree at the Complutense University of Madrid.

3.1 Participants

The research involved 22 students (3 women and 19 men) with an age range from 20 to 30 years (an average age of 24.86 and standard deviation of 2.95) all from the same group (only one group per year follows this course). It is of interest to note that a significant percentage of students are in full-time work due to DSP being a fourth year course and to it being scheduled in the evening (18:00 to 20:00).

A quantitative analysis of the TAM model, based on the theory of structural equation models is not possible due to the small number of students [38] who have taken the subject so a qualitative data analysis will be carried out instead. Moreover, the gender distribution does not allow us to perform any gender analysis such as the one carried out in [20].

3.2 Learning Objectives

The overall goal of the course is to give the students a global understanding of the area of distributed systems and their programming. To this end, they must first familiarise themselves with the basic concepts of the field. Some of these concepts are introduced to the students throughout the course, since introducing them when needed helps them to be assimilated, but many of them must be introduced at the outset.

Building on this conceptual basis, the students are then able to gain understanding of the main architectures, communication paradigms, technologies, languages, techniques and algorithms currently in use in the field of distributed systems programming, to which they are introduced in the rest of the course.

The programming assignments that the students carry out throughout the course provide them with a chance to put the knowledge they have acquired of distributed systems programming into practice. This inevitably involves exploring in greater depth a particular set of distributed systems programming technologies, for which reason, it is important that the chosen technologies are representative and exercise important aspects of distributed systems programming. Moreover, these aspects should very across different assignments.

3.3 Learning Outcomes

The DSP course specification [39] defines the following competencies that the students should acquire:

LO1: The ability to analyse and evaluate parallel and distributed architectures, and develop and optimise software for them.

LO2: The ability to use simulation to analyse the performance and scalability of distributed applications executed on different architectures.

LO3: The capacity to identify the most relevant properties of distributed systems and applications.

LO4: The capability of using some of the well-known distributed algorithms and of identifying their main properties.

3.4 Course Structure

There are four hours of classes per week, two of them lectures and the other two labs. The lectures are practically-oriented and include a certain amount of problem-solving. In the lab sessions, the students solve preparatory exercises and work on the three assignments that are set each year. In the third of these assignments, the students use SIMCAN to model several different distributed architectures and analyse the execution on these architectures of programs that simulate distributed applications, these programs being provided by the lecturer.

3.5 Assignment using SIMCAN

In this section, we describe the SIMCAN-based assignment of the DSP course (Assignment3). Basically, this assignment consists of three different exercises.

Initially, the students use SIMCAN to model a basic client-server architecture, where one node represents the server, which stores different files, and other node represents the client, which executes an application that reads these files remotely using NFS (Network File System). These nodes are connected to a switch using a communications network. This architecture is modelled using the GUI provided by SIMCAN (See Fig. 2). The idea is to analyse the system performance when the underlying system infrastructure is modified, for instance, by using a different communication network or different disk drives in the server node. Thus, students are able to analyse how a single modification in the architecture impacts on the overall system performance (e.g. the observed speed-up of a file reading application when a faster network – or a faster disk drive - is used)

The second exercise consists of modifying the previous architecture. The system being modelled now consists of one client node and 8 server nodes, where each file is distributed among the server nodes. The objective of this exercise

is to analyse the distribution of data and how parallel access to this data may considerably increase overall system performance. To this end, the client node uses a parallel file system to access the data stored in the servers. Again, students then analyse system performance using different network and disk-drive configurations. They also use differing values for the parameters of the parallel file system, such as the block size used for accessing the data.

Finally, the third exercise is to model a complete HPC system. We now divide the system into two parts: computing and storage. The former is modelled using racks, which contains blades where the applications are executed. These blades can be configured to contain a customised number of CPU cores, memory and disk. The latter consists of the storage nodes, which house the files containing the data remotely accessed by the computing nodes. An MPI application is executed in the computing nodes, while the data required to be processed by this application is stored in the storage servers and is accessed using a parallel file system. Both the application and the system architecture are modified in order to study how this affects the overall system performance. The application configuration is varied by using different numbers of processes and different block sizes for accessing and sending data among the processes. The architecture is varied by using different numbers of CPUs per blade, different numbers of computing blades, different numbers of storage servers and different communication networks. In this case, the students must analyse the scalability of the system and locate bottlenecks. Of particular interest in this exercise is when the students observe that using "better components", such as a faster network, does not provide the "expected" performance improvement due to the existence of a bottleneck, for example, in the server nodes.

With reference to the learning outcomes discussed in Section 3.3, the competencies acquired in carrying out this assignment are as follows:

LO1: Little, since only performance aspects of analysing and evaluating parallel and distributed architectures are exercised.

LO2: Greatly, since students gain hands-on experience in using simulation to analyse the performance and scalability of distributed applications on a range of architectures, from a 2-node client-server architecture to a complete HPC system.

LO3: Some, since students gain practice in identifying a few of the most important properties of distributed systems such as scalability and flexibility, as well as performance.

LO4: Significantly, since the assignments used are based on the simulation of well-known distributed algorithms and thereby aid understanding of these algorithms. Moreover, many of the most well-known algorithms of this field could be programmed for the simulator and used in student assignments.

3.6 Evaluation

The overall grade of the course is calculated by taking into account both the three practical assignments and a final exam (a written test). Thus, the overall grade is calculated by the following expression:

$$fg = 0.3 \cdot (0.33a_1 + 0.33a_2 + 0.34a_3) + 0.7fe, \quad (1)$$

where fg is the overall grade, $a_{1..3}$ are the marks for the practical assignments and fe is the mark for the final exam.

There are two exceptions where the final grade is not calculated by the previous expression, that is, the student obtains a mark less than 4 in the written test or the student gets a mark less than 4 in any of the three practical assignments. In these cases the student's overall grade is a fail.

It is important to remark that it is not mandatory that the practical assignments be developed and submitted in the lab. In particular, since SIMCAN can be executed on a regular computer, it provides a high degree of flexibility for developing Assignment3 outside lab hours, thereby allowing the students to invest more time in studying for the final exam. To facilitate deployment of the simulator on other computers, the students were provided with a virtual machine in which SIMCAN was preinstalled.

4 Analysis Performed

To study how the SIMCAN simulator improves learning distributed systems we analyse two different metrics: students' perceptions and course marks.

4.1 Student Perceptions

To measure the students' perceptions, we have used a slight adaptation of the TAM. We designed a survey for evaluating the students' perception both of the course itself and of how SIMCAN improves their learning of distributed systems programming. An English translation of the questionnaire used can be found in Table 1.

[Insert Table1]

The first part of the survey, related to the course itself, is based on the three-constructor original TAM model, whereas the second part, related to SIMCAN, includes a new constructor, *perceived interest* (Fig. 1, dashed line). Moreover, some additional questions have been added to evaluate the students' perception of the lectures and lab sessions. Observe that, in Table 1, the following notation has been used: DS stands for distributed systems, SIM for simulator,

and AQ for additional questions.

The questionnaire uses a five-point Likert scale, ranging from (1) “strongly disagree” to (5) “strongly agree”, to collect students’ opinions, there being several items for each TAM constructor. Students’ perceptions are computed using the mean score of the students’ answers to each constructor.

A student survey was conducted on two separate occasions during the DSP course: at the start and just after the submission of the SIMCAN-based assignment. The two surveys are slightly different since the first focuses on a priori expectations, while the second evaluates a posteriori perceptions. For instance, an item which in the initial survey is "I think I will learn from the lab sessions with the simulator", in the final survey becomes "I've learned from the lab sessions with the simulator" (Table 1 only shows the final survey). This use of two surveys enables us to check whether the course contents and, in particular, the use of the simulator, have lived up to expectations.

The questionnaire responses were anonymous, but students were asked to answer some personal questions (for instance, “what is your favourite (music) band?”), in order to be able to match the two questionnaires in the analysis.

4.2 Assessment Results

We compared the marks obtained both for the assignments and for the final exam with those of the previous year (2016), in which the same assessment scheme was used but the SIMCAN simulator was not.

5 Results

5.1 Student Perceptions

First of all, we analysed the reliability of the survey, calculating Cronbach’s alpha on the proposed questions within each constructor. We obtained in all cases values over 0.7, a quality threshold widely accepted [40].

[Insert Table2]

In Fig. 3 and Table 2 we illustrate the percentages of the different answers for each question. We have grouped “agree” (4) and “strongly agree” (5) as *High*, and “disagree” (2) and “strongly disagree” (1) as *Low*. We do not show the response (3) since “neither agree nor disagree” represents either having no clear view or indifference. From here, we can conclude that the students’ perception, both for the subject and for the simulator, is good and, moreover, this perception is better in the final survey, carried out after studying the subject and using the simulator, than in the first survey, carried out at the beginning of the course.

[Insert Figure3]

If we focus on the final survey, we observe that more than 50% of students have answered “agree” (4) or “strongly agree” (5) to all the items, except for item I07, where only 30% consider that this subject is easy. However, even though more than 60% agree or strongly agree in recommending this simulator to other students (item I23), only 40% think that they will use simulators to learn other subjects (item I24); we interpret this is being due to uncertainty about whether this would be feasible in courses not yet studied.

Moreover, we highlight item I04 (“In general, DS is useful”), for which 81.82% of students agree (4) or strongly agree (5); item I11 (“In general, I think I'll need the DS from now on”), with 59.09%; item I15 (“In general, the simulator helps me to understand DS”), with 72.73%; item I18 (“In general, the simulator is easy to use”) with 63.64%; and item I21 (“In general, I find it enjoyable to work with the simulator”), with 77.27%. Finally, we want to point out that almost 80% of those surveyed find the lab sessions with the simulator interesting (“agree” or “strongly agree”, item I27) and have learned from the lab sessions with the simulator (“agree” or “strongly agree”, item I28).

Table 3 shows the basic psychometric characteristics. It can be observed that the mean values of all constructors are above 3.0 (the “indifferent” value) and, moreover, all of them improve when the students have studied the subject and have used the simulator (final survey), especially the “perceived ease of use” of the simulator (PE_SIM), “additional questions” regarding the demonstrations in lectures (AQ_DS) and “additional questions” regarding the lab sessions with the simulator (AQ_SIM). This shows an increased acceptance of, and interest in, the use of simulation in learning, on the part of the students.

[Insert Table3]

We point out that the mean values are quite high in the constructor measuring the "perceived usefulness" of the subject (PU_DS) and of the simulator (PU_SIM), as well as in the constructor measuring the "perceived interest" that the simulator produces (PI_SIM). Finally, from the mean value of additional questions relating to the simulator (AQ_SIM), we can conclude that the students appreciated the simulator’s utility as a learning tool.

5.2 Evaluation Results

Table 4 shows the mean mark for the practical assignments, the mean mark for the final exam and the mean overall grade (obtained with expression (1)) for the students in the 2016 and 2017 editions of the DSP course. It can be observed that the results in the exam and in the final grade are slightly better in 2017, when the simulator was used, than in 2016.

[Insert Table4]

We studied to what extent the assignment marks can be used as a predictor of the final exam mark and/or the overall grade. Table 5 shows the corresponding linear regression R2 coefficients. This coefficient is bigger for Assignment3, the SIMCAN-based assignment (for example, Figure 4 shows the linear regression between the mark for Assignment3 and the overall grade, R2 = 0.5172). Thus, Assignment3 can be used - albeit with some uncertainty - as the proposed predictor. This fact highlights the import role of this assignment in the DSP course.
[Insert Figure4]

Finally, Table 6 shows the significant correlations among the individual assignment marks, the mean mark obtained in the assignments, the final exam mark and the overall grade. In particular, the importance of Assignment3 can be appreciated, indicating that a well-executed Assignment3 favours learning of DSP since it helps to obtain a good mark in the final exam and, as a consequence, in the overall grade.
[Insert Table5]
[Insert Table6]

6 Conclusion

In this paper we presented a study of the use of the SIMCAN simulator in the teaching and learning of Distributed System Programming (DSP), a fourth year course of the Computer Engineering Degree. The SIMCAN-based assignment introduced in this course allowed the students to exercise aspects of distributed systems programming that could not be exercised otherwise. It involved the students modelling several different distributed architectures and analysing the execution on these architectures of programs that simulate distributed applications, these programs being provided by the lecturer.

Our evaluation of the role of the SIMCAN simulator in improving learning of distributed and HPC systems included the investigation of both the course-assessment results and the students' perceptions. To measure the latter, we used a slight adaptation of the TAM.

We designed a survey for evaluating the students' perception both of the course itself and of how SIMCAN improves their learning of distributed systems programming. It can be observed that the mean values of all constructors are above 3.0 (the "indifferent" value) and, moreover, improved when the students had completed the course and used the simulator. In particular, the mean values were quite high in the constructor "perceived usefulness" for the course itself and for the simulator.

We point out also that almost 80% of those surveyed found the lab sessions with the simulator interesting and believed they had learned from them.

Finally, it is of interest to note that we found a significant correlation between the mark for the SIMCAN-based assignment, on the one hand, and the mark for the final exam and the overall grade, on the other. We also found that the mean mark obtained in the assignments (and even more so, that obtained in the SIMCAN-based assignment) can be used as a predictor to estimate both the mark obtained in the final exam and the overall grade. Moreover, if we take into account that the results, both in the exam and in the final grade, are better in 2017 than in 2016, where SIMCAN wasn't used in the laboratory classes, we can conclude that SIMCAN has a positive impact on the learning process of distributed system programming.

Though the size of our study inevitably means that any conclusions are tentative, the results obtained are promising. We intend to study the use of SIMCAN in distributed systems courses in other universities, from which we expect to gather valuable data from a larger number of students. Finally, for the upcoming editions of the DSP course used in the present study, we are planning to develop stochastic modules for more realistic modelling of the behaviour of components such as solid state disks, memories and schedulers.

References

[1] *Curriculum Guidelines for Undergraduate Programs in Computer Science*, The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM), IEEE Computer Society ACM Press & IEEE CS Press, Dec. 20, 2013.

[2] K. Geven and R. Santa, "Survey Analysis, Time for Student Centred Learning," in *European Student Union*, Bucharest, 2010.

[3] A. Degomme, A. Legrand, G. Markomanolis, M. Quinson, M. Stillwell and F. Suter, "Simulating MPI applications: the SMPI approach," IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 2387–2400, 2017, DOI: https://doi.org/ 10.1109/TPDS.2017. 2669305

[4] H. Casanova, A. Legrand, M. Quinson and F. Suter. "SMPI Courseware: Teaching Distributed-Memory Computing with MPI in Simulation," in Proc. EduHPC-18, Nov 2018, pp.1-10, <hal-01891513>

[5] L. Peterson and V. S. Pai, "Experience-driven Experimental Systems Research," *Communications of the ACM*, vol. 50, no. 11, pp. 38-44, Nov. 2007, DOI: http://dx.doi.org/10.1145/1297797. 1297820.

[6] J. Ryoo, A. Techatassanasoontorn and D. Lee, "Security Education Using Second Life," IEEE Security & Privacy, vol. 7, no. 2, pp. 71-74, March-April 2009, DOI: 10.1109/MSP.2009.49

[7] N. Rutten, W. R. Van Joolingen and J. T. Van der Veen, "The Learning Effects of Computer Simulations in Science Education," Computers and Education, vol. 58, no. 1, pp. 136-153, Jan. 2012, DOI: https://doi.org/10. 1016/j.compedu.2011.07.017.

[8] M. Alnoukari, M. Shafaamry and K. Autoni, "Simulation for Computer Sciences Education," *Communications of the ACS*, vol. 6, no 1, pp. 1-19, Jan. 2013.

[9] A. Núñez, J. Fernández, R. Filgueira, F. García, and J. Carretero, "SIMCAN: A flexible, scalable and expandable simulation platform for modelling and simulating distributed architectures and applications," *Simulation Modelling Practice and Theory*, vol. 20, no. 1, pp. 12–32, Jan. 2012, DOI: https://doi.org/10.1016/j.simpat.2011.08.009.

[10] Venkatesh, V., Morris, M. G., Davis, G. B. & Davis, F. D. (2003). User Acceptance of Information Technology: Toward a Unified View. MIS Q., 27(3), 425–478. Retrieved from http://dl.acm.org/citation.cfm?id=2017197.2017202

[11] F. D. Davis, "Perceived Usefulness, Perceived Ease of Use and User Acceptance of Information Technology," *MIS Quarterly*, vol. 13, no 3, pp. 319–340, Sep. 1989, DOI: http://dx.doi.org/10.2307/249008.

[12] F. D. Davis, "User acceptance of information technology: system characteristics, user perceptions and behavioral impacts," *International Journal of Man-Machine Studies*, vol. 38 no. 3, pp. 475–487, Mar. 1993, https://doi.org/10.1006/imms.1993.1022.

[13] M. Fishbein and I. Ajzen, "Belief, attitude, intention and behavior: an introduction to theory and research," Reading, MA: Addison-Wesley, 1975.

[14] V. Venkatesh, M. G. Morris, G. B. Davis and F. D. Davis, "User Acceptance of Information Technology: Toward a Unified View," *MIS Q.*, vol. 27, no. 3, pp. 425–478, Sep. 2003, DOI: http://dx.doi.org/ 10.2307/30036540.

[15] W. R. King, and J. He, "A meta-analysis of the technology acceptance model," *Information & Management*, vol. 43, no. 6, pp. 740–755, Sep. 2006, DOI: http://dx.doi.org/10.1016/j.im.2006.05.003.

[16] M. Turner, B. Kitchenham, P. Brereton, S. Charters and D. Budgen, "Does the technology acceptance model predict actual use? A systematic literature review". *Information and Software Technology* , vol. 52, no. 5, pp. 463–479, May. 2010, DOI: http://dx.doi.org/10.1016/j.infsof.2009.11. 005.

[17] A. Padilla-Meléndez, A. Garrido-Moreno, and A. R. D. Aguila-Obra, "Factors affecting e-collaboration technology use among management students," *Computers & Education*, vol. 51, no. 2, pp. 609–623, Sep. 2008, DOI: http://dx.doi.org/10.1016/j.compedu.2007.06.013.

[18] H. M Selim, "An empirical investigation of student acceptance of course websites," *Computers & Education*, vol. 40, no. 4, pp. 343–360, May. 2003, DOI: http://dx.doi.org/10.1016/S0360-1315(02)00142-2.

[19] M. A. Rubio, R. Romero-Zaliz, C. Mañoso and A. P. de Madrid, "Enhancing an introductory programming course with physical computing modules," in *Frontiers in Education Conference*, IEEE, 2014, pp. 1019–1026.

[20] M.A. Rubio, R. Romero-Zaliz, C. Mañoso and A.P. de Madrid, "Closing the gender gap in an introductory programming course, " *Computers & Education*, vol. 82, pp. 409–420, Mar. 2015, DOI: http://dx.doi.org/10. 1016/j.compedu.2014.12.003.

[21] I. F. Liu, M. C. Chen, Y. S. Sun, D. Wible and C. H. Kuo, "Extending the TAM model to explore the factors that affect Intention to Use an Online Learning Community," *Computers & Education*, vol. 54, no. 2, pp. 600– 610, Feb. 2010, DOI: http://dx.doi.org/10.1016/j.compedu.2009.09.009.

[22] A. Núñez. "New Contributions for Modeling and Simulating High Performance Computing Applications on Parallel and Distributed Architectures," Ph.D. dissertation, Dept. Computer Science, Univ. Carlos III, Madrid, Spain, 2011.

[23] A. Varga, "The OMNeT++ discrete event simulation system," in *European Simulation Multiconference*, Prague, Czech Republic, 2001.

[24] G. Szaszko, "INET framework for the OMNeT++ discrete event simulator," 2017. [Online] Available: https://github.com/inet-framework/inet.

[25] J. Furnaghan. Oversim, "The Overlay Simulation Framework," 2014. [Online] Available: https://github.com/reines/oversim, 2014.

[26] V. Vesely, "RINAsim. Recursive InterNetwork Architecture simulator," 2017. [Online] Available: https://github.com/ kvetak/RINA, 2017.

[27] A. Núñez, J. Fernández, J. D. García, L. Prada, and J. Carretero, "New Techniques for Modeling File Data Distribution on Storage Nodes," *in Proc. ANNS*, Ottawa, Canada, 2008, pp. 175–182.

[28] A. Núñez, J. Fernández, J. D. García, and J. Carretero, "Analyzing Scalable High-Performance I/O Architectures," in *Proc. PDPTA*, 2008, pp. 631–637.

[29] A. Núñez, C. Andrés, and M. G. Merayo, "Optimizing the Trade-offs Between Cost and Performance in Scientific Computing," *Procedia Computer Science, International Conference on Computational Science*, vol. 9, pp. 498–507, 2012.

[30] A. Núñez, J. Fernández, J. D. García, and J. Carretero, "New techniques for simulating high performance MPI applications on large storage networks," in *Proc. HiperIO*, Tsukuba, Japan, 2008, pp. 444–452.

[31] A. Núñez, J. Fernández, J. Carretero, L. Prada, and M. Blaum, "Optimizing Distributed Architectures to Improve Performance on Checkpointing Applications," in *Proc. HPCC*, 2011, pp. 487–492.

[32] G. Zarza, D. Lugones, D. Franco, and E. Luque, "An Innovative Teaching Strategy to Understand High-Performance Systems through Performance Evaluation," *in Proc. ICCS*, 2012, vol. 9, pp. 1733–1742, DOI: https://doi.org/10.1016/j.procs.2012.04. 191

[33] "Riverbed Modeler,". 2018. [Online] Available: https://www. riverbed.com/fr/products/ steelcentral/steelcentral-riverbed-modeler.html.

[34] "The SimGrid Project," [Online] Available: http:// simgrid.org/, 2018.

[35] B. Donassolo, H. Casanova, A. Legrand, and P. Velho, "Fast and scalable simulation of volunteer computing systems using SimGrid," in *Proc. Workshop Large-Scale Syst. Appl. Perform.*, Jun. 2010, pp. 605–612, DOI: https://doi.org/10.1145/1851476. 1851565

[36] P. N. Clauss, M. Stillwell, S. Genaud, F. Suter, H. Casanova, and M. Quinson, "Single node on-line simulation of MPI applications with SMPI," in *Proc. 25th IEEE Int. Parallel Distrib. Process. Symp.*, 2011, pp. 664–675, DOI: https://doi.org/10.1109/ IPDPS.2011.69

[37] "SMPI Courseware," https : / / simgrid . github . io / SMPI CourseWare/, 2018.

[38] A. Boomsma, "The robustness of maximum likelihood estimation in structural equation models," Cambridge University Press, New York, NY, USA, 1987.

[39] "DSP course specification," [Online] Available: http: / / web.fdi.ucm.es / UCMFiles / pdf / FICHAS_DOCENTES / 2018 / 1455.pdf.

[40] J. R. A. Santos, "Cronbach's alpha: A tool for assessing the reliability of scales," *Journal of Extension*, vol. 37, no.2, pp. 1–5, Apr. 1999.