A Model for Teaching Distributed Computing in a Distance-based Educational Environment

Petra le Roux[1], Alta van der Merwe[12], Aurona Gerber[12]
[1]School of Computing, University of South Africa, Pretoria
[2]Meraka Institute, CSIR, Pretoria
lrouxp@unisa.ac.za, alta@meraka.org.za, agerber@csir.co.za

Abstract - Due to the prolific growth in connectivity, the development and implementation of *distributed systems* receives a lot of attention. Several technologies and languages exist for the development and implementation of such distributed systems; however, teaching students in these new technologies remains a challenge. Even though several models for teaching computer programming and teaching programming in a *distance-based educational environment (DEE)* exist, limited literature is available on models for teaching distributed computing in a DEE. Here our research we examine how distributed computing should be taught in a DEE in order to ensure effective and quality learning for students, specifically by investigating both the specific characteristics of distributed systems technologies and the models used for teaching programming in DEE. The required effectiveness and quality should be comparable to those for students exposed to laboratories, as commonly found in residential universities. This led to the identification of the factors that contribute to the success of teaching distributed computing and determine how these factors can be integrated into a proposed distributed systems distance-based teaching model we call the Independent Distributed Learning Model (IDLM).

I. INTRODUCTION

The design and implementation of software is a difficult and expensive process [1]. This process is also complex, even in a homogeneous environment, that is, an environment consisting of a single, stable platform, using a single operating system and a single programming language, usually from a single vendor. The complexity increases immensely if software is developed in a heterogeneous environment in which the vendors of the hardware and system software may differ. Such an environment adds a number of additional complexities to new software development [2, 3, 1, 4]. The new software must run on a *network* of hosts, and these *hosts* may run different *operating systems*. Furthermore, components of the new software will probably have to be integrated into *legacy* systems. Adding to the complexity is the probable use of different *programming languages* for different components of the new software system. All these complexities generally cause an increase in the cost of developing and maintaining software.

To address some of these complexities, [1] proposes that new software needs to be designed and developed as a set of integrating components that can *communicate* across the boundaries of a network, different operating systems and different programming languages. Furthermore, legacy code has to be *wrapped* so as to resemble *components* to ensure that they can be integrated into new systems. These components also have to be *transferable to new environments* to ensure that they can be integrated into various applications.

Distributed computing focuses on the hardware, software and middleware that allows for a collection of autonomous hosts connected through a computer network to coordinate their activities in such a way that users perceive the system as a single, integrated computing facility. Distributed programming and distributed computing frameworks assist in and simplify the design and development of systems that communicate across the boundaries of a network, running on different operating systems and which were written in different programming languages. These frameworks (commonly referred to as *middleware)* offer flexibility and new ways of integrating existing and new technology, as well as new ways of facilitating communication between systems.

Distributed computing frameworks have raised expectations that these highly functional systems can solve the majority of computing problems. Experience, however, has shown that it is challenging to build such distributed applications [5]. Apart from the difficulties and challenges involved, distributed computing is increasingly being used as a basis for the World Wide Web and distributed network-related software developments.

Furthermore, the use of distributed computing frameworks is complex because the use of distributed computing components (plug and play) presents various challenges. Besides the embedded complexities of this subject, the teaching thereof poses various problems to both student and teacher. In a residential institution, students can be exposed to the various elements of a distributed computing environment under laboratory conditions. This might simplify some of the complexity associated with the learning of distributed computing. In a DEE, however, a different approach is needed to ensure that students receive the same quality of teaching and are able to experience the same degree of learning as under laboratory conditions. Teaching distributed computing without laboratory sessions is more challenging and might fail completely if a set of clear guidelines is not available to direct the teaching and learning process.

We are concerned with the question of how distributed computing should be taught in a DEE to ensure effective

learning for students. The required effectiveness should be comparable to those for students exposed to laboratory conditions such as commonly found at residential universities. Therefore, this paper focuses on determining the factors that contribute to the success of teaching distributed computing and subsequently how these factors can be integrated into a proposed model for the teaching and learning of distributed computing in a DEE. We call this model the Independent Distributed Learning Model (IDLM). Section II provides the background by motivating reasons for teaching an undergraduate course of this nature, as well as discussing existing systems and why there is a need for the proposed IDLM. Section III addresses the approach and method followed to define the IDLM. Sections IV and V discuss the architecture, components, functionality and characteristics of the IDLM. Section VI discusses a case study to validate the proposed Independent Distributed Learning Model. A discussion of experiences gained followed in section VII and a conclusion in section VIII.

II. TEACHING DISTRIBUTED COMPUTING

The rationale for teaching distributed computing as a university elective course is highlighted in the final report of the Computing Curricula 2005 project[1] [6] and are twofold: changes in the computing field and the need for advanced programming courses.

Technical changes in computer science are both evolutionary (exponential increase in available computing power) and revolutionary (the rapid growth of networking after the appearance of the World Wide Web). This rapid evolution of the computer science discipline has had a profound effect on computer science education, affecting both content and pedagogy. Computing education is also affected by changes in the *cultural* and *sociological* context in which it occurs. The following changes, for example, have all had an influence on the nature of the educational process: changes in pedagogy enabled by new technologies, the dramatic growth of computing throughout the world, the growing economic influence of computing technology, greater acceptance of computer science as an academic discipline and broadening of the discipline.

Advanced courses in undergraduate studies serve three purposes: to expose students to advanced material beyond the core, to demonstrate applications of fundamental concepts presented in the core courses, and to provide students with a depth of knowledge in at least one subarea of computer science. One of the advanced programming courses included in the CC2005 is distributed computing.

To determine the components of a model that would contribute to the success of teaching and learning distributed computing in a DEE, we explored the state of affairs when teaching and learning *programming languages*, *programming in DEE* and *distributed computing*.

[1] CC2005 is a joint undertaking of the Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-CS) and the Association for Computing Machinery (ACM) to develop curricular guidelines for undergraduate programmes in computing.

A. Teaching and Learning of programming languages

The available models for the teaching and learning of programming languages are mainly aimed at contact or laboratory sessions, as is the practice in residential universities [7,8]. Typical problems and solutions are identified. All research efforts and development of models which aid in the teaching and learning of programming, especially at introductory level, have one purpose: to ease the learning of the programming language by eliminating all complexities that do not directly contribute to the achievement of the learning objectives [6,7]. For this we developed a generic model consisting of two components: a teaching module and a student module (see Figure 1). The *teaching module* consists of an *expertise module*, which presents the domain knowledge that the teacher intends to be mastered by the student. The text that represents the command of the computer programming languages to be taught is maintained in the *tutoring text*, whereas the *semantic rules* contain the structure of commands to be taught. The s*tudent module* comprises a *GUI*, the *tutoring text*, and the *student profile.*

B. Teaching and Learning programming in Distance-based Educational Environments

Systems used in the teaching of programming in a DEE are classified as intelligent tutoring systems (ITS) [9,10]. An ITS for effective teaching and learning in a DEE is built on the client-server model. In its basic form, the client side consists of a *teaching agent* or *agents* and *student agents*. The server provides the content and infrastructure needed to present a course. A *tutoring agent* and the *Internet* are added to the generic model. The tutoring agent represents the knowledge to be taught; it encompasses modelling of the knowledge, and management and coordination of the learning activities. The Internet serves as a vehicle for communication.

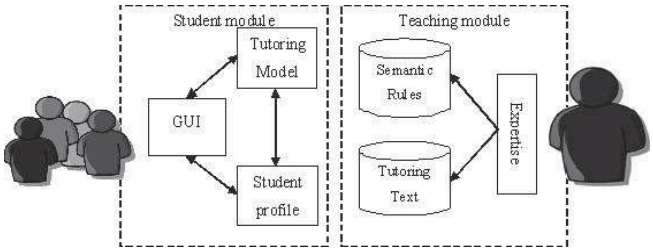

Figure 1. Generic Model for Teaching Programming Languages

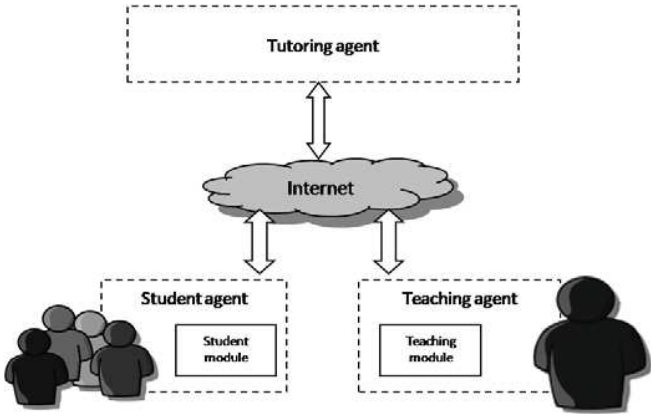

Figure 2. Generic Model for Teaching in a DEE

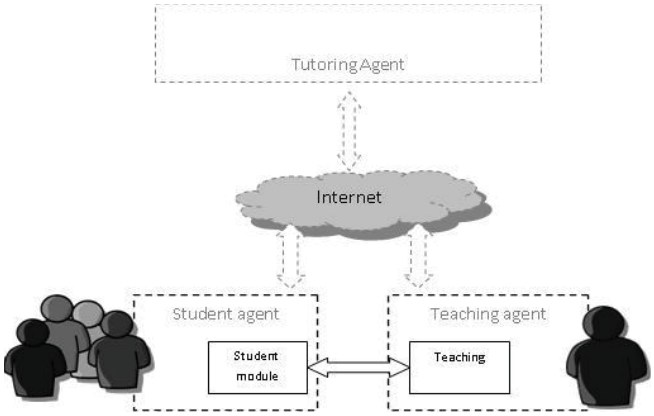

Figure 3. Generic Model for Teaching Distributed Computing

The student and teacher communicate with the tutoring agent through the Internet, or WWW. Thus, the teaching module is contained in a teaching agent, whereas the student module is contained in a student agent. The student agent retrieves the tutoring dialog that a student wants to learn through the Internet. The teaching agent communicates with the tutoring agent to maintain the knowledge to be taught. Figure 2 depicts the updated generic model.

C. Teaching and Learning Distributed Computing

Since distributed algorithms are difficult to grasp and also to implement and debug, models that aid in the demonstration of complex relationships and dynamic processes are identified that have the potential to support teachers and learners [11]. Also, when the Internet is used, use of XML, Java applets, etc. help to ease the complexities, since explicit installation is not needed on the student side; only a Java-enabled browser (available on most computers) is needed. The model consists of two components, a *teaching module* and a *student module*, but the content is presented in a distinctive way. This model is depicted in Figure 3.

D. A Consolidated Approach

When the teaching and learning of distributed systems in a DEE takes place, a number of additional requirements become apparent: (i) a need for a system that is not dependable on a laboratory environment, (ii) a system that addresses the asynchronous and geographically dispersed nature of DEE, and (iii) a system that addresses the economic realities of students. Synchronous communication, broadband and commercially available software may be out of reach for an average student.

The focus is therefore on a model that has the characteristics for effective and quality teaching and learning to take place as well as addressing the above-mentioned challenges and economic realities.

III. METHOD

A three phase approach was used for identification of the proposed distributed computing model. Phase 1 focused on the identification of the elements of distributed computing in order to compile a body of knowledge needed to be present in a model for effective teaching and learning of distributed computing to take place. Phase 2 included an investigation of available models for the teaching of computer programming, programming in a DEE and distributed computing in order to identify success factors for teaching distributed computing in a DEE. Lastly, Phase 3 was used to establish and verify a model for the effective teaching and learning of distributed computing in a DEE.

The research approach used was a *qualitative research* method, which was developed in the social sciences to enable researchers to study social and cultural phenomena; situations in which people and different processes are involved. The research approach employed was *design research,* where the purpose was the creation of an artefact.

The artefact developed was in the form of the independent distributed learning model (IDLM). For data collection and verification of the IDLM, a *case study* approach was followed. The selected case-study environment was a single-case design. A *survey* was also used to gather perceptions on the approach suggested and followed by the researchers.

IV. INDEPENDENT DISTRIBUTED LEARNING MODEL (IDLM)

In order for effective teaching and learning of distributed computing to take place in a DEE, learners need a way to create their own space, where distributed computing concepts can be simulated and studied asynchronously as economic realities do not always allow for synchronous communications. Learners must also be able to work at their own pace and time, with the ability to take full advantage of the synchronous functionalities available in a DEE. These requirements are depicted in Figure 4.

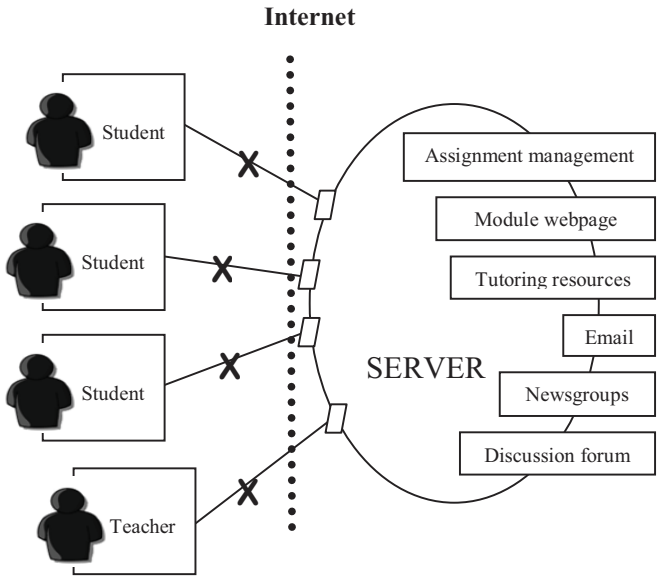

Figure 4. Schematic representation how teaching and learning have to take place in a DEE

Thus, the suggested IDLM consists of different spaces, the resource space, broker space and the learning space (Figure 5). Firstly, the **resource space (RS)** acts as a server and facilitates an environment in which all the resources and functionality needed to accommodate the learning experience reside. These resources are available to the learning space, in which the student and the teacher reside, through the broker space. The **broker space (BS)** acts as a *middleman,* which pairs requests from the learning space with the resource space. The learning space makes its functionality and needs known to the broker space. The main responsibility of the broker space is to identify and match these requests to the resource space. Thus, the broker space is responsible for communication between the resource space and the learning space. The **learning space (LS)** facilitates an environment in which the teacher and student can execute their respective tasks.

Any learning environment requires support infrastructure; so much more an e-learning environment. The infrastructure needed to support e-learning includes inter alia, remote servers, databases and software systems to create the learning space in which teaching and learning can take place. This environment might be synchronous, necessitating continuous remote resource support, whereby the learner and teacher interact in the same time frame with their respective learning or teaching environments and with each other. The environment might also be an asynchronous learning environment in which remote resources are responsible for initial set-up of the learning environment without continuous monitoring and support of this environment. In an asynchronous environment, students work offline for most of their learning session, but might at any time choose to reconnect to the resource space to interact with the resources for tasks such as queries, assignment submission, etc.

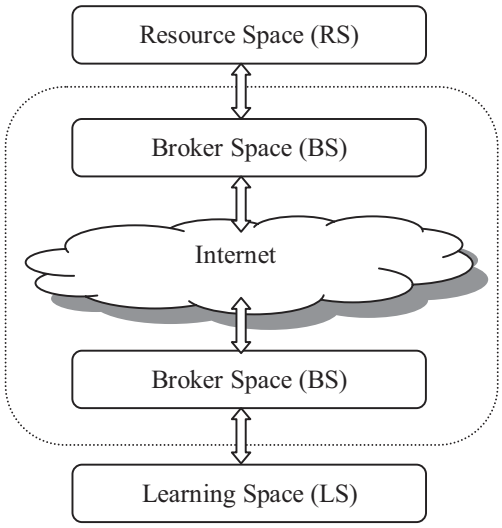

Figure 5. The Independent Distributed Model

A. The Resource Space

The significance of the resource space is that it combines the different resources into an area in which technical support is maintained. Although the infrastructure and the systems within this space might be heterogeneous and distributed as regards specific location, this space consists of the different resource components. Preservation and, consequently, maintenance of a well-defined and specialised space is less complex than when these resources are conceptually scattered throughout every space. The components of the resource space are tutoring resources, assignment management and communication, which includes the course webpage, newsgroups, email and a discussion forum as depicted in Figure 6.

B. The Broker Space

The broker space acts as a *middleman,* which pairs requests from the learning space with the resource space. The learning space makes its functionality and needs known to the broker space. The main responsibility of the broker space is to identify and match these requests to the resource space. Thus, the broker space is responsible for communication between the resource space and the learning space.

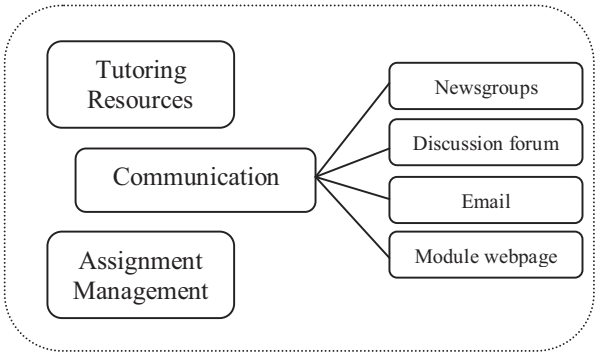

Figure 6. Resource Space

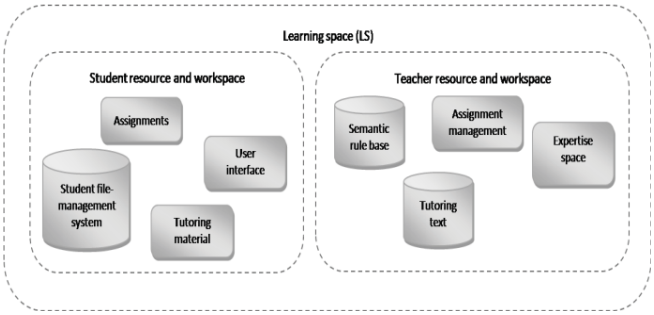

Figure 7. Learning Space

C. Learning Space

The learning space consists of both the s*tudent resource and workspace* and the *teacher resource and workspace* and can be viewed as a facility that allows the student and teacher to fulfil their tasks. The significance of the learning space is that it localises the workspace and resources of both the teacher and the student. The student resource and workspace facilitates an environment in which the student can communicate with the resource space, via a user interface, through the Internet. The components of the student resource and workspace are a user interface, tutoring material and a student file-management system. The teacher resource and workspace facilitates an environment in which the teacher can communicate with the resource space through the Internet. The components of the teacher resource and workspace include an expertise space, semantic rule base, tutoring text and assignment management. See Figure 7.

V. IDLM DETAIL LEVEL

For each of the spaces on the IDLM, more detail is available in [12]. In the specification of the IDLM, UML diagrams were used, more specifically *use-case diagrams,* which model the users and their interactions with the system at a very high level of abstraction, and *activity diagrams,* which are used to describe the workflow behaviour of a system. One example of a use-case diagram of the resource space is depicted in Figure 8.

VI. CASE STUDY: UNIVERSITY OF SOUTH AFRICA

The IDLM was used in a distance educational environment and as guideline in presenting the module to third-year students at the University of South Africa (Unisa). The module is offered by Unisa as part of the undergraduate studies towards a Bachelor in Computer Science or Information Systems. It is an advanced computer science elective module called *Advanced Programming.*

Essentially the software used in the course consisted of a C++ compiler and a CORBA environment. There were a number of permutations of compilers, CORBA ORBs and operating systems which could be used to achieve the objectives of the practical part of the course. However, a balance was struck between allowing students to work in their preferred environment, providing a set of freely distributable tools and providing detailed guidance and support for students and their environments. The prescribed tools were the *Mico ORB*, the *Dev C++ editor*, the *minGW compiler* and the *make facility*. Communication with students took place in the form of *tutorial letters*, the *module webpage*, the d*iscussion forum* and the *module email*.

The student numbers varied from 125 to 200[2], and two lecturers were responsible for the module. The data were collected via questionnaires and were delivered to the students via email, a facility offered by the University's Administrative Department. The number of students registered for the module in question was 247. Seventy-eight (78) students cancelled their studies during the year, leaving a total of 169 students. The number of students who received the questionnaire via email was 151. Thirty (30) of the emails were returned as undelivered mail, possibly because students did not update their personal information via the existing administration channels. Therefore the sample size that received the questionnaire was 121. The number returned was 33, resulting in a response rate of 27%.

After the 1 year period, a survey was conducted on the experiences of the implementation. Questions covered the effectiveness of the teaching methods used, experiences on the usefulness of the tools employed, the effectiveness of the means of communication, suggestions for enhancements and to what extend distributed computing was applicable in the working environment of the student.

Feedback from the students indicated that the methods described in the IDML were found to be helpful by 61%, opposed to 18% of students who found the methods not to be helpful. The number of students who did not have strong feelings about whether the methods contributed to successful learning was 21%.

The data gave strong evidence that the majority of the students made use of the software provided, but found the editor to be least helpful and made use of their own editor. After an initial installation process, which some students found to be cumbersome and difficult, the majority again found the software easy to use.

The results indicated that the discussion forum was found to be most helpful by 91% of students, whereas the module email was found to be least helpful and was used by 24% of students. Tutorial letters, as opposed to the module website, remained the preferred means of communication.

The survey further indicated that the practical component of the module needed more attention, both in respect of tutoring text (additional exercises, more detailed discussions, etc.) and software tools and most of the students did not work in an environment in which the development of distributed systems was applicable.

[2] Enrolment figures from 1997 to 2005.

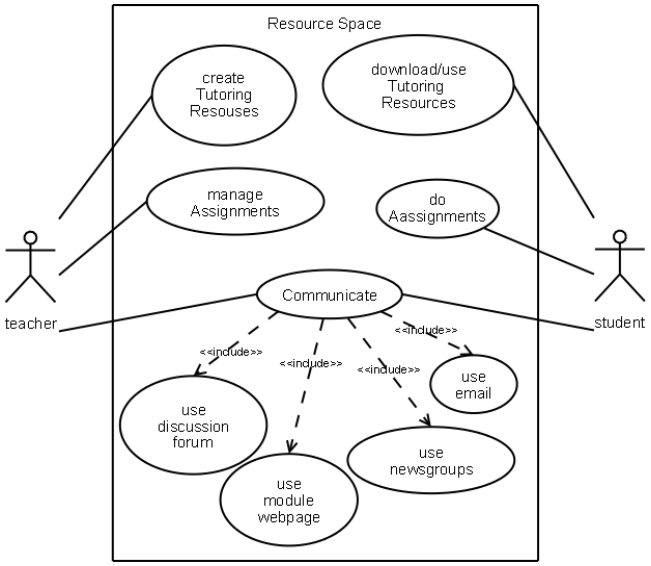

Figure 8. Use-case Diagram – Resource Space

VII. DISCUSSION

This research contributed to the field of teaching a complex subject, namely, distributed computing in a DEE by determining the factors that contribute to the success of teaching distributed computing and subsequently integrating these factors into a model. The Independent Distributed Learning Model (IDLM) is used for the teaching and learning of distributed computing in a DEE. The complexity lies in the fact that presenting this subject matter requires a *specialised developing environment* to develop *specialised softwar*e considering certain *economic realities.*

The IDLM was found to be valuable because it addressed all the challenges experienced. During the case-study in a one-year experiment, the most pertinent challenges included the creation of a specialised development environment. This environment, which was given to the students upon registration, had to be affordable, understandable and easy to use.

The second challenge was to provide supportive material on a regular basis. These supportive materials were presented in various formats, such as tutorial letters, module webpage announcements and forum discussions. The supportive material contributed to the continuity of the module.

The third challenge was to create the student's initial environment and to get the student to write his/her first application. This challenge was due to the various platforms that exist. Since the forum proved to be the single most useful communication medium between students and teacher and students and students, it was used to effectively and efficiently address this challenge. It had to be monitored on a regular basis, especially when assignments or the project was due, which placed a strain on the available resources. However, without the discussion forum, the students felt that the successful completion of distributed computing in a DEE would not have been possible.

VIII. CONCLUSION

The IDLM presented enables teachers within the field of distributed computing and specifically in distanced-based education to present complex subject matter in such a way that students can work *asynchronously* in their own *space*, at their *own time* and *pace*. Therefore, use of IDLM in presenting courses of this nature in this way enabled students to master the complexity.

During use of the suggested model in a one-year experiment, the challenges were the creation of a specialised developing environment, provision of supportive material on a regular basis, initial creation of the student's working environment and the significant value of the forum. All of the above were addressed by the IDLM. Therefore, the use of the proposed model in presenting distributed computing in a distance-based educational environment contributed to the mastering of a advanced and complex course in distributed computing.

References

[1] Condi, S. "Distributed Computing, tomorrow's panacea - An introduction to current technology" *BT Technol Journal* , pp. 13 – 23, April 1999.

[2] Baker, S. *CORBA Distributed Objects Using Orbix.* Addison Wesley Longman, Inc. 1997.

[3] Balen, H. *Distributed Object Architectures with CORBA.* Cambridge University Press. 2001.

[4] Rock-Evans, R. "Component Architectures." *Enterprise Middleware*, 7 – 20, 1999.

[5] Benns, S. B. "What's in the middle?" *BT Technol Journal , 17* (2), pp. 32 – 52, 1999.

[6] IEEE-CS, A. A. *Computing Curricula 2005The Overview Report.* www.acm.org/education/curric_vols/CC**2005**-March06Final.pdf 2005.

[7] Xinoglos, S. "An Integrated Programming Environment for Teaching the Object-Oriented Programming Paradigm." *EurAsia-ICT, LNCS 2510 EurAsia-ICT, LNCS 2510*, pp. 544- 551, 2002.

[8] Kelleher, C. (2005, 37(2)). Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys* , pp. 83-137.

[9] El-Khouly, M. F. Expert tutoring system for teaching computer programming languages. *Expert Systems with Applications* (18), pp. 27-32. 2002

[10] Hartley, J. S. "Towards More Intelligent Teaching Systems." *International Journal of Man-Machine Studies , 5* (2), pp. 215- 236, 1973.

[11] Schreiner, W. A. " Java Toolkit for teaching Distributed Algorithms. *ItiCSE'02,* 2002.

[12] Le Roux, P. *Towards a model for teaching distributed computing in a distance-based educational environment.* MSc UNISA, 2009.