Simulation for Computer Sciences Education

Mouhib Alnoukari[1], Moutasem Shafaamry[2] , Kinaz Aytouni[3]

[1] Syrian Virtual University,
Damascus, Syria
mnoukari@scs-net.org

[2] Syrian Virtual University,
Damascus, Syria
m.shafaamry@gmail.com

[3] Syrian Virtual University,
Damascus, Syria
kinazaytouni@gmail.com

Abstract

Computer Simulation is a powerful technique for education in many different fields. This paper will cover the use of Computer Simulation tools for Computer Sciences education. The paper will try to answer the following question: Are the simulation tools essential in Computer Sciences education or they are just a support method for helping the education process? Our paper will provide more details about using simulation for computer sciences disciplines such as software engineering. The paper will highlight the importance of using simulation tools by adding more insight into aspects of computer sciences functions that cannot easily observed in traditional laboratories.

Keywords: *Computer Simulation, Education, Simulation Tools.*

1. Introduction

Simulation is a powerful technique for systems representations; because it provides a concise way for knowledge encapsulation. Simulation can be used effectively supporting managers in decision making, especially in situations characterized by uncertainty [35].

Morgan and Jones define simulation as "Responsive, provides opportunities to see effects of one's action. Provides some feedback and may develop some intuitive understanding. Some choice and control by student." [23].

Wikipedia defines education as "the learning of knowledge, information, values and skills during the course of life. The simulator will be more than a "living" textbook; it will become an integral part of the practice of higher engineering education."

The main objective of engineering education is to provide more qualified engineer candidates to meet the market and society demand. The engineering practice is a key phase in engineering education. To some extent, we can say that the quality of higher engineering education depends largely on the quality and level of engineering practice.

Many simulation tools were developed to help students, teachers, and wide range of computer experts in many different areas including: software engineering, software project management, computer hardware and architecture, networking, telecommunications, and others.

The importance of using simulation tools in computer networking courses enhances two different education directions. First, teaching computer network concepts is hardly possible without having specialized network laboratories or some other tools suitable for such courses. These laboratories are very expensive to exist in universities especially in poor countries. Secondly, are they flexible enough to be appropriate for a variety of network topologies?

Plus the fast evolution of computer networks concepts makes teaching and learning becoming more difficult task. So, it is necessary for universities and colleges whom have such laboratories, to invest more in order to upgrade these laboratories. Finally, it will be important for students to experiment different scenarios within different environments when designing a network. Simulation tools that can be used in this area include: OPNET, ns, GTNets, Cnet, and others [16, 25, 27].

Hardware simulation in education is also an essential area which allows students to understand computer organizations and architecture in an effective way. As this subject is usually tough, textbooks and traditional experiments do not provide enough experiences. The use of simulation can not only replace experiments, but also can add more insight into aspects of hardware functions that cannot easily be observed in traditional labs. Many simulation tools and languages were developed to help students dig more experiences in this field, such as, HARD, WebMIPS, ESCAPE, CIM++, VisSim, MipsIt, and others [8, 10].

Software engineering simulation tools help students to map software engineering concepts through real-world software projects. Most of undergraduate software engineering courses are taught by presenting concepts and methodologies with small project groups (three to four students). This does not help students face real problems of corporate project environment. Different simulation tools were developed in order to help students put software engineering concepts and methodologies into practice in an associated class project. SimSE is one of these tools that can train students in situations that require understanding and handling of software engineering processes issues [24].

Software project management was one of the main areas where simulation techniques were applied in the domain of software engineering. Software project simulation can help managers to take decisions about a project as they have to consider great number of variables (including quality, effort, duration and cost), and relations between them. Software project simulation can provide managers with a powerful tool that enable them to try different policies, and take decisions according to results obtained. Different type of analysis can be done using simulation at different stages (a priori analysis, project monitoring, and post-mortem analysis). Different simulation tools were developed to help managers enhance their abilities in managing software projects such as Software Project Simulator (SPS),

Computer-Based Training (CBT). Simulation tools can provide students not only with technology-based skills, but, also a basic understanding of typical phenomena occurring during real-life software projects [36].

Simulation tools for communications education support students to understand most of communications materials they find abstract and dry in traditional course environment. Simulation tools such as NS2, RFCL, and others would make these materials seem more concrete, provide students with high level expertise, and make them more motivated to tackle the difficult theoretical and practical principles in the traditional courses [4, 19].

In this paper we will tackle all the previous concepts in more explained and detailed way. It will show how simulation improve education in Computer Sciences different fields, including software engineering, computer architecture and hardware, computer networks, and communications. It will also explain the main features of the simulation tools developed in these areas, and provide the role they are playing to enhance Computer Sciences education.

The next section will explain how simulation tools could be used for computer networks education. Many simulators are provided for networks concepts understanding including: LAN/WAN/MAN network architectures, networks protocols such as: TCP/IP, modeling servers and routers, etc.

Then, we will explain the use of simulators for computer architecture education. Such tools help students to become more familiar with computer architecture basics, without being faced with the complexity of the realistic microprocessor architecture.

In the last section, we will explain how we can use simulators for software engineering education. Such simulators help students to understand future software development jobs and consequent exposures to the software process.

2. Simulation for Computer Networks Education

Teaching computer network concepts, protocols, design and applications at colleges and universities is hardly possible without having specialized network laboratories, but providing such specialized laboratories at universities has many disadvantages and obstacles : First, they are very expensive. It is also necessary to invest a great deal of money to upgrade such laboratories. In addition to that, such laboratories are not flexible because they are not appropriate for a variety of network topologies. And above all, those laboratories can be used by only a small number of students at the same time, because of the strictly physical limitation.

The question is: how can colleges and universities solve these problems and provide better solutions to improve the practical skills for their students? Fortunately nowadays, there are many emerging technologies; one particular technology appears to be very suitable for improving teaching practical computer networks, is network modeling & simulation (NMS) technology [14], which is very appealing for creating virtual laboratories for computer network courses. This approach may be very useful, because it is efficient way to simulate both small and large networks with different technologies and topologies as well.

Network modeling & simulation (NMS) technology provides network simulators. Using network simulators in virtual network laboratories add a new dimension to textbook theory when integrated into the curriculum. They provide a high fidelity software models that simulates the behavior of a real-world network. By changing the configuration, link capacity, traffic volumes, and characteristics of this virtual network model, professors and students can accurately predict the impact of these changes on the real network. This capability enables a broad range of studies including:

- Understanding LAN/WAN/MAN network architectures
- Visualizing TCP/IP mechanisms and variations
- Modeling server and router availability
- Studying various wired and wireless routing protocols
- Designing reliable wireless networks
- Implementing efficient network security
- Testing “What ifs” cases, or gaining insight on "unusual" network traffic.
- Understanding computer network management concepts and protocols

For educational purposes, the virtual network components made available by the tool are sometimes more effective than the physical devices they simulate since the tool can visualize certain logical networking abstractions that do not actually exist in networking hardware found in real life. For example, the Open Systems Interconnect (OSI) seven layers reference model, or Transmission Control Protocol/Internet Protocol (TCP/IP) reference model is difficult for students to associate with a hardware unit like desktop or laptop computers. In some network simulation tools, such as, the OPNET simulation tool [27], students can easily see all the layers of the reference models and data flows including the addition and deletion of headers and trailers when a packet is moving through the different network layers made visible by double clicking on a picture icon representing a host.

Using simulation tools in virtual laboratories provide models for a detail understanding and in-depth analysis of items such as building complex networks from basic building blocks with variety of nodes and links, packet flows, buffer overflow and operating system compromise. Their capabilities have been growing allowing the creation of hypothetical scenarios down to the bit level. They could be utilized for a variety of tasks.

Some simulation software provide a graphical interfaces and libraries tools such as graphical presentation of network node and a pair of arrows going in and out of each node. The arrows in one color indicate an incoming traffic. The box at the very bottom is a symbol used for the physical layer consisting of media (such as network interface cards or fiber optic cables) carrying electronic signals in their raw form. The boxes above the physical layer, labeled as MAC/ARP represent the data link layer. The transition from electronic signals to bits representing frames is visible by clicking on the colored (red) arrow going out of the MAC/ARP rectangle [27]. Students can inspect all the details of the frame such as headers and trailers. Considering that the idea of layers is a highly abstract concept, having access to tangible layers and traffics going through them greatly enhance students' manipulability. When students inspect the details of data coming in and going out to the next layer (network layer, labeled as IP in the rectangle), they realize that the headers and trailers of the frames have now been

removed, and an IP packet has been exposed. Through these exercises students can easily understand the encapsulation and decapsulation concept (i.e., removing headers and trailers of a certain layer for processing in the next layer).

Typical network simulators encompass a wide range of networking technologies and help the students and users to build complex networks. With the help of simulators student can design different network topologies and hierarchical networks using various types of nodes like computers, hubs, bridges, routers, optical cross-connects, multicast routers, mobile units, MSAUs etc.

The virtual laboratory is planned to be used in a computer network course for teaching IP addressing, static and dynamic routing (RIP and OSPF) [4], firewall concepts and for teaching network services (Web and FTP) and features of some of secure network protocols such as Secure e-mail and IPSec [17,29]. It has to be low-cost and easy-to-use solution.

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive [28]. They allow students and engineers to test scenarios that might be particularly difficult or expensive to emulate using real hardware- for instance, simulating the effects of a sudden burst in traffic or a DoS attack on a network service.

Networking simulators are particularly useful in allowing designers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. Others developed some interactive games with simulation to make education techniques more attractive, such as CyberCIEGE game [13]. CyberCIEGE is an interactive tool for assurance training and education allowing players (students or designers) to select network components and tools to design a secure network then verify the level of security, or challenging the other player in finding security breaches and gaps.

2.1 Types of network simulators

There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle network traffic. Graphical applications allow users to easily visualize the workings of their simulated environment. Text-based applications may provide a less intuitive interface, but may permit more advanced forms of customization. Others, such as GTNets, are programming-oriented, providing a programming framework that the user customizes to create an application that simulates the networking environment to be tested.

Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP, etc and Local Area Network (LAN) technologies like Ethernet, token rings etc. can all be simulated with a typical simulator and the user can test, and analyze various standard results apart from devising some novel protocol or strategy for routing etc.
There are several network simulation tools such as OPNET [27], Network Simulator NS–2 (NS), GTNets [16], or Cnet [12]. The only problem with such type of software is the fact that students are not

completely involved in real configuration of network equipment and networks and they don't have enough operating experience.

2.1.1 OPNET, Network Simulation Tools Suit for Education:
OPNET's suite of products combine predictive modeling and a comprehensive understanding of networking technologies to enable customers to design, deploy, and manage network infrastructure, network equipment, and networked applications. In particular OPNET Modeler is a development environment, because it allowing you to design and study communication networks, devices, protocols, and applications

OPNET Modeler is a leading industry and academic solution for modeling and simulating communications networks, devices, and protocols with many features and flexibility. This tool uses an object-oriented modeling approach and provides simulated functionalities and graphical representations of real-life network components [27].

By providing specialized editors, elaborate analysis tools, and off-the-shelf models, it allows highly sophisticated analysis of production networks for enhancing their performance. Because of its open design, new protocols and related approaches can easily be incorporated, simulated, and tested.

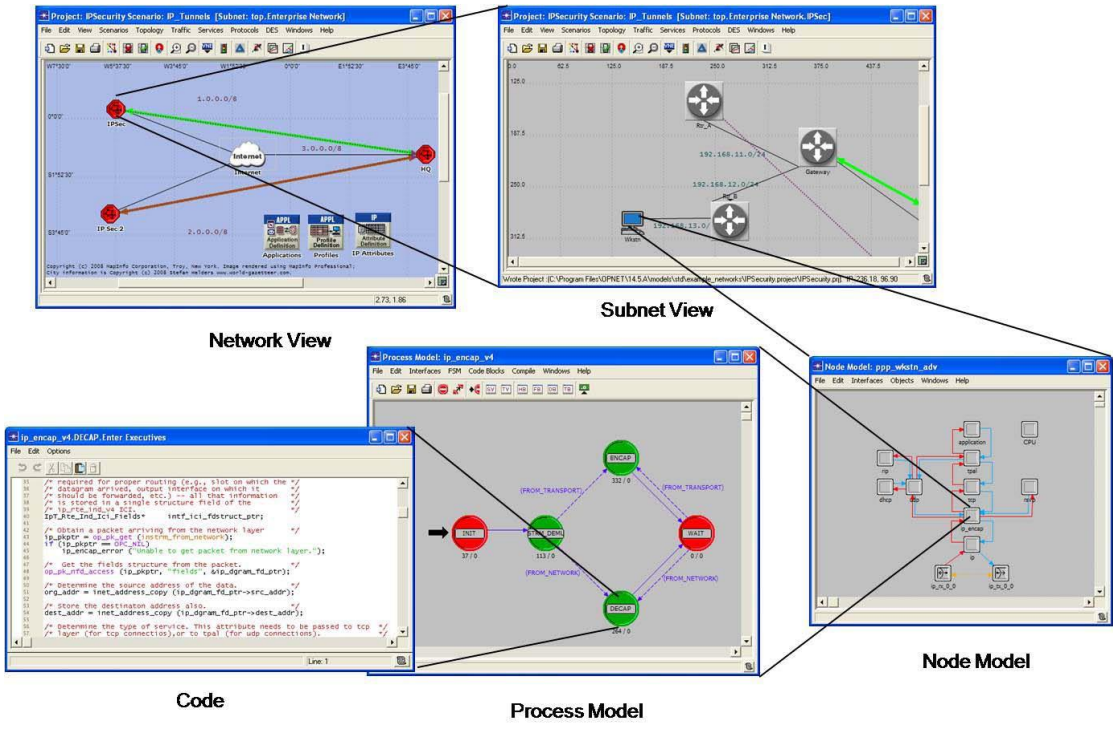

Fig. 1 OPNET network editor views and windows

The network editor graphically represents the topology of a communications network that contains nodes and links between them. The node editor shows the internal architecture of each node by displaying the data flow between different functional network layers called modules. These modules

perform different network functions within each node. A module can send or receive data to or from other modules.

The process editor allows users to specify the behaviors of a module and uses finite state machines to describe its protocol details.

Figure 1 shows different OPNET views provided by the network editor, the node editor, and the process editor.

2.1.2 ns2/ns3, Free, Open Source Network Simulation tools:
“ns” or the network simulator (also popularly called ns-2, in reference to its current generation) is a discrete event network simulator. It is popular in academia for its extensibility (due to its open source model) and plentiful online documentation. “ns” is popularly used in the simulation of routing and multicast protocols, among others, and is heavily used in ad-hoc research. “ns” supports an array of popular network protocols, offering simulation results for wired and wireless networks alike. It can be also used as limited-functionality network emulator. “ns” is licensed for use under version 2 of the GNU General Public License [19].

“ns” was built in C++ and provides a simulation interface through OTcl, an object-oriented dialect of Tcl (originally from "Tool Command Language", but nonetheless conventionally rendered as "Tcl" rather than "TCL"; pronounced as "tickle" or "tee-cee-ell", is a scripting language created by John Ousterhout). The user describes a network topology by writing OTcl scripts, and then the main ns program simulates that topology with specified parameters (Figure 2).

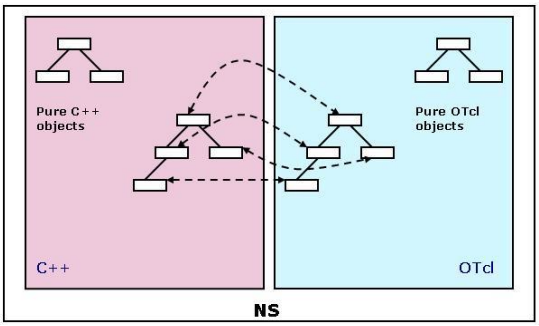

Fig. 2 The design structure of ns simulation tool

"ns" was development in 1989 as a variant of the REAL network simulator. By 1995, ns had gained support from DARPA, the VINT (Virtual Inter Network Testbed) project at LBNL (The Ernest Orlando Lawrence Berkeley National Laboratory), Xerox PARC, UCB (University of California, Berkeley), and USC/ISI (The Information Sciences Institute (ISI) of the University of Southern California).

"ns" is now developed in collaboration between a number of different researchers and institutions, including SAMAN (supported by DARPA), CONSER (Collaborative Simulation for Education and Research)(through the NSF), and ICIR (formerly ACIRI). It is currently maintained by volunteers. Long-running contributions have also come from Sun Microsystems and the UCB Daedelus and Carnegie Mellon Monarch projects, cited by the ns homepage for wireless code additions (ns).
Generation 3 of ns (ns3) has begun development in July 1, 2006 and it is projected to take four years. It is funded by the institutes like University of Washington, Georgia Institute of Technology and the ICSI Center for Internet Research with collaborative support from the planet research group at INRIA Sophia-Antipolis. Currently ns-3 is in development phase. It is an event based network simulator

2.1.3 Cnet Simulation Tool:
Cnet is a simulator of computer networks developed by Chris McDonald at University of Western Australia. It has been specifically developed, and used in, undergraduate computer networking courses taken by thousands of students worldwide. This simulator is a discrete-event network simulator enabling experimentation with various data-link layer, network layer, routing, and transport layer networking protocols [12].

Cnet is a network simulator which enables experimentation with various data-link layer, network layer, routing and transport layer networking protocols in networks consisting of any combination of point-to-point links and IEEE 802.3 Ethernet segments. With reference to the OSI/ISO Networking Reference Model, Cnet provides the Application and Physical layers. User-written protocols are required to "fill-in" any necessary internal layers and, in particular, to overcome the corrupted and lost frames that Cnet's Physical Layer randomly introduces. In addition, advanced users may develop different Application and Physical Layers which exhibit varying statistical characteristics of message generation and data transmission. Simulation sizes may range from two to a few hundred nodes (Figure 3).

Cnet either displays the entire network under Tcl/Tk or runs rather less visually on an ASCII terminal. Under Tcl/Tk, Cnet provides a graphical representation of the network under execution and permits a

number of attributes of the network to be modified while the simulation is running. Nodes may be selected with the mouse to reveal a sub-window displaying the output and protocol statistics of that node. Some of the node's attributes, such as message generation rates and sizes, may be modified while the network is running by selecting choice buttons. Similarly, the default attributes of all nodes in the network may be simultaneously modified by selecting and changing global attributes. From another menu, each node may be forced to reboot, crash, shutdown and reboot, pause and (hardware) fail.

Cnet runs on a variety of UNIX and Linux platforms. It does not run on either Windows or the Apple Macintosh.

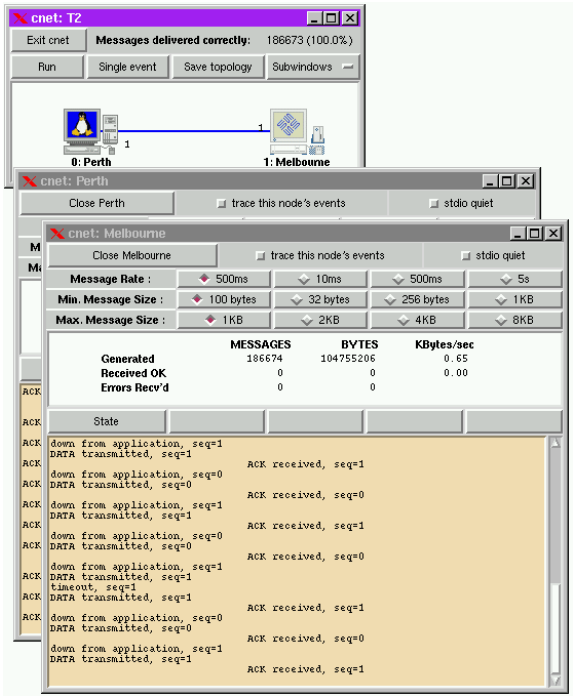

Fig. 3 Some Graphical interfaces of Cnet tools

2.1.4 GTNetS, Georgia Tech Network Simulator:

The Georgia Tech Network Simulator (GTNetS), developed by George Riley, is a full-featured network simulation environment that allows researchers and students in computer networks to study the behavior of moderate to large scale networks, under a variety of conditions. The design philosophy of GTNetS is to create a simulation environment that is structured much like actual networks are structured. For example, in GTNetS, there is clear and distinct separation of protocol stack layers [16]. Packets in GTNetS consist of a list of protocol data units (PDUs) that are appended and removed from the packet as it moves down and up the protocol stack. Simulation objects representing network nodes have one or more Interfaces, each of which can have an associated IP address and an associated link. Layer 4 protocol objects in GTNetS are bound to ports, in a fashion nearly identical to the binding to ports in real network protocols. Connections between protocol objects at the transport layer are specified using a source IP, source port, destination IP, and destination port tuple just like actual TCP connections. The interface between applications and transport protocols uses the familiar connect, listen, send, and send to calls much like the ubiquitous sockets API in Unix environments.

Applications in GTNetS can have one or more associated protocol objects, and can simulate the flow of data (including actual data contents) between applications.
JiST/SWANS, a Simulation Tool for Wireless Networks:

2.1.5 JiST/SWANS Simulation Tool:
Java in Simulation Time/Scalable Wireless Ad hoc Network Simulator (JiST/SWANS) is developed by Rimon Barr at Cornell University [31].

JiST is a high-performance discrete event simulation engine that runs over a standard Java virtual machine. It is a prototype of a new general-purpose approach of building discrete event simulators, called virtual machine-based simulation that unifies the traditional systems and language-based simulator designs.

SWANS is a scalable wireless network simulator built atop the JiST platform. SWANS is organized as independent software components that can be composed to form complete wireless network or sensor network configurations. Its capabilities are similar to ns2 and GloMoSim, but they are able to simulate much larger networks. SWANS leverages the JiST design to achieve high simulation throughput, save memory, and run standard Java network applications over simulated networks. In addition, SWANS implements a data structure, called hierarchical binning, for efficient computation of signal propagation [31].

3. Simulation for Computer Architecture Education

Computer architecture is considered as one of the most complicated subject in computer sciences education. Most of undergraduate students fail to understand computer architecture basics, including microprogrammed architectures, pipelined execution, and branch prediction [10]. Students should fully understand these topics before addressing more advanced issues of the memory hierarchy such as, caching, virtual memory, register optimization, and others.

Simulation environments help students understanding these complex topics by providing them an easy to understand tools, allowing them to become more familiar with computer architecture basics, without being faced with the complexity of the realistic microprocessor architecture.

Simulation tools help computer architect designing computer hardware logically based on current technology and applications [11]. The logical design includes: datapath, control unit, memory, and input/output unites at the abstract level instead of circuit level. Simulating hardware design help in testing the design before chip fabrication to verify if the design is working properly or not. Chu used DCT (Design, Code, and Test) procedure to let his undergraduate students easily understand the concept of computer architecture, and design a simple RISC-Type 32 bit Instruction Set Architecture (ISA) [11]. The result of his DCT education simulation project at Mississippi State University was successful since 78.9% of the students evaluate the project as favorable as they could learn fundamental concepts and the design process clearly and gain confidence in the area of computer architecture.

Many simulation tools were developed to enhance the way of teaching computer architecture education. The following tools are used in different universities around the world to help teaching computer organization and architecture courses.

ESCAPE is one of such tools developed to help teaching course on computer architecture in the computer science and computer engineering under-graduate curriculum of the University of Ghent, Belgium [10].

DARC2 is a DLX architecture pipeline simulator used as a supplementary tool for computer architecture education at De La Salle University, Philippines [33]. DARC2 helps students better understand the flow of data through the architecture, and how each instruction is executed. Simple CPU is a simulator of the internal architecture and operations of the CPU used in the University of Oviedo, Spain as a simulation tool for the first-year students of Computer Science to let them easily understand the basics of computer architecture [3]. Simple CPU effectively allows students to assimilate the internal architecture of the CPU, instructions sequences, operations of the data-path and architecture, etc. in only 20 course hours.

SIM-PL is used at the University of Amsterdam for lab assignment at courses "Architecture and Computer Organization", and "Digital Electronics" [9]. SIM-PL helps students at higher education, and secondary schools in the Netherlands, getting more knowledge about the main components of the calculator, the functions of these components and the relations between them, instruction set and instruction format, the function of the clock, etc. [1] presented the use of simulation and Field Programmable Gate Array (FPGA) technology for teaching computer organization and architecture. The teaching tool has been tested by the 3rd year students enrolled at the computer architecture course at Philadelphia-Jordan University. This tool is able to help students experience their own instruction set, computer programming, and interfacing experiments.

Teachers at the University of Siena, Italy are using a web-based simulator named WebMIPS for the introductory computer architecture course [8]. The main advantage of this tool is that it doesn't need installation; students are able to access it successfully through the Web. Teachers are able to monitor students' activities. WebMIPS main activity is to upload and assemble MIPS code, simulating a five-stage pipeline, and displaying the values of all registers and other pipeline components.

Most of these simulation environments are characterized by the ability to provide valuable insight on the internal operation of the processor, customizable architectures configuration including: size of the register file, number of temporary registers, and memory access time.

Simulation environment for computer architecture education should be characterized by an easy-to-use, interactive interface to help students obtain valuable insights in more complex matters, such as content of registers, memory content, and cache behavior. They should also have visual representation of architectures components such as multiplexers, and bus behavior in order to help students understand their mechanisms. Finally, simulation environments should provide the following characteristics to clarify the way architectures operate: trace generation, pipeline activity and pipeline usage diagrams, cycle-per-cycle or multi-cycle simulation with breakpoints, customizable simulation speed, memory monitor, and microcode editor.

Usage of simulation environment for computer architecture education can help students acquire the following knowledge [1, 3, 8, 9, 10]:

- Learn the basics of microprogramming including the basic synchronous operations at the register transfer level of a datapath (ESCAPE, DARC2, SIM-PL, WebMIPS).
- Learn more about microprogram optimization techniques (WebMIPS).
- Learn the basics of pipelined architecture (ESCAPE, DARC2, SIM-PL).
- Learn more about code optimization techniques including: code motion, register renaming, and software pipelining (WebMIPS, DARC2).
- Learn more about some advanced computer architecture concepts such as superscalar architecture (DARC2).
- Collect quantitative data such as memory speed, code size, and benchmark programs (DARC2).

3.1 SATSim, A Superscalar Architecture Trace Simulator

SATSim is an interactive simulator tool developed in Georgia Institute of Technology for advanced undergraduate computer architecture course, especially the complicated behavioral patterns of superscalar architecture. SATSim allows students to interactively change the parameters of the hardware configuration and observe their effects using a cost effective visual manner [34].

SATSim provides an animated tool for teaching superscalar architecture concepts including: out-of-order execution, in-order commitment, dynamic resolution of data dependencies, and the performance effects of branch prediction accuracy and cache hit rates.

SATSim starts by reading instructions from a trace file, and then it displays them progressively using a simulated microarchitecture (Figure 4). The tool allows users to configure the microarchitecture by selecting some factors parameters.

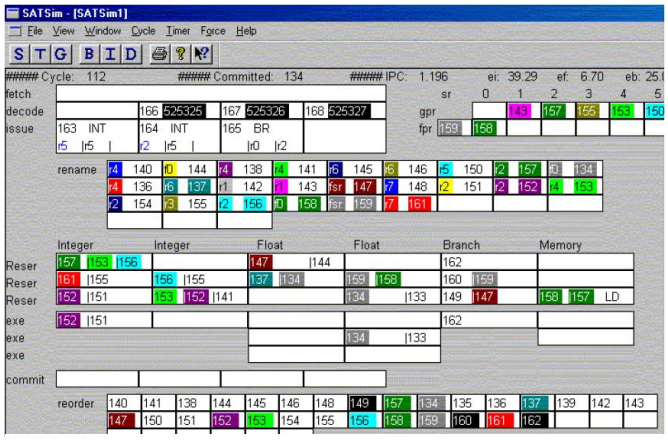

Fig. 4 Screenshot of SATSim animation (Wolff, and Wills, 2000).

SATSim contains associated assignments to help students understanding the advanced topics of superscalar architecture. Such assignments include: the effects of branch prediction on accuracy and cache hit rates on the performance of superscalar architectures, explore the design space by changing architecture parameters, and analysis of design tradeoff using the achieved data.

4. Simulation for Software Engineering Education

Software systems are very complex to build and construct, in addition to the fact that they rely on craftsmanship individual. Systems requirements are divided into functional and non-functional requirements. Functional requirements are to meet system needs within budget and time .The so-called nonfunctional requirements include properties that systems are supposed to display with respect to security, availability, reliability, and performance.

The need to build software systems that function correctly, hold on to non-functional requirements, and are cost-effective, gave rise to a series of techniques and methods, patterned on what engineers do, when they design systems. This discipline, called Software Engineering (SE). According to Barry Boehm, "Software Engineering is the means by which we attempt to produce all of this software in a way that is both cost-effective and reliable enough to deserve our trust.", [7].
Software Engineering is labor intensive:

The supposition is that a more educated and skilled team will produce higher quality in their software product and artifacts. By using data from 26 small-scale software development projects, Beaver and Schiavone studied the relationship between development team skill and the quality of software engineering product [5]. They found that development team skill has a significant effect on the quality of a software product. And the experience of the software development team is a driving factor in software product quality.

Menasc found that software systems rarely meet their performance requirements [22]. He mentioned five issues for the cause, one of them the lack of training in performance issues when we teach software engineers. If the software team stresses quality in all software engineering activities, it reduces the amount of rework that it must do. That results in lower cost, and more importantly, in improving time to market.

Fairley discussed a number of educational issues related to software engineering [15]. He empathized that teaching basic concepts and fundamental principles are essential for software engineering education but not enough to a graduate software engineer to work in a productive way. Also, there is not an educational environment that provides realistic experience in the university environment, especially for communications with user/customer, and the consideration of quality issues.

So what we need is build programs that produce knowledgeable, skilled software engineering graduates. And the question is how to build such programs?

Skills are learned through experiences “Experience is the best teacher.”

While it is impossible to teach experience, it is possible to teach through experience. This observation has guided the development of many software engineering courses being taught today.

Learning through experiences by traditional teaching techniques that use presentations with questions and answers, the “students behavior are almost always passive they are only listening to the instructor" [26]. The instructor attitude is controlling, it causes a gap among students, and among students and the instructor, making the learning process difficult.

PBL (Project Based Learning) classes are effective in improving problem solving and communication abilities of students [21]. Although they are effective, they still have constrains in time and scope. Both lectures and projects are aimed at most effectively introducing students to the software process as experienced in real-world software engineering projects, but not practicing them.

Shaw wishes to provide effective means to the student to build the needed skills for independent lifelong learning [30].

In order to improve teaching techniques to become more efficient and effective, instructors should help students execute the process themselves, and let them learn from practice.

Oh sets out that using graphical interactive software engineering simulation environment can effectively improve student’s learning of the software engineering process by providing a complete, realistic, and practical experience of the concepts taught in lectures without any of the drawbacks involved in a typical class project [26].

Iglesias and Paniagua consider that the process simulator is “the most powerful tool to tackle matters of great complexity, even though some limitations in the facilities" [18]. Therefore using simulation tools will improve student's understanding of complex theory and concepts.

Using simulation environment in teaching process has many advantages. First it could provide more practice chances for the student to adopt, which means conduct different alternatives to enhance student’s management skills. Second it could provide students with more advanced study conditions; finally it could provide students with the “self- study” setting, which builds the needed skills for independent lifelong learning.

4.1 SimSE, a Simulation tool for Software Engineering Education

To better prepare students for their future software development jobs and consequent exposures to the software process, SimSE represents a new approach to teaching and practicing the software process that is complementary to existing software engineering courses [24].

SimSE is a computer-based environment that allows the creation and simulation of software engineering processes. It allows students to virtually participate in a realistic software engineering process that involves real-world components not present in typical class projects, such as teams of people, large scale projects, critical decision-making, personnel issues, multiple stakeholders, budgets, planning, and random, unexpected events. In so doing, it provides students with a platform in which

they can experience many different aspects of the software process in a practical manner without the overarching emphasis on creating deliverables.

SimSE directly addresses the shortcomings of existing classroom approaches by filling the gap between the process knowledge and theory taught in lectures and the small percentage of this knowledge that students actually put into practice in class projects.

SimSE is a single-player game in which the player takes on the role of project manager of a team of developers. The player must manage these employees to complete a particular (aspect of a) software engineering project. Management activities include, among others, hiring and firing, assigning tasks, monitoring progress, and purchasing tools. In general, following good software engineering practices will lead to positive results while blithely ignoring these practices will lead to miserable failure in completing the project.

SimSE basic architecture contains three main components: models that encapsulates a set of cause and effect rules and is used to drive the simulation with particular scenarios, a graphical user interface (GUI) through which the user interacts with the simulation tool and views the visual effects of the simulation, and a simulation engine that executes the model.

The architecture of SimSE is similar to any programming language. Simulation models represent the concept of process in the same manner as any programming languages which represent the concept of programming methods. SimSE GUI is similar to any programming editor, and finally a simulation engine is similar to programming compiler. The similarity between the simulation tool environment and the programming language environment make the practices easy to students.

SimSE simulation environment is responsible for executing a simulation and creating the user experiences that demonstrate the inherent complexity of the software process. The simulation loop itself operates in the following manner (Figure 5): The clock drives the simulation by emitting ticks at equal time intervals. At every clock tick, the rule execution component checks which actions are currently executing by querying the state management component. It then executes the rules associated with these actions, and in turn propagates the effects of these rules on the entities and actions in the state management component. After this update is completed, the clock then signals the user interface to update the display. The user interface then queries the state management component and updates itself to reflect the current state.

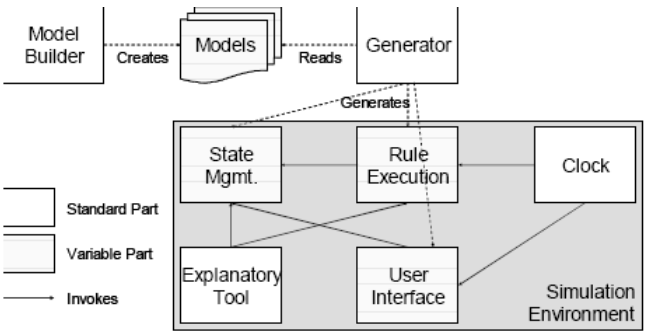

Fig. 5 SimSE Architecture [24].

Software engineering courses always follow a set of concepts lead to process which encompasses problem definition, requirements analysis, design techniques and then programming methodology.

All of the software processes cover the following three main phases; each of them needs different types of skills and experiences to achieve success.

First, requirement engineering phase focus on systematic and interpersonal skills such as interacting with customers, eliciting needs from those customers, and classifying the needs into a set of requirements, then review these requirements and analyze them. In 1976, Barry Boehm put forth the following definition of software requirements engineering:

"Software requirements engineering are the discipline for developing a complete, consistent, unambiguous specification- which can serve as a basis for common agreement among all parties concerned - describing what the software product will do (but not how it will do it; this is to be done in the design specification).", [7].

To conduct requirement engineering successfully there must be a skilled development team equipped with communication skills, and management skills to manage the expectation of changes in requirements, conduct effective communication focused on the need to identify, control and track requirements. This can be achieved by conducting meaningful reviews with stakeholders.

By using the simulation tool SimSE, students will gain skills that are needed for requirements engineering [26]. For instance, after eliciting requirements, and to determine whether the requirements are for successful models, and then building a base set of models that meet these requirements, a student could navigate through the simulated model with the goal of ensuring that the task is completed on time, and follows the standard interaction procedures that typically characterize the inspection process. This can be done using the simulation engine that executes the chosen model step-by-step. In particular, it takes the current state of the simulation, the model, and any relevant user input, in order to calculate the new state of the simulation. It then provides this information to the encompassing simulation environment, which in turn graphically displays the result.

Second, the design engineering phase which is the core of software engineering requires the ability to create a software solution that meets the identified requirements, and the skills in evaluating the set of approaches and tools to implement the needed solution. Design is the place where software quality is established. SimSE provides a modeling language to allow educators to build custom models teaching them the particular software engineering issues or processes they wish to highlight. The modeling language provides facilities to express features such as: the goal to be achieved by the user, the setting in which the simulation takes place, the underlying rules of the simulated process, the visual effects to be displayed, and the available user commands.

SimSE modeling language is very similar to the Unified Modeling Language (UML), Modeling constructs are denoted as rectangles, with the name of the construct in bold in the top part of the rectangle and its attributes listed below. Attributes that are either optional or only present for certain types of that particular construct are shown in parentheses. Relationships between two constructs are

indicated by an arrow drawn between them, and each arrow is labeled with the type of relationship it represents. The cardinality of a relationship is specified at each end of the corresponding arrow.

Third, the implementation phase that demand a working knowledge and skills with the specific development tools including: languages, operating systems, libraries, and development environment.

Bloom suggests that in order to provide better understanding of conceptual theory, the engineering course must include the following objectives [6]:

1. Knowledge: by presenting fundamentals, principals and theory.
2. Comprehension: by analyzing problems and implementations.
3. Application and analysis: by presenting real world problems.
4. Synthesis: by presenting small consent problems.
5. Evaluation: by balancing between internal evidence and external criteria.

In the PBL classes, the theoretical concepts of real engineering problems are presented and simulation software is used to clarify the problem. The main objective is to teach the student how to move toward the problem and analyze the results. In order to understand the engineering feature of the theoretical concept, the example must be a real world problem, with emphasis on design.

In summary, simulation should not replace the existing educational techniques, but rather, serve as a complementary role.

5. Conclusion

In this paper, we clarified the use of simulation tools in Computer Sciences courses. Simulation tools can help students elaborate many of the complex computer science and engineering topics, such as microprogrammed architectures, pipelined execution, and branch prediction in computer architecture basics courses [10]. Communications educational courses are also abstract and very difficult for undergraduate students. Simulation tools are very effective in providing students with the way to understand many Communication topics such as EM waves and fields, antennas, and RF components design [2].

In addition, network modeling & simulation (NMS) technology provides virtual network laboratories with very less cost [14]. NMS technology also adds new dimensions to textbook theory when integrated into the curriculum. NMS technology helps students experiment different scenarios within different environments when designing a network. We described some characteristics of networking simulation tools such as: OPNET, ns, GTNets, Cnet, and SWANS.

Our paper also provides more details about using simulation for other computer sciences disciplines such as software engineering.

In summary, simulation tools help colleges and universities solve the previous problems and provide better solutions to improve the practical skills for their students.

Finally the response of the question we raised previously in the paper abstract, are simulation tools essential in computer sciences education or they are only new means to support the education process? We think that the use of simulation cannot replace experiments, but they can add more insight into aspects of computer sciences functions that cannot easily observed in traditional laboratories.

References

[1] K. M. Al-Aubidy, "Teaching Computer Organization and Architecture Using Simulation and FPGA Applications", Journal of Computer Science, Vol. 3, No. 8, 2007, pp. 624-632.

[2] H. M. Al-Rizzo, A. Al-Habsi, M. Haidar, and R. Addada, (2004). "An Interdisciplinary Simulation-Based Laboratory for Undergraduate Wireless Communications Education", The Second IEEE GCC Conference, 2004, Manama, Bahrain.

[3] J. R. Arias, and D. F. Garcia, "Introducing computer architecture education in the first course of computer science career", Proceedings of the 1998 Workshop Computer Architecture Education. 1998, pp. 19-21.

[4] F. Baumgartner, T. Braun, E. Kurt, and A. Weyland, "Virtual Routers: A Tool for Networking Research and Education", ACM SIGCOMM Computer Communications Review, Vol. 33, No. 3, pp. 127-135.

[5] J. M. Beaver, and G. A. Schiavone, (2006), "The Effects of Development Team Skill on Software Product Quality", Proceedings of the ACM SIGSOFT Software Engineering, Vol. 31, No. 3, pp. 1-5.

[6] B. Bloom, Taxonomy of Educational Objectives. Wikipedia, http://en.wikipedia.org/wiki/Taxonomy_of_Educational_Objectives

[7] B. W. Boehm, "Software Engineering", IEEE Transactions on Computers, Vol. 25, No. 12, 1976, pp. 1226-1241.

[8] I. Branovic, R. Giorgi, and E. Martinelli, "WebMIPS: a new web-based MIPS simulation environment for computer architecture education", Proceedings of the 2004 workshop on Computer architecture education, 2004, Munich, Germany.

[9] B. Bruidegom, and W. Koolen-Wijkstra, "SIM-PL: Software for teaching computer hardware at secondary schools in the Netherlands", http://staff.science.uva.nl/~benb/Publicaties/IMICT_SIM-PL.pdf

[10] J. V. Campenhout, P. Verplaetse, and H. Neefs, "ESCAPE: Environment for the Simulation of Computer Architectures for the Purpose of Education", Proceedings of the 1998 Workshop Computer Architecture Education, 1998, pp. 42-47.

[11] Y. Chu, "A Simple Project Paradigm for Teaching Instruction Set Architecture", Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies, Kaohsiung, Taiwan, 2005pp. 69-71.

[12] Cnet, available at: http://www.csse.uwa.edu.au/cnet/

[13] CyberCIEGE, available at: http://cisr.nps.edu/cyberciege/

[14] D. Dobrilovic, and B. Odadžic, "Virtualization Technology as a Tool for Teaching Computer Networks", Proceedings of World Academy of Science, Engineering and Technology, Vol. 13, 2006, pp.126-130

[15] R. E. Fairley, "Educational Issues in Software Engineering", Proceedings of the 1978 ACM annual conference, 1978, pp. 58-62.

[16] GTNets, available at: http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/

[17] J. Hu, C. Meinel, & M. Schmitt, Tele-Lab IT Security: An Architecture for Interactive Lessons for Security Education, SIGCSE'04, Norfolk, Virginia, USA, 2004.

[18] O. A. Iglesias, and C. N. Paniagua, "Using Online Simulation in Teaching Alternative Analysis and Process Optimization", Current Developments in Technology-Assisted Education, 2006, pp. 2075-2080.

[19] T. Issariyakul, and E. Hossain, "Introduction to Network Simulator NS2", Springer, 2008.

[20] M. Magee, "State of Field Review: Simulation in Education", Final Report, Alberta Online Learning Consortium, 2006.

[21] S. Matsuura, (2006). "An Evaluation Method of Project Based Learning on Software Development Experiment", Proceedings of the 37th SIGCSE technical symposium on Computer science education. Houston, Texas, USA, pp.264-265.

[22] D. A. Menasc, “Software, Performance, or Engineering?”, Proceedings of the 3rd international workshop on Software and performance, Rome, Italy, 2002, pp. 239-242.

[23] R. Morgan, and K. O. Jones, (2001). “The Use of Simulation Software to Enhance Student Understanding”, Proceedings of the IEEE International Symposium on Engineering Education: Innovations in Teaching, Learning and Assessment, Vol. 33, No. 1, pp. 6-33.

[24] E. O. Navarro, and A. V. D. Hoek, “SimSE: An Interactive Simulation Game for Software Engineering Education”, Proceedings of the 7th IASTED International Conference on Computers and Advanced Technology in Education, Kauai, Hawaii, 2004.

[25] NS: Network Simulation, available at: http://www.isi.edu/nsnam/ns/ns.html

[26] E. Oh, “Teaching Software Engineering through Simulation”, Proceedings of the International Conference on Software Engineering Doctoral Symposium, Orlando, Florid, 2002.

[27] OPNET, available at: http://www.opnet.com/

[28] J. Potemans, J. Theunis, B. Rodiers, B. Van den Broeck, P. Ley, E. Van Lil, and A. Van de Capelle, Simulation of a Campus Backbone Network, a case-study , OPNETWORK Conference, Washington D.C., 2002.

[29] J. Ryoo, “Teaching IP Encryption and Decryption Using the OPNET Modeling and Simulation Tool”, Proceedings of the 12th Colloquium for Information Systems Security Education CISSE, University of Texas, Dallas, 2008, pp. 113-118.

[30] M. Shaw, (2000). “Software Engineering Education: A Roadmap”. "The Future of Software Engineering", Anthony Finkelstein (Ed.), ACM Press, 2000.

[31] SWANS, available at: http://jist.ece.cornell.edu/index.html

[32] J. Taylor, and S. Tanev, “Photonic Simulation Software Tools for Education, 10th International Topical Meeting on Education & Training in Optics and Photonics”, Proceedings of the 2007 ETOP Conference, Ottawa, Ontario, Canada, 2007.

[33] R. L. Uy, M. Bernardo, and J. Erica, “DARC2: 2nd Generation DLX Architecture Simulator”, Proceedings of the 2004 workshop on Computer architecture education, Munich, Germany, 2004.

[34] M. Wolff, and L. Wills, “SATSim: A Superscalar Architecture Trace Simulator Using Interactive Animation”, Proceedings of the 2000 workshop on Computer architecture education, Vancouver, 2000, pp. 22-26.

[35] M. Alnoukari, A. El Sheikh, and Z. Alzoabi, “Data Mining and Simulation”, In Handbook of Research on Discrete Event Simulation Environments- Technologies And Applications by Evon M. O. Abu-Taieh and Asim A. El Sheikh (eds). IGI Global. 2009.

[36] D. Pfahl, M. Klemm, and G. Ruhe, “Using System Dynamics Simulation Models for Software Project”, Software Process Simulation Modeling Workshop (ProSim2000), London, 10-12 July 2000.