Simulation Tools for Active Learning in Robot Control and Programming

G. López-Nicolás[1], A. Romeo[2], J. J. Guerrero[3]

Informática e Ingeniería de Sistemas Department, University of Zaragoza
Edificio Ada Byron, C/ María de Luna 1, 50018, Zaragoza, Spain

[1]gonlopez@unizar.es
[2]romeo@unizar.es
[3]jguerrer@unizar.es

***Abstract*— Simulation is a powerful tool in different areas of research, development and education. The simulation has also a very important role in robotics. In this paper, we present an active learning experience in two courses of robotics in the framework of the Industrial Engineering degree. The benefits of active learning activities have been widely acknowledged and discussed. In this paper we contribute with a new activity designed in this context. The activity is based on two simulation tools developed for these courses: RobotScene and SGRobot. Here, we describe the methodology and the simulation tools used in a project-based learning activity. These activities have been carried out successfully during several years. Our experience is that this is a motivating activity for the students and it improves the understanding of the theoretical concepts involved. Besides, they are essential tools for developing control and programming abilities.**

I. Introduction

In a broad perspective Industrial Robotics is an inherently cross-disciplinary subject whose in-depth knowledge involves a variety of tasks such as modeling, design, simulation, control, optimization, and performance evaluation. In addition, successful application of industrial robotics involves the integration of various tools from related disciplines. The multidisciplinary issues involved, like mechanics, control, electronics and computer science [1], as well as the complexity of the theoretical concepts required make this discipline difficult for students to get deep understanding with classical teaching methods.

The Bologna process started with the signing in 1999 of the Bologna declaration by Ministers of Education from European countries. The purpose of this process is to create the European higher education area by making academic degree standards and quality assurance standards more comparable and compatible throughout Europe. The new model comes closer to the North American and Japanese systems. It gives greater weight to practical training and to intensive research projects [2], [3], [4]. Thus, the student has to play the leading role of his learning, basing the learning process in active methodologies and student autonomous work to the detriment of traditional lecture methods, in which professors talk and students listen. In essence, students should do more than just listen. They should read, write, discuss, or be engaged in solving problems. Specially, students must engage in such higher-order thinking tasks as analysis, synthesis, and evaluation to be actively involved. Strategies promoting active learning are defined as instructional activities involving students in doing things and thinking about what they are doing [5]. This is the framework of the project-based learning activity presented.

In this paper we describe an active learning experience in the field of Robotics in the context of a degree on Industrial Engineering. Two aspects are mainly treated in a course of Industrial Robotics for an engineering degree: The modeling of the robot for the design of its control system, and the practice to obtain programming skills using specific robot languages. Both complementary learning aspects can be benefited by simulation tools. The robot modeling has a high geometric and mathematical complexity, which is added to the difficulties (time and budget required) for students working with real robots.

For several years we have proposed this instructional activity in which students must model and solve kinematics for a commercial model of robot manipulator, design some aspects of its control system and program the robot in an industrial application. Simulation tools, developed for this specific project, are essential for student's motivation and understanding of the problem, which gives good learning results. The goal of this paper is to present the active learning experience and the simulation tools developed for this project-based course: RobotScene and SGRobot.

RobotScene is a specific software tool that provides a graphical interpretation for all the geometrical concepts involved at the design stage of an industrial robot (solid links, joints, reference frames, Denavit-Hartenberg parameters, etc) [6]. Moreover, RobotScene provides a framework in which programming the previously created robots. It is composed by three modules specialized on solid, robot and scene creation respectively. SGRobot is a graphical simulator for manipulators where robotic tasks can be programmed using a VAL II-based language [7]. There are many robotic platforms that provide simulation frameworks in which users can develop robotic applications [8], [9], but they are not specifically oriented to robot manipulator design. Although, other platforms provide tools useful for robot kinematics and dynamics simulation, as the Robotic Toolbox for Matlab [10] or Spacelib [11], but they have not elaborated graphical interfaces and do not provide robot programming tools. Some projects as OROCOS [12] and ROBOOP [13] allow covering this empty space, but they do not provide an easy graphical interface and so, their use requires important learning efforts. Furthermore, they need an external compiler in order to perform any simulation. In addition, there are simulation platforms as ROBOGUIDE [14] and RobotStudio [15] developed by robot manufacturers in order to provide off-line programming tools specifically designed for their robots. Nevertheless, RobotScene provides modules for creating solids, robots and robotic scenarios in an easy manner. It has been developed using GLScene (a graphic motor for Delphi based on Open GL). The programming capabilities (needed for both inverse kinematics implementation and robot programming) are provided by PascalScript, a Delphi compatible interpreter that adds Pascal-based scripting support to the Robot constructor module and the Scene Constructor.

II. Robot Modeling: Kinematics

A different commercial robot is assigned to each student. The first stage of the learning project is to model and solve the kinematics of the industrial robot. To achieve this, students follows the Denavit-Hartemberg (DH) convention [16], which allows obtaining forward kinematics in a systematic way.

In order to achieve this stage, students trace three steps using two RobotScene modules: First, by using the Solid Constructor Module, students create all the solids that compose the robot manipulator, assigning each it a reference frame compatible with the DH rules. In a second step, they mount the robot by using the Robot Constructor Module, which allows defining all data related to joints (type, range, DH parameters, home value, etc). Furthermore, this module can be used for creating robotic tools as grippers, welding equipment, etc. Once all the joints are created, students can implement the inverse kinematics equations using a specific programming interface over PascalScript and run it in order to validate them.

A. *Modeling the Robot Solids*

Once the students have analysed the morphology of their assigned manipulator, identifying its joints and its solid links, they can begin to model their robot using RobotScene. The first task to be accomplished in order to model a robot manipulator is defining the solids that compose it. The Solid Creator Module provides basic tools for modeling solids in an easy fashion. It contains a library that includes the basic geometries (prisms, cylinders, pyramids, spheres, etc), the composition makes possible the design of complex solids. Moreover, the module contains specific tools for modeling volumes generated by extrusion and revolution, taking into account the relevance of these geometries in most of the manipulators models. Fig. 1 shows an example of complex solid composed by several pieces.

Finally, the module enables saving the created solids to an ASCII file that contains all the solid attributes, enabling in this form its edition and modification (with or without RobotScene).

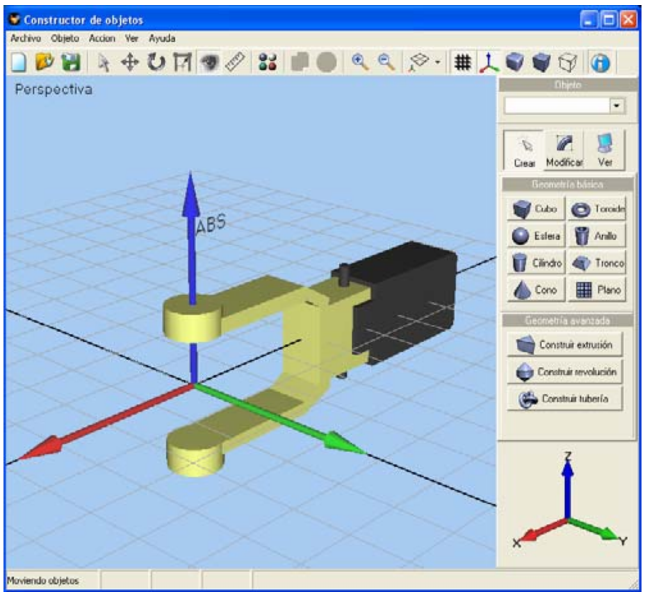
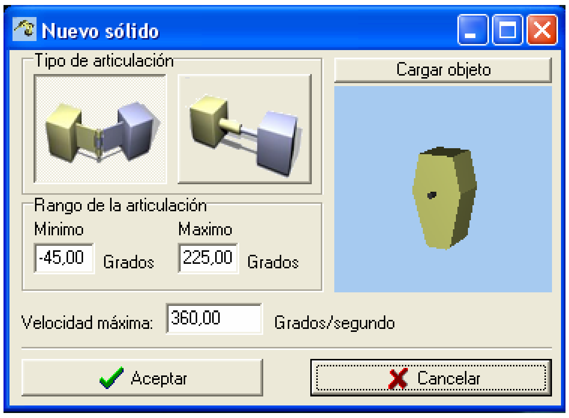
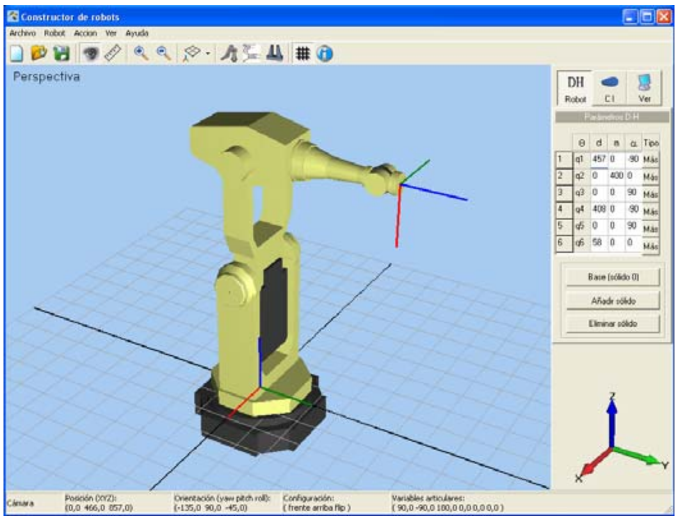


Fig. 1. Example of the definition of a robot solid.



Minimo: -45,00 Grados
Maximo: 225,00 Grados
Velocidad máxima: 360,00 Grados/segundo

Fig. 2. Example of the addition of a solid for robot assembling.



	θ	d	a	α	Tipo
1	q1	457	0	-90	Más
2	q2	0	400	0	Más
3	q3	0	0	90	Más
4	q4	408	0	-90	Más
5	q5	0	0	90	Más
6	q6	58	0	0	Más

Cámara
Posición (XYZ): (0,0 466,0 857,0)
Orientación (yaw pitch roll): (-135,0 90,0 -45,0)
Configuración: (frente arriba flip)
Variables articulares: (90,0 -90,0 100,0 0,0 0,0 0,0)

Fig. 3. Example of the robot obtained after assembling the solids.

B. *Assembling the Robot*

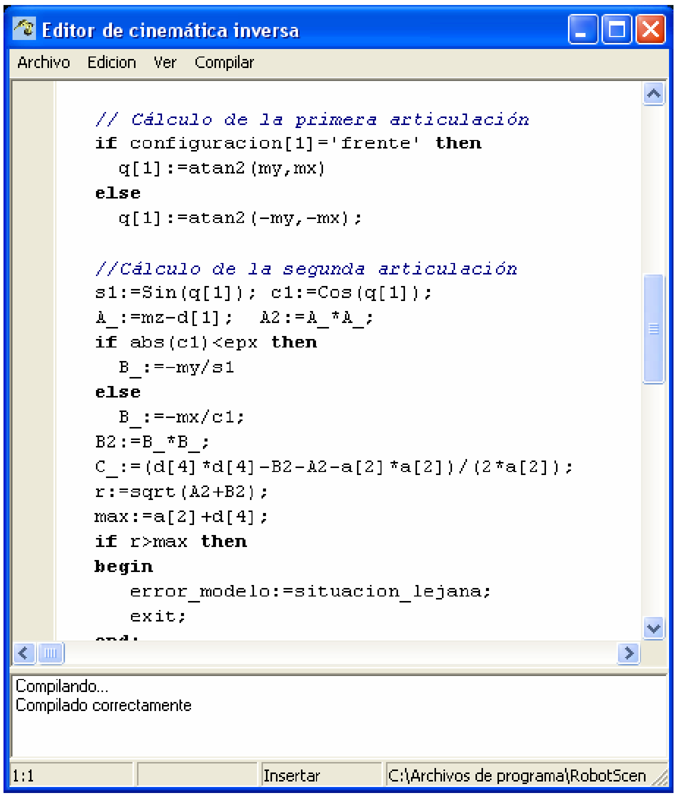
Once all the manipulator solids have been created, students can assemble their robots using the Robot Constructor Module. In this step, students need to have previously determined both the DH parameters as well as the equations that solve the inverse kinematics of their robot. According to these previous tasks, the robot creation stage exhibits two steps: the first one consists of defining in detail all the robot joints while robot is being assembling. The second step is related to the implementation of the equations that solve the robot inverse kinematics, and will be treated in the following subsection.

The complete modeling of each joint requires the definition of the following parameters: the joint type (rotational or translational), the constant DH parameters, the joint range, the home joint value, the joint maximum velocity and finally, the joint maximum acceleration. Fig. 2 shows the modeling of one joint of an articulated robot. Note that, depending on the joint type, the variable DH parameter (θ_i in this case) will be represented in the table by mean of its corresponding variable identifier (q_i). As we can see in the same figure, the joint modeling is considered in the robot construction process as a part of the solid addition procedure. For the usual case of a six degree of freedom manipulator, this addition procedure must be done for seven times (from solid 0 to 6). It is important to note at this point the role of the Denavit-Hartenberg convention in the robot modeling process with RobotScene. It determines not only the main part of the joint modeling, but the form in which each solid has been created as well.

Once students have finished the robot assembly process (Fig. 3), they can use a *robot guidance tool* that allows them to move the robot by dragging the joint-associated cursors or by specifying robot destinations in joint coordinates. Note that they cannot exploit the entire guidance tool potential (i.e. the guidance in user coordinates) because inverse kinematics is not yet implemented. Nevertheless, robot guidance in combination with reference frames visualization, can aid the students to improve their understanding about the sense of DH parameters. Likewise solids created using by the *Solid Creator Module*, robots can be saved in an ASCII file that contains all the joint attributes and the complete file path for all the solids that compose them.

C. *Implementing the Inverse Kinematics Equations*

In order to complete the robot modeling stage, students must implement the equations that solve the inverse kinematics of their robot in closed form. These equations must have been previously derived by using either geometric or algebraic approaches. For making possible the mentioned implementation, the Robot Constructor Module provides a specific programming tool based on PascalScript. As starting point, students have a source file that includes comments with useful information about the function syntax and how to access all the required data (user coordinates, DH parameters and robot configuration data). During this programming phase, students must pay attention to several problems inherent to inverse kinematics, as ill-conditioned equations and singularities detection and their treatment. Figure 4 shows an


Editor de cinemática inversa
Archivo
Edicion
Ver
Compilar

```
// Cálculo de la primera articulación
if configuracion[1]='frente' then
  q[1]:=atan2(my,mx)
else
  q[1]:=atan2(-my,-mx);

//Cálculo de la segunda articulación
s1:=Sin(q[1]); c1:=Cos(q[1]);
A_:=mz-d[1];  A2:=A_*A_;
if abs(c1)<epx then
  B_:=-my/s1
else
  B_:=-mx/c1;
B2:=B_*B_;
C_:=(d[4]*d[4]-B2-A2-a[2]*a[2])/(2*a[2]);
r:=sqrt(A2+B2);
max:=a[2]+d[4];
if r>max then
begin
   error_modelo:=situacion_lejana;
   exit;
```

```
Compilando...
Compilado correctamente
```

1:1
Insertar
C:\Archivos de programa\RobotScen

Fig. 4. Example of inverse kinematics implementation.

extract of an inverse kinematics implementation, that shows the wrist singularity detection and its treatment as a predefined error.

Once inverse kinematics is implemented, students can check its correctness by using the robot guidance tool in the following way: First, they move the robot to a destination specified in joint coordinates. Next, they commute to user coordinates mode in order to obtain them from the forward kinematics. Finally, they move the robot to these user coordinates, having selected previously the adequate robot configuration. If the robot has not moved during the last step, students can assure the correctness of their inverse kinematics solution (the equations and their subsequent implementation).

The inverse kinematics implementation is saved in a specific source code file, and its existence is annotated on the associated robot file.

III. Robot Application Programming

The second stage of the learning project-based experience consists in designing and programming a realistic robotic task. For this part of the activity, a scene containing the robot arm and the objects involved in the task is given (Fig. 5). The task the students have to design consists of several steps: First, the robot has to take the tool, which simulates a glue gun. Then, the tool needs to be calibrated and the objects of the scene located. The glue is spread around the surface of the base object following particular specifications and finally a set of four pieces are mounted. Students have to program the robot by using the provided language. For this purpose the SGRobot tool is used. SGRobot is a graphical simulation tool that allows the programming of robot arms in a graphical way (guided) and textual way (using a defined programming language similar to the commercial language VAL II). The SGRobot consists of three modules: The object building, for building the objects to be manipulated by the implemented robot; the editor-compiler of the robotic programming language CVAL2 developed for SGRobot; and the guidance tool, which is for programming the robot by guidance, graphical representation of trajectories, object manipulation, etc.

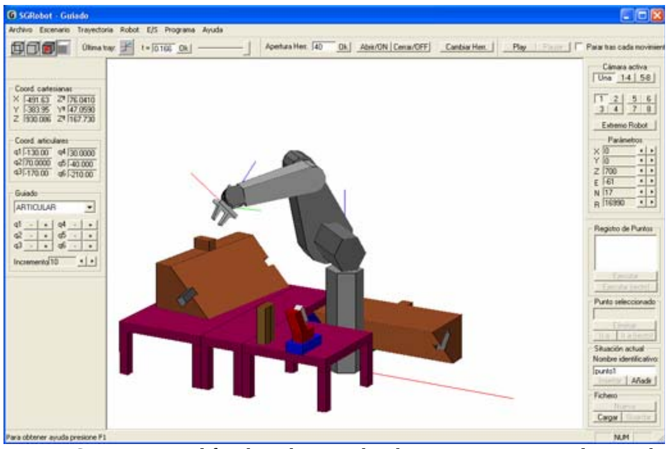


Fig. 5. Scene prepared for the robotic task. The pieces are mounted using the pins and glue.

Robot guidance

The program of *Guiado* in the SGRobot tool is an interactive program that allows guidance of the robot from the keyboard of the computer, record of points, execute trajectories, display plots and some other functions. Once started the program, the user can choose the manipulator to work with. Currently there are three different robots available (apart from the robots the user can add): Puma 560, Standford robot arm and Fanuc arm. All of them are 6 degree of freedom robots. When the desired robot is selected the main window of the program is displayed with the robot selected. Different toolbars are available in the main window and each toolbar group different options which are related depending of their functions. Some of the main functions are described next.

The menu *scenery* allows starting new empty scene, deleting the current one from the window or loading existing scenery. The menu *trajectory* contains functions related with the execution of trajectories of the robot and functions related with the record of points located through guidance. The option *Go to* drives the robot to a target location, which can be specified in articular coordinates or Cartesian, following an articular coordinated motion or a straight line motion. The option *Go to rest* leads the robot to the rest location, in the Puma robot this location is with the arm outstretched up. There are several options related with the definition and use of way points. These way points can be loaded from a file, recorded or deleted. Once the user define a sequence of way points the option *way points trajectory* can be used to move the robot following the way points defined. The robot can execute the trajectory following consecutive way points with an articular coordinated motion or a straight line motion. The window *last trajectory* shows the toolbar corresponding to the last trajectory performed. It allows repeating the motion and shows different plots like the motion of the joints, their velocities or the accelerations along the time. If the option of straight line trajectories at constant velocity is activated, the straight line motion executed will be performed at constant velocity, otherwise they are adapted to the articular motion of the joints.

The straight line trajectories at constant velocity are those in which the robot moves in a linear motion from the initial to the end point, independently of the joint motions needed to follow the straight line. This type of motion has the advantage that the straight line is covered at constant velocity, but it can happen that the motion is not allowed because a joint requires a large motion in articular coordinates not out of the robot limits, for example because a singularity. The concept of the singularity is complex to understand but, thanks to the help of the simulation tool, the student can learn this concept intuitively. We can also obtain the robot the required to execute a programmed task, since the trajectories and robot control is defined in a realistic way using the limitations of the real controllers.

```
//PROGRAM DEMOCONFIG
//Program that shows several configurations of the PUMA 560
void main(){
  //TOOL Calibration:
  T_TRANSF transfTool = TRANSF_EULER(0,0,-100,0,0,0);
  TOOL(transfTool);

  T_TRANSF location = TRANSF_EULER(-600,-700,400,30,90,0);

  BELOW();
  LEFTY();
  FLIP();
  MOVE(location );

  ABOVE();
  LEFTY();
  FLIP();
  MOVE(location );

  BELOW();
  RIGHTY();
  FLIP();
  MOVE(location );
}
```

Fig. 6. Example of program in CVAL2.

The menu *Program* gives the option to compile and execute a program in CVAL2. The menu includes the options *play*, *pause* or running the program step by step. The language exhibits all the features of typical robot programming languages related to robot setup, robot motion, localization, data management, input/output and others. An example in CVAL2 language is given in Fig. 6. Once the task is programmed and tested in the simulation tool the program is translated to VAL II or TPE (FANUC) languages for running in the real robot. In this way, students have a useful and real contact with an industrial robot with important saving in time and money required to program and check code with a real robot.

IV. Learning Improvements

In this section we discuss the learning improvements of the activities described here. We also show the statistics of the student's grades in the courses during years 2003 to 2008. Additionally, we have polled students of the academic year 08-09 about the time they spent in these courses and we discuss the results.

Simulation tools can be a highly useful instructional aid. In this case, two main learning aspects can be benefited from the use of the proposed simulation tools in the context of an industrial robotics related subject: the robot control and modeling in-depth comprehension, and the acquisition of robot programming skills.

The main learning improvements provided by the use of RobotScene during the robot modeling phase, are: the better comprehension of DH convention, and the in-depth understanding of the inverse kinematics problem. RobotScene provides a framework in which students can check in a visual way their kinematics parameters, because any error will be reflected as a wrong solid assembly. They can also benefit from the frame reference viewing in conjunction with the joint guidance tool, which allows them to improve their comprehension of the robot model and its kinematics.

On the other hand, the inverse kinematics programming allows understanding its authentic complexity, because students must take into account some different problems related to its nature and implementation, as singularities detection and treatment, arising the use of ill-conditioned equations, or the robot configuration selection. At this point it must be noted that the use of the robot in the programming stage requires a carefully inverse kinematics implementation. Additionally, SGRobot is useful to know and appreciate the utility of a graphical robotic simulator. Students can learn and practice with the different ways of programming robots. The activity requires that students implement under the simulator a glued and assembling task using guidance and explicit textual language. They also need to document their final application including explanations about the reference assignment and the motion planning performed. In order to perform the robotic task successfully the students need to solve usual problems that appears in programming robotic tasks, like obstacle avoidance, object localization, singularities avoidance, robot configuration selection, etc.

In general, it should be noted that instruction is less effective and less efficient than instructional approaches that place a strong emphasis on guidance of the student learning process [17]. The advantage of guidance in favor of active learning begins to recede only when learners have sufficiently high prior knowledge for autonomous learning. Therefore, in order to perform these activities with guarantees of success a prior knowledge is required from the students.

In the framework of new methodologies, laboratory and project-oriented courses are encouraged. In our case, two main learning aspects are improved from the use of simulation tools in the context of an industrial robotics related subject: the robot control and modeling in-depth comprehension, and the acquisition of robot programming skills.

The improvements observed because of the activity proposed are supported by the rate of success and the good reception of the courses involved. The mean values of the grades obtained by students from the last 5 years (from 2003 to 2008) are shown in Fig. 4. The grades are defined in a scale from 0 to 10 with A between 9 and 10, B between 7 and 9, C between 5 and 7, D failure to pass, and NP if the student does not apply for the evaluation (the presentation of the project). Note that there are no D grades in the results. This is because, given the project-based focus of the course, the student's work is all up to date. Moreover, each student knows before the end of the course if he has acquired the competences necessary to pass. Otherwise, they are advised during the course how to improve they performance in order to complete the course successfully. If not, the student can choose not to apply for the evaluation.

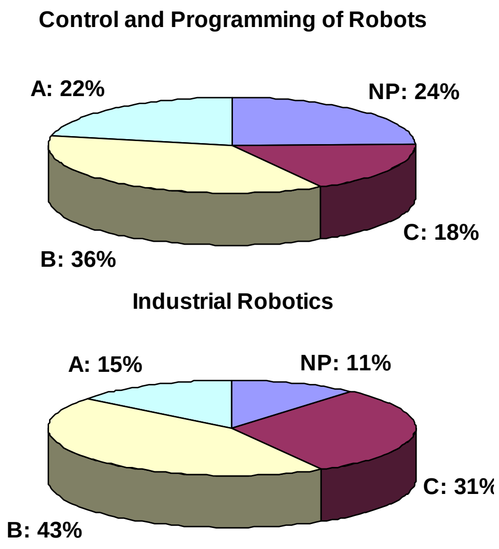


Fig. 4. Mean grades from years 2003 to 2008.

We have polled 18 students during this academic year 08-09 about their dedication to the course of Control and Programming of robots. The mean result is that they spent 50 hours (35%) in studying the theory and assistance to class hours. They spent 43 hours (30%) in practice sessions, including previous preparation, assistance, and elaboration of the corresponding reports. And they spent 51 hours (35%) in their work projects. Note that students estimate that 65% of the time dedicated to the course corresponds to practice, and relevance of practice is a characteristic of the competences-based learning. The course is designed with 4.8 ECTS (European Credit Transfer and accumulation System), approximately 116 hours of student work, including 25 hours of teaching classes, 21 laboratory hours, and 70 hours of personal work and other activities. Comparing with the poll results, students estimate that they spent in mean 28 hours of personal work over the corresponding designed time of the course. In order to reduce this difference we are going to introduce more active learning methodologies in teaching classes to improve acquisition of theoretical concepts required to fulfill the project-based activities.

V. Conclusions

In this paper we describe the use of two simulation tools for active learning in courses of industrial robotics at University of Zaragoza. In the framework of new methodologies, laboratory and project-oriented courses are encouraged. The benefits of active learning have been previously acknowledged and discussed in the literature. In this work we contribute with a new activity taking advantage of simulation tools covering these issues. The improvements observed because of the activity proposed are supported by the rate of success and the good reception of the courses involved. We found that one disadvantage of this type of project-base learning activity is that it can get involved students so much. Then, they could spend too much time to complete the activity, exceeding the minimum required to pass the course to the detriment of other courses. So, this kind of activities needs to be designed carefully.

These simulation tools can be freely distributed and at the moment SGRobot is used for teaching at different universities: University of País Vasco (Spain), University Don Bosco (El Salvador) and University of Las Palmas de Gran Canaria (Spain). In these cases these tools are used in robotics courses in the context of Industrial Engineering degree. It is also used for teaching in secondary level at school of “La Salle - El Carmen” in Melilla (Spain). In this case, the activity involves the learning of the types of robots and their main features, like degrees of freedom, types of sensors and actuators and control system. This particular activity takes advantage of the graphical and intuitive interface of our software and it is designed for students at third level of ESO (14-15 years old). Their broad use shows the versatility and usefulness of these simulation tools for educational purposes at different levels.

References

[1] J. J. Craig, “Introduction to Robotics: Mechanics and Control,” Addison-Wesley Longman Publishing Co., Inc. 1999.

[2] K. De Wit., “The Consequences of European integration for higher education,” International Association of Universities (IAU), in Higher Education Policy, vol. 16, no. 2, pp. 161-178, 2003.

[3] (2009) The official Bologna Process website. [Online]. Available: http://www.bologna2009benelux.org/

[4] T. Halvorsen, Ed., G. Mathisen, Ed., T. Skauge, Ed., “Identity Formation or Knowledge Shopping: Education and research in the new globality,” Norwegian Centre for International Cooperation in Higher Education (SIU), Bergen, 2005.

[5] A. W. Chickering and Z. F. Gamson. "Seven Principles for Good Practice." The American Association for Higher Education Bulletin, 39: 3-7. ED 282 491, March 1987.

[6] A. Romeo, “The Role of Simulation Tools in the Teaching of Robot Control and Programming,” 40th International Symposium on Robotics. Barcelona. Spain, pp. 157-162. 2009.

[7] J. J. Guerrero, “Practicas de Control y Programación de Robots. It includes Simulator SGRobot 2.0”, Dep. legal Z-178-2003, Universidad de Zaragoza. Email to jguerrer@unizar.es to obtain a free copy for teaching.

[8] Microsoft (2008). Microsoft Robotics Developer Studio, msdn.microsoft.com/robotics/.

[9] M. Mellado, C. Correcher, J. V. Catret, and D. Puig. “VirtualRobot: an open general-purpose simulation tool for robotics.” The European Simulation and Modelling Conference (ESM2003), EUROSIS, Naples, Italy, pp. 271–350, 2003.

[10] P. I. Corke, “A robotics toolbox for MATLAB”, IEEE Robotics and Automation Magazine, Vol. 3, No. 1, pp. 24-32, 1996.

[11] G. Legnani, “SPACELIB: a software library for the Kinematic and dynamic analysis of systems of rigid bodies”. U. di Brescia. 2006. http://bsing.ing.unibs.it/~glegnani/.

[12] H. Bruyninckx, “Open robot control software: the OROCOS project.” IEEE International Conference on Robotics and Automation (ICRA), pp. 2523–2528. 2001.

[13] R. Gourdeau, “Object oriented programming for robotic manipulators simulation.” IEEE Robotics and Automation Magazine, Vol.4, No.3. 1997.

[14] Fanuc (2008). ROBOGUIDE: a family of offline robot simulation software, http://www.fanucrobotics.com/.

[15] (2008). Abb “RobotStudio: of offline robot programming for ABB robots,” http://www.abb.com/.

[16] J. Denavit, and R. S. Hartenberg, ”Kinematic notation for lower-pair mechanisms based on matrices.” Transactions of the ASME, Journal of Applied Mechanics, Vol. 23, pp. 215-221. 1995.

[17] P. A. Kirschner, J. Sweller, R. E. Clark, “Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching”, Educational Psychologist 41 (2) 75-86, 2006.