Striving for Excellence in Teaching Graduate Software Engineering by Using a Helycoidal-based Educational Program

Justo N. HIDALGO, Jesús PANCORBO, Pilar VÉLEZ, Alberto LÓPEZ
Department of Computer Science and Engineering, University Antonio de Nebrija, c/Pirineos, 55 MADRID, jhidalgo@nebrija.es, jpancorb@nebrija.es, pvelez@nebrija.es, alopezro@nebrija.es

KEYWORDS: *graduate, helycoidal, project-based, iterative, workflow*

ABSTRACT: *Current Software Engineering practicioners need constant recycling with the new technologies appearing every year and state-of-the-art methodologies and techniques, proven nowadays as "best practices". Teaching software engineering in a Master's degree requires a deep effort by the students, for they must achieve a level of excellence which allows them to successfully participate in many kinds of projects. Faculty, coming from both academic and industrial worlds, must comprise all of their experience in a single year, so that students can understand all of the implications of software engineering, both from technical and business viewpoints. This paper introduces an innovative educational program in which students apply their theoretical knowledge in three different projects: an industrial-based project,a Public Service project, and a start-up internet-based one. This methodology, which we called "helycoidal", is based on a matrix, in which the columns are the different projects exposed to the students, and the rows are the workflows. At the end of each row, the student has mastered one specific discipline (project planning, social abilities, analysis & design, technology and development, deployment and peripheral activities, such as security, law and documentation techniques); the student has learnt how to apply it to a big project, a bureaucratic one, and a short but high-pressured one, understanding the similarities and, also the differences among them. The student also learns how to handle real-world problems, such as configuration management and operating system's version differences. This approach flies away from current programs which are based in a single project - therefore homogeneous- and from the case-based programs, which prevent the students from fully understanding real-size problems. The paper shows in detail how this approach is developed. Besides, three sample project definitions created and monitored by different experts in each sector, are depicted and fully explained to better understand the scope of this program.*

1 INTRODUCTION

Due to the Bologna convention, the European Space for Higher Education is changing dramatically [1]. European universities must face new challenges in undergraduate as well as in graduate studies, so they can better server global students' new requirements and needs.

In particular, Master degrees in Spain have never been included into the regulated education, which has therefore given the chance to private institutions to compete in this area. MBAs and technical masters are offered by public and private universities, and also by private business schools.

Engineering-related masters are now becoming more important than ever. Many technical undergraduate programs are going to be shortened to four years, from a classical duration of four or five years, thus creating the necessity of high-level master degrees which deepen into more specific areas, useful both to research and industrial world.

Software engineering is a very young profession which is maturing very fast. Due to the heterogeneous requirements it has got, from pure sciences such as math or logic, to economics or sociology, four years is clearly not enough time for a practicioner to master everything a project manager must know.

Besides, because of its lack of maturity and technology's fast pace, professionals are likely to become obsolete if they spend too much time in a single project, or in pure managerial aspects. Software engineers who decide to take a Master's degree to update or advance their technical knowledge want to optimize their learning or recycling curve so that in one year they can be adequately trained to face high-level project management and leadership at every possible level –planning, architecture or development-.

Classic software engineering masters are usually divided in different subjects, taught by experts in that particular field, in a sequential way. Students improve their theoretical knowledge of the field, and apply it to some kind of lab, exercise or activity, either individually or in a team. The main problems with this approach are the following:

1. Current software engineering processes are iterative and incremental-based. This model is not replicated in the degree which is usually taught in a sequential way.
2. Software engineering projects are very heterogeneous. Even though it can not be possible to comprise them all in a single course, the typical case method used in MBAs should be adjusted to this situation.
3. This heterogeneity implies the use of different methodologies, development tools and hardware systems, something which is not seen in many technical masters.

2 CHALLENGES OF LEARNING SOFTWARE ENGINEERING

Most of Software Engineers work in very heterogeneous projects when advacing in their professional career, mainly because this engineering is not as specialized as other ones, in which each professional bases their daily work in particular subjects of the area. Thus, a Software Engineer must be able to apply many different techniques, from management to technical, but also economic or emotional, all of which will be different depending on what project they are working on. Of course, mastering all of them takes much time and effort, but a Master degree should, at least allow for the students to understand each of them, and get to know when to apply their different characteristics.

The main challenges to create an effective graduate degree on software engineering must be:

1. To teach the most important theoretical aspects related to software engineering, in topics such as project management, economics, requirements engineering, design or development technologies.
2. To allow for the students to apply these concepts into effective industry-based cases. Even more, that students could develop some of their work on real projects, taking profit of companies which sponsorize the graduate course.
3. These cases should be based on different basis, such as the type of client, hardware or software architecture, or legal issues.

And the most important one: how to achieve a reasonable degree of mastering all of these areas in a single year?

This article proposes an innovative methodology, which is based in three basic ideas:

- The successful implementation of a mechanism which allows the students to success on the different topics addressed by SWEBOK [2] as everything that a Software Engineer should master. Besides, some of the topics in PMBOK [3] are also taken into account.
- The case method, used by most MBAs, is a key point of this methodology, for it helps the student to apply all of the theoretical knowledge explained, in a very realistic way.
- The helycoidal methodology, used in the Business School of University Antonio de Nebrija, is the main innovation of this technical master, for it provides the student a very effective –though stressing- way of learning how each different discipline and topic can be used in different types of projects.

The following chapter describes in detail how this helycoidal-based program is structured.

3 THEMATIC AREAS

Creating a software engineering graduate program useful for both young and expert professionals is a very hard and difficult task. Young people tend to focus their interest on development activities, and still have not got enough experience to understand what industry usually needs from them. On the other side, professionals with three or more years of experience tend to demand more information to the program about project management or emotional skills which allow for them to improve their status in the company they are working at. But also, some of these professionals need to acquire again some of the theoretical concepts which might not have been used by them during last years, or update some of the knowledge –p.e. many experts are still not used to advanced object-oriented or aspect-oriented techniques which are being used in many projects now-.

The educational program presented here takes into account the areas addressed by SWEBOK. This Book of Knowledge presents the topics which a professional Software Engineer should master so to better face an industry-size project.

The different areas which are depicted in the Master's degree are the following:

1. Project Management: this area worries about topics such as resource, cost or time management.
2. Directive abilities and teamwork: every software project is a multidisciplinary job in which emotional abilities are much more important than what it was tought twenty years ago. The concept of a lonely software developer is no longer valid, and being able to teamwork, to negotiate, and having personal characteristics such as empathy or assertivity are crucial for a software engineer and project to succeed. Work by [4], [5] or [6] has been taken into account for this program.
3. Analysis and Design: this area focuses on the architectural and methodological sides of the project. Choosing a software process, such as Unified Process, Metrica v3 or Agile ones, such as Extreme Programming, is crucial for the software to be adequately implemented and finished. Besides, this area will be used for the teachers to discuss which current architecture patterns can be used in small, medium or king-size projects.
4. Technology and Development: what technologies and development tools are being used nowadays, and which ones might find their way in a near future, are the focus of this area.
5. Peripheral activities: due to time constraints, some important topics are left to be discussed in a single area. Concepts such as Patenting, Documentation techniques, cooperative & collaborative software, or software-related laws must be part of a professional learning process.

More detail for each areas are depicted below.

3.1 Area I: Project Management

A project is a set of related tasks which must be managed in order to obtain a product in a given time limit, and with limited resources. Every software engineer must organize themselves so that their tasks can be accomplished with these constraints. PMBOK is used as the basic resource for this area, and, mainly, information about cost, resource and time management is given to the students. Some other topics such as outsourcing control or quality and metric basics are also studied.

3.2 Area II: Directive Abilities and Teamwork

Technical skills are just not enough to excel in a software project. Teamwork is essential, as well as other emotional abilities which, in the last years, have at last been recognized as an integral part of an engineer's education. Factors such as self-awareness, empathy, self-control or assertivity must be learnt, practiced and applied but project leaders, designers and developers as a whole. Students must be evaluated not only as good technicians, but also as corporative professionals.

3.3 Area III: Analysis and Design

But, of course, most of software engineers actually perform their job in the technical side of the project spectrum. Students face in this area two different, but closely related subjects: (1) Software methodologies: depending on the type of the project or client, different methodologies and process must be benchmarked and applied. Most common ones can be the Unified Process, Metrica or Agile ones such as Extreme Programming, but there are many to look after. (2) Design aspects: advanced object-oriented design characteristics, aspect-oriented programming, component-oriented design, or detailed study of most popular and useful architectures –client/server, message-oriented middleware, peer-to-peer, and so on-, along with mastering of distributed systems building, are just a few of the topics that an engineer must, at least, take into consideration.

3.4 Area IV: Technology and Development

In this area students face current and potential technologies, such as .NET [7], J2EE (Java 2 Enterprise Edition) [8], OMG's CORBA [9], LAMP (Linux-Apache-MySQL-PHP) programming [10], wireless or XML (eXtended Markup Language) [11]. All these technologies must be known by the student, but also with some kind of practice so that, when a real project comes, they can not only know of their existence, but also decide about their adequacy for a given set of requirements and constraints.

This area is also the one in which students learn about different hardware systems, such as UNIX server configurations, Microsoft's Windows or Open Source's Linux, and how to handle enterprise-size challenges, like load balancing, high availability, replication, and so on.

At last, students must use configuration and version management tools for a correct implementation of teamwork practices.

3.5 Area V: Peripheral activities

Though young, software engineering is growing and maturing very fast, so even a full and intense year of learning might not be enough. This area tries to pass by a range of subjects which are important enough for a software engineer to take a look at, and grasp the essence of what they might probably need in the near future. Patent information, documentation techniques, quality standards and certification, legal environments or advanced financial concepts –beyond what might have been told in the Area I- are examples of what the student can find here.

Until now, the structure of the course, though very intense and updated, is very classical. Some of the challenges depicted in chapter 2 are not yet solved. Following chapter shows how this areas are actually just part of the whole methodology of the course program.

4 THE HELYCOIDAL-BASED EDUCATIONAL PROGRAM

Classic courses are structured by phases, just as the waterfall lyfecycle model had been used in software project for years. Previous chapter showed a structure which could well had been used by many graduate courses on software engineering.

That way, students are visited by different experts which spend a few days or weeks with the students talking about their experience, and, sometimes, asking for the students to study, design and/or implement some case study.

What is wrong with it? Basically, what these students are doing is the same that undergrads perform: lab-based implementations, which are very small prototypes of real-world cases. Important requirements in enterprise-size products such as load balancing, time performance or stress testing are usually neglected or, at most, briefly mentioned. Besides, each expert tells about their experience, which, even though it might appear as a very interesting idea, if it is not managed and controlled, can provoke many different problems, such as overlapping, lack of global goals, and, in general, a sense of "anarchy", broadly apart from what a structured engineering course should teach. Heterogeneity is a good idea, but as long as it is adequately deployed.

Figure 1 shows the basics of the program structure presented in this paper.

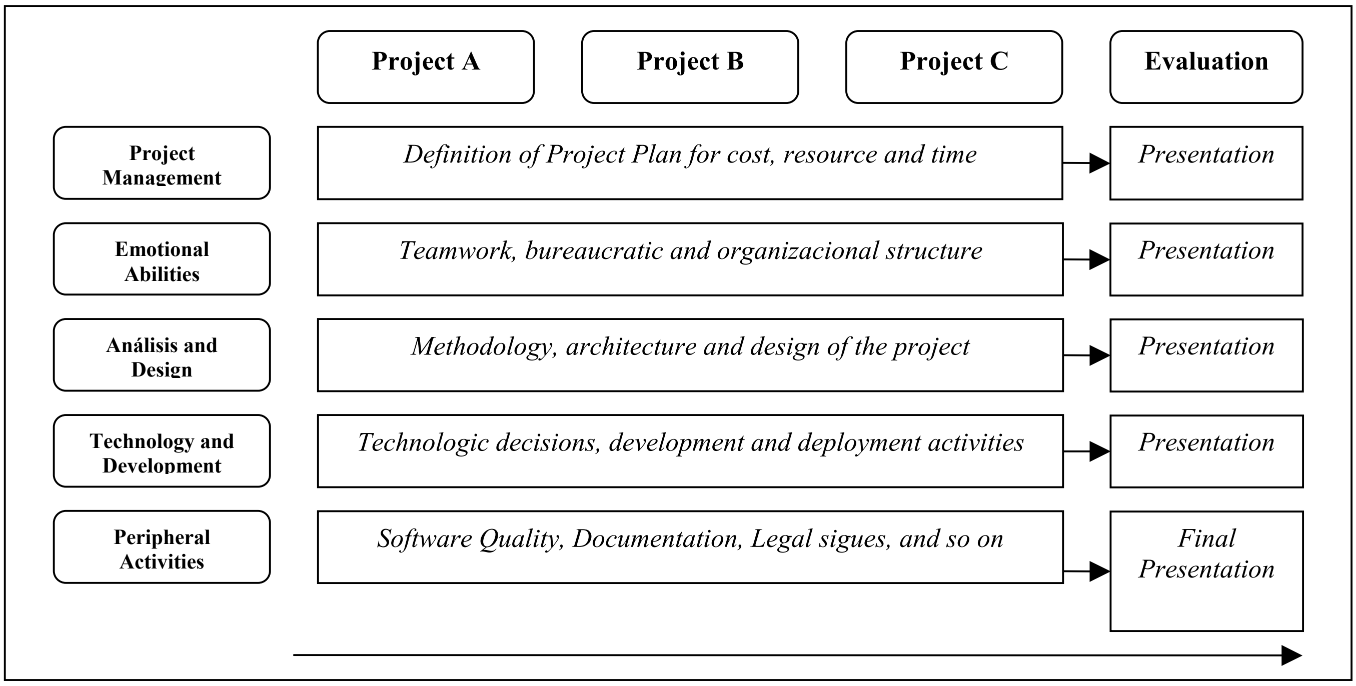
Project A
Project B
Project C
Evaluation
Project Management
Emotional Abilities
Análisis and Design
Technology and Development
Peripheral Activities
Definition of Project Plan for cost, resource and time
Teamwork, bureaucratic and organizacional structure
Methodology, architecture and design of the project
Technologic decisions, development and deployment activities
Software Quality, Documentation, Legal sigues, and so on
Presentation
Presentation
Presentation
Presentation
Final Presentation

Figure 1: general structure of the helycoidal-based program

The idea behind this structure is the following: instead of focusing on the topic areas, the program is based on "project contexts", that is, a set of enterprise-wide case studies, each of which is a very detailed example of a typical project. As it will be explained in the following chapter, the course is initially designed to examine and work on three different projects, but the structure has been built so that any number of projects can be used, as long as it follows every single topic area.

Once the projects have been initially described at the beginning of the graduate course, students face each one of the topic areas, working on it over all of the projects, in a sequential way; obviously, basic and global concepts are explained first, so that students start the projects with a good background. For example, at the beginning of the academic year, students will learn about project management, and how it will be applied to the first project. Once students have mastered it, they will have to use their knowledge to plan the second project. Some of the information used in the first project will be used in the second one, but also some new requirements and constraints will force the students to reconsider some of the decisions they made in the previous one. The same will happen in the third project. At the end of this area, students will have accomplished the following goals:

1. They will have learnt about common decisions which will be usually taken in almost any kind of project, because they will have used it in all three projects.
2. Nevertheless, students will understand that technical, management or political constraints determine some of the deliverables, so they will learn not to take anything for granted, it does not matter how many times they have applied it before.

Whenever students are finished with a particular area, and have come to a set of conclusions, students advance to the following area (p.e. Analysis and Design), starting from the first project.

The course is linearly taught row by row, so at the end of each one, the student has mastered one specific discipline because they have had to apply it to three totally different projects, understanding the similarities and, also the differences among them.

5 ORGANIZATION OF THE PROGRAM

The creation of this program faces many more challenges and difficulties than a classic sequentially-based course. Thus, much more work from the academic board is required:

- Projects must be redefined every single year, for one of the most important keypoints to achieve, comes from the set of requirements and constraints that students know at the beginning of the graduate course, for each individual project.
- Every project must be obtained from real-world situations, so teachers or mentors should come from different sectors of activity, such as enterprise, industry or Public Service institutions. A close and intensive teamwork between the academic board and industry professionals must be done in order to organize, rearrange and structure each case study, and divide them into the different topic areas. To achieve this point, it is recommendable that the program is sponsorized for at least two software development or consulting companies.
- Every project must be an "almost perfect" example of what industry requires from software engineers, either in project management, architecture requirements, technologies utilized or legal aspects. This may require an exhaustive search for different project cases.
- The most crucial aspect of this course is the coordination among all the teachers and instructors. Even though they might come from different companies, universities or institutions, they must share:
    - A common understanding of each of the projects in which they are involved.
    - A common set of goals.
    - The understanding of what the others are teaching, so that no unnecessary overlapping is made.

To accomplish these and other challenges, the required organization is the following (figure 2):

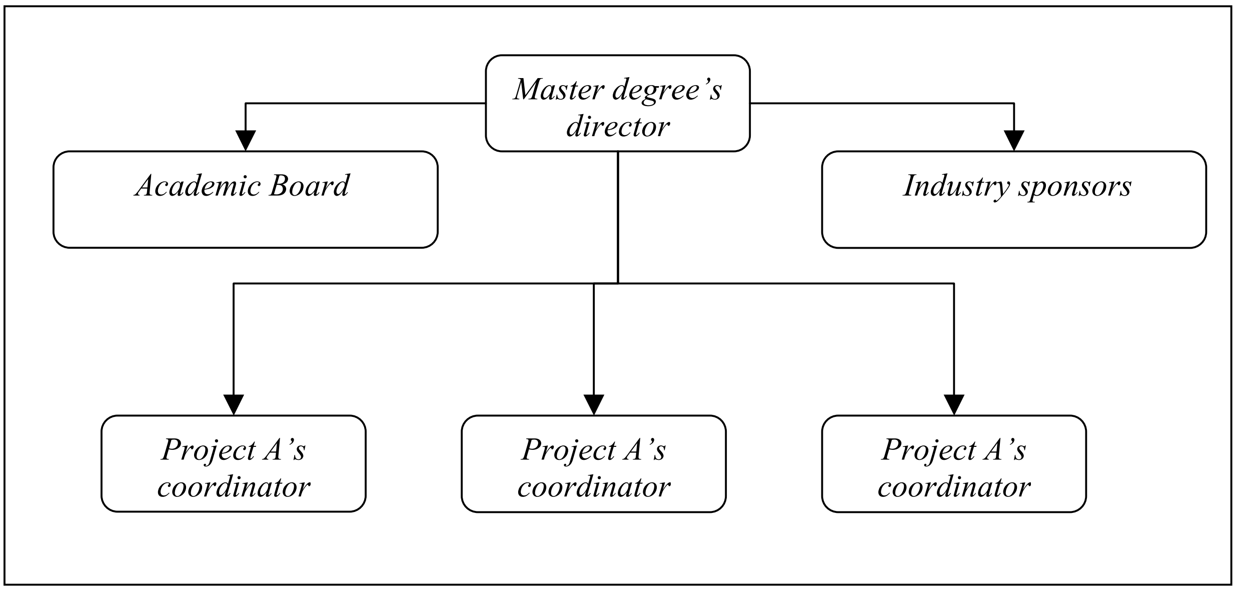

Figure 2: Organizational structure

The Master's director should belong to the Academic Board, but also has some industry background or at least some experience on software engineering projects. This director is also responsible of the organization of each topic area.

As it can be seen on the figure, each project has also a designated project coordinator, responsible for its detailed description, and its adequate relationship with the others. This coordinators should come from industry, and, if possible, be involved in the real project which the case is derived from.

Finally, each teacher is responsible for their subjects, and also for their related labs. They should report both the project coordinator and master's director.

6 EVALUATION OF THE PROGRAM

Another important aspect is how this program can be evaluated, or more specifically, how student's work, effort and results can be appropriately taken into account. The decisions have been the following, along with an explanation of the reasons which led to them:

- Each subject (academic lectures) can be optionally evaluated by a teacher. This helps the Academic Board to check the topics in which each student is stronger or weaker. However, the teacher must remember that students will be working on the projects from the beginning of the course. As usual the teacher can use one or more of the following tools:
    - Written examination.
    - Lab
    - Written report.
- At the end of each area, the artifacts generated by each team –documents, prototypes, and so on- will be evaluated, and a short oral presentation will be given to the Academic Board. Students which fail this evaluation will have another chance at the end of the course. This approach has the following advantages:
    - Because of how the course is organized, professionals might choose to take only a single area. These students will be evaluated at the same time that the whole Master's students.
    - The presentation in front of the Academic Board must be understood by the students as any enterprise's Board meeting, in which a particular project is to be accepted or denied.
    - The presentation helps the students to take profit from the emotional intelligence and directive abilities learnt in class.

At the end of the academic course, another presentation will be performed by each team, but in this case not only the Academic Board will be involved, but also the companies and institutions which might have been involved in the Master's program. This way, students will actually be evaluated on an industry's basis.

7 PROGRAM STRUCTURE AND PROJECT DESCRIPTIONS IN COURSE 2004/05

The structure of this course is designed and planned to be used in the first edition of the Executive Master on Advanced Software Engineering and Development of the Department of Computer Science and Engineering of University Antonio de Nebrija in Madrid, Spain, 2004/05. During this academic course, 2003/04, a minimized version of the structure has been used in two undergraduate courses on Software Engineering –Software Engineering I and II- [12]. The Master's degree will have 300 hours (30 credits of work) of presential teaching during a whole year. The different activities to be realized by the students will have to be achieved by their own, even though a computer lab and different teamwork offices will be available for them for the whole week. The credit division for each topic area is as follows:

1. Project Management - 5% (15h/300h) (1.5 credits)
2. Directive Abilities and Teamwork - 5% (15h/300h) (1.5 credits)
3. Analysis and Design - 25% (75h/300h) (7.5 credits)
4. Technologies and Development - 60% (180h/300h) (18 credits)
5. Peripheral activities - 5% (15h/300h) (1.5 credits)

Project definition, as it has been described previously, is the most important and critical task to be performed each year. The adequacy of each project to the defined project areas, and how they comprise most of the current and potential methodologies, technologies and management techniques, are the keypoint which will assure the success of the program.

For course 2004/05, three projects have been defined:

1. Public Service project
2. Enterprise-wide project
3. Medium-size dynamic project

7.1 Public Service Project

This project will be structured as if students belong to a private consulting and development company, which obtains a contract from some Ministry or Public Administration to build an application which will be used by the citizens of a city, community or state. In course 2003/04, the project will be the construction of citizen's web-based access to public services. Students will have to create a general framework which allows for fast implementation of different services. Therefore, they will have to design a component-based architecture, which main elements will be: security –AAA-, design of electronic signature, content management, service personalization (myPortal), and internacionalization. An application of this framework will be a prototype implementation of Automatic Payment of Taxes.

The software methodology which will be used is Metrica v3, standard in Spanish Public Administration.

The Hardware and Software requirements will be the following:

- Server: Netra 1100
- Operating System: UNIX System (Sun Solaris or HP-UX)
- Web and Application Server: SunONE, BEA WebLogic or IBM WebSphere
- Database: Oracle 9i
- Software development: J2EE/XML/RMI/CORBA/SOAP
- Configuration management: CVS

7.2 Enterprise-wide Project

Students belong to an ISP/ASP (Internet Service Provider / Application Service Provider), and are asked to build all the required infrastructure to provide internet domain hosting and associated services, such as: automatic domain provisioning, electronic mail under own domain, web services, DNS automatic management, and so on.

The software methodology which will be used is the Unified Process as defined by Rational, and with OMG UML as its modeling language.

The Hardware and Software requirements will be the following:

- Server: HP server.
- Operating System: Windows Server 2003.
- Web and Application Server: Microsoft Internet Information Server.
- Database: SQLServer 2000.
- Software development: .NET (using Visual Studio .NET on C#)/XML/WebServices.
- Configuration management: Microsoft SourceSafe.

7.3 Medium-size Dynamic Project

Students belong to the Information Services Department of an academic institution, and are asked to design and implement a low-cost but highly-available tool which allow for teachers and students to work in a cooperative environment.. Teachers will be able to coordinate what they are showing in their computers with what students are seeing, allowing for bidirectional communication. Students will be allowed to talk among them, and with teacher, but every single piece of data will be stored in a server so that further analysis could be done off-line. Students are asked to produce several releases during the duration of the project [13].

The software methodology which will be used is eXtreme Programming, so that students learn to use Agile Programming techniques.

The Hardware and Software requirements will be the following:

- Server: Dell PC
- Operating System: Linux System
- Web and Application Server: Jakarta Apache/ Tomcat
- Database: MySQL
- Software development: PHP
- Configuration management: CVS or phpGroupware.

8 CONCLUSIONS AND FUTURE WORK

This paper shows the structure of an innovative program for teaching advanced Software Engineering to graduate students, focusing on professionals with more than three years of experience. The use of an helycoidal-based structure allows for the student to fully understand each of the topics after having worked in three different projects, each one of them providing different but always stimulating challenges.

The advantage of this program is clear: the student, after a year of hard work, will have managed, analyzed, design, implemented, tested and presented three industry-level projects to a board of academic and enterprise-related professionals which will evaluated their job.

Besides, the teach-by-provoking methodology forces students to make decisions and commit several mistakes, which will help them for the near future.

Technical difficulties arise because every project requires different environments, so a very complete laboratory must be provided. The best answer to this challenge is for these hardware and software providers to become sponsors of the course, to obtain a “win-win” situation.

The main difficulties come from the same reason students must face a hard but fruitful year: academic organization, project description and definition, teacher election, ... must be of the highest quality, and must be revised every single year. It can be said that the creation of this academic course must apply most of the project management and directive abilities that students will study the following year. But the authors of this paper believe that the skills obtained by the students will help Software Engineering to obtain the degree of recognition that the society is demanding.

REFERENCES

[1] Council of Europe. The Europe of Cultural Cooperation: Bologna Process. Available from web: <URL: www.coe.int/T/E/Cultural_Co-operation/education/Higher_education/Activities/Bologna_Process/default.asp>

[2] Guide to Software Engineering Body of Knowledge (SWEBOK). Available from web: <URL: www.swebok.org>

[3] Guide to Project Management Body of Knowledge (PMBOK). Available from web: <URL: www.pmi.org>

[4] GOLEMAN, D. 1997. *Emotional Intelligence. Why it can matter more than IQ*. Bantam. 0553375067.

[5] GOLEMAN, D. 2000. *Working with Emotional Intelligence.* Bantam. 0553378589

[6] SEGAL, J. S. 1997. *Raising your Emotional Intelligence: A Practical Guide*. Henry Holt & Company. 0805051511.

[7] Microsoft .NET Distributed Technology. Available from web: <URL: www.microsoft.com/net>

[8] Sun Java 2 Enterprise Edition. Available from web: <URL: java.sun.com/j2ee>

[9] OMG CORBA. Available from web: <URL: www.omg.org>

[10] ONLAMP: O’Reilly resources about LAMP technologies. Available from web: <URL: www.onlamp.com>

[11] XML.com: Information and resources about XML. Available from web: <URL: www.xml.com>

[12] HIDALGO, J. Nebrija University’s Software Engineering Web Site. Available only in Spanish from web: <URL: www.nebrija.es/~jhidalgo>.

[13] GARRIDO, M., PANCORBO, J., HIDALGO, J. *Collaborative Compilation: An Experience on Using Groupware Tools Applied To Computer Programming Learning.* To be published on the International Conference on Engineering Education and Research, iCEER 2004.