A Modified Agile Methodology for an ERP Academic Project Development

Estellés, E., Pardo, J., Sánchez, F. and Falcó A.
Physical Sciences, Mathematics and Computation Department
University CEU-Cardenal Herrera
Spain

1. Introduction

The economy and market globalization have forced all kind of enterprises (large, small and medium enterprises, SME) to compete against their counterparts from other countries, reaching high competence levels. This fact exposes a clear and essential need to be continuously improving business processes for reaching high levels of competitiveness. Regarding information management, Information Systems (IS) have responded to the increasing necessity of organizations to improve their capabilities to process and manage data. This need arises because the capability of providing the right information at the right time brings tremendous rewards to those organizations (Hossain et al., 2002). In that kind of competitive market, both big and SMEs need information systems for tackling the direct rival pressing, keeping their position in the market (Valor et al., 2007) while improving competitiveness by cost reduction and better logistics.

Due to this situation, a SME from la Comunidad Valenciana (Spain) called Chair's Collection S.L. (CC), contacted our university, Universidad CEU-Cardenal Herrera, in order to establish a collaboration deal consisting in the creation of an Enterprise Resource Planning (ERP). Being a great opportunity, the University suggested the Company that it should consider the ERP as a final degree project for a last academic year student, which was accepted.

Assuming the way the collaboration deal was going to be carried, it was necessary to plan how to face the project. It was clear that a traditional software development approach wouldn't imply the programming features and flexibility needed. Considering the project as a final degree project, there was a variable amount of time available for developing the project (between 10 and 12 months at least), so a methodology which allowed to change requirements and software features without a high effort and time cost was needed. Agile methodologies were quickly taken into account.

These kinds of methodologies are people-centred (allowing the student guide and support and the customer attention and care). They give more importance to software over documentation (allowing us to develop a real project to be used after the final degree project handing in) and respond to change over following a plan (which met one of our first needs: the change acceptance) (Beck et al., 2001). Among all the existing agile methodologies, the

eXtreme Programming (XP) was the chosen one. This choice was taken because XP is made up of easy practices that any student can understand and because it can be easily modelled.
It was also clear that the strict execution of the practices and principles of the XP methodology wouldn't be enough: although that kind of methodology was needed, the real situation was special and couldn't meet all the XP requirements. So it was decided to tailor that XP methodology to the specific situation: roles were distributed, practices and processes modified and an adapted agile methodology, which finally met the minimum situation requirements, was designed.
The result of following the tailored XP methodology was a high modular ERP comprehending all the enterprise areas. However and in spite of being a functional ERP with a perfect task execution, the company didn't use all of its potential, leaving some modules unused, or used them with some constraints. In this chapter we will try to explain the reasons why this happened.
The present chapter begins with a theoretic presentation about Information systems and SME and about Information Systems and Agile Methodologies (Section 2 and 3 respectively). After this, in section 4, the project and the tailored methodology will be explained in detail. In section 5, the results of the project will be shown. Finally, a conclusion section will bring up the mistakes in the student attitude as well as in the methodology exposition, and the lessons learned.

2. Information Systems and SME

2.1 Information Systems and ERP

The Information Systems (IS), as a discipline, is a pluralistic field founded on borrowed knowledge from other, more established, source discipline. These disciplines consist of Psychology (e.g. for the assessment of the impact on individual users) to Economics (e.g. to cost-justify an IS to the organization's management), passing through Mathematics, Linguistics, Semiotics, Ethics, Political Science, Sociology and Statistics and Computer Science. Counting with the influence of so many different disciplines, it seems logical that the areas affected by the IS discipline and the products derived from it are many, obviously business but others like health and government (King, 2006).
Nowadays almost all IS are supported by computer technology (although they existed before it), that is the reason why the IS discipline is closely related to Information Technology (IT). In fact, businesses around the world spend over 2$ trillion a year on IT (Carr, 2003).
Regarding business, IS has responded to the increasing necessity of organizations to improve their capabilities to process and to manage data. From the point of view of the IS, it has been seen as an application of computers to help organizations process their data so that they could improve their management of information (King, 2006). In this respect, Andreu, Ricart & Valor [1996] define IS:

"IS is an integrated set of formal processes, developed in a user-computer environment, that working with a set of structured data from an organization, summarize, process and distribute selectively the necessary information for allowing the organization normal operations and the management activities" (Andreu et al., 1996)

Assuming this definition and the support IS can provide to an organization, they can be distinguished in the following types:

1. Transaction Processing Systems (TPS)
2. Management Information Systems (MIS)
3. Decision Support System (DSS)
4. Group Decision Support System (GDSS)
5. Executive Information Systems (EIS)
6. Expert Systems (EESS)

As TPS tool, highlights the Enterprise Resource Planning (ERP), term coined by Stanford Gartner Group's Computer-Integrated Manufacturing Service at the beginning of the 90s although it existed long before that. The origin of actual ERPs appeared in the 60s and was called Inventory Control Packages (ICP). This software developed through the years: in the 70s it was called Material Requirements Planning (MRP), in the 80s Manufacturing Resources Planning (MRP II), in the 90s appeared the term ERP, and nowadays the ERP has evolved to the ERP II or extended ERP (Hossain et al., 2002).

An ERP has been defined by many authors from different points of view. The American Production and Inventory Control Society defines it as a "method for the effective planning and control of all the necessary resources for producing, sending and accounting the orders made by the customers in a manufacture, distribution or service company" (Ramirez, 2004). Rather than listing and detailing all the ERP definitions found (Kumar & Van Hillsgersberg, Markus et al., Shanks and Seddon, O'Leary, Nah et al between others) (Ramirez, 2004), a combined definition is provided:

"An ERP is an extensive commercial packed software solution composed by some configurable modules that firmly integrate, in an stand-alone system, the central business activities – finances, human resources, manufacture, supply chain, customers management – through the information streams automation and the use of a shared database. It also incorporates in this process the best integration practices for fast making decision, costs reduction and directive control, getting with all these the efficient and effective use of the business resources."

Regardless of the definition used, an ERP system is required to have the following characteristics:

- Modular design compromising many distinct business modules such as financial, manufacturing, accounting or distribution.
- Use centralized common database management system (DBMS).
- The modules must be integrated and provide seamless data flow among the modules, increasing operational transparency through standard interfaces.
- They are generally complex systems involving high cost.
- They are flexible and offer best business practices.
- They require time-consuming tailoring and configuration setups for being integrated within the company's business functions.
- The modules work in real time with online and batch processing capabilities.

2.2 Small and medium enterprises scene

It's accepted that small and medium enterprises (SMEs) represent a vast portion of the firm tissue of almost every developed country irrespective the sizes adopted for considering a business as a SME.
As showed in the 2008 report *"The Small Business Economy. A Report to the President"*, in the U.S.A. there were more than 20 million non-employees firms plus around 6 million more that employed less than 500 employees. This means that more than the 99% of all the enterprises in the U.S.A. were SMEs in 2008 (USA report, 2008).
In the European Union there is a similar scene: micro and SMEs play a central role in the European economy. In the Eurostat report "European Business. Facts & Figures" is shown that in 2004 there where almost 19 million enterprises in the EU-27, of which 99.8% were SME. These SMEs, which number is increasingly growing, provide around 75 million jobs (EUROSTAT, 2007). Inside the EU the records for Spain are in line with the European ones: there were more than 3 million firms of which the 99.87% were SMEs. Of those SMEs, only the micro enterprises provided the 50.5% of the jobs of the country. As for the Comunidad Valenciana the numbers were similar to the Spanish and European ones: the 99.91% of the firms were SMEs (DGPYME, 2007).
Due to the economy and market globalization all these SME are forced to compete against counterparts from other countries that often produce cheaper products (although of variable quality) and get into the markets in a strong position. In the Spanish case, and according to the *"Furniture Sector Results Report 2006"* by ANIEME (National Association of Furniture Industrials and Exporters in Spain), in the year 2006 there was a decrease of the 2.5% (€1,467 million)in the number of exportations and an increase of the 3.8% in the importations (€2,328 million). These data reveal that the Spanish furniture market is being supplied by high-medium quality furniture from the EU (38.1% of the importations) and economical but worst quality from China (31% of the importations) (ANIEME, 2006). This global market and the competence that it implies, exposes a clear and essential need of being continuously improving the business processes for reaching high competitiveness levels.

2.3 Small and medium enterprises & information systems

As it was said before, IS has responded to the increasing necessity of organizations to improve their capabilities to process and manage data. This need arises because the capability of providing the right information at the right time brings tremendous rewards to organizations in a global competitive world of complex business practices, which is recognized by big and SME companies (Hossain et al., 2002). In that kind of competitive market, both big and SMEs need information systems for tackling the direct rival pressing, keeping their position in the market (Valor et al., 2007) and improve competitiveness by cost reduction and better logistics.
All these IS benefits have been well-taken by large organizations until recently. IS, ERP in particular, have been always used only by large organizations which were able to pay the great amount of money that this kind of systems costs. Nowadays the commercial ERP vendors have taken into account the SME and they are developing products according to their size: less modules, adapted to their needs and, obviously, less money costs and implantation time. However, and in spite of the adapted ERP systems, actual literature shows that the organization size plays an important role in the main key dimensions of the ERP system implantation.

Some studies agree that there are some differences between large and SME's companies (Ramirez, 2004):

- The ERP system introduced by small enterprises represents big resources compromise, whereas big enterprises can take advantages to the scale economies.
- Big organizations implant ERP systems by modules but SME's use to make a complete ERP implantation.
- For SME's the ERP system success is associated with the implantation accomplishing the time and budget forecast.
- In SME's the implantation time is shorter, which can indicate that it depends on the enterprise size and complexity.

Before finishing this section, the origin of the strategic value of the IS has to be delimited and clarified. Because what makes a resource truly strategic is not ubiquity but scarcity, it's not the IT that supports the IS (available and affordable to all) the most important feature. The same reason can be used for discarding the software generic applications as the key of the IS strategic value. Obviously, when companies buy a generic application, they buy generic processes as well, so these aren't the key either (Carr, 2003). As for ERP IS, people who will use it play an important role, specially their attitude towards change (the adoption of ERP systems usually involves radical organizational change) and the perceived usefulness and perceived ease of use (Kee-Young, 2006), but even being an important factor, people aren't the key either. As it can be inferred from above, the IS strategic value depends on many variables and it is only the combination of all of them together what makes the IS that valuable.

3. Software development methodologies

3.1 Agile methodologies as good solution for ERP projects

In the late 60s many and different software development methodologies got into the software scene placed by the software engineering community. Each of these methodologies attempted to solve different programming issues and many of them reached a mature and stable level. Among all these traditional methodologies, the waterfall model is the oldest one (Huo et al., 2004).

This model was introduced by Dr. Winston W. Royce in 1970. Thanks to his experience about managing large software developments, he could propose a complete computer program model development, which starts from the two basic steps required for a simple program (analysis and coding) evolving to a more complex set of linear steps (including requirements, design and testing) finishing in an pseudo-iterative model that implies nearly the return to the origin if an error occurs (Royce, 1970).

Its principles are rooted in the Tayloristic paradigm which promotes a strong conformance to a plan through upfront requirements gathering an upfront systems design and encourages strict division of labor and the use of role-based teams (Mauer et al., 2007). Generally understood as a strict linear stepped model with backward step possibility, where the output of one step is the input of the next one, and any stage can't start until the previous step has finished and its results have been approved, this model has been widely used (Huo et al., 2004) and it's still used nowadays in enterprises of many kinds: defense, pharmaceutical, chemical, telecommunications, banking and government industries (including several Fortune 500 companies list) (Neill et al., 2003).

Despite this data and the fact that it has reported success with large and complex systems, it has a well known and literature exposed number of drawbacks due to the Tayloristic paradigm principles (Mauer et al., 2007) (Sommerville, 2006). These drawbacks appeared clearly when in the 90s the programmers tried to apply this heavyweight, plan-based development approach to small and medium-sized business systems (Sommerville, 2006). They are mainly two: the highly ceremonious processes and mainly the inflexibility in the face of changing requirements.

The waterfall model assumes that all the requirements can be accurately gathered at the beginning of the project, assumption that in medium or large projects is impossible to take because of the stakeholder users. They cannot tell developers everything about the system at the beginning of the project, most of the times they don't say once and for all exactly what they want or need, they contradict themselves or they change their minds (Beck, 1999)(Woi, 2006). This requirement drawback resulted in high costs of change in terms of time and money as can be shown in the Barry Boehm cost of change curve (Ambler & Scott, 2004).

Due to the problems of applying the waterfall model to small and medium-sized business systems and taking the high cost of changing software as a challenge, the academic software engineering community started to use different and new practices in combination with older ones and to create technologies like relational databases, modular programming, and information hiding (Beck, 1999) (Sommerville, 2006).

The result of these efforts was a more humanistic and collaborative approach to software development known as "agilism" (Mauer et al., 2007) that led to methodologies as that of Scrum (1986), Dynamic Systems Development Method (1995) or Extreme Programming (1996) among others. The "agilism" is a lightweight process that employs short iterative cycles, actively involve users to establish, prioritize, and verify requirements, and rely on team's tacit knowledge as opposed to documentation (Woi, 2006). It's important to remark that many of the practices involved in these methodologies are not new; for example software inspections were introduced in 1970s and rapid prototyping in 1980s (Beck, 1999)(Mauer et al., 2007).

The different methodologies based on "agilism" received the name of Agile Methodologies in February of 2001, after a meeting in Utah-EEUU, where a group of 17 prominent figures in the field of agile development (including the designers of Scrum, DSDM and Extreme Programming) came together to discuss the keys and values that the development teams should follow for a lighter, faster and better way of developing software. The result of this meeting was the Agile Manifesto, which enumerates 12 keys for getting this achievement (Letelier et al., 2004). These 12 keys can be summarized in 4 main values:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan.

Agile methodologies are composed by a great number of software development techniques that follow the agile manifesto in different ways but that have always the same philosophy and have at least six common features: 1) collaboration, 2) code reviews, 3) small teams, 4)

short release schedules, 5) time-boxing and 6) constant testing (Coram & Bohner, 2005). Agile methodologies are growing in use and popularity, and nowadays companies are reporting success in meeting rapidly changing customer needs through their use. Motorola Wireless Systems Group, for example, has begun to gradually adopt Agile methodologies and several of its projects have shown extremely positive results as improved quality, improved cycle time or reduced risk among others. (Woi, 2006)
Regarding the cost of change, Kent Beck (designer of the Extreme Programming methodology) has reported that the use of his Agile Methodology (XP) implies that the cost of change can be inexpensive even late in the project lifecycle while maintaining system quality. The reason is because in XP techniques which reduce the feedback cycle are followed. Agile techniques have short feedback cycles while traditional ones have longer feedback cycles (Ambler & Scott, 2004).
However all the issues exposed, agile methods are not a silver bullet and agile practices only work in context. Agile methods help to succeed in unpredictable environments (Mauer et al., 2007) while waterfall model is a good choice in large and complex software projects that require systematic discipline with the requisite process overhead to ensure success (Coram & Bohner, 2005), the requirements are well understood and unlikely to change radically during system development (Sommerville, 2006). Also many companies have found that Agile methodologies work better in small projects while traditional methodologies usually work better in large ones (Woi, 2006), although K. Beck reports that XP has been applied in a wide range of projects succeeding in small and big projects (what he also indicates is that the practices must be augmented and altered when the team is composed by many people) (Beck, 2005).

3.2 Why are agile methodologies a good solutions for ERP problems

It's known, and literature shows, that the use of high-ceremony, science-based, project management methods have given rise to failure in ERP projects (Alleman, 2002). Experience has shown to some development teams that traditional development methods are not always adequate nowadays: they need to refine their development process for getting new features to market quicker, to get customer feedback earlier in the process, to test early in the developing process to avoid large re-work efforts before deployment, etc. (Sumrell, 2007) These old methods, particularly the waterfall one, contain several erroneous assumptions that impact negatively in software projects, especially the ERP projects:

- They understand the *project planning* as a set of successive steps that have to be executed without a backward step.
- *Changes* are not welcomed. They can be made, but with a cost that grows when the project is advanced.
- They want a *stable* plan to which they can commit.

In real life it's not possible to produce such a perfect plan: there always appear unanticipated problems and, in certain situations, deferring decisions for taking advantage of new information or new opportunities that imply the change of the plan are necessary. About the change, many times it isn't a problem but a competitive chance that has to be faced for getting advantage (Alleman, 2002). All these problems get worse in ERP projects: these are people-centred projects which rely heavily on change management for success

facing constant change and reassessment of organizational processes and technology (Alleman, 2002)(Kee-Young, 2006).
Attending to the ERP projects characteristics (Alleman, 2002), Agile Methodologies fit better in their needs for many reasons, principally because they emphasize rapid and flexible adaptation to changes in the process, product and business, and deployment environment.
Recent studies show that IS utilization behavior is significantly affected by beliefs about the system's usefulness and ease of use (Kee-Young, 2006). These two characteristics can be powered by the customer or user involvement in the project. In waterfall and other traditional models, customers are involved at the inception and the end of the project. On the contrary agile methods involve customer or user much more frequently and with more influence. In fact, customers in agile methodologies (AM) should be committed, knowledgeable, collaborative, representative, and empowered (Coram & Bohner, 2005). Because requirements of ERP project evolve over time, the stakeholders understanding of them evolve as well, changing their point of view and consequently changing the goals and success criteria of the project (Alleman, 2002). Agile Methodologies embrace this change, turning it into competitive advantage.
Finally, without quality software all the agile methods and practices are useless. In this sense, agile methods include many practices that have quality assurance (QA) potential. Some of them are inside the development phase and some others can be separated out as supporting practices, but all of them have a higher frequency that in waterfall model. Other advantage about QA is that its practices are available in very early process stages. These practices are among others: system metaphor, on-site customer, pair programming, refactoring, continuous integration, acceptance testing or early customer feedback (Huo et al., 2004).

4. Final degree project description

4.1 Project overview

The project, on which this case study is based, consists in the design, implementation and implantation of a tailored ERP for a SME in the Comunidad Valenciana, Spain. It comprises the final degree project (FDP) of a student from the CEU-Cardenal Herrera University that is nowadays in an evolution phase. The SME for which the project is, Chair's Collection (CC), is one of the hundreds of SME of the Comunidad Valenciana dedicated to the furniture sector. CC was founded in 1984, focusing its sales on big installations like hotels in the UE and other countries, specially the USA.
After 25 years of existence without an Informatic Department or other department with similar functions, the result, about data, are three different applications. These applications were built by different suppliers and now each one works with duplicated and unsynchronized data that makes difficult the normal company operations. Besides the data problem, all processes are paper and oral messages managed with the risks that that implies.
When facing the purchase of a commercial ERP, CC had two main problems that limited highly their decision:

- As literature shows, SMEs have in many cases company-specific needs for the ERP software, but they usually have to adapt to the processes and information

structures the ERP systems offers. This was a situation which CC wanted to avoid: they liked the way they worked and didn't want to change it.
- As other SME, and especially in the difficult market situation, they didn't have resources for creating a tailor-made ERP software to cover all their requirements to a professional software company.

Because of this situation, CC, contacted with the CEU-Cardenal Herrera University in order to suggest a collaboration deal. This collaboration resulted in a project focused on the design, implementation and implantation of a tailor-made ERP, covering their most crucial areas. The ERP proposed was a high-modular ERP focused on order processing, inventory management and invoice generation. The other possible modules were discarded for specific reasons: previous correct functional software (accounting) or no interest (Custom Relationship Management or CRM).
Summarizing, the final objectives of the project were:

- The process automation, avoiding papers and extra verbal communications. This automation didn't imply the process change.
- The data integrity in one common data base.
- A set of interrelated web applications for the automation of the process.
- A hardware infrastructure to support the ERP system.

4.2 Project planning

When planning the project development it was considered a series of issues of different kinds: a possible global schedule based on the time we had, the program language to use, etc. but we realized that almost all of them were subject to just one: the programming model. We had two options:

- A traditional model, like waterfall, characterized for being a strict steadily downwards of the 5 phases named before (2.1 Waterfall Vs Agile) without going back (Woi, 2006) or doing it but with a high time cost.
- One of the Agile Methodologies (AM), characterized for being lightweight processes that employs short iterative cycles, actively involve users to determine the requisites and rely on a team's knowledge as opposed to documentation (Woi, 2006).

We explained in 2.3 section why we think that AM are better for ERP projects in SMEs, anyway we are going to describe the two main reasons that impelled us to choose an AM.
A tailor-made ERP supposes a project with high customer collaboration. The first implication of the customer is the requirements issue. He will have to indicate and give a precise description of all the needs and requirements he want the ERP to include. As logical and human, it is possible that as the project develops the customer changes his point of view about many issues or that he realizes that has forgotten some requirement information. Obviously, those changes will imply a requirement modification that will affect to ERP itself. In this way, the ERP development builds a feedback cycle with the customer. The problem appears when the changes have to be made. As it was said before, high-ceremony methods, like traditional ones, have given rise to failure in ERP projects (Alleman, 2002).

This kind of projects need a methodology able to adapt to changes, and one of the main features of the AM is that they can handle unstable requirements throughout the development lifecycle (Huo et al., 2004).
The other reason is of academic nature. One of the objectives that we established for the student training was the knowledge of how the customer relation in real world is. In this respect, AM offered us the opportunity of instructing the student in this field. AM are characterized for being people-centred, and as for customers, these are more involved and have more influence that in traditional methodologies. Also AM are characterized by their fast system releases (from 2 weeks to 4 months) (Coram & Bohner, 2005), each of one implies a customer contact for knowing how the release is working. In this way, the student would have much more contact with the customer. In fact, he started to work on the project in the CC headquarters with the objective of being near the customer and ask him anything he may need to know.
As we decided to build a high-modular ERP, each module could be understood as a small Agile Project, preceded by a business process study. After deciding that the programming method that was going to be used was an AM, the next step before beginning the project development, was to choose an agile methodology and adapt it to our real situation. We counted only on three people: the student, the tutor and an external professional. Although all AM presuppose teams of at least 3 people, as our team, it is obvious that almost all of the tasks would rely on the student. In the next section we will explain the role adaptation. We analyzed some agile methodologies and we decided that the one that better fitted our needs was the Extreme Programming methodology.

The two main reasons of using this AM are:

- The set of roles that are clear and precisely identified (Letelier et al., 2004). These roles have allowed us to share them between our three components keeping the authority range between teacher and student. It also allowed the student to face different role tasks.
- It doesn’t have an exact schedule to follow. Thanks to this feature, we could qualify the student according to a personal academic schedule, allowing the normal working of the project, and allowing the development to evolve beyond the academic subject.

Therefore the different tasks and practices in this AM should be adapted.

4.3 Tailor-made XP

As it was said before, we counted only on three people: the student, the tutor and the external professional, so all the tasks inside the project had to be distributed between these three roles. The development process, life cycle and practices had to be also adapted to the real situation.

The life cycle of XP consists of five phases (Coram & Bohner, 2005):

1. *Exploration*. In this phase customers provide requirements for the first release while the team becomes familiar with the technology, tools, and practices that will be used.

2. *Planning*. The project team and the customer will determine the capabilities needed for the first release.
3. *Iterations to release*. This phase consists in some iterations, that take from one to four weeks, that will produce the first release. Last iteration will finish with the *productionizing* phase.
4. *Productionizing*. The project team will test and check to ensure the releases meet the customer requirements. This is the time for new changes.
5. *Maintenance*. In this phase all the changes exposed in the previous phase will be made.
6. *Death*. This phase is reached when the evolving of the system no longer exists.

Because the project was brought up as a set of modular applications, we first needed to have a wide perspective of the project before beginning the applications development. In our case, we took a *pre-exploration* phase that consisted in four interviews between the student and the customer. Those interviews were recorded and any material used in it (drawings, diagrams, etc.) were stored for requirements identification and project understanding. After each meeting the student joined all the compiled material and created a written document describing the business processes explained and the requirements identified. Then, he presented us that document and we guided him about future interviews, advising him on which areas should focus the interview and how to treat the customer. The second objective of this *pre-exploration* phase was to create an application listing enumerating and describing with a few paragraphs the main application function and features.
After this *pre-exploration* phase, we followed the original life cycle of XP. Other of the XP features to have into account is the *development process*. The original XP development process is as follows (Letelier et al., 2004):

1. The customer defines the business value to implement.
2. The programmer estimates the effort needed for its implementation.
3. The customer selects what to build, according to his priorities and time restrictions.
4. The programmer builds that business value
5. Come back to step 1.

Because the student had to develop many applications and their users would be an important component, we used a *sub-iteration* inside the main iteration favoring the final-user participation. Our development process was:
1. The student confirms to the customer the application that is going to be built. In this moment, customer and student review the applications list to reconfigure it if necessary.
2. The student reviews all the information that has compiled about the business processes relative to that application with the customer, allowing him to correct the student if necessary.
3. Here begins a second iteration where a new role starts to play: the user of the application
   a. The student builds/modifies a prototype refactoring if needed.
   b. The student shows the prototype with limited functionality to the end-user of the actual application. At this moment the end-user can correct

features of the application or give ideas to improve the user interface or functionality. Obviously, only logical and realistic corrections and ideas will be taken into account.

   c. The student reviews the application functionality (old and new added functionalities suggested by the end-user) with the customer. The objective of this checking is to avoid disagreement in the process; all the end-user ideas or suggestions that don't fit the business processes must be commented to the customer.
   d. Returns to point **'a'** until the prototype finishes converting itself into an application with full functionality.
4. The application is used during one week, reporting bugs, corrections and all the modifications needed. During that week, all the documentation will be written if needed and the refactoring process will be finished. Obviously, any requirement change after that week will be taken into account and solved, but as a secondary task. This phase will finish with the final application integration into the system when all the requirements are accomplished.
5. After that week, the process returns to point **'1'**.

Kent Beck designed 13 main practices (Beck, 1999) that we are going to enumerate, summarize and explain, with regard to the needed tailoring:

- *Planning game*. This *"game"* has the objective of allowing the customer to decide the scope and timing of releases based on estimates provided by programmers. Those will only implement the functionality demanded by the customer. We could accomplish this practice limiting the estimates. As a FDP (Final Degree Project), it has to be made in at least one year, so, as we will see later in other practices, the schedule available had to fit that time.
- *Small releases*. It is important to have new releases often because although they don't have full-functionality, they mean a business value and in our case, also educational value. Each release is supposed to be delivered not far than 3 months. Due to the number of applications and to the one year schedule, the time release varied from 3 weeks to 2 months, depending on the developing module size.
- Metaphor. This metaphor, or set of metaphors, is a way of defining the shape of the system. It will describe how the system should work and it is a way of facilitating the understanding between the user and the customer. In our situation, we didn't exactly use a metaphor. The first interviews between the student and the customer were used to lay down the terminology.
- Simple design. Because the small amount of time we disposed of, this was a basic practice we had to follow: design the simplest functional solution and implement it in the right moment during the project. When talking about simplest, we talk about a design that communicates exactly what programmers want, without duplicating code and with the fewest possible classes and methods.
- Tests. There are a set of tests, decided by the customer, that are used for testing code functionality, especially after any modification. The only different thing we had to do in this practice was to insert other tests designed by our team, as well as the customer ones.

- *Pair programming*. This was one of the most problematic practices. It is known that pair programming usually implies an important quality improvement (Richard, 2007) and a significant decrease in the effort to be taken (Woi, 2006). With just one student in the team, he was the only responsible about all the code programming. It is obvious that it was impossible to avoid the effort to be made, that was going to rely on the student, but we realize that we could get the quality improvement in other ways. We decided to set a twice-a-week-code reviews, documenting in-depth all the code, and to support this practice with the next two: *refactoring* and the *collective ownership*.
- *Refactoring*. This practice was used in its original formula to fulfill the incapacity of use the *pair* programming practice. Through the *refactoring*, the student reorganized the code with the objective of removing duplicated code, make it simpler and flexible for facilitating later changes (Letelier et al., 2004). This practice had to be used very often to assure the software quality.
- *Collective ownership*. This practice supposes that any programmer of the team can improve any code. Being the project a FDP, only the student could modify the code. What we did, in order to reinforce the *pair programming* practice, was adapt this practice making the code accessible for the tutor but only in read-only mode. We gave the tutor permanent access to the code so he could revise it at least after each student code review, looking for features to improve. After each tutor review, if he had found code to improve, he planned a meeting with the student to discuss how to do it. The objective was to establish a conversation with the student so he realized the possible improvement.
- *Continuous integration*. This practice increases quality as side-effects of a change are quickly uncovered, what reduces the effort needed for fixing them (Coram & Bohner, 2005). It consists in integrating the new code in the system after no more than a few hours. In this case, we didn't need to make any tailoring.
- *On-site customer*. Because the FDP is inside the academic course, coinciding with some last course subjects, the student couldn't work full-time in the project. Because of this situation, we chose some days of the week (2-3) in which the student went to the companies' offices to work there and have the customer beside. Therefore, we adapted this practice getting a *temporary on-site programmer* one.
- *40-hour weeks*. As it was said before, the FDP wasn't the only task of the student so the 40-hour weeks were impossible. The student had an average of 20-hour weeks although there were more intensive ones.
- *Open workspace*. Because of the project situation, the workplace was dispersed in 3 different locations: the work-place in the companies' offices, the student home where he could work at weekends if needed and the tutor office at the Physics, Mathematics and Computation Sciences department where we had the meetings. In this case, we neither accomplished this practice.
- *Just rules*. Due to the XP tailoring we were trying to carry out, we quickly realized that this was going to be one of the most important principles: all adapted practices had to be tested, used and resulted useful.

The last main feature of the XP methodology is the roles distribution. According to original Beck proposal, there are 7 different roles: programmer, customer, tester, tracker, coach,

consultant and big boss. Our problem was that we only had 4 different roles with different authority level that should be respected: student, tutor, specialist and customer. The customer, obviously, adopted the original *customer* role. Because this project is focused on getting a real application for real enterprises, the customer also took the *big boss* role that supposes a coordination labor. Taking this second rule, we tried to get a highest involvement of the customer in the project.
As we are talking about an academic subject, it's logical that the *hardest* roles relied on the student: *programmer*, *tracker* (gives feedback of the project situation and of each iteration), *tester* (helps the customer to write tests, executes the tests and spreads the results to the team) and big boss. The last role was shared because we thought that would be interesting for the student to help coordinating a team, although he was almost all the team. The tutor was a teacher from the University that had to evaluate the project planning, development and final result. In this case he adopted the following roles: coach (process global responsible that provides guides for following XP practices and features), *tracker*, as a student support, and *big boss*, supporting coordination between customer and student. Finally, the professional role is a person from outside the university that gives continuous real feedback to the team. He adopted the *consultant* original role.

5. Project results

Before talking about the Project results, it is important to point out first which are the objectives, both educative and professional, and which are the conditions required to consider both achieved. When it comes to analyze the selected objectives, it is important to highlight that in the Computer Science degree at the CEU-UCH University, there are 3 projects for each of the last 3 years of the degree. That is the reason why in the two previous projects, the student was trained in some basic aspects.

5.1 Educative objectives

In 2002, the ACM and the IEEE-CS jointly proposed a basis for the education of software engineers called SEEK (Software Engineering Education Knowledge) (SEEK, 2002). SEEK defined 10 basic topic areas that are going to be analyzed below, indicating which were addressed in earlier courses and which were the objectives we proposed ourselves. The basic SEEK areas covered in early courses were: fundamentals (understanding for *fundamentals* mathematics and the foundations of computing and engineering like requirements specification or risk management), software design, software construction, software verification and validation, software engineering process and software quality.
In this last project, four SEEK areas were expected to be practiced: professional practice, software requirements, software evolution and software engineering managements. For these areas, partial success was achieved:

- Professional practice. In order to train good professionals, it's important for the students to have real contact with their future working environment. The tailored "temporary programmer on-site" XP practice, together with the fact that the student was the one who always interacted with the customer (always under the tutor guide), has allowed the student to have that real contact. It's true that the contact hasn't been all the real it could be in a normal project because the project is an academic one, but it has been enough for considering the success achievement of the area. It would have been great to treat issues of the contract.

- Software requirements. In the preceding projects, the students were at the same time the programmers and the customers which specified the requirements, reason why they always knew what they needed or wanted and how they wanted, rarely changing those requirements. As it has been said, XP perfectly adapts to changes reason why it has been a great tool helping the student to deal with two problems: changing requirements and change process. The achievement of this area has been partial achieved. Because the student used to be the customer, in some occasions he took for granted requirements or applications features that later didn't agree with the real customer decisions. It's very important for the student to be humble, realizing that the real customer is the one who dictates the requirements.
- Software evolution. This area is focused on change and how to deal with it. Change is understood as a stage in an activity related to maintenance, reengineering or reuse. The main problem in this area is the time limitation and the project size. For dealing with the software evolution in such a big project, more time is needed.

Regarding the student, and regardless the educative objectives, the motivation and the involvement grade of the student in this kind of real projects are higher than in non-real ones. We checked this reality comparing the scheduling fulfilling of the student of this case study with the scheduling of some of his classmates. In our case, the milestones dates were achieved at time with only one delay at the beginning of the project. The classmates that worked on non-real projects always tended to delay the milestone achievement a week at least.

5.2 Professional objectives

The main professional objective was the development of a high modular ERP. Between all the modules, three were chosen as the key and fundamental ones: the model management, the stock management and the invoice application. The student began with some complementary modules needed like the customer or supplier applications (only inserting, deleting and editing) focusing later on the 3 main modules and finishing them completely. Due to the lack of time, only one more, of the possible modules, was developed: the one focused on reporting tasks and summary information.
The basic objective was achieved (the 3 modules development), but it's true that other modules that would give business value and usefulness to the ERP were not fully developed. These modules (highlighting the BI reporting application) were just designed and, after the FDP evaluation, began their development and are nowadays in the testing phase.

From the university point of view, the methodology used gave us important opportunities:

- It allowed the University to prepare a student for real job world, training and guiding him in a way other universities actually don't do.
- The tailor-made XP designed gives the University a methodology to reuse and improve in following years.

- The successful conclusion of the project will impulse more collaboration deals between our university and other SME of our country. Obviously, as we are talking about a project framed in a final degree project, which implies little time, the adaption of the agile methodology played a fundamental role giving us the flexibility needed to face this project successfully.

5.3 Project functionalities

The tailor-made XP gave as final result a real and actually in use ERP that automates the main business processes with no charge. It's called SIGATYC and it's a high modular system that gives to the end-users the opportunity of working with a set of applications to which they have access. Obviously, not all the users can access to all the applications. There are strict security measures, at application and database levels, that avoid unauthorized use of private data.
After the authentication process, SIGATYC begins with what the student called ***"Control Panel"***, from which the end-users can choose a specific application.

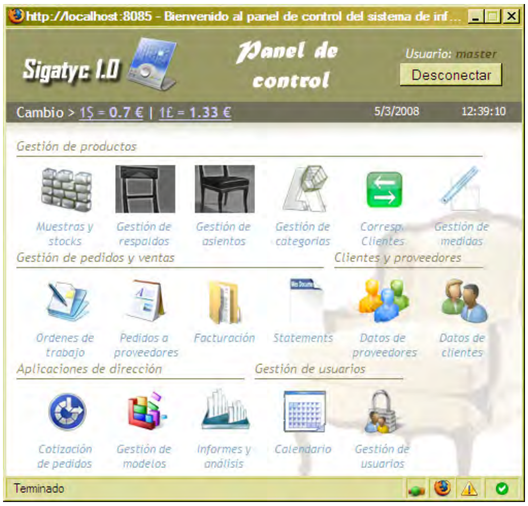
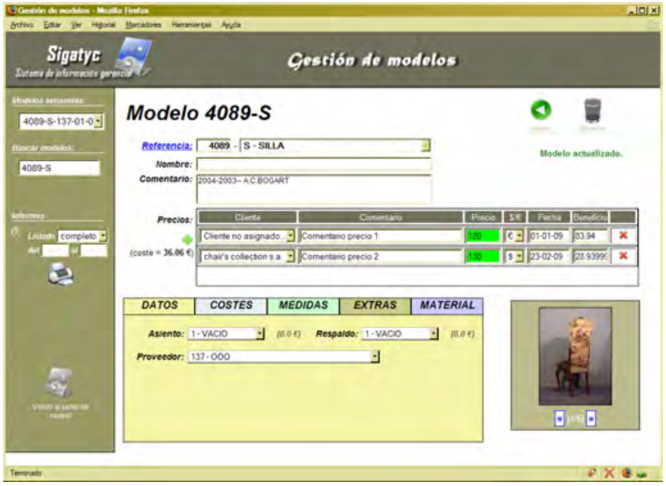


Fig. 1. Control Panel where each icon represents an application

Those applications are classified in 5 main groups:

1. PRODUCTS. This category groups all the applications related to product data management. Inside this group the Stock application can be found. This was one of the fundamental applications to carry out.
2. ORDERS AND SALES. This is a group prepared for further developments, in which end-users would find applications related with orders made by customers and to suppliers, their actual situation and the invoices and delivery notes generated after selling or receiving products. Only the invoice application is available. In the future, the invoice application data will be fed by the customer orders entries.

3. CUSTOMERS AND SUPPLIERS. In this group there are only two applications that records and identifies the existing customer and suppliers. Although this is an interesting group for developing applications focused on reinforcing the company relation with the customers and suppliers, the CC company wasn't interested in this kind of applications.
4. MANAGEMENT. This group is exclusively for the manager. From here he can manage prices, get detailed reports about sells, about customer purchases, etc.
5. USERS MANAGEMENT. These applications will be used for joining new user, delete old users or modify their data and application listing to which they have access.


Gestión de modelos - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
Sigatyc
Gestión de modelos
DATOS
COSTES
MEDIDAS
EXTRAS
MATERIAL
Terminado

Modelo 4089-S

4089-S-137-01-0

4089-S

Listado: completo

Referencia: 4089 - S - SILLA
Nombre:
Comentario: 2004-2003-- A.C.BOGART

Modelo actualizado.

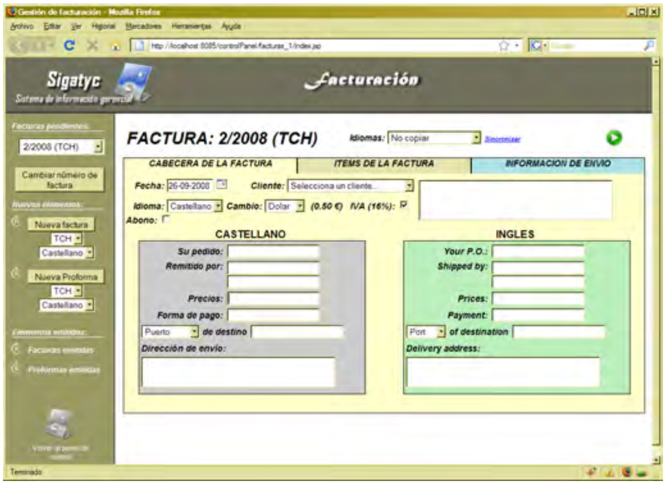
Precios: (coste = 36.06 €)

Cliente	Comentario	Precio	$/€	Fecha	Beneficio
Cliente no asignado...	Comentario precio 1	120	€	01-01-09	83.94
chair's collection s.a.	Comentario precio 2	130	$	23-02-09	28.9399

Asiento: 1 - VACIO (0.0 €) Respaldo: 1 - VACIO (0.0 €)
Proveedor: 137 - OOO

Fig. 2. Model management application with an image gallery web widget

Although the applications are grouped in five different groups, the access rights are granted by application, not by group. In this way, the end-users have a better delimited set of applications they can use.
About existing applications, a remark should be made about "Customers and suppliers" ones. Extended ERPs are characterized for including two well known extensions called *Supply Chain Management* (SCM) and *Customer Relation Management* (CRM). These two extensions enable effective third-party business relationships between the organization, suppliers and the customers (Hossain et al., 2002). Despite the possible benefits both can imply, SIGATYC doesn't include these kinds of applications because of direct company decision.
About the accounting application, this wasn't developed because the company already had a specific independent software package focused on this area. What the student did after the FDP evaluation, was to connect the ERP developed with the data files of the accounting application, allowing the reports applications to access and read the finance data, and connecting both software packages where necessary. This was an important part of the project because the CC company didn't want to lose the money investment made in the accounting application.



http://localhost:8085/controlPanel/facturas_1/index.jsp

Facturas pendientes: 2/2008 (TCH)

Nueva factura: TCH, Castellano

Nueva Proforma: TCH, Castellano

FACTURA: 2/2008 (TCH)

Idiomas: No copiar

Fecha: 26-09-2008 Cliente: Selecciona un cliente...

Idioma: Castellano Cambio: Dolar (0.50 €) IVA (16%): ☑

Abono: ☐

Puerto de destino

Port of destination

Fig. 3. Invoice application. One of the most important applications for the company.

6. Conclusions

The result of using a tailored agile methodology for the design, development and implementation of an ERP by a student of last course could be described as satisfactory. The student, in his relation with the customer and the company, has acquired knowledge impossible to acquire in the classroom. He had to plan the project, being able to experiment the problems and situations that a bad schedule or an undervalued task can cause. He has learnt to treat with a real customer, who has changed the requirements repeated times forcing him to be more flexible (to what the tailored XP has help), with whom the student has been in constant and direct contact communicating via phone or mail using a correct and appropriate language and expressions. All these experiences have helped to train the student as a future Computer Science professional. As for the company, it has obtained the basic portion of the tool that will help in its material management and business process automation.
However, there exist some aspects to improve:

- Project size. Because of the time limitation, there are two options: to develop smaller and simpler projects, which allow the long use of all the XP practices by just one student or create at least two little teams (of two students each one) which should work in parallel but in constant communication between them and with the client. This last option would complicate the client interaction. It's known that clients don't fully understand the benefit of regular developer-client interactions and don't want to be bothered.
- The available time. The time that a final course Project offers is limited: just one year at CEU-UCH studies. Such a short period has two important limitations: on the one hand it limits the size and complexity of the project, on the other hand, it doesn't allow the student to observe the software evolution. As for the time, it is

important to stand out that, although the FPD evaluation takes in a year, the project goes on until it's completely finished.

- The team size. As it has been indicated, it would be preferable to have groups of two students, allowing the exchange of them between the different groups in order to use the "*pair* programming" practice and allowing the students to work with more students. It's important to teach the student to face up dynamic teamwork because it's one of the working characteristics more appreciated by firms.

For further projects, smaller information systems for two student teams will be proposed. For improving the tutor evaluation process and for eliminating part of the documentation, new and different practices can be applied: the mutual explained qualification between the students after a module finalization or evaluation of the degree of ability acquired in the resolution of changes along the project. The objective is to use different kinds of deliverables different from documentation.

7. References

Alleman, Glen B. "Agile Project Management Methods for ERP: How to Apply Agile Processes to Complex COTS Projects and Live to Tell About It", Extreme Programming and Agile Methods: XP/Agile Universe 2002.

Andreu, R; Ricart, J.E.; Valor, J. "Strategy and information systems", Ed. McGraw-Hill, Segunda edición. 1996. ISBN: 8476156669

ANIEME (National Association of Furniture Industrials and Exporters in Spain), "Furniture Sector Results Report 2006". Available online: http://www.anieme.com/actualidad/datos-sector.aspx

Beck, Kent. "Embracing Change with Extreme Programming", Computer, vol. 32, pp. 70-77, 1999.

Beck, K et al. Manifesto for Agile Software Development, 2001. Online: http://www.agilemanifesto.org

Beck, K. "Extreme Programming Explained: Embrace change", 2nd Edition, Addison-Wesley, 2005

Carr, N. "IT doesn't matter", Harvard Business Review (May), pp. 1-10, 2003

Coram, M. Bohner, S. "The impact of Agile Methods on Software Project Management", Proceedings of the 12th IEEE international Conference and Workshops on the Engineering of Computer-Based Systems, 2005

DGPYME; Industry, Turism and Comerce Spanish Department, "PYME of the 2007 description". Available online: http://www.ipyme.org/NR/rdonlyres/D86BB6D9-EB28-4DFC-BCC7-F10F5008E787/0/Retrato2007.pdf

Eurostat Statistical Books. "European Business. Facts & Figures", 2007, available on line: http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-BW-07-001/EN/KS-BW-07-001-EN.PDF

H. Canós, José, Letelier, Patricio and Penadés, Mª Carmen. "Agile methodologies for the software development", 2004

Hossain, Liaquat; Patrick, Jon David and M.A. Rashid, "Enterprise Resource Planning: Global Opportunities & Challenges",. Idea Group Publishing, 2002.

I. Sommerville, "Software engineering", 8th ed. Harlow, England; Pearson Education, 2006.

Kee, Woi H., "Future Implementation and Integration of Agile Methods in Software Development and Testing", Innovations in Information Technology, 2006, pp. 1-5, Nov. 2006

King, John L. (editor) and Lyytinen, Kalle (editor), "Information Systems. The State Of Field", ed. Willey & Sons, 2006.

Kwahk, Kee-Young, "ERP Acceptance: Organizational Change Perspective", Proceedings of the 39th Hawaii International Conference on System Sciences – 2006.

Lawrence, Richard, "XP and Junios Developers: 7 Mistakes (and how to avoid them)", Proceedings of AGILE 2007, IEEE 2007.

Maurer, Frank, and Melnik, Grigori, "Agile Methods: Crossing the Chasm", IEEE 29th International Conference on Software Engineering (ICSE'07 Companion)

Ming Huo, June Verner, Liming Zhu, Muhammad Ali Babar, "Software Quality and Agile Methods", Proceedings of the 28th Annual International Computer Software and Applications Conference 2004 (COMPSAC'04).

Neill, C. J., and Laplante, P. A. "Requirements engineering: the state of the practice". IEEE Software 20, 6 (Nov./Dec. 2003), 40-45;

Patricio Ramírez Correa. "Rol and contribution of the enterprise resource planning (ERP)", PhD thesis, 2004. University of Sevilla.

Royce, Winston W., "Managing the Development of Large Scale Software Systems", Proceedings of IEEE WESCON, pp. 1-9, August 1970.

Sandra Sieber, Josep Valor, Valentín Porta. "Information systems in the Enterprise", McGraw-Hill. 2007. ISBN: 9788448140069

Software Engineering Education Knowledge (SEEK), 2002. Second Draft, December. Avaiable online: http://sites.computer.org/ccse

Sumrell, M. "From Waterfall to Agile – How does a QA Team Transition?", Proceedings of AGILE 2007, IEEE 2007.

USA Government, "The Small Business Economy. A Report to the President (For data year 2007)", United States Government Printing Office, Washington, 2008. Available online: http://www.sba.gov/advo/research/sb_econ2008.pdf

W. Ambler, Scott, "The Object Primer: Agile Model-Driven Development with UML 2.0", 3th ed. Cambridge University Press, 2004