An Approach of a Technique for Effort Estimation of Iterations in Software Projects

José Antonio Pow-Sang
Pontificia Universidad Católica del Perú
Perú
japowsang@pucp.edu.pe

Enrique Jolay-Vasquez
Carnegie Mellon University
USA
ejolay@andrew.cmu.edu

Abstract

The estimation of effort and cost is still one of the hardest tasks in software project management. At the moment, there are many techniques to accomplish this task, such as Function Points, Use Case Points and COCOMO, but there is not much information available about how to use those techniques in non-waterfall software lifecycles such as iterative or spiral lifecycles projects.

This paper shows the results obtained when applying a technique to estimate the effort of each construction iteration in software development projects that use iterative-incremental lifecycles. The results were obtained from software projects of a fourth-year course in Informatics. The technique proposes the use of Function Points and COCOMO II.

1. Introduction

The growing complexity of software development that was caused by the well known "Software Crisis" has been addressed through innovative methods, methodologies, techniques and, paradigms to minimize its impact. The scope of those proposals is not limited exclusively to activities that are related to the systems development itself because it also involves management activities. One of these activities is the estimation of software projects.

Despite the complexity of the software estimation, sometimes it is only performed by an estimation expert himself. In the last decades, some techniques have been developed to estimate the effort of complete software projects such as Function Points [24] and Use Case Points [10]. Even more, these techniques – that are independent of the final development technology – were conceived to be applied in the development of systems that were based on the structured paradigm with a classic or waterfall lifecycle Royce [19] being hard to utilize them for an object-oriented development and iterative-incremental lifecycles that are so used lately. In fact, it also seems interesting that these estimation techniques exhaustively use the information produced by the application of some techniques such as Use cases' method, for their own purposes.

Although there is not much work related to this problem, one interesting work is an approach proposed by Mogagheghi et al [12]. They define a Use Case Point technique adaptation for incremental software development projects.

Acknowledging the advantages present in the experiments involving students [5] and considering the problematic exposed lines above, this article shows an experience n the application of a technique based on Function Points (FP) that proposes a calculation of an estimation of the effort for every iteration. This experience was conducted on junior students of the Computer Engineering Program at *"Pontificia Universidad Católica del Peru (PUCP)"* that were enrolled in Software Engineering course in Spring 04, Fall 04 and Spring 05.

This article has been divided in 7 different parts: Section 2 shows a brief summary of the Function Points technique, COCOMO II and its relation to Use Cases; Section 3 shows the technique to estimate and plan the software development for iterative-incremental lifecycle; Section 4 presents the information of the projects with students; Section 5 shows the obtained results; Section 6 develops a discussion of the results using both techniques and finally, conclusions and future projects related to this experience are presented.

2. The Estimation Techniques and Uses Cases.

This section shows a brief summary of the Function Points Technique and COCOMO II as well as its

relation with uses cases. In addition, an overview of similar projects developed in relation to effort estimation is presented.

2.1. Function Points Technique

Function Points [24] were introduced by Albrecht [1] and its purpose is a software measurement qualifying it by the functionality that provides externally based upon the system logical design. The objectives of this technique are:

- To measure user requirements and what user really gets.
- To provide a software measurement regardless the technology utilized in the system deployment.
- To provide a size metric for quality analysis and productivity purposes.
- To provide an alternative for software estimation.
- To provide a normalization factor to compare different software.

The analysis of Function Points has been conducted considering five basic external system parameters: External Input (EI), External Output (EO), External Query (EQ), Internal Logic File (ILF) and, External Interface File (EIF).

Using these parameters, Unadjusted Function Points (UFP) are determined. After that, an adjustment factor is applied to UFP. This factor is obtained based on subjective valorizations of the system application and its environment

2.2. COCOMO II and Function Points

COCOMO II, proposed and developed by Barry Boehm [4], is one of the best documented and utilized models for cost estimation. This model determines the effort, time and, schedule required when planning a software development project. This estimation is based on the size of the project which is expressed in the number of code lines estimated to develop a software product.

Due to the complexity of software projects, the COCOMO (*COnstructive COst MOdel*) original model was modified obtaining a new one: COCOMO II. This new model can estimate the effort, time and, schedule needed, using three different sub models: Applications Composition, Early Design and Post Architecture models.

The model proposes a set of cost drivers which indicates the context in which the project is currently set. These drivers are used to determine the Effort Adjustment Factor (EAF), which is utilized to compute the necessary effort to complete the project. The relation between these drivers and the post architecture model is shown in Table 1

Table 1. Cost Drivers for the Post-Architecture Model in COCOMO II

Cost Driver	Description
RELY	Required software reliability
DATA	Database size
CPLX	Product complexity
RUSE	Reusability required
DOCU	Documentation match to lifecycle needs
TIME	Execution time constraint
STOR	Main storage constraint
PVOL	Platform volatility
ACAP	Analyst capabilities
PCAP	Programmer capabilities
PCON	Personnel Continuity
AEXP	Applications experience
PEXP	Platform experience
LTEX	Programming language experience
TOOL	Use of software tools
SITE	Multisite Development

A new feature included in the new model is that it can determine the effort and time of a software project using Unadjusted Function Points, which is a big advantage given that in most cases, it is hard to estimate the number of code lines of the software that will be built even more when there is almost nothing or no previous experience in software projects whatsoever. Therefore both models – Function Points and COCOMO II – are very compatible and complementary.

2.3. Use Cases and Function Points

Uses Cases were introduced in 1987 as an Objetory technique tool. Its utilization in Software Engineering Processes was proposed by Ivar Jacobson and published in his book "Object-Oriented Software Engineering" [9].

Nowadays, use cases' utilization has been extended due to its inclusion in UML [13][20], so that its usage in Object-Oriented Software development has turned into a "must".

David Longstreet, in one of his articles [11], indicates that the analysis of Function Points can be applied in conjunction with uses cases in a very straightforward way. This will improve the quality of the documents of user requirements. It will also improve the estimation of a software project. The application of the Function Points technique can verify and validate the content of a specification document of software requirements.

It seems to be interesting that when applying both techniques, the count of Function Points can be updated every time the use cases change so that the impact of a specific use case in the estimation of the complete project development can be determined.

Thomas Fetcke [7] proposes a way to utilize the Function Points technique in along with the Jacobson's OOSE methodology [9]. The relation that he shows is based on the use cases and the class diagrams of the analysis proposed by that methodology. In his proposal, he does not specify the type of lifecycle that should be followed in his technique, so that it can be inferred that the waterfall lifecycle should be used.

3. A technique for planning and estimating iterations of a software project using Function Points

According to [14], a technique to estimate and plan iterations was proposed based on Function Points and COCOMO II. It was also used for this experience.

The technique has 2 phases: First, the uses cases needed for every iteration has to be determined. Secondly, the estimation for each iteration has to be estimated. In the following subsections, the activities involved in each phase are mentioned in detail.

3.1. Determining the Use Cases needed per iteration (phase 1)

The objective in this phase is to determine the use cases that are to be implemented per iteration. For this purpose, it is previously necessary to have identified and specified every use case present in the software. Those use cases also have to be included in the specifications of software requirements (for use cases specifications, recommendations in [2], [3] and, [23] may be used as well as for specifications of software requirements [8] and [18] may be used).

For this activity, it is proposed that a new diagram is used. This diagram will be called "precedence diagram" in which the preconditions of every use case contained in the specifications are presented graphically. The idea of this diagram was taken from Doug Rosengberg [21], who proposes the usage of a similar diagram, specifying the relations: "precedes" and "invoke" to determine user requirements. The figure 1 shows an example of a precedence diagram.

The diagram will be used to know what use case needs a functionality or information that is managed or implemented by some other use case. Indeed, this will be useful in determining what use case must be programmed earlier in such a way that what is necessary in a uses case has already been developed in a previous iteration.

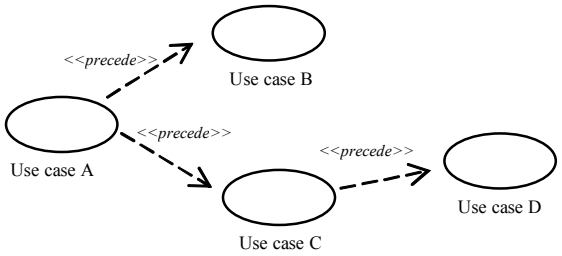


Figure 1. Precedence diagram example

Consequently, the use cases that are on the left of the diagram will be implemented before the ones that are on the right. In figure 1, "Use Case A" will be implemented before "Use Case C".

It is important to highlight that in this diagram, Included and Extended Use Cases have not been considered due to they can be part of other use cases that refers to them.

3.1. Effort estimation per iteration (phase 2).

After Phase 1 is completed, in phase 2 Function Points technique and COCOMO II are proposed to determine the effort that will be needed for each iteration.

First of all, it is necessary to determine the Unadjusted Function Points (UFP) per iteration. The criterion followed to adapt this technique for iterative-incremental lifecycles is to consider that the result of the computation of the UFP of the complete project without considering iterations (Total_UCP) must be equal to the sum of the Unadjusted Function Points computed per iteration separately

$$Total\_UCP = \sum_{i=1}^{n} UCP(i)$$

One of the most important contributions of this proposal is to determine the UFP of the Internal Logical Files (ILF) and the External Interface Files (EIF) for each use case. For this purpose, this technique proposes the usage of the following formula (*File_UCP(j)* = Unadjusted Function Point for a use case "j" due to files ILF/EIF, *TCU(i)* = total of use cases that uses a ILF/EIF "i", *Weight(i)*= Weight due to the complexity of ILF/EIF "i", *i*= ILF/ EIF used in use case "j" and j= use case involved)

$$File\_UCP(j) = \sum_{i=1}^{n} \frac{1}{TUC(i)} xWeight(i)$$

Using the results obtained from the previous formula and knowing which use case will be developed for each iteration, the UFP will be determined by means of the files present in the iteration. Next, the UFP corresponding to the transactions will be added. To accomplish this task, the following formula will be used (*i*=iteration, *TUFP(i)*=Total of UFP for iteration "i", *File_UFP(j)*= UFP for a use case "j" due to ILF/EIFs, *Trans_UFP(j)*= UFP due to transactions (EI,EO,EQ) and *j*=use case subject to be implemented in an iteration "i".

$$TUFP(i) = \sum_{j=1}^{n} \left[File\_UFP(j)\right] + \sum_{j=1}^{n} \left[Trans\_UFP(j)\right]$$

The next step is using COCOMO II and the UFP of each iteration, the effort expressed in man-month is obtained per iteration and using this value, time and human resource needed to complete the whole project can be estimated

It is important to mention that the context of the project is subject to change from one iteration to the other (knowledge of the development platform, integration of the developer team) so that it could be necessary to re-estimate the effort required by next iterations. This can be accomplish by reviewing files (ILF/EIF) and transactions (EI, EO y EQ) in case of change of requirements and recalculating cost drivers proposed by COCOMO in case of organizational or contextual changes.

Although the technique suggest using COCOMO II to calculate effort in man-month per each iteration, the results shown in section 5 of this paper only use COCOMO's EAF factor to get estimated effort in second iteration based on the real effort obtained in first iteration.

4. Projects Information

The projects, that the results are shown in section 5 of this paper, were developed by junior students (4th year) of Informatics Program that were enrolled in the Software Engineering course in Spring 04, Fall 04 and Spring 05. This is a core course from the Software Engineering program area. In this subject the students had to develop a software project. Its duration was fourteen (14) weeks.

In Spring 04, the students did not use the whole technique, they only use precedence diagram to plan construction iterations. The objective was to collect empirical data to determine if the technique could be useful in software projects.

Knowing the good results obtained in Spring 04, two objectives were clearly defined in Fall 04 and Spring 05 terms: Firstly, the students had to apply the Function Points technique to estimate the effort of the second and third iteration based on their own effort in previous iterations. Secondly, it was to collect empirical data and to observe the results in the application of the technique.

4.1. The students and work teams

The students were divided in groups of 11 or 12 people. The groups were formed based on the following criteria: Same proportion of students that had excellent academic performance ("A/A+" students) among the groups and equal number proportion of men and women. These criteria had been considered critical in the success of a software project based on empirical data from past experiences that department's professors had.

The table 2 shows previous knowledge and experiences that the students had prior to begin the academic Term.

Table 2. Previous knowledge and experiences that the students had prior to begin the academic Term.

Characteristic	Knowledge and/or Experience
Programming Language / Programming Environment	• Java, C#, Pascal, C and Prolog. • No previous experience using Delphi.
Databases	• Oracle 8. • No previous experience using MS SQL Server.
Analysis and Design Techniques	• Structured and Object-oriented
Project Management	• Experience in managing developing projects that included short software programming projects (Work Team of 3 or 4 students). • No previous planning and estimation experience.

The vast majority of courses in the Informatics program at PUCP focus software projects as an application of theoretical concepts but they do not demand the application of planning and estimation techniques when developing software projects.

4.2. Project Characteristics

Every team had to develop a software system. The topics for the projects were different in every academic term to avoid plagiarism. The table 3 shows the topics per term. Although the projects were different, both of them correspond to a type of information system which input/output processes are similar. Therefore, results

obtained from the three different academic periods are comparable.

Table 3. Topics chosen per academic Term.

Semester	Topic
Spring 04	Department-store system
Fall 04	Fast-food restaurant system
Spring 05	Library system

The methodology used by every team was RUP[18]. The duration of the project was 14 weeks which is the length of an academic term at PUCP. Before beginning the construction phase, each team had to finish the Software Requirements Specifications document (SRS)[8] as well as use cases. Client/server 2-tiered [22] architecture was used and validated through a prototype.

Every team had to develop three iterations of the construction phase. Each iteration took exactly 2 weeks of work. The number of use cases developed and implemented in the second and third iteration was based on the effort in previous iterations.

A teacher assistant (TA) was assigned to every team. This TA was in charge of the team and helped them throughout the project supporting the students to obtain an adequate SRS for business processes of a real company. It is necessary to mention that TA are constantly working in real software projects at Peruvian companies of software development with an average of 7 years of experience in the field.

The following table shows the software used for the Project development.

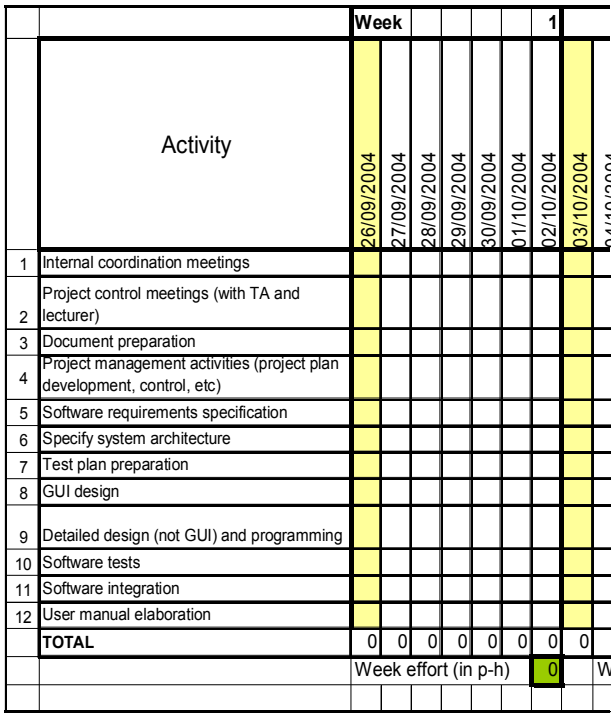
Table 4. Software used for the Project.

Item	Software
Programming Language	Delphi 7.0
Operating System	Microsoft Windows 2000
Database	Microsoft SQL Server 2000
Modeling Software	Rational Rose
Documenting Software	MS Office 2003

4.3. Data Collection

In a weekly basis, students had to turn in a report of the hours needed to work in an activity per day. The format was a MS Excel worksheet as the one as shown in figure 2:

It is important to mention that students were told that the grading was going to be based upon the accomplishment of every use case identified for each iteration. In addition, they were informed that final grade was not influenced by the number of hours invested in the project. This was specified to ensure the honesty in the effort recorded by each team member.



		Week						1		
	Activity	26/09/2004	27/09/2004	28/09/2004	29/09/2004	30/09/2004	01/10/2004	02/10/2004	03/10/2004	
1	Internal coordination meetings									
2	Project control meetings (with TA and lecturer)									
3	Document preparation									
4	Project management activities (project plan development, control, etc)									
5	Software requirements specification									
6	Specify system architecture									
7	Test plan preparation									
8	GUI design									
9	Detailed design (not GUI) and programming									
10	Software tests									
11	Software integration									
12	User manual elaboration									
	TOTAL	0	0	0	0	0	0	0	0	
		Week effort (in p-h)						0		W

Figure 2. MS Excel worksheet to register worked hours

By means of class meetings with each team, the criterion to fill out the worksheets was ensured to be the same for everyone in the group. In the beginning, there were some problems due to some students that recorded more hours than the real ones used in the project.

5. Results

The proposed technique has been use in software projects in which participates only one person having encouraging results (an example of its application is documented in a master thesis document [16]). The obtained results encouraged us to try the proposal in new software projects including those in which more than one person participates.

The following table shows the obtained results of each team in the Spring 2005 term.

Table 5. Results of teams in spring 2005 term.

Iter.	Team A		Team B		Team C	
	UFP	Real Effort*	UFP	Real Effort*	UFP	Real Effort*
1	191.69	307	138.00	248	313.00	151.04
2	200.61	195	234.17	238	170.00	139.29

* Measured in men-hours

Nowadays, there are no people in Peru that are certified in the technique of Function Points so results could not be validated although directions and recommendations included in the Function Points [24] manual were followed. In addition, the documentation was reviewed by someone who has two years of experience in Function Points and was a Master student from the Software Engineering program at Technical University of Madrid in Spain where this technique is mandatory and applied in a final software project.

Team members had worked in projects in previous semesters, and this is why the context of interpersonal relationships was the same in all iterations. Besides, it can be observed that the only change in the context of each iteration was the programming tool knowledge and the application experience (LEXP and AEXP cost driver).

A magnitude of relative error (MRE) was utilized to show results in tables 6, 7 and 8 of this section where y= real effort and $\hat{y}$= estimated effort [6]

$$MRE = \frac{\left| y - \hat{y} \right|}{y}$$

To compute the estimated effort for iterations, the following formula was used where *est_effort(i)*=estimated effort, *EAF(i)*= COCOMO's EAF factor for the iteration "i" and, *real_effort(j)*=real effort for iteration "j".

$$est\_effort(i) = \frac{EAF(i)}{i-1} \times \sum_{j=1}^{i-1} \frac{real\_effort(j)}{EAF(j)}$$

The results obtained of the estimation effort and the MRE for the project in the second iteration in Spring 05 are presented in tables 6, 7 and 8. For more information the reader may find Spring 04 and Fall 04 results included in [15] [17] (in Spanish).

Table 6. MRE of team A in spring 2005 term

Iter.	COCOMO's EAF Factor	Real Effort*	Estimate Effort*	MRE
1	1.46	1.602	-	
2	0.8	0.972	0.878	9.72%

* Measured in men-hours

Table 7. MRE of team B in spring 2005 term

Iter.	COCOMO's EAF Factor	Real Effort*	Estimate Effort*	MRE
1	1.46	1.797	-	
2	0.8	1.016	0.985	3.11%

* Measured in men-hours

Table 8. MRE of team C in spring 2005 term

Iter.	COCOMO's EAF Factor	Real Effort*	Estimate Effort*	MRE
1	1.46	2.072	-	
2	0.8	1.220	1.135	6.96%

* Measured in men-hours

From the results obtained it is observed that in the second iteration the MRE is less than 10% in all teams meaning that the utilized technique was accurate. Unfortunately, real effort of third iteration was not registered by the students and that is why they are not included in tables 6, 7 and 8.

The following table shows the results obtained in three academic terms.

Table 9. MRE of each team by academic term of second iteration

Term	Team	Number of Members	Second Iteration		MRE
			Real Effort*	Estimated Effort*	
Spring 2004	A	10	2.533	2.234	11.79%
Fall 2004	A	11	1.556	1.669	7.28%
	B	12	2.193	2.297	4.72%
Spring 2005	A	10	0.972	0.878	9.72%
	B	10	1.016	0.985	3.11%
	C	9	1.220	1.135	6.96%

* Measured in men-hours/UFP

From the results obtained in the 3 semesters it is observed that in the second iteration the MRE is less than 12% in all teams meaning that the utilized technique was accurate.

It is important to mention that EAF factors used to calculate estimated effort in table 9 were the same for all teams: 1.46 for first iteration and 0.8 for second iteration.

Real effort showed in previous tables does not consider effort of internal and control meetings (row 1 and 2 of figure 2) because we only wanted to consider effective work utilized in software construction (detailed design, programming and testing).

6. Final Discussion

Although the obtained information has been collected from six projects and the results obtained are encouraging for the proposed technique, it cannot be stated that this technique is reliable for some other context. Some comments coming from students in regard to thus technique are

- The technique was a useful tool to measure the size of the software each team had to implement per iteration.
- The recorded effort in the first iteration was useful to compute and estimate next iteration's effort.

It can also be observed that the utilized effort of both teams is different. For instance, in the second iteration, in spring 2005 term, Team A required 0.972 man-hours/UFP. Unlike that, Team B required 1.016 man-hours/UFP. Something similar occurs for all the teams in three semesters (it is shown in table 9).

The fluctuation that is shown in the previous table is due to some factors such as the number of students per team and the capacity of teamwork being some of them hard to quantity.

It is hard to generalize the obtained results to other contexts different than the academic one. The advantage of experimenting with students is that some factors and/or variable can be controlled which is difficult to do in real-world industry projects.

Some of them are:

- Knowledge and expertise of members of the development team. In the real world industry, it is hard that every team member has the same experience in project development. In some cases, team members do not have the necessary knowledge, for instance, about Object-Oriented analysis and design as well as implementation.
- People Reassignment. In the industry, it is frequent that team members are reassigned throughout the project. For instance, they can be assigned to some other project. For this experience, there were no reassignments.
- Change of requirements. It also happens throughout a project. This factor can be controlled in an academic project.
- Sequence of requirements implementation. In this experience, the implementation was conducted following a precedence diagram, which sometimes cannot be done in a real-world project due to the interdependence between construction precedence and user requirements.

Obtained results could be applied in the industry under certain circumstances: A team with a lot of expertise in application development that has a project in which they have to use a new tool and programming language.

7. Conclusions and Future Work

The majority of approaches to estimate projects, even when defining themselves as independent from technologies and lifecycle, have a characteristic highly influenced by waterfall lifecycles. In spite of their validity for some other approaches, for example iterative-incremental lifecycles which does not offer any guide to accomplish the adaptation.

The present article presents a technique that is based on Function Points to estimate the effort in projects that utilize iterative-incremental lifecycles. In addition, results obtained from the application of the technique in students' projects are shown. The advantage of working with students is that many factors and variables that can affect the project (knowledge, level of expertise, context of iteration), can be controlled.

Results obtained from the experience are very encouraging because the proposed technique provides a relative error between the estimated effort and real effort less than 12%. Every team used data coming from previous iterations to estimate the effort needed for next iterations. Those results are not final due to the need of more testing not only involving students but also some real-world projects.

The future work in regard to this article is:

- To adapt the proposed technique in such a way that the files are replaced by UML-classes to have an adaptation to an object-oriented perspective.
- To conduct experiences in which the sequence diagram cannot be followed. Sometimes it happens that the priority of the required implementation in industry projects is very different
- To adapt the proposed technique to other techniques such as Use Case Points.
- To conduct experiences involving students to compare the effectiveness with other techniques such as COSMIC-FFP and Use cases points to estimate the effort of iterations in software projects.

8. Acknowledments

The authors would like to thank Dra. Ana María Moreno, professor from Technical University of Madrid, for her invaluable comments to prepare Fall 04 experience.

Besides, we would like to thank all students who participated in the projects and the anonymous reviewers for their comments. These comments will be

considered to improve this proposal and included in next experiments.

9. References

[1] Albrecht, A. J. Measuring Application Development Productivity, IBM Applications Development Symposium, Monterey, CA, USA, 1979.

[2] Bittner, K., Use Case Modeling, Addison-Wesley, USA, 2003.

[3] Bittner, K., "Why Use Cases Are Not Functions", http://www.therationaledge.com, USA, 2000.

[4] Boehm, B., et al., Software cost estimation with COCOMO II, Prentice-Hall, USA, 2000.

[5] Carver, J., Jaccheri, L., Morasca, S., Issues in Using Empirical Studies in Software Engineering Education, Proceedings METRICS'03, IEEE Computer Society, USA, 2003.

[6] Conte, D., Dunsmore, H., Shen, V., Software Engineering Metrics and Models, Benjamin-Cummings, Menlo Park CA, 1986.

[7] Fetcke, T., Bran, A., Nguyen, T., Mapping the OO-Jacobson Approach into Function Point Analysis. Proceedings of TOOLS-23'97, IEEE, USA, 1997.

[8] IEEE Computer Society, IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications, The Institute of Electrical and Electronics Engineers, USA, 1998.

[9] Jacobson, I., Object-Oriented Software Engineering. A Use Case Driven Approach, Addison-Wesley, USA, 1992.

[10] Karner, G. Metrics for Objectory. Diploma thesis, University of Linköping, Sweden. No. LiTH-IDA-Ex-9344:21. December 1993.

[11] Longstreet, D., Use Case and Function Points, http://www.softwaremetrics.com/, Longstreet Consulting Inc, USA, 2001.

[12] Mohagheghi, P., Anda, B., Conradi, R., Effort Estimation of Use Cases for Incremental Large-Scale Software Development, Proceedings ICSE 2005, ACM-Press, USA, 2005.

[13] Object Management Group, OMG Unified Modeling Language, http://www.uml.org, USA, 1999.

[14] Pow-Sang, J., Imbert R., Estimación y Planificación de Proyectos Software con Ciclo de Vida Iterativo-Incremental y empleo de Casos de Uso, Proceedings IDEAS 2004, Arequipa-Perú, 2004.

[15] Pow-Sang, J., Una Experiencia con Estudiantes para la Estimación del Esfuerzo de cada Iteración en Proyectos de Software, Proceedings XXXI Conferencia Latinoamericana de Informática-CLEI 2005, Pontificia Universidad Javeriana, Cali-Colombia, 2005.

[16] Pow-Sang, J., GESPROMET, Sistema para la Gestión de Proyectos de Software Utilizando MÉTRICA Versión 3. Tesis de Máster en Ingeniería del Software, Universidad Politécnica de Madrid, España, 2002.

[17] Pow-Sang, J., Estudio Comparativo de Técnicas para la Estimación del Esfuerzo de las Iteraciones de Proyectos Software, Proceedings JIISIC'04, Madrid-España, 2004.

[18] Rational Software, Rational Unified Process version 2001A.04.00.13, USA, 2001.

[19] Royce, W. W., Managing the Development of Large Software Systems: Concepts and Techniques. Proceedings WESCON, 1970.

[20] Rumbaugh, I., Jacobson, I., Booch, G., Unified Modeling Language Reference Manual, Addison Wesley, 1997.

[21] Rosenberg, D., Scott, K., Use Case Driven Object Modeling with UML, Addison-Wesley, Massachusets, USA, 1999.

[22] Sadoski, D., Two Tier Software Architectures, http://www.sei.cmu.edu/str/descriptions/twotier.html, SEI, USA, 2004.

[23]Schneider, G. and Winters, J. Applying Use Cases – A Practical Guide, 2nd Edition. Addison-Wesley. USA, 2001.

[24] The International Function Point User Group (IFPUG), Function Point Counting Practices Manual-Release 4.1, USA, 1999.