Definition and Empirical Validation of Metrics for Software Process Models

Félix García, Francisco Ruiz, and Mario Piattini

Alarcos Research Group, University of Castilla-La Mancha, Paseo de la Universidad, 4
13071 Ciudad Real, Spain
`{Felix.Garcia, Francisco.RuizG, Mario.Piattini}@uclm.es,`
`alarcos.inf-cr.uclm.es/english/`

Abstract. Software companies are becoming more and more concerned about software process improvement, when they are promoting the improvement of the final products. One of the main reason of the growing interest in software metrics has been the perception that software metrics are necessary for software process improvement. Measurement is essential for understanding, defining, managing and controlling the software development and maintenance processes and it is not possible to characterize the various aspects of development in a quantitative way without having a deep understanding of software development activities and their relationships. In this paper a representative set of metrics for software process models is presented in order to evaluate the influence of the software process models complexity in their quality. These metrics are focused on the main elements included in a model of software processes, and may provide the quantitative base necessary to evaluate the changes in the software processes in companies with high maturity levels. To demonstrate the practical utility of the metrics proposed at model level, an experiment has been achieved which has allowed us to obtain some conclusions about the influence of the metrics proposed on two sub-characteristics of the maintainability: understandability and modifiability, which besides confirm the results of a subjective experiment previously performed.

1 Introduction

There is a direct correlation between the quality of the process and the quality of the resulting software products and for this reason companies are becoming more and more concerned about software process improvement, when they are promoting the improvement of the final products. To support the software process evaluation and improvement, a great variety of initiatives have arisen establishing reference frameworks. Among these initiatives, of special note are CMM [17], CMMI [18] and the ISO 15504 [11]. Among the above-mentioned improvement initiatives, CMMI (Capability Maturity Model Integration) stands out as being especially important. Within the context of CMMI, the company should continuously understand, control and improve its processes, in order to reach the aims of each level of maturity. As a result of effective and efficient processes, a company will, in return, receive high quality products that satisfy both the needs of the client and of the company itself.

One of the main reason of the growing interest in software metrics has been the perception that software metrics are necessary for software process improvement [7]. Measurement is essential for understanding, defining, managing and controlling the software development and maintenance processes and it is not possible to characterize the various aspects of development in a quantitative way without having a deep understanding of software development activities and their relationships [15].

Therefore, in order to provide the a quantitative basis for the improvement of software process it is necessary the measurement of the process, but before it is necessary to know the elements involved. For this reason the issue of process modelling has received growing attention in the software community during last years. The process model is seen as the starting point to analyse, improve and enact the process, but the need of strict coupling between process modelling and process measurement has not yet clearly emerge [13]. We have treated this issue in previous works [8;9].

In this paper a representative set of metrics for software process models is presented in order to evaluate the influence of the complexity in the software process models in their maintainability. These metrics are focused on the main elements included in a model of software processes, and may provide the quantitative base necessary to evaluate the changes in the software processes in companies with high maturity levels.

To demonstrate the practical utility of the metrics proposed at model level an experiment has been carried out which has allowed us to obtain some conclusions about the influence of the metrics proposed on two sub-characteristics of the maintainability: understandability and modifiability.

Firstly, we present a set of representative metrics for the evaluation of software process models and an example of calculation. In Section 3 the empirical validation of the metrics proposed at model level is presented. Finally, some conclusions and further works are outlined.

2 Proposal of Metrics for Software Process Models

Research on the software process evaluation has been focused in the collection of project data to obtain throughput, efficiency and productivity metrics and for this reason explicit metrics on software process models have not been defined.

The study of the possible influence of the complexity of software process models in their execution (enactment) could be very useful. For this reason the first step is the definition of a collection of metrics in order to characterise the software process model. The main objective to be achieved is the development of a empirical study to demonstrate the influence of the metrics proposed (which are applied on the attributes of software process models) on the maintainability of software process models. These metrics are indicators of a software process model structural complexity, and they could be every useful, taking into account that a software process model with high degree of complexity will be much more difficult to change, and this affects to their maintainability. This affirmation is based on the theoretical basis for developing quantitative models relating to structural properties and external quality attributes provided by [4] which is illustrated in Figure 1. This basis could be applied to the software process in the same way they are applied to software artifacts. Software process models hardly maintainable affect to the execution of projects (more expensive in resources and schedule) and to the final quality of the software products obtained.

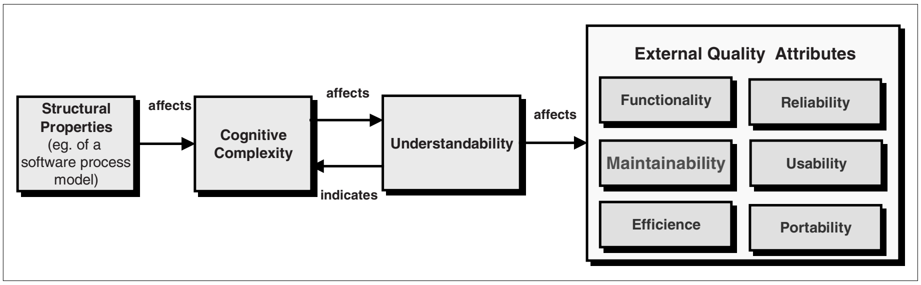

Fig. 1. Relationship between structural properties and external quality attributes

The metrics have been defined following the SPEM terminology [19], but they can be directly applied to other process modeling languages. The conceptual model of SPEM is based on the idea that a software development process consists of the collaboration between abstract and active entities, referred to as process roles, that carry out operations, called activities, on tangible entities, called work products. The basis of software processes consists of the interaction or collaboration of multiple roles through the exchange of work products and triggering the execution , or enactment of certain activities. The aim of a process is to bring a set of work products to a well-defined state.

SPEM has not a graphical notation by itself, but basic UML diagrams can be used to present different perspectives of a software process model. In particular, the following UML notations are useful: Class diagram, Package diagram, Activity diagram, Use case diagram and Sequence diagram. Figure 2 shows an example of a simplified software process model which belongs to the Rational Unified Process [12]. For the graphical representation of the model the Activity Diagram notation and the stereotypes which represent the SPEM constructors has been used.

As we can observe in Figure2, using UML Activity diagrams it is possible to represent a view of the software process in which the different activities, their precedence relationships, the work products consumed or produced and the responsible roles are included.

The metrics have been defined examining the key software process constructors of the SPEM metamodel and they could be classified as:

– **Model Level Metrics**. They are applied in order to measure the structural complexity of the overall software process model. These metrics are repre sented in Table 1 and an example of calculation of the metrics for the software process model represented in the Figure 2 is shown in Table 2.

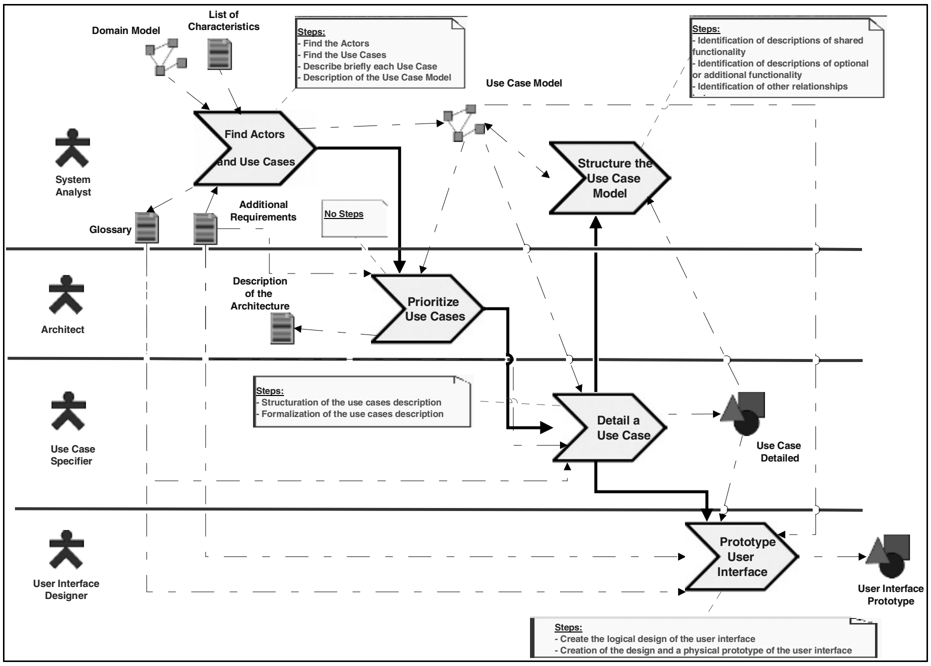
Domain Model
List of Characteristics
Steps:
- Find the Actors
- Find the Use Cases
- Describe briefly each Use Case
- Description of the Use Case Model
Use Case Model
Steps:
- Identification of descriptions of shared functionality
- Identification of descriptions of optional or additional functionality
- Identification of other relationships
System Analyst
Find Actors and Use Cases
Structure the Use Case Model
Glossary
Additional Requirements
No Steps
Architect
Description of the Architecture
Prioritize Use Cases
Steps:
- Structuration of the use cases description
- Formalization of the use cases description
Use Case Specifier
Detail a Use Case
Use Case Detailed
User Interface Designer
Prototype User Interface
User Interface Prototype
Steps:
- Create the logical design of the user interface
- Creation of the design and a physical prototype of the user interface

Fig. 2. Example of a Software Process Model represented with SPEM

- **Fundamental Element Metrics**. They evaluate the structural complexity of the main elements of the software process model: Activity, Work Product and Process Role. The definition of these metrics is presented in [9].

3 Empirical Validation of the Model Level Metrics

In order to prove the practical utility of the metrics it is necessary to run out empirical studies. In this section we describe an experiment we have carried out to empirically validate the proposed measures as early maintainability indicators. We have followed some suggestions provided in [20][14] [5] and [6] on how to perform controlled experiments and have used (with only minor changes) the format proposed in [20] to describe it.

We performed a previous controlled experiment [10], pursuing a similar objective. In it, as in this one, the independent variable is the software process model structural complexity. In the previous experiment the dependent variables were three maintainability sub-characteristics (understandability, analysability and modifiability) measured by means of user ratings on a scale composed of seven linguistic labels (from extremely easy to extremely difficult for each sub-characteristic). Even though the results obtained in the previous experiment reflect that several of the model level

Table 1. Model Level Metrics

Metric	Definition
NA(PM)	Number of **Activities** of the software process model
NWP(PM)	Number of **Work Products** of the software process model
NPR(PM)	Number of **Roles** which participate in the process
NDWPIn(PM)	Number of input dependences of the **Work Products** with the **Activities** in the process
NDWPOut(PM)	Number of output dependences of the **Work Products** with the **Activities** in the process
NDWP(PM)	Number of dependences between **Work Products** and **Activities** $NDWP(PM) = NDWPIn(MP) + NDWPOut(MP)$
NDA(PM)	Number of precedence dependences between **Activities**
NCA(PM)	Activity Coupling in the process model. $NCA(PM) = \frac{NA(PM)}{NDA(PM)}$
RDWPIn(PM)	Ratio between **input dependences** of Work Products with Activities and **total number of dependences** of Work Products with Activities $RDWPIn(PM) = \frac{NDWPIn(PM)}{NDWP(PM)}$
RDWPOut(PM)	Ratio between **output dependences** of Work Products with Activities and **total number of dependences** of Work Products with Activities $RDWPOut(PM) = \frac{NDWPOut(PM)}{NDWP(PM)}$
RWPA(PM)	Ratio of **Work Products** and **Activities**. Average of the work products and the activities of the process model. $RWPA(PM) = \frac{NWP(PM)}{NA(PM)}$
RRPA(PM)	Ratio of **Process Roles** and **Activities** $RRPA(MP) = \frac{NRP(MP)}{NA(MP)}$

Table 2. Values of Model Level Metrics

Metric	Value	Metric	Value
NA(PM)	5	NDA(PM)	4
NWP(PM)	8	NCA(PM)	5/4= 1,25
NPR(PM)	4	RDWPIn(PM)	13/18=0,722
NDWPIn(PM)	13	RDWPOut(PM)	5/18=0,278
NDWPOut(PM)	5	RWPA(PM)	8/5=1,6
NDWP(PM)	18	RRPA(PM)	4/5= 0,8

metrics (NA, NWP, NDWPIn, NDWPOut, NDWP y NDA) were highly related to-software process models maintainability, we are aware that the way we choose to measure the dependent variable was subjective and relies solely on judgment of the users, which may have biased the results. Therefore, we decided to carry out another experiment measuring the dependent variable in a more objective way. This experiment is presented in the following subsections.

3.1 Definition

Using the GQM template [1], for goal definition, the experiment goal is defined as follows:

Analyse	*Software process models (SPM) structural complexity metrics*
For the purpose of	*Evaluating*
With respect to	*their capability of being used as software process model maintainability indicators*
From the point of view of	*Software Process Analysts*
In the context of	*Software engineers of a company for the development and maintenance of information systems.*

3.2 Planning

After the definition of the experiments -*why* the experiment is conducted-, the planning took place. The planning prepares for *how* the experiment is conducted, including the following activities:

- **Context selection.** The context of the experiment is a group of professionals of a software company, and hence the experiment is run on-line (in an industrial software development environment). The subjects have been thirty-one software engineers of the Cronos Iberica Consulting, company dedicated to the development and maintenance of software for information systems. The experiment is specific since it focuses on SPM structural complexity metrics. The ability to generalise from this specific context is further elaborated below when we discuss threats to the external validity of the experiment. The experiment addresses a real problem, i.e., which indicators can be used to assess the maintainability of SPM? To this end it investigates the correlation between metrics and maintainability.
- **Selection of subjects.** The subjects have been chosen for convenience, i.e., the subjects are professionals of a software company who have wide experience and knowledge in software product modelling (UML, databases, etc.), but they have not experience or knowledge in the conceptual modelling of SPM.
- **Variables selection.** The independent variable is the SPM structural complexity. The dependent variable is SPM maintainability.
- **Instrumentation.** The objects have been 18 SPM belonging to different standards and methodologies. The independent variable has been measured through the metrics proposed at process model level (see section 2). The dependent vari-

ables have been measured by the time the subjects spent answering the questions of the first section related with the understandability of each model (understandability time) and by the time subjects spent carrying out the tasks required in the second section of the experiment (modifiability time). Our assumption here is that, the faster a class diagram can be understood and modified, the easier it is to maintain.

- **Hypothesis formulation.** We wish to test the following two set of hypotheses:
  1) Null hypothesis, H_{0e}: There is no significant correlation between structural complexity metrics (NA, NWP, NPR, NDA, NDWP, NDWPIn, NDWPOut, NCA, RDWPIn, RDWPOut, RWPA, RRPA) and the understandability time.
  2) Alternative hypothesis, H_{1e}: There is significant correlation between structural complexity metrics and the understandability time.
  3) Null hypothesis, H_{0m}: There is no significant correlation between structural complexity metrics and the modifiability time.
  4) Alternative hypothesis, H_{1m}: There is significant correlation between structural complexity metrics and the modifiability time.
- **Experiment design.** We selected a within-subject design experiment, i.e., all the tests (experimental tasks) have had to be solved by each of the subjects. The subjects were given the tests in different order.

Table 3. Metric values for each class diagram.

Model	NA	NWP	NPR	NDWPIn	NDWPOut	NDWP	NDA	NCA	RDWPIn	RDWPOut	RWPA	RRPA
1	6	6	3	5	6	11	6	1,000	0,455	0,545	1,000	0,500
2	5	6	4	5	5	10	4	1,250	0,500	0,500	1,200	0,800
3	2	13	2	12	3	15	1	2,000	0,800	0,200	6,500	1,000
4	9	25	9	25	21	46	11	0,818	0,543	0,457	2,778	1,000
5	5	6	4	5	5	10	8	0,625	0,500	0,500	1,200	0,800
6	4	11	4	14	9	23	3	1,333	0,609	0,391	2,750	1,000
7	8	17	1	15	11	26	9	0,889	0,577	0,423	2,125	0,125
8	5	8	4	13	5	18	4	1,250	0,722	0,278	1,600	0,800
9	7	12	1	12	11	23	6	1,167	0,522	0,478	1,714	0,143
10	24	37	10	72	40	112	24	1,000	0,643	0,357	1,542	0,417
11	7	12	5	12	11	23	6	1,167	0,522	0,478	1,714	0,714
12	2	8	3	6	4	10	1	2,000	0,600	0,400	4,000	1,500
13	3	6	1	8	3	11	4	0,750	0,727	0,273	2,000	0,333
14	3	5	7	5	3	8	2	1,500	0,625	0,375	1,667	2,333
15	4	9	1	9	7	16	6	0,667	0,563	0,438	2,250	0,250
16	8	6	4	9	9	18	7	1,143	0,500	0,500	0,750	0,500
17	4	24	1	20	11	31	3	1,333	0,645	0,355	6,000	0,250
18	5	21	3	21	11	32	4	1,250	0,656	0,344	4,200	0,600

3.3 Operation

It is in this phase where measurements are collected, including the following activities:

– **Preparation**. Subjects were given an intensive training session before the experiment took place. However, the subjects were not aware of what aspects we intended to study. Neither were they aware of the actual hypothesis stated. We prepared the material we handed to the subjects, consisting of eighteen SPM and one example solved. These models were related with different universes of discourse but they were general enough to be understood by the subjects. The structural complexity of each diagram is different, because as table 3 shows, the values of the metrics are different for each model.

Each diagram had an enclosed test (see appendix A) that included two sections: the first composed by five questions related with the model and the second composed by different steps to perform for the modification of the model. Each subject had to answer the questions of the section 1 and perform the modifications specified in the section 2. For each section the subject had to specify the starting and ending times. The difference between the two times in the first section is that we call understandability time (expressed in minutes and seconds) and the time difference in the second section is called the modifiability time. The modifications to each SPM were similar, including adding and deleting of activities, work products, roles and their dependences.

- **Execution.** The subjects were given all the materials described in the previous paragraph. We explained how to do the tests. We allowed one week to carry out the experiment, i.e., each subject had to do the test alone, and could use unlimited time to solve it. We collected all the data including the times of understanding and modification, the answers of the first section and the original models modified as a result of the second section.
- **Data Validation.** Once the data was collected, we have controlled if the tests were complete and if the modifications had been done correctly. We have discarded the tests of two subjects because they were incomplete. Therefore, we have taken into account the responses of 29 subjects.

Table 4. Spearman´s correlation coefficients between metrics and understandability and modifiability time

Metric	Spearman´correlation coefficients understandability time	Spearman´correlation coefficients modifiability time
NA(PM)	**0,604** p=0,008	0,171 p=0,496
NWP(PM)	**0,694** P=0,001	0,364 p=0,138
NPR(PM)	0,211 p= 0,402	0,348 p=0,157
NDWPIn(PM)	**0,740** p=0,000	0,383 p=0,117
NDWPOut(PM)	**0,747** p=0,000	0,212 p=0,398
NDWP(PM)	**0,772** p=0,000	0,338 p=0,170
NDA(PM)	**0,529** p=0,024	0,060 p=0,814
NCA(PM)	-0,275 p=0,269	0,151 p=0,549
RDWPIn(PM)	0,142 p=0,573	0,324 p=0,190
RDWPOut(PM)	-0,142 p=0,573	-0,324 p=0,190
RWPA(PM)	0,150 p=0,554	0,117 p=0,644
RRPA(PM)	-0,304 p=0,220	0,101 p=0,691

3.4 Analysis and Interpretation

We had the metric values calculated for each SPM (see table 3), and we have calculated the mean of the understandability and modifiability time. So this is the data we want to analyse to test the hypotheses stated above. We have applied the Kolmogorov-Smirnov test to ascertain if the distribution of the data collected was normal. As the data have been non-normal we have decided to use a non-parametric test like Spearman's correlation coefficient, with a level of significance $\alpha = 0.05$, correlating each of the metrics separately with understandability time and with modifiability time (see table 4).

For a sample size of 18 (mean values for each diagram) and $\alpha = 0.05$, the Spearman cutoff for accepting H_{0e} and H_{0m} is 0,4684 [21]. Because the computed Spearman's correlation coefficients for the understanding time (see table 4) for the metrics NA, NWP, NDWPIn, NDWPOut, NDWP and NDA are above the cutoff, and the p-value < 0,05, the null hypothesis H_{0e}, is rejected. Hence, we can conclude that there is a significant correlation between these metrics and the understandability time. Respect to modifiability time all the correlation values are below the cutoff and for this reason there is not correlation with the metrics defined. We think these results are produced due to that the requested modifications in the different diagrams were similar and the subjects had previously answered the questions related with the understandability. So in future experiments we have to focus specially on the influence of the metrics on the modifiability time.

3.5 Validity Evaluation

We will discuss the various issues that threaten the validity of the empirical study and how we attempted to alleviate them:

- **Threats to conclusion validity.** The only issue that could affect the statistical validity of this study is the size of the sample data (522 values, 18 models and 29 subjects), that perhaps are not enough for both parametric and non-parametric statistic test [3]. We are aware of this, so we will consider the results of the experiment as preliminary findings.
- **Threats to Construct Validity.** The dependent variables we used are understandability and modifiability time, so we consider these variables constructively valid.
- **Threats to Internal Validity.** Seeing the results of the experiment we can conclude that empirical evidence of the existing relationship between the independent and the dependent variables exists. We have tackled different aspects that could threaten the internal validity of the study, such as: differences among subjects, knowledge of the universe of discourse among SPM, precision in the time values, learning effects, fatigue effects, persistence effects and subject motivation. The only issue which could affect internal validity was the fatigue effects because the average duration of the experiment was two hours and twenty-four

minutes. But in this experiment we think this does not affect because subjects are professionals. In future empirical studies we have to consider to plan experiments with more reduced duration if the subjects are students.

- **Threats to External Validity.** Three threats to external validity have been identified which could limit the realism of the experiment [16] and the ability to generalize the research results to the population under study:
  - Materials and tasks used. In the experiment, we have used software process models based on standards and methodologies found in the bibliography and tasks representative of real cases, but more empirical studies, using real software process models from software companies, must be carried out.
  - Subjects. The experiment has been performed by professional subjects which eases the generalization of the results.
  - Environment. The experiment was performed in the company but the tasks had to be done by using pen and paper. In future experiments we could consider the use of software tools to perform the activities required in order to provide a more realistic environment.

4 Conclusions and Future Work

In this work a set of representative metrics have been proposed in order to evaluate the influence of the complexity in the software process models in their quality. These metrics are focused on the main elements included in a model of software processes, and may provide the quantitative base necessary to evaluate the changes in the software processes in companies with high maturity levels.

In order to evaluate the relationship between the structural complexity of the software process models, measured through the metrics proposed, and their maintainability we have carried out an experiment to select the metrics related with the time necessary to understand and modify a software process model. This experiment has allowed us to obtain some conclusions about the influence of the metrics proposed at model level in the maintainability of the software process models through two of its sub-characteristics (understandability and modifiability). As a result of these experiments performed we could conclude that the metrics NA, NWP, NDWPIn, NDWPOut, NDWP and NDA are good understandability indicators, however we cannot say the same about the metrics NPR, NCA, RDWPIn, RDWPOut y RWPA. These results confirm part of the results of one previous subjective experiment [10]. However with this experiment we have not demonstrated the relationship between the metrics and the modifiability. We have to conduct new experiments related with the modification of the models to confirm or discard these results.

Although the results obtained in these experiments are good respect to the understanding, we can not consider them like definitive results. It is necessary elaborate new experiments centered in the evaluation of concrete metrics we consider relevant and that in this experiment seem not to be correlated like the metric NCA and NPR. According to the issues previously identified, the lines for improvement for future studies, we can point out the following:

– Development of replicas [2] to confirm the results obtained. In this replicas we could also consider the perception of the participants about the understandability and modifiability of the process models in order to detect possible relationships among this perception (measured in a subjective way), the metrics, and the times employed to understand and modify he models.
– Carrying out a new experiment centered in the evaluation of concrete metrics we consider relevant (NRP, NCA) and that according the previous experiments performed not to seem be correlated with the maintainability of software process models.
– Carrying out case studies using real software process models of companies.
– Consideration of other views related with the modelling of software processes, like for example roles and their responsibilities on work products, in order to define and validate new possible metrics.

Acknowledgements. This work has been partially funded by the TAMANSI project financed by "Consejería de Ciencia y Tecnología, Junta de Comunidades de Castilla-La Mancha" of Spain (project reference PBC-02-001) and by the MAS project partially supported by "Dirección General de Investigación of the Ministerio de Ciencia y Tecnología" (TIC 2003-02737-C02-02).

References

1. Basili, V. and Rombach H. The TAME project: towards improvement-oriented software environments. IEEE Transactions on Software Engineering, 14(6), (1988), 728-738.
2. Basili, V., Shull, F. and Lanubile, F. Building Knowledge through Families of Experiments. IEEE Transactions on Software Engineering, 25(4), (1999), 435-437.
3. Briand, L., El Emam, K. and Morasca, S. Theoretical and empirical validation of software product measures. Technical Report ISERN-95-03, International Software Engineering Research Network (1995).
4. Briand., L., Wüst, J. and Lounis, H. A. Comprehensive Investigation of Quality Factors in Object-Oriented Designs: an Industrial Case Study. In Technical Report ISERN-98-29, International Software Engineering Research Network (1998).
5. Briand L., Arisholm S., Counsell F., Houdek F. and Thévenod-Fosse P. Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions. Empirical Software Engineering, 4(4), (1999), 387-404.
6. Calero, C., Piattini, M. and Genero, M.: Empirical Validation of referential metrics. Information Software and Technology". Special Issue on Controlled Experiments in Software Technology. Vol.43, Nº 15 (2001).
7. Fenton, N.: Metrics for Software Process Improvement. Software Process Improvement: Metrics, Measurement and Process Modelling. Haug, M., Olsen, E. W. and Bergman, L (eds). Springer (2001), 34-55.
8. García, F., Ruiz, F. and Piattini, M. Metamodeling and Measurement for the Software Process Improvement. Proceedings of ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'03). Tunis (Tunisia). 14-18 July (2003).

9. García, F., Ruiz, F., Cruz, J.A., Piattini, M. Integrated Measurement for the Evaluation and Improvement of Software Processes. 9th European Workshop on Software Process Technology (EWSPT'9). Lecture Notes in Computer Science. Helsinki (Finland), 1-2 September (2003).
10. García, F., Ruiz, F. and Piattini, M. Proposal of Metrics for Software Process Models. Accepted for publication in Software Measurement European Forum 2004, Rome, 28-30 January, 2004.
11. ISO/IEC: ISO IEC 15504 TR2:1998, part 2: A reference model for processes and process capability, (1998).
12. Jacobson, I, Booch, G. and Rumbaugh, J. The Unified Software Development Process. Addison Wesley (1999).
13. Morisio, M.: Measurement Processes are Software Too. Journal of Systems and Software, vol 49(1), December (1999).
14. Perry, D., Porte, A. and Votta, L. Empirical Studies os Software En-gineering: A Roadmap. Future of Software Engineering. Ed:Anthony Finkelstein, ACM, (2000), 345-355.
15. Pfleeger, S.L.: Integrating Process and Measurement. In Software Measurement. A. Melton (ed). London. International Thomson Computer Press (1996) 53-74
16. Sjoberg, D., Anda, B., Arisholm, E., Dyba, T., Jorgensen, M., Karahasanovic, A., Koren, E. and Vokác, M. Conducting Realistic Experiments in Software Engineering. Proceedings of the 2002 International Symposium on Empirical Software Engineering (ISESE'02).
17. Software Engineering Institute (SEI). The Capability Maturity Model: Guidelines for Improving the Software Process, (1995). In http://www.sei.cmu.edu/cmm/cmm.html
18. Software Engineering Institute (SEI). Capability Maturity Model Integration (CMMI^{SM}), version 1.1. March (2002). In http://www.sei.cmu/cmmi/cmmi.html
19. Software Process Engineering Metamodel Specification; adopted specification, version 1.0. Object Management Group. November (2002). Available in http://cgi.omg.org/cgi-bin/doc?ptc/02-05-03.
20. Wohlin C., Runeson P., Höst M., Ohlson M., Regnell B. and Wesslén A. Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers. (2000).
21. http://department.obg.cuhk.edu.hk/ResearchSupport/Minimum_correlation.asp

Appendix A

SPM 7. With the SPM shown (Figure 2), you have to perform the following tasks:

Tasks: Part I. Answer the following questions:
Write down the starting hour (indicating hh:mm:ss): ___________
1.- Can the <u>Use Case Specifier</u> participate in **Structure the Use Case Model**? __
2.- Does **Structure the Use Case Model** precede to **Prototype User Interface**?__
3.- Is it necessary to use the product *Use Case Detailed* like input of the activity **Structure the Use Case Model**? __
4.- Is the work product *Use Case Model* output of **Prioritize Use Cases**?
5.- When **Prototype the User Interface** is executed, has the *Use Case Detailed* product already produced? __

Write down the ending hour (indicating hh:mm:ss): ____________

Tasks: Part II. Carry out the necessary modifications to satisfy the following requirements:

Write down the starting hour (indicating hh:mm:ss): ___________

1.- The activity **Detail a Use Case** uses the *Description of the Architecture* like input.

2.- The activity **Detail a Use Case** is considered not to precede to **Structure the Use Case Model** and this last one will be precede by **Prototype the User Interface**.

3.- After the activity **Find Actors and Use Cases** is desired to perform a **Checking of the consistence Requisites-Domain,** which receives like inputs the *Domain Model*, the *Use Case Model* and an *Historical of the Domain*. This new activity precedes to **Prioritize Use Cases**.

4.- The Verification Group is the responsible for **Checking of the consistence Requisites-Domain.**

Write down the ending hour (indicating hh:mm:ss): ___________