Torgeir Dingsøyr (Ed.)

LNCS 3281

Software Process Improvement

11th European Conference, EuroSPI 2004
Trondheim, Norway, November 2004
Proceedings

Springer

Volume Editor

Torgeir Dingsøyr
SINTEF ICT
S P Andersens vei 15, 7465 Trondheim, Norway
E-mail: Torgeir.Dingsoyr@sintef.no

Library of Congress Control Number: 2004113832

CR Subject Classification (1998): D.2, K.6, K.4.2

ISSN 0302-9743
ISBN 3-540-23725-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11341086 06/3142 5 4 3 2 1 0

Preface

This was the first year that the European Software Process Improvement Conference – EuroSPI – had a separate research track with its own proceedings. The EuroSPI conference is in its eleventh year, and has become the main meeting place in Europe for the software industry and academia to discuss software process improvement. The conference deals with software process improvement in a broad sense, investigating organizational issues as well as methods and tools for software process improvement.

EuroSPI is an initiative financed by a consortium of Nordic research centers and user networks (SINTEF, DELTA and STTF), ASQF, a German quality assurance association, and ISCN in Ireland, the coordinating network partner.

The research papers describe innovative and significant work in software process improvement, which is relevant to the software industry. The papers are readable for a scientific and industrial audience, and support claims with appropriately described evidence or references to relevant literature.

Thirty-one papers were submitted in this year's research track, and each paper was sent to three or four members of the program committee or additional reviewers. Papers were evaluated according to originality, significance of the contribution, quality of the written and graphical presentation, research method applied, and appropriateness of comparison to relevant research and literature. Almost 100 reviews were received and 18 papers were selected for presentation in the research track, giving a rejection rate of 42%. Many high-quality submissions had to be rejected because of limited space in the conference program.

The selected papers cover a wide area in software process improvement, from improving agile development methods, techniques for software process improvement, and knowledge management in software companies to effort estimation and global software development.

I would like to thank the paper authors for providing papers of high quality, and the program committee and additional reviewers for critiques, praise and advice on how to make the papers even better.

For further information about future EuroSPI conferences, see www.eurospi.net.

August 2004 Torgeir Dingsøyr

EuroSPI 2004 Conference Organization

General Chair

Dr. Richard Messnarz, ISCN, Ireland

Local Chair

Nils Brede Moe, SINTEF ICT, Norway

Scientific Programme Committee Chair

Dr. Torgeir Dingsøyr, SINTEF ICT, Norway

Industrial Programme Committee Chairs

Jørn Johansen, DELTA, Denmark
Mads Christiansen, DELTA, Denmark
Risto Nevalainen, STTF, Finland

Industry Chair

Dr. Bernd Hindel, ASQF, Germany

Exhibition Chair

Robert Treffny, ASQF, Germany

Tutorial Chair

Dr. Richard Messnarz, ISCN, Ireland

Research Track Program Committee

Pekka Abrahamsson, VTT Electronics, Finland
Vincenzo Ambriola, University of Pisa, Italy
Aybuke Aurum, University of New South Wales, Australia
Stefan Biffl, Vienna University of Technology, Austria
Miklos Biro, University of Budapest, Hungary
Christian Bunse, Fraunhofer IESE, Germany
Marcus Ciolkowski, University of Kaiserslautern, Germany
Karl Cox, NICTA, Australia
Kevin C. Desouza, University of Illinois, USA

Taz Daughtrey, James Madison University, USA
Howard Duncan, Dublin City University, Ireland
Tore Dybå, SINTEF, Norway
Jan Pries-Heje, IT University of Copenhagen, Denmark
Natalia Juristo, Polytechnical University of Madrid, Spain
Karl Heinz Kautz, Copenhagen Business School, Denmark
Jyrki Kontio, Helsinki University of Technology, Finland
Dieter Landes, Coburg University of Applied Sciences, Germany
Mikael Lindvall, Fraunhofer Center, USA
Patricia McQuaid, California Polytechnic State University, USA
Jürgen Münch, Fraunhofer IESE, Germany
Matthias Müller, University of Karlsruhe, Germany
Markku Oivo, University of Oulu, Finland
Elixabete Ostolaza, European Software Institute, Spain
Ita Richardson, University of Limerick, Ireland
Gunther Ruhe, University of Calgary, Canada
Per Runeson, Lund Institute of Technology, Sweden
Kurt Schneider, University of Hannover, Germany
Martin Shepperd, Bournemouth University, UK
Magne Jørgensen, Simula Research Laboratory, Norway
Tor Stålhane, Norwegian University of Science and Technology, Norway
Timo Varkoi, Tampere University of Technology, Finland
Claes Wohlin, Blekinge Institute of Technology, Sweden

Additional Reviewers

Ulrike Becker-Kornstaedt, USA
M. Letizia Jaccheri, Norwegian University of Science and Technology, Norway
Daniel Karlström, Lund Institute of Technology, Sweden
Ana M. Moreno, Universidad Politecnica de Madrid, Spain
Knut H. Rolland, Norwegian University of Science and Technology, Norway
Maribel Sanchez-Segura, Universidad Carlos III de Madrid, Spain

Table of Contents

19. The KYKY model, Software Technology Transfer Finland, http://www.sttf.fi/ (Unpublished)
20. Laryd, A., Orci, T.: Dynamic CMM for Small Organizations. In Proceedings of the 1st Argentine Symposium on Software Engineering (ASSE 2000) (2000) 133-149
21. Moitra, D.: Managing Change for Software Process Improvement Initiatives: a Practical Experience-Based Approach. Software Process Improvement and Practice, Vol. 4, No. 4, (1998) 199-207
22. Rainer, A., Hall, T.: Key Success Factors for Implementing Software Process Improvement: A Maturity-Based Analysis. Journal of Systems and Software, Vol. 67, No. 2, (2002) 71-84
23. Saukkonen, S., Oivo, M.: Teollinen ohjelmistoprosessi: Ohjelmistoprosessin parantaminen SIPI-menetelmällä. Helsinki, TEKES, (1998) (in finnish)
24. SPICE assessment forms: Finnish Software Measurement Association, http://www.fisma-network.org/ (Unpublished)
25. tSoft-project: University of Joensuu, Finland, http://cs.joensuu.fi/tSoft/ (in finnish)
26. Stelzer, D., Mellis, W.: Success Factors of Organizational Change in Software Process Improvement. Software Process Improvement and Practice, Vol. 4, No. 4, (1998) 227-250

An Experimental Replica to Validate a Set 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
http://alarcos.inf-cr.uclm.es/english/

Abstract. The software process measurement plays an essential role in order to provide the quantitative basis necessary for software process improvement. Traditionally, this measurement has been focused in the project and product measurement, but nowadays software process models (SPM) are entities very relevant due to the increasing number of companies which model and manage their processes in order to reach high maturity levels. We have defined a set of metrics for software process models in order to evaluate the influence of the software process models complexity in their maintainability. These metrics are focused on the main elements included in a software process model. To demonstrate the practical utility of the metrics proposed a replica of 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 set of experiments previously performed in the context of a family of experiments.

1 Introduction

Software process improvement has obtained a great attention in companies in the last few years. Companies are becoming more and more concerned about the improvement of the quality of their processes in order to promote the final quality of the software products obtained. As a matter of fact, the continuous software process improvement is a fundamental objective for organizations that desire to reach higher levels of maturity according to standards and proposals like CMMI [14] and the ISO 15504 [10]. Within the context of these initiatives, to reach the aims of each level of maturity the processes have to be improved by understanding, controlling and applying improvement actions.

The successful management of the software process is necessary in order to satisfy the final quality, cost, and time of the marketing of the software products and to carry out this management, four key responsibilities need to be assumed [7]: Definition, Measurement, Control and Improvement of the process. Taking these responsibilities into account, it is very important to consider the measurement of software processes

T. Dingsøyr (Ed.): EuroSPI 2004, LNCS 3281, pp. 79–90, 2004.

to establish the quantitative basis necessary for the identification of the areas which are candidate for the improvement. One of the main reasons of the growing interest in software metrics has been the perception that software metrics are necessary for software process improvement [6]. Measurement is essential for understanding, defining, managing and controlling the software development and maintenance processes [12].

In the measurement of software processes the following basic kind of entities can be identified:

- **Software Process Model.** The models constitute the starting point in order to understand and carry out the software process through its enactment in concrete projects. Software process modeling has become a very acceptable solution for treating the inherent complexity of software processes, and a great variety of modeling languages and formalities can be found in the literature, known as "Process Modeling Languages" (PML). With a software process model (SPM) the different elements related to a software process are precisely represented, without ambiguity.
- **Software Projects.** They are concrete enactments of software process models and their measurement is fundamental in order to know their performance measured mainly through schedule and cost metrics.
- **Software Products.** As a result of carried out software projects different products could be obtained and these are also candidate for the measurement. The quality of the process has to be reflected in the products obtained and for this reason it is necessary to measure the software products.

In the literature, the research on the software process measurement has been focused in the measurement of the projects and products, but explicit metrics on software process models have not been defined. Due to the importance of software process models in the improvement of software processes it is important to consider the influence on the processes of the quality of their models.

With this aim in mind, we have defined a representative set of metrics for software process models in order to evaluate the influence of the complexity in the software process models in their maintainability (see Table 1). The metrics are focused on the main elements included in a SPM and may provide the quantitative basis necessary to choose the model with the most easiness of maintenance among semantically equivalent SPM in organizations which are changing their models to improve their processes. It can greatly ease the evolution of their SPM.

The metrics defined are indicators of a software process model structural complexity, and they could be every useful as maintainability indicators, taking into account that a software process model with high degree of complexity will be much more difficult to change, and this can affect to their maintainability [3].

The metrics have been defined following the SPEM (Software Process Engineering Metamodel) terminology [15] by examining its key software process constructors, but they can be directly applied to other process modeling languages. The metrics defined are **Model Scope Metrics** (see Table 1), because they measure the structural complexity of the overall software process model.

Table 1. Model Scope 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** $RRPA(PM) = \frac{NPR(PM)}{NA(PM)}$

To demonstrate the practical utility of the metrics proposed as maintainability indicators a replica of 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. In the following section we present the empirical validation we have performed in the context of a family of experiments. In Section 2.1 we describe the results obtained in the previous experiments performed and then (section 2. 2) we describe with detail a replica of the last experiment. In Section 2.3 the results of the third experiment and its replica are compared. Finally, some conclusions and further works are outlined.

2 Empirical Validation of the Model Scope 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 (replica of a previous experiment) we have performed to empirically validate the proposed measures as early maintainability indicators. This experiment is part of a family of experiments we are carrying

out, according to the guidelines provided in [2], in order to establish if there is relationship between the metrics proposed at model scope (which evaluates the structural complexity of SPM) and the SPM maintainability measured through the understandability, modifiability and analysability (dependent variables).

2.1 Previous Experiments

In the experiments previously performed in the context of the family of experiments we provided the subjects 18 SPM and they rated their maintainability. These experiments can be classified in the following two groups:

- **Subjective Experiments**. In these two experiments the subjects (students, researchers and assistant professors) rated each of the three maintainability subcharacteristics according to a scale composed of seven linguistic labels (from extremely easy to extremely difficult for each sub-characteristic). The results obtained in these experiments [8] reflect that several of the model level metrics (NA, NWP, NDWPIn, NDWPOut, NDWP y NDA) were highly related to software process models maintainability.
- **Objective Experiment**. Even though the results of the subjective experiments were good, we were aware that the way of measuring the dependent variables was subjective and relied solely on judgment of the users, which may have biased the results. Therefore, we decided to carry out another experiment [9] in which the subjects were professionals of a software company and we measured the dependent variable in a more objective way. In this experiment the dependent variables considered were the understandability and the modifiability of the SPM. To measure these variables the subjects had to answer five questions and perform four modifications on the models. We obtained the time the subjects spent answering the questions (understandability time) and the time subjects spent carrying out the tasks required (modifiability time) in order to demonstrate if there was relationship between the metrics and the understandability and modifiability time. As a result of this experiment we could validate some metrics (NA, NWP, NDWPIn, NDWPOut, NDWP and NDA) respect to the understandability time, but we could not demonstrate any relationship between the metrics and the modifiability time. We think these results were produced because the subjects – before performing the modifications – had previously answered the questions related with the understandability. This fact was taken into account in the planning of the 4th Experiment.

2.2 Description of the 4th Experiment

The experiment described in this section is the fourth experiment of the family and it is a replica of the last one. To carry out this experiment we have followed some suggestions provided in [16][11] [4] and [5] on how to perform controlled experiments. In the following subsections are described the results obtained according to the format proposed in [16].

2.2.1 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	*Researchers*
In the context of	*Students of third course of Software Engineering*

2.2.2 Planning

The planning was carried out according to the following steps:

- **Context selection.** The context of the experiment is a group of undergraduate students and hence the experiment is run off-line (not in an industrial software development environment). The subjects were two groups of students enrolled at the Department of Computer Science at the University of Castilla-La Mancha in Spain. The first group was composed of forty-six students enrolled in the final-year (third) of the Computer Science (BSc) in the speciality of Management and the second group were forty-one students enrolled in the final-year in the Systems speciality of the Computer Science (BSc). The experiment is specific since it is focused on SPM structural complexity metrics. The ability to generalize from this specific context is further elaborated below when discussing threats to the experiment. The experiment addresses a real problem, i.e., what indicators can be used for the maintainability of SPM? With this end in view it investigates the correlation between SPM structural complexity metrics and two maintainability subcharacteristics (understandability and modificability).
- **Selection of subjects.** The subjects have been chosen for convenience, i.e., the subjects are students who have 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 variables are the understandability and the modifiability.
- **Instrumentation.** The objects have been 10 SPM belonging to different standards and methodologies. These models are a representative subset of the 18 original models provided in the previous experiments of the family which were selected to avoid the fatigue effects because in this experiment the subjects were students. To select this representative subset we considered for each model its structural complexity (metrics values) and its understandability time obtained in the previous experiment. The independent variable has been measured through the metrics proposed at model scope (see Table 1). The dependent variables have been measured by the time the subjects spent in carrying out the exercises included in each model, which could be one of the following: answering five questions related with the understandability of the model (understandability time) or carrying out four modifications activities on the model (modifiability time) (see Appendix A). Our assump-

tion here is that, the faster a class diagram can be understood and modified, the easier it is to maintain.

- **Hypotheses formulation.** We wish to test the following two set of hypotheses:
  1) Null hypothesis, H_{0c}: There is no significant correlation between structural complexity metrics and the understandability time.
  2) Alternative hypothesis, H_{1c}: 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 and the ten models selected respect to the original material [9] were grouped in the following way: Group X: Models 1, 2, 3, 9 and 10; Group Y: Models 4, 6, 7, 12 and 17.
  For each model two exercises sheets were prepared: one in which it was required to answer the understandability questions (X_u, Y_u) and another one containing the modification exercises (X_m, Y_m). To distribute the material, and considering that we had to provide ten SPM for each subject (five with understandability exercises and five with modifiability exercises) we decided to provide to the subjects of the Management Group the material packages X_u, Y_m and to the subjects of the Systems Group the material packages Y_u and X_m.

2.2.3 Operation

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

- **Preparation.** The experiment took place in the same day but in different timetable for each group of subjects (management and systems). Subjects were given an intensive training session before the experiment took place. However, subjects were not aware of what aspects we intended to study. Neither were they aware of the actual hypotheses stated. We prepared the material handed to the subjects consisting of ten 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 2 shows, the values of the metrics are different for each model.
  Each model had an enclosed test (see appendix A) that included one of the following two sections: the first composed of five questions related with the model and the second composed of different steps to perform for the modification of the model. Depending on the model and the group of subjects (management or systems), each subject had to answer the questions or perform the modifications specified. The modifications to each SPM were similar, including adding and deleting of activities, work products, roles and their dependences.

Table 2. Metric values for each model

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
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
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
12	2	8	3	6	4	10	1	2.000	0.600	0.400	4.000	1.500
17	4	24	1	20	11	31	3	1.333	0.645	0.355	6.000	0.250

- **Execution.** The subjects were given all the materials described in the previous paragraph. We explained how to do the tests. We allowed one hour and a half to carry out the experiment (we previously performed a pilot experiment to determine the average duration). We collected all the data consisting of the times of understanding and modification, the understandability answers and the modifications performed on the models.
- **Data Validation.** Once the data was collected, we have controlled if the tests were complete and if the modifications had been done correctly. We discarded the tests of one subject in the management group because the times in three models were missing. Therefore, we have taken into account the responses of 45 subjects in the group of management and 41 subjects in the group of systems.

2.2.4 Analysis and Interpretation

We had the metric values calculated for each SPM (see table 2), and we have calculated the mean of the understandability and modifiability time. So these are 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 were 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 (table 3).

For a sample size of 10 (mean values for each diagram) and $\alpha = 0.05$, the Spearman cutoff for accepting H_{0c} and H_{0m} is 0,6320 [17]. Because the computed Spearman's correlation coefficients for the understandability time (see table 3) for the metrics NA, NWP, NDWPIn, NDWPOut, NDWP, NDA and NCA are above the cutoff, and the p-value < 0,05, the null hypothesis H_{0c}, is rejected. Hence, we can conclude that there is a significant correlation between these metrics and the understandability time. Respect to modifiability time there is relationship between the metrics NA,

NWP, NDWPIn, NDWPOut and NDWP and the mean time required to perform the modifications required on the models.

Table 3. Spearman's correlation coefficients between metrics and understandability and modifiability time

Metric	Spearman'correlation coefficients understandability time	Spearman'correlation coefficients modifiability time
NA(PM)	**0.841** p=0.002	**0.640** p=0.046
NWP(PM)	**0.826** P=0.003	**0.650** p=0.042
NPR(PM)	0.074 p= 0.838	0.377 p=0.283
NDWPIn(PM)	**0.786** p=0.007	**0.738** p=0.015
NDWPOut(PM)	**0.886** p=0.001	**0.791** p=0.006
NDWP(PM)	**0.893** p=0.001	**0.707** p=0.022
NDA(PM)	**0.821** p=0.003	0.599 p=0.067
NCA(PM)	**-0.752** p=0.012	-0.44 p=0.203
RDWPIn(PM)	0.79 p=0.828	0.115 p=0.751
RDWPOut(PM)	-0.79 p=0.828	-0.115 p=0.751
RWPA(PM)	-0.116 p=0.751	-0.30 p=0.934
RRPA(PM)	-0.560 p=0.092	-0.141 p=0.697

2.2.5 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 size of the sample data (860 values, 10 models and 86 subjects) constitutes a significant size that allow us to obtain a conclusion validity.
- **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 exists empirical evidence of relationship between the independent and the dependent variables. 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. With respect to the previous experiment with professionals, in this experiment we specially focused in the subjects motivation by giving them a special lecture about software process modeling and technology and we reduced the average duration of the experiment to avoid fatigue effects.
- **Threats to External Validity.** The following threats to external validity that have been identified which could limit the realism of the experiment [13]:
  - 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 students and for this reason we could not generalize the results in the same way that in the previous experiment with professionals. However, the results obtained in this experiment has confirmed the results obtained in the experiment with professionals.
  - Environment. The experiment was 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.

2.3 Comparison of Results

In table 4 we summarize the results we have obtained from the third experiment and its replica in order to select the metrics that influences in the understandability (Und) and modifiability (Mod) of SPM:

Table 4. Metrics validated in the 3rd and 4th experiments

		NA	NWP	NDWPIn	NDWPOut	NDWP	NDA	NCA
3rd Exp	Und	✓	✓	✓	✓	✓	✓	
	Mod							
4th Exp (replica)	Und	✓	✓	✓	✓	✓	✓	✓
	Mod	✓	✓	✓	✓	✓		

As we can observe in table 4, the results of the replica confirm the results obtained in the previous experiment because the metrics NA, NWP, NDWPIn, NDWPOut, NDWP and NDA are related to the understandability. Besides, with the carrying out of the replica it has been possible to demonstrate the relationship between some metrics (NA, NWP, NDWPIn, NDWPOut and NDWP) and the modifiability. These results confirm our conclusion at the end of the third experiment that the understandability tasks had influence in the time used by the professionals to perform the modifications required [9]. Also, as a result of the replica seems that there could be a relationship between the metric NCA and the understandability. It is necessary to confirm this in future studies.

3 Conclusions and Future Work

Software process measurement plays an essential role in order to provide the quantitative basis necessary for software process improvement. Traditionally, this measurement has been focused in project and product measurement, but nowadays the software process models (SPM) are entities very relevant due to the increasing number of companies which model and manage their processes in order to reach higher maturity levels.

In this work we have proposed a set of representative metrics to provide a quantitative basis to evaluate the influence of the SPM complexity in the quality of the process. These metrics are based on the main elements included in a SPM and may be

very useful to evaluate software processes evolution 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 a replica of an experiment to select the metrics related with the time necessary to understand and modify a software process model. This replica has allowed us to confirm some conclusions about the influence of the metrics NA, NWP, NDWPIn, NDWPOut, NDWP, NDA and NCA as good understandability indicators and the metrics NA, NWP, NDWPIn, NDWPOut and NDWP as good modifiability indicators. These results confirm the results obtained in the experiments previously performed in the context of a family of experiments.

Although the results obtained are encouraging, we cannot consider them definitive results. It is necessary elaborate new experiments centered in the evaluation of concrete metrics we consider relevant like the metrics NCA and NPR. According to the issues previously identified, we can point out the following lines for improvement in future studies:

- Carrying out new experiments focused on the evaluation of concrete metrics we consider relevant (NPR, NCA) and that according the experiments of the family performed until now seem not to be clearly correlated with the maintainability of software process models.
- Carrying out case studies using real software process models.
- Consideration of other views related with the modeling 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 funded by the MAS project partially supported by "Dirección General de Investigación of the Ministerio de Ciencia y Tecnología" (TIC 2003-02737-C02-02).

Appendix A

Management Group:

SPM 1. (see Figure 1). Answer the following questions:

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

1. Can the Technical Designer **Define the User Interface**? ____
2. Is it possible to initiate the activity **Refine the User Interface** before the activity **Define the User Interface**? ____
3. Is it necessary to use the product *User Work Processes* for the activity **Refine the User Interface**? ____
4. Is the product *User Interface (refined)* output of the activity **Design Process Model**? ____

5. When the activity **Refine User Interface** is carried out, have the *Technical Requirements* been produced? ____

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

Systems Group:

SPM 1. (see Figure 1). Carry out the necessary modifications to satisfy the following requirements:

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

1. It is necessary to use the product *Requirements Preliminary Information* for the execution of the activity **Define Requirements.**
2. It is not necessary to finish the activity **Define Requirements** to start the **Definition of the User Interface**, but it is necessary the **Definition of the User Interface** to be executed after the activity **Design the Process Model**.
3. It is desired to include the new activity **Final Review**, after the **Building of the Application.** This new activity receives as input the *Application* and produces an *Approval Document*.
4. The Programmer is responsible of the **Final Review** and also participates in the Building of the Application.

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

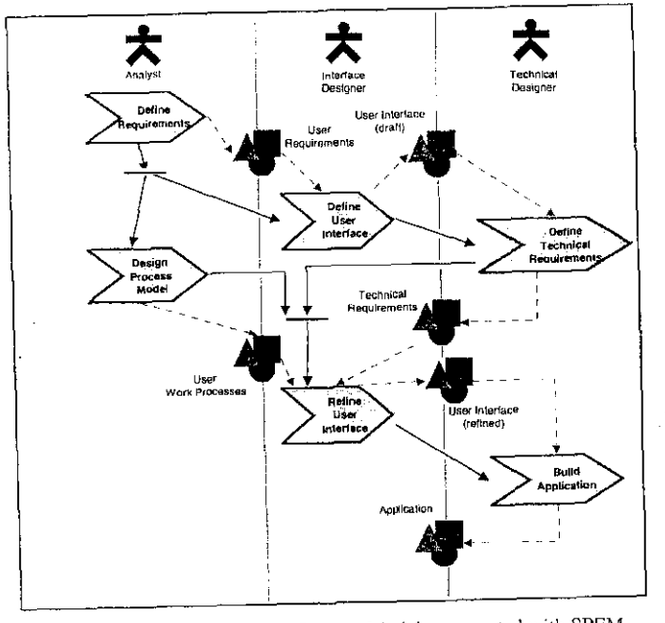

Fig. 1. Example of a Software Process Model represented with SPEM

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., 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).
4. 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.
5. 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).
6. 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.
7. Florac, W. A. and Carleton, A.D. Measuring the Software Process. Statistical Process Control for Software Process Improvement. SEI Series in Software Engineering. Addison Wesley (1999).
8. García, F., Ruiz, F. and Piattini, M. Proposal of Metrics for Software Process Models. *Software Measurement European Forum (SMEF 2004)*, Rome, 28-30 January (2004), 406-416.
9. García, F., Ruiz, F. and Piattini, M. Definition and Empirical Validation of Metrics for Software Process Models. *Proceedings of the 5th International Conference Product Focused Software Process Improvement (PROFES'2004)*. Lecture Notes in Computer Science (LNCS 3009). Kansai Science City (Japan), April (2004), 146-158.
10. ISO/IEC: ISO IEC 15504 TR2:1998, part 2: A reference model for processes and process capability, (1998).
11. 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.
12. Pfleeger, S.L.: Integrating Process and Measurement. In Software Measurement. A. Melton (ed). London. International Thomson Computer Press (1996) 53-74.
13. 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).
14. Software Engineering Institute (SEI). Capability Maturity Model Integration (CMMI^{SM}), version 1.1. March (2002). In http://www.sei.cmu/cmmi/cmmi.html
15. 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.
16. Wohlin C., Runeson P., Höst M., Ohlson M., Regnell B. and Wesslén A. Ex-perimentation in Software Engineering: An Introduction, Kluwer Academic Publishers. (2000).
17. http://department.obg.cuhk.edu.hk/researchsupport/Minimum_correlation.asp

Using Measurement Data in a TSP^{SM} Project

Noopur Davis[1], Julia Mullaney[1], and David Carrington[2]

[1] Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
[2] School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia, QLD 4072, Australia
davec@itee.uq.edu.au

Abstract. One of the challenges for software engineering is collecting meaningful data from industrial projects. Software process improvement depends on measurement to provide baseline status and confirming evidence of the effect of process changes. Without data, any conclusions rely on intuition and guessing. The Team Software Process^{SM} (TSP^{SM}) provides a powerful framework for data collection and analysis, in addition to its primary goal as a basis for highly effective software development. In this paper, we describe the experiences of, and benefits realized by, a team using the TSP for the first time. By reviewing how this particular team collected and used data, we show features of the TSP that make it a powerful foundation for software process improvement.

1 Introduction

The Capability Maturity Model® for Software (SW-CMM®) [7] is one of the earliest and most influential models for software process improvement. Because it focuses on the organizational level and describes "what" rather than "how", the model can prove challenging to apply, particularly to small organizations or projects. Humphrey [3] developed the Personal Software Process^{SM} (PSP^{SM}) using all the applicable SW-CMM practices up through level 5 to demonstrate how the principles are just as applicable to individual software engineers.

Humphrey then developed the Team Software Process (TSP) [1, 4, 5] to build and sustain effective teams using the PSP. TSP creates self-directed teams with a defined process and responsibility for improving it. It also provides a practical measurement framework. For most organizations, TSP "out of the box" is a significant improvement over their current practice, but it incorporates a continuous improvement philosophy that encourages tailoring based on personal and team experience.

Our goal for this paper is to demonstrate how the TSP provides a framework for data collection and analysis for industrial software engineering projects. To do this, we describe a project at a multi-national company who worked with the Software Engineering Institute to pilot test the TSP. As we describe the project from start to completion, we explain what the TSP is, and the types of data that are collected and

---

SM *Personal Software Process, PSP, Team Software Process and TSP are service marks of* Carnegie Mellon University.

T. Dingsøyr (Ed.): EuroSPI 2004, LNCS 3281, pp. 91-101, 2004.