ICSOFT 2006

First International Conference on Software and Data Technologies

Proceedings

SETÚBAL, PORTUGAL · SEPTEMBER 11 - 14, 2006

Volume 1

ORGANIZED BY

INSTICC

SPONSORED BY

ENTERPRISE IRELAND

INSTITUTO POLITÉCNICO DE SETÚBAL

IN COOPERATION WITH

OMG OBJECT MANAGEMENT GROUP

INSTICC PRESS

Proceedings of ICSOFT
First International Conference on Software and Data Technologies
ISBN: 972-8865-69-4
ISBN (13 digits): 978-972-8865-69-6
http://www.icsoft.org

ICSOFT 2006

Proceedings of the
First International Conference on
Software and Data Technologies

Volume 1

Setúbal, Portugal

September 11 – 14, 2006

Organized by
INSTICC – Institute for Systems and Technologies of Information, Control and Communication

Sponsored by
Enterprise Ireland
Polytechnic Institute of Setúbal

In Cooperation with
Object Management Group (OMG)

Hosted by
School of Business of the Polytechnic Institute of Setubal

Edited by Joaquim Filipe, Boris Shishkov and Markus Helfert

Printed in Portugal

ISBN: 972-8865-69-4

ISBN (13 digits): 978-972-8865-69-6

Depósito Legal: 246927/06

http://www.icsoft.org

secretariat@icsoft.org

BRIEF CONTENTS

KEYNOTE LECTURES

Leszek A. Maciaszek

Macquarie University

Australia

Juan Carlos Augusto

University of Ulster at Jordanstown

U.K.

Tom Gilb

Norway

Dimitris Karagiannis

University of Vienna

Austria

Brian Henderson-Sellers

University of Technology

Australia

Marten J. van Sinderen

University of Twente

The Netherlands

TUTORIAL

Tom Gilb

Norway

ORGANIZING AND STEERING COMMITTEES

Conference Chair

Joaquim Filipe, INSTICC/Polytechnic Institute of Setúbal, Portugal

Program Co-chairs

Markus Helfert, Dublin City University, Ireland

Boris Shishkov, University of Twente, The Netherlands

Proceedings Production

Paulo Brito, INSTICC, Portugal

Marina Carvalho, INSTICC, Portugal

Helder Coelhas, INSTICC, Portugal

Bruno Encarnação, INSTICC, Portugal

Vitor Pedrosa, INSTICC, Portugal

Webdesign and Graphics Production

Marina Carvalho, INSTICC, Portugal

Secretariat and Webmaster

Mónica Saramago, INSTICC, Portugal

PROGRAM COMMITTEE

Hamideh Afsarmanesh, University of Amsterdam, The Netherlands

Jacky Akoka, CNAM & INT, France

Tsanka Angelova, Uniccord Ltd, Bulgaria

Keijiro Araki, Kyushu University, Japan

Lora Aroyo, Eindhoven University of Technology, The Netherlands

Colin Atkinson, University of Mannheim, Germany

Juan Carlos Augusto, University of Ulster at Jordanstown, U.K.

Elisa Baniassad, Chinese University of Hong Kong, China

Mortaza S. Bargh, Telematica Instituut, The Netherlands

Joseph Barjis, Georgia Southern University, U.S.A.

Noureddine Belkhatir, LSR-IMAG University of Grenoble, France

Fevzi Belli, University Paderborn, Germany

Alexandre Bergel, Trinity College, Ireland

Mohamed Bettaz, Philadelphia University, Jordan

Robert Biddle, Carleton University, Canada

Maarten Boasson, University of Amsterdam, The Netherlands

Wladimir Bodrow, University of Applied Sciences Berlin, Germany

Marcello Bonsangue, LIACS - Leiden University, The Netherlands

Jonathan Bowen, London South Bank University, U.K.

Mark van den Brand, Technical University of Eindhoven, The Netherlands

Lisa Brownsword, Software Engineering Institute, U.S.A.

Barrett Bryant, University of Alabama at Birmingham, U.S.A.

Cinzia Cappiello, Politecnico di Milano, Italy

Antonio Cerone, UNU-IIST, China

W. K. Chan, Hong Kong University of Science and Technology, China

Kung Chen, National Chengchi University, Taiwan

Samuel Chong, Accenture, U.K.

Chih-Ping Chu, National Cheng Kung University, Taiwan

Peter Clarke, Florida International University, U.S.A.

Rolland Colette, University Paris1 Pantheon Sorbonne, France

Alfredo Cuzzocrea, University of Calabria, Italy

Bogdan Czejdo, Loyola University, U.S.A.

David Deharbe, UFRN/DIMAp, Brazil

Serge Demeyer, University of Antwerp, Belgium

Steve Demurjian, University of Connecticut, U.S.A.

Nikolay Diakov, CWI, The Netherlands

Jan L. G. Dietz, Delft University of Technology, The Netherlands

Jin Song Dong, National University of Singapore, Singapore

Brian Donnellan, National University of Ireland, Ireland

Jürgen Ebert, University Koblenz, Germany

Paul Ezhilchelvan, University of Newcastle, U.K.

Behrouz Far, University of Calgary, Canada

Bernd Fischer, University of Southampton, U.K.

Gerald Gannod, Arizona State University, U.S.A.

Jose M. Garrido, Kennesaw State University, U.S.A.

Dragan Gasevic, Simon Fraser University, Canada

Nikolaos Georgantas, INRIA Rocquencourt, France

Paola Giannini, Università del Piemonte Orientale, Italy

Paul Gibson, National University of Ireland, Maynooth, Ireland

Wolfgang Grieskamp, Microsoft Research, U.S.A.

Daniela Grigori, University of Versailles, France

Klaus Grimm, Daimlerchrysler Ag, Germany

Rajiv Gupta, University of Arizona, U.S.A.

Tibor Gyimothy, University of Szeged, Hungary

Naohiro Hayashibara, Tokyo Denki University False, Japan

Jang Eui Hong, Chungbuk National University, Korea

Shinichi Honiden, National Institute of Informatics, Japan

PROGRAM COMMITTEE (CONT.)

COMMITTEE (CONT.)

Marten van Sinderen, University of Twente, The Netherlands

Joao Sousa, Carnegie Mellon University, U.S.A.

George Spanoudakis, City University, U.K.

Peter Stanchev, Kettering University, U.S.A.

Larry Stapleton, ISOL Research Centre, Ireland

Stoicho Stoichev, Technical University-Sofia, Bulgaria

Kevin Sullivan, University of Virginia, U.S.A.

Junichi Suzuki, University of Massachusetts, U.S.A.

Ramayah Thurasamy, Universiti Sains Malaysia, Malaysia

Yasar Tonta, Hacettepe University, Turkey

Yves Le Traon, France Télécom R&D, France

Enrico Vicario, University of Florence, Italy

Bing Wang, University of Hull, U.K.

Kun-Lung Wu, IBM Watson Research, U.S.A.

Hongwei Xi, Boston University, U.S.A.

Haiping Xu, University of Massachusetts Dartmouth, U.S.A.

Hongji Yang, De Montfort University, U.K.

Yunwen Ye, University of Colorado, U.S.A.

Yun Yang, Swinburne University, Australia

Gianluigi Zavattaro, University of Bologna, Italy

Xiaokun Zhang, Athabasca University, Canada

Jianjun Zhao, Shanghai Jiao Tong University, China

Hong Zhu, Oxford Brookes University, U.K.

Andrea Zisman, City University, U.K.

AUXILIARY REVIEWERS

Alessandro Aldini, Università degli Studi di Urbino, Italy

Pete Andras, University of Newcastle, U.K.

Xiaoshan Li, UNU-IIST, China

Shih-Hsi Liu, University of Alabama at Birmingham, U.S.A.

Michele Pinna, University of Bologna, Italy

Riccardo Solmi, University of Bologna, Italy

Hongli Yang, UNU-IIST, China

Chengcui Zhang, University of Alabama at Birmingham, U.S.A.

Liang Zhao, UNU-IIST, China

Wei Zhao, University of Alabama at Birmingham, U.S.A.

SELECTED PAPERS BOOK

A number of selected papers presented at ICSOFT 2006 will be published by Springer, in a book entitled Software and Data Technologies. This selection will be done by the conference chair and program co-chairs, among the papers actually presented at the conference, based on a rigorous review by the ICSOFT 2006 program committee members.

FOREWORD

This volume contains the proceedings of the first International Conference on Software and Data Technologies (ICSOFT 2006), organized by the Institute for Systems and Technologies of Information, Communication and Control (*INSTICC*) in cooperation with the Object Management Group (*OMG*), sponsored by Enterprise Ireland and the Polytechnic Institute of Setúbal and hosted by the School of Business of the Polytechnic Institute of Setubal.

The purpose of this conference is to bring together researchers, engineers and practitioners interested in information technology and software development. The conference tracks are "*Software Engineering*", "*Information Systems and Data Management*", "*Programming Languages*", "*Distributed and Parallel Systems*" and "*Knowledge Engineering*".

Software and data technologies are essential for developing any computer information system, encompassing a large number of research topics and applications: from programming issues to the more abstract theoretical aspects of software engineering; from databases and data-warehouses to management information systems and knowledge-base systems; Distributed systems, ubiquity, data quality and other related topics are included in the scope of ICSOFT.

ICSOFT 2006 received 187 paper submissions from more than 39 countries in all continents. To evaluate each submission, a double blind paper evaluation method was used: each paper was reviewed by at least two internationally known experts from ICSOFT Program Committee. Only 23 papers were selected to be published and presented as full papers, i.e. completed work (8 pages in proceedings / 30' oral presentations), 44 additional papers, describing work-in-progress, were accepted as short paper for 20' oral presentation, leading to a total of 67 oral paper presentations. There were also 26 papers selected for poster presentation. The full-paper acceptance ratio was thus 12%, and the total oral paper acceptance ratio was 35%.

In its program ICSOFT includes a panel to discuss the future of software development, by six distinguished world-class researchers; furthermore, the program is enriched by one tutorial and six keynote lectures. These high points in the conference program, involving top researchers worldwide, experts in different knowledge areas, have definitely contributed to reinforce the overall quality of the conference.

The program for this conference required the dedicated effort of many people. Firstly, we must thank the authors, whose research and development efforts are recorded here. Secondly, we thank the members of the program committee and the additional reviewers for their diligence and expert reviewing. I would like to personally thank the Program Chairs, namely Boris Shishkov and Markus Helfert, for their important collaboration. The local organizers and the secretariat have worked hard to provide smooth logistics and a friendly environment, so we must thank them all and especially Mónica Saramago for her patience and diligence in answering many emails and solving all the problems. Last but not least, we thank the invited speakers for their invaluable contribution and for taking the time to synthesize and prepare their talks.

A successful conference involves more than paper presentations; it is also a meeting place, where ideas about new research projects and other ventures are discussed and debated. Therefore, a social event including conference banquet was organized for the afternoon and evening of September 13 (Wednesday) in order to promote this kind of social networking.

We wish you all an exciting conference and an unforgettable stay in the lovely city of Setúbal. We hope to meet you again next year for the 2nd ICSOFT, in Barcelona (Spain), details of which will be shortly made available at http://www.icsoft.org.

Joaquim Filipe

INSTICC/Polytechnic Institute of Setúbal, Portugal

(Conference Chair)

CONTENTS

INVITED SPEAKERS

KEYNOTE LECTURES

Programming Languages

FULL PAPERS

SHORT PAPERS

POSTERS

Software Engineering

FULL PAPERS

POSTERS

A SYSTEMATIC REVIEW MEASUREMENT IN SOFTWARE ENGINEERING
State-of-the-art in Measures

Oswaldo Gómez, Hanna Oktaba
Institute of Investigations in Applied Mathematics and Systems, Autonomous National University of Mexico UNAM
Scholar Circuit University City, Coyoacán 04510, Mexico City, Mexico
oswaldog@uxmcc2.iimas.unam.mx, ho@hp.fciencias.unam.mx

Mario Piattini, Félix García
Alarcos Research Group, Department of Computer Science, University of Castilla-La Mancha
Paseo de la Universidad/4,13071,Ciudad Real, Spain
Mario.Piattini@uclm.es, Felix.Garcia@uclm.es

Keywords: Software Measurement, Measure, Systematic Review.

Abstract: The present work provides a summary of the state of art in software measures by means of a systematic review on the current literature. Nowadays, many companies need to answer the following questions: How to measure?, When to measure and What to measure?. There have been a lot of efforts made to attempt to answer these questions, and this has resulted in a large amount of data what is sometimes confusing and unclear information. This needs to be properly processed and classified in order to provide a better overview of the current situation. We have used a Measurement Software Ontology to classify and put the amount of *data* in this field in order. We have also analyzed the results of the *systematic review,* to show the trends in the software measurement field and the software process on which the measurement efforts have focused. It has allowed us to discover what parts of the process are not supported enough by measurements, to thus motivate future research in those areas.

1 INTRODUCTION

It is a well-known fact nowadays that software measurement helps us to better understand, evaluate, and control the products, processes, and software projects from the perspective of evaluating, tracking, forecasting, controlling and understanding (Ebert et al., 2004). On the one hand, software measurement allows organizations to know, compare and improve their software quality, performance, and processes. On the other hand, software measurement helps organizations to estimate and predict software characteristics to support better decisions (Pfleeger, 1997; Florac et al., 1999). As a consequence, software measures are proving to be very effective for understanding and improving software development and maintenance projects (Briand et al., 1996), showing problematic areas in system quality and institutionalizing software process improvement.

It should also be noted that there is a large amount of studies in software measurement, which makes it very easy to lose information and to get confused. For this reason, it is important to follow a specific, strict, and very well defined method for searching in the current literature. If we take a look at software measurement, we realize that it is considered to be among the youngest disciplines, and it is currently in the phase in which terminology, principles, and methods are still being defined and consolidated (Briand, 2002). This means that there is not a general agreement about the exact definitions of the main concepts related to measurement. In addition, no single standard contains a complete vision of software measurements (García et al., 2004).

With respect to the issues identified above, this article carries out a systematic review with a predefined search strategy, in order to summarize and classify the current and ongoing efforts in this field. The systematic review has been conducted according to the (Kitchenham et al 2004) proposal, which is very suitable for looking for information about measures on different sources in a disciplined and systematic way. Hence, Systematic review allows us to recognize, evaluate and do even more; it helps us to identify issues for planning future investigation and provides us with information about the consistency of our results (Travassos et al., 2005). We chose systematic review because of its scientific methodology that goes one step further than a simple overview.

The goal of this work is to find and clarify the answers to three different questions: What to measure, when to measure and how to measure. This is achieved by analyzing from the results of the literature review, the following issues: proportion of measured entities; measured attributes; validated measurement; measurement focus; and measurement in life cycle software process.

This paper is organized as follows. After this introduction; an overview of the systematic review process is given. In the third section, the way in which the systematic review has been carried out on the software measurement field is explained. Then, an analysis of the results is provided. Finally, the conclusions and future work are dealt with.

2 SYSTEMATIC REVIEWS

It is often recognized in Software Engineering that different research studies are generally fragmented and limited, not properly integrated, and without agreed standards (Kitcheham et al., 2004). In order to avoid those problems we chose the systematic review to carry out this investigation on software measures. Systematic review aims to present a fair evaluation of a research topic by using trustworthy, rigorous and auditable methodology, along with a very well defined strategy that allows the completeness of the research to be executed (in this case on software measures). Furthermore, systematic literature review is a formal and methodological process that allows us to identify, evaluate, and interpret all existing studies that are related to our investigation on software measures based in this case on a research question, but it could be also based on topic area, or phenomenon of interest. This is done in such a way that it helps us to summarize the evidence that is currently available concerning a treatment or technology. It also serves to identify any gaps in the current research, and thus suggest areas for further investigations, and finally provide a framework/background to position new research activities appropriately.

The review provides us with the necessary information to properly address the software measures, by mapping the measure field, finding the relevant data, ideas, techniques and their correlation with our investigation. Besides, it can support the planning for a new piece of research. Moreover, with this systematic literature review we can integrate empirical investigation, in order to find out generalizations. We do this by establishing specific objectives to create critical analysis. An overview of the systematic review is provided in the next subsection.

2.1 The Systematic Review Process

In order to address and present a fair evaluation of a research topic, the systematic review is composed of the following phases:

Review Planning Phase: Here the investigation's goals are established. The *Review Protocol*, which is the most important item in this phase, is generated. First and foremost, this protocol defines the research question and the methods that will be executed in the review. In a broad manner, this phase involves the following, summarized, activities, defined by (Travassos et al., 2005):

Question Formulization: This activity is considered to be among the most important in the systematic review process. Here the investigation targets must be defined by focusing the question and by establishing its Quality and Amplitude.

Source Selection: Primary studies from sources are selected here, by defining a source selection criterion, setting the studies' languages, identifying and selecting the sources after an assessment of them and checking references.

Study Selection: It describes the process and criteria for the evaluation and selection of studies.

Review Execution phase: This phase involves identification, selection and evaluation of primary studies, based on the inclusion and exclusion criteria defined in the *Review Protocol*. It is composed of the following steps, in summary form:

Selection Execution: This section aims to register the selection process for primary studies by evaluating them with quality criteria.

Information Extraction: Once primary studies are selected, the relevant data must be extracted by following an *Information Inclusion and Exclusion Criteria Definition*, by defining *Data Extraction Forms*, and by resolving divergences among reviewers.

Result Analysis: In this phase all the information from the different studies is analyzed. This phase involves the next step: *Result Summarization*, which presents the data resulting from the collected studies by doing *Calculus Statistical*, *Results Tables*, *Sensitivity Analysis*, *Plotting*, which will lead to the *Conclusion* and *Final Comments*.

The whole process must be stored and the planning and the execution have to guarantee that the research can be done. It is worth mentioning here that the *Review Protocol* must be evaluated by experts. Finally, many of the activities of the review process involve iteration to refine the process, and therefore they are not necessarily sequential.

In the next section, we describe how the review process, which was designed as appropriate to our research goals, was performed

3 SYSTEMATIC REVIEW ABOUT SOFTWARE MEASURES

First of all, it must be emphasized that this paper is an attempt to answer this fundamental question: What are the most current and useful measures in the literature? Since our whole protocol was produced around this question, this is the main step in our *Review Planning Phase*. Moreover, we hope that this work will be useful for project managers and software developers. The defined strategy was the following: first and foremost, the large collection of paper in current literature about software measurements was examined. Due to the great diversity of topics in this field, and with the aim of clarifying and summarizing them in the best way possible, we used the classifications of concepts defined in the Software Measurement Ontology proposed by (García et al., 2004). This ontology aims at contributing to the harmonization of the different software measurement proposals and standards, by providing a coherent set of common concepts used in software measurement.

In order to do the research we built the following combinations of search strings:

"(measure OR metric OR quality OR quantitative) AND (process OR engineering OR maintenance OR management OR improvement OR Software testing OR development)".

All the possible combinations with these words were tested in the following web search engines: ACM Digital Library, Search IEEE magazines, Wiley Interscience, and Science@Direct.

The results obtained on the web engines are shown in Table 1.

Table 1: Total Search Results.

Sources	Search Results	Reviewed	Accepted
Science@Direct	3569	78	10
ACM	950	85	28
IEEE	3740	111	32
Wiley	653	20	8
TOTAL	8912	294	78

As we can see in Table 1, search engines provided us with 8912 papers. Nevertheless, it should be pointed out that only 78 were accepted, which represents about 1 % of the total articles, hardly even that. It is apparent that many articles were rejected. This is so because if a more limited search had been carried out, it would certainly have been true that we would have started with fewer results from the search engines, but at the same time we would have lost important articles. Therefore, a very less restrictive search was defined: as a result of this, we obtained too many articles, of which very few were considered apt. Furthermore, we have discarded those measures that were outside the scope of our model. We have also discarded measures that did not provide any relevant information, as well as repeated measures proposed by more than one author so that each measure is included only once. Hence, our attention focused on papers where keywords and titles included the research strings. These strings were also searched for in the whole document by some search engines.

Regarding the execution phase of the systematic review, the selection and evaluation of information was initiated using the terms of the inclusion and exclusion criteria defined in the review protocol. These criteria established that selected studies were in English and that all of them showed current, useful software measurements, basically only studies about measures for software development, software project administration and maintenance were selected. All papers had to satisfy our quality criteria and in this sense it is important to point out that all the searched-for sources are serious and that the quality of their papers is guaranteed. Moreover the search engines were validated by experts. For this reason, our quality criteria also trusted in the quality of the sources.

Once the papers were selected, the information was extracted by means of an extraction template for objective results which includes study name, author, institution, journal, date, methodology, results, problems and subjective results which includes information through authors, general impressions and abstractions, according to the proposal provided by Trassvasos et al., (2005); in particular, the aims of this template are to store the results of the execution phase process by extracting, not only the objective information, but also the subjective information from each article analyzed.

Finally, in the results analysis phase we analyzed the measures in order to show, among other aspects, the information about attributes, the entities measured and their characteristics, the amount of measures in a specific attribute or entity, etc. This phase is described in more detail in the following section.

4 RESULT ANALYSIS

The measures extracted from the studies were summarized in terms of the Software Measurement Ontology, which helped us to find out what kinds of measures exist. More specifically, this ontology supported us in defining a template by categorizing the measures in the following three different ways: What to measure? How to measure? And When to measure?

Consequently, in order to summarize the existing measures, the ISO 15504, CMM, and CMMI establish a quality background for the improvement of maturity levels defining the Project, Process and Product as the kind of *entities* that can be measured. That is why we extracted *attribute* and *sub-attributes* (Fenton and Pfleeger, 1997) measured of these *entities*, from the articles reviewed and classified them into internal or external. With this part of the analysis we try to answer the question: What to measure? This is the first way in which we categorized the measurements. Table 3 shows these attributes.

Table 3: Definition of entities.

What?						
Entibies			Attributes	Sub-attributes	**Type of Attribute**	
Project	Process	Product			Internal	External

Once the measurements were collected and stored in our template table, we analyzed the amount of measures which have been defined for the Process, Project and Product kind of entities. As we can see in Figure 1, the most measured kind of entity is the product, and the entities whose measurement has been less supported by the current literature are the project and process. The reason is that measuring product is easier than measuring process and project, in which we usually find ambiguous definition of attributes. For products, quality and technical attributes are very well defined because quality has been strongly focused on product. Finally, measurements on product entities help to measure process and project ones.

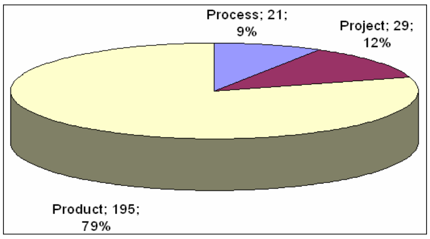

Figure 1: Proportion of measured entities.

Next, we shall look at another closely-related issue, which is the amount of measured attributes. Figure 2 shows the proportion of measure attributes according to our analysis of the accepted papers. As Figure 2 shows, size is one of the most measured attributes. The point is that the size is a base measure, not only needed in most of the derived measures, but the size measure is also easier to obtain because it focuses on one of the most "tangible" attributes which is the source code. Moreover, size has very well defined scales, units and methods of measurement like functions Points (FP) (IFPUG, 2004); therefore it is very difficult to get confused with size measurements. Furthermore, cost estimation is derived from size and the overall productivity, and finally the schedule is based on the size and cost estimates (Ebert et al., 2004). Hence size is used on most of control measures in a software project. The arguments set out here lead to an explanation of why size has one of the highest values in Figure 2.

In order to show in a in a better way the information displayed in Figure 2, Table 4 show the attributes order by the most measured.

In connection with the most measured attributes, the complexity attribute is used in different contexts, for example: source code complexity, Design complexity, UML Diagrams complexity, Architecture complexity, etc. Hence it can be seen that complexity has gathered many measurements from its different applications. If we take a look at Figure 2 in greater detail, it should be pointed out that attributes like Activity, Role, Work products and Accuracy are the least measured. That is due to the fact that these attributes are mostly related with

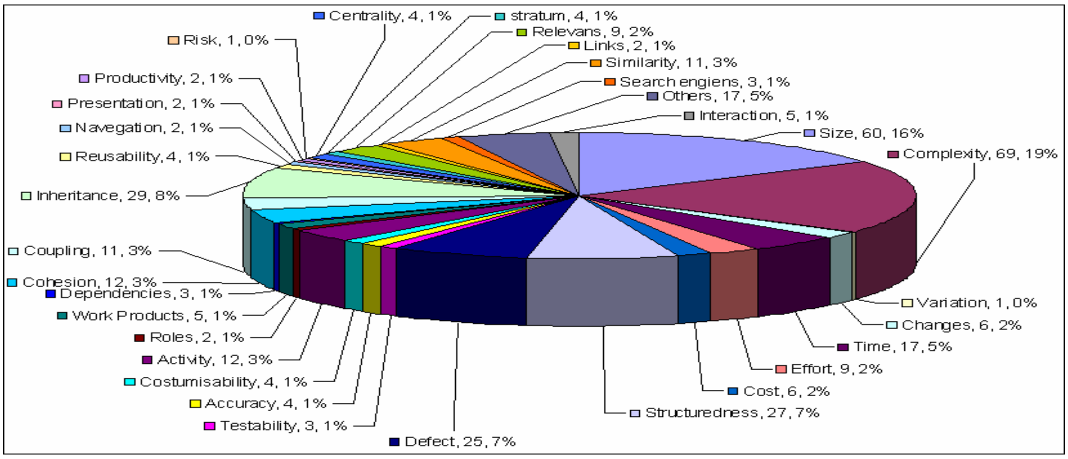

Figure 2: Measured attributes.

Table 4: Measures attributes.

Complexity	19%	Productivity	1%
Size	16%	Testability	1%
Inheritance	8%	Costumisability	1%
Defect	7%	Roles	1%
Structuredness	7%	Work Products	1%
Time	5%	Dependencies	1%
Others	5%	Reusability	1%
Activity	3%	Navegation	1%
Accuracy	3%	Presentation	1%
Cohesión	3%	Centrality	1%
Coupling	3%	Stratum	1%
Similarity	3%	Links	1%
Changes	2%	Search engiens	1%
Effort	2%	Interaction	1%
Cost	2%	Variation	0%
Relevans	2%	Risk	0%

process and project kind of entities, for which there is not a well defined basic attribute.

Once the "What to Measure?" question was analyzed. The next step was to tackle the question: "How to measure?" To answer this question we gathered how the measurements of attributes in the selected papers were made and classified them in terms of the following characteristics: Representation, Description, Base or Derived Measurement, Scale (Fenton y Pfleeger, 1997), Empirically (Wohlin et al., 2000; Juristo and Moreno, 2001; Basili et al., 1999; Perry et al., 2000) or Theoretically (Weyuker, 1988; Briand et al., 1996; Whitmire, 1997; Zuse, 1998; Poels y Dedene, 2000) validated. This analysis is summarized in Table 5.

Let us have a look at the last characteristic, which has as its goal to discover if a measure has been validated empirically and/or theoretically. The aim of theoretical validation is to check whether the intuitive idea of the attribute being measured is considered in the defined measure. The main goal of empirical validation is to obtain objective information concerning the usefulness of the proposed metrics. Theoretical validation by itself is not enough to guarantee the usefulness of the measure, because it may occur that a measure is valid from a theoretical point of view, but it has no practical relevance in relation to a specific problem. As a consequence, a measure which has not been validated is not demonstrated to be useful. We therefore classified the measures in such a way as to know how many had been empirically and/or theoretically validated. This is shown in Figure 3.

As can be observed in Figure 3, about half of the measures found in the selected papers had been only empirically validated. This leads us to the conclusion that there is a great tendency to empirical validation. Furthermore, we can see that (24%) of the measurements had been validated only theoretically, although it was recognized in the papers that they need empirical validation. Finally only (20%) of the measurement had been both empirically and theoretically validated. It should be pointed out that it is necessary to get a common agreement to validated measures theoretically. Moreover empirically validation needs more data extracted from "real projects" in order to get practical conclusions.

Regarding the measurement focus found in the articles analysed, we have discovered the following approaches: Structured (Briand et al., 1996a), measurement focussing in Process, Object Oriented (OO) (Chidamber y Kemerer., 1994; Brito e Abreu y

Table 5: Definition of measure attributes.

HOW?						
		Measure				
Representation	**Description**	**Based**	**Derived**	**Scale**	***Validation***	***Measure* focus**

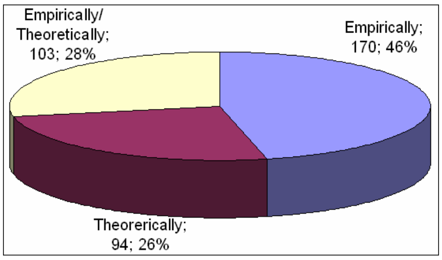

Figure 3: Validated measures.

Carapuca, 1994; Lorenz y Kidd, 1994; Marchesi, 1998; Bansiya et al., 1999; 2002), Quality (Piattini y García., 2003). Function Points (IFPUG Release 4.2, 2004), UML (Marchesi, 1998), Complexity (McCabe, 1976; Henry y Kafura, 1981), *Project* (Putnam y Myers, 1992) and OCL (Reynoso et al., 2004).Figure 4 shows the amount of measurement in each approach. It shows us that the most supported approaches by measure are Object Oriented (OO) ones. This is due to this kind of projects are currently the most popular in software development. Continue with this part of the analysis, there are efforts to get a universal WEB measures definition, with this review we found conceptual models and frameworks in order to classify WEB measures.

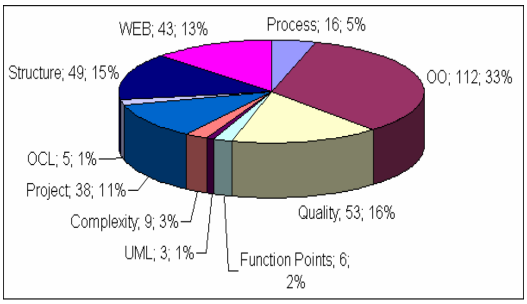

Figure 4: Measure focus.

Finally, we analyzed the third question: When to measure?, To classify in what parts of the lifecycle project the measure must be taken for projects and process entities, the PMBOK guide (ANSI/PMI, 2004) was selected. In order to group when the measurements are taken for the product entity, the waterfall lifecycle model was applied. We chose these two models due to their wide acceptation and genericity. Figure 5 shows the proportion of product measurements in the different phases of the software life cycle:

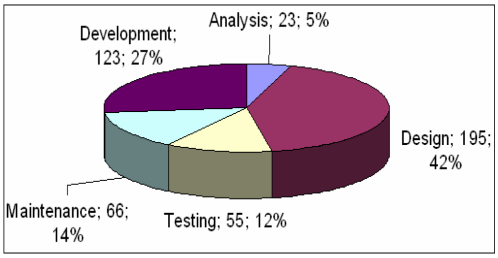

Figure 5: Measure in life cycle software process.

As we can see in Figure 5, most measurements are carried out during the Design, Testing and Development phases of the waterfall lifecycle software process. In the Design phase, products such as architecture, system designs, requirements analysis, etc. are generated. Hence it is necessary to support this phase with measurements, in order to know characteristics of these products when carrying out the design. Moreover, measurement in the Design phase can support the future products to be generated, which mean that this phase is one of the most measured. Continue with this analysis, it should be pointed out that the Development phase is one of the most measured, because most of the software products are created here, such as: manuals, source code and, among other products, the software itself. Therefore, it is possible to collect quantity information about these products here. According to PSP (Humphrey, 2005), measures about size, effort, time, faults, defects, LOC, etc. are commonly taken in this phase. Another factor to take into account is that once the software system is created, it is necessary to validate if this system fulfils the quality requirements. The counting faults and deriving the reliability is the most widely applied and accepted method used to validate systems; most of this information focuses on the product and is commonly reported in terms of measurements. This is done in not only in the early phases but also especially in the testing phase, which is another of the most-measured phases in lifecycle software process.

In addition, the PMBOK guide defines the following general phases for project life: Initial, intermediate, and final phases. In Figure 6 we show the distributions of measures through these phases.

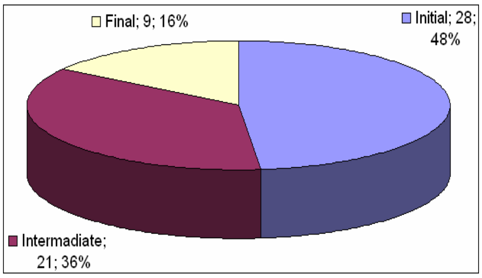

Figure 6: Measure in life cycle projects.

It is worth mentioning here that in the initial phase there could be sub-phases with one or more deliverables, according to the kind of project. In these sub-phases the following are usually measured: size, complexity, level of risk, cash, etc. Most measurements concentrate on the Initial phase, as in this phase the planning for the whole project is executed- this in turn constitutes the main effort in project management. In the Intermediate phase, many control activities are carried out in order to ensure the success of the project. Periodical reports are thereby generated with quantity information about process and project measures and indicators. For these reasons this phase is also one of the most measured in project lifecycle for project and process entities.

5 CONCLUSIONS AND FURTHER WORK

Software measurements are very important in software development process, because they help us, to control, estimate and improve process, projects and products, among other things.. With that in mind, this article attempts to provide the state of art in software measurement, by carrying out a systematic review whose purpose is to summarize the most current and useful measures in the literature.

With this systematic review, we find out the following results:

(1) Measures are strongly aligned to product entity. Since this kind of entity has better attribute definition than project and product entities have, there are large amount of measures for the product. This leads to the conclusion that if an entity has a few measures, it is due to the fact that it doesn't have specific attribute.

(2) Complexity gathered a great amount of measures because this attribute is used in different contexts. While size is also one of the most measured attributes since it is used in cost and development schedule estimation

(3) There is a great tendency to obtain empirical validation. But it is necessary to get more data extracted from "real projects", in order to get practical conclusions and to improve software quality.

(4) Development and Design are the most measured phases in lifecycle software process because it is in these phases that most software products are generated.. It should be also noted that the testing phase is also one of the most measured phases. This is thanks to the fact that this phase involves quality activities for evaluating software quality characteristics, generally reported in terms of quantity values. But quality measures are considering in the early software development phases by counting faults which is the most widely applied method to determine software quality.

(5) For projects and process entities most measurements are concentrated in the Initial and Intermediate phases. That is because it is here that the project planning and control activities are developed.

(6) There are a large number of measures for OO projects. This is because these kinds of projects are currently the most popular in software development. Hence a lot of research has been done in this field.

(7) So many efforts had been made to get a universal WEB measures definition. In this review we found conceptual models and frameworks in order to classify WEB measures.

Finally, we need to relate the measurements found in this article to a specific software development process. The aim of this is to settle *when* a measure must be taken. To reach this goal, in our specific research, further work will take in the Process Model for the Software Industry (MoProSoft), which focuses on small companies and which is also the Mexican norm.

ACKNOWLEDGEMENTS

This article was supported by the Process Improvement for Promoting Iberoamerican Software the Competitiveness of Small and Medium Enterprises (COMPETISOFT) and Science and Technology for Development (CYTED).

REFERENCES

ANSI/PMI (2004), *A Guide Project Management Body of Knowledge (PMBOK Guide) an American National Standard*, ANSI/PMI 99-001-2004, Third Edition, Project Management Institute, Inc., United Estates of America.

Basili, V., Shull, F. y Lanubile, F., 1999, Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4), pp. 435-437.

Bansiya J. y Davis C., 2002, *A Hierarchical Model for Object-Oriented Design Quality Assessment*. IEEE Transactions on Software Engineering, 28(1), 4-17.

Briand, L., Morasca, S. y Basili, V., 1996. Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering*, 22(1), pp. 68-86.

Brito e Abreu, F. y Carapuça, R., 1994, Object-Oriented Software Engineering: Measuring and controlling the development process. *Proceedings of the 4th International Conference on Software Quality*, McLean (USA).

Calero, C., Ruiz, J., Piattini, M., 2005, Classifying web metrics using the web quality model. *Online Information Review.* Vol 29 No 3, pp. 227-248

Chidamber, S. y Kemerer, C., 1994. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6), pp. 476-493.

Ebert, C., Dumke R., Bundschuh, M., Schmietendorf, A., 2004, *Best Practices in Software Measurement. How to use metrics to improve project and process performance*, 295 Seiten-Springer. Berlin, 1st Edition.

Fenton, N. y Pfleeger, S.L., 1997, *Software Metrics: A Rigorous & Practical Approach*, PWS Publishing Company, Second Edition.

Florac, W. A., Carleton, A. D., 1999. *Measuring the Software Process. Statistical Process Control for Software Process Improvement*, Addison-Wesley. United States of America, 1st Edition.

García, F., Bertoa, M. F., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M., Genero, M., 2005. Towards a consistent terminology for software measurement. *Information and Software Technology.* xx (2005), pp. 1-14

Henry, S. y Kafura, S., 1981, Software Structure Metrics Based on Information Flow. *IEEE Transactions on Software Engineering*, 7(5), pp. 510-518.

Humphrey, S.H., 2005, *PSP A Self-Improvement Process for Software Engineers*, Addison-Wesley. United States of America, 1st Edition.

IFPUG, (2004), *IFPUG: Function Point Counting Practices Manual*, *Release 4.2*. International Function Point Users Group, USA –IFPUG, Mequon, Wisconsin, USA.

Juristo, N. y Moreno, A. (2001). *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers.

Kitchenham, B., 2004. *Procedures for Performing Systematic Reviews*. Joint Technical Report Software Engineering Group, Department of Computer Science Keele University, United King and Empirical Software Engineering, National ICT Australia Ltd, Australia, pp. 1-28.

Lorenz, M. y Kidd, J., 1994, *Object-Oriented Software Metrics: A Practical Guide*. Prentice Hall. Englewood Cliffs, Nueva Jersey.

McCabe, T., 1976. A Software Complexity Measure. *IEEE Transactions on Software Engineering*, 2, pp. 308-320.

Marchesi, M., 1998. OOA Metrics for the Unified Modeling Language. 2nd Euromicro Conference on Software Maintenance and Reengineering, 1998, 67-73

Perry, D., Porte, A. y Votta, L. (2000). Empirical Studies of Software Engineering: A Roadmap. Future of Software Engineering, Ed. Anthony Finkelstein, ACM, pp. 345-355.

Pfleeger, S. L.,1997. Assessing Software Measurement. *IEEE Software.* March/April. pp. 25-26.

Piattini, M., García, F. O., 2003. *Calidad en el desarrollo y mantenimiento de software*, Ra-Ma. Spain, 1st Edition.

Poels, G. y Dedene, G. (2000). Distance-based software measurement: necessary and sufficient properties for software measures. *Information and Software Technology*, 42(1), pp. 35-46.

Putnam, L. H. y Myers, W., 1992. *Measures for Excellence - Reliable software on time, within budget*, Prentice Hall, New Jersey.

Raynus, J., 1999. *Software Process Improvement with CMM*, Artech House. United States of America, 1st Edition.

Reynoso L., Genero M. y Piattini M. Measuring OCL Expressions: An Approach Based on Cognitive Techniques, 2004. Chapter 5 in "Metrics for Software Conceptual Models" (Eds. Genero M., Piattini M. and Calero C.). Imperial College Press, UK.

Travassos G. H., Boilchi, J., Mian , P. G., Natali, A. C. C., 2005. *Systematic Review in Software Engineering*. Technical Report Programa de Engenharia de Sistemas e Computaçâo PESC, Systems Engineering and Computer Science Department COPPE/UFRJ, Rio de Janerio, pp. 1-30.

Weyuker, E., 1988. Evaluating Software Complexity Measures. *IEEE Transactions on Software Engineering*, 14(9), pp. 1357-1365.

Whitmire, S., 1997. *Object Oriented Design Measurement*. John Wiley & Sons, Inc.

Wohlin, C., Runeson, P., Höst, M., Ohlson, M., Regnell, B. y Wesslén, A.,2000. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers.

Zuse, H. (1998). *A Framework of Software Measurement.* Berlin. Walter de Gruyter.