

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <<http://www.upgrade-cepis.org>>

UPGRADE is the anchor point for UPENET (UPGRADE European NETWORK), the network of CEPIs member societies' publications, that currently includes the following ones:

- Mondo Digitale, digital journal from the Italian CEPIs society AICA
- Novática, journal from the Spanish CEPIs society ATI
- Piroforiki, journal from the Cyprus CEPIs society CCS
- Pro Dialog, journal from the Polish CEPIs society PTI-PIPS

#### Publisher

UPGRADE is published on behalf of CEPIs (Council of European Professional Informatics Societies, <<http://www.cepis.org/>>) by Novática <<http://www.ati.es/novatica/>>, journal of the Spanish CEPIs society ATI (Asociación de Técnicos de Informática <<http://www.ati.es/>>).

UPGRADE is also published in Spanish (full issue printed, some articles online) by Novática, and in Italian (abstracts and some articles online) by the Italian CEPIs society ALSI <<http://www.alsi.it/>> and the Italian IT portal Tecnoteca <<http://www.tecnoteca.it/>>.

UPGRADE was created in October 2000 by CEPIs and was first published by Novática and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <<http://www.svifsi.ch/>>).

#### Editorial Team

Chief Editor: Rafael Fernández Calvo, Spain, <[rfoalvo@ati.es](mailto:rfoalvo@ati.es)>  
Associate Editors:

- François Louis Nicolet, Switzerland, <[nicolet@acm.org](mailto:nicolet@acm.org)>
- Roberto Carniel, Italy, <[carniel@dgt.uniud.it](mailto:carniel@dgt.uniud.it)>
- Zakaria Maamar, Arab Emirates, <[Zakaria.Maamar@zu.ac.ae](mailto:Zakaria.Maamar@zu.ac.ae)>
- Soraya Kouadri Mostéfaoui, Switzerland, <[soraya.kouadrimostefaoui@unifr.ch](mailto:soraya.kouadrimostefaoui@unifr.ch)>

#### Editorial Board

Prof. Wolfgang Stucky, CEPIs Past President  
Prof. Nello Scarabottolo, CEPIs Vice President  
Fernando Piera Gómez and Rafael Fernández Calvo, ATI (Spain)  
François Louis Nicolet, SI (Switzerland)  
Roberto Carniel, ALSI – Tecnoteca (Italy)

#### UPENET Advisory Board

Franco Filippazzi (Mondo Digitale, Italy)  
Rafael Fernández Calvo (Novática, Spain)  
Panicos Masouras (Piroforiki, Cyprus)  
Andrzej Marciniak (Pro Dialog, Poland)

**English Editors:** Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green, Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson.

Cover page designed by Antonio Crespo Foix, © ATI 2004

Layout: Pascale Schürmann

Editorial correspondence: see "Editorial Team" above

Advertising correspondence: <[novatica@ati.es](mailto:novatica@ati.es)>

Upgrade Newsletter available at

<<http://www.upgrade-cepis.org/pages/editinfo.html#newsletter>>

#### Copyright

© Novática 2004 (for the monograph and the cover page)

© CEPIs 2004 (for the sections MOSAIC and UPENET)

All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, contact the Editorial Team.

The opinions expressed by the authors are their exclusive responsibility.

ISSN 1684-5285

Next issue (December 2004):  
"Cryptography"

(The full schedule of UPGRADE is available at our website)

- 2 Editorial  
Four Years of UPGRADE

## SPT, Software Process Technology

Guest Editors: Francisco Ruiz-González and Gerardo Canfora

### Joint monograph with Novática\*

- 3 Presentation  
Software Process Technology: Improving Software Project Management and Product Quality – *Francisco Ruiz-González and Gerardo Canfora*
- 6 Software Process: Characteristics, Technology and Environments – *Francisco Ruiz-González and Gerardo Canfora*
- 11 Key Issues and New Challenges in Software Process Technology – *Jean-Claude Derniame and Flavio Oquendo*
- 17 A Taxonomy of Software Engineering Environment Services: The Upcoming ISO/IEC Standard 15940 – *Dan Hyung Lee and Juan Garbajosa-Sopeña*
- 22 Open Source and Free Software: A New Model for The Software Development Process? – *Alfonso Fuggetta*
- 27 Applying The Basic Principles of Model Engineering to The Field of Process Engineering – *Jean Bézivin and Erwan Breton*
- 34 Software Process Modelling Languages Based on UML – *Pere Botella i López, Xavier Franch-Gutiérrez, and Josep M. Ribó-Balust*
- 40 Supporting the Software Process in A Process-centred Software Engineering Environment – *Hans-Ulrich Kobialka*
- 47 Managing Distributed Projects in GENESIS – *Lerina Aversano, Andrea De Lucia, Matteo Gaeta, Pierluigi Ritrovato, and Maria-Luisa Villani*
- 53 Software Process Measurement – *Félix García-Rubio, Francisco Ruiz-González, and Mario Piattini-Velthuis*
- 59 Process Diversity and how Practitioners Can Manage It – *Danilo Caivano and Corrado Aaron Visaggio*

## MOSAIC

- 67 Data Architecture  
A Disquisition on The Performance Behaviour of Binary Search Tree Data Structures – *Dominique A. Heger*
- 75 News & Events: News from CEPIs and EUCIP; SCI 2005 (Call for Papers)

## UPENET (UPGRADE European NETWORK)

- 77 From **Novática** (Spain):  
IT and Disabilities  
Braille and The Pleasure of Reading: We Blind People Want to Continue Reading with Our Fingers – *Carmen Bonet-Borrás*
- 84 From **Piroforiki** (Cyprus):  
Information Technology in Today's Organizations  
Is the IT Productivity Paradox Resolved? – *Kyriakos E. Georgiou*

\* This monograph will be also published in Spanish (full issue printed; summary, abstracts and some articles online) by **Novática**, journal of the Spanish CEPIs society ATI (*Asociación de Técnicos de Informática*) at <<http://www.ati.es/novatica/>>, and in Italian (online edition only, containing summary abstracts and some articles) by the Italian CEPIs society ALSI (*Associazione nazionale Laureati in Scienze dell'informazione e Informatica*) and the Italian IT portal Tecnoteca at <<http://www.tecnoteca.it/>>.

# Software Process Measurement

*Félix García-Rubio, Francisco Ruiz-González, and Mario Piattini-Velthuis*

*The measurement of software processes plays a vital role in their improvement as it provides the necessary quantitative basis for the identification of aspects on which to focus improvement programmes. However, the measurement of software processes is no easy task due to the great diversity of factors and elements involved. Thus, in order to be able to measure processes effectively and to facilitate improvement focused decision-taking, we need to identify which entity types we want to measure. We also need to carry out measurement programmes that, in addition to measuring the relevant entities in isolation, enable the information obtained from the measurement process to be integrated and related. This article provides a general overview of the software process measurement, highlighting its importance in improvement focused process management. The relevant entities that can be measured in relation to the process are also identified and an example is given of how to measure one of these entity types: process models.*

**Keywords:** Process Measurement, Software Metrics, Software Process, Software Process Improvement, Software Process Model.

## 1 Introduction

The improvement of software processes has become one of the main aims of companies dedicated to the development and maintenance of computing systems. The need to improve processes stems from the fact that the quality of a process is closely related to the quality of the product, which means that

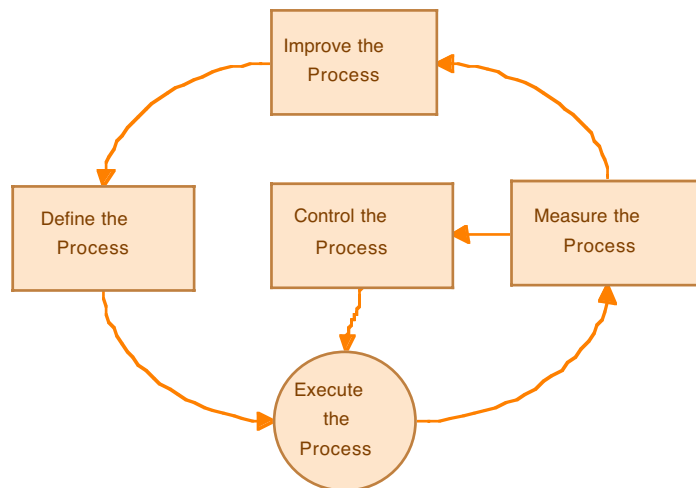
to in order to get better products you need to have better processes. If software processes are to meet quality requirements, they need to deliver the expected results, be correctly defined, and be improved as required by business needs, which in competitive companies can be highly changeable. These are the objectives of “Software Process Management” (SPM) [5] which, in order to be applied effectively, should address four key responsibilities: to define, measure, control and improve the process. These responsibilities and the way they relate to one another are set out in Figure 1.

*Félix García-Rubio* has an MSc in Computer Science from the *Universidad de Castilla-La Mancha*, Spain. He is currently his PhD studies at UCLM. He is an Assistant Professor in the Department of Computer Science at UCLM, in Ciudad Real, Spain. He has authored several papers and book chapters on software processes management, from the point of view of their modelling, measurement and technology. He is a member of the Alarcos Research Group, <<http://alarcos.inf-cr.uclm.es/english/>>, specialized in information system quality. His research interests are software processes and software measurement. <[Felix.Garcia@uclm.es](mailto:Felix.Garcia@uclm.es)>

*Francisco Ruiz-González* has a PhD in Computer Science from the *Universidad de Castilla-La Mancha* (UCLM), Spain, and an MSc in Chemistry-Physics from the *Universidad Complutense de Madrid*, Spain. He is a full time Associate Professor of the Department of Computer Science at UCLM in Ciudad Real, Spain. He was the Dean of the Faculty of Computer Science between 1993 and 2000. Previously, he was the director of computer services at the aforementioned university (1985–1989) and he has also worked in private companies as an analyst-programmer and project manager. He is a member of the Alarcos Research Group, <<http://alarcos.inf-cr.uclm.es/english/>>. His current research interests include software process technology and modelling, software maintenance, and methodologies for software projects planning and management. He has also worked in the fields of GIS (Geographical Information Systems), educational software systems and deductive databases.

He has written eight books and fourteen chapters on the abovementioned topics and he has published 90 papers in Spanish and international journals and conferences. He has sat on nine programme committees and seven organizing committees and he belongs to several scientific and professional associations: ACM, IEEE-CS, ATI, AEC, AENOR, ISO JTC1/SC7, EASST, AENUI and ACTA. <[Francisco.RuizG@uclm.es](mailto:Francisco.RuizG@uclm.es)>

*Mario Piattini-Velthuis* has an MSc and a PhD in Computer Science from the *Universidad Politécnica de Madrid*, Spain, and is a Certified Information System Auditor Manager by ISACA (Information System Audit and Control Association). He is a full professor in the Department of Computer Science at the *Universidad de Castilla-La Mancha*, Ciudad Real, Spain. He has authored several books and papers on databases, software engineering and information systems, and leads the ALARCOS research group of the Department of Computer Science at the same university. His research interests are: advanced database design, database quality, software metrics, and software maintenance and security in information systems. He has co-edited several books: “Advanced Databases: Technology and Design”, 2000, Artech House, UK; “Auditing Information Systems” Idea Group Publishing, 2000, USA; “Information and database quality”, 2002, Kluwer Academic Publishers, USA, etc. and “Metrics for Software Conceptual Models”, 2004, Imperial College Press, UK. He is co-editor of the Database section of *Novática*. <[mario.piattini@uclm.es](mailto:mario.piattini@uclm.es)>



**Figure 1:** Key Responsibilities of Software Process Management.

According to these responsibilities (which are fundamental to the successful management of software processes) in order to improve a process efficiently it is necessary to bear in mind the following aspects:

- **Process definition.** This is the first key responsibility that must be addressed in order to provide effective management orientated towards process improvement. To do this it is necessary to model the processes; that is, to represent the elements of interest involved in those processes. Given the diversity of elements that need to be taken into account when considering a software process, the definition of software processes is by no means a simple task. Various modelling languages and formalisms, known as “Process Modelling Languages” (PML), can be found in literature on the subject. The aim of these languages is to represent the various interrelated elements in a precise and unambiguous way. In a software process it is usually possible to identify the following elements or general concepts (albeit using different notations and terms) in the different PMLs [3]: Activity, Product, Resource, Organization and Role. In order to provide a common reference for the representation of software processes, the OMG consortium (Object Management Group) has for several years been working on the development of the metamodel SPEM (Software Process Engineering Metamodel) [14], which is a generic language extending UML (Unified Modelling Language) for the descriptive modelling of software processes without including aspects related to their enactment. Currently, SPEM is a specification which is expected to produce a process modelling standard which may be as important and universally accepted by industry as the UML standard is today.
- **Process Execution and Control.** A company’s software projects should be carried out in accordance with defined process models. It is important to be able to be in permanent control of the execution of these projects (and consequently in control of the corresponding processes) in order to ensure that the expected results are achieved.

- **Measurement and Improvement.** There is a significant correlation between the measurement and the improvement of software processes. Prior to improving a process, it is necessary to carry out an assessment process to identify which aspects of the process can be improved. To do this we need to establish an effective framework to help us identify the most important entities to measure. The results of measuring processes provide non-subjective information enabling the necessary actions for improvement to be planned, identified and carried out in an efficient manner.

In this article we aim to look at improvement focused software process measurement. The following section describes the most important entities related to software processes from a measurement point of view. Section 3 presents a representative set of metrics for evaluating the structural complexity of software process models. The final section outlines some conclusions and looks at some issues yet to be resolved.

## 2 Software Process Measurement Entities

One of the main reasons for the growing interest in software metrics has been the increasing awareness that metrics are necessary for process improvement [4]. Before it is possible to apply improvement plans in an organization we need a quantitative basis enabling us to make an objective determination of the strengths and weaknesses of its processes. Software metrics provide the base which enables us to carry out the assessment process and the subsequent improvements to the processes evaluated. Consequently, measurement is given considerable importance in assessment models such as ISO/IEC 15504 [10], which defines a measurement process, or CMMI (Capability Maturity Model Integration) [16], which includes a key process area at maturity level two called “Measurement and Analysis”. With regard to support for the measurement process, there are several interesting frameworks, such as GQM (Goal Question Metric) [1][15] or PSM (Practical Software Measurement) [12], plus some standards, the most important of which are ISO 15939 [12] and IEEE Std 1061-1992 [9]. The objective of these

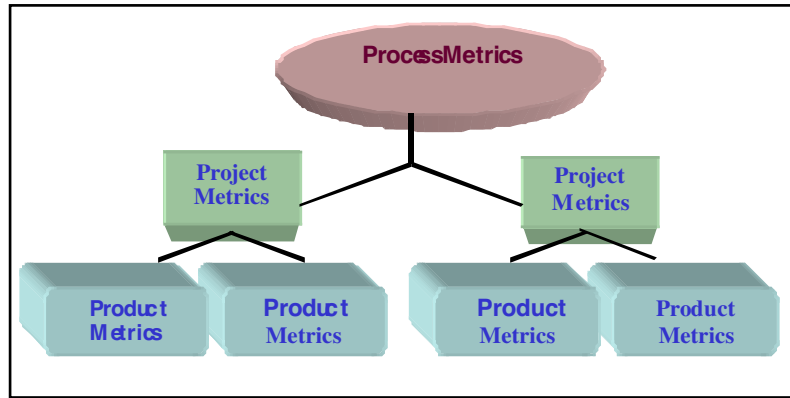


Figure 2: Process, Project and Product Metrics.

standards and frameworks is to provide the necessary reference to carry out the measurement process effectively and systematically, based on the premise that measurement must always be carried out in pursuit of a series of goals. The different concepts and terms involved in software process measurement can be found in “Software Measurement Ontology” [8].

According to the assessment and improvement models ISO 15504, CMM (Capability Maturity Model) and CMML, when increasing the maturity level of an organization it is necessary to establish a quantitative base which, in ascending order of maturity, should be focused on the process, the projects and the products (Figure 2).

Software processes form the base from which the work of an organization is carried out. In practice these processes are applied in the form of projects. Products are obtained as the result of the performance of specific projects. Therefore, in order to establish a measurement framework within an organization, the following three dimensions must be taken into account:

- **Process Measurement** is based on the study and control of the capacity of the processes and on the change management in these processes. Process metrics are extracted by measuring the characteristics of specific software engineering tasks in order to obtain metrics on faults detected before the delivery of the software, defects detected and reported by end users, human effort and time consumed, adjustments made

to the planning, etc. When measuring the process, indicators referring to the process model itself should also be established, as the complexity and quality of the model may affect the execution, and therefore the quality, of the end product. However, literature referring to studies performed on process metrics have up to now concentrated on the measurement of projects and products. In an attempt to make up for this omission, the following section describes how to measure a software process at a conceptual level, while presenting a set of software process model metrics, and highlighting the aspects of the process that can be assessed using these metrics.

- **Measurement of the Project** is based on project management, which has been a mature research field for several years now.
- **Product Measurement.** The main reason for measuring software products is to evaluate the quality of deliverables. Much has been written regarding this subject, including studies, proposals and metrics, some of which are as well known and established, such as source code lines, function points, or McCabe’s cyclomatic complexity.

### 3 Process Measurement at A Conceptual Level

Since the study of process assessment has focused on the gathering of data from the project in order to obtain measurements relating to performance, productivity, efficiency etc.,

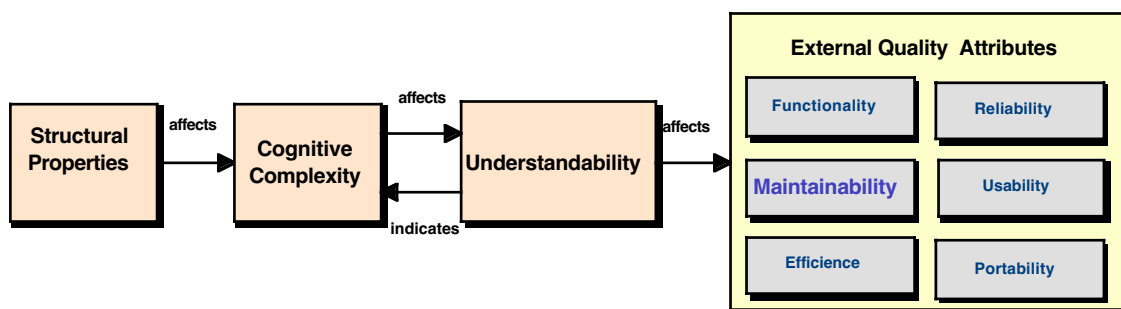


Figure 3: Structural Complexity Model [2].

specific metrics for process models have not been defined. The definition and validation of metrics for process models will enable us to study the influence of the structural complexity of these process models on their maintainability (ease of maintenance or change), bearing in mind that a highly complex process model will be more difficult to alter and therefore the ease with which it can be maintained will be significantly reduced. This will also influence process improvement which may, in turn, affect the projects (making them more costly in terms of resources and time) and the quality of the end products. Briand et al. [2] set out the principal theoretical basis for the development of quantitative models, which relates the attributes of structural complexity to those of the external quality of software artefacts, as shown in Figure 3.

As proposed in Figure 3, maintainability can be estimated through a set of metrics that measures the structural properties of the models in question (size, coupling, etc.). The maintenance of software process models involves modifying them with the aim of improving them, correcting any errors they may have, adapting them to new necessities, or improving some of their properties (such as quality). For example, it may be necessary to correct a process model in which there are activities that do not receive input or that do not generate output, or alternatively it may be necessary to improve a model by eliminating unnecessary dependencies among activities. Software process models that are difficult to maintain may have a negative effect on the execution of the projects and on the quality of the end products.

In conclusion, in order to evaluate the maintainability of software process models, it is necessary to: 1) define a set of metrics enabling us to evaluate structural complexity; 2) prove the usefulness of those metrics by carrying out empirical studies to ensure that they can be used as indicators of maintainability of the models. The following subsections present a set of metrics for process models and an example of how they are calculated.

### 3.1 Metrics for Software Process Models

The following metrics have been defined using SPEM terminology [14], but they can be applied directly with other modelling languages since practically the only differences are terminological. The conceptual model of SPEM is based on the idea that a software development process consists of a collaboration between abstract and active entities, known as “process roles”, that carry out operations called “activities” on tangible entities called “work products”. When process model metrics are established two levels of impact are considered:

- **Model Level.** These metrics are applied to measure the structural complexity of the process model as a whole. They are represented in Table 1.
- **Fundamental Model Element Level** (Activity, Process Role and Work Product). These metrics are described in other publications [6].

Figure 4 shows an example of a simplified process model belonging to the Rational Unified Process (RUP). SPEM does not have its own graphic notation, but as it is defined as a UML profile, UML diagrams (class, package, activity, use case, and

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

Table 1: Software Process Model Metrics.

sequence) can be used to represent the different views of the process. The stereotypes used by SPEM (the icons in the figure) must be added to the UML diagrams.

As can be seen in Figure 4, a software process view can be represented using UML activity diagrams. This view includes the different activities, their precedence relationships, the used or produced work products and the responsible roles. The values obtained from the metrics defined at process model level (presented in Table 1) can be consulted in Table 2.

In order to demonstrate the practical usefulness of a metric, some empirical validation by means of experiments is required. It has thus been possible to demonstrate the correlation between the metrics NA, NPT, NDPTin, NDPTout, NDPT, NDPA, and NCA and the maintainability of the software processes [7].

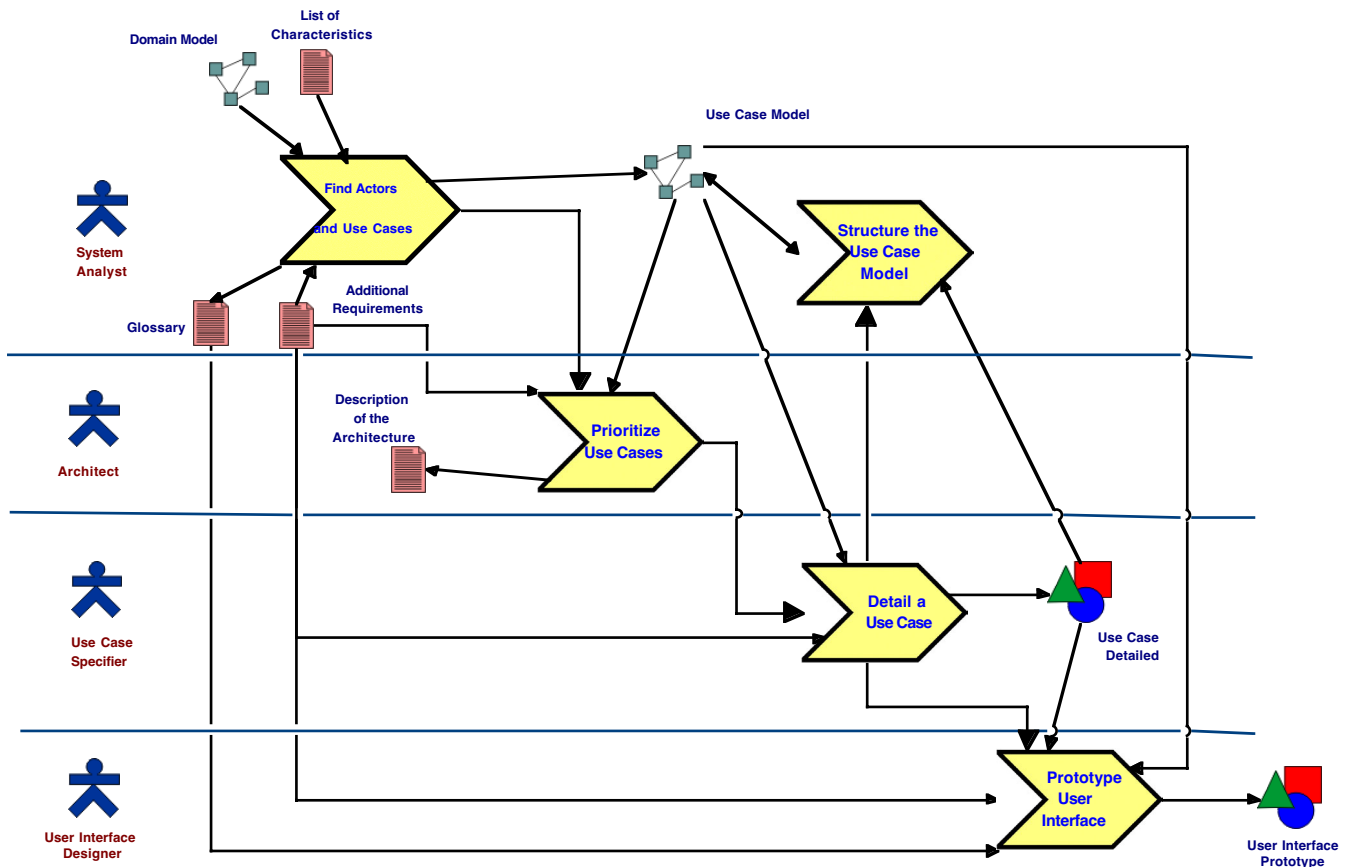


Figure 4: Example of A Software Process Model.

#### 4 Conclusions and Challenges for The Future

This article presents a general overview of the measurement of software processes, a basic view of the management of these processes which can be used to establish the quantitative base required for their improvement.

Traditionally, software process measurement has focused on the measurement of projects and products, but as a result of the increasing interest shown by software companies in the institutionalizing, modelling and improvement of their processes, software process models have become an important entity to be taken into consideration. In this article we present a representative set of metrics for the evaluation of the maintainability of software process models. These metrics are useful for predicting the maintainability of process models and provide useful indicators for companies implementing process improvement programmes. An important future line of research in this area is the development of empirical studies to establish relationships between the maintainability of process models and the

results obtained from their enactment in the form of software projects. Finally, by means of the integrated measurement of software process related entities, we can obtain the required quantitative basis from which the correlation existing between process and product can be objectively evaluated.

These software process modelling and measurement issues will be dealt with more efficiently in the years to come thanks to the convergence of software process technology with two recent technologies: “Workflows” and “Web Services”.

“Workflow Management Systems” [17] provide support to modelling, enactment and management of business processes. Therefore, as some authors have suggested [13] they can be a useful tool for software engineers to manage and implement their development and maintenance processes. In this regard, the new standards for representing processes by means of workflows are very interesting. The best example of these is the “Workflow Process Definition Interface – XML Process Definition Language (XPDL)” [18].

NA	NWP	NPR	NDWPIn	NDWPOut	NDWP	NDA	NCA	RDWPIn	RDWPOut	RWPA	RRPA
5	8	4	13	6	19	4	1,25	0,68	0,32	1,6	0,8

Table 2: Values of Metrics for The Example in Figure 4.

The issue of process modelling in the area of Web Services technology has also been the subject of some study. It would be true to say that the design of an application based on invoking a collection of web services is very similar to the design of the business process model that the application supports. As a result, the web services community has also developed standards and languages for process modelling: “Business Process Modelling Language” (BPML), “Business Process Specification Schema” (BPSS), “Business Process Execution Language for Web Services” (BPEL4WS), and “Web Service Choreography Interface” (WSCI) are some of the most important. In this regard, readers may find of interest the October 2003 issues of *Communications of ACM* (dedicated to services oriented computing) and *IEEE Computer* (dedicated to software as a service).

The convergence and integration of these technologies will provide new ways for the software engineers to perform their work, particularly regarding aspects related to process management and improvement. We should therefore expect software process models used for the development and maintenance of software to be designed and managed via a Workflow Management System, which in order to carry out certain automatic and semi automatic activities, will call on Web services that will act as both CASE tools (for example, a compilation service or unitary tests) and as support for management and organizational activities. All this foreseeable development in future years will mean that software engineers will not only have to pay special attention to what they produce (the product) but also to how they make it (the process). And, good engineers as they are, they will have to measure both the product and the process.

## References

- [1] V. Basili and D. Weiss. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*, 10, pp. 728–738, 1984.
- [2] L. Briand, S. Arisholm, F. Counsell, F. Houdek, and P. Thévenod-Fosse. Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions. *Empirical Software Engineering*, 4(4), pp. 387–404, 1999.
- [3] J.C. Derniame, B.A. Kaba, and D. Wastell. *Software Process: Principles, Methodology and Technology* (Vol. LNCS 1500). Springer-Verlag, 1999.
- [4] N. Fenton. Metrics for Software Process Improvement. In M. Haug, E. W. Olsen & L. Bergman (Eds.), *Software Process Improvement: Metrics, Measurement and Process Modelling* (pp. 34–55). Springer, 2001.
- [5] W. A. Florac and A. D. Carleton. *Measuring the Software Process. Statistical Process Control for Software Process Improvement*. Addison Wesley, 1999.
- [6] F. García, F. Ruiz, J. A. Cruz, and M. Piattini. Integrated Measurement for the Evaluation and Improvement of Software Processes. 9th European Workshop on Software Process Technology (EWSPT'9), Helsinki (Finland), pp. 94–111, 2003.
- [7] F. García, F. Ruiz, and M. Piattini. Definition and Empirical Validation of Metrics for Software Process Models. 5th International Conference Product Focused Software Process Improvement (PROFES'2004), Kansai Science City (Japan), pp. 146–158, 2004.
- [8] F. García, F. Ruiz, M. F. Bertoa, C. Calero, M. Genero, L.A. Olsina, M. A. Martín, C. Quer, N. Tondori, S. Abrahao, A. Vallecillo, and M. Piattini. Una Ontología de la Medición del Software. Technical Report, Depto. de Informática, Universidad de Castilla-La Mancha. Available, in Spanish, at <http://www.info-ab.uclm.es/trep.php?&codtrep=DIAB-04-02-2>, 2004.
- [9] IEEE. IEEE Std 1061-1992, “IEEE Standard for a Software Quality Metrics Methodology”, 1992.
- [10] ISO/IEC. ISO IEC 15504 TR2:1998, Software Process Assessment – Part 4: Guide to conducting assessment. International Organization for Standardization, 1998.
- [11] ISO/IEC. ISO 15939: Software Engineering – Software Measurement Process, 2002.
- [12] J. McGarry, D. Card, C. Jones, B. Layman, E. Clark, J. Dean, and F. Hall. *Practical Software Measurement. Objective Information for Decision Makers*. Addison-Wesley, 2002.
- [13] C. Ocampo, and P. Botella. Some Reflections on Applying Workflow Technology to Software Processes. Universitat Politècnica de Catalunya, Dep. de Llenguajes y Sistemes Informàtics, technical report TR-LSI-98-5-R, Barcelona, 1998.
- [14] OMG. *Software Process Engineering Metamodel Specification; adopted specification*. Object Management Group, 2002.
- [15] H.D. Rombach. Design measurement: some lessons learned. *IEEE Software*, 7(3), pp. 17–25, 1990.
- [16] SEI. *Capability Maturity Model Integration (CMMISM)*, version 1.1. Software Engineering Institute, 2002.
- [17] WfMC. TC00-1003 1.1: *The Workflow Reference Model*. Workflow Management Coalition, January-1995.
- [18] WfMC. TC-1025 final draft 1.0: *Workflow Process Definition Interface – XML Process Definition Language (XPDL)*. Workflow Management Coalition, October-2002.