The ICFAI University Press

PROJECTS & PROFITS

Special Issue

June 2005 www.icfaipress.org Rs. 60

Software Development Projects

CONTENTS

PROJECTS & PROFITS

SPECIAL ISSUE

ARTICLES

Offshore Software Project Management: A Hands-on Approach 06

Process Discipline for Project Improvements 09

A Stitch in Time: Timing in Software Project Management 17

Improving Estimations in Software Projects with Data Mining Techniques 25

Software Process: Characteristics and Measurement 39

An Introduction to Function Point Counting 51

Five Quality Myths in Software Project Management 55

Earned Value Management: A Simple Project Management Tool 61

Risk Management in Software Projects 65

Knowledge Management in Software Companies: An Experience Factory Approach 70

Identifying your Organization's Best Practices 83

FEATURES

Perspective 5

Research Summary 91

PROJECTS & PROFITS

Volume-V: 6 June 2005

EDITOR
E N Murthy

EXECUTIVE EDITOR
Ch Rajeshwer

CONSULTING EDITOR
Y Chandra Sekhar

EDITORIAL TEAM
A Anand, Arshia Anwer, P Ravindra Varma

PRODUCTION MANAGER
A L Subrahmanyam

PRODUCTION TEAM
N Lalitha Devi, Aparna Marthi

CHIEF VISUALIZER
Bangaru Babu A

VISUALIZERS
S Ganesh, PRV Prasad

GRAPHIC DESIGNERS
Md. Nazeer Hussain, KPPS Prasad

ILLUSTRATOR
Anjaneyulu B

ADVERTISEMENT ENQUIRIES
Business Manager, The ICFAI University Press, # 23, Nagarjuna Hills, Panjagutta, Hyderabad - 500 082.
Tel: +91(40) 23430-431 to 436
Fax: +91(40) 5563-9711

SUBSCRIPTION DETAILS		
	By Post	By Courier
1 year	Rs. 625	Rs. 750
3 years	Rs. 1650	Rs. 2000
5 years	Rs. 2000	Rs. 2500

Payment to be made by crossed Demand Draft drawn in favor of The ICFAI University Press, Hyderabad.
Visa & MasterCard holders can subscribe online or fax their subscription order indicating their credit card number, name, amount, expiry date, duly signed.
For subscriptions and related enquiries, write to:
Executive (Subscriptions), The ICFAI University Press, # 52, Nagarjuna Hills, Panjagutta, Hyderabad - 500 082.
Tel: +91(40) 23435-368 to 374, Fax: +91(40) 2335-2521
E-Mail: serv@icfaipress.org
Also visit www.icfaipress.org

ISSN 0972-5334

Printed and published by E N Murthy on behalf of The Institute of Chartered Financial Analysts of India, The ICFAI University, # 52, Nagarjuna Hills, Panjagutta, Hyderabad-500 082, Andhra Pradesh. Printed at M/s ICIT Software Center Pvt. Ltd., # 1, Technocrat Industrial Estate, Balanagar 'X' Roads, Hyderabad-500037, Andhra Pradesh and published from The Institute of Chartered Financial Analysts of India, The ICFAI University, # 52, Nagarjuna Hills, Panjagutta, Hyderabad-500 082, Andhra Pradesh.

Editor: E N Murthy

How to Reach Us

Send your feedback to

The Editor, The ICFAI University Press,
6-3-354/1, Stellar Sphinx, Road No. 1, Banjara Hills, Panjagutta, Hyderabad-500 082.
Tel: +91(40) 23430-448, 449, 450, 451
Fax: +91(40) 23430-447
e-mail to info@icfaipress.org, Website: www.icfaipress. org

Advertising

For information and tariff:
E-mail: mkt@icfaipress.org, or Tel: +91(40) 23430-431 to 436
Fax: +91(40) 5563-9711

Articles

Articles may be sent to
info@icfaipress.org

For more information, please visit
www.icfaipress.org

Subscriptions

Please contact your nearest ICFAI Center for details:

Agra (Ph: 0562-2527035); **Ahmedabad** (Ph: 079-6563042, 6562458); **Ajmer** (Ph: 0145-2622707); **Allahabad** (Ph: 0532-2420255); **Anantapur** (Ph: 08554-249334); **Aurangabad** (Ph: 0240-5624774 / 775); **Bangalore** (Ph: 080-2899363 / 262); **Bhopal** (Ph: 0755-5277253, 2576975); **Bhubaneswar** (Ph: 0674-2506203/204); **Chandigarh** (Ph: 0172-381128 749504); **Chennai** (Ph: 044-28235633, 28235688); **Coimbatore** (Ph: 0422-2541190, 5366447); **Dehradun** (Ph: 0135-2654002 / 03); **Guntur** (Ph: 0863-2238958); **Gurgaon** (Ph: 0124-222-3595, 5556); **Gwallor** (Ph: 0751-2322273); **Hubli** (Ph: 0836-2371738); **Hyderabad** (Ph: 040-55663661, 55681180/81); **Indore** (Ph: 0731-5069003, 5068247); **Jabalpur** (Ph: 0761-5066886); **Jaipur** (Ph: 0141-2363695, 2373689); **Jalandhar** (Ph: 0181-5074769); **Jamshedpur** (Ph: 0657-2434957); **Kakinada** (Ph: 0884-2387772); **Kanpur** (Ph: 0512-233-0912 / 1145); **Kochi** (Ph: 0484-2369763, 2382294); **Kolhapur** (Ph: 0231-2655142, 2654491); **Kolkata** (Ph: 033-22873161/ 22817802); **Kurnool** (Ph: 08518-249811); **Lucknow** (Ph: 0522-220-4205/ 4559); **Ludhiana** (Ph: 0161-2772523); **Madurai** (Ph: 0452-2342169); **Mangalore** (Ph: 0824-2432050); **Mumbai** (Ph: 022-22040888, 22823173); **Mysore** (Ph: 0821-543803); **Nagpur** (Ph: 0712-2564314, 2547124); **Nasik** (Ph: 0253-2570413); **Nellore** (Ph: 0861-2301222); **New Delhi** (Ph: 011-23739157, 23739169); **Noida** (Ph: 0120-2592410, 2516024); **Patna** (Ph: 0612-2237942); **Pune** (Ph: 020-4026975, 4026976); **Raipur** (Ph:0771-5061361); **Rajahmundry** (Ph: 0883-2448813); **Ranchi** (Ph: 0651-2306922); **Thane** (Ph: 022-25375836, 25382659); **Trivandrum** (Ph: 0471-2320853); **Udaipur** (Ph: 0294-5102254); **Vadodara** (Ph: 0265-2341780); **Vijayawada** (Ph: 0866-2473620, 5563620); **Visakhapatnam** (Ph: 0891-2752653, 2598650); **Warangal** (Ph: 0870-2552610)

PROJECTS & PROFITS

Software Development Projects

The number of software development projects has drastically increased during the last decade. However, their success rate is very low as the majority of projects experience budget overruns, are completed only behind schedule, miss a critical function or a combination. According to the Standish Group Reports, about 80% of software development projects are unsuccessful, with about 30% of them poorly executed, which results in termination of the project midway. Although the success rate of software development projects is slowly improving, the rate of improvement is still slow, partially because software projects are getting larger and more complex.

Software projects differ from hardware projects, like infrastructure projects, in three ways. First, it is very difficult to identify a deviation from the plan in case of a software project, and most often, the deviation is identified only during the later stages of the development cycle. Second, the progress achieved by the project is hard to measure since most of the development or progress happens virtually in the minds of the team members. Lastly, testing the product and the software is not easy. It largely depends on the interpretation of the developer who designs and codes the software. Therefore, the testing of the product and integration is largely the function of requirements interpretation by the developer.

Undoubtedly, there is a significant change in the way software development projects were managed earlier and how they are managed today. Identifying the changes that arise as a result of technology-driven factors or environment-driven factors, and then making structural adjustments will lead to successful completion of projects. Some of the best practices that were adopted for software development projects that can be identified as a result of the changes in the recent past include choosing the appropriate development life cycle process, gathering and agreeing on requirements, choosing an appropriate architecture and design strategy, construction of the code with functionalities like continuous integration and self-testing code, conducting peer reviews, planning for testing, using simulation to test performance, effective configuration management, establishing quality and defects management, providing system operations and support after deployment, and application of project management tools and techniques.

Though the best practices for one project may not work in a similar way for another development project, adopting some of these practices will definitely increase the success rate and the chance of completing the software project successfully will be higher. This *Special Issue* of *Projects & Profits* is an attempt to look at the projects from the perspective of practitioners and provide insights into better ways of managing the software development projects.

Y Chandra Sekhar

Software Process

Characteristics and Measurement

Francisco Ruiz, Félix García and Mario Piattini

In this article we present the concept of Software Process (SP) and the properties that characterize and distinguish these processes from other process types and Software Process Technology (SPT), that allow us to automate and to integrate the production and management processes in order to carry out software projects. Software process measurement issues play a vital role in providing the necessary quantitative base for the identification of aspects on which to focus improvement programs. The relevant entities that can be measured in relation to the process are also identified and an example is described of how to measure one of these entity types: Process models.

Characteristics of the Software Processes

The definition of SP supplements the concept of a life cycle in the sense that it defines the skeleton and philosophy of carrying out an SP. But it is not sufficient to guide and control a development and/or maintenance project. An SP is a "coherent collection of policies, organizational structures, technologies, procedures and artifacts that are necessary to conceive, develop, install and maintain a software product" (Fuggetta, 2000).

The special nature of the SP is determined by the following characteristics:

a. They are complex.

b. They are not typical processes of production; since they are directed by exceptions, they are very affected by unpredictable circumstances, and each has peculiarities that distinguishes it from the others.

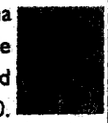
c. They are not processes of "pure" engineering either, since the appropriated abstractions are not known. (There isn't an experimental science to back them up), they depend a lot on too many people, the design and production are not clearly defined, and the budgets, schedules and quality can not be programmed in a sufficiently reliable way.

d. They are not (completely) creative processes because some parts can be described in detail, and some procedures are established previously.

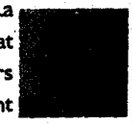
e. They are based on discoveries which depend on communication, coordination and cooperation within a pre-defined framework; deliverables generate new requirements; the costs of changing the software are not normally known; furthermore, its success depends on the participation of the user and the coordination of many roles (sales, technical development, client, etc.).

About the Authors

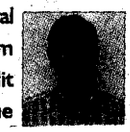
Francisco Ruiz, is Ph.D. in Computer Science for the University of Castilla-La Mancha (UCLM) and MSc in Chemistry-Physics for the Complutense University of Madrid (Spain). He is full time Associate Professor of the Department of Computer Science at UCLM in Ciudad Real (Spain). He has been Dean of the Faculty of Computer Science between 1993 and 2000. Previously, he was Computer Services Director's in the mentioned university (1985-1989) and he has also worked in private companies like analyst-programmer and project manager. His current research interests include: Business process modeling, management and measurement, software process technology and modeling, software maintenance, and methodologies to software projects planning and managing.



Félix García is MSc and Ph.D. in Computer Science by the University of Castilla-La Mancha (UCLM). He is Assisstant Professor at the Department of Computer Science at UCLM and Member of Alarcos Research Group. Author of several papers and book chapters on software processes managament, from the point of view of their modeling, measurement and technology. His research interests are: Business process measurement, software processes and software measurement.

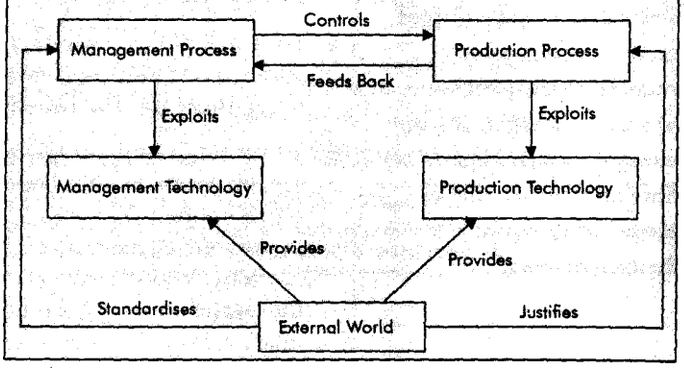


Mario Piattini Alarcos is a MSc and Ph.D. in Computer Science from the Politechnical University of Madrid. MSc in Psychology from the UNED. Certified Information System Auditor and Certified Information Security Manager from ISACA (Information System Audit and Control Association). Full Professor at the Department of Computer Science at the University of Castilla-La Mancha, in Ciudad Real, Spain. Author of several books and papers on databases, software engineering and information systems. He leads the ALARCOS research group specialized in information system quality. His research interests are: Software quality, advanced database design, metrics, software maintenance, information system audit and security.



The necessity for human participation in a creative way and the absence of repetitive actions makes, neither the software development nor the maintenance a production process. However, there are some similarities between both types of processes which are useful for the understanding of the software processes in a wider perspective. Like in the production process, the software processes consist of two inter-related sub-processes; the production process and the management process (McLeod,1990). The production process strictly relates to the production and maintenance of the product, while the management process provides the necessary resources for the production process, and controls it. This control is possible when the production process feeds back information of its situation to the management process. These relationships are represented in Figure 1. Also, the relationships between the processes and the external environment are shown: The request for a product comes from outside (the external world). In other words, the external environment is why the

Figure 1: Production Process vs. Management Process

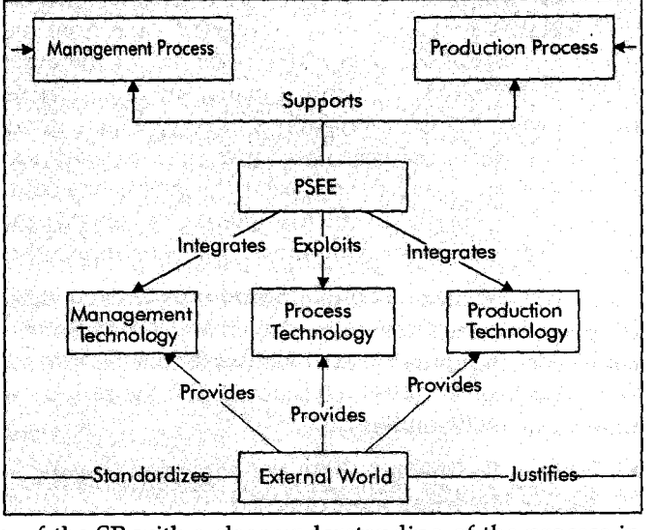


production process exists. Furthermore, the management has to meet the current standards of the external environment: Again, the external environment influences the process of production, indirectly. In summary, the production and management processes exploit technologies that come from the external environment.

Software Process Technology

The essence of SPT is that it allows the integration of production and management technologies in a new work environment, known as "Process-centered Software Engineering Environment" (PSEE), that supports the management and production processes in an integrated way. Figure 2 shows the impact of this new technology and how the PSEE implements, controls and improves the flow of information in which the management process controls the production process. The main objective of the SPT is to control the inherent complexity of the SP with a clear understanding of the process in itself, and by an automated support, by means of a PSEE. A fundamental aspect to achieve this objective is the computerized process support, which is the presence of a model of processes and the right means of defining, modifying, analyzing and enacting it (Derniame *et. al.*, 1999).

Figure 2: Impact of Software Process Technology



As seen from the previous definition, SPT benefits from a wide range of areas and concepts:

1. **Technologies** of software development and maintenance which provide the necessary tools and infrastructures to make it possible and economically feasible, to create and maintain complex software products that satisfy present and future needs.
2. **Methods and Techniques** in the software development and maintenance which entails the essential methodological support to benefit wisely from the technologies and carry out with success the development and maintenance activities.
3. **Organizational Behavior,** this is to say, the science of the organizations and people is useful because, in general, the software projects are carried out in groups that have to be coordinated and directed inside an effective organizational structure.
4. **Marketing and Economy,** since the efforts of the projects of software development and maintenance are not autonomous, like any other product, the software must be directed to satisfy the needs of real clients and users.

In consequence, in the development and maintenance of software, it is necessary to pay attention to the complex relationships that arise between the various organizational, cultural, technological and economic factors.

A. Process Orientation

The importance which process-oriented SEE (or PSEE) has on the SPT, has already been stated. In fact, the principal role of an SEE is to provide support to effectively carry out the SP. This point of view is gaining weight, because the software development and maintenance processes have converted each time more, in complex and laborious intellectual activities, with a high potential for the improvement in the quality and the productivity, based on discipline, management, and the help of PSEE and other computing technologies. Many organizations have problems defining and carrying out the steps, which transform the user necessities in a software product, in a way that should be repeatable, measurable in respect to the quality objectives, and adaptable or improvable. Therefore, the help of an SEE to implement a defined process during the performance of a software project can provide substantial benefits in a short time.

In an PSEE, the process management services contribute to the effective support of the SP, providing facilities oriented to the end user in order to define and use processes that can replace the undisciplined, difficult to control, and tedious invocation of isolated tools. Garg and Jazayeri (1996) considered that the process support in PSEE is based on the following functionalities:

- **Process Definition:** Software engineers use the PSEE to define a process to be followed by one or more projects.
- **Process Analysis:** A process model within a PSEE can be analyzed to verify its consistency, completeness, and correctness.
- **Process Presentation:** A PSEE includes support for the graphical display of the SP (activity flows) and the products (structured diagrams).
- **Process Simulation:** The PSEE supports the use of simulations to be able to evaluate the suitability of a process before committing full resources to it.
- **Process Automation:** Once a process has been defined, activities which do not require human intervention can be identified and automated by the PSEE.
- **Process Monitoring:** The PSEE monitors the execution of a process and records the history of the activities carried out. This process history can be used later for future process developments and improvements.
- **Process Change Support:** The PSEE allows an organization change its process definitions without interrupting the work.
- **Openness:** The PSEE provides tools to interchange data and metadata with other non-integrated tools, or with other PSEE.
- **Multi-User Support:** Typically, software engineering projects are realized by teams of people with different roles, therefore, the PSEE must provide services to all the people that work together in a process.

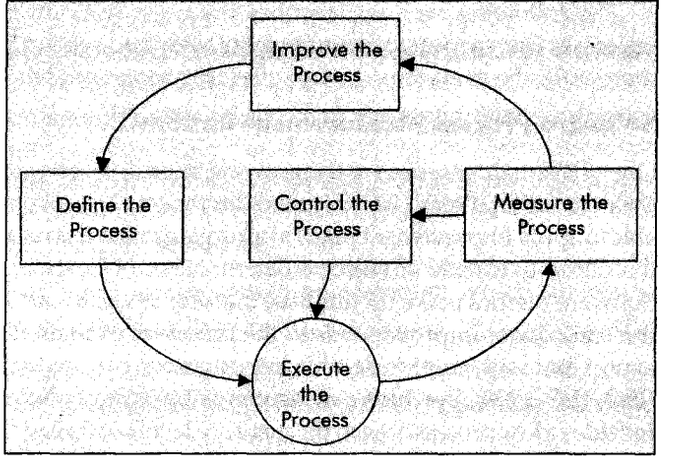
- **Process Guidance:** Software engineers use the PSEE to carry out various process steps. The PSEE must offer help to choose the next steps in the light of the process model and the current status.
- **Task-specific User Interface:** Based on the modelled process, the PSEE can adapt the user interface to the necessities of each task and, in this way, avoid an excess of information being presented to the user.

It is becoming more frequent, that the software product development and maintenance be carried out with the collaboration of various companies or organizations. As a result, in the last years there has been a great interest in the study of the problems that arise when various separate and different PSEE are asked to collaborate, and more concretely, that there be inter-operability between the processes that they support. Amongst the various proposals made to solve this problem, those of the "PSEE federation" stand out. In this line, some authors have proposed the international alliance metaphor where every organization manages its own processes (like every country has its own laws), and the inter-organizational processes should act in a similar way to the international treaties between countries. In the bibliography, two types of conceptual architectures have been proposed for PSEE federations: Control-based, that favors centralization since models of common processes exist; and state-based, which has a space where common state is stored (Estublier *et. al.*, 1998).

Software Process Management

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 is due to the fact that the quality of the process is closely related to the quality of the product, meaning that in order to obtain better products it is necessary to avail of better processes. In order to satisfy the quality requirements of software processes they must produce the expected results, be correctly defined and any improvements made should be in accordance with the objectives of the enterprise, which are often very changeable in highly competitive companies. Those are the objectives of the "Software Process Management" (Florae and Carleton, 1999) which in order to be applied effectively involves four key responsibilities: To define, measure, control and improve the process. These responsibilities and their relations are represented in Figure 3.

Figure 3: Software Process Management Key Responsibilities



In relation to these responsibilities (which are basic for the successful management of software processes) in order to efficiently carry out the improvement of the process it is necessary to bear in mind the following aspects:

- **Definition of the Process:** This is the first key responsibility that must be assumed in order to carry out an effective management which is orientated towards process improvement. To do this it is necessary to model the processes, that is to say, to represent the elements of interest involved in them. Given the diversity of elements that must be taken into account when speaking about a software process, the definition of software processes is not a simple task. In the literature on the subject different modeling languages and formalisms can be found, known as "Process Modeling Languages" (PML), whose objective is to represent the different related elements in a precise and non ambiguous way. Usually, in a software process it is possible to identify the following elements or general concepts (although using different notations and terms) in the different PMLs (Derniame *et. al.*, 1999): Activity, Product, Resource, Organization and Role. With the aim of providing a common reference for the representation of software processes, the OMG consortium (Object Management Group) has been working for several years on the development of the metamodel SPEM (Software Process Engineering Metamodel) (OMG, 2002), which is a generic language that extends UML (Unified Modeling Language) for the descriptive modeling of software processes without including aspects related to their enactment. Currently SPEM constitutes a specification from which in the future a process modeling standard is expected to be derived which could equal the importance and acceptance by the industry of the present day UML standard.
- **Process Execution and Control:** The software projects of a company are carried out in accordance with defined process models. It is important to be able to be permanently in control of the realization of these projects (and consequently the corresponding processes) in order to ensure that the desired results are achieved.
- **Measurement and Improvement:** There is a significant correlation between the measurement and improvement of software processes. Prior to improving a process, it is necessary to carry out an assessment process aimed at identifying the aspects of the process that could be improved. In order to do this it is useful to have an effective framework at one's disposal which facilitates the identification of the candidate relevant entities to be measured. The results of the measurement of the processes provide non-subjective information enabling the necessary actions for improvement to be planned, identified and carried out in an efficient fashion.

The following section describes the most relevant entities related to the software processes from a measurement point of view and section 5 presents a representative set of metrics for the evaluation of the software process models' structural complexity.

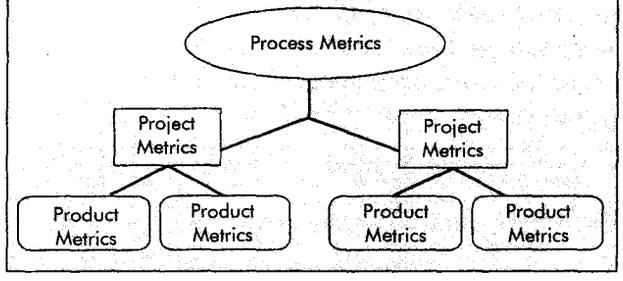
Software Process Measurement Entities

One of the main reasons for the growing interest in software metrics has been the increasing awareness that metrics are necessary for process improvement (Fenton, 2001). Before being able to apply improvement plans in an organization it is necessary to have a quantitative base that allows us to make an objective determination of the strong and weak points of the processes. Software metrics make up this base that enables us to carry out the assessment process and the consequent improvements to the processes evaluated. As a result, measurement is an aspect that is given considerable importance in assessment models such as ISO/IEC 15504 (ISO/IEC, 1998), in which a measurement process is defined, or CMMI (SEI, 2002), which includes a key process area in the maturity level two called "Measurement and Analysis". As

far as support for the measurement process is concerned, several work frameworks are worthy of note such as GQM (Goal Question Metric) (Basili and Weiss, 1984; Rombach, 1990) or PSM (Practical Software Measurement) (McGarry *et. al.*, 2002), in addition to standards of which ISO 15939 (ISO/IEC, 2002) and IEEE Std 1061-1992 (IEEE, 1992) should be highlighted. The objective of these norms and frameworks is to provide the reference necessary to effectively and systematically carry out the measurement process, based on the premise that measurement must always be carried out in accordance with a series of goals. The different concepts and terms involved in software process measurement can be found in the "Software Measurement Onthology" (Ruiz *et. al.*, 2003).

According to the assessment and improvement models ISO 15504, CMM and CMMI when increasing the maturity level of an organization it is necessary to establish a quantitative base which from a lower to a higher degree of maturity, should be focused on the process, the projects and the products (Figure 4).

Figure 4: Process, Project and Product Metrics



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

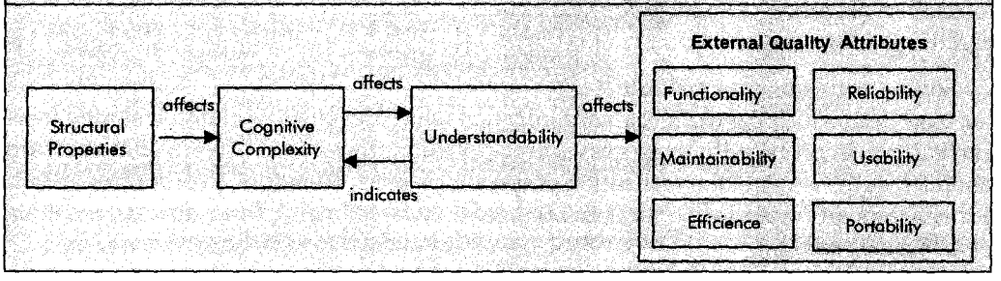
- **Process Measurement,** based on the study and control of the capacity of the processes and on the change management in these processes. Process metrics are extracted by measuring the characteristics of tasks that are specific to software engineering and by obtaining as results metrics related to faults detected before the software is delivered as well as defects detected and reported by the final users, human effort and time consumed, adjustments made to the planning, etc.. When measuring the process it is also recommendable to establish indicators related to the process model itself, as the complexity and other quality characteristics of the model could affect its realization and consequently the quality of its products. Despite this, studies looking at process measurement in literature on the subject have focused on the measurement of projects and products. In an attempt to compensate this shortcoming the following section describes how to carry out the measurement of a software process at conceptual level, presents a set of software process model metrics and highlights the aspects of the process that can be evaluated using these metrics.
- **Measurement of the Project** based on the project management which is already a mature research field.
- **Product Measurement:** The objective of the measurement of software products is to evaluate the quality of the deliverables. A lot has been written regarding this subject including studies, proposals and metrics, some of which are well-known and

established such as source lines of code, function points, or McCabe's cyclomatic complexity.

Process Measurement at a Conceptual Level

Due to the fact that 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., specific metrics for process models have not been defined. The definition and validation of metrics for process models can 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 improvement of the process which as a result may affect the projects (making them more costly in terms of resources and time) and the quality of the end products. *Briand et. al.* (1999) lay down the main theoretical basis for the development of quantitative models that relate the attributes of structural complexity to those of external quality of software artefacts. Figure 5 illustrates this in a graphical way.

Figure 5: Model of Structural Complexity (Briand *et. al.*, 1999)



As proposed in Figure 5, maintainability can be estimated through a set of metrics that measures the structural properties of the models in question (size, coupling, etc.). Maintenance of software process models involves modifying them with the aim of improving them, correcting errors that they may have, adapting them to new necessities, or improving some of their properties (such as quality). For example, it may be necessary to carry out a correction in a process model in which there are activities that do not receive input or that do not generate output, or alternatively it may be a case of improving a model by eliminating unnecessary dependencies between activities. Software process models that are difficult to maintain may have a negative effect on the realization 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 that enables us to evaluate structural complexity; 2) Prove the usefulness of these 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 their calculation.

A. Metrics for Software Process Models

The following metrics have been defined using SPEM terminology (OMG, 2002), but they can be applied directly with other modeling languages as practically the only differences are those of terminology. The conceptual model of SPEM is based on the idea that a software

development process consists of collaboration between abstract and active entities, called "process roles", that carry out operations called "activities" on tangible entities called "work products". When the 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 (Garcia *et. al.*, 2003).

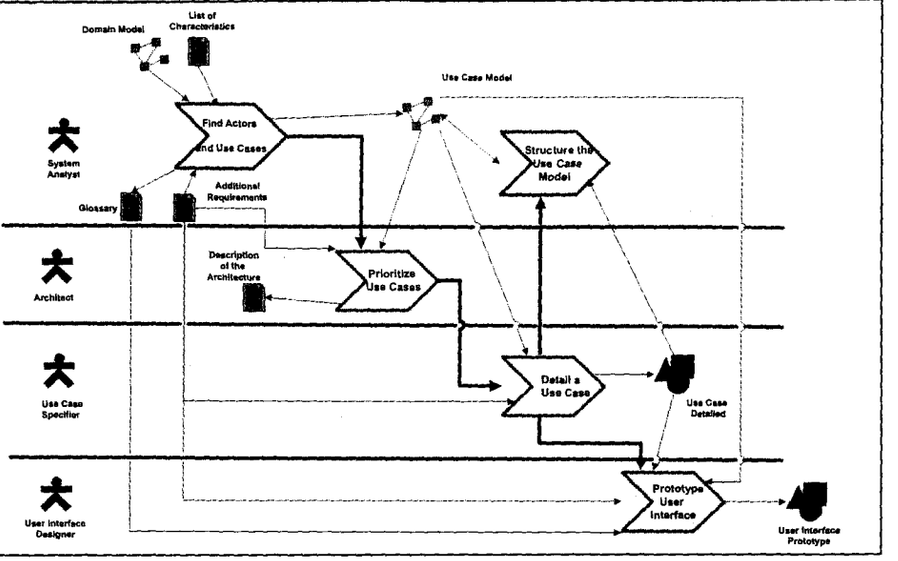
Figure 6 shows an example of a simplified process model belonging to the Rational Unified Process (RUP). SPEM does not have its own graphic notation, but as it is defined

Table 1: Software Process 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 $RRPA(PM) = \frac{NPR(PM))}{NA(PM)}$

as a UML profile , UML diagrams (class, package, activity, use case, and 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 referred UML diagrams.

Figure 6: Example of a Software Process Model



As can be seen in Figure 6 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, empirical validation must be carried out by means of experiments. By so doing it has been possible to demonstrate the correlation between the metrics NA, NPT, NDPTin, NDPTout, NDPT, NDPA, and NCA and the maintainability of the software processes (Garcia *et. al.*, 2004).

Table 2: Values of Metrics for Example of Figure 6

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

Conclusions and New Challenges and Opportunities

In this article we have summarized the key aspects of Software Process Technology: The object of interest (software process and their characteristics); the utility and justification of giving automatic support to these processes; and the requirements and functionality of process-orientation that should have the tool collections to achieve these goals. This article also presents a general overview of the measurement of software processes, a basic aspect of the management of these processes and one which is used to establish the quantitative base needed for their improvement.

Traditionally the 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, modeling and improvement of their processes, software process models have become an important entity to be taken into consideration. Furthermore, 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 ease of maintenance of the process models and represent useful indicators for enterprises that are carrying out process improvement programs.

These issues of software process modeling and measurement will be dealt with more effectiveness in next years thanks to the convergence of software process technology with two recent technologies: "Workflows" and "Web Services". The "Workflow Management Systems" (WfMC, 1995) provide support to modeling, enactment and management of business processes. Therefore as some authors have suggested (Ocampo and Botella, 1998) they can be a useful tool for software engineers to manage and carry out their development and maintenance processes. In relation to this the new standards for representing processes by means of workflows are important. The main example of these is the "Workflow Process Definition Interface — XML Process Definition Language (XPDL) " (WfMC, 2002).

In the area of Web Services technology the problem of process modeling has also been studied. 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 model of the business process that the application supports. As a result, the community dedicated to Web services has also developed standards and languages for process modeling: "Business Process Modeling 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 relation to this the reader may find interesting the October 2003 issues of Communications of ACM (dedicated to Services oriented Computing) and of IEEE Computer (dedicated to Software as a Service).

The convergence and integration of these technologies will provide new ways for the software engineer to carry out his work particularly regarding aspects related to management and improvement of processes. Thus it is to be expected that software process models used for the development and maintenance of software will be designed and managed using a Workflow Management System, which in order to carry out certain automatic and semi automatic activities, will call on Web services which will act as both CASE tools (for example, a compilation service or unitary tests) and as support for management and organizational activities. All this forseeable development in future days means that software engineers will not only have to pay special attention to what they make (the product) but also to how they make it (the process). Moreover, as good engineers they will have to measure both, the product and the process. ❖

References

1. Basili, V and Weiss, D (1984). "A Methodology for Collecting Valid Software Engineering Data". *IEEE Transactions on Software Engineering, 10*, pp. 728-738.
2. Briand, L, Arisholm, S, Counsell, F, Houdek, F and Thevenod-Fosse, P (1999). "Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions". *Empirical Software Engineering, 4(4)*, pp. 387-404.
3. Derniame, JC, Kaba, BA, and Wastell, D [editors] (1999): *Software Process: Principles, Methodology and Technology*. LNCS 1500, Springer-Verlag.

4. Estublier, J, Cunin, PY and Belkhatir, N (1998): Architectures for Process Support System Interoperability. Proceedings of the Fifth *International Conference on the Software Process* (ICSP'98), June 15-17, Chicago (USA), pp. 137-147.
5. Fenton, N (2001). "Metrics for Software Process Improvement". In M Haug, EW Olsen and L Bergman (Eds.), *Software Process Improvement: Metrics, Measurement and Process Modeling* (pp. 34-55). Springer.
6. Florae, WA.and Carleton, AD (1999). *Measuring the Software Process. Statistical Process Control for Software Process Improvement.* Addison Wesley.
7. Fuggetta, A (2000): "Software Process: A Roadmap". 22nd *International Conference on Software Engineering* (ICSE'2000), Future of Software Engineering Track, June 4-11, Limerick (Ireland). ACM.

Process Improvement (PROFES'2004), Kansai Science City (Japan), pp. 146-158.
10. Garg, PK and Jazayeri, M (1996): "Process-centered Software Engineering Environments: A Grand Tour". In Fuggetta, A and Wolf, A (editors); *Software Process.* John Wiley & Sons.
11. IEEE. (1992). *IEEE Std 1061-1992, "IEEE Standard for a Software Quality Metrics Methodology".*
12. ISO/IEC. (1998). *ISO IEC 15504 TR2:1998, Software Process Assessment - Part 4: Guide to conducting assessment.* International Organization for Standardization.
13. ISO/EEC. (2002). *ISO 15939: Software Engineering – Software Measurement Process.*
14. McGarry, J, Card, D, Jones, C, Layman, B, Clark, E, Dean, J and Hall, F (2002). *Practical Software Measurement. Objective Information for Decision Makers.* Addison-Wesley.
15. McLeod, R jr (1990): *Management Information Systems.* McMillan Publishing, New York.
16. Ocampo, C and Botella, P (1998): *Some Reflections on Applying Workflow Technology to Software Processes.* Universitat Politecnica de Catalunya, Dep. de Lenguajes y Sistemas Informaticos, technical report TR-LSI-98-5-R, Barcelona (Spain).
17. OMG. (2002). *Software Process Engineering Metamodel Specification; adopted specification.* Object Management Group.
18. Rombach, HD (1990). "Design Measurement: Some Lessons Learned". *IEEE Software, 7(3),* pp. 17-25.
19. Ruiz, F, Genero, M, Garcia, F, Piattini, M and Calero, C (2003): "Proposal of a Software Measurement Ontology". *Argentine Symposium of Software Engineering* (ASSE'2003). September 1-3 2003, Buenos Aires (Argentina).
20. SEI. (2002). *Capability Madurity Model Integration (CMMf^M), version 1.1.* Software Engineering Institute.
21. WfMC (1995): TCOO-1003 1.1: *The Workflow Reference Model.* Workflow Management Coalition, January-1995.
22. WfMC (2002): TC-1025 final draft 1.0: Workflow Process Definition Interface – XML Process Definition Language (XPDL). Workflow Management Coalition, October-2002.

Reference # 17M-2005-06-05-01