ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/221595006

Software Process Modelling

Conference Paper · January 2001

Source: DBLP

CITATIONS
48

READS
1,808

2 authors, including:

Silvia Teresita Acuña
Universidad Autónoma de Madrid
90 PUBLICATIONS **730** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Tutorías y mediación hipermedia View project

Open Source Software Development Process View project

All content following this page was uploaded by Silvia Teresita Acuña on 05 June 2014.

The user has requested enhancement of the downloaded file.

Software Process Modelling

Silvia T. ACUÑA
Departamento de Informática, Universidad Nacional de Santiago del Estero
Avenida Belgrano (S) 1912, (4200) Santiago del Estero, Argentina,
E-mail: silvac@unse.edu.ar

and

Xavier FERRÉ
Facultad de Informática, Universidad Politécnica de Madrid
Campus de Montegancedo, (28660) Boadilla del Monte, Madrid, Spain
E-mail: xavier@fi.upm.es

ABSTRACT

In this paper the most relevant works related to software process are reviewed. Software process modelling tries to capture the main characteristics of the set of activities performed to obtain a software product, and a variety of models have been created for this purpose. A process model can be used to define a recommended software process. This paper deals with the prescriptive software process modelling and the most relevant results of this approach is presented.

Keywords: Software Process, Modelling, Prescriptive Models, Modelling Approaches, Review.

1. INTRODUCTION

The software process is a critical factor for delivering quality software systems, as it aims to manage and transform the user need into a software product that meets this need. In this context, software process means the set of activities required to produce a software system, executed by a group of people organised according to a given organisational structure and counting on the support of techno-conceptual tools. General concepts about software process will be presented in this paper and some specific approaches will be highlighted.

Software process modelling describes the creation of software development process models. A software process model is an abstract representation of the architecture, design or definition of the software process [25]. Each representation describes, at different detail levels, an organisation of the elements of a finished, ongoing or proposed process and it provides a definition of the process to be used for evaluation and improvement. A process model can be analysed, validated and simulated, if executable. The process models are used mainly for software process control (evaluation and improvement) in an organisation, but they can be also used for experimenting with software process theory and to ease process automation.

This aspect of the software development process that are going to be addressed in this paper (sections 3 and 4 deal with software process modelling). Firstly, some basic concepts related to this subject are presented in section 2.

2. SOFTWARE PROCESS BASICS

2.1. What is the Software Process?

Originally, the term life cycle was employed in every software development project [19]. Even though the notion of software process was present in these projects, the concept of software process was not clearly identified [12][47]. As software and software construction were further investigated, the software process acquired an identity in its own right and has been the subject of research and investigation in recent years [26][33]. The view taken of this concept is as follows [1].

Software process is a partially ordered set of activities undertaken to manage, develop and maintain software systems, that is, the software process centres on the construction process rather than on the product(s) output. The definition of a software process usually specifies the actors executing the activities, their roles and the artefacts produced. An organisation can define its own way to produce software. However, some activities are common to all software processes. A *software process model* is an abstract representation of the software process. The two key international standards that prescribe processes for developing and maintaining software are IEEE 1974-1991 [36] and ISO/IEC 12207 [37]. Both standards determine a set of essential, albeit unordered activities, which have to be completed to obtain a software product. They do not prescribe a specific life cycle. Each organisation that uses the standard must instantiate the prescribed process as a specific process. ISO/IEC 12207 presents a process of adaptation that defines the activities required to tailor the standard to an individual software project. A variety of software process models have been designed to structure, describe and prescribe the software system construction process [38][10][50][36][42] [20][48][41][6][26][37][28][9][1]. These models will be dealt with in more detail in section 4, which addresses software process definition.

2.2. Software Process Modelling

The *modelling of the software process* refers to the definition of the processes as models, plus any optional automated support available for modelling and for executing the models during the software process. Finkelstein et al. [26] define a process model as the description of a process expressed in a suitable process modelling language. There are other possible uses of software process models that will not be considered, such as the

introduction of a new process in an organisation and personnel training/motivation.

Curtis et al. present some of the specific goals and benefits of modelling the software process [18]:

1. *Ease of understanding and communication*: requiring a process model containing enough information for its representation. It formalises the process, thus providing a basis for training.
2. *Process management support and control*: requiring a project-specific software process and monitoring, management and co-ordination.
3. *Provision for automated orientations for process performance*: requiring an effective software development environment, providing user orientations, instructions and reference material.
4. *Provision for automated execution support*: requiring automated process parts, co-operative work support, a compilation of metrics and process integrity assurance.
5. *Process improvement support*: requiring the reuse of well-defined and effective software processes, the comparison of alternative processes and process development support.

Different process models can define different points of view. For example, one model may define the agents involved in each activity, while another may centre on the relationship between activities. There is a model that addresses the organisational culture and focuses on the behavioural capabilities or duties of the roles involved in the software process [2][3]. This means that each model observes, focuses on or gives priority to particular points of such a complex world as software construction [22][25]. A model is always an abstraction of the reality and, as such, represents a partial and simplified description of the reality; that is, not all the parts or aspects of the process are taken into account in the model. Generally, a process model can be divided into several sub-models expressing different viewpoints or perspectives. Additionally, there is a variety of notations for defining the process models as described in section 3.2.

The results achieved so far in software process research are comprehensively reviewed in [4] and [21], and [30] presents an overall critical evaluation and possible directions for future research.

3. ELEMENTS AND APPROACHES OF SOFTWARE PROCESS MODELLING

Firstly, we are going to deal with the basic process-related elements, as well as their possible interrelationships. The different approaches to modelling will then be presented from the viewpoint of the information they can address.

3.1. Basic Process Modelling Elements

The software process basically consists of two interrelated processes: the production process and the management process. The first is related to the construction and maintenance of the software product, whereas the second is responsible for estimating, planning and controlling the necessary resources (people, time, technological, ...) to be able to carry out and control the production process. This control makes it possible to generate information about the production process, which can be used later on by the management process with a view to improving the software process and increasing the quality of the developed software. There are two main submodels in process modelling: the management process model and the production process model [21].

Different elements of a process, for example, activities, products (artefacts), resources (personnel and tools) and roles, can be modelled [34]. There are several classifications concerning the main elements of a process model [44][25][16]. The elements most commonly modelled are presented below [11][26][47][29]. Figure 1 shows these elements and their interrelationships. Other possible elements like event, project/organisation, workspace (repository), user viewpoint, model of co-operation, versioning, transactions, quality, etc. will not be considered.

- *Agent* or *Actor*: is an entity that executes a process. Actors can be divided into two groups as regards their close relationship with the computer: a) human actors, who are the people who develop software or are involved in the software process and are possibly organised as teams; b) system actors or system tools, which are the computer software or hardware components. An actor is characterised by the properties of its role and its availabilities. An actor, person or system can play several roles, which are composed of consistent sets of activities. A role can be judged by several co-operative actors.
- *Role*: describes a set of agent or group responsibilities, rights and skills required to perform a specific software process activity. This is an association between agents and activities in terms of a defined set of responsibilities executed by agents.
- *Activity*: is the stage of a process that produces externally visible changes of state in the software product. An activity can have an input, an output and some intermediate results, generically termed products (they can be software products, requirements specification documents, database schemata, etc.). The activity includes and implements procedures, rules, policies and goals to generate and modify a set of given artefacts. This activity can be performed by a human agent or using a tool. The activities can be divided into more elementary activities; that is, there are elementary or compound activities. The activities can be organised in networks with both horizontal (chained) and vertical (decomposition) dimensions. The activities are associated with roles, other activities and artefacts.
- *Artefact* or *Product*: is the (sub)product and the "raw material" of a process. An artefact produced by a process can be used later as raw material for the same or another process to produce other artefacts. An artefact or software product is developed and maintained in a process. An artefact can have a long lifetime, and there may be different versions of each artefact as the software process evolves. The (sub)products can be created, accessed or modified during a process activity. A set of software artefacts to be delivered to a user is named software product. Therefore, a relationship "composed of" can appear between the products.

The relationships between these elements, such as activity/role/actor interaction, activity-activity interaction, product-activity interaction, product-product interaction, are described in [11].

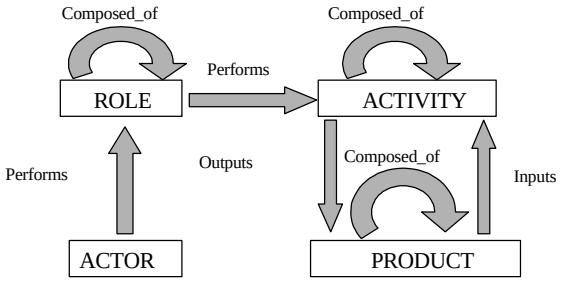

Figure 1. Basic process modelling components

Definitions of the process modelling concepts are found in [15][44] and the main software process models that use these concepts are described in [26].

3.2. Modelling Approaches

There are different types of process modelling. Processes can be modelled at different levels of abstraction (for example, generic models versus tailored models) and they can also be modelled with different goals (descriptive models versus prescriptive models). The distinguishing features of these models have been described in [35][46][47][49]. The types of information in a process model can be structured from different viewpoints. Curtis presents the following list of *information perspectives* commonly found in the literature [18]:

- *Functional*: represents which process elements are being implemented and which information entity flows are important for the above process elements.
- *Behavioural*: represents when (that is, sequentiality) and under which conditions the process elements are implemented.
- *Organisational*: represents where and by whom in the organisation the process elements are implemented.
- *Informative*: represents the information entities output or manipulated by a process, including their structure and relationships.

The models are built according to the languages, abstractions or formalisms created for representing the specific information about these characteristics. None of these abstractions (procedural programming language, systems analysis and design, artificial intelligence languages and approaches, events and triggers, control flow, state transition and Petri nets, functional languages, formal languages, data modelling, object modelling, etc.) covers all the classes of information. Therefore, most of the models found in the literature present languages based on more than one of these abstractions. These models consider the need to integrate multiple representation paradigms (that is, different process models), although this generally makes the model more complex to define [57].

On the one hand, a wide variety of notations have been used to model processes. A text that describes different notations for modelling the process is [55]. There are no data on which of the available notations are more useful or easier to use under given conditions.

On the other hand, as mentioned above, a variety of multi-paradigm approaches have been proposed for modelling software processes. These approaches have two main characteristics. Firstly, excepting mainly Petri net-based approaches, they all use textual representations. Secondly, most of them are already at a programming language level of abstraction. Thus, they make it difficult to model a problematic situation directly, whereas they force the engineer or process manager to take into account many technical aspects of the modelling formalism [24].

4. OVERVIEW OF PRESCRIPTIVE SOFTWARE PROCESS MODELS

The goal of prescriptive modelling is to define the required or recommended means of executing the process [47]. The software process models that go by the name of *prescriptive models* answer the question, "how should the software be developed?" [44]. McChesney [47] divides this group of models into two categories (manual prescriptive models and automated prescriptive models), depending on the goal of the prescriptions.

Manual prescriptive models can be standards, methodologies and methods centred on management, development, evaluation and software life cycle and organisational life cycle support processes. The models belonging to this category include: a) traditional structured methodologies; b) object-oriented methodologies; c) knowledge engineering methodologies; d) organisational design methodologies, which are considered as some of the most representative in the field of socio-cultural systems [27][51]; e) software life cycle development process standards; and f) software process evaluation standard- or model-based methods.

The *automated prescriptive models* perform activities related to assistance, support, management and/or computer-assisted software production techniques. The models belonging to this category include: ALF [13]; IPSE 2.5 [58]; Marvel [39]; PMDB [48]; SOCCA [24]; SPADE [7][8]; TRIAD [50]. These models are computerised specifications of software process standards. Their main aim is to act as a guide for the software modelling process; that is, they are directed at aiding process agents by mechanically interpreting software process models [45]. The automated prescriptive models can be divided into two categories, depending on the guiding criterion selected to address software process modelling: a) activity-oriented models and b) people-oriented models.

Activity-oriented models focus on the functions, activities of and information about parts of the management, development and software life-cycle support processes. These include TRIAD, Marvel, IPSE 2.5 and SPADE, for example. People-oriented models focus on the specifications of the people involved in the software process and their relationships. They include SOCCA, ALF, PMDB. People are the least formalised factor in existing software process models. Their importance, however, is patent [54]: their behaviour is non-deterministic and subjective and has a decisive impact on the results of software production, which is a basically an intellectual and social activity [31]. Additionally, the non-specification of human resources means that the process does not reflect the actual status of the software process of the modelled organisation, having the added risk of processes not suited to the capability of the organisation's human resources being executed [52][59].

As mentioned above, a variety of multi-paradigm software process modelling approaches have been proposed. Most focus on one main paradigm, like rules (Marvel, for example),

imperative programs (TRIAD/CML, for example), activity-oriented programs (IPSE 2.5) or Petri nets (SPADE, for example). The integration of formalisms to describe both the software process and the members of the process and their relationships, which explicitly includes the human components and any interaction in which they are involved, has been proposed by approaches such as SOCCA, ALF and PMDB. The complexity of the resulting software, which over-stretches both the technical and organisational management, can be better dealt with in this manner.

Each of these classes is addressed below from the viewpoint of model criteria, representation criteria and methodological criteria in order to identify the elements that represent the software process and, therefore, model the process in question. The prescriptive models considered are shown in Figure 2. Just two process definition standards were considered for manual models. The most important automated models were chosen [5][45][47][30]. These prescriptive models are characterised according to the above-mentioned criteria in Table 1. If they meet a table characteristic, the respective table cell is marked with X, otherwise the space is left blank.

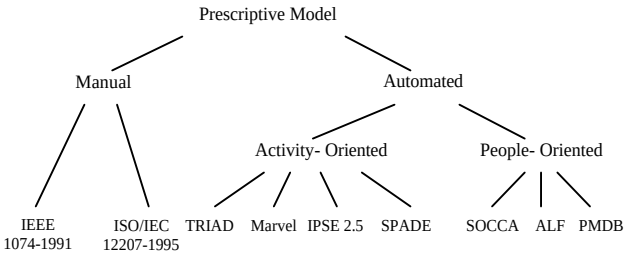

Figure 2. Prescriptive models

In relation to the manual prescriptive models, the software life cycle process standards are informal and consider only the functional perspective of the information in the model, as shown under the columns labelled *Notation characteristics - From the viewpoint of the information quality* (informal) and *Information perspectives* (functional).

The table shows that the representations of prescriptive software process models have focused on three elementary process features: the activity, the artefact and the agent (human and computerised) [47]. However, other characteristics, like human and organisational roles, have been empirically proven to have a big impact on the production process [32][17][14][56][23][53][40]. Most of the existing software process models deal with roles partially [43][26], while the organisation is considered as separate from the characteristics applied for modelling the software process or it is ignored [24]. This is because the organisation is the software process environment and is not, therefore, explicitly modelled. Therefore, these software process models are not integral and joint modelling approaches to the technical and organisational characteristics of the software process.

According to Table 1, the most applicable model is SOCCA, as the model formally addresses human resources and the software process interactions in which they are involved and specifies a procedure for building and adapting generic software process models, as reflected under the columns labelled *Modelling procedure (developed) and Procedure definition (defined)*. However, the proposed guides do not cover all the desired concepts, objects, characteristics and software process environments nor is there a fully comprehensive and formal modelling process [45], as observed under the column *Procedure coverage(partial)*.

Table 1. Characterisation of prescriptive models

MODEL	***Model Criteria***										***Representation Criteria***									***Methodological Criteria***						
	Process elements represented by the model					Process environments addressed by the model					Information perspectives				Notation characteristics					Modelling procedure		Procedure coverage		Procedure definition		
							cultural								from the viewpoint of information quality			from the viewpoint of formal notation									
	agent	activity	artefact	role	event	organisational	creative ability	social interaction	environment flexibility	scientific / technological	functional	behavioural	organisational	informative	informal	formal	automated	text	graphic	non-developed	developed	partial	sufficient	undefined	semi-defined	defined
IEEE 1074 - 1991		X	X							X	X				X					X						
ISO/IEC 12207 - 1995	X	X	X			X	X			X	X				X						X	X			X	
TRIAD	X	X		X						X	X	X		X			X	X		X						
Marvel	X	X	X							X	X	X					X	X		X						
IPSE 2.5	X	X	X	X						X			X	X		X		X		X						
SPADE	X	X	X		X					X	X	X	X				X		X	X						
SOCCA	X	X	X	X		X					X	X	X	X		X		X	X		X	X				X
ALF	X	X	X		X	X				X	X	X	X	X			X	X		X						
PMDB	X		X			X				X			X	X			X	X		X						

Owing to the complexity of and the requirements for modelling the software process, the best thing appears to be for each low-level modelling or software process implementation to be derived from a high-level specification step, where all the information perspectives can be modelled: a) separately and b) as integrated components of a full specification. Such a modular specification reduces the complexity of the modelling activity and increases the possibility of software process model change and evolution. SOCCA, ALF and PMDB consider these aspects, generally taken into account from the software engineering viewpoint. These are the three most advanced and complete models. They model the software project life cycle process and investigate the use of a process model, in this case SOCCA, ALF or PMDB, as a conceptual and user interaction model and as a framework for integrating tools and/or agents and roles. The approach is incremental, as the whole life cycle is long and complex. Special emphasis is placed on process formalisation, large project support, generalisation and extendibility. Additionally, as mentioned above, SOCCA gives directions on an aspect that is often neglected in software engineering: the problem of the specification under development having to describe not only the technical but also the human parts of the software process, or better still, the human resources, the software process and all classes of interaction between the different non-human and human parts. However, SOCCA does not deal with the cultural software process modelling questions, absent as well in the other models analysed or considered informally in the socio-cultural models [27][51], as observed under the *creative capability, social interaction and environment flexibility* columns of the *Cultural environment* of the process addressed by the model. Table 1 shows that the organisational, cultural and scientific/technological environments are not integrated in the models in question (as shown under the columns of the same name).

Finally, Table 1 evidences one of the main limitations of existing models: the lack of proposed guides for identifying and defining all the essential organisational and software process conceptual modelling process components. In other words, it confirms that all the prescriptive models considered lack a *defined and fully comprehensive software process modelling procedure*. This procedure should provide a formally integrated model of both the software processes and the organisational environments [24][13][57].

5. REFERENCES

[1] S. T. Acuña, M. Lopez, N. Juristo, A. Moreno, "A process model applicable to software engineering and knowledge engineering". *International Journal of Software Engineering and Knowledge Engineering* **9**, 5 (1999) 663-687.

[2] S. T. Acuña, G. Barchini, M. Sosa, "A culture-centered multilevel software process cycle model", *Proceedings of the 22nd International Conference on Software Engineering* (June 2000) 775.

[3] S. T. Acuña, G. Barchini, C. Laserre, A. Silva, M. Sosa, V. Quincoces, "Software engineering and knowledge engineering software process: Formalizing the who's who", *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineeering* (July 2000) 221-230.

[4] V. Ambriola, R. Conradi, A. Fuggetta, "Assessing process-centered software engineering environments". *ACM Transactions on Software Engineering and Methodology* **6**, 3 (1997) 283-328.

[5] P. Armenise, S. Bandinelli, C. Ghezzi, A. Morzenti, "A survey and assessment of software process representation formalisms". *International Journal of Software Engineering and Knowledge Engineering* **3**, 3 (1993) 401-426.

[6] S. C. Bandinelli, A. Fuggetta, C. Ghezzi, "Software process model evolution in the SPADE environment". *IEEE Trans. Software Engineering* **19**, 12 (December 1993) 1128-1144.

[7] S. C. Bandinelli, A. Fuggetta, C. Ghezzi, L. Lavazza, "*SPADE: An environment for software process analysis, design, and enactment*". In Software Process Modelling and Technology chapter 9. Research Studies Press (1994) 223-247.

[8] S. Bandinelli, A. Fuggetta, L. Lavazza, M. Loi, G. Picco, "Modeling and improving an industrial software process". *IEEE Transactions on Software Engineering* **21**, 5 (1995) 440-454.

[9] L. Baresi, F. Casati, S. Castano, M. G. Fugini, I. Mirbel, B. Pernici, "WIDE workflow development methodology". In *Software Engineering Notes in Work Activities Coordination and Collaboration: Proceedings of the International Joint Conference* **24**, 2 (ACM Press, 1999) 19-28.

[10] V. R. Basili, H. D. Rombach, "The TAME project: Towards improvement-oriented software environments". *IEEE Trans. Software Engineering* **14**, 6 (June 1988) 758-773.

[11] K. Benali, J. C. Derniame, "Software processes modeling: what, who, and when". *Proceedings of the Second European Workshop on Software Process Technology* (September 1992).

[12] B. I. Blum, *Software Engineering: A Holistic View*. (Oxford University Press, 1992).

[13] G. Canals, N. Boudjlida, J. C. Derniame, C. Godart, J. Lonchamp, "*ALF: A framework for building process-centred software engineering environments*". In Software Process Modelling and Technology chapter 7. (Research Studies Press, 1994) 153-185.

[14] R. Carasik, C. Grantham, "A case study of CSCW in a dispersed organization". *Proceedings of CHI'88*, Human Factors in Computing Systems, ACM (1988) 61-65.

[15] R. Conradi, C. Fernström, A. Fuggetta, R. Snowdon, "Towards a reference framework for process concepts". *Proceedings of the Second European Workshop on Software Process Technology* (September 1992) 3-17.

[16] R. Conradi, C. Fernström, A. Fuggetta, "*Concepts for evolving software processes*". In Software Process Modelling and Technology chapter 2. (Research Studies Press, 1994) 9-31.

[17] B. Curtis, H. Krasner, N. Iscoe, "A field study of the software design process for large systems". *Communications of the ACM* **31**, 11 (November 1988) 1268-1287.

[18] B. Curtis, M. Kellner, J. Over, "Process modeling". *Communications of the ACM* **35**, 9 (September 1992) 75-90.

[19] A. Davis, *Software Specification. Events, Objects and Functions*. (Prentice Hall, 1993).

[20] W. Deiters, V. Gruhn, "Software process analysis based on FUNSOFT nets". *Systems Analysis Modelling Simulation* **8**, 4-5 (1991) 315-325.

[21] J. C. Derniame, B. A. Kaba, D. Wastell, *Software Process: Principles, Methodology and Technology*. Lecture Notes in Computer Science 1500. (Springer, 1999).

[22] M. Dowson, B. Nejmeh, W. Riddle, "Concepts for process definition and support". *Proceedings of the Sixth International Software Process Workshop* (October 1990).

[23] L. Druffel. "Professionalism and the Software Business". *IEEE Software* **11**, 4 (July 1994) 6.

[24] G. Engels, L. Groenewegen, "*SOCCA: Specifications of coordinated and cooperative activities*". In Software Process Modelling and Technology chapter 4. (Research Studies Press, 1994) 71-102.

[25] P. H. Feiler, W. S. Humphrey, "Software process development and enactment: Concepts and definitions". *Proceedings of the Second International Conference on Software Process* (February 1993) 28-40.

[26] A. Finkelstein, J. Kramer, B. Nuseibeh, *Software Process Modelling and Technology*. (Research Studies, 1994).

[27] R. Flood, M. Jackson, *Creative Problem Solving: Total Systems Intervention*. (John Wiley & Sons, 1991).

[28] X. Franch, J. M. Ribó. "Using UML for software process modelling". *Proceedings of the Second International Conference of the Unified Modeling Language*, UML'99, IEEE (October 1999) 292-307.

[29] A. Fuggetta, A. Wolf, *Software Process*. (John Wiley & Sons, 1996).

[30] A. Fuggetta, "*Software process: A roadmap*". In The Future of Software Engineering, Ed. A. Finkelstein, (ACM Press, 2000) 27-34.

[31] V. Gruhn, "Software processes are social processes". Technical Report University of Dortmund. (February 1992).

[32] R. Guindon, B. Curtis, "Control of cognitive processes during design: What tools would support software designers?". *Proceedings of the CHI'88*, Human Factors in Computing Systems, (ACM, 1988) 263-268.

[33] D. S. Hinley, "Software evolution management: A process-oriented perspective". *Information and Software Technology* **38**, 11 (November 1996) 723-730.

[34] K. Huff, "*Software process modeling*". In Software Process. (John Wiley & Sons, 1996).

[35] W. Humphrey, *Managing the Software Process*. (Addison Wesley, 1989).

[36] *IEEE Standard for Developing Software Life Cycle Processes*, IEEE Standard 1074-1991.

[37] ISO/IEC *International Standard: Information Technology. Software Life Cycle Processes*, ISO/IEC Standard 12207-1995.

[38] G. E. Kaiser, P. H. Feiler, "An architecture for intelligent assistance in software development". *Proceedings of the Ninetieth International Conference on Software Engineering* (April 1987) 180-188.

[39] G. E. Kaiser, P. H. Feiler, S. S. Popovich, "Intelligent assistance for software development and maintenance". *IEEE Software* (May 1988) 40-49.

[40] P. Kawalek, D. G. Wastell, "Organisational design for software development: a cybernetic perspective". In *Lecture Notes in Computer Science, Software Process Technology: Proceedings of the 5th European Workshop* 1149 (Springer-Verlag, 1996) 258-270.

[41] M. I. Kellner, "Software process modelling support for management planning and control". *Proceedings of the First International Conference on Software Process* (October 1991) 8-28.

[42] J. Lonchamp, K. Benali, J. C. Derniame, C. Godart, "Towards assisted software engineering environments". *Information and Software Technology* **33**, 8 (October 1991) 581-593.

[43] J. Lonchamp, "Supporting social interaction activities of software processes". *Proceedings of the Second European Workshop on Software Process Technology*, EWSPT'92 (September 1992) 34-54.

[44] J. Lonchamp, "A structured conceptual and terminological framework for software process engineering". *Proceedings of the Second International Conference on Software Process* (February 1993) 41-53.

[45] J. Lonchamp, "An assessment exercise". In Software Process Modelling and Technology chapter 13. (Research Studies Press, 1994) 335-356.

[46] N. H. Madhavji, "The process cycle". *Software Engineering Journal* **6**, 5 (September 1991) 234-242.

[47] I. R. McChesney, "Toward a classification scheme for software process modelling approaches". *Information and Software Technology* **37**, 7 (1995) 363-374.

[48] M. H. Penedo, C. Shu. "Acquiring experiences with the modelling and implementation of the project life-cycle process: the PMDB work". *Software Engineering Journal* **6**, 5 (September 1991) 259-274.

[49] S-L. Pfleeger, *Software Engineering: Theory and Practice*. (Prentice-Hall, 1998).

[50] J. Ramanathan, S. Sarkar, "Providing customized assistance for software lifecycle approaches". *IEEE Trans. Software Engineering* **14**, 6 (June 1988) 749-757.

[51] R. A. Rodríguez Ulloa, *Strategic Management and Control Processes in the Peruvian Culture: The Systemic Methodology for Strategic Management* (SM: SM). Ph.D Thesis, University of Lancaster, Lancaster, (1991).

[52] C. B. Seaman, V. R. Basili, "OPT: An approach to organizational and process improvement". *Proceedings of AAAI Symposium on Computational Organizational Design* (March 1994).

[53] K. Sherdil, N. H. Madhavji, "Human-oriented improvement in the software process". In *Lecture Notes in Computer Science, Software Process Technology: Proceedings of the 5th European Workshop* 1149 (Springer-Verlag, 1996) 145-166.

[54] Sommerville, T. Rodden, "*Human, social and organisational influences on the software process*". Lancaster University. Computing Department. Cooperative Systems Engineering Group. Technical Report: CSEG/2/1995. (1995) 1-21.

[55] SPC, *Process Definition and Modeling Guidebook*. Software Productivity Consortium, SPC-92041-CMC. (1992).

[56] J. D. Valett, F. E. McGarry, "A summnary of software measurement experiences in the Software Engineering Laboratory". *Journal of Systems and Software* **9**, 2 (1989) 137-148.

[57] F. M. de Vasconcelos Jr., C. M. L. Werner, "Software development process reuse based on patterns". *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering* (June 1997) 97-104.

[58] B. Warboys, "The IPSE 2.5 project: Process modelling as the basis for a support environment". *Proceedings of the First International Conference on System Development Environments and Factories* (May 1989).

[59] E. S. K. Yu, J. Mylopoulos, "Understanding "why" in software process modelling, analysis, and design". *Proceedings of the 16th International Conference on Software Engineering* (May 1994) 1-10.

View publication stats