

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>
Associate Editors:

- François Louis Nicolet, Switzerland, <nicolet@acm.org>
- Roberto Carniel, Italy, <carniel@dgt.uniud.it>
- Zakaria Maamar, Arab Emirates, <Zakaria.Maamar@zu.ac.ae>
- Soraya Kouadri Mostéfaoui, Switzerland, <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>

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/>>.

Key Issues and New Challenges in Software Process Technology

Jean-Claude Derniame and Flavio Oquendo

In the last two decades we have seen a tremendous development in software process research. During that time there has been considerable progress in developing the technological base for supporting software engineering processes. However, the changing face of technology and methodology (in particular agile methods), the ever increasing complexity of software systems, and the revolutionary development in the Internet have led to many interesting challenges and opportunities for new developments in Software Process Technology. This paper examines some of the important trends of software process in research and practice, and speculates on the important emerging challenges.

Keywords: Process Enactment, Process Modelling, Research Directions, Software Process Technology.

1 Introduction

The *software process* of developing and maintaining a product or a service plays a crucial role in determining the quality level of the product or service but also the cost of developing, supporting and maintaining it.

Process has been recognized important for decades in manufacturing, but it became, more lately, a priority in software production and high-tech service provisioning. In fact, software producers and telecommunications service providers, from small vendors to giants like Microsoft and AT&T, have started to model, analyse, and re-engineer or improve the processes used to produce, support and maintain their products and services. Process improvement has finally been identified as a major area in the high-tech industry.

The main stream of effort in industry, standardization bodies and institutions has been devoted to process analysis and assessment, which is fundamental when exchanging products. Main stream of effort in research has been on defining languages and building process-centred software engineering environments. From 1990 up to now a lot of them has been prototyped, [6] and some of them industrialized.

Software engineering researchers have studied the software production process quite thoroughly for many years now [4]. They have set two research goals: (1) developing a process modelling, analysis and improvement methodology and (2) improving process support technology.

The first goal motivated the development of numerous process life cycle models, such as the waterfall model and the spiral model [2], and methodological approaches to structuring, organizing, documenting and formally describing processes in order to evaluating or improving them, such as the Capability Maturity Model (CMM) [14], Bootstrap [9], and the SPICE ISO/IEC 15504 Standard.

The second goal motivated the development of Process-centred Software Engineering Environments (PSEE), which are software systems that assist in the modelling and automa-

Jean-Claude Derniame is currently teaching at *Institut Polytechnique de Lorraine* at Nancy, France. He holds a Research Direction Habilitation in Computer Science from the University of Nancy, France. He has been a Full Professor in Computer Science at the University of Nancy since 1979. Prof. Derniame has published more than 150 refereed papers in a range of software engineering areas (especially programming environments, software architecture, software process and computer-assisted environments, but also in Graph theory, social impact of new technologies, new technologies and developing Countries). He has advised 70 theses. He had active participation and responsibilities in several projects such as PCTE (ESPRIT), PCTE+ (IEPG-TA13), ALF (ESPRIT), PCIS (NATO), PCIS II (US-DOD French MOD), Wise Dev (World Bank), SIMES (European PCRD). He was animator of the European research groups on software process PROMOTER I and II (ESPRIT BRA-WG), and chaired the steering committee of the EWSPT series. <derniame@loria.fr>

Flavio Oquendo holds a PhD and a Research Direction Habilitation in Computer Science from the University of Grenoble, France. He has been a Full Professor in Computer Science at the University of Savoie since 1995. Prof. Oquendo has a high level of expertise and experience in the field of Software Engineering, including active participation in 10 European R&D Projects, including ALICE (MAP), PCTE (ESPRIT), PCTE+ (IEPG-TA13), PACT (ESPRIT), ALF (ESPRIT), SCALE (ESPRIT), PROMOTER I/II (ESPRIT BRA-WG), PIE (ESPRIT LTR), and ARCHWARE (IST). Prof. Oquendo has published well over 100 refereed papers in a range of software engineering areas (especially in software architecture, software process and computer-assisted environments). He has served on programme committees of 18 international conferences and workshops (last year he has served as Programme Chair of the 9th European Workshop on Software Process Technology – EWSPT 2003, Springer-Verlag LNCS 2786). He has also act as referee for many international journals. His current research interests include formal description and development techniques for software architecture and process specification, refinement, implementation, monitoring, and evolution. <flavio.oquendo@univ-savoie.fr>

tion through enactment of software development processes. Results of this research and development work have been presented in several major international conferences and workshops: EWSPT (European Workshop on Software Process Technology) series published since 1992 as LNCS (Lecture Notes in Computer Science) by Springer Verlag, ISPW (International Software Process Workshop) published by IEEE, and ICSP (International Conference on the Software Process) series published by ICSA (The International Software Process Association), ACM Press. Promising approaches, technological advances, and experiences in applying the process technology have been published in proceedings and journals. Some books also propose synthesis of the domain [8][4][15].

Since 1984 there has been considerable progress in developing the technological base for supporting software processes, including Process Modelling Languages (PML) and Environments (see [1] for a comparative review of the state of the art). However, the changing face of technology and methodology (in particular agile methods), the ever increasing complexity of software systems, and the revolutionary development in the Internet have led to many interesting challenges and opportunities for new developments in Software Process Technology. This paper examines some of the important trends and challenges of software process in research and practice.

The remainder of this paper is organised as follows. Section 2 introduces a conceptual and terminological framework in order to structure software process concerns. Section 3 introduces the CMM levels of software process maturity. Section 4 summarizes where we are today in Software Process Technology. Section 5 speculates on the important emerging challenges of Software Process Technology. Finally, Section 6 concludes the paper.

2 Process Framework

We will use a conceptual and terminological framework [5] in order to present key issues and future directions in Software Process Technology. This conceptual framework, sketched in Figure 1, introduces three software process domains:

- Process model domain,
- Process enactment domain,
- Process performance domain.

The process model domain contains characterizations of processes or fragments of processes, expressed in some notation, i.e. a software process modelling language, in terms of how they could or should be enacted/performed. A *software process model* is a software process abstraction described with a formal or semi-formal software process modelling language.

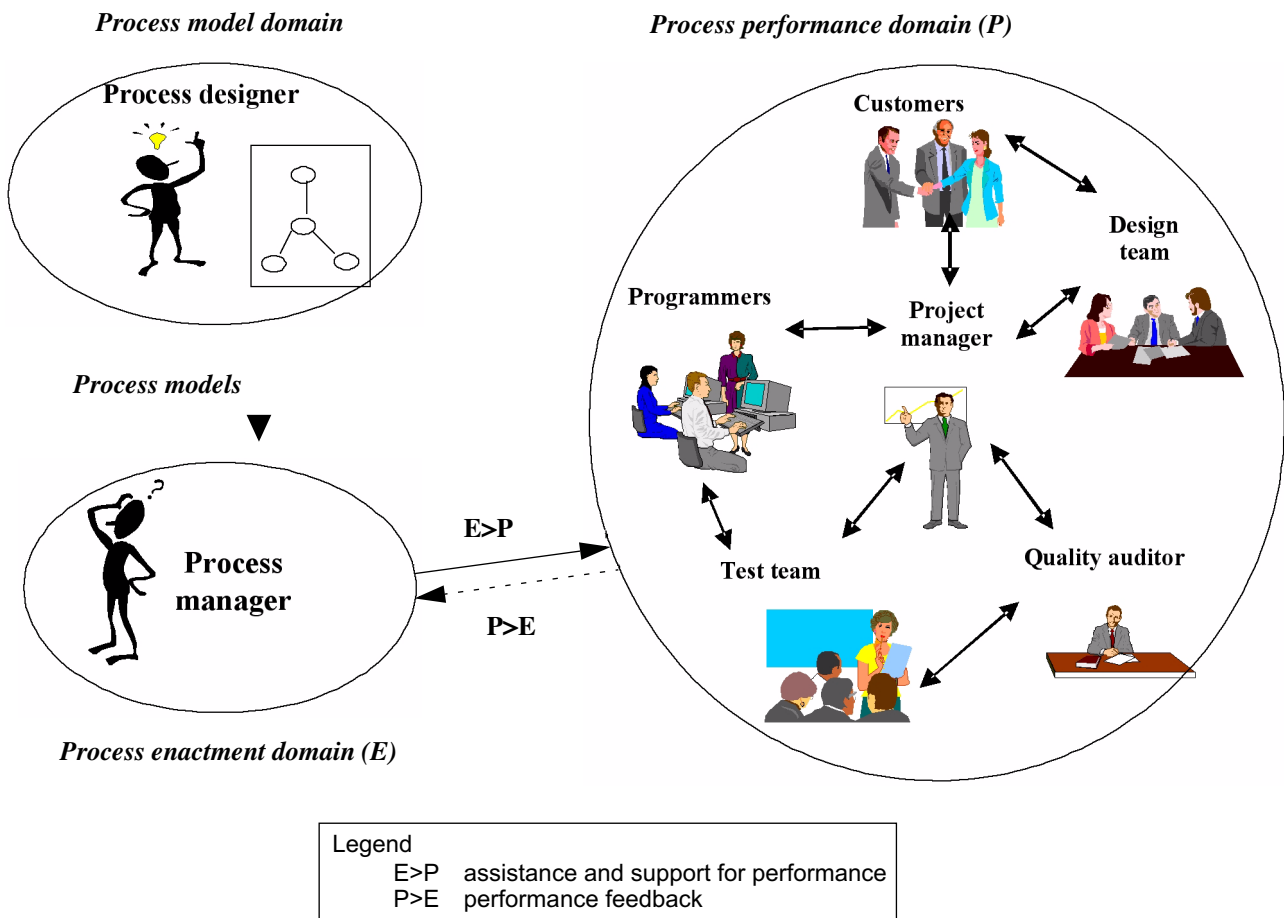


Figure 1: Software Process Domains.

A *meta-process model* represents the set of (meta-)activities to model, to analyse software processes and to support their evolution.

The process enactment domain encompasses what takes place in a process-sensitive software engineering environment to support process performance governed by process models. Enactment comprises customization and instantiation. A *customized process model* is said to be *instantiated* when its artefacts are linked with concrete products and project resources. A *customized process model* is the result of the refinement and adaptation of a generic process model to a specific project. A *Process-centred Software Engineering Environment* encompasses the set of mechanisms that provides a variety of support (assistance, guidance, monitoring, automation, etc.) to software process performers by enacting an explicit representation (i.e. model) of this process.

The process performance domain encompasses the actual tasks and activities that are performed by the process agents (human or not) in the course of a software process. A *software process* is defined by the set of technical and managerial activities carried out in the production and the maintenance of software. It is a partially ordered set of activities each of them is associated with its related artefacts, human and computerized resources, constraints, policies, etc. In the process performance domain, one may discern between the *performer* who may be a project manager, a programmer, a system analyst, a quality auditor, a tester, or the *end-user* who is the user of the product that is developed.

3 Process Maturity

Software process maturity is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. As stated in [13], maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization. The software process is well-understood throughout a mature organization, usually through documentation and training, and the process is continually being monitored and improved by its users. The capability of a mature software process is known. Software process maturity implies that the productivity and quality resulting from an organization's software process can be improved over time through consistent gains in the discipline achieved by using its software process. Process maturity levels proposed by CMM are sketched in Figure 2.

These five levels of software process maturity can be characterized as follows:

- 1) **Initial.** The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.
- 2) **Repeatable.** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
- 3) **Defined.** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organi-

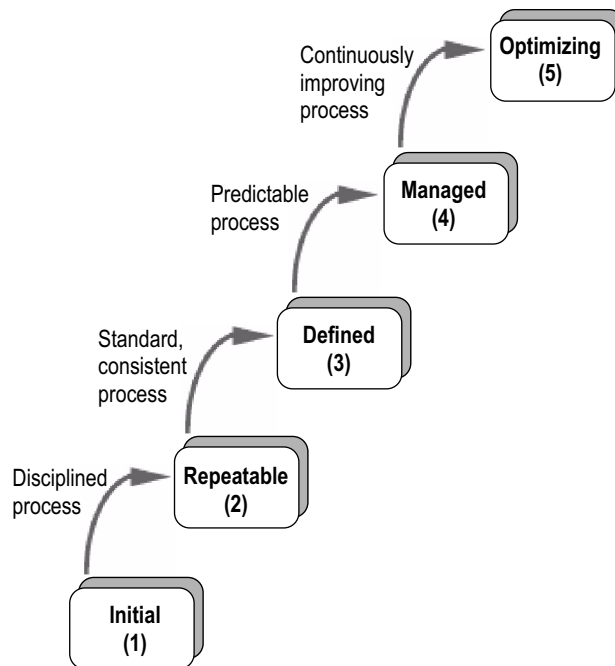


Figure 2: The Five Levels of Software Process Maturity.

zation. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

- 4) **Managed.** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
- 5) **Optimizing.** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Maturity levels 2 through 5 can be characterized through the activities performed by the organization to establish or improve the software process. But why Software Process Technology is relevant to practitioners wanting to move to higher process maturity levels? Because they cannot get to CMM levels 3, 4, and 5 without Software Process Technology. Indeed, Software Process Technology can provide support for definition, measurement, analysis, monitoring, guidance, and automation.

4 Yesterday and Today

On the one hand, software process modelling and improvement are key aspects to practitioners to move to higher process maturity levels, thereby mastering the process of developing and maintaining software. On the other hand Software Process Technology provides a fundamental support for getting to higher process maturity levels. How much progress have we made during these two decades?

Yesterday

In the distant past of twenty years ago, software process was largely an ad hoc affair. Definitions relied on informal diagrams, which were rarely followed by software engineers in organizations. The notion of software process was ill-understood.

Today

Much has changed in the past two decades. Although there is wide variation in the state of the practice, generally speaking, software process is much more visible as an important and explicit support activity in software development. Job titles now reflect the role of software process engineers and companies recognize the importance of software process maturity and invest in order to reach higher process maturity levels.

In addition, the technological basis for software process has improved dramatically. Important advancements have been the development of process modelling languages and process-centred software engineering environments as summarized hereafter.

4.1 Process Modelling Languages

A lot of PML proposals has been done relying on different paradigms such as logic-based, procedural, rule-based, multi-agents, active-databases, Petri nets, object oriented languages. Process-centred environments have been built for them. Very few have transformed to become product but a lot of ideas have been used in current products. The main objective for an industrial company, using PML, is probably to capitalize on the employees experience to enhance the products quality. Capitalizing on experience means to be able to reuse some parts of process models, to share them, to distribute them, to adapt them, to assembly components, to make components working together. All this is a plea to have modular PMLs and to standardise them. Standards exists, e.g. ISO/IEC 12207 and OMG's SPEM (Software Process Engineering Metamodel).

The proposed PMLs enable an unambiguous description of the process (level 2). Most of them focus on performance support, and they permit, as well as standards, to reach the level 3 in CMM classification. Some of them address the process model improvement problem. An unambiguous representation of the process may also provide to the project management team the mean to monitor the project under development with respect to the plan, the mean to react to the deviations and to trace the ongoing process. Progress in this direction is obviously desirable to reach level 4 and 5.

Describing and modelling software processes can be done for many different purposes from understanding, towards performance, passing through evaluation, performance, guidance, improvement, etc. each purpose can be addressed with different strategies. Hoping to make a unique PML largely adopted is certainly utopian. As a consequence, we need several PMLs or, preferably, several facets, issued from a common PML architecture, acting in the same environment, and supporting these different purposes,

Maybe, these languages are not yet at the adequate level of abstraction to reach this objective. Reaching the good level of abstract should be based on a more experimental approach.

4.2 Process-centred Software Engineering Environments

This multi-purpose dimension is peculiarly important if the PSEE must support process performance, which means providing a variety of supports (assistance, guidance, monitoring, automation, etc) to software process performers. In this case,

requirements concern at one and the same time PML and PSEE. Between them, some address real challenges [1].

- *The PSEE should support dynamic ordering of activities:* If ordering of activities can be dynamically built and modified, the PSEE enactment engine will be able to continue to support and assist process performance. Humans interacting with the PSEE, distributing and managing the "control" between them is a key issue: with a balance to offer in terms of discipline vs. initiative, modelling and controlling facts vs. intentions, etc. The fundamental discussion about "Software Processes are software too" raised by [12] and [10] is obviously not finished. In order to progress towards level 4, the PML, and the PSEE as PML support, should support flexibility to be usable within the strategy adopted by a company, which can go from a strict disciplined "plan-driven" process towards a fully free process where "deviation is a standard".
- *The PSEE should support software process distribution,* which encompasses process modularity, heterogeneity, interoperability, composability, process fragments federation. It implies also that the PSEE must be able to support communication, coordination, cooperation and negotiation between user performers with their different roles. The PSEE process representation formalism must provide the means of representing performers' social interaction in the process enactment state and to keep this state updated about what happens in the performance domain. Performers' communication and negotiation may result in unexpected changes and decisions about the software process. In order to maintain consistency between enactment and performance states, these changes and the social interaction that leads to them needs to be represented in the enactment state.
- *The PSEE should support software process evolution:* off-line evolution and on-line evolution. In this case, consequences on on-going current processes and processes having yet passed beyond the point of change in the model must be considered. Most of the PSEEs proposed by researchers explore solution for a shared evolution but not when conflicts occur with process fragments already performed being in conflict with the changed model. PSEEs must also support private evolution: change will be local to the process model instance that is currently enacted, without impacting nor the enacted model, nor the model itself: process deviations against the model must be supported, negotiable, and their impact must be managed.

5 Tomorrow

What about the future? Although software process is on a much more solid footing than two decades ago, it is not yet established as a discipline that is taught and practised universally across the software industry. One reason for this is simply that it takes time for new approaches and perceptions to propagate. Another reason is that the technological basis for software process is still immature. In both of these areas we can expect that a natural evolution of the field will lead to steady advances.

However, the world of software engineering and the ways in which software is being developed are changing in significant

ways. These changes promise to have a major impact on how software process is practised. In the remainder of this section we consider three of the more prominent trends and their implications for the field of software process.

5.1 Agile Software Development

In the past few years there has been a rapidly growing interest in agile (aka “lightweight”) software development methodologies. Similarly, plan-driven methodologies¹ have been described as rigorous, disciplined, bureaucratic, heavyweight, and industrial-strength.

Several methodologies fit under this agile banner, including [7]:

- **XP (Extreme Programming):** XP builds an evolutionary design process that relies on refactoring a simple base system with each iteration. All design is centred around the current iteration with no design done for anticipated future needs. The result is a design process that is disciplined, yet startling, combining discipline with adaptivity in a way that arguably makes it the most well developed of all the adaptive methodologies.
- **Crystals:** The Crystals share a human orientation with XP, but this people-centredness is done in a different way. It explores a least disciplined methodology that could still succeed, consciously trading off productivity for ease of execution. It also puts a lot of weight in end of iteration reviews, thus encouraging the process to be self-improving. His assertion is that iterative development is there to find problems early, and then to enable people to correct them. This places more emphasis on people monitoring their process and tuning it as they develop.
- **ASD (Adaptive Software Development):** At the heart of ASD are three non-linear, overlapping phases: speculation, collaboration, and learning. It views planning as a paradox in an adaptive environment, since outcomes are naturally unpredictable. In traditional planning, deviations from plans are mistakes that should be corrected. In an adaptive environment, however, deviations guide us towards the correct solution.
- **Scrum:** Scrum focuses on the fact that defined and repeatable processes only work for tackling defined and repeatable problems with defined and repeatable people in defined and repeatable environments. Scrum divides a project into iterations (which they call sprints) of 30 days. Before you begin a sprint you define the functionality required for that sprint and then leave the team to deliver it. The point is to stabilize the requirements during the sprint. However management does not disengage during the sprint. Every day the team holds a short (fifteen minute) meeting, called a scrum, where the team runs through what it will do in the next day.
- **FDD (Feature Driven Development):** FDD like other adaptive methodologies focuses on short iterations that deliver tangible functionality. In FDD’s case the iterations are two weeks long. FDD has five processes. The first three are done

1. Plan-driven was coined by Barry Boehm [3] to characterize the opposite end of the planning spectrum from agile methodologies.

at the beginning of the project: Develop an Overall Model, Build a Features List, and Plan by Feature. The last two are done within each iteration: Design by Feature and Build by Feature. Each process is broken down into tasks and is given verification criteria.

Agile methodologies imply disciplined processes, even if the implementations differ in extreme ways from traditional software engineering and management practices [13]. The challenge here for Software Process Technology is how to support agile methodologies, even though agile methodologies appear to be incompatible in principle with the discipline of “plan-driven” software process modelling and enactment. In addition, the implementation of those methodologies must be aligned with the spirit of the agile philosophy and with the needs and interests of the customer and other stakeholders. Perhaps the biggest challenge in providing process languages and process-centred environments for effectively addressing both agile and plan-driven methodologies is dealing with extremists in terms of process deviation.

5.2 Open Source Software Development

There is a definite way of doing things in the open source community, and much of their approach is as applicable to closed source projects as it is to open source. In particular their process is geared to physically distributed teams, which is important because most adaptive processes stress co-located teams.

Open source software development is distributed by nature. Most open source projects have one or more maintainers. A maintainer is the only person who is allowed to commit a change into the source code repository. Different projects handle the maintainer role in different ways. Some have one maintainer for the whole project, some divide into modules and have a maintainer per module, some rotate the maintainer, some have multiple maintainers on the same code, others have a combination of these ideas. Even if the coordination process is devised, the software process for open-source is not well written up.

The challenge here for Software Process Technology is how to support the freedom of Open Source Software Development while enforcing personal and coordination processes.

5.3 Global Software Development

Global software development is increasingly becoming common practice in the software industry, as the ability to develop software at remote sites in projects such as development outsourcing allows organizations to ignore the geographical distance and benefit from access to a qualified resource pool and a reduction in development costs.

Indeed, the increased globalization of software development creates software process challenges due to the impact of temporal, geographical and cultural differences, and requires development of models to address these issues. Besides addressing these issues, the challenge here for Software Process Technology is how to support distributed, heterogeneous, dynamically created and managed software processes while maintaining and improving the “global view”.

6 Concluding Remarks

In this paper we have attempted to provide a high-level overview of key issues and new challenges in software process technology by presenting where we have come over the past years and speculating about needs for the coming years. Indeed, the field of software process is one that has experienced considerable growth over the past two decades. As software engineering matures into an engineering discipline, there are a number of process-related challenges that will need to be addressed. Many of the solutions to these challenges are likely to arise as a natural consequence of maturation of software process practices and technology that we know about today. New challenges arise because of the shifting landscape of software engineering methods and the needs of process support for agile and Internet-enabled software development. Other challenges will come from new paradigms for engineering software such as the product line approach to software development and model-driven engineering.

References

- [1] S. Arbaoui, J.-C. Derniame, F. Oquendo, H. Verjus. "A Comparative Review of Process-Centered Software Engineering Environments", *Int. Journal: Annals of Software Engineering*, Special Issue on Process-Based Software Engineering, Vol. 14, No. 1-4, December 2002.
- [2] B. W. Boehm. "A Spiral Model of Software Development and Enhancement", *ACM SIGSOFT Software Engineering Notes*, Vol. 11, No 4, 1986.
- [3] B. W. Boehm. "Get Ready for Agile Methods, With Care", *IEEE Computer*, January 2002.
- [4] J. C. Derniame, D. Wastell, A. Kaba (eds). *Software Process: Principles, Methodology and Technology*, LNCS N°1500, Springer Verlag, January 1999.
- [5] M. Dowson. "Consistency Maintenance in Process Sensitive Environments", *Proceedings of the Workshop on Process Sensitive SEE Architectures*, Boulder, VA, USA, 1992.
- [6] A. Finkelstein, J.Kramer, B.J, Nuseibeh (eds). *Software Process Modeling and Technology*, Wiley& Sons, London, 1994.
- [7] M. Fowler. *The New Methodology*, <<http://www.martinfowler.com/articles/newMethodology.html>>, April 2003.
- [8] A. Fuggetta, A. Wolf (eds). *Software Process*, Vol. 4, *Trends in Software*, J. Wiley & Sons, New York, 1996.
- [9] P. Kuvaja. "Software Process Assessment and Improvement: The Bootstrap Approach", Blackwell, Oxford, UK, 1994.
- [10] M. M. Lehman. "Process Models, Process Programming, Programming Support", *Proceedings of the 9th International Conference on Software Engineering*, Monterey,1987 (Response to an ICSE'9 Keynote Address by Leon Osterweil).
- [11] F. Oquendo (Ed). *Proceedings of the 9th European Workshop on Software Process Technology (EWSPT 2003)*, Springer-Verlag, LNCS 2786, Helsinki, Finland, 2003.
- [12] L. J. Osterweil. "Software Processes are Software too", *Proceedings of the 9th International Conference on Software Engineering*, Monterey,1987
- [13] M. C. Paulk. "Extreme Programming From a CMM Perspective", *IEEE Software* 34-11, 2001.
- [14] M. C. Paulk, C.V. Weber, S.M. Garcia, M.B. Chrissis, M. Bush. *Key Practices of the Capability Maturity Model, Version 1.1*, Technical Report, CMU/SEI-93-TR-025, ESC-TR-93-178, Pittsburgh, PA, USA, 1993.
- [15] B. Westfechtel. "Models and Tools for Managing Development Processes", LNCS 1646, Springer Verlag, Berlin, Germany, 1999.