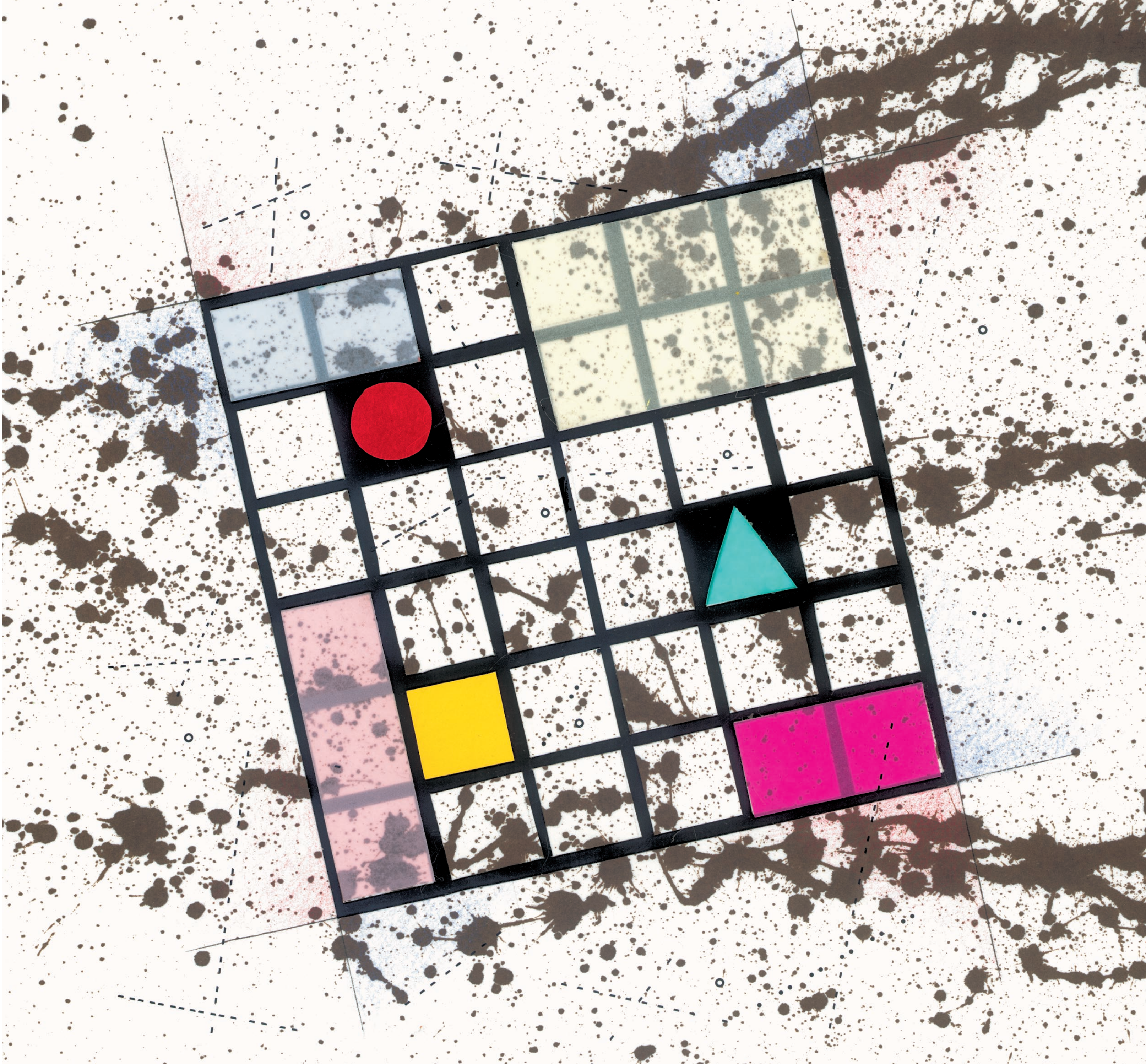
CEPIS

UPGRADE

The European Journal for the Informatics Professional

http://www.upgrade-cepis.org

Vol. V, No. 5, October 2004



SPT
Software Process Technology

C. Foix 04

CEPIS

<http://www.cepis.org>

CEPIS, Council of European Professional Informatics Societies, is a non-profit organisation seeking to improve and promote high standards among informatics professionals in recognition of the impact that informatics has on employment, business and society.

CEPIS unites 36 professional informatics societies over 32 European countries, representing more than 200,000 ICT professionals.

CEPIS promotes

EUCIP
European Certification of
Informatics Professionals

ECDL ®

CEPIS UPGRADE
The European Journal for the Informatics Professional
http://www.upgrade-cepis.org

<http://www.eucip.com> <http://www.ecdl.com> <http://www.upgrade-cepis.org>

UPGRADE
The European Journal for the Informatics Professional
http://www.upgrade-cepis.org

Vol. V, No. 5, October 2004

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

UPGRADE is the anchor point for **UP**ENET (**UP**GRADE 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
- Pliroforiki, 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/>).

Cover page designed by Antonio Crespo Foix, © ATI 2004
Layout: Pascale Schürmann
Editorial correspondence: see "Editorial Team" above
Advertising correspondence: <novatica@ati.es>
Upgrade Newslist available at <http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

ISSN 1684-5285

Next issue (December 2004): "Cryptography"

(The full schedule of **UP**GRADE is available at our website)

SPT, Software Process Technology

Guest Editors: Francisco Ruiz-González and Gerardo Canfora

Joint monograph with Novática*

MOSAIC

UPENET (**UP**GRADE European NETwork)

* 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>.

Editorial

Four Years of UPGRADE

Dear Readers,

It was four years ago (October 2000) when **UP**GRADE, “The European Journal for the Informatics Professional”, was first launched. An initiative of the Council of European Professional Informatics Societies (CEPIS), our digital journal was then published, on its behalf, by INFORMATIK/INFORMATIQUE and NOVÁTICA, the journals of the CEPIS societies SVI/FSI (Switzerland) and ATI (Spain), respectively.

Bearing in mind that the principal objective of our bimonthly digital journal *(to provide its readers with high quality contents, ranging from practical business experiences to academic research)* is, by definition, a dynamic one, twenty four issues later we can say that this European undertaking, **UP**GRADE, has a number of important achievements of which to be proud:

- Since its inception, **UP**GRADE has had a Spanish printed edition, published by NOVÁTICA, and, since December 2001, an Italian digital edition, promoted jointly by the Italian CEPIS society ALSI and the Italian IT portal Tecnoteca.
- The readership of our digital journal has increased steadily, with a current monthly average of 10,000 unique visits and 45,000 PDF downloads from countries all over the world, albeit mostly from Europe.
- The **UP**GRADE newsletter, created in 2003, now has 1,118 subscribers.
- **UP**GRADE is ranked number 1 on Google, MSN Search, Lycos, Yahoo, Hotbot and All the Web, and number 2 on AOL, if you search for the string *“European informatics journal”*.
- In 2004 **UP**GRADE was admitted to the ACM Guide to Computing Literature and to ACM Computing Reviews, being now included in many important ICT journal listings.
- In February 2004 a section named MOSAIC was created in order to expand the contents of our digital journal from its original monographic approach to a broader one, including articles covering any ICT field, always subject to peer review procedures. In order to manage this new section, two new members joined **UP**GRADE’s Editorial Team.
- In April 2004 **UP**ENET (**UP**GRADE European NETwork) was born. **UP**ENET is the network of CEPIS publications, whether paper or electronic.

Its current members are **Mondo Digitale**, published by the Italian CEPIS society AICA; the abovementioned **NOVÁTICA**; **Pliroforiki**, published by the Cyprus CEPIS society CCS; and **Pro Dialog**, journal co-published in Polish and English by the Polish CEPIS society PTI-PIPS and the Poznan University of Technology – Institute of Computing Science. Other journals belonging to CEPIS societies have expressed a serious interest in joining this network soon.

Although there is still a great deal of room for improvement and expansion, we can see from all the above that UPGRADE is now a valuable CEPIS asset and a solid reality in the worldwide landscape of scientific journals in the ICT field, a reality we would like to celebrate.

Thanks should be given to all those who make this reality possible, namely authors, reviewers, members of the Editorial Team and other personnel in charge of daily operations, CEPIS societies involved in this initiative (especially ATI), and, last but not least: you, our readers (ICT professionals, academics and students in Europe and elsewhere).

Happy birthday, **UP**GRADE!

Jouko Ruissalo
President of CEPIS
<jouko.ruissalo@ttlry.fi>

Ps. At the closing of this issue we have received the good news that the **OCG Journal**, published in German by the Austrian CEPIS society OCG, has just joined **UP**ENET, as approved by the Board of this society. *Willkommen!*

CEPIS UPENET

UPGRADE European NETwork

The network of CEPIS member societies’ publications

Presentation

Software Process Technology: Improving Software Project Management and Product Quality

Francisco Ruiz-González and Gerardo Canfora

1 Introduction

One of the main lines of work on the enhancement of software product quality is the study and improvement of the processes by which software is developed and maintained. This statement is based on the assumption that there is a direct correlation between a quality process and product quality. The area of study in the field of software engineering addressing this problem is known as "Software Process Technology" (SPT), or simply "Software Process" (SP).

Research into SPT as a separate discipline began in the 80s *(International Software Process Workshop, European Workshop on Software Process Technology, Journal of Software Process Technology: Improvement and Practice,...)*, but it is only in the last 5 or 6 years that it has acquired a certain maturity in terms of its real use in software engineering projects. The first important contribution of SPT was to confirm that the development and maintenance of software are complex processes which require a collective and creative effort. Thus the quality of a software product is heavily dependent on the people, the organisation and the procedures involved in creating, delivering and maintaining it.

2 The Contents of This Monograph

This monograph issue opens with the article *"Software Process: Characteristics, Technology and Environments"* which the authors of this presentation have written as an introduction to the topic. It deals with three essential aspects: software process specific characteristics; the justification of SPT as a way of providing integrated support to both production and management processes; and Software Engineering Environments (SEEs). In the last point, we stress the different dimensions of software tool integration within an SEE and the proposed process orientation for SEEs (Process-centred Software Engineering Environment, PSEE).

"Key Issues and New Challenges in Software Process Technology" was written by ***Jean-Claude Derniame*** and ***Flavio Oquendo*** (both have played major roles in the EWSPT – European Workshop on Software Process Technology – series of conferences). It is an analysis of the evolution and results of this field over its twenty years of existence and the key unresolved challenges SPT has today: the support of typical agile processes, open source software, and worldwide software development (globalization). The first part is an introduction to SPT – which complements this introductory article – including a generic process framework and the relationship between SPT and process maturity.

The Guest Editors

Francisco Ruiz-González has a PhD in Computer Science from the *Universidad de Castilla-La Mancha* (UCLM), Spain, and an MSc in Chemistry-Physics from the *Universidad Complutense de Madrid*, Spain. He is a full time Associate Professor of the Dept. of Computer Science at UCLM in Ciudad Real, Spain. He was the Dean of the Faculty of Computer Science between 1993 and 2000. Previously, he was the Director of Computer Services at the aforementioned university (1985-1989) and he has also worked in private companies as an analyst-programmer and project manager. He is a member of the Alarcos Research Group, <http://alarcos.inf-cr.uclm.es/english/>. His current research interests include software process technology and modelling, software maintenance, and methodologies for software projects planning and management. He has also worked in the fields of GIS (Geographical Information Systems), educational software systems and deductive databases. He has written eight books and fourteen chapters on the abovementioned topics and he has published 90 papers in Spanish and international journals and conferences. He has sat on nine programme committees and seven organizing committees and he belongs to several scientific and professional associations: ACM, IEEE-CS, ATI, AEC, AENOR, ISO JTC1/SC7, EASST, AENUI and ACTA. <Francisco.RuizG@uclm.es>

Gerardo Canfora is a full Professor of Computer Science at the Faculty of Engineering and the Director of the Research Centre on Software Technology (RCOST) of the *Università degli Studi del Sannio*, Benevento, Italy. He served on the programme and organizing committees of a number of international conferences. He was a programme co-chair of the 1997 International Workshop on Program Comprehension, the 2001 International Conference on Software Maintenance, and the 2004 European Conference on Software Maintenance and Reengineering. In 2003 he was the general chair of the European Conference on Software Maintenance and Reengineering. His research interests include software maintenance and evolution, programme comprehension and reverse engineering, software process improvement, knowledge management, and service oriented software engineering. On these topics he has published more than 100 articles in international journals and conferences. He is an associate editor of the IEEE Transactions on Software Engineering and serves on the editorial board of the Journal of Software Maintenance and Evolution: Research and Practice. <canfora@unisannio.it>

As a demonstration of the industrial maturity that SPT is reaching, *"A Taxonomy of Software Engineering Environment Services: The Upcoming ISO/IEC Standard 15940"*, by ***Dan Hyung Lee*** and ***Juan Garbajosa-Sopeña***, presents us the future ISO standard of which they are co-editors. The authors classify, enumerate and define all the possible services that a SEE can provide to give automatic support to the various processes of software life cycle.

The close relationship between business models used by the software industry and processes that are carried out during software development and maintenance makes the reflections, analyses and explanations put forward by ***Alfonso Fugetta*** in *"Open Source and Free Software: A New Model for The Software Development Process?"* highly interesting, useful and illuminating.

The next two articles refer to SP modelling. In *"Applying The Basic Principles of Model Engineering to The Field of Process Engineering"*, ***Jean Bézivin*** and ***Erwan Breton*** introduce model-driven system engineering, presenting the MDA (Model-Driven Architecture) proposed by OMG (Object Management Group) whose aim is to separate platform-independent aspects from the platform-dependent aspects in the design of software system architecture. As proof of the strength of this new "model-driven" development paradigm, the authors show how it can be applied to processes using a non-MDA based COTS (Commercial Off-The-Shelf) software, MS-Project, demonstrating that this approach enables us to design and build more general solutions.

In *"Software Process Modelling Languages Based on UML"*, ***Pere Botella i López, Xavier Franch-Gutiérrez*** and ***Josep M. Ribó-Balust*** introduce the reader to Process Modelling Languages (PMLs). In particular they analyse the possibilities of UML (Unified Modelling Language) to model the structural and behavioural aspects of processes, and present two PMLs, namely SPEM and PROMENADE, that take advantage of this notation to model software processes.

Focusing on another interest point, the next two articles are devoted to technological aspects of SEEs. In *"Supporting the Software Process in A Process-centred Software Engineering Environment"*, ***Hans-Ulrich Kobialka*** carries out a systematic study of the process support requirements a PSEE should satisfy, and proposes a list of ingredients (groups of services) to this end. The author presents the mechanisms available in LMP ALADYN for process automation (triggers, task patterns, constraints, etc.) and impact control (permissions).

The article *"Managing Distributed Projects in GENESIS"* was written by ***Lerina Aversano, Andrea De Lucia, Matteo Gaeta, Pierluigi Ritrovato***, and ***Maria-Luisa Villani***. They propose an approach and an environment to support the management of distributed software projects allowing the definition and enactment of software process models in a decentralized and autonomous multi-site manner.

In *"Software Process Measurement"*, ***Félix García-Rubio, Francisco Ruiz-González***, and ***Mario Piattini-Velthuis***, argue the importance of measuring SPs to be able to carry out evaluation and improvement. The authors identify the measurable entities of a SP and, as a use case, they present a set of metrics that can be used to estimate the maintainability of a process model.

It is usual for a process to pass through various adaptations due to the different operational contexts in which the process is performed. These adaptations involve the creation of distinct versions from the same generic process which are known as specialized processes. In *"Process Diversity and how Practicioners Can Manage It"*, ***Danilo Caivano*** and ***Corrado Aaron Visaggio*** present a framework based on process patterns to manage and to maintain all these different process models. The application of this framework to software system maintenance is also included as a case study.

We hope this collection of articles (thanks must go the authors for their valuable contributions) provides an introduction to and an overview of Software Process Technology. We believe that, by means of automation and the integration of various engineering and managerial processes, this field can be a major help to software engineers in years to come.

Useful References on SPT

These references, additional to the ones included in the papers this monography consists of, enlarge the field of Software Process Technology for readers interested in knowing more about this matter.

Associations

- ISPA – International Software Process Association. <http://www.ece.utexas.edu/~perry/prof/ispa/>.

Books

- A. Fuggetta and A. Wolf (eds). Software Process, Vol 4, Trends in Software. J. Wiley & Sons (USA), 1996. <http://serl.cs.colorado.edu/~alw/doc/Trends.html>.
- J. C. Derniame, B. A. Kaba, & D. Wastell (eds.). Software Process: Principles, Methodology and Technology. Springer-Verlag, serie LNCS nº 1500, 1999.
- B. Westfechtel. Models and Tools for Managing Development Processes. Springer-Verlag, serie LNCS nº 1646, 1999.

Journals

- Journal of Software Process Improvement and Practice. Quarterly journal published by Wiley. <http://www.wiley.com/WileyCDA/WileyTitle/productCd-SPIP.html>.

Conferences & Events

- EWSPT – European Workshop on Software Process Technology. Scientific biennial workshop whose ninth edition took place in Helsinki, Finland, in September 2003. Proceedings: <http://www2.springeronline.com/sgw/cda/frontpage/0,11855,5-40109-22-7063147-0,00.html>.
- ICSPI – International Conference on Software Process Improvement. Its second edition was held in Washington DC, USA, in June 2004. <http://www.icspi.com/>.
- PROFES – International Conference on Product Focused Software Process Improvement. Its fifth edition was held in Kansai Science City, Japan, in April 2004. <http://www.vtt.fi/ele/profes2004/>.

Web Sites

- Capability Maturity Models. University of Carnegie Mellon, Software Engineering Institute (EE.UU.). <http://www.sei.cmu.edu/cmm/cmms/cmms.html>.
- Graphical Development Process Assistant. University of Bremen, Germany. <http://www.informatik.uni-bremen.de/uniform/vm97/>. Specialized web site that includes over 6,000 pages on this topic, including concepts and definitions, process models, environments, standards, methodologies, process elements, SPT modelling classes and approaches, projects, tools and classified bibliographic about all the above.

Papers

- V. Ambriola, R. Conradi, and A. Fuggetta. Assessing Process-centered Software Engineering Environments. ACM Transactions on Software Engineering and Methodology, 6:3, 1997, pp. 283–328.
- G. Cugola and C. Ghezzi. Software Processes: a Retrospective and a Path to the Future. Software Process: Improvement and Practice, 4(3), 1998, pp. 101–123.
- B. Curtis, M. I. Kellner, and J. Over. Process Modeling. Communications of the ACM, 35(2), September 1992, pp. 75–90.
- L. Osterweil. Software Processes are Software Too. Proc. of 9th International Conference on Software Engineering. Washington DC (USA). IEEE Comp. Soc. Press, 1987.

Translation by ***Steve Turpin***

Software Process: Characteristics, Technology and Environments

Francisco Ruiz-González and Gerardo Canfora

In this introductory article we present the concept of Software Process (SP) and the properties that characterize and distinguish this process from other types of processes (e.g. typical industrial production processes). We go on to justify the interest in having Software Process Technology (SPT) to enable us to automate and to integrate production and management processes in order to carry out software projects. We also present Software Engineering Environments (SEE), collections of integrated tools whose purpose is to provide support to the abovementioned processes. We conclude by summarising the problem of how to integrate the tools making up an environment and how to create a process-oriented environment.

Keywords: Software Engineering Environment, Software Process, Software Process Technology, Process-Orientation, Tools Integration.

1 Characteristics of Software Processes

The definition of *Software Process* (SP) complements the concept of software life-cycle in the sense that it defines the skeleton and philosophy for carrying out an SP, but it is not in itself sufficient to guide and control a development and/or maintenance project. An SP is a "*coherent set of policies, organizational structures, technologies, procedures and artifacts that are needed to conceive, develop, deploy, and maintain a software product*" [3].

The special nature of SPs can be defined as follows:

a) They are complex.

b) They are not typical production processes, since they are exception driven, are strongly affected by highly unpredictable circumstances, and each process has peculiarities that distinguishes it from all others.

c) They are not 'pure' engineering processes either, since we do not know the appropriate abstractions, (there is no experimental science behind them), they depend too much on too many people, their design and production are not clearly differentiated, and their budgets, schedules and quality cannot be programmed reliably enough.

d) They are not (entirely) creative processes because some parts can be described in detail while some procedures are previously enforced.

e) They are finding-based and depend on communication, coordination and cooperation within predefined frameworks. Their delivery generates new requirements, the cost of changing software is not usually recognised, and their success depends on user involvement and the coordination of many different roles (sales, technical development, customer, etc.).

The necessity for creative human participation and the absence of repetitive actions means that neither the development nor the maintenance of the software is a production process. However, there are some similarities between the two

Francisco Ruiz-González has a PhD in Computer Science from the *Universidad de Castilla-La Mancha* (UCLM), Spain, and an MSc in Chemistry-Physics from the *Universidad Complutense de Madrid*, Spain. He is a full time Associate Professor of the Dept. of Computer Science at UCLM in Ciudad Real, Spain. He was the Dean of the Faculty of Computer Science between 1993 and 2000. Previously, he was the Director of Computer Services at the aforementioned university (1985–1989) and he has also worked in private companies as an analyst-programmer and project manager. He is a member of the Alarcos Research Group, <http://alarcos.inf-cr.uclm.es/english/>. His current research interests include software process technology and modelling, software maintenance, and methodologies for software projects planning and management. He has also worked in the fields of GIS (Geographical Information Systems), educational software systems and deductive databases. He has written eight books and fourteen chapters on the abovementioned topics and he has published 90 papers in Spanish and international journals and conferences. He has sat on nine programme committees and seven organizing committees and he belongs to several scientific and professional associations: ACM, IEEE-CS, ATI, AEC, AENOR, ISO JTC1/SC7, EASST, AENUI and ACTA.
<Francisco.RuizG@uclm.es>

Gerardo Canfora is a full Professor of Computer Science at the Faculty of Engineering and the Director of the Research Centre on Software Technology (RCOST) of the *Università degli Studi del Sannio*, Benevento, Italy. He served on the programme and organizing committees of a number of international conferences. He was a programme co-chair of the 1997 International Workshop on Program Comprehension, the 2001 International Conference on Software Maintenance, and the 2004 European Conference on Software Maintenance and Reengineering. In 2003 he was the general chair of the European Conference on Software Maintenance and Reengineering. His research interests include software maintenance and evolution, programme comprehension and reverse engineering, software process improvement, knowledge management, and service oriented software engineering. On these topics he has published more than 100 articles in international journals and conferences. He is an associate editor of the IEEE Transactions on Software Engineering and serves on the editorial board of the Journal of Software Maintenance and Evolution: Research and Practice. <canfora@unisannio.it>

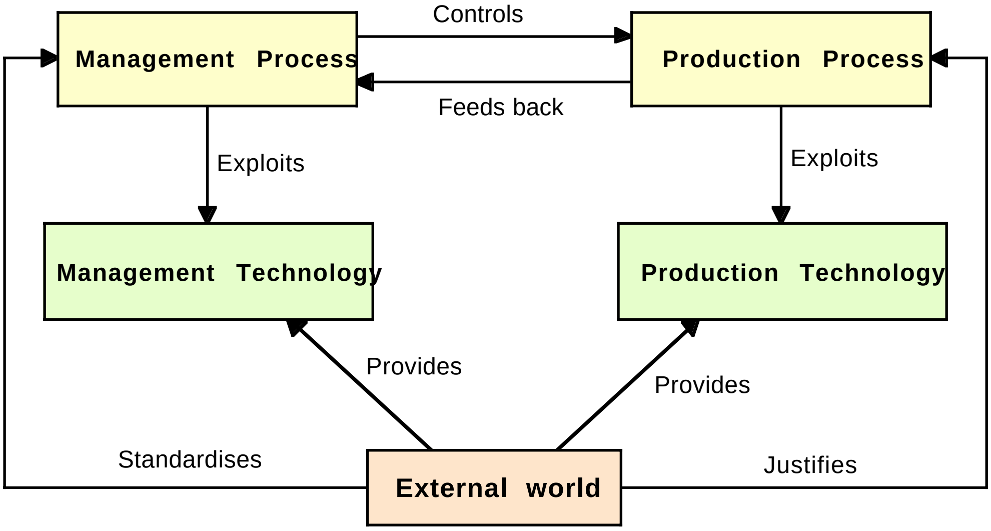


Figure 1: Production Process vs Management Process.

kinds of processes which are useful for understanding software processes in a broader perspective. Like production processes, software processes consist of two inter-related sub-processes; the production process and the management process [5]. The production process relates to the actual 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 about its situation to the management process. These relationships are represented in Figure 1, as are the relationships between processes and the external environment. The request for a product comes from outside (the external world); in other words, the external environment justifies the production process's existence. Management also has to conform to the current standards of the external environment; in other words, the external environment also has an indirect influence on the production process. In short, production and management processes make use of technologies originating from the external environment.

2 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-centred Software Engineering Environment* (PSEE), which provides support to management and production processes in an integrated manner. Figure 2 shows the impact of this new technology and how PSEE implements, controls and improves the flow of information by which the management process controls the production process. The main objective of SPT is to control the inherent complexity of the SP via an in depth understanding of the process itself, and by means of the automated support provided by a PSEE. An essential factor in achieving this objective is computerized support; in other words, having a process model and the proper means of defining, modifying, analysing and enacting it [1].

Following on from the previous definition, SPT makes use of a wide range of areas and concepts:

1. **Software development and maintenance technologies** which provide the necessary tools and infrastructure to make it possible and economically feasible to create and maintain complex software products that meet present and future needs.
2. **Methods and techniques** for software development and maintenance which provide the essential methodological support required to make efficient use of the abovementioned technologies and successfully perform development and maintenance activities.
3. **Organizational behaviour**, in other words, the science of organizations and people, is useful because software projects are generally carried out by groups that need to be coordinated and directed within an effective organizational structure.
4. **Marketing and economy**, since software development and maintenance projects are not executed in isolation, and, as is the case with any other product, software must be oriented towards the needs of real customers and users.

To sum up, when developing and maintaining software it is necessary to pay attention to the complex relationships that arise between the various organizational, cultural, technological and economical factors.

3 Software Engineering Environments

Although the use of tools in helping developers to produce software has existed in one way or another since the early days of computing, the concept of *Software Engineering Environment* (SEE), is a relatively recent development. SEE is defined as "*a set of tools providing full or partial automated support to software engineering activities*". Normally these activities are carried out within the framework of a software project and refer to aspects such as specification, development, reengineering and maintenance of software systems. SEE has

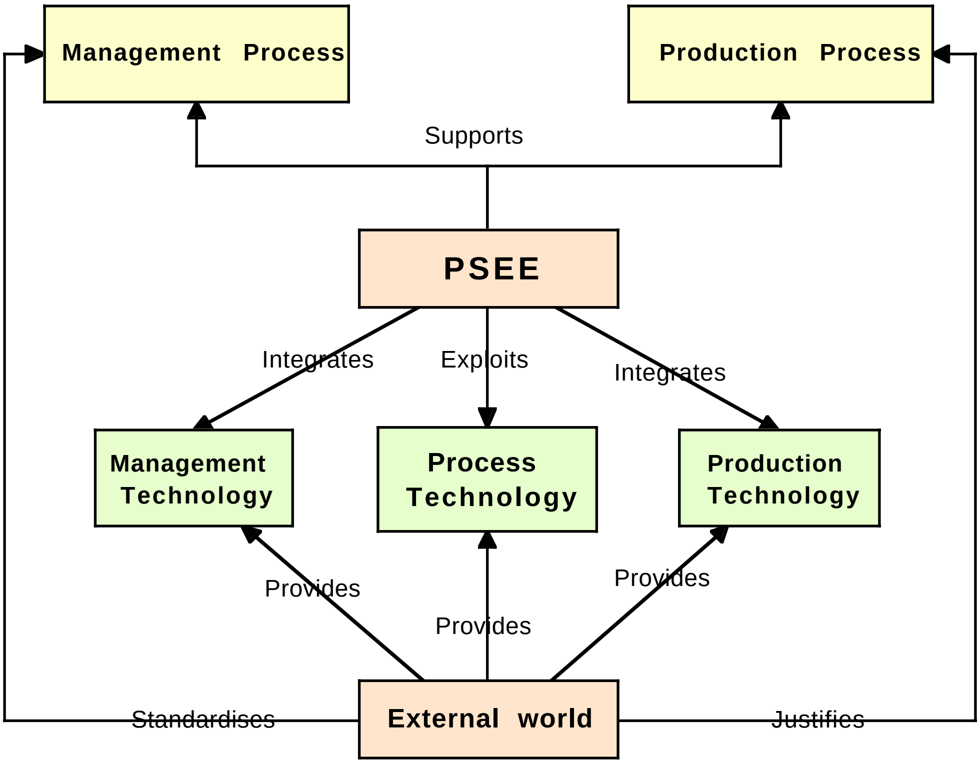


Figure 2: Impact of Software Process Technology.

also been known by other names: IPSE (*Integrated Project Support Environment*), ISEE (*Integrated Software Engineering Environment*), *CASE tools coalition*, *Federated CASE tools*, or ISF (*Integrated Software Factory*).

The term SEE can be applied to a wide range of systems: from a set of a few tools running on the same system, to a completely integrated environment able to manage and control all data, processes and activities of a software product life-cycle. Thanks to the automation of activities (partial or total), SEE can produce significant benefits for an organization: reduction in costs (high productivity), improved management and improved quality of the end product. For example, the automation of repetitive activities, such as the performance of test cases, not only improves productivity but also helps to ensure the completeness and consistency of testing activities.

SEE normally deals with information related to:

a) Software under development or maintenance (specifications, design information, source code, tests data, project plans...)
b) Project resources (costs, computing resources, personnel, responsibilities and duties...)
c) Organizational aspects (organizational policies, standards and methods used...)

SEE gives support to human activities by means of a set of services that describe the environment's capabilities. These services provide the correspondence between a chosen set of software life-cycle related processes and their automation by the use of tools. In most cases a tool's functionality is related to one or more services.

Interest was first aroused in SEE in the early 90s when the first reference models were proposed and the first taxonomy of the services to be included was produced [8]. But it was not until the beginning of the 21st century that environments were developed which actually tried to meet the ambitious objectives that the definition of SEEs entails [6].

3.1 Integration

The concept that most differentiates an SEE from a simple set of tools working on a computer under the same operating system, is the degree of integration that it provides. The concept of integration applied to an SEE can mean many related but different things:

- The degree to which different tools can communicate effectively between one another within the framework of an SEE.
- A measure of the relationships between SEE components.
- The simplicity, interoperability, portability, scalability, productivity, etc., produced by the seamless interaction of SEE components.

Sharing the same object management system (repository manager) instead of having a separate file system for each tool is an important feature of integration, but not the only one. SEEs must have a series of interfaces enabling cooperation between tools from different manufacturers. Integration can be said to involve the three following aspects:

- **A set of services.** Most of the services described later are applicable to integration. For example, using a common object management system with common schemes enables the tools to share objects; using global presentation charac-

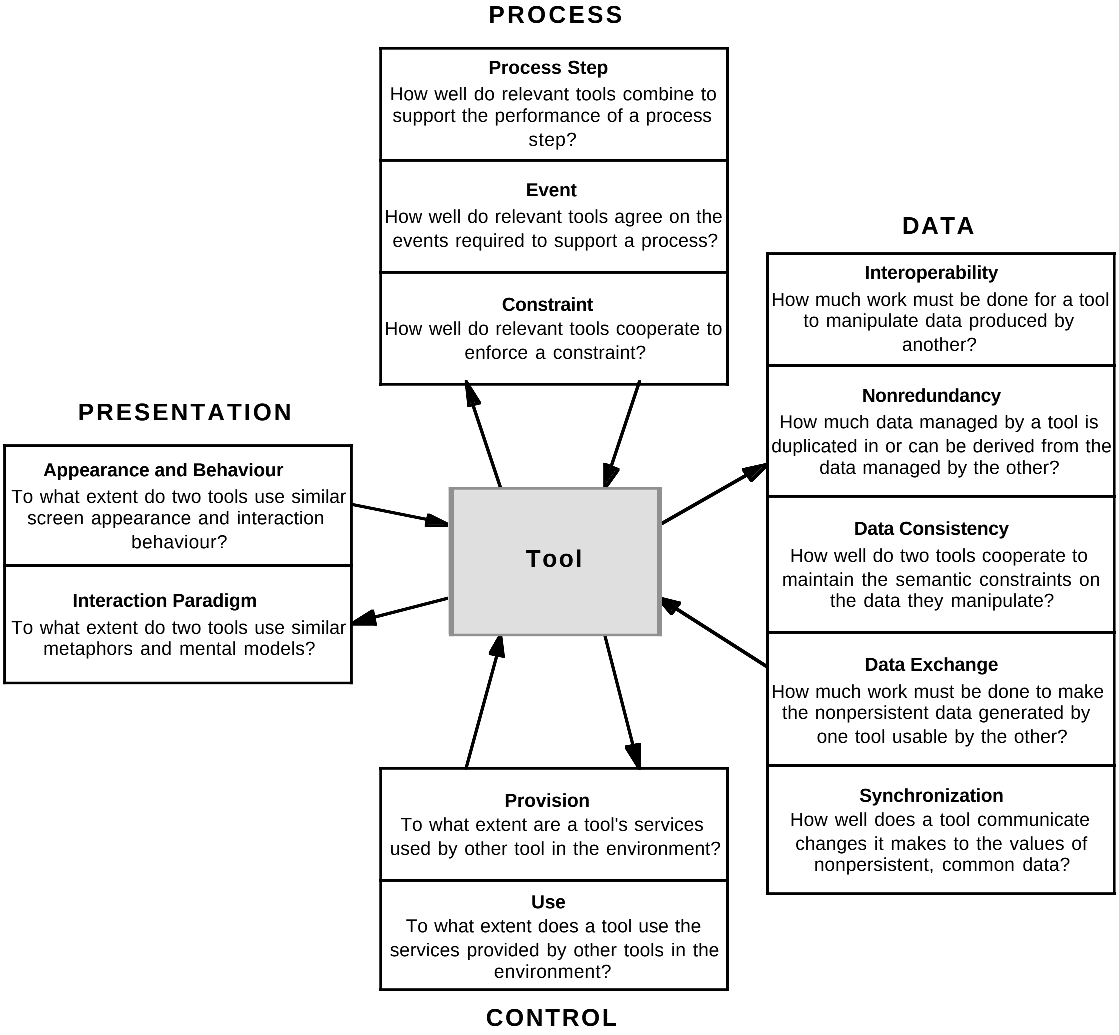

PROCESS
Process Step
How well do relevant tools combine to support the performance of a process step?
Event
How well do relevant tools agree on the events required to support a process?
Constraint
How well do relevant tools cooperate to enforce a constraint?
PRESENTATION
Appearance and Behaviour
To what extent do two tools use similar screen appearance and interaction behaviour?
Interaction Paradigm
To what extent do two tools use similar metaphors and mental models?
Tool
DATA
Interoperability
How much work must be done for a tool to manipulate data produced by another?
Nonredundancy
How much data managed by a tool is duplicated in or can be derived from the data managed by the other?
Data Consistency
How well do two tools cooperate to maintain the semantic constraints on the data they manipulate?
Data Exchange
How much work must be done to make the nonpersistent data generated by one tool usable by the other?
Synchronization
How well does a tool communicate changes it makes to the values of nonpersistent, common data?
Provision
To what extent are a tool's services used by other tool in the environment?
Use
To what extent does a tool use the services provided by other tools in the environment?
CONTROL

Figure 3: Properties of Tool Integration in SEE.

teristics in the user interface enables all tools to have a similar "visual aspect"; and management process and communication services are necessary for the tools to communicate with each other.

- **A new dimension for each service.** Having common services permits integration but does not make it obligatory (tool developers are not obliged to use them). This new dimension indicates the degree to which a service can help to increase integration.
- **A policy.** It is also necessary to implement policies so that developers of tools, frameworks and platforms can use integration services efficiently. An example of this are the "style guides" intended for tool constructors.

According to [7], the need for integration in an EIS involves several different dimensions (see Figure 3):

- **Data.** Data integration is the ability to share information in the SEE. The degree of data integration can be high (tools use a common database with a common scheme), medium (common data formats) or low (uses transformation mechanisms). Another characteristic that data integration can include is composition.
- **Control.** Control integration is the ability to flexibly combine the functionalities provided in an environment. These combinations may correspond to project preferences and be driven by the underlying software processes.
- **Presentation.** Presentation integration is the ability to interact with environment functionalities with similar screen appearance and similar modes of interaction.
- **Processes.** Process integration is the ability to access environment functionalities using a pre-defined SP enacted with automated support.

3.2 Process Orientation

We have already mentioned the importance which process-oriented SEEs (or PSEE)s have in SPT. In fact, the principal role of an SEE is to provide support to order to enact the SP effectively. This point of view is gaining ground as software development and maintenance processes have become intellectual activities of an ever more complex and laborious nature, with a great potential for improvement to quality and productivity, based on discipline, management, and the help of PSEEs and other computing technologies. Many organizations have problems defining and performing the steps which transform user requirements into a software product in such a way that they are replicable, measurable in terms of quality objectives, and adaptable or improvable, so the use of an SEE to implement a defined process during the performance of a software project can provide considerable short term benefits.

In a PSEE, process management services contribute to the effective support of SPs, providing end-user oriented facilities in order to define and use processes that can replace the undisciplined, difficult to control, and tedious invocation of individual tools. Garg and Jazayeri [4] believe that process support in PSEE is based on the following functionalities:

- *Process definition.* A process engineer 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 analysed to verify its consistency, completeness, and correctness.
- *Process presentation.* A PSEE includes support for the graphical display of SPs (activity flows) and 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 to change its process definitions without interrupting its work.
- *Openness.* The PSEE provides tools to exchange data and metadata with other non-integrated tools or with other PSEEs.
- *Multi-User support.* Typically, software engineering projects are worked on by teams of people with different roles. The PSEE must therefore provide services to all the people working together on a process.
- *Process guidance.* Software engineers use the PSEE to carry out various process steps. The PSEE must provide help in choosing among possible next steps based on the modelled process and the current state.
- *Task-specific user interface.* Based on the modelled process, the PSEE can scope the user interface to the needs of each task, thereby preventing an excess of information from being presented to the user.

It is becoming ever more common for software product development and maintenance to be carried out with the collaboration of various companies or organisations. As a result, in recent years there has been increasing interest in the study of the problems that can arise when several separate and distinct PSEEs are required to collaborate with one another, and, more specifically, in the need for interoperability between the processes supported by those PSEEs. Among the various proposals made to address this problem, perhaps the most interesting is that put forward by a number of authors who, using the metaphor of international alliances, support the idea of forming federations of PSEEs, whereby each organization would manage its own processes (just as each country has its own laws), and inter-organizational processes would act in a similar way to international treaties between countries. In the relevant bibliography, two types of conceptual architectures have been proposed for PSEE federations: control based, favouring centralization given the existence of common process models; and state based in which there is a workspace in which the common state is stored [2].

4 Conclusions

In this article we have presented the key aspects of Software Process Technology: the object of interest (software process and their characteristics); the interest in and justification for giving automatic support to these processes; and the requirements, functionality and characteristics of integration and process-orientation that the collection of tools need to have to achieve these goals.

References

[1] J. C. Derniame, B. A. Kaba, and D. Wastell, D. (eds.). Software Process: Principles, Methodology and Technology. LNCS 1500, Springer-Verlag, 1999.

[2] J. Estublier, P. Y. Cunin, and N. Belkhatir. Architectures for Process Support System Interoperability. Proceedings of the Fifth International Conference on the Software Process (ICSP'98), 15–17 Junio, Chicago (Estados Unidos), pp. 137–147, 1998.

[3] A. Fuggetta. Software Process: A Roadmap. 22nd International Conference on Software Engineering (ICSE'2000), Future of Software Engineering Track, June 4–11, Limerick (Irlanda), ACM, 2000.

[4] P. K. Garg and M. Jazayeri. Process-centred Software Engineering Environments: A Grand Tour. In Fuggetta, A. y Wolf, A. (eds.); Software Process. John Wiley & Sons, 1996.

[5] R. McLeod Jr. Management Information Systems. McMillan Publishing, New York, 1990.

[6] H. Ossher, W. Harrison, and P. Tarr. Software Engineering Tools and Environments: a Roadmap. International Conference on Software Engineering (ICSE) – Future of SE Track. Limerick (Irlanda), pp. 261–277, 2000.

[7] I. Thomas and B.A. Nejmeh. Definitions of Tool Integration for Environments. IEEE Software, 9(2), pp. 29–35, 1992.

[8] M. V. Zelkowitz. Software Engineering Environment Capabilities. Journal of Systems and Software. Elsevier Science, 35(1), pp. 3–14, 1996.

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.

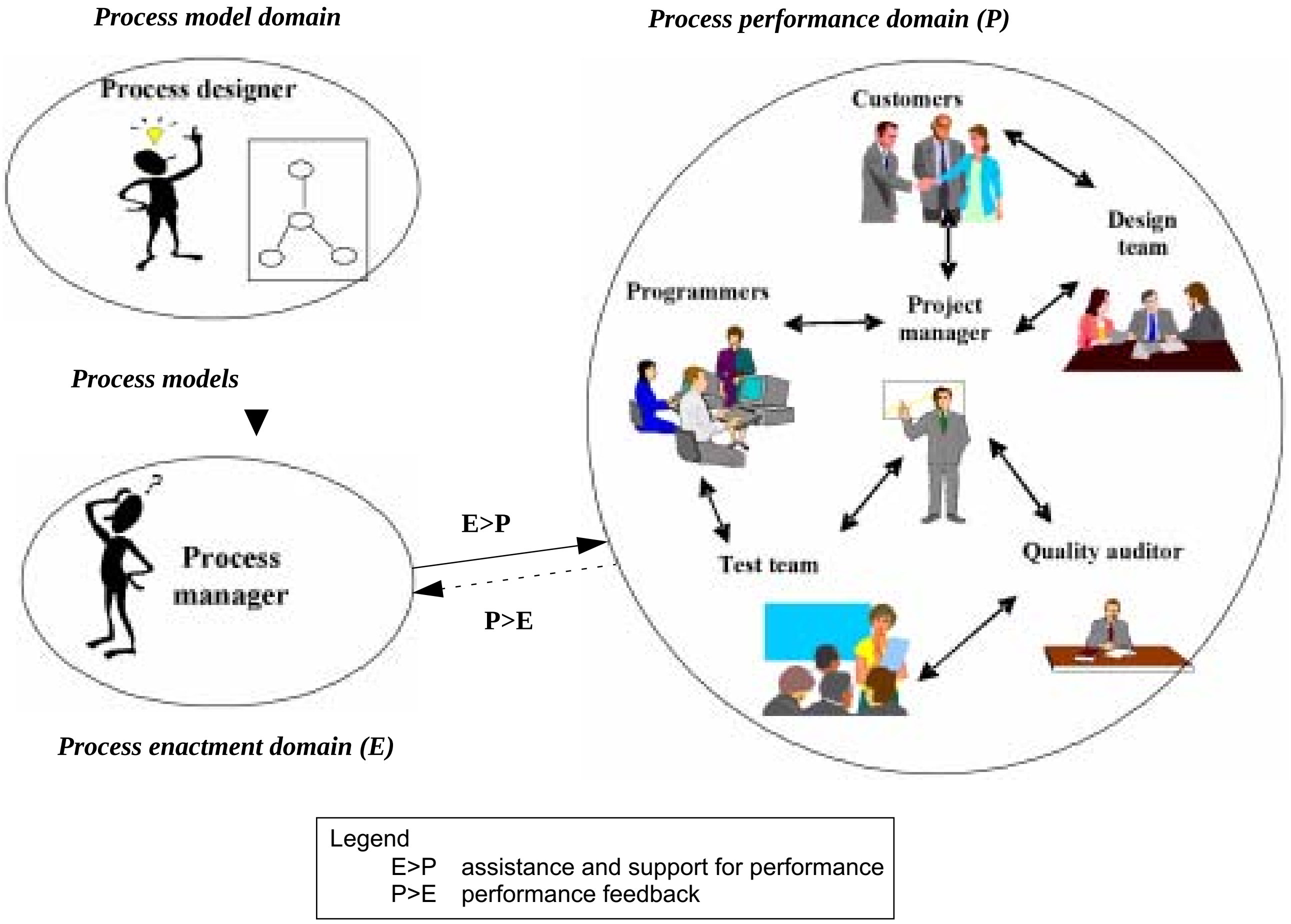


Figure 1: Software Process Domains.

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 computerizes 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.

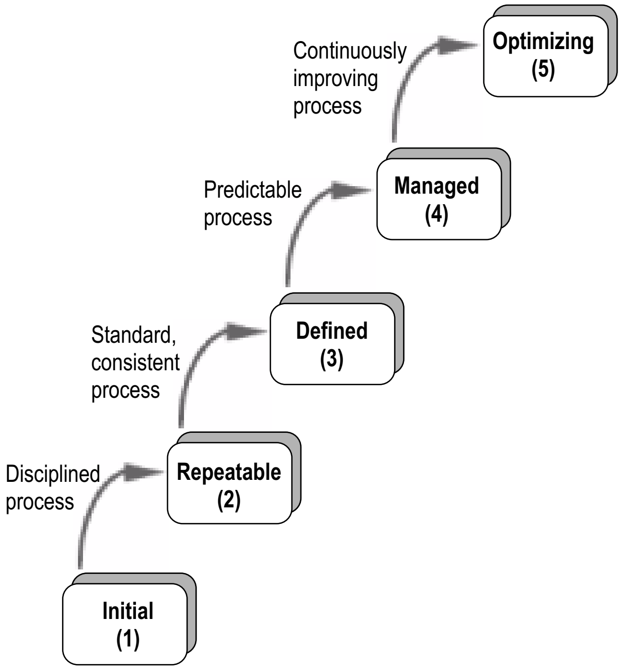


Figure 2: The Five Levels of Software Process Maturity.

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 organization. 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 methodologies[1] 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 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".

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

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.

A Taxonomy of Software Engineering Environment Services: The Upcoming ISO/IEC Standard 15940

Dan Hyung Lee and Juan Garbajosa-Sopeña

This paper introduces the upcoming ISO/IEC software engineering standard 15940 – Software Engineering Environment Services – and describes its motivation, background and basic guidelines. This standard presents the set of services to be provided to the different kinds of users in software engineering environments from the point of view of software lifecycle processes.

Keywords: Assisted Software Process, CASE tools, ISO/IEC 12207, ISO/IEC 14102, Software Development Environment, Software Lifecycle Processes, Software Process Automation

1 Introduction

The automation of software engineering lifecycle processes is still an open issue for the software engineering community. For many years the main obstacle to achieving automation was seen as a basically technical problem, focused mainly on tool integration. That was the case in the early 90s when platforms such as PCTE (Portable Common Tool Environment) [1] were seen as part of the solution to the problem, the goal of which was to achieve integrated environments via reference models. A number of opinions refuting these commonly accepted approaches can be found in [2]. A reference model published jointly by the European Computer Manufacturers Association (ECMA, <http://www.ecma-international.org/>) and the US National Institute for Standards and Technology (NIST, <http://www.nist.gov>) was released in 1993 [3]. While this report is well structured and comprehensive, the services included are mainly of a technical nature and do not address many of the end-user services, reflecting the extent to which technical requirements were considered as the main concern. Later, end user services were addressed in [4].

Meanwhile further progress was being made in the definition of lifecycle processes, with ISO/IEC 12207 [5] being published in 1995. Reference [5] describes a comprehensive set of processes, activities and tasks to be performed when acquiring or developing software. While it does not address their implementation or automation it has in all probability been one of the standards which has had the greatest impact on the software community, with the exception of the ISO 9000 series. It covers areas such as the specification, development, re-engineering, acquisition, supply, or maintenance of software based systems. During the 90s, lifecycle processes gained an ever greater importance within the community, while process maturity and improvement were also considered.

Software Engineering Environments (SEE) are intended to provided at least partially automated support to software lifecycle processes. But the term SEE is used to denote a wide range of entities: from a juxtaposition of tools running under the same operating system to a fully integrated environment, able to control all data, processes and activities in the software lifecycle.

For a user interested in a specific process, there is no clear vision of what services an SEE might provide, what the relationships between those services might be, or how an SEE relates to the software engineering life cycle as a whole. For this reason, it is difficult to make a proper assessment of products that claim to be an SEE. It is frequently difficult to understand the role a software engineering tool might play in an SEE without an overall view of what an SEE is. It is difficult for an organization to achieve the proper level of automation in its process improvement efforts without a well-defined set of software engineering environment services. This problem can be resolved by generating a comprehensive, objective description

Dan H. Lee is currently the Director of the Software Technology Institute and a Professor at the Information and Communications University, Seoul, Korea. He is a WG4 tools and environment convenor for the ISO/IEC JTC1 SC7 Software and Systems Engineering Subcommittee. Before joining the Information and Communications University, Seoul, he was president and CEO of the Korea IT Industry Promotion Agency, Sr. Executive Vice President and Chief Technology Officer at LG-EDS Systems, and Executive Vice President at Systems Engineering Research Institute. <danlee@icu.ac.kr>

Juan Garbajosa-Sopeña is a Lecturer at Universidad *Politécnica de Madrid* (UPM), Spain, where he teaches courses in software engineering and database administration, and system integration and validation. On the ISO/IEC JTC1 SC7 Software and Systems Engineering Subcommittee he is co-editor of project 15940 Software Engineering Environment Services and of a number of other projects related to tool standardisation, and he is a convenor of the WG20 Software Engineering Body of Knowledge. Before joining UPM he spent more than 15 years in industry. He is affiliated to IEEE, ACM, ATI, and Ada-Spain. His current research interests are in tools and environments supporting complex systems development and operation, including systems and process modelling and tool design and construction. He is also extremely interested in system testing and validation processes. <jgs@eui.upm.es>

of the services that make up an SEE. In the light of all the above, a study period led to the production of a draft standard, NP 15940, *Software Engineering – Software Engineering Environment Services*, prior to producing a definitive standard that would include all those services required by a software lifecycle process. Reference [3] was used as a key input, given the widespread support it enjoyed.

Our paper is organised as follows. Following this first section, the introduction, comes Section 2, dedicated to the concepts underlying the standard, i.e. the future ISO/IEC 15940. Next, in Section 3, we provide an introduction to current reference model categories for SEE services as they are today. And, finally, we present a series of conclusions.

2 Concepts Underlying The Standard

SEEs refer to a collection of services, partially or fully automated by tools, that are used to support software engineering activities. An SEE provides automated services for the engineering of software systems and the management of software processes. It includes the platform, system software, utilities, and installed CASE (Computer-Aided Software Engineering) tools. A service is an abstract description of support for activities and tasks for the improvement of issues such as productivity, quality, or performance. A service may be assisted by CASE tools.

A service is self-contained, coherent, discrete, and may be composed of other services. A CASE tool is a software product that can assist software engineers by providing automated support for software life-cycle activities as defined in [5]. Finally, an automated process is a software process that is enacted with either full or partial support of CASE tools.

The term SEE may cover several situations: from the mere juxtaposition of a few tools running on the same operating system, up to a fully integrated environment able to handle, monitor, and even control all the data, processes, and activities in a software life cycle. An SEE provides support to human activities through a series of services that describe the capabilities of the environment. The software process supported by an SEE becomes an assisted or automated software process. This standard describes SEE services and relates them to [5] in a manner applicable to a range of organisations. When defining a lifecycle process an organisation needs to find the appropriate level of automation. This may result in establishing a new SEE or improving an existing one. As an SEE's capabilities are expressed by means of services, this emphasises the fact that an individual performs activities with the help of the SEE. Services provide a link between a set of chosen software life cycle processes and their automation by means of tools. In most cases, a tool's functionality can be related to one or more services.

Through the partial or full automation of activities an SEE provides benefits to an organisation in the form of lower costs (higher productivity), improved management, and the higher product quality that can be produced. For example, the automation of repetitive activities such as the execution of test cases provides not only productivity gains but can also help to ensure completeness and consistency in testing activities.

Specific criteria or the process used in the selection of one or more CASE tools, or recommendations to adopt CASE tools within an organization are outside the scope of project 15950. These criteria and detailed CASE tool characteristics can be found in [6] and [7], currently under revision by ISO/IEC JTC1 SC7 Software and Systems Engineering Sub-Committee.

3 Reference Model: Categories for SEE Services

The upcoming ISO/IEC 15940 will provide a reference model for SEE services. As a reference model, ISO/IEC 15940 will make use of a set of conceptual descriptions to describe each service used in a project support environment. "Conceptual description" means that description is performed from a reference viewpoint, and does not deal with any specific implementation. The description is therefore general and does not assume any specific application domain, life cycle model, or tool in a project. In this way ISO/IEC 15940 will be applicable to any defined organisational environment.

An actual environment is realized from a reference model containing conceptual descriptions. Therefore, an actual description of a specific environment would reflect a particular activity with its tools and standards. In the current draft CD ISO/IEC 15940, services are grouped into six categories that reflect broad functional activities within a typical software engineering organisation. The six categories of services are:

- Technical engineering: Technical Engineering Services support activities related to the specification, design, implementation, testing, and maintenance of software.
- Technical management: The services in this section fall into a category that considers both technical engineering and project management. These services pertain to activities that are often shared by engineers and managers.
- Project management: The services in this section support the activities related to planning and executing a project. Following project initiation, it will be necessary to carry out detailed planning of the project activities, together with ongoing monitoring and re-planning of the project to ensure its continued progress.
- Process management: The services in this section support projects with a view to achieving discipline, control, and clear understanding in their life-cycle development processes as understood in IS 12207 and in the individual process steps.
- Support: Support services include services that the rest of the services will require to become operational. They generally include the services associated with processing, formatting, and disseminating human-readable data.
- Framework: These services comprise the infrastructure of an SEE and will be required to support an SEE once it is actually implemented.

For each service it is possible to enumerate Basic Services Operations (BSO) and a number of service tasks that can be automated, known as Automated Operations (AO). An example for each is provided in the next section.

4 General Breakdown of SEE Services

The following section takes a more in-depth look at some SEE services to give the reader a better understanding of the structure of SEE services.

4.1 Technical Engineering Services

- **Software Requirements.** Provides the ability to capture, represent, analyse, and refine the system requirements that are allocated to software components. Examples: for BSO, to elicit and capture software requirements; for AO, requirements traceability.
- **Software Design.** Provides the ability to capture, represent, create, analyse, and refine the design attributes of the software components of a system or subsystem. Examples: for BSO, to translate requirements into design elements; for AO, traceability and consistency checking from software requirements specification to design elements.
- **Software Simulation and Modelling.** Provides the ability to simulate and model in order to determine the effectiveness of alternative designs with regard to such attributes as user interface characteristics or execution flow. Examples: for BSO, to build a model (graphical, logical, mathematical, etc.) from requirements; for AO, assistance for graphical operations.
- **Software Verification.** Provides the ability to confirm by examination and provision of evidence that the specified requirements have been fulfilled. Examples: for BSO, to analyse specifications for consistency; for AO, inconsistency identification.
- **Component Based Software Generation.** Provides the ability to automatically and semiautomatically generate software components using existing components or component templates. Examples: for BSO, to generate a parser from a syntactic language description; for AO, traceability from the components to the design specifications.
- **Source Code Generation.** Provides the ability to generate modules from design specifications. Examples: for BSO, to generate modules from design specifications; for AO, module generation from design specifications
- **Compilation.** Provides the ability to support for the translation and linking of software components written in various programming languages. Examples: for BSO, to find code and inheritance dependencies among a set of software components; for AO, to provide a compilation error list and a description indicating module names and line numbers.
- **Software Static Analysis.** Provides the ability to provide static analysis or source code analysis of software components in order to determine structure within the component. Examples: for BSO, to collect raw statistics from component; for AO, the estimation of a computational metric of the complexity of a component.
- **Debugging.** Provides the ability to locate and repair source code errors in individual software components by controlled or monitored execution of the code, and by tracking down errors and replacing code. Examples: for BSO, the execution of programs incrementally; for AO, execution output monitoring and saving.
- **Software Testing.** Provides the ability to test software systems at individual software component level (unit testing), and to test collections of software components (integration testing), and complete software systems (system testing). Examples: for BSO, to generate test cases; for AO, to record and store test cases.
- **Component integration.** Provides the ability to support the development of software components that are uniquely defined and combine these into a larger system or product version that can be managed as a whole. Examples: for BSO, to prepare software components for use; for AO, component interface management
- **Software Reverse Engineering.** Provides the ability to capture design information from source or object code, and to produce structure charts, call graphs, and other design documentation to provide new functionality or support a new environment. Examples: for BSO, generate design from source code; for AO, to design generation.
- **Software Reengineering.** Provides the ability to take a new or modified set of software requirements and the existing design as input and produce a new or modified design. Examples: for BSO, to perform impact analysis of new design on existing software components; for AO, modelling support.
- **Software Traceability.** Provides the ability to record the relationships between items of the development process. Examples: for BSO, to create, update, and destroy relationships between two items; for AO, to analyse results presentation for traceability analysis.
- **Software Prototyping.** Provides the ability to produce a software system that reproduces the user interface and emulates the functionality and behaviour of the final system to be built. Examples: for BSO, to build a model from requirements; for AO, model build from requirements.
- **Documentation.** Provides the ability to support the development, integration, configuration management and traceability analysis of online and paper documentation. Examples: for BSO, building online documentation into a delivery package; this can also be provided as automated support.

4.2 Technical management services

- **Configuration Management.** Supports the identification, documentation, and control of the functional and physical characteristics of configuration items to ensure traceability. Examples: for BSO, to create a baseline definition; for AO, to uniquely identify all configuration items and all changes to configuration items.
- **Change Management.** Supports the creation of change requests, change orders, and an audit trail of changes to product components. Examples: for BSO, to create a change request in response to a reported error, omission, or required update; for AO, provision of a historical record of a change request item.
- **SEE Repository Management.** Provides the ability to create, access, and modify information objects (i.e. requirements specifications, test cases, simulation cases, E-R – Entity-Relationship-diagrams, etc) in SEE repository man-

agement and to record the relationships between them. Examples: for BSO, to create, access, and modify groups of information objects; for AO, any of the basic services.

- **Reuse.** Supports the storage, inspection, and reuse of assets related to the engineering processes. Examples: for BSO, to catalogue, register, and classify the asset; for AO, to provide asset registration and cataloguing.
- **Metrics Collection and Analysis.** Provides facilities for the collection and organisation of primitive data into meaningful information to the end-users of the SEE. Examples: for BSO, to compare a data set against a predicted model; for AO, primitive data collection.
- **Quality Assurance.** Supports the definition, tracking, and performance of quality assurance activities and the analysis of their results. Examples: for BSO, to establish and maintain records of quality assurance activities; for AO, impact analysis of quality assurance failure on a specific process assurance item.
- **Audit.** Supports the planning and performance of audits and the analysis and reporting of their results. Examples: for BSO, to maintain a set of audit checklists; for AO, an audit checklist linked to requirements.

4.3 Project Management Services

- **Planning.** Provides operations that permit data handling according to a set of project objectives relevant to a project's constraints. Examples: for BSO, to compute event lead times; for AO, time graphing of key project events.
- **Estimation.** Supports the quantification, analysis, and prediction of project costs and resource needs. Examples: for BSO, to create and modify cost, size, and resource estimates; for AO, to estimate changes linked to requirement changes where appropriate.
- **Risk Analysis.** Supports the planning and assessment activities that consider elements related to the success or failure of a project. Examples: for BSO, to perform tradeoff analyses based on differing parameters for resource allocation and scheduling data; for AO, cost and schedule measuring.
- **Tracking.** Supports the tracking of project progress including the cost, schedule, and user requirements. Examples: for BSO, gathering metrics related to the current status of a project and its constituent work activities; for AO, trend analysis for cost deviation, and size.
- **Evaluation.** Supports the analysis, evaluation, and decision making associated with service tracking, data collection metrics, and user acceptance criteria. Examples: for BSO, to elicit user acceptance for each requirement of the project product; for AO, to assess project/product outcomes against user acceptance criteria.

4.4 Process Management Services

- **Process Definition.** Provides for the establishment of the organisational processes covering the software life cycle via the adaptation and tailoring of a set of higher order reference processes. Examples: for BSO, to analyse process requirements, including domainspecific analysis and application-specific analysis; for AO, any of the basic operations.
- **Process Library.** Supports reuse capabilities for processes, including the creation, update, deletion, certification, measurement, and management of process assets (activities, tasks, etc.). Examples: for BSO, to create, update, and delete process assets; for AO, process assets storage and versioning.
- **Process Initiation.** Supports the assignment of a life cycle model, a set of processes and an SEE to meet the requirements and constraints for a particular project. Examples: for BSO, to review project criteria and constraints, and select a life cycle model; for AO, relationship definition, and tailoring of processes and activities.
- **Project Process Usage.** These services include capabilities for user selection and guidance, selection and control of process steps, navigational and help facilities for users to query the installed process for information on successful actions. Examples: for BSO, the specification, collection, and reporting of project process metrics; for AO, process utilisation and status querying and reporting.
- **Process Monitoring.** Supports the observation, detection, logging, and tracking of process activities (within projects). Examples: for BSO, to set up monitoring conditions and criteria; for AO, detection and log monitoring.
- **Process Improvement.** Supports the assessment, measurement and modifications of the organisational and project specific processes, and project life cycles. Examples: for BSO, define effectiveness goals; for AO, measurement data collection.
- **Process Documentation.** Supports those services related to process documentation. Examples: for BSO, to identify the documentation requirements; for AO, documentation design, production, and editing.

4.5 Support Services

Support services include services that the rest of the services will require in order to become operational. They generally include services associated with processing, formatting, and disseminating human-readable data. This section describes the following services:

- Common Support.
- Publishing.
- Team Support.
- User Communication Support.
- SEE Administration.
- Policy Enforcement.

4.6 Framework Services

These services make up the infrastructure of an SEE and will be required to support an SEE once it is actually implemented. This section describes the following services:

- SEE Infrastructure Management.
- Communication.
- Object Management.

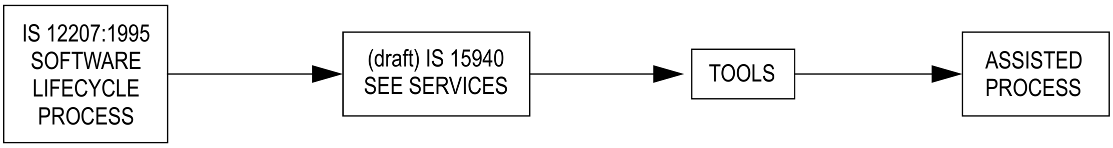


Figure 1: Path from Processes to Assisted Software Processes.

5 Mapping Services onto Processes

Since SEE services are understood to be closely linked to software lifecycle processes, it is possible to map [5] activities onto SEE services. Examples of this are, activity 5.3.4 from [5], software requirements analysis, onto software requirements engineering, software prototyping and user communication. Another example from [5], 5.3.8, software integration, onto software testing, component integration and metrics collection and analysis. This is not only a consistency issue. This is also a way of establishing a path from processes to services, and then on to the tools that help processes to be performed, as described in Figure 1.

6 Conclusions

This paper has described the main guidelines and concepts underlying the upcoming ISO/IEC 15940 Software Engineering – Software Engineering Environment Services standard. At present this project is at committee draft stage at ISO/IEC JTC1 SC7 Software and Systems Engineering. A baseline for SEE services in the context of the widely adopted [5] has been set, while some issues may change during the standard drafting process.

References

[1] ISO/IEC 13719-1:1995 Information technology – Portable Common Tool Environment (PCTE) – Part 1: Abstract specification (ECMA-149). < http://www.ecma-international.org/publications/standards/Ecma-149.htm>.

[2] Learning From IPSE's Mistakes. Alan W. Brown, John A. McDermid. IEEE Software, March/April 1992 (Vol. 9, No. 2) pp. 23–28. < http://csdl.computer.org/comp/mags/so/1992/02/s2023abs.htm>.

[3] Reference Model for Frameworks of Software Engineering Environments, 3rd Edition (NIST Special Publication 500-211/Technical Report ECMA TR/55). 1993.

[4] Reference Model for Project Support Environments. 2nd edition. (ECMA Technical Report TR/69 NIST special publication 500-213) 1994. <http://www.ecma-international.org/publications/files/ECMA-TR/TR-069.pdf>.

[5] ISO/IEC 12207:1995, Information Technology – Software Life Cycle Processes. <http://www.software.org/quagmire/descriptions/iso-iec12207.asp>.

[6] ISO/IEC 14102:1995, Information technology- Guideline for the evaluation and selection of CASE tools (under revision by ISO/IEC JTC1 SC7).

[7] ISO/IEC TR 14471:1999, Guidelines for the adoption of CASE tools.

Open Source and Free Software: A New Model for The Software Development Process?

Alfonso Fuggetta

Open source software is having a significant impact on the ICT market. Unfortunately, many claims associated with open source software are either misleading or simply false. This makes it difficult to really appreciate and exploit the potential of open source software. This paper proposes some considerations and reflections that aim at critically revising some assumptions about open source software. The ultimate goal is not to deny the role of open source software; rather, the paper aims at identifying the really novel and original characteristics of open source software with respect to more traditional approaches.

Keywords: Business Model for Software, Open Source, Software Development Processes.

1 Introduction

Open Source Software (OSS) is certainly one of the most important and relevant phenomena of this decade. The success of Linux and Apache is pushing practitioners and researchers to reconsider some of the classic assumptions about software development. Even software giants such as Microsoft [5], Sun, and IBM have been changing or adapting their strategy to take into account this unconventional approach.

1.1 The Reasons of A Success

The supporters of OSS claim that open source is able to address and solve a number of issues. In particular, it is supposed to be a more effective and efficient way of developing high quality software. Moreover, it makes it possible to disseminate innovation and technology more easily and effectively. It is also an extremely attractive approach to lower the costs of IT investments. In Europe, many observers consider open source an effective strategy to counterbalance the American dominance in software technology and, consequently, to revamp the European software and computer industry, which was dramatically weakened in the 90' by the collapse of BISON (i.e., Bull, ICL, Siemens, Olivetti, and Nixdorf).

Beside these technical and economic issues, open source software is also – and for many supporters, primarily – an ethical and cultural issue. In particular, free software advocates, such as Richard Stallman, claim that software must be 'free' (as in "free speech") because proprietary/close software violates five basic users' rights:

- The freedom to use the software.
- The freedom to study the source code.
- The freedom to modify it.
- The freedom to copy it.
- The freedom to redistribute it.

Indeed, free software and open source are often considered equivalent concepts. However, even if most practical consequences are basically the same, the two approaches have different backgrounds and motivations. For the sake of simplicity, in the remainder of the paper I will use the term OSS to identify the whole world of open/free software. When needed or convenient, the distinction between these two notions will be made explicit.

1.2 Some Basic Concepts

Software products can be classified in two main categories: packages and custom (or bespoke) software.[1]

- *Packages* are software products developed to address general needs for a number of different users. Moreover, they are distributed through licenses that define customers' rights and obligations. The license defines a package as proprietary or open source (or some intermediate variant). Typical examples are Windows, Linux, Office, and StarOffice.
- *Custom software* is developed for specific needs of a customer, who pays the cost of software development. Usu-

Alfonso Fuggetta is a Full Professor at *Politecnico di Milano* (Italy), *Dipartimento di Elettronica e Informazione*. He is also Director of CEFRIEL, the ICT "Centre of Excellence for Research, Innovation, Education, and Industrial Labs" partnership established in 1988 by *Politecnico di Milano*, the Regional Council of Lombardy, and the most important ICT companies operating in Italy. Fuggetta is a Faculty Associate of the Institute for Software Research of the University of California, Irvine, since 2002. He is also Chairman of the Scientific Committee for New Economy, Innovation, and Scientific Research of Lombardy, and of several committees of the Italian Government including the Government Committee on Open Source Software in the Public Administration. <alfonso.fuggetta@cefriel.it>

[1] Of course, there are also intermediate situations where a software product is a combination of packages and custom software.

ally, this is accomplished by signing a service contract between the customer and a software house or system integrator. Such contract can and should always guarantee to the customer the full ownership of the software (possibly non-exclusively), as it pays the entire cost of development. In particular, full ownership means unrestricted access to the source code, i.e., even more than what is granted by open source licenses such as the GPL: a customer might even decide to put under public domain the custom software it purchased! A typical example of custom software is the software used to manage a specific procedure in a Public Administration.

From the above observations it can be argued *that the notion of open source apply primarily to packages*, as discussed in detail in the remainder of the paper.

1.3 Open Source, Open Standards, and Open Format

OSS is often associated with open standards and open format. Indeed, these concepts are orthogonal.

An open standard is a *set of requirements* that are not controlled by a single company. An open standard is useful to guarantee that any product adhering to the standard provides compatible and coherent features and operations. This is crucial to provide interoperability and the possibility to replace a product with a compatible one.

An open standard may be adopted by both OSS and proprietary software. Actually, Internet defines a set of "open standard" protocols such TCP (Transmission Control Protocol), which are available both in proprietary and open source operating systems (e.g., Linux and Windows).

Certainly, proprietary software must interact with the "rest of world" using open standards and open formats. Again, this is not the same as requiring that software must be open source in order to be open standard.

1.4 Misleading Beliefs, Unjustified Expectations

The notion of "open source" is associated with three different concepts:

1. A set of *licenses*, such as the well-known General Public License (GPL).
2. A *range of software development practices*, which exploit the notion of open source to facilitate cooperation, quality assurance, and innovation.
3. *Products* that are developed and distributed using an open source license and open source development practices.

In turn, these three factors are supposed to induce a whole new way of conceiving and running the software business. In practice, software developers and producers are supposed to make money from selling services (distributions, training, documentation, consulting, …) rather than from license fees.

In general, there is a growing trend towards considering OSS as 'the' strategy for the future of the software industry. Unfortunately, most of the beliefs and claims about OSS are fascinating on the surface, but false or misleading when studied in more detail: very often, they are either unjustified or simply *independent of the nature of software* (i.e., they equally apply to open and proprietary software). The effect of this situation is the growth, especially in Europe, of unjustified expectations that might turn out to overlook the real problems, ignore key issues, and devaluate the real impact and significance of OSS itself [4].

1.5 Goal of This Paper

This paper presents some considerations about the many claims and expectations associated with OSS. The goal is not to deny the role and opportunities associated with OSS. Rather, it aims at identifying the real and novel characteristics of OSS in order to effectively exploit them. For this reasons, the following sections will discuss some main questions related to technical, economic, and social aspects of OSS. The last section will provide some concluding remarks and suggestions for future work.

2 Is OSS A New Development Process?

OSS supporters claim that the openness of software and the cooperation styles supposedly used in many OS projects define a new and extremely effective software development process. Such process is based on the notions of decentralized development, distributed testing, and effective exploitation of distributed expertise and knowledge. The originator of this position is Raymond, whose seminal work on "The cathedral and the bazaar" has been followed by many other studies and contributions trying to define the "open source development process".

Indeed, *there is no such process*. There are two main motivations that support this claim [4]:

- Most software development initiatives are carried out by a limited (often just one) number of developers. This observation has been corroborated by several studies of open source repositories such as Sourceforge.net [6]. Large open source projects such as Linux and Apache do have a very well structured and organized process, which resembles those of other proprietary products [8].
- The features and characteristics of the "open source development process" can be applied and observed also in proprietary software. For instance, Microsoft exploits daily builds and feature orientation as a way to make software development flexible [3]. Approaches such the Spiral model and Extreme Programming (XP) have stressed the notion of incremental and evolutionary development, which is supposed to be a main characteristic of OSS. Actually, those approaches can be equally applied to proprietary and open source software.

In general, from a technical viewpoint it is hard to consider open source as a totally new development paradigm. Certainly, the vision behind OSS represents a strong motivation factor that has been able to involve and influence a very large number of developers and users. This is one of the real novel aspects of OSS: its fascinating ability to motivate people.

Actually, even if not directly related to the hypothetical open source development process, some ideas and suggestions related to open source do have a role to play in specific situations. The most relevant one is the sharing and joint development of custom software in a community, e.g., the Public Administration sector. There are hundreds (even thousands) different Public Administrations that share the same problems and requirements. For example, town councils have the same needs and, therefore, can share the same software. This can be achieved by either using the same package or reusing the same custom software. Custom software, which should be in the full ownership of the purchasing administration, can and should be shared with other administrations using some form of open source licensing (probably, something similar to community sourcing). This is another aspects of OSS that should be much more deeply considered, even if it is not directly related to classical packages such as Windows and Linux.

3 Is OSS The Only Way to Protect Customers' Rights?

OSS advocates claim that open source is the only way to protect customers' right. This is at least misleading. As for custom software, the customer should always own it and, therefore, *the problem does not exist by definition*. Indeed, the problem does exist for packages.

A first problem is to access the source code in order to check what the software really does. This is useful to guarantee that a software package does not accomplish undesired or illegal operations or, also, to support testing of custom software developed using package features (e.g., a custom software using Oracle DB, Windows, and .Net). In order to solve this problem, *it is sufficient to make the source code of a package 'accessible' to the user* (i.e., the user can see and modify it, but it cannot redistribute or copy it illegally). This requirement is much weaker than making it open source. Still, it is sufficient to solve the problem and is probably acceptable to most producers of proprietary software.

A second problem concern the inability of proprietary packages' users to change the company in charge of maintaining the software, or to take control of the software whenever the developer is unable or unwilling to continue maintaining the package. This is a critical problem that can be at least partially solved by introducing specific norms to protect customers. For instance, if software is sold with a license that does not have time limitation, the producer should not be allowed to drop support and maintenance services once a new version of the package is released, unless the source code of the older version is made open. Similarly, a company that is patently unable to provide support and maintenance (for instance because it has deep financial problems) should be forced to release the source code.

4 Is OSS An Effective Way to Disseminate Knowledge?

OSS is supposed to be the means to solve a number of problems related to dissemination of knowledge, international cooperation, and the digital divide. Again, this is misleading.

First, knowledge about software cannot be distributed by simply looking at the source code. As a provoking statement, I argue that *the larger (and significant) is the code, the smaller is the amount of information that can be disseminated by simply looking at that code*. Indeed, software engineering research and practice have demonstrated that developers need high-level requirement and architectural documents that are able to describe a software system. In the past years, an entirely new discipline has been started (reverse engineering), whose goal is to extract meaningful information from source code.

Second, even assuming that looking at the source code does transfer knowledge, it would be sufficient to make the code 'accessible' (as mentioned above). It is not necessary to make it open source, i.e., to grant the customer also the right to copy and redistribute the software.

5 Is OSS Cheaper?

Stallman strongly argue that OSS does not mean 'free' as in "free beer". OSS can be commercially distributed, as companies such as Red Hat do (see later on). Also, software needs to be maintained and supported. OSS advocates say that open source developers make money by selling services. Therefore, *OSS does cost*, as any other software.

As a consequence, any conclusion about OSS being cheaper than proprietary software should be based on a detailed analysis of all the costs related to owning and operating a specific software solution. This is called *Total Cost of Ownership* (TCO). Someone argues that TCO has been invented to support the claims of proprietary software producers. Indeed, this notion is widely used to decide any sort of purchase (e.g., a car or computer hardware) and therefore it is difficult to understand why it should not be valid and applicable also in the case of software.

6 Is OSS An Effective Business Model?

OSS is considered a viable and even unique way to build a convincing and enduring business model. People will be less inclined to pay for software licenses and will rather prefer to pay for specific services associated with using the software. This appears as an additional motivation for opening the code of software packages.

The argument is once again ill conceived. First, software costs and it is not clear if companies can survive and make profits by simply selling services. In his latest book [2], Michael Cusumano has accomplished an analysis of existing companies, their profits and strategies, and their success stories. He argues that in the future most software companies (including Microsoft, IBM, and other producers of proprietary software) will increasingly rely on a model where revenues will be *a mix of product licenses and services*.

It is useful to assess Cusumano's comments by considering in more detail the real business models where OSS is supposed to play a role. There are five basic models to consider:

1. *Development and distribution of 'pure' open source packages (and related services).* A typical example is Red Hat, who makes money distributing and supporting Linux.
2. *Development and distribution of open source packages (and related services) developed for open source and proprietary platforms.* This is the case of companies such as Zope, which has created an open source development platform for both Windows and Linux.
3. *Development of proprietary packages (and related services) for open source and proprietary platforms.* This is the case of StarOffice (not OpenOffice!), developed by Sun for

Linux and Windows. Similarly, IBM sells a number of proprietary packages for the Linux platform.

4. *Development of open source packages (and related services) that make a product line more attractive*. This is once again the strategy of IBM and Sun, which promote the diffusion of Linux and other open source packages (in particular, Apache) on their hardware products to increase their competitiveness with respect to the Wintel platform.
5. *Development of custom software (and related services) using open source platforms.* This is the case of many software houses and system integrators who base their developing activities on Linux, Zope, Tomcat, and JBoss rather than on Windows (and related technologies).

There are also companies that are exploiting hybrid approaches. For instance, MySQL uses a dual-license approach that integrates open source and proprietary concepts.

In general, the above models appear to cover all the possible basic 'bricks' of a hypothetical business model based on open source. However, *only the first two strategies are really and truly exploiting open source.* The third and fourth alternatives are commercial strategies of companies who want to promote their proprietary software and proprietary hardware platforms. As for the fifth one, indeed the business model of a system integrator does not change that much. A system integrator (i.e. a developer of custom software) has always been used to consider alternative platforms. In some cases, this is caused by specific requirements of the customer. In other situations, where such requirements are missing or weaker, a system integrator is used to look for the most convenient platform for its development. Certainly, open source packages such as Linux, Apache, and Tomcat make it possible to build software systems with lower costs for licenses. In general, a system integrator will consider the TCO of each different alternative and select the most convenient one. This already happened in the past. For instance, the change from large mainframes to minicomputers and, later on, to PC networks was motivated by the relatively much lower entrance costs of the newer technology. Summing up, for system integrators there is nothing really and dramatically new.

In conclusion, will these models be the future of the software industry? Hard to say. First, only two of them are really a significant departure from traditional models. Second, the number of companies who are really succeeding using those approaches is still relatively small. Even Red Hat, perhaps the most successful and large open source player, is still in search of the right strategy to be market profitable.

7 Is OSS A Strategy to Strengthen The Software Industry?

A final important claim of many OSS supporters is that open source can be an important means to strengthen the software industry, especially outside the US. Even if I totally share the concerns of those that correctly require a stronger non-US software industry, I am very skeptical that open source alone might have a significant impact. Europe, for instance, misses a clear and committed industrial strategy for software. Mobile phone producers are a typical example. Microsoft (and PalmOS) is promoting a standard platform for mobile phones. This standard can repeat the success of the personal computer: there are many hardware producers, but one standard software. European producers, conversely, have multiple and incompatible platforms that make it difficult for developers to invest in developing applications. If an application is developed for Windows Smartphone or Pocket PC, a developer is sure that it can run on any device compatible with those systems. Conversely, can an application developed for the Symbian-based Ericsson P900 run also on the Symbian-based Nokia Communicator? The answer is no. So, an application developer will hardly consider Symbian an attractive platform, while he/she will be certainly inclined to invest in Pocket PC or PalmOS development. Eventually, the end user will make his/her choice on the basis of the applications available on each platform.

This is the real problem. Europe misses an industrial strategy. "Supporting open source" is not a strategy. At best, it is a request to fund open source projects with public money. This is not what we need. We should consider lessons such as Airbus. By creating an aircraft company able to compete with Boeing, Europe has become the main player in the market. This was accomplished using public money, but there was a clear industrial strategy aiming at creating innovative products. Software is not innovative just because it is open source. *Europeans are starting from the tail (a dissemination strategy such as open source) rather than from the head (the identification of the innovative products to develop).*

8 A General Remark

Indeed, the more I consider the literature on OSS, the more I am convinced that in many situations the really important thing for many customers is that open source is 'free' as in "free beer". This is a huge risk. *Software is not 'free'*; it does cost. The issue is then who is going to pay this cost. Imagining that software can be created at no or low cost is a major threat to innovation. Indeed, how many open source projects are really innovative? Why most of them are just replicas of existing software? Customers should not be led to believe that software could be acquired "for free".

9 Conclusions

This paper has summarized some of the main issues and discussions about OSS. The paper does not assume an a-priori position in favour or against open source. Rather, it tries to critically discuss and assess many claims made on OSS. Unfortunately, many of them are ill conceived, misleading, or patently false. This is dangerous to OSS itself in the first place.

Certainly, it is important to deepen the discussion on the real novel aspects of open source. Even more important, we need to find effective and convincing solutions to the problems that our society and the software industry have to address in the next years. This papers wants to be a contribution in this direction.

References

[1] R. Conradi, A. Fuggetta. Improving software process improvement. IEEE Software, July 2002.

[2] M. Cusumano. The Business of Software. Free Press, 2004.

[3] M. A. Cusumano and R.W. Selby. Microsoft secrets. The Free Press, 1995.

[4] A. Fuggetta. Open source software: an evaluation. Journal of Systems and Software, April 2003.

[5] J. Greene. Microsoft's Midlife Crisis. Business Week, April 19, 2004.

[6] K. Healy, A. Schussman. The ecology of open source software development. Technical report, University of Arizona. Available at <http://opensource.mit.edu/online_papers.php>.

[7] D. G. Messerschmitt, C. Szyperski. Software Ecosystem. The MIT Press, 2003.

[8] A. Mockus, R. T. Fielding, J. Herbsleb. Two case studies of open source development: Apache and Mozilla. ACM TOSEM, Vol. 11, Issue 3, July 2002.

Applying The Basic Principles of Model Engineering to The Field of Process Engineering

Jean Bézivin and Erwan Breton

A new information system landscape is emerging that will probably be more model-centred than object-oriented, characterized by many models of low granularity and high abstraction. These models may describe various aspects of a system such as software product properties, static and dynamic business organization, non-functional requirements, middleware platforms, software processes, and many more. Each model represents some particular point of view of a given system, existing or in construction, and this model conforms to a precise metamodel or DSL (Domain Specific Language). In this paper we present some advantages of using the unification framework of model engineering to deal with the various facets of process engineering. As the view of the software life-cycle is progressively shifting from a simple definition and composition of objects to a sequence of model transformations, the need to characterize this by a precise process is becoming urgent. Description of software artifacts, processes and transformations may all be uniformly captured by different forms of models. This approach provides a regular framework where business and software production process models are going to play an increasingly important role. In this paper we illustrate some possibilities of model-based process engineering.

Keywords: Domain Specific Languages, MDA, Model-Based Process Engineering, MS-Project, SPEM.

1 Introduction

As stated in the seminal work of Osterweil [10], *"Software processes are software too"*. Therefore, the same technologies may be used for supporting both software artifacts and software processes. Among these technologies, Model Driven Engineering (MDE) has recently taken an important place. One of the best known families of MDE is the Model Driven Architecture (MDA™ organization) proposed by OMG (Object Management Group) in November 2000 [11]. As a consequence we can today envisage a new situation where software products are models and software processes are models too. At a lower level of abstraction, object technology tried twenty years ago to unify the field of software engineering by considering objects as first-class entities. The new trend of model engineering considers instead models as first-class entities. A model is a representation of a given system and is written in the language of its metamodel (it conforms to its metamodel). These relations of *representation* and *conformance* are characteristics of new model engineering techniques [2].

MDA gives a central role to models and metamodels. Taking account of the various aspects of information system engineering can only be achieved through the well-organized management of a complex lattice of related metamodels. The emerging tendency is to separate business knowledge expression from implementation as two separate aspects captured by specific models. The invariants of a business domain may be held apart from the execution code, preserving them from technology obsolescence. The information of a complex enterprise system originates from various sources (system engineers, managers, software engineers, quality engineers, etc). All the corresponding domains may be expressed using their own specific languages.

This paper discusses process engineering using MDA-based techniques. We present in Section 2 a brief overview of MDA principles. In Section 3, we focus on software process-related standards. Building on this, we show the practical interest and

Jean Bézivin is Professor of Computer Science at the University of Nantes, France, member of the newly created ATLAS research group in Nantes (INRIA & CNRS/LINA). He has been very active in Europe in the Object-Oriented community, starting the ECOOP series of conference (with P. Cointe), the TOOLS series of conferences (with B. Meyer), and more recently the <<UML>> series of conferences (with P.-A. Muller). His present research interests include legacy reverse engineering, general model engineering (MDE/MDA™) and more especially model-transformation languages and frameworks. He is currently involved in the Modelware and Interop european projects.
<Jean.Bezivin@lina.univ-nantes.fr>

Erwan Breton is a Consultant in the Sodifrance/SoftMaint company in Nantes, France, working on the subjects of process modelling and workflow. He is presently in charge of the R&D strategy for the company. After obtaining a Master degree from the University of Lille 1, France, he got a PhD in Computer Science from the University of Nantes. He has published several papers in the area of model-based process engineering.
<ebreton@sodifrance.fr>

the industrial applicability of this vision in Section 4, based on a simple example. General considerations on present capabilities and limits of the approach are summarized in the conclusion together with some current perspectives on model-driven platforms.

2 Model Driven Engineering and The OMG/MDA Organization

In the IBM manifesto [4], the principles of model engineering are presented as the three vertices of a triangle:

- *Direct representation*, meaning that for each aspect of a system under construction or maintenance, there is a domain specific language (DSL) dedicated to handling this aspect;
- *Automation*, meaning for example that a mapping from DSL programs to executable frameworks may be automatically handled, mainly by model transformation procedures;
- *Open standards*, on which these DSLs and transformations will be based to allow the cooperation of various tools for capturing and operating on models.

Each DSL corresponds to a given metamodel. In the OMG MDA stack, all metamodels conform to a unique meta-metamodel named the MOF (Meta-Object Facility). The MOF is a language to write metamodels. Furthermore the MOF has a number of common facilities to deal with metamodels and their models. Some of them are related to establishing bridges to other technical spaces. MDA is one such technical space that could be called modelware but middleware, grammarware, executable programming languages, XML (eXtensible Markup Language) document management, data bases, Semantic Web and ontology engineering are other examples of technical spaces. There is a standard mapping between MDA and CORBA/IDL middleware (CMI, Corba Model Interchange [7]) but there are also other mappings to XML document management (XMI, XML Model Interchange) and to the Java technical space (JMI, Java Model Interchange).

The so-called OMG MDA stack is thus composed of the three following layers:

- At level M3, the MOF defines a representation system based on constrained graphs. These graphs are similar to UML (Unified Modelling Language) class diagrams. In addition to several facilities mentioned above, an assertion/navigation language named OCL is also provided at this level. The main idea is that level M3 contains all that is domain independent.
- Level M2 corresponds to the domain-dependent definitions. It is composed of a set of DSLs, each defined by its metamodel. This collection of metamodels is rapidly growing and may serve many purposes. Among the most known elements at this level, we may list the UML, SPEM (Software Process Engineering Metamodel), CWM (Common Warehouse metamodel), EDOC (Enterprise Distributed Object Computing metamodel).
- At level M1 various extensional definitions corresponding to intentional definitions of level M2 may be found. This is the level of models, each model conforming to a given M2 level metamodel.

The issue of tooling is important in this MDA engineering evolution. Developing an industrial tool is costly and takes time. Since there was no available tool and a very narrow market at level M3, the trick has been to consider temporarily an alternate way for a MOF editing tool. Instead of defining DSLs as full fledged MOF metamodels, it was made possible to define them as ‘specializations’ of UML, as so-called profiles. Two parallel competing techniques could then be used for defining the DSLs from scratch or as extensions of UML. The industrial success of UML allowed many industrial or open-source tools to become widely available in a short time, giving an easy path to defining DSLs as UML profiles. Some standards like the SPEM were even jointly defined as a full-fledged MOF metamodel and as UML profile at the same time. Many bridges were also provided to convert between UML profiles and MOF metamodels. Today both techniques are still used, the direct MOF metamodelling way being more widely used to define small well focused and precisely defined DSLs.

Looking backwards, the main turn was taken at OMG when the Analysis and Design Task Force (ADTF) decided to give up the quest for a Unified Method (i.e. a unique object-oriented software development method). This goal was considered as too ambitious and instead the OMG concentrated on defining a DSL for describing object-oriented software artifacts. This DSL was rapidly named UML. Its acceptance was followed later by the definition of another DSL for describing related software processes. Initially known as UPM (for Unified Process Model) it was later renamed as SPEM. Having two related languages, one (UML) for defining the basic software artifacts and one (SPEM) for defining the process for using or producing these artifacts could be considered, as an afterthought, as one of the major achievements of the ADTF definition group. Not all problems of relations between these two independent but related metamodels are yet solved, nor even clearly understood, but the original idea of separating the DSL for software products and the DSL for software processes was clearly an important step that later lead to the foundation of the MDA proposal. This is still a central motivating example and source of inspiration for studying aspect separation and weaving in the context of MDE.

The MDA is often presented as the separation from the platform-independent and platform dependent aspects of software systems. As a matter of fact this corresponds to the definition of DSLs for enterprise description (e.g. EDOC) and DSLs for platform descriptions (e.g. CCM: CORBA Component Models). But programs in these DSLs are intended to be interwoven in order to produce so-called PSMs (Platform Specific Models) from PIMs (Platform Independent Models). The real scope of MDA is indeed larger and encompasses aspect separation issues other than only business and platform aspects. One of them is for example the product/process aspect separation and weaving, either at the level of business (i.e. weaving models of business objects, rules and processes) and at the level of software production (i.e. UML and SPEM as mentioned earlier).

The generation of PSMs from PIMs should be made as automatic and generic as possible. This means that the target platform could be easily changed. This also means that we need

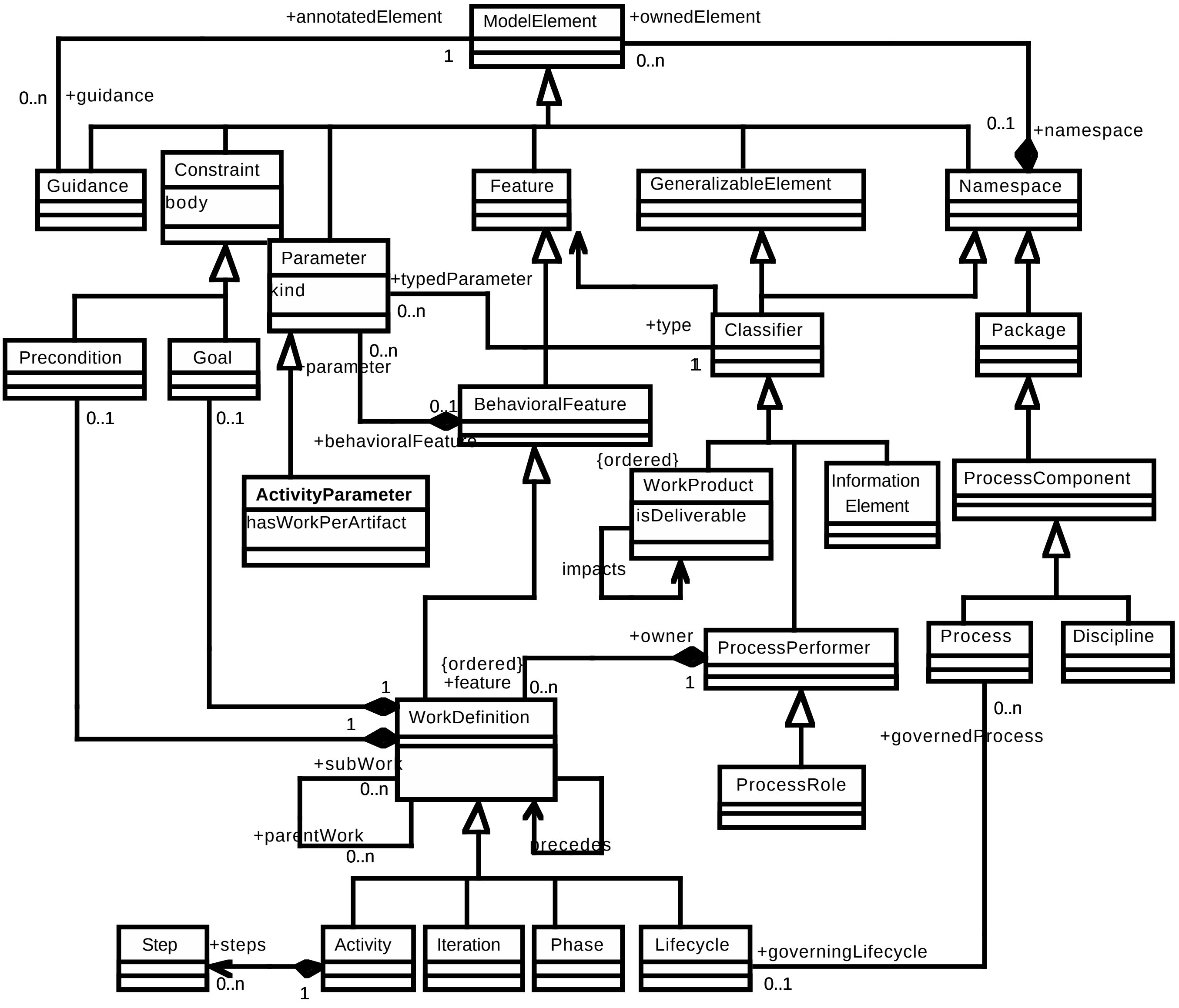


Figure 1: Fragment of SPEM Meta-model.

some transformation language – yet another DSL – to translate between any pairs of source and target DSLs. In order to provide this generic technology, the OMG has launched the MOF/QVT request for proposal (Queries/View/Transformation). The MIA [9] and the ATL [1] languages and systems may be viewed as QVT-compliant transformation systems.

3 Process Modelling

There are several specifications of the OMG that address the issue of process definition. UML introduces the concept of the activity graph. EDOC defines a process as a component with an internal choreography. SPEM is more specifically dedicated to software process description. Other recommendations deal with various forms of business processes.

SPEM defines a generic software process DSL (see Figure 1). A process is described as a set of work items (Work Definition). These work items are performed by process roles. The results of work definition are work products. A work product may be typed by another product DSL or part of, like UML or UML use cases or collaboration diagrams. For the time being the explicit references between SPEM models and software product models are sometimes left implicit.

Beyond these various OMG DSLs, other technical spaces also have their own DSLs in the field of process management, the most active being the XML document space with such proposals as ebXML (Electronic Business XML Initiative) or BPEL4WS (Business Process Execution Language for Web Services) for example. One example is BPMN (Business Process Modelling Notation) [12], but we may also mention IDEF, Activity-Decision Flow (ADF) diagrams, RosettaNet, LOVeM, Event-Process Chains (EPCs), ABC (Activity-Based Costing), REA (Resource/Event/Action), etc.

Studying all these notations for process definitions, we came to several conclusions in [5], and in particular:

- that a common description of all these notations with a unique framework, based for example on the MOF, could be useful to compare them. We expressed several of these formalisms as explicit MOF metamodels;
- that this homogeneous description of different process DSLs with the same metamodelling language could suggest using a common kernel (similar to the PIF, Process Interchange Format, or to the PSL, Process Specification Language) on top of which several extensions could be built;
- that similar concepts were often expressed in different ways in various technical spaces;
- that a clear expression with a precise metamodelling notation like the MOF could lead to well controlled modularity and extensibility. To give an example of this, the SPEM intended to define forward-based software process could be extended to take into account backward-based process dealing for example with legacy recovery and software modernization. The MDA approach is currently applied to both approaches, taking into account not only the platforms of the present and the platforms of the future but also the platforms of the past (COBOL, RPG, PL/1, etc.). Transforming a legacy PSM into a PIM is however probably harder than transforming a PIM into a PSM for EJBs, DotNet or the Grid;
- that process management encompasses different needs (process definition, project planning and enactment) that may be addressed with different tools (modeler, planning tools, workflow systems), which may be integrated using model transformation.

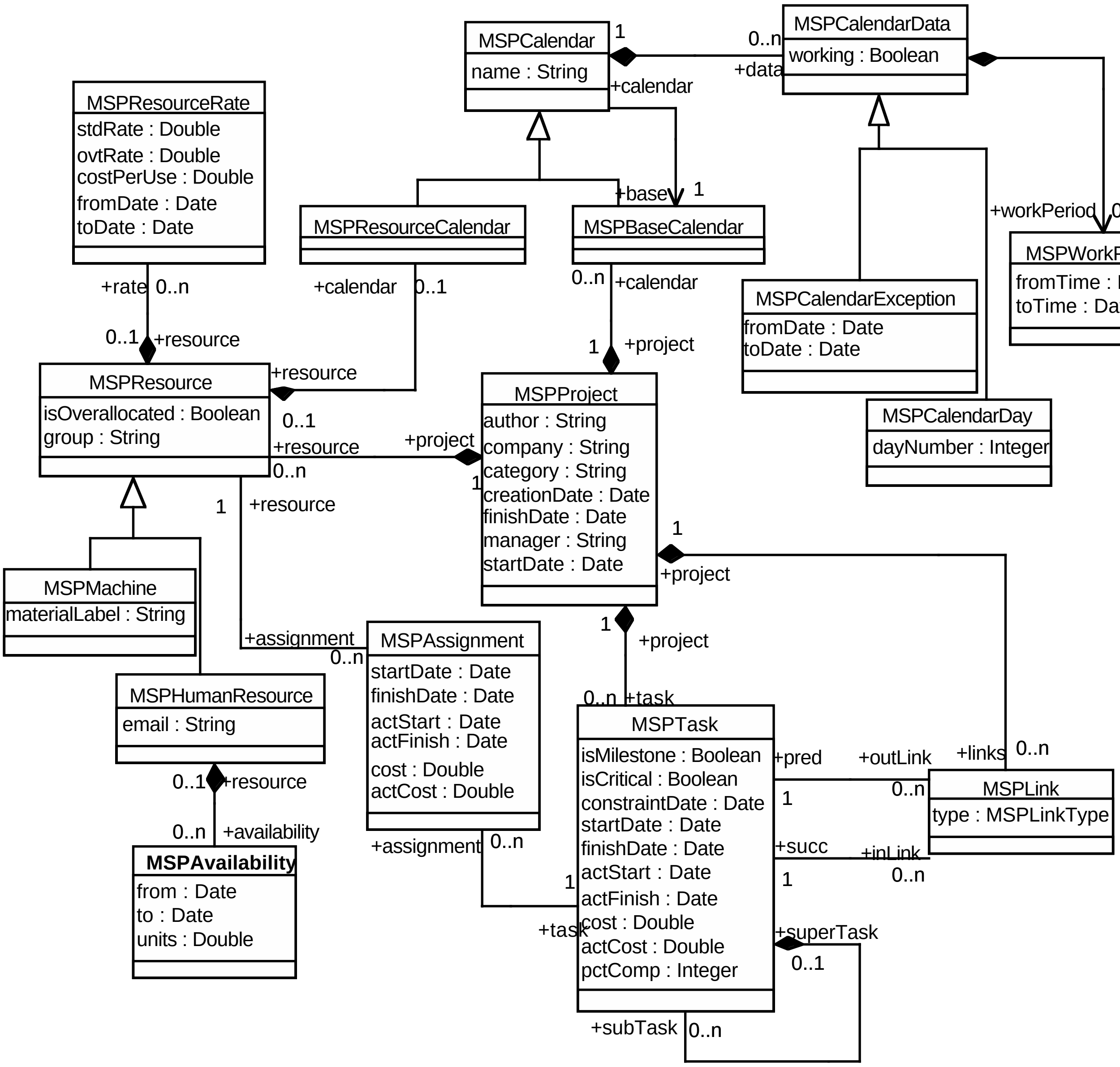


Figure 2: Fragment of MS-Project Meta-model.

It became also obvious that the expression of processes was often a matter of implementation platforms. Mapping abstract processes, expressed in DSLs based on a common metamodelling language like the MOF, onto these platforms could be of great interest. In other words we applied the MDA principle of PIM and PSM to the process themselves. The example proposed in the next section is an illustration of this possibility. Many other examples could have been provided like translating BPMN to executable BPEL4WS.

Another advantage of having a homogeneous and precise representation for processes is that we can deal much more easily with the execution side. The usual situation is that we have a process description on one side, expressed in a given DSL and an execution engine on the other side. To deal with this situation we must usually first elicit the implicit DSL on which the execution engine is based and make it explicit within the same metamodelling framework. Then the two DSLs are compared and a mapping may be defined like the one suggested above between BPMN and BPEL4WS. If later a choice different from BPEL4WS is made, only the mapping has to be changed. Moreover, part of this mapping could be reused if the transformation language has suitable properties.

4 Model-based Process Engineering

Moving from ‘contemplative’ to ‘productive’ model management means that most of the steps in the software production and maintenance chain may be considered as precisely defined operations on model artifacts.

A model-based workbench may be viewed as a software bus on which a number of multiple service tools may be plugged with standard interfaces and protocols. The new idea in this organization is that exchanged artifacts are models, now considered as first-class elements.

As an illustration of the possibilities of such an organization, we provide the following example which shows how metamodelling and model transformation may be used for integrating different tools, and particularly legacy tools which do not belong to the MDA technological space.

Microsoft-Project is a widespread planning tool. It is not based on an explicit meta-model. However, its underlying formalism may be extracted and formulated through the meta-model shown in Figure 2.

MS-Project meta-model is based on the concept of project. A project is described as a set of tasks. Tasks are ordered using links. These are assigned to resources.

Creating a project planning from a process model may then be envisioned as a model transformation between SPEM and MS-Project metamodels. A mapping has been established between concepts from both sides (SPEM process and MS-Project project, SPEM work definition and MS-Project task, etc.). Some more complex algorithms have also been designed for reproducing ordering dependencies. Figure 3 illustrates the net result of part of this project as it appears to the end user.

Actually, the transformation was not so straightforward, as the SPEM process is usually not expressed using a SPEM tool, but with a UML modeler extended using a SPEM UML profile (i.e. a lightweight extension of UML as discussed earlier). A transformation from UML to SPEM had first to be applied. The resulting SPEM model could then be used for producing the MS-Project model.

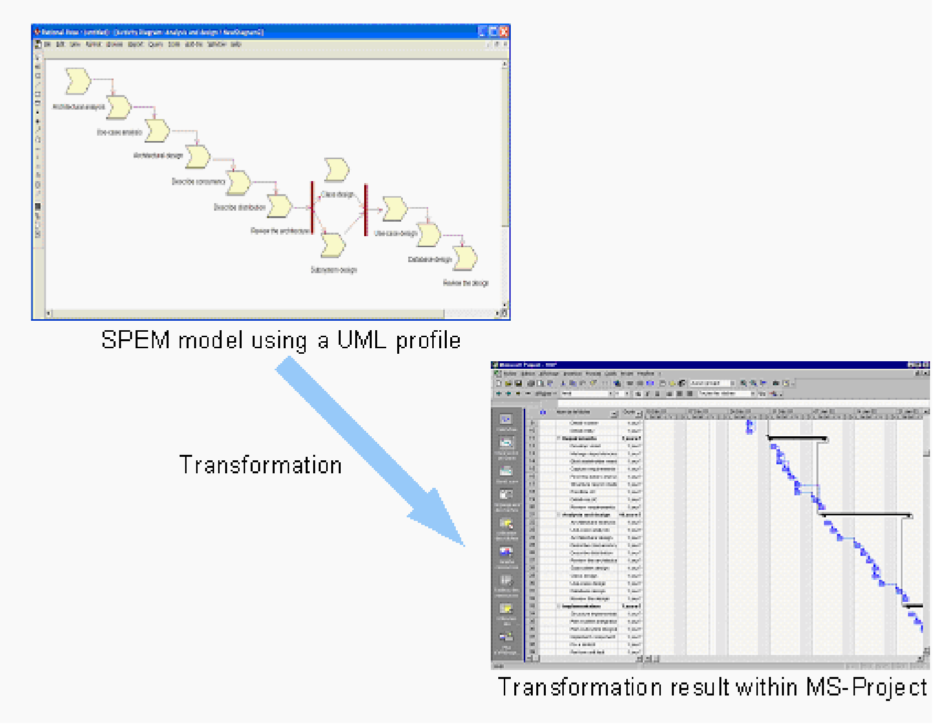


Figure 3: Transformation from SPEM to MS-Project.

Using SPEM as a basis allows the reuse of transformations with different scenarios. A natural way to produce a SPEM process is to use a UML modeler with a SPEM profile. However, there may be some tools producing native (MOF-based) SPEM models. Moreover, we can imagine CAPE (Computer-Assisted Process Engineering) tools with a proprietary formalism. These tools may either propose a SPEM export or a transformation from their proprietary formalism to SPEM has to be provided.

Such an approach has already been used for application management processes in previous projects. From a process model (expressed using a proprietary metamodel because SPEM did not exist at that time), an execution environment was completely generated, including graphical user interface, workflow model and documentation [5].

An important benefit of this approach is that it allows integrating many different points of view through process lifecycle, each point of view being described by a particular meta-model, each transition from one point of view to another by a transformation. Various tools may be integrated since they expose their interface through a meta-model.

5 Conclusion

We have proposed a global method that suggests using model-engineering principles to deal with process definitions. There are several such process DSLs, but they may be related and mapped. Sometimes two process DSLs are similar or one is an extension of another one. In many cases model transformation among model-based process representations has proved to be an interesting idea.

As part of many undergoing projects, the idea of defining a common, open-source MDE platform for various tools is becoming popular. This idea is not new and one may recall projects of the 80's like AD/Cycle (Application Development/Cycle) or PCTE (Portable Common Tool Environment) with similar software buses. However the basic principle that these tools will exchange compatible models, i.e. models conforming to metamodels conforming themselves to common meta-metamodels has proven to be quite powerful in practice. We can envisage a URI-like (Universal Resource Identifier) naming scheme for models like *"model:MMM/MM/M"* where the metamodel *MM* of model *M* is clearly defined as well as the metametamodel *MMM* of *MM*. All the metamodels could be identified in global or local registries called 'megamodels' [3]. A megamodel is a model that contains elements and metadata on the various models, metamodels, services like transformations, tools, etc. For example an explicit semantic relation between the UML and the SPEM metamodel could be expressed in a megamodel. Also the megamodel may contain various metadata on metamodels, models, services, tools, etc. A workflow engine could be registered as a tool for example, referencing its standard model. If we need to execute a process model, we may interrogate the megamodel for a compliant execution engine or we may apply a transformation upon this model. The transformation itself could be applied by another tool linked to the same model-based platform and also described in the megamodel together with its model.

Such scenarios as described in the previous paragraph are becoming realistic in the scope of the present technology. Notice that we do not need to stick to a unique common meta-metamodel as long as the correspondences between the various meta-metamodels are precisely defined. However OMG/MOF and Eclipse/EMF are the two currently aligned industrial dominant standards.

The benefits of such approaches are important. Since everything is a model in this MDE platform, we may include a lot of common facilities like storage and retrieval in model repositories at level M1 and metamodel repositories at level M2, version management, transactional access, etc. Software product and software process models and metamodels will be similarly edited, browsed, stored, retrieved, etc., allowing many economies in tool development cost but also in other areas like user training. We have seen first generation MDE tools (mainly UML tools) with "variable model" capabilities. We are now witnessing the arrival of much powerful tools with "variable metamodel and metametamodel" capabilities. These tools will offer process-engineering or requirement engineering capabilities among others. But what is interesting is that it will be also possible to link to the MDE platform not only MDE tools with extended capabilities, but also legacy tools which were not originally intended to be used in this environment. To integrate such a legacy tool (like MS-Project for example), we have first to define a DSL for the tool, in the form of a standard metamodel like the one of Figure 2, for extraction and injection of proprietary data. From this metamodel, precise guidelines will allow the generation of the injectors/extractors that will enable this tool to be connected to the common MDE platform. From any pair of source/target metamodels we can define syntactic and semantic bridges to convert from one formalism to a similar one or map a process model edited with various kind of editing tool (textual or graphical) onto a given execution engine with a suitable implicit or explicit metamodel.

Model-based process engineering is presently much more than a vision. Based on the three MDE principles (Direct Representation, Automation and Open Standards) and making use of the two basic MDE relations (representation and conformance), it is being deployed in several ongoing industrial projects. However, even if practical problems of product and process model weaving have found ad hoc solutions, more research effort is still needed for a deep understanding of these issues. The development of ambitious open source MDE platform projects hosting simultaneously industrial tools and research prototypes may help to explore still unsolved concrete problems and to improve the state of the art in this field.

Acknowledgements

The work of one of the authors (J. Bézivin) has been supported by a grant from Microsoft.

References

[1]
ATL, ATLAS Transformation Language reference site. <http://www.sciences.univ-nantes.fr/lina/atl/>.

[2]
J. Bézivin. In search of a Basic Principle for Model Driven Engineering, Novatica/Upgrade, Vol. V, N°2, (April 2004), pp. 21–24. <http://www.upgrade-cepis.org/issues/2004/2/upgrade-vol-V-2.html>.

[3]
J. Bézivin, S. Gérard, P. A. Muller, L. Rioux. MDA Components: Challenges and Opportunities, Metamodelling for MDA, First International Workshop, York, UK, (November 2003)., <http://www.cs.york.ac.uk/metamodel4mda/onlineProceedings Final.pdf>.

[4]
G. Booch, A. Brown, S. Iyengar, J. Rumbaugh, B. Selic. The IBM MDA Manifesto The MDA Journal, May 2004. <http://www.bptrends.com/publicationfiles/05-04%20COL%20IBM%20 Manifesto%20-%20Frankel%20-3.pdf>.

[5]
E. Breton. Contribution à la representation de processus par des techniques de méta-modélisation, PhD thesis of the University of Nantes, (June 2002).

[6]
S. Cook. Domain-Specific Modelling and Model Driven Architecture, The MDA Journal, (January 2004. <http://www.bptrends.com/publicationfiles/01-04%20COL%20Dom%20Spec%20 Modeling%20Frankel-Cook.pdf>.

[7]
D. S. Frankell, P. Hayes, E. F. Kendall, D. McGuiness. A Model-Driven Semantic Web Reinforcing Complementary Stengths, The MDA Journal, (July 2004). <http://www.bptrends.com/publicationfiles/07%2D04%20COL%20Semantic%20Webs %20%2D%20Frankel%20%2D%20et%20al%2Epdf>.

[8]
J. Greenfield, K. Short. Software factories Assembling Applications with Patterns, Models, Frameworks and Tools. OOPSLA'03, Anaheim, Ca. <http://www.softmetaware.com/oopsla2003/mda-workshop.html>.

[9]
MIA-Software MIA-Transformation Tutorial. <http://www.model-in-action.fr/download/transformation/3.2.0/tutorial_en.pdf>.

[10]
L. Osterweil. Software processes are software too. ICSE'87, Monterey, Ca.

[11]
R. Soley & the OMG staff MDA, Model-Driven Architecture, (November 2000). <http://www.omg.org/mda/presentations.htm>.

[12]
S. A. White. Introduction to BPMN BPTrends, (July 2004). <http://www.bptrends.com/>.

Software Process Modelling Languages Based on UML

Pere Botella i López, Xavier Franch-Gutiérrez, and Josep M. Ribó-Balust

A Software Process Model (SPM) is a description of the structural and behavioural aspects of a process in the field of software development using some Process Modelling Language (PML) as description formalism. In the last 15 years, process modelling and, particularly, software process modelling, has gained a growing importance as a mechanism which allows, on the one hand, a better understanding of the process, which facilitates its assessment and improvement; and, on the other hand, the ability to automate to a certain extent its enactment, as it is usual in other engineering fields. A fundamental challenge concerning SPMs is how to find a standard PML to describe them. For this reason, in the last few years, an important research effort has been made in order to adapt UML (Unified Modelling Language) to the specific requirements of SPMs and, as a result, some UML profiles and metamodels (like SPEM or PROMENADE), which propose software process modelling formalisms based on UML, have emerged. In this article we outline the state of the art in the subject of software process modelling, we present its challenges and we focus specially in the use of UML as PML.

Keywords: Process Modelling Language, Software Process Model, UML, UML Extension.

1 Introduction: The Issue of Standardization in SPM

Software Process Technology has emerged in the late eighties, but in two fronts. One of them, starting from the seminal work of Leo Osterweil (*Software processes are software too*) [7] has been named Software Process Modelling (SPM), and has been very active from the research point of view, with a lot of contributions in general or specialised conferences and magazines, but with a relative small impact in the industrial world. The main goal of SPM has been the design of Process Modelling Languages (PML) able to describe software processes in a formal way. The other front refers to Software Process Assessment (SPA) and Software Process Improvement (SPI). It comes from the pioneering works by Watts Humphrey [4] developed at the SEI (Software Engineering Institute) and which has led to the definition of the CMM (Capability Maturity Model).

With a strong impact in the industry, this front has evolved with new models, as the CMMi (Capability Maturity Model integration) family, the ISO standard SPICE and other models. It has become an established industrial practice. Theoretically speaking, SPM is a good basis for SPA and SPI, since a formally described process will be easier to assess and, then, to improve. But in real life, all the SPA&SPI models do not need a formal description as a starting point. This fact, together with the proliferation of languages and notations that have emerged to describe software processes [2][1], are probably the reasons for the low impact of the SPM research. Probably, the first step to be taken so that the software industry may benefit from the SPM research is to standardize PMLs.

Pere Botella i López is Full Professor at the UPC (*Universitat Politècnica de Catalunya*), Barcelona, Spain. He has been active in the software engineering field from more than 25 years. Has been Dean of the Faculty of Informatics (1992-1998) and Vice-rector of the UPC (1982-1986, 1998-2002). From 1989 to now, Director of the Master program on Software Engineering at the UPC. Author of more than 40 publications. Program committee member of several international conferences, including ESEC, ICSE, RE, etc., being executive chair and co-editor of ESEC'95. Member of the Steering Committees for ESEC and JISBD (the spanish main event on software engineering). He has been coordinator in Spain for RENOIR (European Network of Excellence in Requirements Engineering. <botella @lsi.upc.es>

Xavier Franch-Gutiérrez is Associate Professor in the Software Department (LSI) at the UPC (*Universitat Politècnica de Catalunya*), Barcelona, Spain. He received his BSc and PhD in Software from the UPC. He is and has been a principal and co-investigator of several funded research projects. He is currently leading the research group in Software Engineering for Information Systems (GESSI) at the LSI, <http://www.lsi.upc.es/~gessi/>, compound of more than 10 full-time researchers. His current lines of research include requirements engineering, selection of COTS components, quality model construction and software process modelling. He has over 40 refereed publications in conferences, journals and books. <franch@lsi.upc.es>

Josep M. Ribó-Balust holds a BSc degree (1990) and a PhD (2002) in Computer Science, both from UPC (*Universitat Politècnica de Catalunya*), Barcelona, Spain. His thesis work defined PROMENADE as a UML-based software process modelling language. Since 1997 he is a member of the research group in Software Engineering for Information Systems (GESSI) at the UPC. He also works as a Lecturer at the Computer Science Dept. of the *Universitat de Lleida*, Spain. Currently, his research interests address the design of methodologies for the correct definition and extension of metamodels. <josepma@eps.udl.es>

One of the main requirements for a PML regarding standardization is that it should rely on a notation and semantics which are standard in the software industry, so that the chosen PML can be used without the burden of having to learn yet another notation and another software tool and also with the possibility to make it easily understandable to other software engineers.

UML (Unified Modeling Language) seems to be a natural candidate for such a standard process modelling formalism since it has become a standard *de facto* in the modelling of object-oriented systems and an important trend among the researchers of the field is to consider a software process itself as a piece of software [7].

The question that arises at this point is whether UML is able to deal with the specific requirements of SPM. The research on this topic began in the late nineties, in which several research groups tested the capabilities of UML as a modelling formalism in the related fields of software and workflow processes. A preliminary result of this research is that, although UML seems to be powerful enough to address the structural aspects of a process, it lacks some degree of expressiveness and flexibility in order to model its behavioural part (specially in the case of software processes in which the flexibility requirements are usually more important).

Therefore, UML should be adapted somehow in order to model software processes: not only should we add new constructs to deal with some behavioural issues but we should also introduce in the language the vocabulary of the field (*activities, artifacts, agents, tools, roles*, etc.) and its specific features. This language adaptation is usually referred to as *UML extension*.

In this article we introduce which mechanisms are provided by UML in order to extend the language and how an extended UML may be used in order to model the structural and behavioural aspects of a software process. Finally, we will outline some examples of proposals for using UML as a PML for software processes.

2 UML Extension Mechanisms

We have already introduced the necessity to extend the UML language in order to model software processes. There are two different ways in which we may do this:

2.1 By Means of An Explicit Extension of The UML Metamodel (Heavyweight Extension)

The UML metamodel [11] contains the definition of the elements that are used in a UML model (e.g., *class, association, dependency, generalization,* etc.). For instance, Figure 1 shows two UML classes (*Company* and *Employee*) and an association (*works-for*) between them. The precise definition of what a class and an association are is stated in the UML metamodel. In particular, the definition of the concept of *class* is given by means of the element of the UML metamodel called *Class*, whereas the notion of *association* is described by means of the element *Association* of the metamodel. The elements of the metamodel are called *metaelements* (e.g., metaclasses and metaassociations). Any UML model is an instance of the UML metamodel, which means that any element that comes up in a UML model is an instance of some metalement (e.g., *Company* is an instance of *Class* and *works-for* is an instance of *Association*).

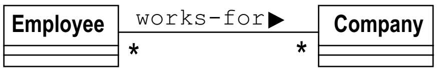


Figure 1: A Simple UML Model.

The ability of UML to model software processes may be enhanced by adding to the UML metamodel new metaclasses and metaassociations between them. For instance, if we need the concept of *Activity* in order to model specific tasks to be carried out during software development, we may add to the UML metamodel a metaclass for that purpose. In the same way, we may add other metaclasses to model the notions of *Role* (a specific responsibility within the process, e.g., *test engineer*) and *Document* (a product which is developed in an activity and which may be used in other ones) and some metaassociations to describe the relationships between these metaclasses (e.g., an activity generates a document; a role is responsible for an activity and also for a document). Figure 2 shows a fragment of this UML metamodel extension.

As a result, a software process model may be created with elements which are instances of the metaclasses *Document*, *Activity* and *Role*.

The main advantage of this alternative is that it constitutes a very elegant and powerful extension approach. A SPM built in this way is a proper instance of a well-formed UML extended metamodel. However, this approach does not result in UML-compliant models (i.e., they are not instances of the UML metamodel, but of an extended metamodel). This issue challenges standardization. For instance, how do we represent in a SPM instances of the new metaclasses in a UML-compliant way? How do we represent in the model the new features introduced by the metamodel (e.g., the role which is responsible for a specific activity)?

2.2 By Means of UML Profiles (Lightweight Extension)

A UML profile is an adaptation of the existing UML metaclasses by means of the built-in extension mechanism of UML so that they can fit in a specific domain (such as SPM). It is important to notice that UML profiles do not provide a first-class extension mechanism in the sense that they do not modify

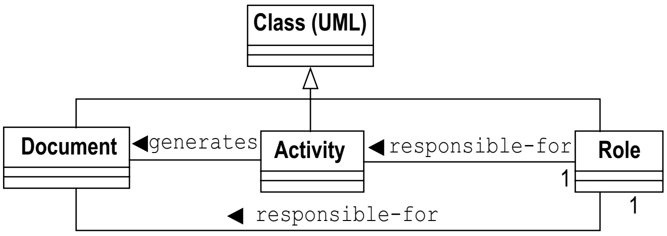


Figure 2: A Fragment of An Extension of The UML Metamodel.

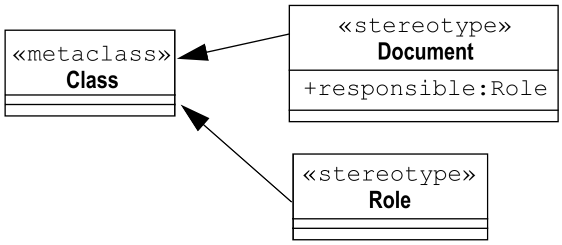


Figure 3: Stereotype Definition.

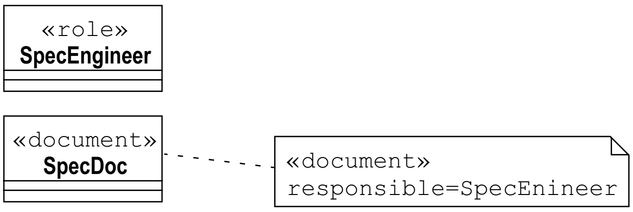


Figure 4: A UML Model That Uses Stereotypes.

the UML metamodel. This specific extension mechanism provided by UML is based on the notion of *stereotype*. A stereotype defines how an existing metaclass may be adapted/ extended. This extension may involve some of the following: a new name for that extended metaclass, new properties for that metaclass and/or new constraints.

UML supplies a notation to depict the extension provided by stereotypes in a model. For instance, Figure 3 shows the definition of the stereotypes *Document* and *Role* which extend the metaclass **Class** (the notation introduced by UML 2.0 has been used). The stereotype *Document* adds the property *responsible* of kind *Role*. As a result, it is possible to define in a specific SPM a class *SpecDoc*, with stereotype *Document* (which means that *SpecDocument* is a kind of *Document*) which has *SpecEngineer* as value of the property *responsible* (which means that the role class that is responsible for the documents of class *SpecDocument* is *SpecEngineer*). Figure 4 shows how this situation can be represented in a UML class diagram.

Since UML profiles do not modify the UML metamodel and follow the UML built-in extension mechanisms, they provide a full UML compliance. However, stereotypes do not have the same semantics as actual metaclasses and their expressiveness is poorer. Finally, this approach does not supply a representation for the extension at the metamodel level, which impairs the comprehensibility of the resulting model and of the profile itself. The notation introduced by UML 2.0 to define profiles contributes to reduce this comprehension problem but it does not eliminate it.

As suggested in the UML 2.0 infrastructure document, it is an ongoing matter of discussion whether to use the *heavyweight* or the *lightweight* approaches in order to extend the UML metamodel to fit it in a specific domain.

3 Modelling The Structural Part of A SPM

UML is a language intended to model object-oriented systems. The expressiveness of the object-oriented paradigm has proven appropriate to model the structural aspects of software processes [5, RF99]. Hence, the constructs offered by UML to model structural aspects of systems are globally suitable for this purpose. In particular, UML *class diagrams* may be used for modelling both software process elements and also relationships of different kinds among them.

Software process elements may be defined as stereotyped classes. A different stereotype may be defined for each kind of software process element (documents, activities, roles, etc.). If necessary, some constraints and class properties may also be added to the stereotypes. Indeed, a complete UML profile can be defined. Figures 3 and 4 above show examples of the definition of some stereotypes that could take part in a profile intended to model software processes and also the use of one of those stereotypes in order to define a couple of classes of a specific SPM.

UML associations, generalizations and dependencies may be used to model the required relationships among classes. For the sake of an example, we mention here one specific structural requirement of software processes: *composition* of documents and activities.

Activities (also documents) may be atomic or composite. Composite activities are constituted of *smaller* (less complex) subactivities. In order to complete a composite activity, its subactivities should be carried out in a specific manner (which is stated in the behavioural description of the process; see Section 4). In their turn, these subactivities may be further decomposed in even smaller ones. This decomposition process can take place as many times as necessary. This can be modelled in UML by means of (possibly stereotyped) composite aggrega-

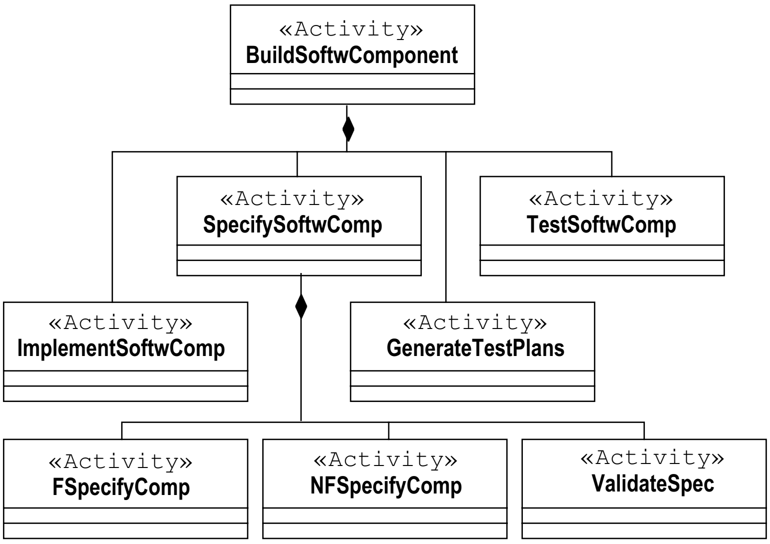


Figure 5: Activity Composition.

tions. Figure 5 shows how to model the process in which an activity class *BuildSoftwComponent* may be decomposed into the subtasks of *ImplementSoftwComp*, *SpecifySoftwComp*, *GenerateTestPlans* and *TestSoftwComp*. In its turn, *SpecifySoftwComp* is further decomposed.

Another interesting relationship between documents or activities is *refinement*, which can be modelled in UML through inheritance. Other relationships between software process classes may be modelled by stereotyping the general-purpose *dependency* UML relationship.

4 Modelling The Behavioural Part of A SPM

A SPM should describe, in addition to the software process structural aspects, the behaviour of such a process. This behaviour may be described using two different paradigms, namely *proactive control* and *reactive control*. Proactive control states the enactment of process activities according to a pre-established plan. Reactive control, in its turn, is based on describing the enactment of some actions as a response to events due to the occurrence of some condition.

PMLs should be expressive enough to offer both types of behaviour description. Let us show how UML may deal with both controls.

4.1 Proactive Control

There exist different approaches that a PML may follow in order to support proactive control. Probably, the most popular one consists in providing imperative descriptions of which activity is to be enacted after the end of a specific one. This approach can be modelled in UML by means of activity diagrams, in which a *transition* from activity *a* to activity *b* is triggered whenever activity a finishes its enactment. This transition grants the permission to start *b.*

Although this approach is possible, it often leads to a control flow description in terms of a set of activities which are enacted either following a strict sequence (possibly with conditions and loops, which have been incorporated to UML 2.0) or in a concurrent way (no other policy in between is allowed).

Usually, modelling software processes requires more flexible and expressive constructs: software developers tend to perform several activities at the same time and move from one to another depending on their interactions with the rest of the team or on some specific requirements (for instance, a deadline).

This reflection leads to other approaches to model the proactive behaviour of a software process. A popular one is the representation of different forms of precedence requirements between activities [8, 9]. For instance, instead of modelling something of the sort: *the implementation of a component will start once its specification has finished*, it seems better to overlap both activities to a certain extent: *the implementation of a component should begin some time after the starting point of its specification and should finish after the end of its specification*.

The second form of modelling uses *precedence relationships*, instead of pure transitions in order to describe behavioural relationships between tasks. A precedence relationship is stated between a set of source activities and another set of target ones and establishes in a *declarative* way which requirements concerning the state of the source activities are needed in order to start/finish the enactment of the target ones. Usually, the second modelling approach (using *precedence relationships*) provides not only a higher degree of freedom when enacting it, but also a more expressive, realistic, and less strict model.

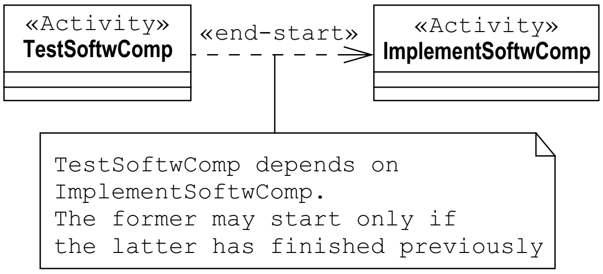


Figure 6: A Precedence As A Dependency.

The problem with precedence relationships is that UML does not support them in a direct way. Hence, a UML extension is necessary if such relationships are to be incorporated into the PML. A precedence relationship may be modelled in UML as a special kind of dependency between activities (see Figure 6).

4.2 Reactive Control

An expressive model for a software process should describe, not only the proactive plan that the process is supposed to follow during its enactment, but also its reaction to certain events that may occur during such enactment (e.g., a notification is received from the supervisor to abort the process or to start at once a specific activity).

The *reactive control* of a PML is responsible for expressing the behaviour of activities as a response to events. UML offers several powerful reactive elements that may be used for this purpose (i.e., *state machines*, *events*, *signals*, *actions*, etc.). However, it is quite usual that current PMLs in the field of software processes use some elements to model reactive control which are not explicitly defined in UML. This is the case of Event-condition-action (ECA) rules, which have become very popular in the context of PMLs in order to model reactive behaviour [2]. These rules establish the execution of an action as a response to an event in the case that a certain condition holds. ECA rules may be associated to model entities (e.g., activities). The introduction of ECA rules in a UML-based PML, or other reactive elements, may lead to an extension of

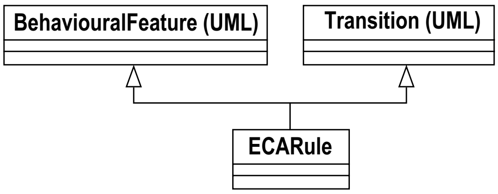


Figure 7: Definition of ECA-rules in PROMENADE (Fragment).

the UML metamodel. For instance, a UML-based PML called PROMENADE (see Section 5 for an outline of it) provides a definition of ECA rule both as a (UML) behavioural feature (so that it can be associated to an entity as one of its features) and as a (UML) transition (hence, it inherits the reactive behaviour modelled by transitions); see Figure 7.

5 Some Proposals To Model Software Processes with UML

Although most of the PMLs in the field of SPM that can be found in the literature are not based on UML, in the late nineties some approaches investigated the use of UML as a modelling formalism in the related fields of software and workflow processes. These approaches did not made an entire proposal of a PML but just focused on some of its elements. They shown some drawbacks of activity diagrams and proposed some alternatives to describe process behaviour. Some of them were quite unnatural (e.g., by means of class diagrams with stereotyped associations for showing the control and data flow [5]).

The next step was to propose an extension of UML to describe software processes. As we have seen, this can be achieved either by defining a UML profile or by means of a heavyweight extension.

The approach of defining a UML profile has been followed in the own UML definition (in versions prior to 2.0 [11]), which define a profile for software development processes. However, this profile is quite basic, since it only defines some stereotypes, no properties and almost no additional constraints. On the other hand, the defined stereotypes do not seem to be sufficient to deal with the terminology introduced by SPMs. Even more, no behavioural aspects are mentioned in this definition.

In the last few years, two new UML-based proposals (namely, SPEM [10] and PROMENADE [8, FR03]) have come up sharing virtually the same innovative definition approach: they are defined as heavyweight extensions of the UML metamodel but they overcome the limitations suffered by these kinds of extensions (namely, not full UML-compliance and, thus, no standard languages) by offering a transformation of the extended metamodel into a UML profile. This seems to be a good solution to keep the best of both worlds: while the heavyweight extension ensures a proper language definition at the metamodel level, its transformation to UML profile guarantees conformance with UML. Apart from this coincidence in the metamodelling approach, both proposals turn out to be quite different.

SPEM is an adopted specification of the OMG done by several companies with the purpose of modelling a family of software development processes (namely, RUP, SI Method, OPEN, etc.). A process in SPEM is defined by means of *iterations*, *phases*, *activities* and *steps*. Activities are not normally decomposed beyond steps, which are atomic (therefore, only two-levels of activity-subactivity decomposition are normally considered). Control-flow in SPEM is based on dependencies between activities (however, only *finish-start* and *finish-finish* dependencies have been defined). SPEM does not define reactive control-flow constructs. As a consequence of these features, SPEM does not seem to be designed to describe detailed processes.

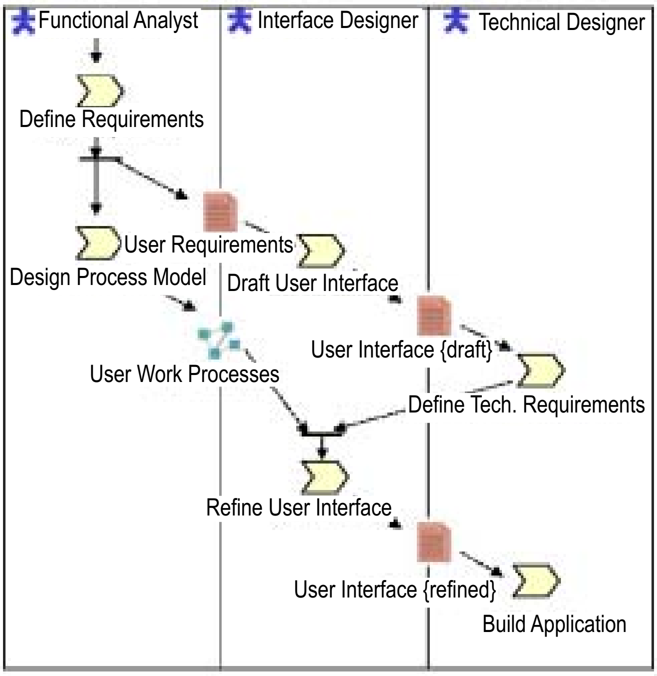


Figure 8: Use of Activity Diagrams in SPEM.

When the extended metamodel is transformed into an UML profile, some stereotypes like *Step*, *Guidance*, *Activity*, etc. do come up. Each of these stereotypes are given a suggested notation and may be used within usual UML diagrams in order to model the various aspects of a process. For example, Figure 8 shows an activity diagram to describe the behaviour of a process fragment.

PROMENADE is a PML which has been defined with the objectives of improving expressiveness, flexibility, standardization and modularity in SPM construction. Remarkably, it allows the modelling of complex and/or detailed software processes in an expressive way by providing both proactive and reactive ways to express model behaviour. Proactive control relies on the notion of *precedence relationship*, which has been defined in the framework of UML as a special kind of dependency. Different families of such precedence relationships have been defined. Furthermore, the modeller may define his/her own new kinds of precedence relationships. In its turn, reactive control is based on ECA rules. Figure 6, above, shows a very simple example of a UML-compliant representation of a PROMENADE precedence relationship between two activities.

6 Conclusion

One important challenge for the software process modelling research community is the development of standard notations to describe software processes. In all probability, such standardization will contribute both to disseminate the research results in software process technology all over the software engineering community (including, software industry) and to incorporate the formalization of software processes into the

SPA&SPI initiatives. UML seems a good choice for such a standard notation since it has become a standard *de facto* in the modelling of object-oriented systems. While UML offers enough expressiveness to model structural aspects of software processes, some authors argue that more constructs should be added to the language in order to model behavioural ones (e.g., proactive and reactive aspects). For this reason (and also to define properly the specific concepts of software processes in this language) UML-based PMLs should extend UML. Two possibilities do exist to do that: the *lightweight* and the *heavyweight* extensions. The former defines a UML profile and the latter, performs an explicit extension of the UML metamodel.

Several proposals have been defined in the last three or four years in this sense: from quite basic UML profiles to more sophisticated approaches like SPEM or PROMENADE.

It is worth mentioning that the use of UML as the basis to construct PMLs will lead to standardization in the sense that the defined PML will rely on a well-known syntax and semantics, and it will be possible to use widespread UML-based tools in order to model software processes. However, it is also true that different PMLs may use UML in different ways (e.g., they may use different constructs or the same constructs to model different aspects). In any case, the use of UML will constitute a qualitative improvement regarding SPM standardization.

References

[1] J. - C. Derniame, B. A. Kaba, D. Wastell (eds.). Software Process: Principles, Methodology and Technology. Lecture Notes in Computer Science, Vol. 1500. Springer-Verlag, Berlin Heidelberg New York, 1999.

[2] A. Finkelstein, J. Kramer, B. Nuseibeh. Software Process Modelling and Technology. Advanced Software Development Series, Vol. 3. John Wiley & Sons Inc., New York Chichester Toronto Brisbane Singapore, 1994.

[3] X. Franch, J. M. Ribó. Using UML for Modelling the Static Part of a Software Process. In Proceedings of UML '99, Forth Collins CO (USA). Lecture Notes in Computer Science (LNCS), Vol. 1723, pp. 292–307. Springer-Verlag, 1999.

[4] W. S. Humphrey. Managing the Software Process. SEI Series in Software Engineering. Addison Wesley, 1989

[5] D. Jäger, A. Schleicher, B. Westfechtel. Object-Oriented Software Process Modelling. Proceedings of the 7th European Software Engineering Conference (ESEC), LNCS 1687 Toulouse (France), September 1999.

[6] X. Franch, J. M. Ribó. A UML-based Approach to Enhance Reuse within Process Technology 9th International Workshop, EWSPT 2003, Helsinki, Finland. Lecture Notes in Computer Science (LNCS), Vol.2786. Springer-Verlag, 2003.

[7] L. Osterweil. Software Processes are Software Too. In Procs. of the Intl. Conf. on Software Engineering (ICSE-9), 1987.

[8] J. M. Ribó, X. Franch. A Precedence-based Approach for Proactive Control in Software Process Modelling. In Proceedings of the SEKE-2002 conference (Software Engineering and Knowledge Engineering). ACM Press. Ischia (Italy). September, 2002.

[9] C. Schlenoff, M. Gruninger et al. The Process Specification Language (PSL) Overview and Version 1.0 Specification. NIST Internal Report (NISTIR) 6459, 1999.

[10] Software Process Engineering Metamodel, version 1.0. Adopted specification formal/2002-11-14 of the OMG (accessible at <http://www.omg.org>).

[11] Unified Modelling Language, version 1.5. Adopted specification formal/03-03-01 of the OMG (accessible at <http:www.omg.org>). Also, UML2 Infrastructure Final Adopted Specification and UML 2 Superstructure Final Adopted Specification (accessible at <http:www.omg.org>).

Supporting the Software Process in A Process-centred Software Engineering Environment

Hans-Ulrich Kobialka

In software projects there exist tools, working schemes, and collaboration. Turning such an environment into a Process-centred Software Engineering Environment (PSEE), involves: 1) Augmenting the process with an additional flow of information and a dedicated user interface. 2) Supporting wanted process steps. 3) Disabling unwanted process steps. The reason for this is that it is not practical to claim that all activities in software projects have to be completely defined and supported by a PSEE. Instead, process support has to be introduced incrementally. This paper illustrates how this can be achieved.

Keywords: ALADYN, Process Modelling Language (PML), Process-centred Software Engineering Environment (PSEE), Software Process Enactment, Software Process Support.

1 Introduction

A software process encompasses a significant number of steps performed by many people. Several years ago it was stated that software processes should be described in a formal way by using Process Modelling Languages (PMLs), and that models written in a PML can be enacted in Process Centred Software Engineering Environments (PSEEs) [13].

Current PMLs/PSEEs cause difficulties when used during process enactment. This is because many PMLs/PSEEs have adopted approaches from specification and programming languages which were not designed to adapt to change during execution. Process programs often turn out to be quite complex which users have found difficult to both understand and apply changes to correctly.

Section 2 introduces the requirements which have to be fulfilled for effective process enactment. The concepts of current PMLs/PSEEs are discussed in Section 3. We then propose a number of ingredients which enable successful process support (Section 4). While this is done on a rather general level, in Section 5 we give a more technical view on Event-Condition-Action (ECA) rules and impact control.

2 Requirements of Process Support

Process support raises several challenges which have to be met by a PML, and its runtime environment the PSEE. Here we make some statements and identify the corresponding requirements.

a) Process support will never cover the entire process.

We suspect that support for a non-trivial process will never be complete in the sense that each possible sequence of actions is considered. This may be due to incomplete knowledge about the process, but also because process designers do not want to specify too much detail if there is no equivalent pay back in productivity. Therefore, only parts of the process which need support are specified in the PML – unsupported process steps are enabled but not explicitly written down on the level of process implementation (although they may be contained in handbooks or high-level process specifications).

Requirement 1) The PSEE should empower the user to work with incomplete process support.

It should be possible to perform working steps currently not specified or supported. In the extreme case, users should be able to work without writing any process program in advance. They should be able to build up working environments manually and to pass around results. This requires a number of *services* to be provided by the PSEE, e.g. communication services.

b) Process change during enactment is the normal case, not the exception.

Process support is likely to change during a process. It is usually impossible to define the entire software process support from the very beginning. In most processes only the principal aspects (e.g. some data & control flows) are known, but their volume or frequency cannot be determined in advance.

Software processes “run” for a long time. During this time things may change which again might force the process to be adapted or even changed completely.

Hans-Ulrich Kobialka got his MSc in Computer Science in 1985 at TH (*Technische Universität Darmstadt*), Germany, and his PhD in 1998 at TUB (*Technischen Universität Berlin*) Cottbus, Germany. Since 1985 he worked as a scientist at GMD and since 2001 at Fraunhofer AIS. His research interests include robot control architectures, real time computer vision, and software process support. <hans-ulrich.kobialka@ais.fraunhofer.de>

Requirement 2) The change of a process program has to be a simple and save operation.

Installing a change should be possible at runtime by the push of a button without the risk of any hazardous effects. There should be a means of defining and controlling permissions for process change.

In case a change has not achieved the expected benefits, it should be possible to undo that change easily.

c) Process programs cannot be assumed to be totally correct.

Process programs are difficult to test, because they operate on large databases, interact with many users, and are written by different authors. The impact of changes to process programs are difficult to predict. Thus, when a process program is recognized to be incorrect, the state of enactment (i.e. the states of all current processes) has to be updated as well:

Requirement 3) The user should be able to complement or compensate for the effects of process programs.

The interactive user should be able to perform every action that can possibly be performed by a process program. The service interface of the PSEE has to cater for basic constraints of consistency.

d) The probability of misunderstandings and errors is a key factor.

Although errors should be tolerable, they should be avoided as much as possible. Manual repair actions are time-intensive and can create further errors, especially when carried out by different authors. A certain frequency of errors can reveal a PSEE to be totally unusable in an industrial context.

Requirement 4) The effect of process programs should be easily understood by users.

The *impact* of a statement should be obvious to the reader. In particular, the side effects on other statements should be limited as much as possible. The *size* of the process support to be written should permit parts to be located easily and understanding of the support to be total. The PML should support *abstractions* of the software process domain. Appropriate abstractions may also reduce the size and complexity of process programs.

e) Process support will be more effective if it can be customized for each task.

The characteristics of tasks, and the skills of teams, can differ substantially. Much productivity (and acceptance of process technology) can be gained if a process can be adapted in such a way that these characteristics and skills can be exploited. Although projects interact with each other, they require some *autonomy* in adapting their processes.

Requirement 5) Process support should be adaptable to specific (sub)projects.

The effects of a process support, which has been installed for a particular task, has to be limited to this part of the project.

3 Related Work

PMLs are often classified into net-based, object-oriented or rule (or event-trigger) based approaches. Most PSEEs offer a hybrid PML with multiple paradigms (or multiple PMLs respectively), see Table 1. Due to space limitations, in this paper we discuss the above approaches rather than each individual system.

3.1 Net-based PMLs

Net-based PMLs (Process Weaver [8], Slang/SPADE-1 [1], LEU [9], Endeavors [2], APEL (the PML of the Adele PSEE) [6], Little-JIL [4]) have several advantages: they can be used to describe a process in an intuitive way, they can be analysed and simulated, and in most cases the impact of their elements (states, transitions) can be easily understood.

Net-based PMLs permit a transition to execute some code which may access any database and communicate with concurrent transitions or remote services. In these cases, the models are no longer genuine Petri-nets, because the latter requires the firing of a transition to depend solely on its inputs. Thus, simulation and impact analysis can become difficult as databases and communication are used in a concrete net.

The main drawback of net-based PMLs is that users tend to specify only the main workflows. Unexpected situations are difficult to handle if a net state has to change during execution. Some PSEEs (e.g. SPADE-1) permit changes at runtime, but this puts the user in the position of editing a net, i.e. still having to plan steps before executing them. If tasks have to be created and connected dynamically, net-based models offer no appropriate execution model.

3.2 Object-oriented Approaches

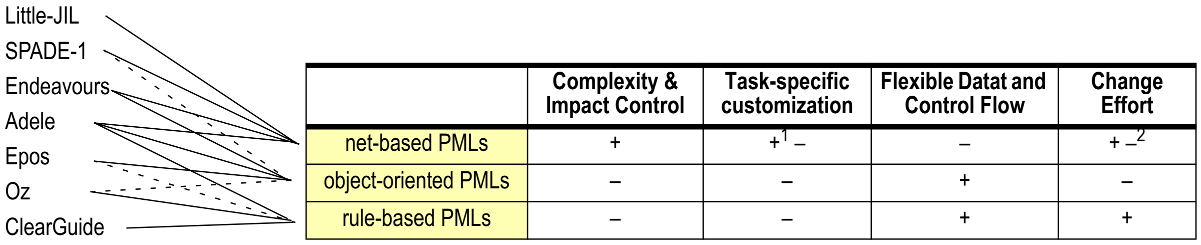
Object-oriented PMLs (EPOS [5], Adele) offer more flexibility, as they allow users to create and connect tasks (activities) dynamically (incremental evolution of task networks). Dynamic instantiation is also used in non object-oriented approaches, e.g. DYNAMITE, ClearGuide [12].

Many object-orientated approaches (e.g. EPOS, Adele, Endeavors, SPADE-1) make intensive use of the definition of object-oriented class hierarchies. We argue that this is not the right approach to implement and change process support during enactment because the units of change are schemas (type hierarchies) which tend to become quite large and contain complex dependencies. Therefore, schema updates can be difficult (especially if existing objects can be invalidated), and the transfer of improvements from one schema to a similar one is not trivial.

3.3 Rule-based PMLs

Rule-based and event-trigger based approaches are very powerful. However process descriptions tend to become complex, especially if rule chaining (e.g. in Marvel/Oz [3]), or other kinds of side effects between rules/triggers are used. APEL and ClearGuide do not encourage the user to use rule chaining. Usually side effects occur between rules.

Multiple processes. Different process instances can coexist and cooperate within most PSEEs. If rules of different process-



	Complexity & Impact Control	Task-specific customization	Flexible Datat and Control Flow	Change Effort
net-based PMLs	+	+[1] –	–	+ –[2]
object-oriented PMLs	–	–	+	–
rule-based PMLs	–	–	+	+

1. Task-specific support could be possible, if the PSEE allows to use different variants of a (sub)net within different tasks. Not all net-based PMLs allow this.
2. The effort can be reasonable if several instances of the net are currently existing. Each instance has to be stopped and investigated whether the placement or contents of the tokens conflict with the change of the net.

Table 1: The Strengths and Weaknesses of Different PML Paradigms.

es (possibly written by several authors) interfere or conflict, most PSEEs do not consider who has installed the conflicting rules. Only ClearGuide knows during execution of a rule, for which task this rule has been defined.

Control of Permissions. Context information (the current user, the task he is currently performing, and in the case of a rule the task/process for which it is defined) can be used for conflict resolution and access control. As PSEEs offer only limited context information, their ability to control permissions is also limited.

4 Ingredients for Process Support

Here we propose a number of ingredients for the design/ architecture of PMLs and PSEEs in order to enable effective process support.

4.1 PML = Service Interface

As process steps should be carried out concurrently by process programs there has to be some service which is accessed by multiple processes and which coordinates their activities. If this service interface is also accessible via a command interface, or a graphical user interface (GUI), and if this interface offers an appropriate abstraction level, it fulfils requirement 1: the user can perform process steps even if a process programs is incomplete or does not exist. This also meets requirement 3: in case of an incorrect/incomplete process program, the user can manually repair the state of enactment.

The interface of process service defines the PML. The process server acts as an interpreter of this PML. The remainder of Section 4 considers the question “What are the right services for process support?”.

4.2 Inter-related Process Objects

The PML services should provide basic concepts of processes, tasks, users and communication. It should be possible to dynamically create/connect/delete instances of these objects thereby managing networks of interrelated objects.

Process support should offer some notion of task (also called activity) and relationships between tasks. A special relationship is the task-subtask relationship: a task is divided in several subtasks; the manager of a task also heads (is the managers of) the subtasks. Users are assigned to tasks, thereby fulfilling special roles (e.g. manager). There should be some communication infrastructure e.g. one which permits a message to be sent to the manager of task X rather than to a specific user.

A task may produce documents as results. These documents may again be associated as inputs to other tasks. Documents are typically stored in some configuration management system (e.g. CVS, ClearCase, Adele).

4.3 Process Definition on the Instance Level

In a network of interrelated objects, additional process knowledge has to be added e.g. process procedures and constraints. Also the process can be parameterized using process variables, similar to environment variables e.g. a process variable may simply denote which editor should be invoked by default.

We propose that process variables, procedures and constraints should be contained in process programs which are associated with task objects.

Note that there is no need for type hierarchies, or other kinds of dependencies on the type level, which make change difficult. Processes are defined on the instance level: tasks are instantiated and specific process knowledge is added to these instances by assigning process programs to them.

4.4 Process Permissions

The creation of tasks, subtasks, and relationships to other objects is a privileged operation. The same holds for associating a process program to a task. Such operations need special permission.

We propose that such operations can only be performed by users who play a certain role (“manager”) in a task. Of course, a manager can only perform privileged operations on his task and its subtasks. Because process procedures and constraints are associated to a task by its manager, they are performed with the same permissions (as if these actions were performed by the manager manually).

4.5 Process Automation

Active mechanisms, like the Model-View-Controller pattern, Event-Condition-Action (ECA) rules, triggers, or similar mechanisms e.g. in ClearGuide or APEL, have many advantages

- They are very powerful and flexible. E.g. it is very time-consuming to document all possible action sequences which may be caused by a few ECA rules.
- They decouple the event source from the action. There is no need to modify the event source if an ECA rule is added, modified or removed.
- They are quite robust with respect to side-effects from other components, *if they are written carefully.* ECA rules do not know when they will be executed. Other rules associated with the same event may be executed before, possibly modifying/deleting objects. An ECA rule has to check all its assumptions first before performing any action. In contrast to this, the 'sequential programming' paradigm is more sensitive to change. For example in the sequence "A; B; C;" (or "A→B→C"), statement C usually contains assumptions about the previous statements. So when changing statement A, all subsequent statements are possibly affected.

Constraints

Elaborated process services and active mechanisms offer a very powerful framework. But not every possible sequence of actions is desirable. Manual detection and repair of unwanted states is not feasible in an automated environment. So if a user issues a command which violates consistency constraints, this has to be aborted on the database level as if it was never executed.

The principle behind this is that there is some model defined which initially offers a great deal of freedom. As more knowledge about the process is obtained over time, that freedom can be restricted to exclude unwanted actions. According to Peter Wegner: "*Constraints are more powerful because they are applicable to non-compositional behaviours. Michaelangelo's sculptures realized by chipping marble slab could not have been realized by gluing together small bits of marble.*".

5 Process Automation with ALADYN

The above proposed ingredients for process support are described on a rather general level. In order to give a more concrete impression of how process support may look like, we describe the process automation and impact control in a concrete system. Here we refer to a PSEE named ADDD, which offers a PML called ALADYN [10].

Besides the organization of work, the task hierarchy is used to organize process descriptions, called ***policies***. Multiple aspects are grouped and treated together in a policy: *parameters* and *procedures* used for customization, and *triggers* used for process monitoring, notification, automation and consistency control.

A policy can be instantiated for several tasks (Figure 1). ***Instantiation of a policy*** means instantiation of all its parameters, procedures, and triggers for the denoted task i.e. the contents of the policy file is parsed, and objects are created and inserted into system tables. A policy, which is instantiated for a particular task, influences the execution of this task and its subtasks.

For a task at most one policy can be instantiated. The instantiation of a policy for this task implies that the parameters, procedures, and triggers of a former instantiated policy are removed from the system.

5.1 Triggers and Constraints

Instantiated triggers are Event-Condition-Action (ECA) rules as used in active database systems. They are used to implement reactive behaviour.

In the trigger syntax, the concerned tasks are specified by a task pattern.

Task Patterns

A task has a pathname which is generated from its name and the names of its parent tasks in the task hierarchy. Task pathnames are similar to those of files in a file system, e.g. /projectA/Test/test_release.2.5. Task patterns may contain wildcards e.g. as known from the Unix shell. For example, the pattern *test* may be used to match all test tasks in the system.

However, general pattern matching is not selective enough. Usually a trigger should be applied to tasks located in the context of the task for which the policy is instantiated.

Therefore, a pattern can contain one of the keywords SELF, PARENT, or PROJECT at the beginning. These keywords refer to the task for which the policy is instantiated (SELF), its parent task (PARENT), or the top-level task containing it (PROJECT). Each time an event is matched against the trigger, the keyword is replaced by the pathname of the denoted task. For example, if a policy is instantiated for the task /projectA/ Test and one of the policy's triggers has the task pattern SELF*, the pattern is expanded to /projectA/Test* (i.e. the pattern matches /projectA/Test and all its subtasks). Similarly, a trigger containing PARENT* looks at the task /projectA and all its subtasks.

"Happened on" versus "Caused by"

During command execution a task can be in two positions. First, it can be the active task on behalf of which the user issues a command and implicitly *causes* events. Second, a task can be a passive object which is modified by a command and on which events are *happening*.

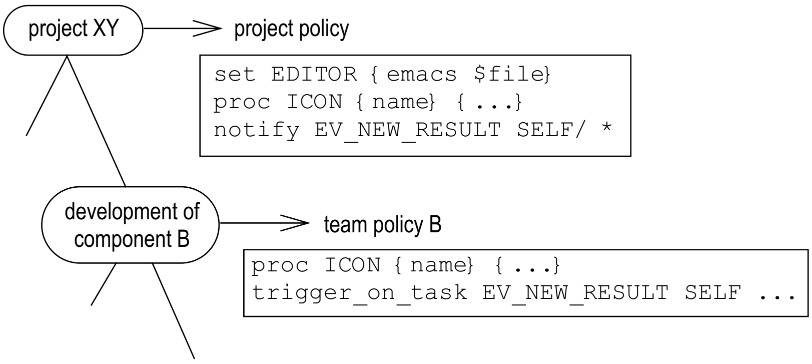
ALADYN distinguishes between (1) triggers which wait for events happening on a particular task (i.e. the task is modified), and (2) triggers observing events caused by a particular task (i.e. the current execution context refers to this task).

```
trigger_on_task EV_NEW_RESULT <task pattern> route_result_to_review
```

This trigger defines that the action route_result_to_review is executed each time when the event EV_NEW_RESULT *happens on* a task matching the task pattern, i.e. when a result of this task is published. This trigger fires regardless of which task has caused the event. Such types of trigger are denoted by the suffix _on_task.

```
trigger EV_NEW_CR <task pattern> handleChangeRequest
```

This trigger says that each time a change request has been created (i.e. *is caused by*) the tasks matching the task pattern,



```
set EDITOR { emacs $file}
proc ICON { name} { ...}
notify EV_NEW_RESULT SELF/ *
```

```
proc ICON { name} { ...}
trigger_on_task EV_NEW_RESULT SELF ...
```

Figure 1: Example of Policies Defined on Different Levels in The Task Hierarchy.

an action (handleChangeRequest) should be executed automatically.

Optionally, the execution of triggers can be controlled by conditions, i.e. the action is only executed if the condition is fulfilled. Conditions and actions of triggers are procedures. When a trigger is defined in a policy file, its condition and action procedures should be written into the same file.

Constraints

In some cases, process support has to restrict certain actions in order to preserve consistency constraints. Usually it is very difficult to undo violations after they have occurred and other users have seen them or triggers have reacted on them. In ALADYN, constraint triggers can be used to abort modifications on the database level (including all triggered procedures). This resets the system to the state before the offending command was issued (i.e. as if it had never been executed).

The constraint and constraint_on_task trigger declarations are similar to trigger and trigger_on_task, except that the triggered action is implicitly “abort”. For instance, the following trigger prevents the tasks matching the task pattern from assigning any policy file to a task:

constraint EV_NEW_POLICY_FILE <*task pattern*>

5.2 Controlling Impact of Process Programs

We illustrate how the impact of triggers, parameters, and procedures is limited to the task for which they are instantiated. This impact can be further restricted by permissions and constraints.

5.2.1 Impact of Process Parameters and Procedures

Process parameters and procedures defined in the policy of a task are inherited by its subtasks. For instance in Figure 1 the ***project policy*** is instantiated for task “projectXY” and defines that the emacs editor should be used. This parameter is inherited recursively by all subtasks of projectXY, i.e. everyone working in this project will use emacs.

On the other hand, parameters and procedures can be redefined in the policies of subtasks. This means that within the execution context of the subtask, the redefined value/implementation is retrieved for a parameter/procedure, while commands executed on behalf of the parent task still retrieve the value defined in its policy. In Figure 1 the procedure ICON is redefined by ***team policy B***. This leads to (some) items being displayed with different icons for people working on B than are displayed for people working on the rest of the project.

5.2.2 Permissions and Constraints

If a process procedure is invoked, it is performed within the execution context of the command, i.e. the procedure is executed with the same permissions as the invoking command.

A user can be assigned to a task either as a manager or a developer. A manager of a task has certain permissions to configure the task and its subtasks (creating subtasks, instantiating policies, assigning people, passing input to a task, etc.), but the manager is *not* allowed to do this for tasks located elsewhere in the task hierarchy. Thus the permissions of managers are limited.

Developers have no management permissions and work with the configuration management service (e.g. checkin/checkout of versions and configurations).

The permissions of managers and developers can further be restricted in a task-specific way by constraints ***if*** these constraints are defined by one of their managers, as explained below.

5.2.3 Impact of Triggers

In contrast to parameters and procedures, a trigger which is instantiated for a task cannot be inherited, redefined or disabled by a subtask. For example, in Figure 1 the ***project policy*** defines that the team members of project XY should be notified whenever one of its subtasks produces any results. This trigger is not implicitly instantiated for subtasks, i.e. task “development of component B” is *not* notified if one of its subtasks produces a result.

The impact of a trigger can be determined first of all by its task pattern. This defines the tasks which are viewed or controlled. Other parts (the event, the condition and the action procedures) also restrict the impact, but the task pattern is usually the most selective part of a trigger i.e. it filters most of the events occurring in the system.

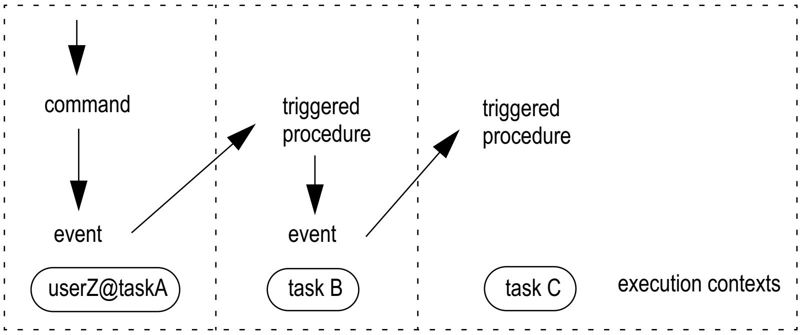


Figure 2: The Execution Context Is Switched for Each Triggered Procedure (the arrows denote the flow of control.)

Permissions of triggered procedures:

A command can raise several events which may cause triggers to be fired i.e. procedures are executed. For each triggered procedure the context is switched i.e. the procedure is executed in the context of the trigger's task (Figure 2).

A policy can only be instantiated for a task by a manager. Therefore the triggers are authorized by this manager and accordingly have the same permissions. For example, a triggered procedure can only abort a command of userZ@taskA if the constraint is declared in a policy of taskA or one of its parent tasks.

5.3 Abstractions appropriate for the Process Domain

A clear understanding of the impact of each statement is an important factor in understanding the whole program. In addition, the size and the complexity of the program are also significant factors.

Beside general ECA rules, ALADYN offers the definition of *specialized triggers*. For example, the following trigger defines that a notification should be sent to the managers of the policy's task each time one of its subtasks produces a result.

```
notify EV_NEW_RESULT SELF/* MANAGER
```

Such events can also be logged in a file:

```
log EV_NEW_RESULT SELF/* $TEMP/taskIO.log
```

Specialized triggers offer abstractions which are appropriate for the process domain. While ECA rules stem from active databases, concepts like notification, logging, and configuration management belong to the process domain. Such abstractions free the domain expert from writing domain-specific solutions in terms (and many lines) of a general-purpose language.

We have implemented support for the ISPW9 scenario [14] in ALADYN [10]. The solution contains 9 triggers which invoke procedures, and 19 "trivial" triggers which do not. Only one trigger causes another "non-trivial" trigger to fire, so programming required some care in this case. This solution has moderate size and complexity, although we have aimed for a comfortable level and not minimal support.

6 Conclusions

Effective process support depends on several ingredients such as high level services, user interfaces to these services, process definition on the instance level, active support for automation and consistency control, clearly defined impact, and control of permissions.

In this paper we discuss a number of requirements, and some problems existing PMLs have in meeting these requirements. We then propose a set of ingredients which we regard as helpful, or even mandatory, to remedy the problems. Because this is done on a general level we then illustrate some aspects by describing process automation using the ALADYN PML.

The ALADYN PML contains no mechanisms to enforce control flow explicitly as can be done with net-based PMLs. Instead ALADYN supports the flow of results and automatic notification, and lets the user decide what to do next. Control flow enforcement requires the programming of ALADYN constraints. An interesting approach is to define control & data flow in a net-based language and then generate PML code for the PSEE to be used. In [7], process descriptions defined in UML were used to generate code for the OPSS PSEE. Such an approach can be used to generate support for the specified process without prohibiting non-specified processes. This combines the benefits of precise specifications with the flexibility of an interpreted PML.

Because of space restrictions, this paper does not discuss how process support can be improved by process-specific user interfaces. For this important point we refer to [11].

References

[1] Sergio Bandinelli, Elisabetta Di Nitto, and Alfonso Fuggetta. "Supporting cooperation in the SPADE-1 environment." IEEE Transactions on Software Engineering, 22(12):841–865, December 1996.

[2] Gregory Bolcer and Richard Taylor. "Endeavors: a process system integration infrastructure." In Proceedings of the 4th International Conference on the Software Process, pages 76 – 89. IEEE Computer Society Press, 1996.

[3] Israel Ben-Shaul and Gail Kaiser. A Paradigm for Decentralized Process Modeling. Kluwer, 1995.

[4] Aaron G. Cass, Barbara Staudt Lerner, Eric K. McCall, Leon J. Osterweil, Stanley M. Sutton, Jr., and Alexander Wise, "Little-JIL/Juliette: A Process Definition Language and Interpreter", In

Proceedings of the 22nd International Conference on Software Engineering, pp. 754–757, June 2000.

[5]
R. Conradi, M. Hagaseth, J. O. Larsen, M. N. Nguyen, B. P. Munch, P. H. Westby, W. Zhu, M. L. Jaccheri, and C. Liu. “EPOS: Object-oriented cooperative process modeling.” In B. Nuseibeh, A. Finkelstein, and J. Kramer, editors, Software Process Modelling and Technology, pages 33–70. John Wiley and Sons, 1994.

[6]
Samir Dami, Jacky Estublier, and Mahfoud Amiour. “APEL: A graphical yet executable formalism for process modeling.” Automated Software Engineering, 5(1):61–96, January 1998.

[7]
Elisabetta Di Nitto, Luigi Lavazza, Marco Schiavoni, Emma Tracanella, and Michele Trombetta: “Deriving executable process descriptions from UML”, In Proceedings of the 24th International Conference on Software Engineering, pages 155–165. ACM Press, May 2002.

[8]
Christer Fernström. “Process WEAVER: Adding process support to UNIX.” In Proceedings of the Second International Conference on the Software Process, pages 12–26. IEEE Computer Society Press, February 1993.

[9]
Volker Gruhn and Stefan Wolf. “Software process improvement by business process orientation.” Software Process–Improvement and Practice, Pilot Issue:49–56, August 1995.

[10]
Hans-Ulrich Kobialka. “Implementing support for software processes in a process-centered software engineering environment.” Ph.D. Thesis. GMD Research Series 15/1998. <http://www.ais.fraunhofer.de/~kobi/img/SupportForSoftwareProcesses.pdf>.

[11]
Hans-Ulrich Kobialka and Claus Lewerentz. “User interfaces supporting the software process.” In Proceedings of the 6th European Workshop on Software Process Technology, Lecture Notes in Computer Science, September 1998.
<http://www.ais.fraunhofer.de/~kobi/img/ewspt6.3.pdf>.

[12]
David B. Leblang. “Managing the software development process with ClearGuide.” In Software configuration management: ICSE 97 SCM-7 Workshop, pages 66–80. Lecture Notes in Computer Science 1235, Springer, May 1997.

[13]
Leon J. Osterweil. “Software processes are software too, revisited.” In Proceedings of the 19th International Conference on Software Engineering, pages 540–548. ACM Press, May 1997.

[14]
Maria H. Penedo. “Life-cycle (sub) process scenario.” In C. Ghezzi, editor, Proceedings of the 9th International Software Process Workshop, pages 141–143. IEEE Computer Society Press, October 1994.

Managing Distributed Projects in GENESIS

Lerina Aversano, Andrea De Lucia, Matteo Gaeta, Pierluigi Ritrovato, and Maria-Luisa Villani

The success of large software projects conducted by different organization sites may be determined by the inter-site coordination and cooperation of the working teams, thus automated support to distributed project management can be useful. In this context we present the GENESIS (Generalized ENvironment for procESs management in cooperatIve Software engineering) approach to distributed process modelling and enactment, realized through an event dispatching architecture whose distinctive feature is a decentralized and autonomous definition of the multi-site software processes.

Keywords: Coordination and Cooperation, Distributed Process, Large Software Projects, Multi-Site Software Projects, Project Management, Process Modelling, Software Processes, Workflow Management.

1 Introduction

Software projects are generally complex and may be distributed across sites, so they require the coordination and cooperation of teams of software engineers from different geographical locations and possibly belonging to different organizations. This scenario is now becoming commonplace as a result of globalization and therefore automated tools for the management and execution of these projects are highly desirable [15][4][10]. In fact, the support for distributed project management is a relevant problem for two reasons.

- projects may involve a big number of concurrent activities, which impose an adequate coordination support to keep them on track and under control;
- when a project spans multiple sites that generally work in a largely autonomous manner, these sites will not necessarily be following the same process models, nor they will be all employing the same methods and tools. Hence, it may be required that, whenever it is possible, parts of the project should be under the responsibility of local project managers,

Lerina Aversano received a BSc in Computer Engineering from the *Università degli Studi del Sannio*, Benevento, Italy, in 2000, and a PhD in Computer Engineering from the same university, in 2003. She is currently an assistant researcher at the RCOST (Research Centre on Software Technology) of the mentioned university. Her research interests include process and information system modelling, workflow management, document management, business process reengineering, software reengineering and migration. <aversano@unisannio.it>

Andrea De Lucia received a PhD in Electronic Engineering and Computer Science from the *Università degli Studi di Napoli*, Naples, Italy, in 1996. He is an Associate Professor of Software Engineering at the Department of Mathematics and Informatics of the *Università degli Studi di Salerno*, Italy. His research interests include software maintenance, reverse engineering, visual languages, workflow management, and document management. He is the program co-chair of the 11th IEEE Working Conference on Reverse Engineering (WCRE 2004) and of the 3rd Workshop on Cooperative Supports for Distributed Software Engineering Processes (CSSE2004). Prof. De Lucia is a member of the IEEE and the IEEE Computer Society. <a.delucia@unisa.it>

Matteo Gaeta is Researcher for Systems of Information Processing and University Teacher of Information Fundaments at the Faculty of Engineering of the *Università degli Studi di Salerno*, Italy. He is Director and Member of Board of Directors of CRMPA (Centre of Research in Pure and Applied Mathematics), a no-profit Consortium between industrial enterprises and the university; he is also Deputy Director of the Centre of Excellence on Learning and knowledge of the same university. He is a member of the Technical Committee by the Italian Minister of Productive Activities. He is a member of the Steering Committee of the Italian Ministry of Agricultural and Forest Policies for the consultancy and the scientific, organisational support to research, development and innovation. <gaeta@crmpa.unisa.it>

Pierluigi Ritrovato is Research and Technology Director of CRMPA (Centre of Research in Pure and Applied Mathematics), *Università degli Studi di Salerno*, Italy. He has participated in, or coordinated, several Research and Development projects either Italian or European on themes relative to: distributed systems and architecture, <http://www.elegi.org>; GRID technologies and architectures, <http://www.eu-grasp.net>; Distributed Software Engineering, <hhttp://www.genesis-ist.org>; application interoperability and cooperation; CSCW, and Internet Technologies. <ritrovatog@crmpa.unisa.it>

Maria-Luisa Villani is a Research Assistant at RCOST (Research Centre on Software Technology) of *Università degli Studi del Sannio*, Benevento, Italy. She graduated (cum laude) in Mathematics at the University of Napoli in 1994. From September 1995 to September 1999 she was at the University of Warwick – England, where she obtained a MSc degree (1996) and a PhD (1999) in Matematics. In September 2001 she got a Master degree in Software Technology at the *Università degli Studi del Sannio* and joined RCOST. Her current research interests are service-centric software engineering, software process technology and modelling, model checking techniques for program verification. <villani@unisannio.it>

who can better organize the local activities (i.e., processes and sub-processes) and resources involved.

In particular, the latter issue poses problems concerned with decentralized and autonomous modelling of multi-site software processes. Most works on distributed process management focus on developing paradigms and architectures for the enactment of these processes but they scarcely address decentralized modelling [16][14][13][9][4][10][7][8]. In most cases process modelling is a centralized activity and the enactment of portions of the process is distributed among different workflow engines. In some cases, the central process model is collaboratively edited with the contribution of people from different sites [10] and each site has visibility of the overall process model. A different approach is used in OzWeb [4] where process models are autonomously defined on the different sites and cooperate through specifically designed interfaces.

In this paper we present the GENESIS (Generalized ENvironment for procESs management in cooperatIve Software engineering) approach to distributed process modelling. The GENESIS platform is an outcome of a research project aimed at designing and developing a non-invasive and open source system to support software engineering processes in a highly distributed environment. In GENESIS, the global view of the project is modelled and enacted at the coordinator site (that is the technical leader of the distributed software project [14]), while sub-processes can be autonomously modelled and executed on different organizational sites. The global process model can be collaboratively edited by the project managers of the different sites.

The paper is organized as follows. Section 2 discusses related work. Section 3 presents an overview of GENESIS, with particular reference to the flexible approach adopted for distributed process modelling and enactment. Section 4 details an asynchronous communication protocol for distributed project definition and management, while Section 5 concludes.

2 Related Work

In global and virtual enterprises, software processes consist of multiple sub-processes that may span over organizational boundaries. The current commercial workflow technology does not provide the necessary functionality to model, enact, and manage distributed processes due to its mostly centralized server architecture. Numerous are coordination functionalities that cannot be fulfilled by traditional workflow systems [14][5], such as the support for distributed execution of a workflow; shared access to data and the use of groupware tools.

Modern workflow management systems exploit the web as a mean to enable distributed access to the facilities provided by the workflow engine [1][12][5][15]. However, most of these systems are still based on a client-server architecture and the problem of designing architectures for distributed process modelling and enactment of the process is still a research issue [16][14][13][9][4][10][7][8][17]. PROSYT is an artifact based PSEE [7]. Each artifact produced during the process is an instance of some artifact type, which describes its internal structure and behaviour. All the routing in this model is based on the artifact and the operations on them. PROSYT also allows for distributed enactment facilitated by an event-based middleware [8].

In [9] the authors propose an approach for the distributed execution that exploits an event notification service, named READY. Workflow participants, both workflow engines and agents, can subscribe to events that trigger the start of workflow activities and processes, and events that describe state changes in the workflow processes they are interested in. Therefore, the configuration of the participants in a workflow can be dynamically changed without requiring any modifications to the existing architecture.

The Endevors project [13] proposes an approach to provide a coordination mechanism for distributed process execution and tool integration by using the Hypertext Transfer Protocol (HTTP). The system uses a layered object model to provide for the object-oriented definition and specification of process artifacts, activities, and resources. The intent for distribution is to support a wide range of configurations with varying degrees and kinds of distribution: stand-alone with a base system configuration without distributed components, multi-user with a single remote data-store, multi-user with a single remote data-store are the configuration experimented for distribution.

Kötting and Maurer [14] propose an extension of MILOS [14] which focuses on the process support for virtual corporations. They propose three different approaches for distributed process enactment: replicated workflow engines; central coordinator site, and a peer-to-peer architecture for data exchange. The authors do not address the problem of decentralized process modelling.

Grundy et al. [10] focus on distribution problems in process modelling. The proposed system provides mechanisms for collaboratively editing process models both in a synchronous and an asynchronous way, together with version management support. The architecture is based on a central site maintaining the process model and distributed sites enacting portions of that model.

In the Ozweb environment the peer-to-peer paradigm for distribution is adopted [4]. Here a decentralized system consists of independent sub systems spread among multiple sites. In particular, the authors focus on the process autonomy of each sub system that should be self contained and operationally independent. To this aim they introduce the concept of ‘treats’ to guarantee compliance of the artifacts exchanged between sub-processes.

Our approach mixes both these features: we have the notion of a coordinator site where a global process can be defined in a collaborative way by the project managers of the different cooperating sites of a virtual organization. Sub-processes executed on different sites are autonomously defined and only have to respect the interfaces defined at global level. Moreover, the depth of the global process model is not limited to just two levels, as it is possible for a partner of the virtual corporation to have further sub-contractors.

3 Distributed Process Management in GENESIS

Traditional workflow management systems do not provide adequate support for the evolution of software organi-

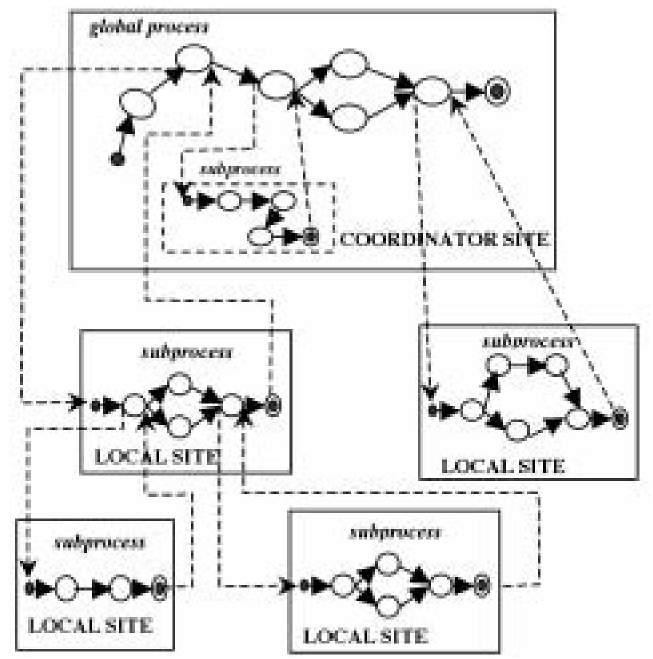


Figure 1: Hierarchical Process Decomposition.

zations towards distributed virtual models. The main open problem remains the systematic definition of distributed process models and their enactment across multiple sites using appropriate abstractions and mechanisms. The GENESIS environment has been developed with the aim to provide solutions to these problems; to this end the environment provides a special support for a distributed scenario, from the modelling of a distributed process to its enactment.

Distributed projects in GENESIS require a *project coordinator site*, managing the overall project and a number of local sites, managing specific project workpackages. The coordinator is in charge of modelling and executing the global process for the project, while the local sites are in charge of defining and executing sub-processes of that model, and concerning their own workpackages (see Figure 1).

The architecture of each GENESIS site includes different components as depicted in Figure 2:

- a workflow management system to model and enact software processes;
- an artifact management system to store and retrieve the artifacts produced within a process;
- a resource management system to allocate resources, in particular human resources, to a project;
- an event engine and communication system to collect and dispatch events raised during process management, such as the termination of an activity or the production of an artifact;
- a metric engine in charge of collecting metrics and presenting synthetic reports about the project status.

3.1 Process Modelling in GENESIS

In order for companies to use, to a certain extent, their existing practices and to ensure the quality of the overall process, support is given both for the top-down and bottom-up definition of processes. These may be achieved through a *multi-level process definition*, where an activity at one level (i.e. a super-activity) may correspond to a sub-process at lower level: the activity may be assigned to a different site, where it is independently modelled in a decentralized manner, and executed. In fact, the only requirement is that the sub-process interface (in terms of input / output artifacts) is conform to that of the super-activity (see Figure 1). This process componentization also enables integration of (sub)process models in a bottom-up fashion.

Besides super-activities, a process model can also contain global activities, i.e. activities that can be collaboratively performed by workers from different sites. The project manager of each site who participate in a global activity is in charge of providing the needed human resources for the activity.

In GENESIS the process modelling language is the same both at global level, to model the global software process with the coordination of the composing sub-processes, and at local level, to model the sub-processes at the single GENESIS sites. In order for the system to be the least invasive, a coarse grained definition of the process elements (activities and artifacts) has been decided. Activities are essentially described by the artifacts they will produce, and freedom is left to the worker(s) to decide how to actually perform them, in accordance with the organization standards.

GENESIS provides two process modelling stages. A process designer may create abstract process models, through the process definition tool, according to standards of the specific organization. Abstract process models include description of activities (including roles of people performing an activity and types of input and output artifacts) and enactment rules (or transitions) that basically describe control and data (artifact) flow between activities and are expressed through the Event-Condition-Action (ECA) paradigm. Abstract process models have to be customized for each project by specifying project data, such as the actual people performing activities and the actual artifacts. Project managers can use the abstract process definitions as templates to create concrete process models, through the project management tool as discussed in Section 4. These models can then be executed by the workflow enactment service, and different process instances may be enacted from the same concrete process model. Further details concerning the process modelling language are described in [3].

3.2 Distributed Process Enactment in GENESIS

The need to avoid a big bang approach to process modelling has been a guideline for the design of the GENESIS environment. As underlined by the industrial partners of the project, the usability of the environment could be significantly impacted by a rigid approach to process modelling and enactment. Indeed, an incremental approach at both activity and process definition level is required for the management of real projects. Often the start up of a project precedes a complete definition of the process models to be used for supporting and controlling it. In fact, for some scenarios, it may happen that the big picture of the process cannot be completed at design time because

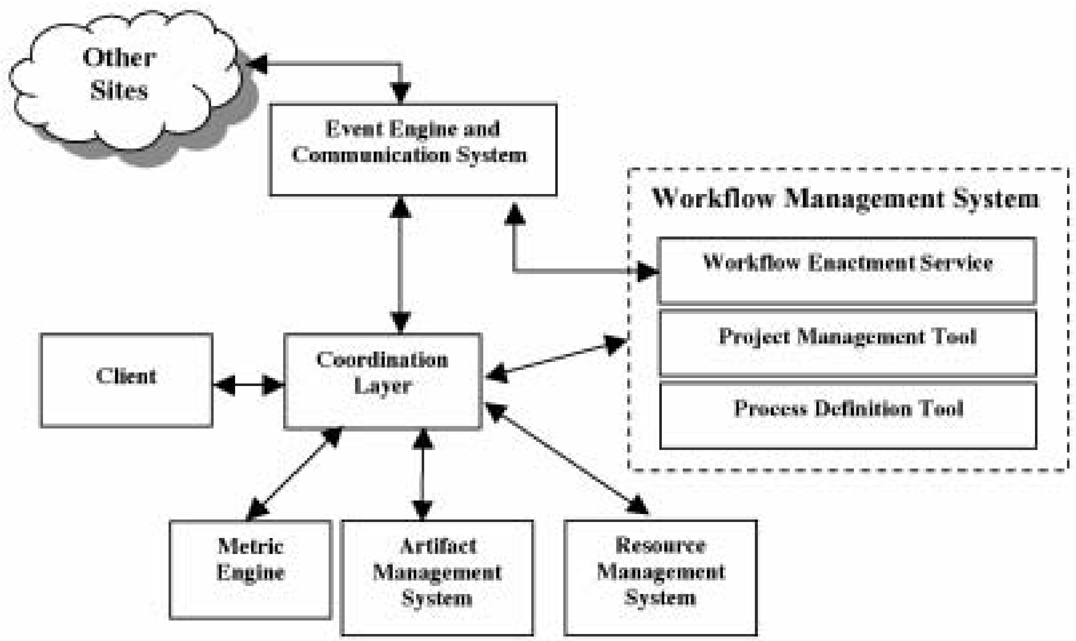

Other Sites
Event Engine and Communication System
Workflow Management System
Workflow Enactment Service
Project Management Tool
Process Definition Tool
Client
Coordination Layer
Metric Engine
Artifact Management System
Resource Management System

Figure 2: GENESIS Site Architecture.

there is not enough information for that. Thus, it is realistic to assume that the process models to be followed should be incrementally defined: starting from a simple model and 'complicating' it on the basis of the arising needs. In this respect, the GENESIS workflow engine is able to enact rough defined processes and facilities are offered to the project manager by the project management tool to refine the process definition at run-time.

Indeed, the model concretization process may be incremental as the process can start as soon as the needed resources have been assigned to the initial activity, and independently of the rest. In fact, checks are made during enactment to make sure that each activity has its resources allocated when it needs to be started. The project manager will be notified to staff the activity in order for the process to proceed. Most importantly, process may be dynamically modified. In fact, facilities are provided by the platform to both:

- change properties of the process instance, without changing the process model definition (i.e. the process map does not change). For example, a change of an activity assignment property;
- restructure the process map at run-time (e.g. adding/deleting activities), in order for the process instance to better match the real process.

These facilities are especially useful to handle unforeseen exceptions. In this respect, it should be noted that the distribution and decentralization of the process model through super-activities allows to restrict the scope of exceptions within the local sites, as long as they do not impact the border with the upper level process.

4 An Asynchronous Protocol for Distributed Project Definition

In GENESIS an asynchronous protocol has been defined for the communication between the global coordinator level and the local coordinated sites during the instantiation of a distributed software project. We distinguish three main phases: the creation of the project both at the coordinator site and the local sites, where the resource managers associate people to the project and select the project managers; the definition of the global process involving project managers of the different sites; and the definition of the local processes, independently defined by the different local project managers.

4.1 Project Creation

At the start of a new project the resource manager of the global site creates a project using the resource management tool of the platform. This means that s/he selects the human resources allocated on the global site and the local sites participating to the project. The allocated resources of the global site and in particular the global project manager are notified through the event engine and communication system.

A "Global Project Creation" event is also sent to the involved sites. Each resource management system of a local site automatically stores the received event and notifies the local resource manager. At this point, the local resource manager decides the allocations of the human resources and the local project manager who are notified about. The event "Local Project Creation" is sent to the resource management tool of the coordinator site, to store the needed information at global level and notify the global project managers.

4.2 Global Process Definition

Once the global project manager has received the notification concerning the "Global Project Creation", s/he can start defining the needed concrete process models for the project, starting from available abstract process models (if a suitable abstract process model is not available, it has to be created first). The global process model includes super-activities to be assigned to local sites and global activities, carried out by groups of people distributed among different sites. Local project managers can collaborate with the global project manager for the definition of the global process, as soon as they are selected by the local resource managers.

Each super-activity has to be assigned by the project manager to a site participating in the project. In this case a "Super Activity Creation" event is sent to the local site together with information concerning the super-activity (start and end date, artifact types, etc.). The project management tool of the local site automatically stores and associates this information to the corresponding project. The event is also notified to the project manager of the local site, as soon as s/he is appointed.

For each global activity, the global project manager selects the sites that have to provide human resources to collaboratively work on the activity (examples of such activities are project reviews to be conducted by the global and local project managers). A "Global Activity Creation" event is sent to each site involved in the global activity together with the role and number of required people.

It is worth noting that a concrete global process can start independently of the local process definition status (see next sub-section).

4.3 Local Process Definition

For each super-activity assigned to a local site, the project manager creates the corresponding concrete local process model (again, starting from an available abstract process model). As soon as the enactment of the concrete local process can start, a "Local Process Model Creation" event is sent to the project management tool of the coordinator site, which stores this information and associates it to the corresponding super-activity. The event is also notified to the global project manager.

The project manager of a local site involved in a global activity must select the human resources that will participate in that activity. When this is done, a "Global User Assigned" event is sent to the project management tool of the global site, to store this information at global level. The event is also notified to the global project manager.

5 Conclusion

In this paper we have presented the GENESIS approach to distributed project management. The definition of the GENESIS platform requirements for distribution, especially for the process modelling facilities, followed a strict interaction with the pilot users (the industrial partners) of the GENESIS project. The project started on September 2001 and ended on November 2003. The total effort was of 294 man-months and about 104,000 lines of code have been produced. Each GENESIS site is realized as a web application: the user interface and the coordination layer are realized using JSP (JavaServer Pages) and servlets (Tomcat being the web server), while the other components have been developed using the Java 2 Platform Standard Edition. The communication between the coordination layer and the different subsystems composing a GENESIS site is based on Java RMI, while the communication among the different sites is based on SOAP. The supporting database is based on MySQL Server.

The GENESIS platform has been evaluated by both the industrial and academic partners of the GENESIS consortium in different pilot projects. The code of the system has been delivered as open source software and can be downloaded from the SourceForge web site, <http://sourceforge.net>.

Acknowledgments

This work has been supported by the European Commission under Contract No. IST-2000-29380, Project GENESIS (Generalized ENvironment for procESs management in cooperatIve Software engineering, <http://www.ist-genesis.org>).

References

[1] C. K. Ames, S. C. Burleigh, and S. J. Mitchell. "WWWorkflow: World Wide Web based workflow", Proceedings of the 13th International Conference on System Sciences, pp. 397–404, 1997.

[2] V. Ambriola, R. Conradi, and A. Fuggetta. "Assessing Process-Centered Software Engineering Environments", ACM Transaction on Software Engineering and Methodology, vol. 6, no. 3, pp. 283–328, 1998.

[3] L. Aversano, A. Cimitile, A. De Lucia, S. Stefanucci, M. L. Villani. "GENESIS Workflow: Managing Distributed Software Processes", in Cimitile A., De Lucia A, and Gall H. (editors), Cooperative Methods and Tools for Distributed Software Processes, Franco Angeli, Italy, pp. 87–108, 2003.

[4] I. Z. Ben-Shaul and G. E. Kaiser. "Federating Process-Centered Environments: the Oz Experience", International Journal of Automated Software Engineering, 1997.

[5] Gregory Alan Bolcer and Richard N. Taylor. "Advanced workflow management technologies", Software Process Improvement and Practice, vol 4 n. 3, pp 125–171, 1998.

[6] D. Chan, and K.R.P.H. Leung. "A workflow Vista of the software Process", IEEE 8th International Workshop on Database and Expert Systems Applications (DEXA '97), 1997.

[7] G. Cugola. "Tolerating Deviations in Process Support Systems via Flexible Enactment of Process Models", IEEE Transactions on Software Engineering, vol. 24 no. 11, pp. 982–1001, 1998.

[8] G. Cugola, E. Di Nitto, and A. Fuggetta. "The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS", IEEE Transactions on Software Engineering, vol. 27, no. 9, pp. 827–850, 2001.

[9] J. Eder, E. Panagos. "Towards Distributed Workflow Process Management", AT&T Research Labs, 1999.

[10] J. C. Grundy, M. D. Apperley, J. G. Hosking and W. B. Mugridge. "A decentralized architecture for software process modelling and enactment", In IEEE Internet Computing, v 2 no 5, pp. 53–62, 1998.

[11] D. Georgakopoulos, H. Hornick, and A. Sheth. "An Overview of Workflow Management: from Process Modelling to Workflow Automation Infrastructure" Distributed and Parallel Databases, vol. 3, no. 2, pp.119–153, 1995.

[12] G. Q. Huang, J. Huang, and K. L. Mak. "Agent-based workflow management in collaborative product development on the internet", International Journal of Computer Aided Design, vol. 32, no. 2, pp. 133–114, 2000.

[13] A. S. Hitomi, P. J. Kammer, G. A. Bolcer, R. N. Taylor. "Distributed Workflow using HTTP: Example using Software Prerequirements", Proceedings of the International Conference on Software Engineering ICSE'98, 1998.

[14] B. Kötting, F. Maurer. "Approaching Software Support for Virtual Software Corporations", Proceedings of the International Conference on Software Engineering ICSE '99, Los Angeles, California, 1999.

[15] F. Maurer, B. Dellen, F. Bendeck, S. Goldmann, H. Holz, B. Kötting, and M. Schaaf. "Merging Project Planning and Web-Enabled Dynamic Workflow for Software Development", IEEE Internet Computing, vol. 4 no.3, pp. 65–74, 2000.

[16] F. Ranno and S. K. Shrivastava. "A Review of Distributed Workflow Management Systems", Proceedings of the international joint conference on Work activities coordination and collaboration, San Francisco, California, United States, 1999.

[17] Workflow Management Coalition. "Workflow Management Coalition Interface 1: Process Definition Interchange Process Model", Document no. WFMC-TC-1016-P, 1999. Available at <http://www.aiim.org/wfmc/standards/docs/if19910v11.pdf>.

Software Process Measurement

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

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

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

1 Introduction

The improvement of software processes has become one of the main aims of companies dedicated to the development and maintenance of computing systems. The need to improve processes stems from the fact that the quality of a process is closely related to the quality of the product, which means that to in order to get better products you need to have better processes. If software processes are to meet quality requirements, they need to deliver the expected results, be correctly defined, and be improved as required by business needs, which in competitive companies can be highly changeable. These are the objectives of "Software Process Management" (SPM) [5] which, in order to be applied effectively, should address four key responsibilities: to define, measure, control and improve the process. These responsibilities and the way they relate to one another are set out in Figure 1.

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

Francisco Ruiz-González has a PhD in Computer Science from the *Universidad de Castilla-La Mancha* (UCLM), Spain, and an MSc in Chemistry-Physics from the *Universidad Complutense de Madrid*, Spain. He is a full time Associate Professor of the Department of Computer Science at UCLM in Ciudad Real, Spain. He was the Dean of the Faculty of Computer Science between 1993 and 2000. Previously, he was the director of computer services at the aforementioned university (1985–1989) and he has also worked in private companies as an analyst-programmer and project manager. He is a member of the Alarcos Research Group, <http://alarcos.inf-cr.uclm.es/english/>. His current research interests include software process technology and modelling, software maintenance, and methodologies for software projects planning and management. He has also worked in the fields of GIS (Geographical Information Systems), educational software systems and deductive databases. He has written eight books and fourteen chapters on the abovementioned topics and he has published 90 papers in Spanish and international journals and conferences. He has sat on nine programme committees and seven organizing committees and he belongs to several scientific and professional associations: ACM, IEEE-CS, ATI, AEC, AENOR, ISO JTC1/SC7, EASST, AENUI and ACTA. <Francisco.RuizG@uclm.es>

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

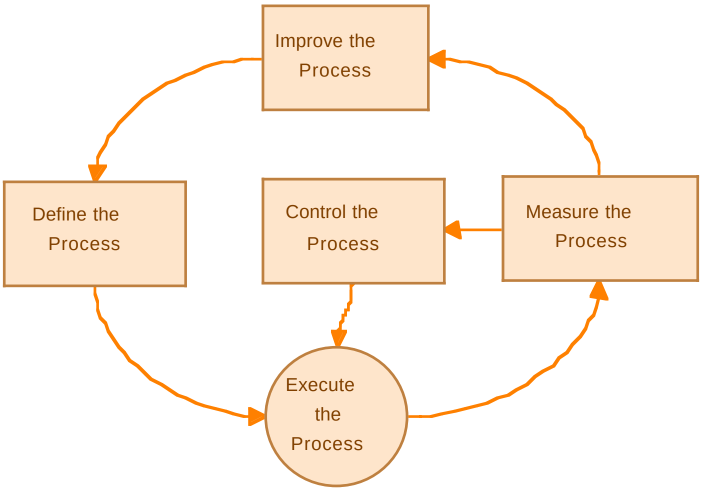


Figure 1: Key Responsibilities of Software Process Management.

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

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

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

2 Software Process Measurement Entities

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

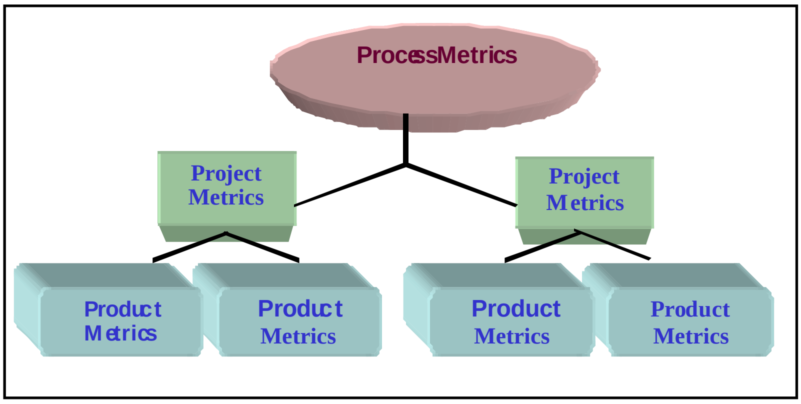


Figure 2: Process, Project and Product Metrics.

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

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

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

- **Process Measurement** is based on the study and control of the capacity of the processes and on the change management in these processes. Process metrics are extracted by measuring the characteristics of specific software engineering tasks in order to obtain metrics on faults detected before the delivery of the software, defects detected and reported by end users, human effort and time consumed, adjustments made to the planning, etc. When measuring the process, indicators referring to the process model itself should also be established, as the complexity and quality of the model may affect the execution, and therefore the quality, of the end product. However, literature referring to studies performed on process metrics have up to now concentrated on the measurement of projects and products. In an attempt to make up for this omission, the following section describes how to measure a software process at a conceptual level, while presenting a set of software process model metrics, and highlighting the aspects of the process that can be assessed using these metrics.
- **Measurement of the Project** is based on project management, which has been a mature research field for several years now.
- **Product Measurement.** The main reason for measuring software products is to evaluate the quality of deliverables. Much has been written regarding this subject, including studies, proposals and metrics, some of which are as well known and established, such as source code lines, function points, or McCabe's cyclomatic complexity.

3 Process Measurement at A Conceptual Level

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

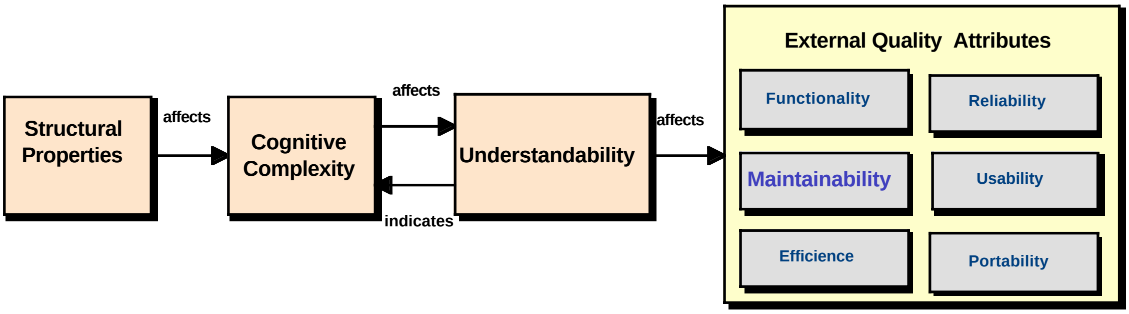


Figure 3: Structural Complexity Model [2].

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

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

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

3.1 Metrics for Software Process Models

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

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

Figure 4 shows an example of a simplified process model belonging to the Rational Unified Process (RUP). SPEM does not have its own graphic notation, but as it is defined as a UML profile, UML diagrams (class, package, activity, use case, and sequence) can be used to represent the different views of the process. The stereotypes used by SPEM (the icons in the figure) must be added to the UML diagrams.

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 dependencies of **Work Products** with the **Activities** in the process
NDWPOut(PM)	Number of output dependencies of **Work Products** with the **Activities** in the process
NDWP(PM)	Number of dependencies between **Work Products** and **Activities** $NDWP\ (PM) = NDWPIn\ (MP) + NDWPOut\ (MP)$
NDA(PM)	Number of precedence dependencies between **Activities**
NCA(PM)	Activity Coupling in the process model. $NCA(PM) = \frac{NA(PM)}{NDA(PM)}$
RDWPIn(PM)	Ratio between **input dependencies** of Work Products with Activities and **total number of dependencies** of Work Products with Activities $RDWPIn\ (PM) = \frac{NDWPIn\ (PM)}{NDWP(PM)}$
RDWPOut(PM)	Ratio between **output dependencies** of Work Products with Activities and **total number of dependencies** 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 between **Process Roles** and **Activities** $RRPA\ (PM) = \frac{NPR\ (PM)}{NA\ (PM)}$

Table 1: Software Process Model Metrics.

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

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

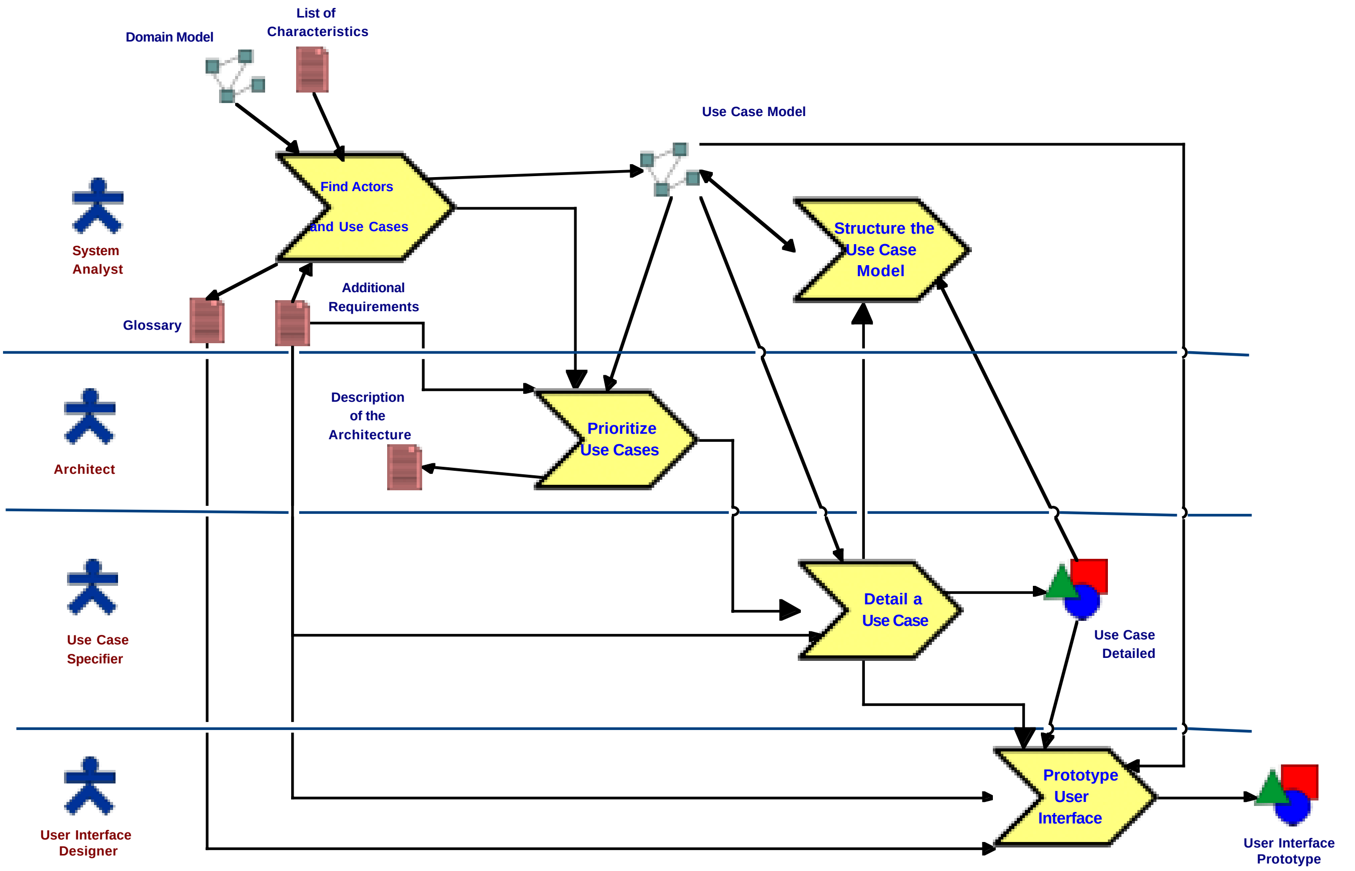


Figure 4: Example of A Software Process Model.

4 Conclusions and Challenges for The Future

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

Traditionally, software process measurement has focused on the measurement of projects and products, but as a result of the increasing interest shown by software companies in the institutionalizing, modelling and improvement of their processes, software process models have become an important entity to be taken into consideration. In this article we present a representative set of metrics for the evaluation of the maintainability of software process models. These metrics are useful for predicting the maintainability of process models and provide useful indicators for companies implementing process improvement programmes. An important future line of research in this area is the development of empirical studies to establish relationships between the maintainability of process models and the results obtained from their enactment in the form of software projects. Finally, by means of the integrated measurement of software process related entities, we can obtain the required quantitative basis from which the correlation existing between process and product can be objectively evaluated.

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

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

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

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

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

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

References

[1] V. Basili and D. Weiss. A Methodology for Collecting Valid Software Engineering Data. IEEE Transactions on Software Engineering, 10, pp. 728–738, 1984.

[2] L. Briand, S. Arisholm, F. Counsell, F. Houdek, and P. Thévenod-Fosse. Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions. Empirical Software Engineering, 4(4), pp. 387–404, 1999.

[3] J.C. Derniame, B.A. Kaba, and D. Wastell. Software Process: Principles, Methodology and Technology (Vol. LNCS 1500). Springer-Verlag, 1999.

[4] N. Fenton. Metrics for Software Process Improvement. In M. Haug, E. W. Olsen & L. Bergman (Eds.), Software Process Improvement: Metrics, Measurement and Process Modelling (pp. 34–55). Springer, 2001.

[5] W. A. Florac and A. D. Carleton. Measuring the Software Process. Statistical Process Control for Software Process Improvement. Addison Wesley, 1999.

[6] F. García, F. Ruiz, J. A. Cruz, and M. Piattini. Integrated Measurement for the Evaluation and Improvement of Software Processes. 9th European Workshop on Software Process Technology (EWSPT'9), Helsinki (Finland), pp. 94–111, 2003.

[7] F. García, F. Ruiz, and M. Piattini. Definition and Empirical Validation of Metrics for Software Process Models. 5th International Conference Product Focused Software Process Improvement (PROFES'2004), Kansai Science City (Japan), pp. 146–158, 2004.

[8] F. García, F. Ruiz, M. F. Bertoa, C. Calero, M. Genero, L.A. Olsina, M. A. Martín, C. Quer, N. Tondori, S. Abrahao, A. Vallecillo, and M. Piattini. Una Ontología de la Medición del Software. Technical Report, Depto. de Informática, Universidad de Castilla-La Mancha. Available, in Spanish, at <http://www.info-ab.uclm.es/trep.php?&codtrep=DIAB-04-02-2>, 2004.

[9] IEEE. IEEE Std 1061-1992, "IEEE Standard for a Software Quality Metrics Methodology", 1992.

[10] ISO/IEC. ISO IEC 15504 TR2:1998, Software Process Assessment – Part 4: Guide to conducting assessment. International Organization for Standardization, 1998.

[11] ISO/IEC. ISO 15939: Software Engineering – Software Measurement Process, 2002.

[12] J. McGarry, D. Card, C. Jones, B. Layman, E. Clark, J. Dean, and F. Hall. Practical Software Measurement. Objective Information for Decision Makers. Addison-Wesley, 2002.

[13] C. Ocampo, and P. Botella. Some Reflections on Applying Workflow Technology to Software Processes. Universitat Politécnica de Catalunya, Dep. de Lenguajes y Sistemas Informáticos, technical report TR-LSI-98-5-R, Barcelona, 1998.

[14] OMG. Software Process Engineering Metamodel Specification; adopted specification. Object Management Group, 2002.

[15] H.D. Rombach. Design measurement: some lessons learned. IEEE Software, 7(3), pp. 17–25, 1990.

[16] SEI. Capability Madurity Model Integration (CMMISM), version 1.1. Software Engineering Institute, 2002.

[17] WfMC. TC00-1003 1.1: The Workflow Reference Model. Workflow Management Coalition, January-1995.

[18] WfMC. TC-1025 final draft 1.0: Workflow Process Definition Interface – XML Process Definition Language (XPDL). Workflow Management Coalition, October-2002.

Process Diversity and how Practitioners Can Manage It

Danilo Caivano and Corrado Aaron Visaggio

Since IT projects are unique regarding their combination of specific goals, technologies in use, and characteristics, providing 'general' processes it is not an effective solution. Instead effective and efficient processes custom tailored to a project and based on experience collected during past projects execution are required. This is in contrast with the industry practices where reuse-oriented process descriptions and goal-oriented planning are often missing. Usually a process can undergo a certain numbers of modifications, due to the different operative contexts in which it is executed. The modifications generate many different versions of the process, named specialized processes. Each one of these must be managed properly in order to govern a just evolution consistently with all the others. Considered the dimension of the actual scenarios, maintaining all the processes and their specialized versions is not a trivial task. We have defined a process pattern based framework to accomplish this purpose. In this paper we present the framework, that we are realizing with an Italian enterprise, and an explanatory case study we are developing within the Research Centre on Software Technology in Bari, Italy.

Keywords: Empirical Software Engineering, Software Engineering, Software Process Management, Statistical Process Control.

1 Introduction

The literature proposes many kinds of processes ranging from business processes to software development and maintenance process.

These definitions describe processes at a coarse grain: they formalize a paradigm, as a general solution to a problem. These directives must be properly developed by designing processes aiming at fulfilling specific constraints and business goals. This operation is usually named *specialization or customization of Software Process*. When dealing with process specialization, three kinds of concerns should be addressed.

The first one regards the *culture of the organization*. With culture we mean the way activities are usually performed in the Organizations, due to: Quality System, methodology adopted, technologies used, coding standards, kinds of documentation produced.

These factors can affect the design of one process much in depth, up to significantly change definition and sequence of activities also if well and precisely codified. E.g., the unit test can be performed in many different ways if considering different testing tools, different artifacts for reporting test beds and results, procedures to calculate the paths to be tested and guidelines in use. Consequently, the same task (e.g., unit test) can be reached with different sub-processes.

Another common concern should be taken into account: the *re-use of sub-processes*. Large enterprises usually exploit hundreds of processes. A typical situation is that many departments execute the same sub process without sharing any information about it, such as: the process model, results of monitoring, improvement initiatives. As an effect, the evolution of the same sub process follows different directions in the same organization, with a clear disadvantage. The solution for this problem is a formal sharing of the process knowledge, so that each department can benefit from the experiences realized in the other departments.

Danilo Caivano received his BSc in Computer Science and his PhD degree in Software Engineering from the *Università degli Studi di Bari*, Italy. He is affiliated to the Software Engineering Research Laboratory, <http://serlab.di.uniba.i>, and Research Centre On Software Technology – BARI. He is Assistant Professor at the Department of Informatics, University of Bari. He has also been consultant for many small to medium companies, where he has carried out both empirical investigations for validating research results in industrial contexts, and technological transfer through pilot projects. His research interests fall in the area of software process management and improvement within co-located and distributed contexts, software estimation, and empirical software engineering. At the moment he is involved in a research project for evaluating the efficacy of Statistical Process Control as a means for monitoring geographically distributed software processes. <caivano@di.uniba.it>

Corrado Aaron Visaggio obtained his BSc in Electronic Engineering at the *Politecnico* of Bari, Italy, in 2001. He developed his Master Thesis at the *Fraunhofer IESE*, Germany. In 2002 he started attending PhD courses in Software Engineering at *Università degli Studi del Sannio*, in Benevento, Italy. Currently he is working as researcher at the Research Centre of Software Technology (RCOST), in Benevento. His main topics of research are: software process modelling and management, agile methodologies for developing software, and knowledge management in Software Engineering. <visaggio@unisannio.it>

A third concern regards the *technology used for managing the process*. Process management is supported by different kinds of technologies targeted to validate, simulate, and activate the related workflow. In the same organization on the same process different technologies can be used for the same functionality. E.g., different workflow engines, different modelling tools, different simulators.

The three concerns are related to *Process Diversity* [1][17]. As pointed out in [2][18], many factors, from here on called *"diversity factors"*, are essential components in forming process diversity, and affect process management: business environment, technology, industrial standard, quality program, vision, budget, size, structure and culture of organization. By coupling each diversity factor with the appropriate values, the actual context can be rigorously defined. This can be referred as the "profile of an operative context" (from here on referred to as context profile for conciseness).

Theoretically, each context profile requires a custom tailored process, but this implies an understanding of process variations and knowledge about when to use which process variants. In fact, according to [18]:

- each process alternative needs to be elicited and explicitly described;
- process alternatives need to be characterized and constrains/rules on their use need to be formulated;
- a characterization of the context is needed in order to able to select a process variant.

A framework for managing process models in highly variable context profiles and for accomplishing reuse of experience acquired in previous process modelling cases is needed as a means to make changes in a safe and economic way.

In order to address these needs we have developed ProMisE. The key ideas are:

1. to use a framework based on process patterns (a tuple problem-solutions) to manage process diversity and thus to face process customization in an effective way. Given a problem, that is a product or a service to be delivered, the pattern allows process engineer to associate a family of processes, each of them corresponding to a process variant of the root-process, as its solution. The root-process describes the more generic solution. On the other hand, each variant represents a specialization of the root-process with respect to a specific context profile; the overall set of variants encloses those ones that have been experimented till the current moment.
2. to define a package of functions for managing the repository of the defined patterns. The package contains the functions mainly corresponding to the phases of the process pattern management process and spans its entire lifecycle.

The framework was presented in [3] and now the authors are driving the realization of its technological support within a company operating in the field of Enterprise Resource Planning. In this paper we will present the framework together with *an explanatory case study we are developing within Research Centre on Software Technology*, Bari, Italy. We will also discuss the problems we are coping with during its production. The paper proceeds as follows: Section 2 presents some related work, Section 3 introduces the framework, Section 4 discusses an exemplar application of the framework, and, finally, Section 5 draws conclusions.

2 Related Work

Some researchers have been focusing their attention on the "pattern concept", trying to define a complete description [4][5]. In [6] the authors give a definition for a software pattern: it *"identifies a recurring problem and a solution, describing them in a particular context to help developers understand how to create an appropriate solution. Patterns aim to capture and explicitly state abstract problem-solving knowledge that is usually implicit and gained only through experience.*"

In [7] the authors propose an approach to structure and store experience in order to enable effective reuse: *"The main idea of this approach is a rearrangement and reprocessing of captured experience into quality patterns which are based on a problem-solution strategy.*"

As shown in [8][9][10], mostly patterns are applied for supporting software development lifecycle.

This paper refers to the same well known concept of pattern and applies it to processes instead of products.

As shown in the following, other authors have studied how diversity factors affect process designing and management, highlighting the need for properly customizing the software models rather than designing new ones as context profile changes.

In [11] Sutton states *"The ability of repeating a process can critically affect a start-up's success"*. The author refers to a start-up company producing a family of products that the company treats more or less similarly. Repeatable processes span over the life cycle, including development, quality assurance, documentation, and training. *"In such case specializing readily processes and test plans is more useful than customizing processes and plans for each product-family member. You can apply some technologies in various ways in many contexts and reuse them as the organization and its processes evolve.*" Sutton doesn't recommend adopting procedures, technologies, and protocols for one product or life cycle phase unless special needs exist or they can be reused for other products or phases.

In [12] how process diversity affects the field of software reuse is explained. The authors have focused on the reuse processes applied to four companies; during this work they have identified the reuse process characteristics for each company: reuse approach, reuse technology, reuse processes and roles, which develop assets, when assets are developed, reuse processes added, non-reuse processes modified. All these parameters can be used to tailor a reuse process to the particular organization according to its specific characteristics.

In [17] several articles are presented that show how process diversity affects software maintenance and the need for customizing maintenance process to context characteristics.

Finally in [18], the author presents a tool-based technique for customize a process model to project constrains.

All these works motivate researchers and practitioners to further investigate this area of interest.

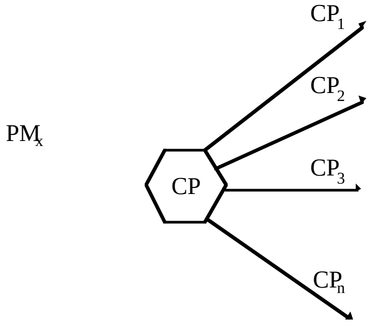


Figure 1: Specializing the Process Model.

3 ProMisE Framework

The PromIsE framework can be summarized as follows: given a process PM_x, from here on called *root-process*, a specialized process SP_{xj} is defined as $SP_{xj} = f(PM_x,CP_j)$; where f() is the mechanism, CP_j is the context profile dictating specialization needs and the consequent specialization actions. Each specialized process is a variant of the correspondent process model. This was necessary in order to adapt the latter to a specific context profile. A context profile is a set of instantiated diversity factors DF_i. The generic DF_i where i=1…n is a diversity factor having a definition domain $[DF_i]=\{FI_{i1}, FI_{i2}, FI_{i3},\ldots FI_{ik}\}$. FI_{ih} where h=1…k is a factor instance of $[DF_i]$. Let's define a generic characterization of CP, indicating $CP_j=\{[DF_1]_j, [DF_2]_j, [DF_3]_j,\ldots [DF_n]_j\}$ where $[DF_i]_j$ is a factor instance $FI_{ih} \in [DF_i]$ with n= number of diversity factors included in CP_j. After that, the concept of a *context profile* can be generalized as $CP=\{CP_1,CP_2,CP_3,\ldots,CP_N\}$ where $\forall i,j\in[1,N] : CP_i\neq CP_j$.

This characterization allows to explicit and formalize the diversity among the processes that usually result as implicit information within the same processes.

3.1 The Framework

The framework consists mainly of a collection of process patterns (association between a problem and a family of solutions) and a package of functions to manipulate the process patterns.

It provides the process engineer with a mechanism that, given a starting problem, allows to:

1. choose the candidate root-process model solving the problem when present in the dedicate database;
2. identify all the process model's variables allowing the process model be specialized, if necessary, to diverse context profiles;
3. guide the process engineer in choosing the suitable variant (of the root-process) for the context profile mirroring the real world environment, by using a decision model that points out the changes carried out on the beginning process.

A process model describes a set of methods, practices, skills, tools and the relationships among them to define a product or a service with a certain degree of quality.

The activity of tailoring a process model consists in specializing a process model to the context profile of interest. Formally speaking, given a process model PM_x, it is necessary to define the context profile in which the process PM_x can be executed. By establishing the proper values FI_{ih} of each diversity factor $[DF_i]_j$, the correspondent context profiles may be obtained: CP_1, CP_2, CP_3, …, CP_n. As a consequence the PM_x is modified in order to reflect the differences among the diverse context profiles. For this reason, a reviewed version of the original process model can be associated to each context profile, determining the specialized processes SP_{x1}, SP_{x2}, SP_{x3}, …, SP_{xn}. In Figure 1 this concept has been depicted.

3.2 Process Patterns

Starting from the problem to solve, a process pattern should show an initial solution, represented by PM_x and the path of actions (mainly process fragments) to be applied in order to obtain the correct SP_{xj} according to a specific CP_j. In this way, the ProMisE Process Pattern expresses the pair problem-solutions for defining a family of solutions (the variants SP_{xj} of the PM_x) related to a problem. It should present the following elements:

Name: identifier of the pattern.
Problem: description of the problem supported by the pattern.
Process Model (PM_x): the root-process model for which the specialized ones are created.
Decision Model: the decision model defines the path of actions for specializing the PM_x according to a predefined CP_j.
Solutions: the specialized processes SP_{xj}.
Relationships: they consist of other process patterns (PP_i) that specialize the current one. These patterns refer to sub processes that detail an SP_{xj} phase.
Experiences: experiences reported in using the pattern.

Each specialized process (SP_{xj}) contained in a pattern can also have relationships with many other patterns. This can occur when a SP_{xj}'s phase is detailed by a sub-process associated to a pattern.

Every time a pattern refers to other patterns, the last one details the first one. If a given pattern doesn't refer to other ones, then it is either at its highest level of detail or it lacks a detail pattern.

The Decision Model can be represented by using a variant of a decision table that emphasizes which actions must be accomplished on the process model for specializing it to a context profile:

1. the conditions represent the diversity factors (DF), i.e. the features of the context profile in which the process will be carried out;
2. the actions represent the "specialization action" to be accomplished in order to obtain a specialized process SP_{xj} of the PM_x;
3. the rules (represented by the columns) merge a combination of factor instances FI_{ih} (expressing the context profile CP_j) with the specialization action to perform, in order to obtain the appropriate SP_{xj} for the context profile CP_j;

Continuous reuse of a pattern will most likely provide the organization with three fundamental advantages.

First of all, like all other patterns, it is easily traceable within the repository because it is characterized according to the problem it refers to, and at the same time it probably corresponds to the problem the process engineer is trying to solve.

Second, pattern reuse may highlight variants that haven't been forecasted in the context profile that the decision model is based on; in this case the model itself is extended with the unexpected context model. Therefore, the pattern with reuse extends the number of variants it refers to and consequently becomes more complete.

Third, it may point out that a variant can be formalized in a more appropriate way within the context it refers to. This increases the pattern's efficacy and extend the process knowledge.

According to what has been previously mentioned, a pattern is a collector of knowledge generated by various sources and transferable independently from who generated it in first person. In other words it is an experience package. For this reason, ProMisE framework can be considered itself as an experience base [13].

3.3 Functions

We have defined the main functional requirements that the technological support of the framework should own. In this section an overview is provided. The functions related to the management and use of the pattern in the experience base are listed below.

Pattern Creation. This function must be used when a new problem arises. All the components of the pattern must be defined. Particular attention must be paid in defining the relationships that may exist between the pattern just created and the other ones already stored inside the experience base. When necessary, the process engineer must update or create the decision model from scratch and describe how the new pattern is linked to the context profiles involved.

Update Pattern. This function modifies a pattern already present in the repository. A modification can consist in adding a context profile; adding or deleting a factor instance in a diversity factor included in the pattern; modifying the specialization actions or the rules of the decision model; adding new relationships etc.

Select Pattern. It selects the pattern corresponding to the problem that the user needs to solve.

Generate Process Model. After having selected a pattern we have identified a process model (PM) and its variants; from here on referred to as root pattern. By assigning values to the diversity factors, the changes that must be carried out on PM are identified. This allows to produce a more specific process model (SP_{xj}) for the operative context CP_j. Also, with refer to the combination of values of the diversity factors, the decision model may identify one or more patterns (PP_i) the process is related to. Each of these patterns PP_i corresponds to a family of variants of a process detailing an SP_{xj} component (for example, an SP_{xj} phase). Thus throughout the relationships, the root pattern is specialized in further levels of detail leading to the specification of another process SP_{xj1} detailing SP_{xj}. This mechanism goes on throughout the relationships chain, pattern after pattern, until having explored all of them. The result is a specialized process within the operative context having the highest possible level of detail according to the knowledge stored in the experience base.

4 Case Study

The framework presented in the previous sections has been applied, through a case study, to formalize the experience matured by industrial projects with the aim of clarifying how the framework can be used in practice. In particular, acquired knowledge in previous years within the Software Engineering Research Laboratory (SERLAB) projects is related to extraordinary maintenance of software systems. For further details see [14][15][16].

Legacy system rejuvenation is straightforward according to various factors: goals, budget, resources, economical value and quality of the legacy, and so on. When rejuvenating an aging system, one or more types of renewal processes are used. For instance: *reverse engineering* analyses a subject system to identify its components with their inter-relationships, and to create a representation of the system in another form or at a higher level of abstraction; *restructuring* improves the structure of a program automatically, without taking the program purpose into account; *restoration* improves the structure of programs, and of data according to their meaning in the programs; *reengineering* examines and alters a subject system to reconstitute it in a new form with improved quality. This may include modifications with respect to new requirements not met by the original system; *rehosting* refers to migration of the system to a different architecture; *migration* involves changing the software environment the legacy system runs on Chifosky (1990).

Before deciding the most appropriate combination of renewal processes to use, some preliminary activities must be carried out:

- **Portfolio analysis:** consists of the analysis of system's capabilities to be taken into account for the maintenance process.
- **Quality Assessment:** this activity identifies the quality level of the existing system and that of the modified one.
- **Economy Evaluation:** this assessment helps understand the cost and return of modifications.
- **Risks Assessment:** this assessment is headed to identify and mitigate risks.

These activities are necessary for deciding the renewal process to adopt according to the improvement goals of the process and of the constraints of the project.

The project goals considered within the case study are described as follows:

Diminishing the cost of application administration. There is a vast variety of cost sources in maintenance process; it is not possible to consider all of them, because some are specific to the Organization; others are hidden or difficult to identify, such as: transfer of knowledge, distance between the supplier of management services of the software system and the organization using the system.

Part of the cost taken into account are: Cost price of the software System; Cost of maintenance and operation of the system:

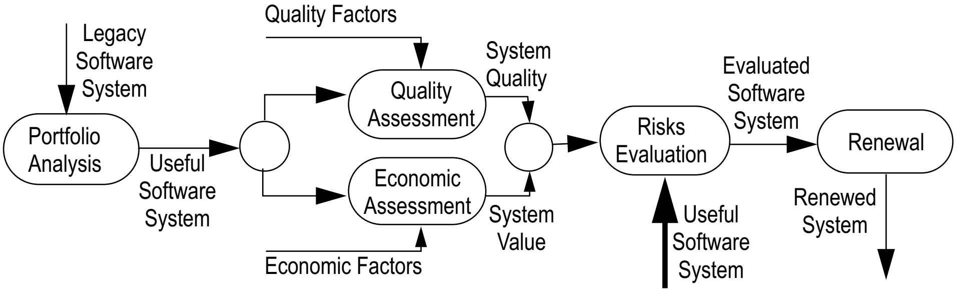


Figure 2: Root Process.

this includes also the costs for human resources and the relative logistics involved in the activity; Cost of assistance, basically refers to the cost of the experts; Cost of diffusion of knowledge of the package to maintainers, managers, assistants and users; Cost of the hardware and software platforms necessary for using, managing and maintaining the software system; Cost of coping with bad or incomplete functioning of the software system.

Relying on experiments developed in industrial settings it results that each cost of those listed before can be reduced by activating specific maintenance activities (re-engineering, rehosting, reverse engineering, restructuring).

Improve engineering features of application. When analysing a system it is possible to consider different detail levels. The two main ones are the code's structure and the engineering features. The latter refers to the general organization of the system, including the division in modules, the relationships between them, the data access and updating. The former regards more directly the intrinsic quality's aspects of the code. This goal often requires the legacy to be reengineered. Note that reengineering may also include some types of reverse engineering followed by restoration or restructuring in order to make the system more evolvable in future.

Change the application deployment. A maintenance process can require new capabilities of the development environment, in order to be executed properly and successfully. Such improvements can refer to, e.g., configuration management, development or maintenance environment, hardware settings, print management, and others. This often requires rehosting.

Change the middleware in use. Sometimes, in order to rejuvenate the legacy system (for example to be able to adapt to new technologies, peripherals, services, …), the modification of the middleware of software system is required and thus migration is needed.

Improve code's structure. This activity aims at improving the quality factors of the code itself, such as coupling, cohesion, pollution, lexicon, and modularization. This goal is often reached by using restructuring.

Improve readability and understandability. In order to achieve a high quality maintenance task, programmers must be able to understand the code directly from the listing. Consider that the programmer works basically on it, also if he could use further project documentation. In order to reach this objective a restoration process is used.

Update system's documentation. When modifications are brought on the code, the system documentation must be properly modified, in order to keep it aligned with the code. We mean all the documentation: requirements specification, analysis and design documentation, user manuals, test beds. In this case reverse engineering is needed.

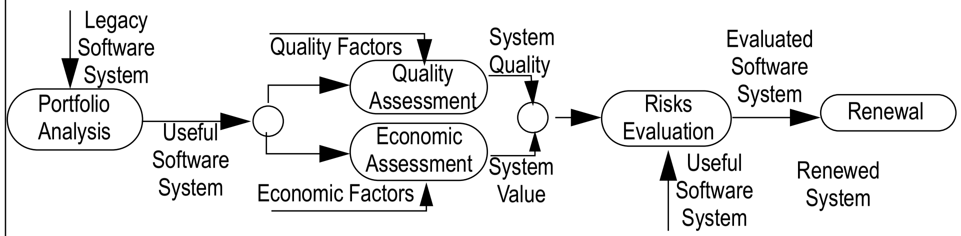
Finally, when the legacy has a poor documentation, a low quality level, a low economic value and the previous improvements are all required, it is probably more convenient to *Write the system from scratch*. When Rehosting, Migration, Restructuring, Restore, Reengineering or Write from Scratch are executed, the Equivalence test is required in order to assure the equivalence between the legacy system capabilities and renewed system capabilities.

In Figure 4 the process pattern of extraordinary maintenance, is illustrated. In the following a brief description is given:

- **Name:** Renewal of legacy system
- **Process Model (PM_x):** it describes the core activities of a renewal process and the artifacts exchanged between process activities. There is also an activity named "renewal" that is further refined by using decision model. The root process is depicted in Figure 2.
- **Problem:** evaluate degradation and rejuvenate an aging system.
- **Decision Model:** the decision model defines the path of actions for specializing the PM_x according to a predefined CP_j. The decision model is structured as follows: in the gray part there are 7 diversity factors DF_i that represent the possible goals of the Extraordinary Maintenance (*Diminishing the cost of application administration; Improve engineering features of application; Change the application deployment; Change the middleware in use; Improve code's structure; Improve readability and understandability; Update system's documentation*). The factor instances FI_{ik} correspond to the possible values that a DF_i can assume. In this case they consist in "*Y*" or "*N*" that indicate the need to reach the correspondent maintenance goal. In the white part there are the 12 sub-processes that can be selected for properly specializing the "renewal" activity of the root process. The symbol "x" indicates that the extraordinary maintenance process must include the correspondent sub process

Process Pattern
Name: Renewal of legacy system

Process Model: *Extraordinary Maintenance*



Problem: to Evaluate degradation and rejuvenate an aging system

DECISION MODEL

Diversity Factors	Factor Instance																															
1. Dimishing the cost of application administration	Y																								...	...	N					
2. Change the application deployment	Y																								...	...	N					
3. Change middleware in use	Y																N								...	...	N					
4. Improve code's structure	Y								N								Y								...	...	N					
5. Improve readability and understandability	Y				N				Y				N				Y				N				...	...						
6. Update system's documentation	Y		N		Y		N		Y		N		Y		N		Y		N		Y		N		...	...	N		Y		N	
7. Improve engineering features of application	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	...	...	Y	N	Y	N	Y	N
1. Portfolio Analysis	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	...	...	x	x	x	x	x	x
2. Quality Assessment	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	...	...	x	x	x	x	x	x
3. Economic Assessment	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	...	...	x	x	x	x	x	x
4. Rehosting	.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	.	x	x	x	x	x	x	x	...	...	.	.	.	.	.	.
5. Migration	.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.	...	...	.	.	.	.	.	.
6. Restructuring	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.	.	x	x	x	x	x	x	x	...	...	.	.	.	.	.	.
7. Restore	.	x	x	x	.	.	.	.	x	x	x	x	.	.	.	.	.	x	x	x	.	.	.	.	...	...	x	x	.	.	.	.
8. Reverse Engineering	.	x	.	.	x	x	.	.	x	x	.	.	x	x	.	.	.	x	.	.	x	x	.	.	...	...	.	.	x	x	.	.
9. Reenigneering	–	–	x	–	x	–	x	–	x	–	x	–	x	–	x	–	–	–	x	–	x	–	x	–	...	...	x	.	x	.	x	.
10. Equivalence Test	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	...	...	x	x	x	.	x	.
11. Risk Evaluation	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	...	...	x	x	x	x	x	x
12. Write from scratch	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	...	...	.	.	.	.	.	.
Solutions	SP_{1}	SP_{2}	SP_{3}	SP_{4}	SP_{5}	SP_{6}	SP_{7}	SP_{8}	SP_{9}	SP_{10}	SP_{11}	SP_{12}	SP_{13}	SP_{14}	SP_{15}	SP_{16}	SP_{17}	SP_{18}	SP_{19}	SP_{20}	SP_{21}	SP_{22}	SP_{23}	SP_{24}	...	...	SP_{123}	SP_{124}	SP_{125}	SP_{126}	SP_{127}	SP_{128}
RELATIONSHIPS																																
Quality Assessment	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	X	x	x	x	x	x	x			x	x	x	x	x	x
Economic Evaluation	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	X	x	x	x	x	x	x			x	x	x	x	x	x
Restructuring	.	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.	.	X	x	x	x	x	x	x			.	.	.	.	.	.
Restore	.	x	x	x	.	.	.	.	x	x	x	x	.	.	.	.	.	X	x	x	.	.	.	.			x	x	.	.	.	.
Reverse Engineering	.	x	.	.	x	x	.	.	x	x	.	.	x	x	.	.	.	X	.	.	x	x	.	.			.	.	x	x	.	.
Reenigneering	–	–	x	–	x	–	x	–	x	–	x	–	x	–	x	–	–	–	x	–	x	–	x	–			x	–	x	–	x	–
Risk Evaluation	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	X	x	x	x	x	x	x			x	x	x	x	x	x

EXPERIENCES: This pattern is the result of the experience collected during several renewal projects. More details are presented in [14][15][16]

Figure 4: Extraordinary Maintenance Process Pattern. (b)

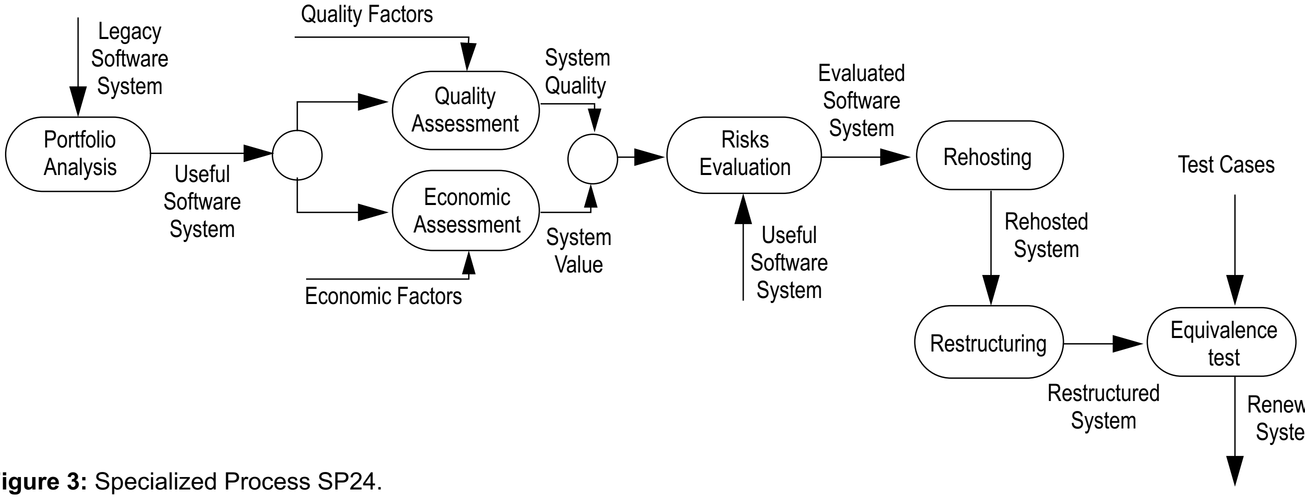


Figure 3: Specialized Process SP24.

reported on the same row. Thus, each column contains the specialization rules to obtain a specialized version SP_{xj} of the PM_x on the CP_j. In Figure 3, only a part of the decision model is presented. It includes a total of 2^7=128 rows that correspond to the different CP_s considered.

- Solutions: The decision model individuates 128 different SPxj of the root process PM_x. As an example, let's consider the case of the column 24: the context profile is CP_{24}={*Y,Y,N,Y,N,N,N*}, i.e. the goals of the correspondent specialized process are:
  - *Diminishing the cost of application administration,*
  - *change the application deployment,*
  - *Improve code's structure*

  The rule on the column 24 dictates that the SP_{24} must include the following activities:
  - *Portfolio Analysis,*
  - *Quality Assessment,*
  - *Economic Evaluation,*
  - *Rehosting,*
  - *Restructuring,*
  - *Equivalence Test,*
  - *Risk Evaluation.*

  The resulting specialized process SP_{24} is showed in Figure 3. The rational for specialization was elicited from [14][15][16].
- **Relationships:** this section specifies other process patterns (PP_i) that specialize the current one. These patterns refer to sub processes that detail an SP_{xj} phase In the case of the SP_{24} of the pattern in Figure 4, in order to obtain a fully specialized process the following PP need to be further considered:
  - *Quality Assessment,*
  - *Economic Evaluation,*
  - *Restructuring,*
  - *Risk Evaluation.*
- **Experiences:** indicates the source of experiences used to define the pattern or projects executed by using the pattern considered. For what concerns pattern in Figure 4, the source of experience is represented by the references [14][15][16] included in the bibliography.

5 Discussion and Conclusion

The innovative part of ProMisE consists in keeping an experience base updated by recording the continuous changes in the real world. The major advantages in using ProMisE can be synthesized in the next two points:

1. *To improve the comprehension of the characterization.* Over the time, the Organization implementing the process can better detail the set of operative contexts, by adding (or deleting) some DF or by adding or deleting some FI. It is easy to imagine which sort of chain effects this has on the experience base: all the patterns and the decision models involved must be appropriately changed.
2. *To improve the comprehension of the relationship between the characterization and the specialization.* Some evolutions in the actual operative context can lead to choosing another specialized process segment for the context profile CP rather than the current one recorded in the experience base.

Processes are adopted in frequently changing environments; this condition leads to expensive and complicated process tailoring, deploying money and consuming time to ensure stability and avoiding lacks of capability.

Given a problem and a first solution for it, thanks to the concept of patterns, ProMisE allows to specialize the initial solution according to the context in which it will be used. Furthermore, it points out all the changes made throughout the specialization process. In this way it addresses for reusing experience.

ProMisE aims to extend the well known concept of pattern and more precisely it wishes:

1. To create a family of solutions for a family of problems just starting from specific pairs problem-solution. This allows the process engineer to apply the work and the knowledge developed in specific experiences to many other different ones, but placed in the same family of process-solution pair.

2. To continuously enrich the experience base by properly modifying the decision models and the process patterns as real world evolutions occur and consequently to make the organization more capable to face process diversity problems.

We are realizing the system in collaboration with a company operating in the field of Enterprise Resource management.

Some problems have been highlighted and are discussed in the following.

First of all the management of experience is a central concern. The knowledge base of the system grows as the captured experience increases. The experience needs to be elicited in the appropriate way by the field. This is not a trivial issue, if considered scalability and competence of the process engineer. The competence of the process engineer regards basically the experience to be stored (too much can create pollution in the base, and if it is not enough it can make usefulness the effort of maintaining the system).

Scalability is another problem to be faced; as matter of fact, decision models can grow very fast. An excessive growth can affect usability of the system (navigation between the decision tables, comprehension of the content of decision table) and the maintainability of the system (validate consistence of the data and identify the impact of the changes).

Finally, packages of integration should be properly defined in order to create an appropriate process by the combination of the single components. Parameters to be used in the packages must be identified properly, values must be defined and the couples parameter-value have to be updated when needed. From the realization of the system we aim at elaborating solutions for these problems and validating them by their application.

References

[1] M. Lindvall and I. Rus, “Process Diversity in software Development”, IEEE Software Vol.17 N°.4, IEEE Computer Society, Los Alamitos, July/August 2000, pp. 14–18.

[2] K. M.Dymond. A guide to the CMM-understanding the Capability Maturity Model for Software, Press Inc.

[3] M. T. Baldassarre, D. Caivano, C. A. Visaggio, and G. Visaggio, “promise: a framework for Process Models customization to the operative context”, proc. of IEEE International Symposium on Empirical Software Engineering, 2002, IEEE Computer Society, pp. 103–111.

[4] D. C.Schmidt, M. Fayad, and R.E. Johnson, “Software pattern”, Communications of the ACM Vol.39, No.10, ACM, New York, October 1996, pp. 37–39.

[5] B. Appleton, “Patterns and software: Essential concepts and terminology”, 2000. <http://www.enteract.com/~bradapp/docs/patterns-intro.html>.

[6] T. Winn, P. Calder, “Is this a pattern?”, IEEE Software Vol.19, N°1, IEEE Computer Society, Los Alamitos, January/February 2002, pp. 59–66.

[7 F. Houdek and H. Kempter, “Quality Patterns – An approach to packaging software engineering experience”, Proceedings of the 1997 Symposium on Software Reusability, ACM Software Engineering Notes Vol. 22, Num. 3, ACM, New York, Mai 1997, pp. 81–88.

[8] P. W. Fach, “Design Reuse through Frameworks and patterns”, IEEE Software Vol.18, N°.5, IEEE Computer Society, Los Alamitos, September/October 2001, pp. 71–76.

[9] L. Rising, “Patterns in postmortems”,Twenty-Third Annual International Computer Software and Applications Conference, IEEE Computer Society, Phoenix, Arizona, October 1999, pp. 314–315.

[10] M. Fredj and O. Roudies, “A pattern based approach for requirements engineering”, 10th International Workshop on database & Expert Systems Applications,. IEEE Computer Society, Florence, Italy, September 1999, pp. 310–314.

[11] S. M. Sutton, Jr., “The role of process in a Software Start-up” IEEE Software Vol.17 N°.4, IEEE Computer Society, Los Alamitos, July/August 2000, pp. 33–39.

[12] M. Morisio, C. Tully, and M. Ezran, “Diversity in Reuse Processes”, IEEE Software Vol.17 N°.4, IEEE Computer Society, Los Alamitos July/August 2000, pp. 56–63.

[13] V. R. Basili, G. Caldiera, and D. Rombach, “The Experience Factory”, Encyclopedia of Sofwtare Engineering – 2 Volume Set,1994, pp. 469–476.

[14] G. Visaggio, “Value-based decision model for renewal processes in software maintenance”, Annals of Software Engineering, 9 (2000), Kluwer Academic Publishers, pp. 215–233.

[15] G. Visaggio, “Ageing of a data –intensive Legacy System: symptoms and remedies”, Journal of Software Maintenance and Evolution: Research and Practice 13 (2001), John Wiley, pp. 281–308.

[16] A. Bianchi, D. Caivano, G. Visaggio. “Quality Models Reuse: Experimentation on Field”, Proceedings of the International Computer Software and Applications Conference (COMPSAC) – IEEE Computer Society, Oxford, England, August 2002.

[17] I. Rus, C. Seaman, M. Lindvall, “Process Diversity”, Journal of Software Maintenance and Evolution: Research and Practice, 15(2003) John Wiley, pp. 1–8.

[18] J. Munch, “Transformation-based Creation of Custom-tailored Software Process Models”, Proceedings of the 5th International Workshop on Software Process Simulation and Modelling 2004, ProSim 2004 May 2004

Data Architecture

A Disquisition on The Performance Behaviour of Binary Search Tree Data Structures

Dominique A. Heger

From a performance perspective, applications and operating systems are faced with the challenge to store data items in structures that allow processing fundamental operations such as insert, search, or remove constructs as efficiently as possible. Over the years, a variety of structures have been proposed, focusing on the efficient representation of data items. Some of the structures include direct addressing schemes such as hash tables, while others incorporate comparison schemes such as binary search trees. This study briefly elaborates on the internal characteristics of 5 tree-based data structures and focuses on their performance behaviour under various workload conditions. The conducted empirical studies revolve around expected run-time performance, as well as key-comparison and rotational behaviour. The goal was to identify the most efficient data structure under different workload scenarios. The 5 data structures chosen for this study represent 2 balanced (AA and red-black) and 3 unbalanced (treap, skip list, and radix) binary search tree implementations, respectively.

Keywords: Data Structures, Performance, Binary Tree

1 Introduction

Binary search trees are the most basic (non-linear) data structures utilized in the realm of application and operating system development. Their wide range of applicability can be explained by their fundamentally hierarchical nature, a property induced by their recursive definition. A binary tree structure can be defined as a finite set of nodes that are either empty or consist of a root and the elements of two disjoint binary trees, referred to as the left and right subtrees of the root. Binary tree structures support 2 primary application categories. First, they may represent hierarchical structures and second, they may be utilized to implement efficient data storage and retrieval mechanisms. In a generic setup, the individual tree components consist of 3 fields. First, a data field that holds the key. Second, a pointer to the root-node of the left subtree and third, a pointer to the root-node of the right subtree. In such a tree representation, NULL-pointers indicate empty subtrees, and the argument can be made that this representation is not space-efficient, as most of the pointers are referencing NULL. An alternative to such a design is known as a threaded binary tree structure, a tree construct that utilizes the space more effectively [12]. Instead of pointing to NULL, the leaf node pointers are linked to expedite lookup and tree traversal operations. In general, the challenge faced is to differentiate among the high-level tree features and operations, as well as the representation model, in an effective way that does not break the algorithms. Another venerable issue is that the tree balancing mechanisms (the maintenance operations per se) are from a performance perspective rather expensive, as well as complex to implement. The goal of this study was threefold. First, to quantify the performance behaviour of the red-black, the AA, the treap, the skip-list, and the radix tree data structures under varying workload conditions [5][17][20][21]. The focus was on implementation complexity, expected time complexity, key comparison, as well as on the restructuring operations (in the case of the 2 balanced binary search tree implementations). Second, to analyse the impact that some rather simple code-changes in the treap implementation have on the key comparison behaviour. Third, to quantify the performance delta of the tree traversal operation in a threaded and a non-threaded red-black tree environment. Of the discussed data structures, the AA and the red-black tree represent balanced structures, where all the individual operations (insert, remove, search) are bounded by an asymptotic upper bound of *O(log n)*. In the case of the treap, the radix, and the skip-list implementation, the underlying unbalanced binary search tree structures result in performing the individual operations in an expected time complexity of *O(log n)* as well. With theses 3 data structures though, an ergodicity of *O(n)* exists. Some other tree constructs such as AVL trees [5] or hash-based solutions were not incorporated into this study. The reader is referred to [27] for a comprehensive discussion on dynamic hashing.

2 Red-Black Trees

Binary search trees perform best when they are either balanced, or the path length from the root to any leaf node is within some bounds. The red-black tree algorithm represents a method for *balancing* trees [5]. Red-black trees are a variation of the classic binary

Dominique Heger has been with IBM for over 9 years. Prior to his work at IBM, he spent 5 years with Hewlett-Packard. His focus is on operating systems performance, performance modelling, algorithms and data structures, and I/O scalability. He has been part of several research projects that focused on UNIX scalability, and has published many systems performance and modelling related papers. He holds a PhD from NSU (Nova Southeastern University), Florida, USA, in Information Systems.
<dheger@us.ibm.com>

search trees (BST) that utilize a rather efficient mechanism for keeping the tree in balance. The name derives from the fact that each node is coloured *red* or *black*, and that the colour of the node is instrumental in determining the balance of the tree. During insert and delete operations, nodes may be rotated to maintain the tree balance. In general, both average and worst-case search time complexity equals to *O(log n)*. More specifically, the red-black tree design incorporates the following properties:

1 Every node is coloured red or black
2. The root node has to be black
3. Every leaf is a NIL node, and is coloured black
4. If a node is red, then both its children are black
5. Every simple path from a node to a descendant leaf contains the same number of black nodes

The number of black nodes on a path from the root to a leaf is known as the *black-height* of a tree. The properties mentioned above guarantee that any path from the root to a leaf is no more than twice as long as any other. All operations on the tree must maintain the properties listed above. In particular, operations that insert or delete items must abide within these very specific rules [5]. The amount of memory required to store a red-black node should be kept to a minimum. This is especially true if many small nodes are being allocated. In most cases, each red-black tree node has a left, a right, and parent pointer. In addition, the colour of each node has to be recorded. Although this requires only one bit, more space may be allocated to ensure that the size of the structure is properly aligned. To reiterate, on a static red-black tree implementation, the operations *minimum*, *maximum*, *search*, *successor*, *predecessor* can be executed in *O(log n)* time. Tree maintenance operations such as insert or delete require dynamic changes to the tree structure, and therefore require rather sophisticated implementations to meet the *O(log n)* time criteria. It has to be pointed out that a simple rotation is being executed in *O(1)* time. A threaded (red-black) search tree represents a data structure where the un-utilized child pointers are used to point to either the successor (right child pointer) or the predecessor (left child pointer) nodes, respectively. From an implementation perspective, the pointers have to be *flagged* to disclose if they represent a normal or a threading scenario [5]. One of the benefits of threading a tree structure is that it is feasible to process an in-order traversal in constant space, as it is not necessary to remember the entire path from the root to the current position. Therefore, a threaded tree structure represents a stack free solution that is beneficial if lookup (find) and tree traversal operations dominate the workload.

3 AA-Tree Data Structure

Andersson [1] introduced the AA-Tree design in 1993, as basically a quest to present new maintenance algorithms for balanced tree structures. Additional work by Weiss (1996) resulted into a much broader dissemination of the AA-Tree design [21]. The AA-Tree is considered as a simpler to code variant of the red-black tree and satisfies the following properties:

1. Every node is coloured red or black
2. The root node has to be black
3. Every leaf is a NIL node, and is coloured black
4. If a node is red, then both its children are black
5. Every simple path from a node to a descendant leaf contains the same number of black nodes
6. Left children may not be red.

The advantages of an AA-Tree design (compared to red-black trees) are that half the restructuring cases are eliminated, and that the delete operation is substantially simplified. In other words, if an internal node has only one child, that child has to be a red right child. Further, it is always possible to replace a node with the smallest child in the right subtree, as it either will represent a leaf node or it will have a red child. In the AA design, the balancing information is stored in each node as *the level*. The actual level is defined by the rules that (1) if a node is a leaf, its level is set to 1. (2) If a node is red, its level equals to the level of its parent. (3) If a node is black, its level equals to 1 less than the level of its parents. The level represents the number of left links to a NULL (or NIL) reference. The AA design further introduces the term horizontal link, in the sense that a *horizontal link* represents a connection between a node and a child with equal levels. In other words, horizontal links can be referred to as right references.

Based on the AA design, (1) it is not possible to have 2 consecutive horizontal links in the tree. (2) Nodes at level 2 or higher have to have 2 children, and (3) that if a node has 0 right horizontal links, its 2 children have to be at the same level. Compared to the red-black tree implementation, the vast number of rebalancing cases is simplified in the AA design by utilizing two rather simple maintenance operations labelled as *skew* and *split*. The skew operation removes left horizontal links, whereas the split operation addresses the issue of removing consecutive horizontal links (that are by design not allowed). Both operations are part of the AA insert and delete maintenance set. All the unbalanced situations that are imaginable in an AA-Tree based scenario can be eliminated by a sequence of at most 3 skew and 2 split operations, respectively. This statement holds true based on the fact that the maintenance work may affect a higher level, and therefore has to be propagated upward in a recursive manner. The fact that the left children may not be red greatly simplifies the delete operation (compared to the red-black paradigm), and therefore an AA-Tree solution should be considered if delete operations represent a significant portion of the actual workload profile.

Operation	Time Complexity
Find/Search/Access	O(log n)
Insert	O(log n)
Delete	O(log n)
Rotations per update	2
Update – rot. subtree s = O(s)	O(log n)
Update – rot. subtree s = O(s log^k s)	O(log^k+1*n)
Joining 2 trees (sizes m & n)	O(log max{*m,n*})
Splitting a tree (into size m & n)	O(log max{*m,n*})

Table 1: Treap Performance Characteristics – Generic Operations.

4 Treap Data Structure

A *treap* is the basic data structure underlying randomized search trees. The name itself refers to synthesizing a tree and a heap structure [5], [19]. More specifically, assuming that *x* represents a set of items where each item is associated with a *key* and a *priority.* A *treap* for a set *x* represents a special case of a binary search tree, in which the node set is

Operation	Time Complexity
Insert with handle	O(l)
Delete with handle	O(1)
Finger search over distance d	O(log d)

Note: handle, finger, split, and join operations require additional pointers.

Table 2: Treap Performance – Advanced Operations.

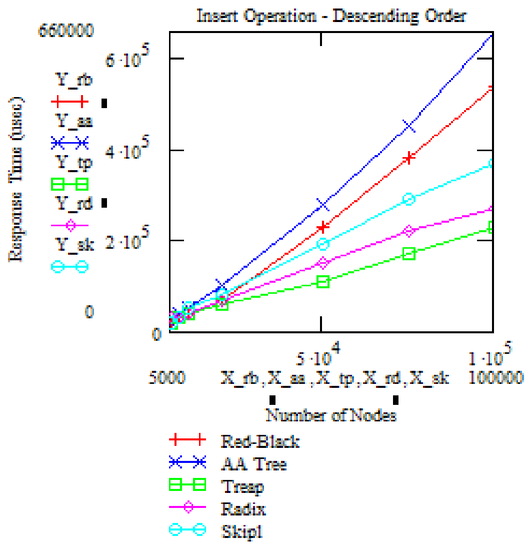


Figure 1: Insert Operations.

arranged *in order* (in respect to the keys) as well as in *heap fashion* (in regards to the priority). Further, assuming that *t* represents the treap structure storing a set of items *x*. Given the scenario where the key of an item is known, the location in *t* can easily be determined via a simple search tree algorithm. In a treap, the access or search time is proportional to the depth of an element in the tree. An insert of a new item *z* into *t* basically consists of a 2-step process. The first step consists of utilizing the item's key to attach to *t* at the appropriate leaf position and second, to use the priority of *z* to rotate the new entry up in the structure until the item locates the parent node that has a larger priority. The process of deleting an item *z* from a treap structure *t* represents the reversed scenario. The first step consists of locating the item, and second to rotate the item down in the tree structure until it becomes a leaf, where the item can be removed.

In some implementations, treap *split* and *join* operations may be necessary. A split operation is used to separate a set of items *x* into 2 sets (*x1* and *x2*). The separation utilizes a heuristic where items are being placed in the 2 sets based on the item's key values in comparison to the key of a reference element *a*. To accomplish the split, the operation inserts an item with key *a* that is affiliated with an infinite priority. Based on the heap-order property, the new item has to represent the root of the heap. Based on the *in-order* property, the left subtree represents the treap *x1*, whereas the right subtree represents the treap *x2*. In a similar fashion, the join operation is utilized to combine the two sets *x1* and *x2* into a single construct. The assumption made is that the keys in *x1* are smaller than the keys in *x2*. The implementation of the join operation creates a dummy root item, where the left subtree consists of *x1* and the right subtree represents *x2*. In a second step, the join operation performs a delete on the dummy root item, finalizing the combined treap structure. In some circumstances, *handles* or *fingers* are being used to expedite some of the maintenance operations. To illustrate, in the case that a handle is referring to a specific node *x*, deleting the node *x* can be accomplished by only rotating it down into a leaf position and freeing the item, circumventing the otherwise necessary search operation. In a similar fashion, to insert a new item *x* where a handle to either the successor or the predecessor *y* of node *x* is available, the search for the location for *x* can start at the reference point *y* (instead of at the root item). The term *finger search* for a node y in a treap refers to following the unique path between *x* and *y*, where node *x* incorporates a handle that points to it. Another aspect of treap implementations is that split and joint operations can be processed more efficiently if handles are available to the min and max key items, respectively. A randomized search tree that stores *n* items reveals the expected asymptotic upper bound time complexity (see Table 1 and Table 2).

The time complexity for a successful *search* operation in a treap environment is proportional to the number of ancestors of *x*, and can be expressed as *O(log n)*. An unsuccessful search for a key that falls between the keys of successive items (x^- & x^+) takes on an expected time complexity of *O(log n)* as well [14]. In order to *insert* an item into a treap, the first step is to locate its leaf position (based on its key value), and in second step to rotate the item up in the tree structure based on its priority. The number of rotations can at most be equal to the search path, hence the time to insert an item is proportional to the time required to complete an unsuccessful search, which as already discussed (in expectation) equals to *O(log n)*. In the case of a *delete* operation, the insert operation is being inverted, therefore the conclusion that the time complexity equals to *O(log n)*. The number of downward rotations during a delete operation equals to the sum of the length of the right spine of the left subtree of *x*, and the left spine of the right subtree of *x*, respectively. A scenario that (in expectation) is < 2 for a randomized binary search tree.

5 Skip List Data Structure

A skip list represents an ordered linked list, in which every node contains a variable number of links to other nodes in the structure [13][16]. To illustrate, the n^{th} link of a given node points to subsequent nodes in the list, and by design, skips over some number of intermediary nodes. Therefore, these skipped nodes have fewer than n links associated. As most nodes have a variable number of links, a skip list can be referred to as a collection of linked lists of different levels. In order to quickly traverse the structure, seeking for some target key, the search operation seeks on the upper level list until either the target data is encountered, or the operation locates a node with a key that is smaller than the target. At this point, that particular node links to a subsequent node. In this case, the search continues by repeating the same procedure (now starting at the node that incorporates the smaller value than the target) and by continuing on the skip list. Skip lists can be considered as a probabilistic alternative to balanced trees.

Skip lists have balance properties that are similar to the search trees that are built via random insertions. Balancing a data structure probabilistically is easier than explicitly

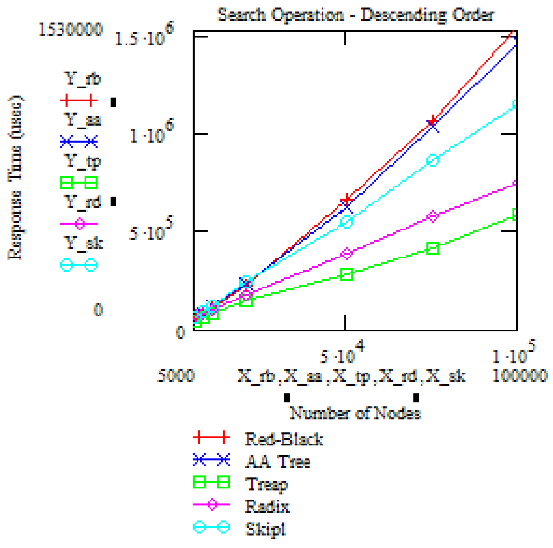


Figure 2: Search Operations.

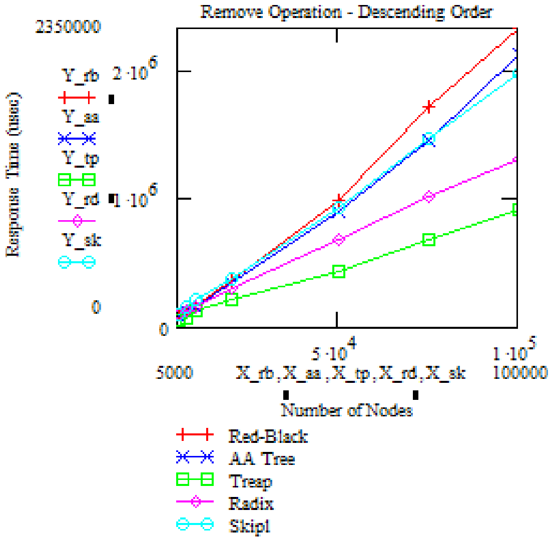


Figure 3: Remove Operations.

maintaining the balance. For many applications, skip lists represent a more natural layout than tree structures, and therefore are generally leading towards simpler algorithms. The ramification is that the simplicity of skip list algorithms allows easier implementations, and provides in some cases a (constant factor) speed improvement compared to the balanced and self-adjusting tree algorithms. Skip lists are rather space efficient. They can easily be configured to hold (on average) 1 1/3 pointers per element, and do not require balance or priority information to be stored within each node. The varying size of the nodes may be regarded as a disadvantage of skip lists. As a s skip list is balanced in a probabilistic fashion (by using a random number generator), the average search, insert, and delete operations are processed in an expected time complexity of *O(log n)*. The probability of encountering significantly worse performance is rather slim, but nevertheless exists. In other words, as the balance criteria is chosen randomly, the chance of encountering the *O(n)* worst case scenario is very small, as any input sequence into a skip list will not consistently produce the worst case performance scenario.

6 Radix Tree Data Structure

A standard radix search tree design is similar to a digital search tree [2][5][21]. However, in a radix search tree, all data items are stored as leave objects, and therefore the internal nodes of the radix tree do not have any key values associated with them. An internal node's child represents either another internal node or an actual data item. During a search operation, the *individual bits in the search key are examined*, and either the left or the right pointer to a child node is being activated according to the specific bit value. Therefore, unlike the digital search tree, the radix search tree does not have to encounter any key comparison overhead per se at each node that is being traversed. Instead, the traverse operation continues until the corresponding bit in a child node's *link filed* is zero. The child link entity refers to a two-bit field entry, where bits 0 and 1 specify the child pointers. In either case, if the bit value equals to zero, the pointer references either NULL or points to a data item. Otherwise, if the bit is 1, the pointer references another node in the radix tree. In other words, if the child link field equals to 0, either a NULL pointer or a data item has been located. Further, the root node always remains in the radix search tree, even in the case when there are no items in the tree. From a performance perspective, the number of nodes, as well as the length of the key value govern the efficiency of a radix tree. In general, large key values have a rather detrimental impact on performance.

Along these lines, the radix sort is a rather good illustration of how lists and deques can be combined with other container's [5]. In the case of a radix sort, a vector of deques is manipulated, similar to a hash table. In a radix sort, the values are successively ordered on a per *digit position* basis, normally from right to left (straight radix sort). This is accomplished by copying the values into buckets, where the index for the bucket is determined based on the position of the digit being sorted. The straight radix sort algorithm operates in *O(nk)* time, where *n* represents the number of items, and *k* refers to the average key length. The greatest disadvantage of a radix sort algorithm is that the implementation can not be constructed to execute *in place*. Therefore, *O(n)* additional memory space is required. Furthermore, a radix sort implementation requires 1 pass for each symbol of the key, and therefore is rather inefficient if long key values are processed. The reader is encouraged to consult [26] for a comprehensive discussion on radix trees, extendible hashing, B-Trees, and performance.

7 Benchmark Results

To conduct the performance comparison, all the data structures were implemented in *ANSI C*. The implementation of the data structures were based on work conducted in [5][12][16][17][21]. Where applicable, the same random number generator and the same seed were used throughout the study. All the data structures were exposed to the same workload scenarios. The analysis was decomposed in 3 sections. Section 1 focused on the individual insert, search, and remove performance. The operations were benchmarked either in an ascending, descending, or random order while scaling the number of nodes from 5,000 to 100,000. Next to the response time comparison, the study introduced the term *aggregate structure performance factor*, describing the mean performance of a data structure as quantified over the set of invocation scenarios used in this study. To illustrate, the insert performance was quantified based on ascending, descending, and random data distributions. Therefore, the overall consistency factor for the insert operation incorporates the 3 invocation scenarios. Section 1 further discusses the performance behaviour based on a mixed workload profile, consisting of a chain of insert, search, and remove operations, respectively. Section 2 quantified the data structure performance focusing on the number of key comparisons and (where applicable) the number of rotate operations. For the treap data structure, code changes surround-

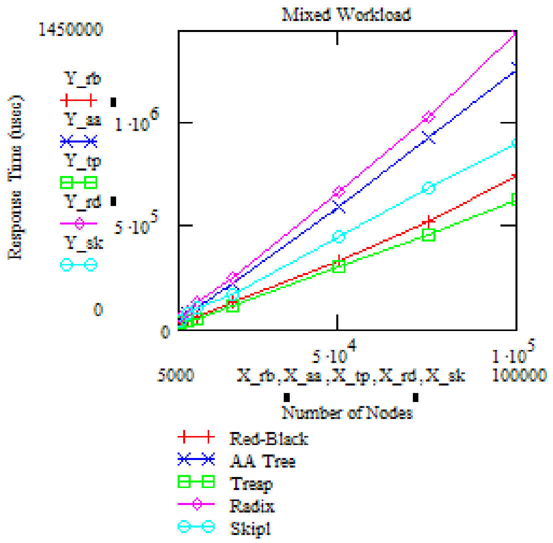


Figure 4: Mixed Workload.

ing the placement of the *equality*, the *less than*, and the *greater than* operations are proposed and analysed. Section 3 discusses the performance of the tree traversal operation, comparing a standard red-black tree and a threaded red-black tree implementation.

The test environments for the benchmarks in Section 1 and 2 consisted of a single CPU, 256MB memory, Linux 2.6 system that was equipped with a single disk configured with the XFS file system. For the benchmarks described in Section 3, a 4-way, 1GB memory system, configured with a 5-disk RAID-5 I/O subsystem that utilized the Linux 2.6 and the JFS file system was used. All the benchmarks were executed 100 times. The performance numbers reported in this study reflect the mean over all the test runs.

7.1 Insert, Search, and Remove Operations

The basic data structure benchmarks were conducted on the single CPU system. In the case where the nodes were inserted in descending order, the treap outperformed the other data structures by a rather significant margin (Figure 1). As discussed, the treap reflects a light-way data structure compared to either a red-black or the AA implementation, respectively. Therefor, the insert operations are completed more efficiently, as the expensive maintenance functions embedded in the balanced data structures are much more relaxed in a treap implementation. The delta between the fastest (the treap) and the slowest (the AA tree) structure equalled to 430 milliseconds (at the 100,000-node level). At the 10,000-node level, all 5 data structures reported mean response time values within 10 milliseconds. In the case the nodes were either inserted in an ascending or random order, the radix tree proved to be the most efficient solution (Appendix A).

From a *structure performance factor* perspective, at the 100,000-node level, the radix tree's insert operations outperformed the other data structures. Further analysing the fluctuation among the different insert scenarios (ascending, descending, and random) revealed that the red-black tree performed most consistently. At the 100,000-node level, the fluctuation among the ascending, the descending, and the random insert operations was approximately 40 milliseconds. This can be compared to a delta of 440 and 130 milliseconds for the treap and the radix tree, respectively. The insert benchmarks disclosed that the skip list and the AA tree experienced scalability issues, especially in the random insert scenario.

Nodes	Operation	Treap	Radix	Skip
10,000	Insert	98,771	299,953	247,439
	Search	169,977	320,032	251,764
	Remove	108,758	309,974	184,254
50,000	Insert	555,109	1,499,955	1,420,190
	Search	1,007,434	1,600,032	1,489,230
	Remove	605,099	1,549,975	1,512,343
100,000	Insert	1,219,770	2,999,956	2,932,223
	Search	2,070,369	3,200,032	2,969,512
	Remove	1,319,758	3,099,975	2,865,642

Table 3: Key Comparisons – Unbalanced Data Structures.

The benchmarks conduced revolving the search operations in a descending order revealed a similar picture (Figure 2). From a mean response time perspective, the treap data structure outperformed the skip list, as well as the radix tree, whereas the latter two data structures were able to outperform the more complex red-black and AA tree structures. At the 100,000-node level, the delta between the fastest (the treap) and the slowest (the red-black tree) data structure was 960 milliseconds. At the 10,000-node level, the difference between the most (the treap) and the least (AA and skip list) efficient implementations equalled to 40 milliseconds. The search benchmarks conducted in ascending order disclosed a similar behaviour as experienced for the insert operations (see Appendix A). The radix tree outperformed the treap, which outperformed the other 3 implementations. At the 100,000-node level, quantifying the aggregate *structure performance factor* showed the radix tree and the treap in a dead heat. At the same time, the other 3 data structures trailed by 530 milliseconds, 725 milliseconds, and 790 milliseconds for the skip list, the AA, and the red-black tree structures, respectively.

From a consistency perspective (smallest delta between the search scenarios), the radix tree outperformed the red-black and the AA tree, which outperformed the treap and the skip list. At the 100,000-node level, the delta between the ascending and the descending search operations was 10 milliseconds for the radix tree, 150 milliseconds for the red-black tree, and 380 milliseconds for the treap, respectively. Benchmarking the remove operations in descending order revealed that the treap was once again capable of outperforming the other 4 data structures (Figure 3). The mean delta between the treap and the slowest structure (red-black tree) at the 100,000-node level equalled to 1,430 milliseconds. At the 10,000-node level, the difference between the treap and the least efficient implementation (skip list) equalled to 90 milliseconds. Analysing the remove performance in ascending

Nodes	Operation	AA	Red-Black
10,000	Insert	168,244	211,383
	Search	114,707	128,853
	Remove	114,037	103,671
50,000	Insert	1,016,999	1,286,225
	Search	685,766	754,794
	Remove	697,229	639,197
100,000	Insert	2,183,976	2,772,389
	Search	1,471,511	1,609,564
	Remove	1,494,441	1,378,359

Table 4: Key Comparisons – Balanced Structures.

Nodes	Operation	AA	Red-Black
10,000	Insert	9,982	9,976
	Height	18	24
	Remove	3,340	2,489
50,000	Insert	49,976	49,971
	Height	21	29
	Remove	16,676	12,468
100,000	Insert	99,987	99,969
	Height	22	31
	Remove	33,339	24,985

Table 5: Height and Rotations – Balanced Structures.

order disclosed the treap as the most efficient implementation (Appendix A). As the mathematical and structural analysis of the red-black and the AA tree design suggested, the AA tree outperformed the red-black implementation in every remove scenario that was benchmarked in this study. Analysing the aggregate *structure performance factor* at the 100,000-node level showed the treap with the lowest mean response time, followed by the radix, the skip list, the AA, and the red-black tree.

From a consistency perspective (smallest delta between the remove scenarios), the radix and the red-black tree outperformed the other 3 implementations, underlying the robustness of these data structures under different operation patterns. To further quantify the performance behaviour of these data structures, the study utilized a mixed workload profile. The profile triggered a chain of insert (100% of the nodes in ascending order), search (randomly for 10% of the nodes), remove (randomly for 50% of the nodes), and search (randomly for 10% of the nodes) operations. The conduced benchmark runs showed that at every node level, the treap outperformed the other 4 data structures (Figure 4). At the 100,000-node mark, the treap outperformed the slowest data structure (radix tree) by 820 milliseconds. Decomposing the conducted test runs into a small (5,000 to 50,000 nodes) and a large (greater than 50,000 up to 100,000 nodes) category, and conducting the analysis accordingly did not change the performance picture in any significant way. In the small mixed workload category, the treap represents the most efficient implementation, whereas the radix tree encounters a rather steep increase in response time at the 20,000 and the 50,000-node levels, respectively. The same behaviour is reflected in the large mixed workload category.

Overall, the red-black tree performed well at every node level, as the data structure was capable of outperforming the more light-way implementations of the radix tree and the skip list at every measured data point.

7.2 Key Comparisons and Rotations

The next few experiments in this study focused on the number of key comparisons performed by the data structures while processing a certain workload. The results in Table 3 and 4 outline key comparisons for a descending permutation, attempting to model a realistic situation where the inserted elements are in a nearly sorted order. Evaluating the mean number of key comparisons (across the 3 operations) showed the treap as the most efficient implementation at all the benchmarked node levels. The radix tree represents the structure that processes the most key (actual bit) comparisons. Despite processing more key comparisons, the simplified AA remove function outperforms the red-black tree implementation from a response time perspective. As the design suggests, the 2 balanced tree structures disclose the lowest number of key comparisons for the search operation. The number of key comparisons processed by the 2 balanced structures (while operating on remove scenarios) are in line with the most efficient (treap) unbalanced data structure, and clearly outperform the other 2 (radix and skip list) solutions. The rebalancing operations necessary in these 2 data structures though squander that advantage, which is reflected in the response time behaviour (Figure 3). Evaluating the key comparison behaviour on random data sets revealed that the 2 balanced data structures outperformed the 3 unbalanced solutions. The analysis showed that the red-black tree slightly edged the AA implementation at all the benchmarked node levels (see Table 3 and Table 4).

In order to further investigate the key comparison behaviour, and the impact on response time, the study varied (in the treap solution) the order in which the *equality*, the *less than*, and the *greater than* operations were processed. The following pseudo code documents the 2 experiments conducted for the treap search operation.

```
Option 1:
1 f key_searched = key_current then found
2 else if key_searched < key_current go to left child
3 else go right

Option 2:
1 f key_searched = key_current then found
2 else if key_searched > key_current go to right child
3 else go left
```

The benchmarks conduced for the treap search operation (at various node levels and random input sets) revealed that option 2 outperformed option 1 (response time wise) by approximately 4%. The study further showed that moving the comparison in line 1 further down and executing the greater than operation first, results in fewer key comparisons but a higher overall response time.

For the red-black and the AA structures, Table 5 discusses the height and the number of rotations executed at different node levels with a descending input set. Both structures revealed almost identical numbers of rotations while inserting the data items. While processing remove operations, the red-black tree executed approximately 25% less rotations than the AA implementation. In all the benchmarks utilizing ordered data sets, the AA tree presented a significantly flatter tree hierarchy than the red-black implementation. As outlined in Table 5, a height delta of 6, 8, and 9 at the 10,000, the 50,000, and the 100,000 node-levels was reported. Studies conducted on a random data set revealed that the red-black tree executed on the insert as well as the remove operations fewer rotations than the AA tree. Further, with a random sample set, the height of the tree structures only varied by 2, 3, and 3 at the 10,000, the 50,000, and the 100,000-node levels, respectively (see Table 5).

The benchmarks revealed that the compiler, the systems architecture, and the time complexity of the key comparisons significantly impacts the response time behaviour. The search operations for all the tree-based structures were essentially identical. Despite the similar search solutions, methods that executed fewer key comparison operations not always revealed the most efficient response time behaviour. Processing 50,000 search operations in ascending order resulted in 1,600,032 and 1,067,079 key comparisons for

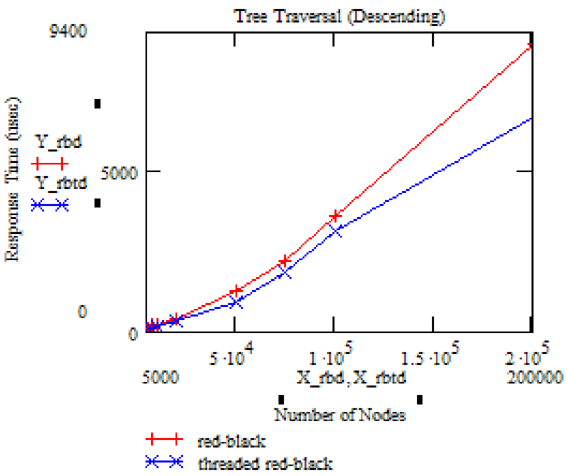


Figure 5: Tree Traversal.

the radix tree and the treap data structures, respectively. From a response time perspective, the radix tree outperformed the treap for this data point by 110 milliseconds. Similar tests conduced on an larger SMP system (running a commercial UNIX flavour) revealed that based on the different compiler architecture, instruction pipelining features, and cache replacement polices, a slightly different execution behaviour of some of the data structure operations. The results presented in this Section for the treap, the skip list, and the red-black tree data structures are comparable to the performance data reported in studies conducted by Sahni [24] and Papadakis [25].

7.3 Threaded Red-Black Tree Performance

The final experiment conducted in this study focused on quantifying the performance behaviour of the tree traversal operation utilizing a regular and a threaded red-black tree solution. The benchmark was conducted on the SMP system discussed in Section 6.0. The benchmark results depicted in Figure 5 reveal the improved traversal behaviour of the threaded red-black solution. Additional tests conducted on random and ascending data sets disclosed the same pattern, as in all the experiments, the threaded implementation outperformed the generic red-black data structure by an average of 12% (Appendix A). Analysing the insert performance of the 2 data structures showed the complexity increase of maintaining the additional pointers in the threaded solution though, as the regular red-black tree implementation outperformed the threaded data structure by an average of 5%.

8 Summary and Conclusion

The empirical analysis conducted for this study supports the mathematical abstractions for the tree data structures. In other words, the theoretical study of the tree structures and the resulting performance claims were highlighted through the conducted benchmarks. To illustrate, the AA implementation was capable of outperforming the red-black tree structure in all the remove cases. To summarize, in the mixed workload environment, the treap data structure outperformed the other 4 implementations by a rather significant margin. In the insert scenarios, the radix and the treap structure outperformed the more complex AA and red-black data structures. A similar picture was drawn by the search operation based benchmarks. It is interesting to point out that in the case of random insert operations, the red-black tree outperformed the treap in the case that 20,000+ nodes were populated into the data structure. The study revealed that based on the mixed workload profile, the treap represents the most efficient implementation whereas overall, the red-black data structure excelled from a consistency perspective. In other words, the red-black tree provided a rather consistent response time behaviour under varying workload patterns. The more light-way data structures (such as the treap or the skip list) on the other hand were rather fluctuation prone (a statement made based on the ascending, descending, and random operations executed at the same node level). Based on the mathematical and empirical study, the ramification is that the implementation of a red-black tree data structure in operating systems is considered as an effective and (to a lesser extent) efficient solution. The consistency factor reported in this study more than justifies the usage of a red-black tree implementation. To address the efficiency issue, one approach may be to explore the possibility of converting a standard red-black tree data structure into a threaded implementation, a design change that would allow expediting the search and traversal operations. As a search operation is part of any remove scenario (the node has to be located first), the remove operation benefits from the enhancement as well. The actual tradeoff revolves around faster lookup operations and increased pointer maintenance. Further, the treap has to be considered as a valuable alternative to any data structure. To illustrate, despite all 3 unbalanced solutions representing rather simple data structures, the treap outperformed the radix tree and the skip list in the mixed workload scenarios, whereas the radix tree represented the least efficient solution of all the benchmarked data structures. This study further discussed the performance gain that is possible by re-ordering the logical operations used in the data structures, and addressed the impact of compiler, systems architecture, and time complexity on the response time behaviour.

References

[1] A. Andersson. “Balanced Search Trees Made Simple”, WADS, 1993.

[2] A. Andersson, S. Nielsson. “A New Efficient Radix Sort”, FOCS, 1994.

[3] S. Baase. “Computer Algorithms, Introduction to Design and Analysis”, 3rd ed., Addison-Wesley, 2000.

[4] M. Black. "Skip Lists vs. B-Trees”, CSI Essex, 2001.

[5] T. Cormen. “Algorithms”, Second Edition, MIT Press, 2001.

[6] M. Garey, D. Johnson. “Computers and Intractability: A Guide to the Theory of NP-Completeness”, Freeman, 1979.

[7] G. Gonnet, R. Baeza-Yates. “Handbook of Algorithms and Data Structures”, 2nd. ed., Addison-Wesley, 1991.

[8] R. Graham, D. Knuth, O. Patashnik. “Concrete Mathematics”, Addison-Wesley, 1989.

[9] T. Hagerup, C. Rueb. “A Guided Tour of Chernoff Bounds”, 1990.

[10] D. Hochbaum. “Approximation Algorithms for NP-Complete Problems”, PWS, 1997.

[11] E. Horowitz and S. Sahni, Fundamentals of Computer Algorithms, Computer Science Press, 1978.

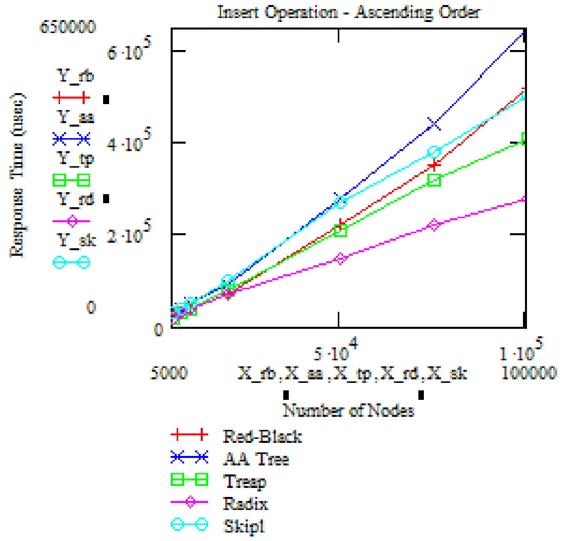
[12] D. Knuth. “The Art of Computer Programming”, Volumes 1 and 3, Addison-Wesley, 1997 and 1998.

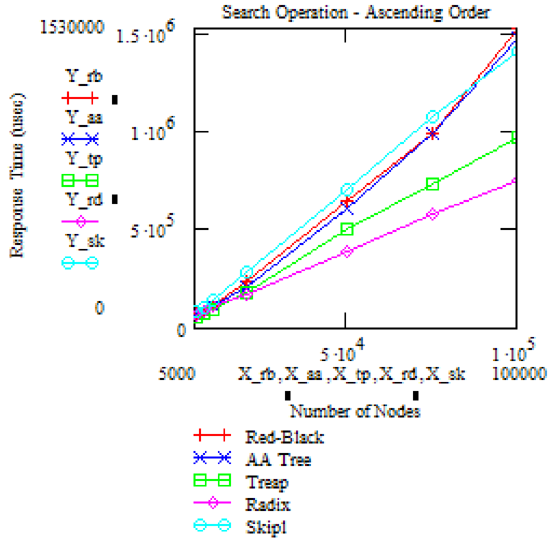
[13] P. Messeguer. "Skip Trees, an Alternative Data Structure to Skip Lists in a Concurrent Approach", 1997.

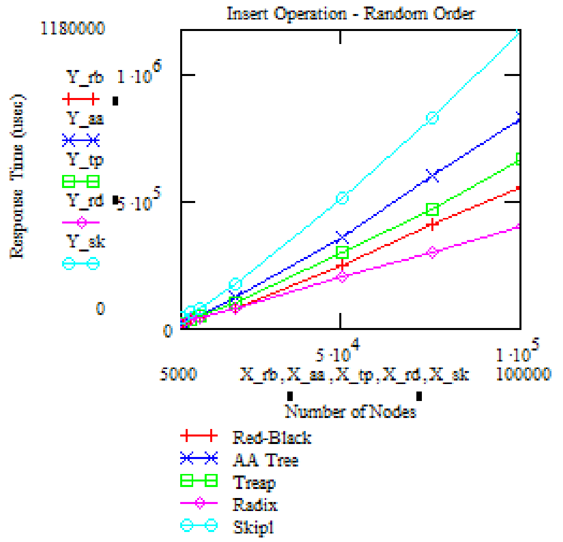
[14] R. Motwani, P. Raghavan. “Randomized Algorithms”, Cambridge Univ. Press, 1995.

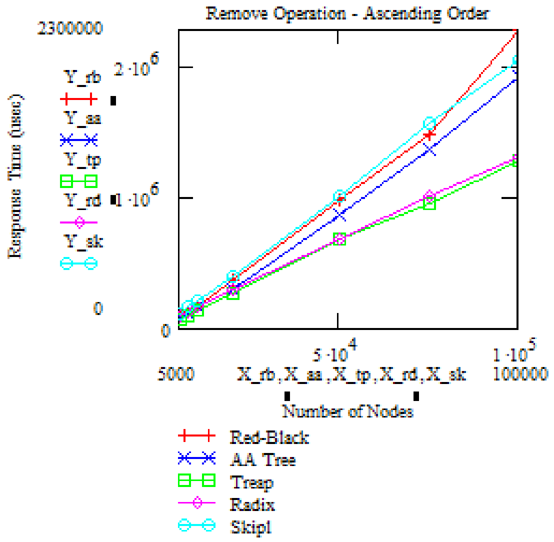
[15] C. Papadimitriou. “Computational Complexity”, Addison-Wesley, 1994.

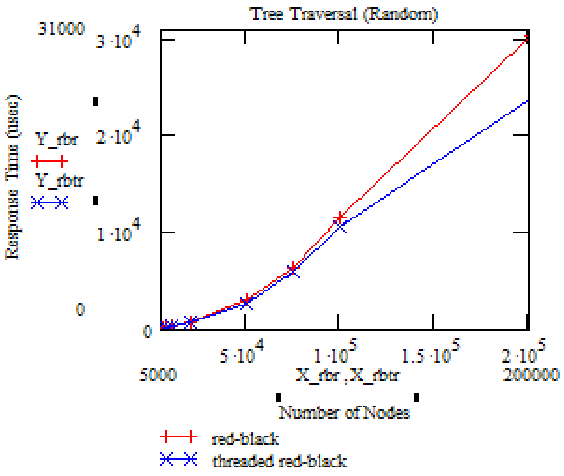
Appendix A: Additional Benchmark Charts

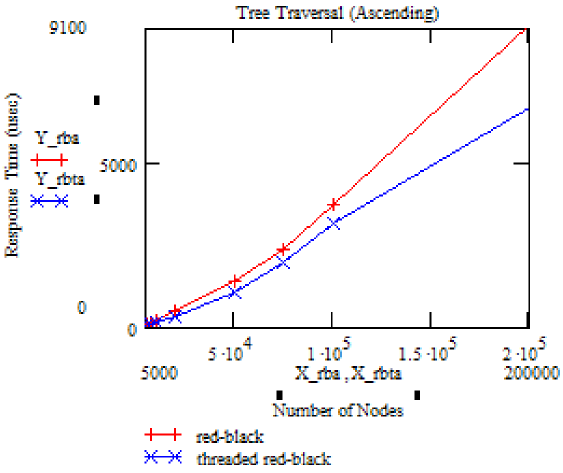












[16]
W. Pugh. “Skip Lists – A Probabilistic Alternative to Balanced Trees”, ACM, 1990.

[17]
R. Sedgewick. “Algorithms” 2nd ed., Addison-Wesley, 1988.

[18]
R. Sedgewick, F. Lajolet. “An Introduction to the Analysis of Algorithms”, Addison Wesley, 1996.

[19]
R. Seidel, C. Aragon. “Randomized Search Trees”, Algorithmica 16, 1996.

[20]
C. Van Wyk. “Data Structures and C Programs” Addison-Wesley, 1988.

[21]
M. Weiss. “Data Structures and C Programs” Addison-Wesley, 1997.

[22]
N. Wirth. “Algorithms + Data Structures = Programs”, Prentice Hall, 1978.

[23]
A. Harrison. “VLSI Layout Compaction using Radix Priority Search Trees”, 1991.

[24]
S. Sahni, S. Cho. “A New Weight Balanced Binary Search Tree”, University of Florida, TR 96-001, 1996.

[25]
T. Papadakis. “Skip Lists and Probabilistic Analysis of Algorithms”, Ph.D. Dissertation, U. of Waterloo, 1993.

[26]
R. Fagin, J. Nievergelt, N. Pippenger, H. Strong. “Extendible Hashing – A Fast Access Method for Dynamic Files”, ACM Transactions on Database Systems, 1979.

[27]
R. Enbody, H. Du. “Dynamic Hashing Schemes”, ACM Computing, 1998.

News & Events

CEPIS Present in the European e-Skills 2004 Conference
Long Term Strategies for E-Skills Development in Europe (Press Release)

The European Union should adopt a comprehensive strategy for improving ICT skills and training across all sectors, at all levels and for all citizens. This was one of the main messages of the European e-Skills 2004 Conference which ended Tuesday, 21 September at Cedefop in Thessaloniki, Greece. More than 150 experts took part in this major event, two years after the European e-Skills Summit organised by the Commission and the Danish Presidency in Copenhagen in 2002.

Among the participants, there were several representatives of EU Member States and acceding countries, of five Directorates General of the European Commission (Enterprise and Industry, Education and Culture, Employment and Social Affairs, Information Society and Eurostat) as well as the European Investment Bank, senior executives from leading ICT companies such as Microsoft, Nokia, Cisco Systems, IBM, Certiport, CompTIA etc. and researchers, academic and training world, representatives of European and international professional ICT associations (Council of European of Professional Informatics Professionals), consortia (Career Space, e-Skills Certification Consortium, e-Learning Industry Group, Project Management Institute) and delegations of the social partners (EICTA, Uni Europa and European Metal Workers Federation).

The Synthesis Report of the European e-Skills Forum “E-Skills in Europe: Towards 2010 and beyond” which constituted the basis for the discussions during this event pinpointed the threats of moving European ICT jobs to low-cost countries such as India and China (offshore or international outsourcing). For example, it is expected that by 2010 about 272.000 jobs will be lost in the UK alone due to international outsourcing. There is a tendency for companies to outsource services such as call centres, commercial handling and accounting, to countries with low labour costs. Central and Eastern European Countries and the new Member states, notably the Czech Republic, are increasingly attracting foreign direct investments from ICT companies because of their comparatively lower level of salaries and relative high skill level of their labour force. Highly skilled people are also being recruited in Europe, however, at lower speed.

This creates a serious dilemma for the EU Member States. On the one hand, their firms can lower labour costs by moving (in part or entirely) to low-cost countries, and thus improve competitiveness internationally. But at the same time, losing jobs in the ICT sector threatens social cohesion: ICT has been the main source of new employment in a time when more traditional sectors have been shedding employment opportunities. Mismatches and skill gaps persist however as many ICT jobs remain vacant due to the lack of qualified personnel. The number of current ICT specialists in Europe is 3.7 million and is estimated to reach 5.1 million by 2010.

Among priority actions discussed for 2005, the European e-Skills 2004 Conference also concluded that the European Commission should support alongside Cedefop and industry partners a “European level ICT skills meta- or reference framework” for better planning of investments in training and skills and must also further develop common principles for quality standards and for certification, whether public or private, profit or non-profit oriented. For these purposes it was proposed to create a European network of e-skills experts and a policy advisory group to develop foresight scenarios and further promote e-skills policies at the European level. The conference also proposed the creation of a European ICT career portal and of a central link between all educational institutions working in ICT, whether public or private.

More information at <http://www.eskills2004.org/>.

September 24, 2004

EUCIP News

Norway: EUCIP[1] and Abelia – Measuring Life Long Learning

Abelia (the Association of Norwegian ICT and Knowledge-based enterprises) was host to the conference “Measuring Life Long Learning” held on 14 September in Oslo in conjunction with partners Mintra AS, Norsk Test, EUCIP Norway, NITH and Energibedriftenes Landsforening (EBL).

The schedule contained sessions on net-based and interactive testing and a presentation about EUCIP by Renny Bakke Amundsen, in conjunction with one of Norway’s largest learning providers NITH who are accredited to run the EUCIP programme in Norway.

13th September 2004

The 9th World Multi-Conference on Systemics, Cybernetics and Informatics: Call for Papers

This conference will take place in Orlando, Florida, USA, from July 10–13, 2005.

SCI 2005 is an international forum for scientists and engineers, researchers and, consultants, theoreticians and practitioners in the fields of Systemics, Cybernetics and Informatics. It is a forum for focusing into specific disciplinary research, as well as for multi, inter and trans-disciplinary studies and projects. One of its aims is to relate disciplines fostering analogical thinking and, hence, producing input to the logical thinking.

The conference´s Call for papers can be found at <http://www.iiisci.org/sci2005/website/callforpapers.asp>.

The best 10% of the papers will be published in Volume 3 of SCI Journal, <http://www.iiisci.org/Journal/SCI/Home.asp>. 12 issues of the volumes 1 and 2 of the Journal have been sent to about 200 university and research libraries. Free subscriptions, for 2 years, are being considered for the organizations of the Journals’ authors.

We are emphasizing the area of Wireless/Mobile computing.

You can find information about the suggested steps to organize an invited session in the Call for Papers and in the conference web page: <http://www.iiisci.org/sci2005>.

If by any reasons you are not able to access the page mentioned above, please, try the following pages:
<http://www.iiis.org/sci2005>.

More information at
<http://www.iiisci.org/sci2005>.

[1] EUCIP (European Certification of Information professionals, <http://www.eucip.com>) is a new pan-European qualification scheme, promoted by CEPIS, for people entering the IT profession and for IT professionals wishing to continue their professional development. EUCIP has been developed as an independent, globally recognised scheme for IT professionals in a similar fashion to the ECDL (European Computer Driving Licence) which is aimed at the IT User. The qualification will enable existing IT professionals to document their competencies and skill sets for employers or prospective employers and in addition, increase their market value.

UPGRADE European NETwork

The network of CEPIS member societies' publications

IT and Disabilities

Braille and The Pleasure of Reading: We Blind People Want to Continue Reading with Our Fingers

Carmen Bonet-Borrás

This paper was first published, in its original Spanish version, under the title *“El Braille y el placer de la lectura: los ciegos queremos seguir leyendo con los dedos”*, by **Novática** (issue no. 169, May–June 2004, pp. 67–72). **Novática**, <http://www.ati.es/novatica>, a founding member of **UP**ENET, and publisher of **UP**GRADE on behalf of CEPIS, is the bimonthly journal and magazine of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*, <http://www.ati.es>.)

In this article the author describes her long experience as a user of Braille, the language for the blind, and of other technological aids for the blind. The author expresses her love of Braille and her conviction that, in spite of all the major technological advances there have been in this field, Braille will continue to be an essential tool for the human and intellectual development of people who suffer from visual impairment, and a gateway to enjoyment and culture.

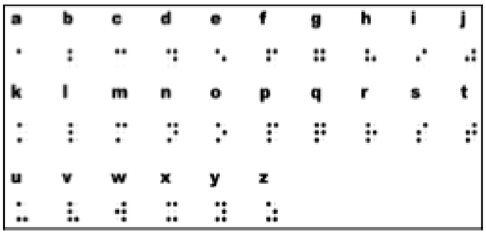


Figure 2: The Alphabet Braille – Spanish Version (Source: The Caragol Foundation).

Keywords: Braille, Blindness, Personal Experience, Digital Technologies Aiding the Visual Impairment, Visually Impaired People.

1. Presentation

This article is based on my personal experience over many years as an IT professional and a user of special tools for the visually impaired due to my having been totally blind since the age of six (see photo of the author in Figure 1). I am, therefore, an experienced user, though I am not an expert in disabilities and new technologies.

Figure 1: Photo of The Author at Age 6, for Her ONCE (Spanish National Organization for the Blind) Card.

2 What Is the Braille System?

When in the first half of the 19th century, while he was barely an adolescent, the Frenchman Louis Braille laid the foundations for the system of reading and writing named after him, the Braille system, a code in which letters are represented by raised dots (Figure 2), he could not have suspected that nearly two centuries later it would still be a sufficiently interesting and topical subject to be worthy of your attention in this journal. During the intervening time, blind people all over the world have been using the Braille method to read and write, and no other system has ever threatened to replace it. And, in my opinion, this is no accident; the fact is, the system is hard to improve on. For those of us without sight, what could be better than to read with our hands and, our fingers being the way they are, we are not equipped to use *smaller or thinner* letters, or use characters of a more complicated design.

I remember the **Optacon** (Optical-to-Tactile Converter), which was probably the first device to put electronic technology at the service of blind people. It was, and in fact still is (I still use mine) a machine developed in 1963 at Stanford University, California, by Prof. John Linvill, which worked by scanning printed paper and making hundreds of little

Carmen Bonet-Borrás has a BSc in Mathematics from the *Universidad Complutense de Madrid*, Spain. She is Systems Engineer at IBM España, member of ONCE (*Organización Nacional de Ciegos de España* – Spanish National Organisation for the Blind), and member of the Working Group on Informatics and Disabilities of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*.) <carmen.bonet@is.ibm.com>

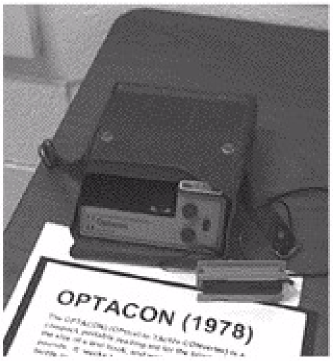


Figure 3: The Optacon Device (1978 Version).

pins in the shape of the letters vibrate on a pad under the user's index fingertip (Figure 3).

However, this device had a very short life; it came on the market in the late 70s and was useful to some of us for a while. It was the tool with which I started out on my career as an IT professional, although, in spite of its great intrinsic value as a tool enabling blind people to read printed texts directly, it did not really catch on, mainly due to its poor performance. Because of the complex shape of letters and the variables created by varying print quality, different types of paper, etc., the reading speed attained by most users was not high enough to meet their everyday needs. And over time, new developments offering access to reading materials have taken its place.

Meanwhile, Braille lives on.

And not only does it live on, but it has been given new life, has been renewed and is growing under the influence of the new technologies. The arrival of computers gave rise to the idea of computerized Braille and whole new horizons opened up. The number of different characters which can be represented by 6 dots is 64. For literature, by using the trick of sometimes doubling up characters, this was enough, but for computer use it is simply not sufficient. For a start, doubling up characters is not viable because it causes an imbalance between the original text and the Braille representation (screen, Braille display), which could create problems, and there are also far more characters to represent. The solution was to switch to an 8 dot Braille system, giving 256 characters, which is another kettle of fish altogether.

But there is an unresolved issue concerning Braille: each country has drawn up its own character correspondence table and there is an urgent need for these tables to be standardised and for the differences between one country and another to be ironed out as soon as possible. I recently heard that the ONCE (*Organización Nacional de Ciegos de España* – Spanish National Organisation for the Blind, <http://www.once.es>) is addressing this problem, so I expect some progress to be made in this area soon.

Innovative new ways of producing the raised dots used in the Braille system are also being tried out. The normal method up until now has been to perforate the paper, but there has been a constant search for alternative methods which, little by little, is bearing fruit. The application of new serigraphic techniques, whereby tiny drops of very special plastic substances are stuck to the paper in order to achieve an equally legible relief printing effect, is particularly interesting. This technique is being used for visiting cards, for example, and it may be a good alternative for certain situations such as labelling on cardboard, or for other cases where the text is short and repetitive.

2.1 Who Uses Braille?

Braille is a reading and writing system for all visually impaired people whose residual vision is not sufficient to make use of the methods available to people with unimpaired sight. We should also include in that group visually impaired people who, while still able to manage a pen, have been diagnosed as having a certain risk of their residual vision being reduced or lost altogether.

It is true that there is a certain relation between age and the difficulty of learning Braille due to physiological reasons, since the sense of touch of an adult's finger does not develop in the same way as that of a child. This may give rise to a degree of resistance in some adults with acquired blindness, to the extent that the sense of touch of a person who goes blind very late in life may not even develop enough to cope with Braille at all. But, in any event, everyone should try to learn it, since everyone, whether blind or sighted, should be able to read and write. We should do everything in our power to prevent adults who lose their sight from becoming what we have come to call *functional illiterates*; in other words, people who know how to read and write but are unable to make use of that skill. We should therefore make a special effort to break down the resistance to learning of sufferers of late onset blindness, and involve and commit them to learning Braille by making every possible resource available to them.

There is another group of people for whom the use of Braille is even more vital. These are the deaf and blind, for whom listening as a substitute for reading in order to access the written word is not an option.

Another group of blind people who would find life very difficult without Braille are musicians and music students. Musical notation is complex enough to begin with, but when it comes to converting all the information contained in a five-line staff into dots, I cannot even begin to imagine what it would be like 'listening' to an entire score.

2.2 Braille and Daily Life

But is Braille so important? Braille is by turns convenient, necessary or indispensable, from first thing in the morning to the last thing at night. If, as I believe we should, we intend to give blind people the greatest degree of self sufficiency in their everyday life (as in any other facet of life), such self sufficiency will be all the more viable and real the more knowledge they have of their environment and the less they have to depend on the help of a sighted person.

So, if the first thing we do after getting up is to get under the shower, what better way to start the day than by being able to identify the shower gel or hair conditioner by its name, printed on the containers in Braille by the manufacturers? If I have two similar pairs of trousers and I don't want to put the wrong pair on by mistake, the one which doesn't go with the coloured sweater I've picked out to wear, wouldn't it be good if easily confused clothes could be labelled in Braille with their colours? If I want to choose between full fat and semi-skimmed milk, or pick out the CD I want to listen to, I have no option but to label them, in Braille of course, because I can think of no other way round the problem. And so on, throughout the whole day.

Packaged foods, canned and bottled foods, soft drinks... should all reach the shops appropriately labelled in Braille, especially medicines due to the particularly dangerous consequences of a mix up.

With perseverance and effort, and by using our memory to the full, trying to be very ordered and systematic, and inculcating the people with whom we live with the same attitudes, we can alleviate the problem of not being able to read. But there can be no doubt that the correct identification of the things around us would be a major contribution to our quality of life.

This idea is far from being a fantasy. Products from brands such as Pescanova (frozen foods) or Doña Jimena (confectionery and biscuits), Sanex (soap) or some models of Newpol washing machines, plus a number of medicines, have turned this idea into a reality. We need to keep up the pressure so this becomes the rule rather than the exception. A good example to follow is that of lifts: there is a regulation regarding safety measures, dimensions, the control panel... and it is nice to be able to go to a building and reach the door you want without it having to be on the ground floor.

It is up to blind people to be skilled in Braille and benefit from this effort. That way we can avoid having to change the menu at the last minute because that can of tomato sauce we opened for spaghetti turned out to be green beans. Reading, as a gateway to culture, study and as a leisure activity in general, plays an important role in everyday life, and if you can read in Braille, so much the better.

2.3 Reading by Listening

Isn't listening to a talking book reading? If we look up the Spanish words *leer* (read), *oír* (hear), and *escuchar* (listen) in the Spanish Royal Academy dictionary, 2001 edition, we find the following definitions:

Leer (From the Latin *legere*).

1. Vb tr. To look at something written or printed while understanding the meaning of the characters used.

2. Vb tr. To understand the sense of any kind of graphic representation. To read the time, a musical score, a plan.

3. Vb tr. To understand or interpret a text in a certain way.

Oír (From the Latin *audire*).

1. Vb tr. To perceive sounds with the ear.

Escuchar (From the vulgar Latin *ascultare*, Latin *auscultare*).

1. Vb tr. To pay attention to what is heard.

We can see that according to the first two definitions of the word *leer* (read), the answer to the above question would be no, but according to the third definition the answer would be yes. Far from being contradictory, the two answers are in fact complementary. It is a matter of addition, not confrontation. The most important thing, over and above any other consideration, is to access information, and if we are discovering the literal content of a book, there can be no objection to calling that reading, whatever the method used to access a book. Nevertheless, since everything is relative, I would choose listening as better than nothing, but I would go for reading over listening every time.

Ideally each user would be able to choose, something which doesn't happen very often. After many years' experience of doing both things, I am in no doubt about my priorities and I have to admit that, although there may be occasions when I decide to read by listening, these are few and far between, because you can get into a book more by using your hands than using your ears.

Human beings are characterised by our ability to think and express ourselves through language. Reading and writing, together with speaking and listening, are the ways we express ourselves through language, ways which we should not and cannot give up. Reading with our fingers makes it possible for us to read at our own pace, with our own intonation, and allows us to appreciate the style and content of what we are reading, thereby creating that dual link with the written word that cannot be achieved by any other method. When we read with our ears, firstly we are deprived of knowing how something is written, and secondly, we lose a certain amount of the content, since the text has already been partly 'interpreted' by the person reading it out loud. And worse still if it's a machine reading the text (a technology already in an advanced stage of development); then we will have all the functionality we need, but as for pleasure...

In my case at least, if what I have to read is even slightly complicated, includes a large number of figures or requires a certain degree of analysis, memorisation or study, reading it in Braille allows me to perform at a much higher level than if I have to listen to it; listening only works for me when it's an easy and superficial text, because I find it very hard to avoid distractions, and if I'm just a little tired, I drop off into a delicious snooze, especially if the reader has a pleasant voice. I've asked a great many of my colleagues about this matter, and all of them who are at least minimally fluent in Braille agree with me.

And by the way, reading with your ears can come in handy when you have a big pile of ironing to do; there's a time and a place for everything!

3 Technology at The Service of The Visually Impaired Readers

3.1 How Can Technology Help Me when I'm Reading?

To a greater or lesser extent, technology is already present in the day to day life of every citizen, including those with impaired vision, and has an influence, not always, and not necessarily a positive one, but a considerable influence nonetheless, on everything we do. I will leave it to the sociologists to assess and analyse this general influence: in this article I will limit myself to analysing where we are and where we want to go with regard to texts and access to texts. And the first thing we need to understand is that this issue cannot be dealt with in isolation from the general context.

These days we make use of technology for all reading associated activities:

- In order to produce reading material
- In order to produce alternative reading mechanisms to replace paper or audio cassettes
- In order to make a radical change to the way we read: browsing, querying...

Bibliographical production, whether in Braille or in print, is inextricably linked to technology, and blind users wishing to make use of the material produced by this technology will also need to evolve and adapt to new ways of reading. In addition to the normal methods involving paper or cassettes they will need to use standalone devices or, better still, PCs, even if this involves making an extra effort to acquire general computer skills and learn how to handle adaptations for blind people or visually impaired users (at user level, naturally, not at expert level; we'll leave that for the producers). Given that all published material is stored on a digital medium, if this medium were available to blind people then logically we would have potential access to everything that is ever published. But the issue is not so simple.

3.2 What Medium Should A Publication Be Stored On So That A Visually Impaired Reader Can Access It?

An analysis of the great variety of digital media around these days – text format, graphics format, Internet file (basically HTML), preprint layout... – is far beyond the scope of this article. The choice of medium is up to the producer of the material, since there are all kinds of manipulation tools capable of converting any input source –keyboard, scanner, voice – into the desired format, depending on various criteria, not the least of which is personal preference.

Let's leave that part of the problem to be solved by the tools which serve as an interface between the computer and the visually impaired person, and let us focus on the range of products that those interfaces allow us to use, bearing in mind that our choice of solution is also conditioned by which interface tool we use, which in turn is dependent on our operating system, while even the computer itself (type, make) can have an influence. In other words, we need be careful not to make sweeping generalisations but rather speak in terms of probabilities. At the moment, and I stress 'at the moment', because of the breakneck speed of development of software and even the very operating systems themselves, we can access information in Microsoft Word, Wordpad, Notepad, PDF, TFL, Daisy, HTML and others.

3.3 What Aids Does A Visually Impaired Person Have Access To?

Depending on a number of factors, such as residual vision, degree of computer literacy, disposable income, training, personal preferences, users will try to read in whatever way they find easiest or most pleasurable, and there are a number of different kinds of interfaces to choose from:

- Computer installed screen magnifier.
- Computer installed voice synthesized screen reader.
- Computer connected Braille display.
- Braille printer for producing paper books.
- Standalone voice device.
- Standalone Braille device.
- Computer installed voice operated Internet browser.

Each type of interface has its pros and cons which I will not be going into either exhaustively or systematically. I will, however, be mentioning them all briefly with a relevant example, keeping the focus always on reading as my main purpose.

At the risk of stating the obvious, let me say that no one interface is better than another; each reader should be able to use whichever interface is best suited to his or her particular circumstances.

Nowadays the range of available products covers every possible facet of the problem. However, it is clear from the outset that Braille is losing the battle to voice. While users of voice screen readers can, in the main, make use ordinary technology applied to that purpose, any solution using Braille requires at least some specialized technology, which inevitably pushes up the final cost. Since all, or nearly all, Braille users can also hear, voice based solutions tend to come out on top. If we look just at standalone devices for blind people, voice solutions far outstrip Braille based ones, both in quantity and in price. And there are a hundred talking Braille devices in circulation for every device with a Braille output. However, neither the number of users nor the cost should be the priority factor with regard to the research and marketing of these kinds of aids, or the transcription of books into Braille. Instead we need to create mechanisms, funded by state subsidies, or subsidies from other suitable organisations, such as the ONCE (Spanish National Organisation for the Blind), to counteract this trend and thereby prevent the use of Braille from being limited to those who are both visually impaired and wealthy, circumstances which do not necessarily go hand in hand.

3.4 What Braille Displays Or Printers Are on The Market Nowadays?

Braille displays are devices which can be connected to a computer to enable users to read the text which appears on the screen in Braille. Their size will depend on the number of characters they can represent at any one time (20, 40, 70, 80). Their weight has been steadily coming down and they are not as weighty nowadays as when they first came out.

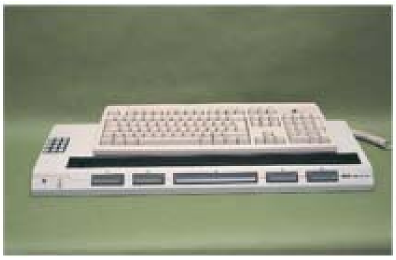


Figure 4: Ecoplus 80 Braille Display.

They do not normally have batteries, although there are some on the market which come equipped with them. 70 or 80 character devices could be said to be transportable, though not really portable. Each cell is a separate element within the display. They are built using highly resistant ceramic components which are mainly what makes these devices so expensive.

20 character displays tend to be used on standalone devices such as notebooks, organisers. For computer use the larger models are recommended; these currently tend to have 70 characters rather than 80, because that 12.5% reduction enables manufacturers to bring down the price, weight and size, and for the Windows environment it is considered to be sufficient. In Spain, the supplier of Braille displays is ONCE, which assembles and distributes them exclusively for the Spanish market. The model they are currently supplying is the 80 character Eco Plus shown here in Figure 4 (smaller models were ruled out due to their poor price/quality ratio). They are shortly to bring out the 70 character Satellite display, selling for around 4,500 euros.

The first Braille displays to come on the market were connected to the computer by hardware (i.e. by a card). Nowadays the design has changed and they are now managed by software, and are under the control of the screen reader; in other words, it's the screen reader that controls the device. This means that in order to use Braille you have to begin by operating a screen reader which provides voice options, with the result that the step of 'adding' Braille will often be skipped. It also requires the existence of a driver to enable the screen reader to recognise the display; in other words, users will be limited in their choice of Braille displays to those that their chosen screen reader can recognise.

The ONCE has already distributed among its members, either for purchase (the least common option), or as a workstation adaptation, or as a study station adaptation, some 1,500 screen readers with Jaws software from the company Freedom Scientific <http:// www. freedomscientific.com>. Some 1,100 Braille displays are supported by this chosen screen reader package.

With regard to printers, ONCE supplies a personal model, the *PortaThiel*, which prints 'interpoint' (doublesided) Braille at 14 characters per second, at a cost of around 1,500 to 2,000 euros, depending on the customer (in this case not exclusively in the Spanish market). Plus a couple of professional models: the *Impacto Texto*, which can print 250 characters per second (i.e. 800 pages an hour) without a graphics connector, and the Impacto 600 which is made to order and is somewhat slower but does have a graphics connector. For either device the price ranges from between 13,000 and 14,000 euros. There is currently nothing available in the mid-price range, although older models of Thiel printers cover this need.

Something which never ceases to amaze is that, after all the years Windows has dominated the market, drivers for printers under Windows have only become available relatively recently. ONCE's bibliographic production centres are still working under MS-DOS, although they are now finally planning a switch to Windows.

3.5 How Do The New Forms of Reading Benefit A Visually Impaired Person?

Braille books have been, are, and will always be big and beautiful, and so it continues to be a practical impossibility to keep a library of Braille books at home. However, if the storage medium is a CD, how many books could we keep? A whole shedload, if you'll pardon the expression. And if we can read that CD by using device X which will also provide us with a Braille version, we will have discovered the first huge advantage. If, on top of that, some of those CDs contain a dictionary or an encyclopaedia, we will have another incredible advantage: that of putting reference books within the reach of most blind people. This is a substantial change which will benefit a great many groups of people, especially students.

Another fundamental improvement is the possibility of accessing the content of a book wherever you want – at a particular chapter or at the index – and, in general, move around the book with an ease which is simply not possible with an audio cassette or a printed book. In short, thanks to technology we will be able to, and in fact we already can, read more and better. And any effort we need to make to adapt to a more complicated way of managing reading material will be more than repaid.

4 Digital Libraries

4.1 Where Can Digitalized Books Be Found (Including on the Internet)?

I am sure there is no topic in the world about which there is no information or documentation available on the Internet, so I am equally sure that there are books on the Internet. And there are. There are several web pages where books are to be found, responding to different criteria, in different formats, etc. etc., but books nonetheless, and a fair number of them at that. But not only on the Internet. They are to be found in other forms too.

I will tell what I know as a user interested in reading and technology, but before I begin, I would like to make a plea for an exhaustive and organised study of what there is, so we can all benefit from the efforts being made by various groups in several different directions with the aim of creating digital libraries, publicizing there existence, and making a wide ranging and comprehensive bibliographic collection available to blind people everywhere.

I view positively all the efforts made by the ONCE and other institutions who are addressing this task so that we blind people can read, but it is no secret that there is an overwhelming disproportion between the number of books published and the number that are available to blind people, and that there is normally a long wait between wanting to read a book and actually managing to read it, especially if the book has only just been published. Or that there is an enormous lack of dictionaries and encyclopaedias or reference books in general.

4.1.1 A Few References of Spanish Literary Works Available Online

- The Cervantes Virtual Centre (Cervantes Institute) includes the following works in its collection *"Clásicos hispánicos"*, <http://cvc.cervantes.es/obref/clasicos/>:
- Anon: *Historia de Enrique, fi de Oliva*. Critical edition by José Manuel Fradejas Rueda. Text and edition based on the first edition of 1498.
- Gustavo Adolfo Bécquer: *Rimas*. Annotated critical edition by Luis Caparrós Esperante.
- Miguel de Cervantes: *Don Quijote de la Mancha*. Critical edition with commentary by Francisco Rico, published by the Cervantes Institute.
- Lope de Vega: *El perro del hortelano*. Edited by Rosa Navarro Durán.
- John Minsheu: *Diálogos*. Published by the Cervantes Institute under the direction of Jesús Antonio Gil. A bilingual book, originally, which was used to teach Spanish in Tudor England. The first edition was in 1599.

The Cervantes Virtual Centre has also published online an anthology of texts as recommended reading intended for students of Spanish as a foreign language. The section is called *Lecturas paso a paso* and can be found at <http://cvc. cervantes.es/aula/ readings/>.

There are other digital libraries in Spanish on the Web, the best known of which is the Miguel de Cervantes Virtual Library, <http://www.cervantesvirtual.com/>, with its infant and youth section, <http://www. cervantesvirtual.com/portal/platero/>. We should also mention Libros en Red, <http://librosenred.com>, and El Alpeh.com, <http://www. elaleph.com/>.

4.2 Will I Be Able To Read, Legally, Everything I Want to?

An important issue we still need to resolve, at least here in Spain, is the matter of copyright. First we need to set up the necessary legal channels and, at the same time, reach agreements with the publishing houses, and with the authors themselves if need be, to find a solution so that whenever a book is published, it is transmitted by the agreed channel to a digital library accessible to blind people, where the appropriate conversions or adaptations can be made, in order to guarantee both its protection and its accessibility. As I understand it, this is absolutely viable today with the technological resources we already have, although it may not necessarily be a simple task.

4.3 Are There Already Digital Libraries For Visually Impaired People?

Although the degree of implementation varies from country to country, the answer to the question is yes. In 1996, in response to a clear need, the DAISY Consortium <http://www.daisy.org> was set up with a very specific purpose: to define an international standard for the production, exchange and use of the new generation of digital talking books.

As a result, the standard known as DAISY, an acronym of Digital Audio based Information System, was created. This standard complies with specifications for textual information and its structure set out in the standards published by the World Wide Web Consortium, <http//www.w3c.org>. The consortium is a non-profit organization working with third party companies, which over the years has been developing the tools required for the production and distribution of digital talking books, from workstations and tools for converting old analogical talking books, to the new playing systems that will be needed in the future.

Within the consortium there are two kinds of members, full members (like ONCE), and associate members. The difference between the two types of members lies basically in the rights to use the software developed through the consortium. For full members there is no limit to the number of licences they can use.

Alongside this software, some countries have also developed related programmes, which, as is the case of the Swedish state library TPB (Library of Talking Books and Braille, <http://www.tpb.se/english/index. htm>) or the Japonese JSRPD (Japanese Society for Rehabilitation of Persons with Disabilities, <http://www.jsrpd.jp/index_e. html>), are distributed to consortium members at no cost. These include programmes for playing digital talking books on a computer, such as the Player 2000 from TPB, or for the production of talking books, like the Japanese Sigtuna.

Other countries are developing tools to cover all aspects of talking book production, such as the special editor created by the Danish Library for the Blind, DBB (*Danmarks Blindebibliotek*, <http://www.dbb.dk/ English/>). In some cases these tools are free and in others you have to pay – it varies from country to country – though it should be noted that the abovementioned programmes are usually in the original language and in some cases in English. One important aspect of this project is the possibility of using voice synthesisers to 'record' the books, which is a great technological aid to reducing production costs.

Based on this format, the library Bookshare, in the United States <http://www. bookshare.org/web/Welcome.html>, is already up and running and providing a free service for US citizens, while, the Japanese and the Swedes are also already in full production.

In Spain, the ONCE already has more than 2,500 titles prepared. Currently all their recordings are created in this format and before too long we can expect the trial period to be deemed completed, whereupon this type of talking book should be made available to all the organisation's members.

There are several types of talking book players on the market, such as those produced by the Canadian company Visuaide, <http://www.visuaide.com/>, or the Japanese Plextor. The Víctor, from Visuaide, is a desktop reading device which is very user friendly, with large and well separated keys making for easy operation – I see it as the modern version of the traditional cassette player.

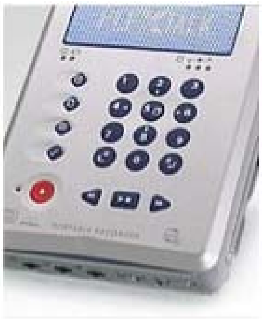
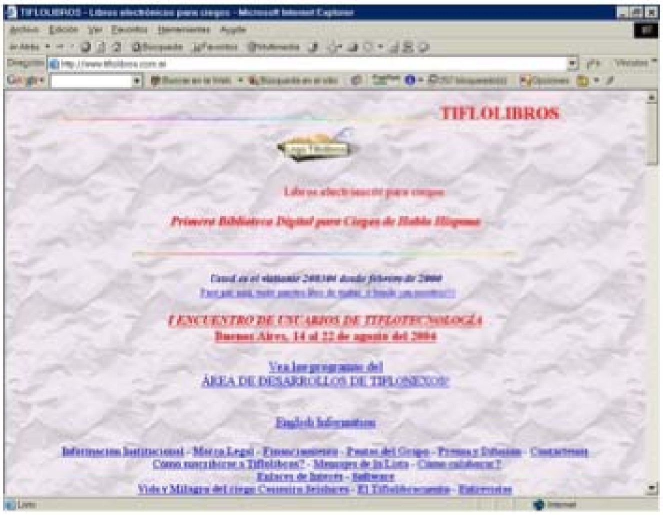


Figure 5: Plextalk Reader-Recorder, from Plextor.



TIFLOLIBROS

Libros electrónicos para ciegos

Primera Biblioteca Digital para Ciegos de Habla Hispana

[illegible]

I ENCUENTRO DE USUARIOS DE TIFLOTECNOLOGÍA
Buenos Aires, 14 al 22 de agosto del 2004

Vea los programas del
ÁREA DE DESARROLLOS DE TIFLONEXOS

English Information

[illegible]

Figure 6: Argentinean Portal Tiflolibros, <http://www.tiflolibros.com.ar>.

Plextalk, from Plextor, <http://www.plextalk.com/europe/>, is a reader-recorder which can be used both as a player and as a creator of DAISY books (figure 5). It has been equipped with more functionalities and it is rather more complicated than the previous one if you want to make use of all its capabilities. It can be connected to and operated via a computer. Braille, however, is the great absentee in this device. We will have to wait.

4.4 Are There Any More Libraries for The Blind?

Our Argentinean colleagues have set up an interesting experiment, which, to the battle cry of "*let's scan and read, we are all brothers*" have created a library called **Tiflolibros** which has just turned four years old and already has more than 9,000 titles. This library can be found at <http://www.tiflolibros.com.ar> (Figure 6). They are doing a magnificent job since, as well as sharing experiences and pooling the expertise of a large number of people, they have created some proprietary software for the protection of books which should not be accessible to everyone. These books are created in a format known as.TFL and require special software to read them. This software is provided free of charge to the library's readers, and to register as a reader you have to provide documentary proof of your disability, thereby guarding against copyright infringements by unauthorised users. They have already successfully signed an agreement with at least one publishing house. And their effort is all the more laudable given the fact that they are working on a volunteer basis. The books in this library are generated in various formats: Word, MS-DOS, HTML, Braille, Adobe PDF, as well as the special format we mentioned earlier,.TFL. They can only be read via a computer so the way they are read will depend on the interface the user has, i.e. Braille, voice, or character enlargement. They currently have 900 readers spread all over the Spanish speaking world as well as in countries where there are students of Spanish.

Last November, in Colombia, ahead of 140 other projects, Tiflolibros won the Betinho Communications Prize recognising people-centred technology initiatives in Latin America and the Caribbean, awarded by the Association for Progressive Communications.

4.5 Are Only We Blind People Interested in Digital Books?

By no means. If that were the case, there wouldn't already be so much bibliographic material on the Web. However, I would say that this is a culture which is changing, albeit very slowly. While bibliophiles continue to prefer printed books, with their characteristic smell and feel, the use of digital formats for dictionaries, encyclopaedias and reference books in general is becoming ever more commonplace.

At this point in time, the only digital booksellers that I have ever heard of has unfortunately had to close down. The publishing house RD Textos was created with the dual purpose of creating a virtual bookshop, i.e. as a business, and providing a service to blind or visually impaired people by selling books at modest prices in both HTML and in Braille. This publishing house, set up with every legal angle covered, including copyright, failed due to a lack of turnover. The bibliographic collections that they had prepared, which are no longer under copyright, have been donated to the Manuel Caragol Foundation, <http://www.funcaragol.org/>, which has published them on the Web, from where they can now be downloaded free of charge. I am really sorry that they failed because I believe they were doing a good job.

4.6 Are There Any New Projects Underway ('Hybrid' Books)?

Some time ago now, a new idea emerged from Hungary which is now being developed by various projects. The idea is to implement a system using what we might call 'hybrid' or mixed technology in such a way that any book created by that technology could be read either by voice or by Braille. If this project comes to fruition, we would no longer have to choose between voice and Braille when adapting a book for blind readers.

These initiatives originally received funding from the European Copernicus project. Later, in various stages, the country itself financed the development of a special player and the production of several books. Later still, from France, the project Culture 2000 funded the development of a tactile component, the 3T-book, <http://www.Braillenet.org/Tbook/traduc.htm>. Now, once again with local funding, they are currently working on adapting the browser to an electronic organizer type device. Among the benefits expected from this hybrid technology are:

- The user can make searches for sequences recorded in human voice.
- The user can read texts recorded in human voice on a Braille display.
- It is already prepared for use with various browsers.
- Tools are already available for creating books in this format in Hungarian, French and Italian.
- Blind people can use digital book creation tools in this format.

There is already a prototype of the reader with a Braille keyboard, CD-ROM, and several CDs in the three available languages (French, Italian and Hungarian). In my opinion this is a very interesting project, because being able to decide on the fly how you want to read will make it possible to get the best out of both worlds. Why force a decision if it can be avoided?

5 Conclusion: Braille as Pleasure And Culture

Can Braille be a source of enjoyment? I think the answer is clear throughout this article, but before I close, let me repeat that I wholeheartedly support Braille as provider of pleasure, leisure and enjoyment; reading is one of the treasures which I would not give up for the world. To touch a book, feel it under my fingers, caress it and immerse myself in all the wonderful literature that has been written. We blind people also deserve to savour the pleasure of reading José Hierro or José Saramago but, without Braille, it would be like looking at faded, discoloured painting. Now, with the aid of technology, the aim of reading any book right away should become true for blind people.

In the pursuit of standardisation, for the sake of the blind, let's read.

Translation by ***Steve Turpin***

Information Technology in Todays' Organizations

Is The IT Productivity Paradox Resolved?

Kyriakos E. Georgiou

© **Pliroforiki** 2004
This paper was first published, in English, by *Pliroforiki* (9th issue, September 2004, pp. 23-28). ***Pliroforiki***, ("Informatics" in Greek), a founding member of **UP**ENET (Upgrade European Network), is a journal published, in Greek or English, by the Cyprus CEPIS society CCS (Cyprus Computer Society, <http://www.ccs.org.cy/about/>)

This article addresses the issue of the Information Technology (IT) Productivity Paradox. The paradox was formed as a result of the apparent failure of substantial investments in IT to produce the desired results. The main school of thought in the USA uses econometric studies to measure the effect of IT on the performance of firms, sectors of the economy and the economy as a whole. This line of work comes under criticism from the European (read British) school of thought that consider it as too simplistic and one dimensional. Most recent research work suggest a highly positive relationship between investment in IT and organizational performance. The main emphasis of the research these days is to determine the business value of IT.

Keywords: Business Value of IT, Economics, Information Technology, Information Technology Paradox, Productivity, Production Functions.

Introduction

The issue of the effects, if any, that Information Technology (IT) has on the productivity of organisations has been one of the most critical issues in the IT field [Blake, 1994], [Brancheau. et. al, 1996]. The topic has extended beyond the boundaries of academia and it has captured the attention of the press [Bowen, W. 1986, Magnet 1994], Business Week 1993], the mass electronic media as well as the policy makers all over the globe. The term "information technology paradox" has entered the lexicons and everyday conversations. Organisations, researchers, policy makers and the press have begun to question the benefits for organisations from their substantial investments in IT.

Productivity

Productivity is a measure of performance and the only measure of competitiveness. It can be defined as the relationship between the inputs applied and the outputs that result, that is to say, the ratio of outputs to inputs. Outputs can be any combination of goods, products, or services whilst inputs can be human or material resources transformed in the process. Productivity is directly related to efficiency and effectiveness.

Efficiency refers to the optimal utilization of existing resources. In economics efficiency is the ratio of what an organization actually produces and what it could optimally produce with its existing resources, knowledge, and ability. Effectiveness refers at a minimum with the achievement of the goals of the Organization. A more proper definition attributed to Peter F. Drucker (1995) is the ability to expand the limits of the organization in terms of the opportunities to produce revenues, to create markets and to change the economic characteristics of existing products and markets. In the long term effectiveness is much more important for sustained productivity.

Traditional measures of productivity which has its roots in industrial engineering and agriculture do not address properly intangible factors such as product and service quality, variety and reliability and customer satisfaction. These intangible factors are key elements of competitive success in the strategy and marketing literature.

The Theoretical Background

Virtually all the studies before the late 1980's do not show a significant positive impact of IT on the productivity of organisations. This phenomenon led Dr. Robert Solow of MIT back in 1987 to comment that computers can be seen *"everywhere except in the productivity statistics"* (Solow). More recently there was a shift in this thinking since most recent studies show a positive relationship between investment in IT and the productivity of the firm.

Researchers in the field among others, include Erik Brynjolfsoon of MIT and his associates at the MIT Centre of e-Business (<http://ebusiness.mit.edu/>) and Lorin Hitt of the University of Pennsylvania, Wharton School of Business, associated with the MIT team. In New York Yanis Bakos of the University of New York Stern School of Business and Frank Lichtenberg of Columbia University. In California Kenneth L. Kraemer, Vijay Gurbaxani, Nigel Melville and Ronald V. Ramirez associated with the Centre for Research on Information Technology and Organizations (CRITO) at the University of California at Irvine, <http://www.crito.uci.edu/2>. These researchers have conducted extensive research, utilizing econometric analysis of multifactor productivity, in the USA. This type of work, though, has come under some criticism from other researchers in the field most of them in the UK [Dan Remenyi & Frank Bannister, 1999], [L. P. Willcocks and S. Lester, 1996, 1997] and [Jean Noel Ezingeard 1998].

Recent research has shown that longitudinal three-to-five year firm level studies demonstrate better results than single year studies, or macroeconomic studies of sectors of the economy or the whole economy [Hitt, 1996], [Mooney 1996], [Shu 1998], [Brynjolfsson, E. and Hitt, L.M.1995a], [Brynjolfsson, E. and Hitt, L.M.,1995b], [Brynjolfsson, E. and Hitt, L.M.,1996]. It appears that there are at least four reasons for the positive results of more recent studies:

(a) The sophistication of the research has evolved substantially;
(b) IT has matured and is now a much more powerful and useful set of tools;

Kyriakos E. Georgiou is one of the editors of ***Pliroforiki***, the journal of the Cyprus Computer Society (CCS). He is a Senior Manager for Professional Services for NetU Consultants. He occasionally teaches management courses in local colleges. He has a Bachelors degree in Mechanical Engineering and a MBA, both from the University of Houston, Houston, Texas, USA. The efficient and effective use of IT in organizations is one of his main professional and academic interests. <Kyriakosg@netu.com.cy>

(c) Organisations as a whole have been more successful in using IT in the context of achieving business objectives; and

(d) In most cases successful IT implementation is coupled with organizational change. Organizational change programs include employee participation programs, Total Quality Programs and Process Reengineering.

The first studies on IT productivity originated back in the 1970's. However there is a general agreement among the research community that those early years are not really representative. In the 1980's and early 1990's when the term the "IT productivity paradox" was coined by Baily and Gordon (1988) most of the studies showed negative correlation between capital expenditures on IT and measured productivity gains. Over the years since then, the balance has shifted towards more positive results.

Most researchers, though, it seems that they have a more or less ideological position on either side of the argument and most of their work is in support of this a priori 'ideological' position. As Bakos (1995), states *"(we) put to rest once and for all the old idea that computers are not productive"*. Quinn and Baily (1994) add the point that *"(if) managers did not think IT improved performance, they would not have continued to invest so heavily"*.

It is fair to say that the empirical results that most academic researchers, in the USA, reach these days show positive returns on IT investment. The work of Bryjolfsson and Hitt (1993; 1994; 1996) and Lichtenberg (1995) among others are along these lines. On the other hand the work of Paul Strassman (1985; 1990; 1997) and Stephen Roach (1989a; 1889b; 1996) point in the opposite direction.

There are two fundamental research questions:

1. Does IT offer the potential to enhance productivity and firm performance?
2. If IT provides potential benefits how do organizations manage and use IT to enhance productivity and create value?

In general the field is influenced by research carried out in the USA and not enough attention is paid to research carried out in the United Kingdom. Established practitioners in the field there include Professors Arthur Money, Dan Remenyi and Twite, A. [Remenyi et al. 1993], Jean Noel Ezingeard (1998) associated with Henley College but also L.P. Willcocks and S. Lester (1996, 1997) associated with Oxford just to name a few.

Most British researchers are critical of the economic approach that American researchers prefer. Brynjolfsson (1993) for instance in order to reinforce the point that the economic production theory provides an excellent method for measuring the efficient application he writes *"the bottom line of any technology is not how it changes work, but whether it increases productivity.*" Remenyi and Bannister (1999) on the other hand quoting a similar point from the same article suggest that *"This narrow perspective is (perhaps deliberately) limited in its understanding of the nature of IT value"*. Along the same lines Ezingeard (1998) suggests that there is now a recognition that evaluation should be concerned with more than simple 'efficiency' metrics. Academic techniques and constructs have yet to be widely adopted by practitioners. The emphasis in research should be placed on the impact that IT has on the entire organisation.

In any case the following statements are practically universally accepted:

1. The findings of IT productivity impacts research are inconclusive. This is a major theme of all the research in the field and it is precisely the issue that gave rise to the term "Productivity Paradox".
2. Firm level research offers more meaningful results in terms of accuracy and reliability as opposed to economy, industry and sector levels research. The reason for this is that beyond the firm level the data becomes difficult to use because it contains items that cancel each other out.
3. Longitudinal designs are more appropriate rather than single year or cross- sectional ones. Longitudinal designs are more in line with the process nature of the IT implementation.
4. More recent research is more probable to show a positive correlation between investment in IT and business performance.

The Econometric Model

The development of reliable, consistent and meaningful techniques for assessing the relationship between investments in Information Technology (IT) and firm performance is a very crucial issue since it lies in the centre of the research in this field. The emphasis of the literature review is on *ex post* measurement of the impacts of IT investments on firm performance rather than *ex ante* evaluation of the potential impact of proposed IT systems.

The role of firms in the economy is to produce goods and services called in generic terms "outputs of production", utilising a number of inputs that can be grouped into five groups namely capital (K), labour (L), energy (E), materials (M) and purchased services (S). Using different quantities of each production input that best suit its needs a firm tries to compete in the market and produce the most optimum product mix. In this presentation one form of production input can be replaced by another. The most common substitution is capital and outsourcing services for Labour. In a more formal presentation as a production function (F) these relations can be presented as:

$$Y = F\ (K, L, E, M, S)\ (1)$$

In studies of the productivity of IT the two most important independent variables are Capital (K) and Labour (L) that have two components each. One component is associated with the use of IT identified by (1) and the other refers to non IT related forms of capital or labour identified by (0).

I believe that an extension of the theory would be to examine how purchased services (outsourced services) affect the productivity of firms. This is important since outsourcing has become an integral part of the application of IT. Jablonski (1995) uses a production function that includes capital, labour and intermediate purchases to measure multifactor productivity in the U.S. textiles sector. Also Brynjolfsson and Hitt (2000) in a MIT Working Paper are extending their previous work and include Research and Development (R&D) as the third independent variable. On the other hand energy (E) and Materials (M) are of lesser significance especially when it comes to firms in the service sector of the economy. In this case the production function should include purchased services (S) and it should look like this:

$$Y = F\ (K0, K1, L0, L1, S0, S1)\ \ (2)$$

The Cobb–Douglas model is used nearly exclusively in previous IT Business Value research at the firm level [Bryjolfsson and Hitt 1996], [Gurbaxani, et al. 1998], [Lichtenberg 1995]. Empirical research has proved that this model is a reasonable model for estimating the returns to IT investment. A basic Cobb-Douglas production function has the following form:

$$Y = A\ K0^{\beta1}\ K1^{\beta2}\ L0^{\beta3}\ L1^{\beta4}\ S0^{\beta5}\ S1^{\beta6}\ \ (3)$$

The term A is the technical efficiency of a or Multifactor Productivity (MFP). MFP signifies contribution to output that is not accounted for by inputs to production. By taking logs the Cobb-Douglas model would take the form:

$$\mathrm{Log}(Y) = \alpha + \beta1\mathrm{logK0} + \beta2\mathrm{logK1} + \beta3\mathrm{logL0} + \beta4\mathrm{LogL1} + \beta5\mathrm{logS0} + \beta6\mathrm{logS1}\ \ (4)$$

In more complete form equation (4) has three control variables that account for time (t), industry (j) and the specific firm (i). By introducing these variables the production function takes the form:

$$\mathrm{Log}(Ytij) = \alpha_{tij} + \beta1\mathrm{log}\ K0_{tij} + \beta2\mathrm{logK1}_{tij} + \beta3\mathrm{logL0}_{tij} + \beta4\mathrm{LogL1}_{tij} + \beta5\mathrm{logS0}_{tij} + \beta6\mathrm{logS1}_{tij}\ \ (5)$$

The coefficients (βi) represent the output elasticity of input i and their estimation represents the contribution of IT investment to firm output.

The Cobb-Douglas production functions has two general assumptions:

1. The elasticity of substitution is assumed to be constant and unitary. This implies that an n percent change in the marginal rate of technical substitution will yield a n percent change in input mix.
2. It exhibits constant returns to scale that is if all inputs increase by a factor n then output increases by factor n.

And $\beta1+ \beta2 + \beta3 + \beta4 + \beta5+ \beta6 = 1 \Rightarrow$

$$\sum \beta i = 1 \quad (6)$$

Reading through the literature review a reader can truly appreciate the methodological and theoretical issues that form the core of the research. This type of research is carried out using essentially secondary data and the researcher has to make a lot of assumptions in order to fit the data into the model. These assumptions and the way they operationalise their variables at the end of the day can have a significant effect on the result of the research. Also the sample size can be problematic as many of the studies are based on a limited sample size of even less than 50 observations. I believe that primary data from a respectable sample size of at least 100 organisations over a three to five year period interval within the context of a given economy will probably provide a better set of data for running this type of analysis.

Both Brynjolfsson-Hitt and Lichtenberg in their respective work are using sales revenues as their output variable of the production function. Other researchers such as Stewart (1991), Drucker (1995) and Strassmann (1997) and Ramirez (2003) suggest that EVA (Economic Value Added) is a more appropriate and objective measure of an organisation's productivity and performance and they also suggest that it is used as the output or depended variable. EVA is defined as the return on capital minus the Weighted Average Cost of Capital (WACC) multiplied by the capital outstanding at the beginning of the year (Steward, 1991).

The Hypotheses

The core research question is the relationship between investments is IT and firm level productivity. Based on the model developed in the previous section a number of hypotheses can be formulated to provide the desired answers. The hypotheses evolve around the marginal products, or the related parameter of the output elasticity of the independent variables.

By taking derivatives of equation (5) the output elasticity of input X becomes

$$Bi = d \log (Y) / d \log X =$$
$$(dY / dX) (X/Y) = MP (X/Y)$$

Where MP is the marginal product of variable X.

There are a number of hypotheses that one can draw based on these formulations but for the purpose of this exercise we concentrate on those that have proven more powerful and interesting [Money, 1996], [Lichtenberg 1995], [Bryjolfsson and Hitt, 1993]

Hypothesis 1. The marginal product of IT capital (K_1) is positive.

More formally we test the hypothesis

$$H_0 : \beta_2 \le 0$$

Against the alternative hypothesis

$$H_1 : \beta_2 > 0$$

Hypothesis 2. The marginal product of IT capital (K_1) is significantly higher than the marginal product of non-IT capital (K_0) relative to their rental price.

$$MP_1 / MP_0 \le R_1/R_0$$

More formally we test the hypothesis

$$H_0 : \beta_2 – (R_1 K_1 / R_0 K_0) \beta_1 \le 0$$

Against the alternative hypothesis

$$H_1 : \beta_2 – (R_1 K_1 / R_0 K_0) \beta_1 > 0$$

Hypothesis 3. The marginal product of IT labour (L_1) is positive.

More formally we test the hypothesis

$$H_0 : \beta_4 \le 0$$

Against the alternative hypothesis

$$H_1 : \beta_4 > 0$$

Hypothesis 4. The marginal product of IT Labour (L_1) is significantly higher than the marginal product of non-IT labour (L_0) relative to their respective wage rates.

More formally we test the hypothesis

$$H_0 : \beta_4 – (W_1 L_1 / W_0 L_0) \beta_3 \le 0$$

Against the alternative hypothesis

$$H_1 : \beta_4 – (W_1 L_1 / W_0 L_0) \beta_3 > 0$$

Hypothesis 5. The marginal product of IT Services (S_1) is positive.

More formally we test the hypothesis

$$H_0 : \beta_6 \le 0$$

Against the alternative hypothesis

$$H_1 : \beta_6 > 0$$

Hypothesis 6. The marginal product of IT Services (S_1) is significantly higher than the marginal product of non-IT Services (S_0) relative to their rental price.

More formally we test the hypothesis

$$H_0 : \beta_6 – (R_1 S_1 / R_0 S_0) \beta_5 \le 0$$

Against the alternative hypothesis

$$H_1 : \beta_6 – (R_1 S_1 / R_0 S_0) \beta_5 > 0$$

Conclusion

Like in any research the application of production theory to firm level studies has certain strengths and potential short comings, mainly the linearity of the function. Although the method is quite robust, reliable and established the search for the perfect productivity measure is still an elusive target, especially in a relatively new field such as IT. Intangible attributes such as service and product quality are hard to quantify and measure and there is always the argument that, unlike work in a production environment, measuring productivity in a knowledge work environment is not appropriate. All these factors add to the complexity of the issue but do not subtract from its importance.

The econometric approach is one of a "black box" where the researcher knows very little of what actually goes on within the "black box", that is, the firms in question. In the next instalment of the article we are going to explore some of the issues, strategic and behavioural, that take place within the "black box" and explore the business value of IT.

References

[M. N. Baily and R. J. Gordon, 1988]
"The Productivity Slowdown, Measurement Issues and the Explosion of Computer Power", The Brookings Institution, Brookings Papers on Economic Activity: pp 347–431.

[J. Y. Bakos,1995]
"Are Computers Boosting Computers", Computerworld, March 27:128–130.

[Ives Blake, 1994]
"Editors Comments: Probing the Productivity Paradox", MIS Quarterly 18(2).

[W. Bowen, 1986]
"The Puny Payoff From Office Computers", Fortune, May 26.

[J. C. Brancheau, B. D. Janz, J. C. Wetherbe, 1996]
"Key Issues Information Systems Management: 1994–95 SIM Delphi Results", MIS Quarterly, June: 225–242.

[E. Brynjolfsson and L. M. Hitt, 1995]
"Productivity without Profit: three Measures of Information Technology's Value", 15th International Conference on Information Systems.

[E. Brynjolfsson and L. M. Hitt, 1995]
"Information Technology as a Factor of Production: The Role Differences among Firms", Economics of Innovation and New Technology 3(4): 183–200.

[E. Brynjolfsson and L. M. Hitt, 1996]
"Paradox Lost? Firm Level Evidence on the Returns to Information Technology Spending", Management Science 42(4): 541–558.

[E. Brynjolfsson and L. M. Hitt, 2000]
"Computing Productivity: Firm-Level Evidence", MIT Working Paper.

[Business Week 1993]
"The Technology Payoff: special report", Business Week, June 14 1993: 56–79.

[Peter F. Drucker, 1995]
"The Information Executives Truly Need", Harvard Business Review 73(5): 54–62.

[J. N. Ezingeard, 1998]
"Towards Performance Measurement Processes for Manufacturing Information Systems", 5th European Conference on the evaluation of Information Technology, Reading U.K.

[Lorin M. Hitt, 1996]
"Economic Analysis of Information Technology and Organization", Dissertation, Massachusetts Institute of Technology.

[V. Gurbaxani, N. Melville, and K. Kraemer 1998]
"Disaggregating the Return on Investment to IT Capital", ICIS Proceedings 1998.

[M. Jablonski 1995]
"Multifactor productivity: cotton and synthetic broadwoven fabrics", Monthly Labour Review, July 1995: 29– 38.

[F. R. Lichtemberg, 1995]
"The Output Contributions of Computer Equipment and Personal: A Firm- Level Analysis", Economics of Innovation and New Technology, 3 201–217.

[Myron Magnet, 1994]
"The Productivity Payoff Arrives", Fortune, June 27: 79–84

[John G. Mooney, 1996]
"The Productivity and Business Value of Information Technology Economic and Organizational Analysis", Dissertation, University of California, Irvine.

[J. B. Quinn and M. N. Baily, 1994]
"Information Technology Increasing Productivity In Services", Academy of Management Executive, 8,3: 28–48

[Ronald Vincent Ramirez, 2003]
"The Influence of Information Technology and Organizational Improvement Efforts of the Performance of Firms", PhD Dissertation, University of California at Irvine.

[Dan Remenyi, Frank Bannister, 1999]
"In Defence of Instinct: Value and IT Investment Decisions", Henley Working Paper HWP 9914.

[D, S. J. Remenyi, A. Money, and A. Twite, 1993]
A Guide to Measuring and Managing IT Benefits, 2nd Edition, NCC Blackwell Oxford England.

[Dan Remenyi, Michael Sherwood-Smith, and Terry White, 1997]
Achieving Maximum Value from Information Systems: A Process Approach, John Wiley and Sons Ltd. West Sussex England.

[Stephen S. Roach, 1989a]
"Pitfall of the 'New' Assembly Line: Can Services Learn From Manufacturing?", Morgan Stanley Special Economic Study, New York.

[Stephen S. Roach, 1989b]
"America's White Collar Productivity Dilemma", Manufacturing Engineering, (August): 104.

[Stephen S. Roach, 1996]
"The Hollow Ring Of The Productivity Revival", Harvard Business Review 74(6): 81– 89.

[Wesley Szu-Way Shu, 1998]
"From Different Angles to Solve the Puzzle: Macro-Economic and Micro-Economic Analyses of Information Technology Productivity", Dissertation, The University of Arizona.

[Robert Solow, 1987]
"We'd Better Watch Out" New York Times Book Review, 36.

[Ill Stewart, G. Bennett 1991]
The Quest for Value, Harper Collins Publisher. New York

[Paul A. Strassmann, 1985]
Information Payoff: The Transformation of Work in the Electronic Age. Free Press, New York, NY.

[P. A. Strassmann, 1990]
The Business Value of Computers. The Information Economics Press, New Canaan, Connecticut.

[P. A. Strassmann, 1997]
The Squandered Computer: Evaluating the Business Alignment of Information Technologies, Information Economics Press, New Canaan, Connecticut.

[P. A. Strassmann, 2004]
"Defining and Measuring Information Productivity" Information Economics Press, New Canaan, Connecticut.

[L. P. Willcocks and S. Lester 1996]
"Beyond the IT Productivity Paradox", European Management Journal 14(3): 279–290.

[L. P. Willcocks and S. Lester 1997]
"In search of Information Technology productivity: Assessment issues", Journal of The Operational Research Society 48(11): 1082–1094.