

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

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

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

#### Publisher

UPGRADE is published on behalf of CEPIs (Council of European Professional Informatics Societies, <<http://www.cepis.org/>>) by Novática

<<http://www.ati.es/novatica/>>, journal of the Spanish CEPIs society ATI (Asociación de Técnicos de Informática <<http://www.ati.es/>>).

UPGRADE is also published in Spanish (full issue printed, some articles online) by Novática, and in Italian (abstracts and some articles online) by the Italian CEPIs society ALSI

<<http://www.alsi.it/>> and the Italian IT portal Tecnoteca <<http://www.tecnoteca.it/>>.

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

#### Editorial Team

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

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

#### Editorial Board

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

#### UPENET Advisory Board

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

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

Cover page designed by Antonio Crespo Foix, © ATI 2004

Layout: Pascale Schürmann

Editorial correspondence: see "Editorial Team" above

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

Upgrade Newsletter available at

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

#### Copyright

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

© CEPIs 2004 (for the sections MOSAIC and UPENET)

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

The opinions expressed by the authors are their exclusive responsibility.

ISSN 1684-5285

Next issue (December 2004):  
"Cryptography"

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

- 2 Editorial  
Four Years of UPGRADE

## SPT, Software Process Technology

Guest Editors: Francisco Ruiz-González and Gerardo Canfora

### Joint monograph with Novática\*

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

## MOSAIC

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

## UPENET (UPGRADE European NETWORK)

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

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

# 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 provide 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 convener 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](mailto: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 convener 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](mailto: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 require-

ments, 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

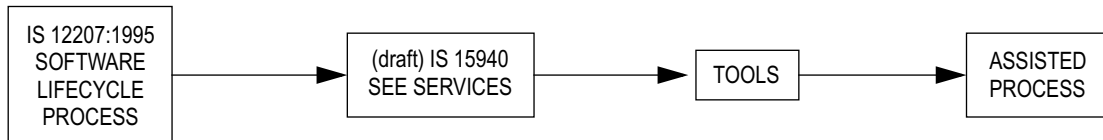
Support services include services that the rest of the services will require in order to become operational. They generally include services associated with processing, formatting, and disseminating human-readable data. This section describes the following services:

- 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.