A Maturity Environment to Software Engineering Education

Kechi Hirama[1], Selma Shin Shimizu Melnikoff[2]

[1,2]Dept. of Computer Engineering, Escola Politécnica of University of São Paulo, São Paulo, Brazil,

kechi.hirama@poli.usp.br[1], selma.melnikoff@poli.usp.br[2]

Abstract

A maturity environment is presented, aiming to support software engineering education. This environment is part of a processes framework being developed in the Software Technology Laboratory (LTS) of the Department of Computer Engineering of Escola Politécnica of University of São Paulo, Brazil. The framework is a three-level model with Reference Model, Maturity Environment and Real Environment. The framework is based on ISO/IEC 12207 standard and on the teaching, researching and extension activities of LTS members. The Reference Model defines a meta-model that guides the processes definition to Maturity Environment. The processes are Management, Development and Software Quality Assurance. The Maturity Environment is a generic model that details Reference Model processes. Further, this environment can be instantiated in an organization that constitutes a Real Environment. This last level could be a classroom, a software didactical laboratory or a software factory organization. The framework will be improved based on continuous improvement strategy, such as PDCA cycle. The benefits of the Maturity Environment are applying different teaching approaches in undergraduate, graduate and extension courses, developing projects and researches in an integrated environment, evaluating the performance of disciplines and improving communications among students, professors and researchers in the LTS.

Introduction

he Department of Computer Engineering of Escola Politécnica of University of São Paulo (Brazil) is constituted by 11 research laboratories, among which the Software Technology Laboratory (LTS). The LTS was launched in 1999 by a group of researchers interested in the Software Engineering area. Nowadays, the LTS has 9 researchers and its members also conduct activities in undergraduate, graduate and extension courses. [5, 6]

In 2004, a restructuring work of LTS activities was started to improve the interactions among researchers in the laboratory. After many issues were detected and solved, the LTS had a new structure to guide all its members´ expectations. In this context, the LTS aims at the development of human resources, integrated research programs, and technology transfer to be a reference center in Software Engineering.

The LTS is interested in the following areas [5, 6]

- Software architectures: visions, frameworks, components, patterns, distributed objects, reference models;
- Databases: object-oriented and distributed databases, data modeling, knowledge databases, data warehouses, data mining;
- Requirements engineering: elicitation techniques, analysis, validation, requirement management techniques and processes;
- Software project management: models, estimation techniques, risk analysis, project performance analysis, personnel management, acquisition and supplying;
- Human computer interface: usability engineering, human reliability, groupware interfaces;
- Formal methods: formal requirements specification, formal verification and validation, model checking;
- Software processes: life cycle models, component-based development, software factory, unified process;
- Software quality: standards and models, quality management system, metrics, assessments, evaluations;
- Software reuse: processes and techniques, patterns, components, metrics;
- Object orientation: modeling, development processes, architectures.

After the discussions of its objectives, a project named Maturity Environment was started in 2004 with the LTS members. The Maturity Environment should define the framework of the main LTS processes based on knowledge areas. This framework should provide a better communication among professors, researchers and research groups in common subjects, improve the resource sharing and update students' education in Software Engineering.

Maturity Environment is one level of the LTS framework. [3, 5, 6]

The LTS Framework

The LTS framework (presented in Figure 1) is based on ISO/IEC 12207 standard [8] and teaching, research and extension activities of LTS members. It intends to provide an intuitive structure according to academic results developed by researchers, professors and students of the Software Engineering course. It is organized in a three-level model with Reference Model (conceptual level), Maturity Environment (logical level) and Real Environment (physical level). [5, 6]

The conceptual level is constituted by processes based on the main activities developed by LTS members. It was based on the analysis of many materials developed in disciplines, dissertations, thesis, papers and professional activities. After that, the Management, Development and Quality Assurance processes were chosen as the first configuration of the Reference Model. The Reference Model aims to guide the LTS to identify new research areas, integrating efforts, resources and promoting the interactions among its members. This level has been constantly fed according to new results. Thus, other processes of ISO/IEC 12207 standard could be included in the Reference Model.

The logical level represents the detailed processes identified at a conceptual level. At this level, the Management, Development and Quality Assurance processes were described and their activities, inputs, outputs, controls and roles were defined. The process components are also based on the ISO/IEC 12207 standard.

The physical level represents the instances defined from logical level processes. These instances can be created in undergraduate, graduate and extension courses and activities. Each instance can be defined according to organization, for example, in laboratories, classrooms or a software factory environment. At this level, an instance can use the concepts, definitions, standards and models defined by the Maturity Environment or, in some cases, it can define specific activities to be more realistic in its needs.

In the students' point of view, the instances allow conducting experiments, developing projects and participating in researches to complement their learning. In the professors' and researchers' point of view, the instances allow researching and developing new approaches to disciplines in an integrated framework. In the organization point of view, the instances allow implementing new technologies or methods that will improve the productivity and the quality of products. Besides, the instances can be used to train personnel in specific activities to disseminate new practices.

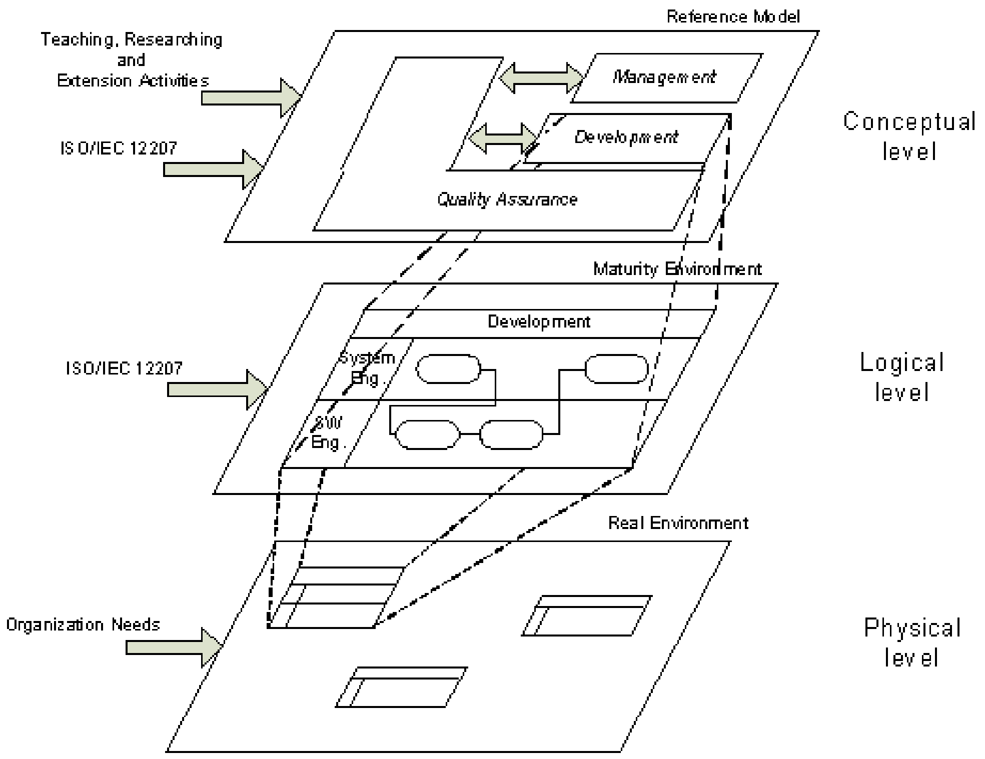
The Maturity Environment

The Maturity Environment is a set of organization processes that can be instantiated to develop and implement processes in specific projects, software factories, classrooms and didactical laboratories. In this way, each LTS member has a uniform vision of activities and can improve his/her concepts in Software Engineering. [11, 12]

The Maturity Environment aims to reduce the distance between Software Engineering and organizations practices besides integrating the LTS activities to this environment.

Nowadays, the regular updating of technology courses is necessary to be in a state of art position in new academic approaches and tools launched by different suppliers, besides the complex software applications that embed many concepts; these should be understood to enhance productivity and quality in organizations.

Figure 1: LTS framework. [5, 6]



In this scenario, the Maturity Environment is the result of knowledge alignment that was built along researching, teaching and extension activities developed over time at LTS.

The expectation concerning this environment is to develop it gradually so as to transfer the knowledge and the practices to the students, from their first year in the Computer Engineering course to the conclusion. The student could develop his/her capabilities from basic concepts of computing in the beginning to complete projects in the last year. Furthermore, this approach allows teaching processes to be evaluated during the courses and to propose indicators to improve them.

Further on, the student can start academic researches choosing themes among the many possibilities indicated by the Maturity Environment. To the students, there will be the possibility to continue the researches in graduation programs at the Department of Computer Engineering (PCS).

One of the challenges of the Maturity Environment is defining, updating and maintaining the necessary consistency and relevance of disciplines in undergraduate, graduate and extension courses offered by PCS. One of the important aspects of the Maturity Environment is allowing monitoring the technology evolution so that students can understand; after they have concluded their courses, they can be prepared to the daily tasks at organizations with productivity and quality.

Case Studies

In order to apply and evaluate the proposed Maturity Environment, three disciplines were chosen as instances, two undergraduate (PCS 2053 and PCS 2426) and one graduate one (PCS 5774) in the Computer Engineering course.

First of all, each discipline was evaluated considering its original contents proposed. The PCS 2053 discipline is focused on management and quality assurance, whereas PCS 2426 focuses on fundamentals of Software Engineering and PCS 5774 on software quality. After tailoring the Maturity Environment processes in the discipline contents, they were instantiated in three classes, each with about 20 students for two semesters last year.

The Management process activities, according to the ISO/IEC 12207 standard, are initiation and scope definition, planning, execution and control, review and evaluation and closure. [8] All of processes related to ISO/IEC 12207 have been included in the Management process in Maturity Environment. In the instantiated management process (PCS 2053), only the closure activity was not considered because there is not enough time for it in regular classes. More than 10 projects were concluded. The teams did not conduct lessons learned meetings and this activity was deemed necessary to improve PCS 2053. The main point was creating a historical database of projects.

The Development process activities according to ISO/IEC 12207 standard are process implementation, system requirement analysis, system architectural design, software requirements analysis, software architectural design, software detailed design, software coding and testing, software integration, software qualification testing, system integration, system qualification testing, software installation and software acceptance support. [8] Only activities system integration, system qualification testing, software installation and acceptance support were not included in the Development process in the Maturity Environment. In the instantiated development process (PCS 2426), the activities related to software coding and testing were not considered because the system and software analysis and design were defined as a first approach. The students developed static and dynamic models based on customer needs specified by the professor. During the software design activities, they detected the artifacts that did not meet the users' requirements. The students concluded that a requirements traceability matrix tool or similar to support tracking requirements could improve PCS 2426. The main point was long wasted time to correct artifacts during software modeling and design.

Finally, the Quality Assurance process activities according to the ISO/IEC 12207 standard are process implementation, product assurance, process assurance, assurance of quality systems. [8] Only the assurance of quality systems activity was not included in the Quality Assurance process in the Maturity Environment. In the instantiated quality assurance process (PCS 5774), the product and process assurance activities were considered. Many works and papers about software quality were studied and discussed by students in seminars. In the end, the students considered that assurance of quality systems based on the ISO 9001 standard would be important to treat issues concerning standards and quality model compliances in PCS 5774. This was detected when the students had difficulties in applying best practices in projects and in independent assessment, such as quality assurance, and deemed it would be important.

Table 1 presents the most important points of the case studies.

Table 1: Most important points of the cases studies.

ISO/IEC 12207	Maturity Environment	Instances
Management	**Management**	**PCS 2053**
Initiation and scope definition, planning, execution and control, review and evaluation, and closure.	All the former, except closure activity.	Closure activity should be considered.
Development	**Development**	**PCS 2426**
Process implementation, system requirements analysis, system architectural design, software requirement analysis, software architectural design, software detailed design, software coding and testing, software integration, software qualification testing, system integration, system qualification testing, software installation and software acceptance support.	All the former, except system integration, system qualification testing, software installation and acceptance support activities.	Requirements traceability matrix tool or similar should be considered.
Quality Assurance	**Quality Assurance**	**PCS 5774**
Process implementation, product assurance, process assurance, assurance of quality systems	All the former, except assurance of quality systems.	Assurance of quality systems based on the ISO 9001 standard should be considered.

The instantiation of processes starting with Maturity Environment indicated to be adequate according to the results and conclusions reached by students. All methods, techniques, templates, activities and tasks applied in some disciplines should be first analyzed in the Reference Model context. Then, new processes of the ISO/IEC 12207 standard could be considered in a structured manner following the levels of LTS framework.

Consequently, these processes in Maturity Environment could be instantiated in different organizations.

Conclusions

The Maturity Environment improves the performance of Software Engineering education, following the best practices evolution proposed by researches and learned lessons, establishes a common terminology and patterns and allows transferring technology in extension activities. The continuous improvement cycles, such as PDCA [9], IDEAL [4], could improve the processes to enhance students' learning.

The adoption of the ISO/IEC 12207 standard as a reference to LTS framework has shown to be adequate, mainly at logical level, where Maturity Environment is defined. It should be emphasized that either processes structure or the Maturity Environment itself is robust to support the LTS mission. Besides, the application of the ISO/IEC 12207 standard in a different context of a software development organization has allowed knowing the standard and confirming the possibility to apply it in a research laboratory, such as the LTS.

The Maturity Environment also follows well known standards and models practices in software process organizations, such as ISO/IEC 9001 [10], CMMI [2] and MR-MPS [13, 14]. In this sense, another important point is that processes of this environment could be evaluated to obtain official maturity certifications, by applying the ISO/IEC 15504 standard [7], SCAMPI [1] or MA-MPS [14], to maintain and improve the processes.

The benefits of the Maturity Environment are applying different teaching approaches in undergraduate, graduate and extension courses, developing projects and researches in an integrated environment, evaluating performance of disciplines and improving communications among students, professors and researchers in the LTS.

References

01. AHERN, D. M.; ARMSTRONG, J.; CLOUSE, A.; FERGUSON, J. R.; HAYES, W.; NIDIFFER, K. E. (2005). CMMI SCAMPI Distilled – Appraisals for Process Improvement. Pearson Education Limited. 218p.

02. CHRISSIS, B.; KONRAD, M.; SAND, S. (2006). CMMI for Development – Guidelines for Process Integration and Product Improvement. 2nd. Edition. Addison-Wesley.

03. COSTA, H. A. X.; ALMEIDA, S. A.; CUNHA, G. J.; HIRAMA, K.; RESENDE, A. M. P. (2007). Artefatos de Software do Processo de Desenvolvimento da Norma ISO/IEC 12207. In: I Congresso de Ciência da Computação e Sistemas de Informação. Annals. Lavras – MG. Brazil.

04. GREMBA, J.; MYERS, C. (1997). The IDEAL(SM) Model: A Practical Guide for Improvement. v1.1, SEI. (http://www.sei.cmu.edu accessed in 04/23/2008).

05. HIRAMA. K. (2008). Um Ambiente de Maturidade para um Laboratório de Pesquisa na Área de Engenharia de Software. Livre Docência Thesis. Escola Politécnica of University of São Paulo. São Paulo. Brazil. 156p.

06. HIRAMA, K.; MELNIKOFF, S. S. S.; BECERRA, J. L. R. (2005). Projeto de Ambiente de Maturidade de um Laboratório de Pesquisa na Área de Engenharia de Software. In: XXXIII Congresso Brasileiro de Ensino de Engenharia. Annals. Campina Grande – PB. Brazil.

07. ISO. ISO/IEC 15504-1 Information Technology – Process Assessment – Part 1: Concepts and Vocabulary. ISO. 2004.

08. ISO. ISO/IEC 12207 (2008). System and Software Engineering – Software Life Cycle Processes. ISO.

09. KINULLA, A. (2001). Software Process Engineering Systems: Models and Industry Cases. Oulu University Press. 115p.

10. MUTAFELIJA, B.; STROMBERG, H. (2003). Systematic Process Improvement ISO 9001:2000 and CMMI. Artech House. 300p.

11. PRESSMAN, R. S. (2004). Software Engineering: A Practitioner's Approach. McGraw-Hill Higher Education. 880p.

12. SOMMERVILLE, I. (2007). Software Engineering. 8th edition. Pearson Education Limited. USA. 552p.

13. WEBER, K. C.; ROCHA, A. R.; ALVES, A.; AYALA, A. M.; GONÇALVES, A.; PARET, B.; SALVIANO, C.; MACHADO, C. F.; SCALET, D.; PETIT, D.; ARAÚJO E.; BARROSO, M. G.; OLIVEIRA, K.; OLIVERIA, L. C.; AMARAL, M. P.; CAMPELO, R. E. C.; MACIEL, T. (2004). Modelo de Referência para Melhoria de Processo de Software: Uma Abordagem Brasileira. In: XXX Conferencia Latinoamericana de Informatica. 27 de setembro a 01 de outubro. Arequipa – Peru.