Turkish Journal of Electrical Engineering and Computer Sciences

Volume 12 | Number 2 Article 2

1-1-2004

An Information System for Streamlining Software Development Process

SERKAN NALBANT

Turk J Elec Engin, VOL.12, NO.2 2004, © TÜBİTAK

An Information System for Streamlining Software Development Process

Serkan NALBANT
Software Process Engineer, MilSOFT Yazılım Teknolojileri A.Ş.
ODTÜ-Teknokent İkizleri 06531, Ankara-TURKEY
snalbant@milsoft.com.tr

Abstract

In this paper an information system to be employed by software development organizations is proposed, which automates software development process. The proposed system aims to lower cost, improve schedule performance and enhance quality of the software projects by the means of automation and unifying of operational information. The characteristics of the proposed system are described. Furthermore, its use is illustrated via the explanation of an exemplary software system called PACE that serves as an information system for planning, controlling, measuring and improving software development process and projects. The relationship of PACE with Software Capability Maturity Model (CMM) is also provided.

1. Introduction

Today, software is a basic component of many businesses. In some areas, it is even impossible to survive for an organization without the use of associated computer softwares (e.g. banking, telecommunications). Because of this fact along with the increasing competition, advances in technology and enhancing capabilities of software development organizations, the need for more and more sophisticated software systems is growing constantly. The realization of such systems requires successful completion of complex projects. Obviously, this can not be accomplished without the existence of effective software processes which underpins the software development activities.

Software Capability Maturity Model (CMM) of Software Engineering Institute (SEI) defines software process as; a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals) [1]. A number of software process models have been developed in the last two decades such as TickIT, ISO 9001, BOOTSTRAP, CMM, ISO/IEC 12207, ISO/IEC TR 15504 (SPICE), Cleanroom Reference Model [1-7]. In addition to these, OPEN Process Specification should also be mentioned here, due its object oriented structure. OPEN is a framework for third-generation object oriented software development methods providing strong support for process modeling by using lifecycle patterns, strong support for requirements capture, and offering the ability to model intelligent agents [8].

SEI's CMM deserves special attention among the above mentioned models, since it is probably the most widely used and recognized one by software organizations (although its extension to system level,

namely CMMI, is starting to replace CMM). CMM was developed to help both those organizations which develop software to improve their software processes and those organizations which acquire software to assess the quality of their contractors. To fulfill the former goal, CMM provides a guide for software organizations in selecting process improvement strategies by determining the current level of their process maturity and identifying the key factors that would lead to improvement. The underlying assumption, as in all process models, is better processes lead to improved quality in the product [9].

The software organizations should employ various software tools for completing their projects properly (in terms of budget, schedule and quality) according to a defined software process. The necessity of using tools for software development is increasing steadily due to cost& schedule pressures on software projects and increasing complexity of projects in terms of management and technical aspects. Actually, it is impossible to perform most of the tasks without the use of corresponding tools. As the use and importance of these tools is increasing, their integration becomes an issue under consideration. The integration of such tools enables the streamlining of individual tools by providing the sharing of data and methods among applications. There exists studies regarding the integration of these tools, although they are not in the desired level [10,11]. These studies focus on the achievement of colloborative working of tools with each other. However, the need for the integration which will collect and unify the high-level operational information in order to enable quantitative management (i.e. planning, execution, monitoring) of software projects, remains uncovered. In this paper, we describe a solution which enables the integration of the operational information among various activities performed during software development process such as cost estimation, project planning, quality audits etc. Actually, this solution is a model for the software development tools, which we call software development information systems.

Section 2 describes the tools used in software development and the issues regarding their integration. Then in section 3, the aims of and the needs for the proposed software development information system are outlined. Section 4 presents the main functions of the proposed software development information system. Software Development Process Definition and Control System (PACE) is introduced in section 5 which is the implementation of the proposed system. Finally, section 6 provides a brief discussion.

2. Literature on Tools for Software Development

As others, software development organizations (also) must employ softwares (tools) in order to produce the software, which delivers what customer needs without any defects, on time and on budget. The usage of tools for software engineering is necessary and important, since it is a difficult discipline (especially in cognitive terms). Tools reduce the effort required to produce a work product and/or accomplish project milestones, and, provide new ways of doing and designing things for software engineers and managers. Thus they enable making better decisions, achieving higher quality levels and completing projects with less cost and on-time.

Regarding the context of this paper, these tools are classified into two as: Computer-aided Software Engineering (CASE) Tools and Software Development Information Systems. Ghezzi, Jazayeri, Mandrioli defines CASE tools as the tools and environments aiming at automating individual activities that are involved in software engineering [9]. Whilst, Software Development Information Systems aim to manage and integrate various activities performed, while producing software, by

- sharing & unifying information,
- automating many aspects of the software organization,

- providing necessary planning, control and measurement mechanisms.

The scope of Software Development Information Systems is all activities carried out by a software development enterprise, whilst CASE tools try to optimize and/or automate the tasks performed in order complete single or several software development activities. Figure 1 illustrates the difference between CASE tools and Software Development Information Systems. It shows that separate CASE tools enable the effective and efficient operation of sub-processes such as Configuration Management, Coding, Testing etc., whilst Software Development Information Systems integrates these sub-processes by enabling the execution of interfacing portions them in the same environment and, thus sharing of the produced information for planning and monitoring purposes. In this paper, a model for the tools of second type will be introduced and explained. In order to do this, Software Development Process Definition and Control System (PACE)[1] of MilSOFT will be employed.

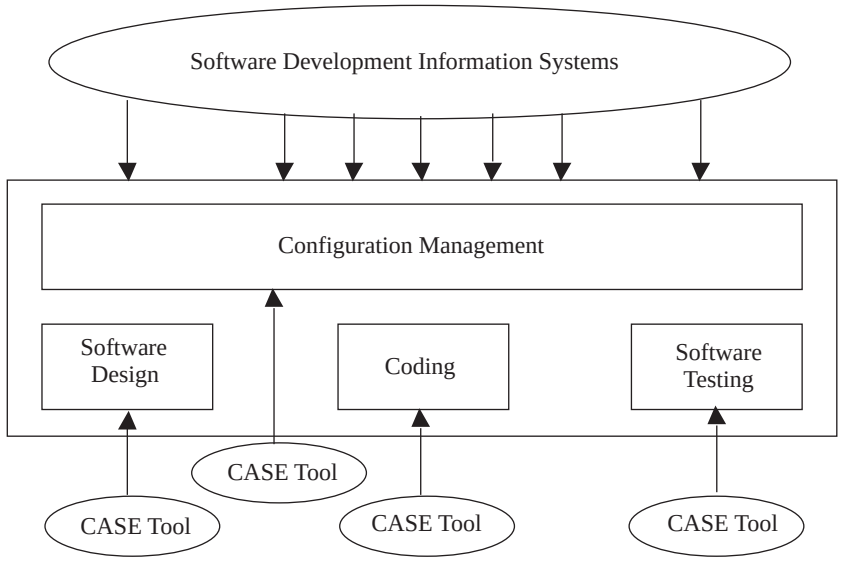

Figure 1. Interaction of tools with software activities.

The literature on software engineering tools provides many examples of CASE tools that are intending to assist software development process by allowing repetitive and well-defined actions to be automated, thus reducing cognitive and effort load on software engineers and project managers. SWEBOK of IEEE categorizes these tools into the following main and sub categories [12]:

- Software Requirements Tools (Requirements Modeling, Traceability)
- Software Design Tools
- Software Construction Tools (Program Editors, Compilers, Interpreters, Debuggers)
- Software Testing Tools (Test Generators, Test Execution Frameworks, Test Evaluation, Test Management, Performance Analysis)
- Software Maintenance Tools (Comprehension, Re-engineering)

[1]The implementation of PACE was sponsored by The Scientific & Technical Research Council of Turkey (TÜBİTAK) and Technology Development Foundation of Turkey (TTGV).

- Software Engineering Process Tools (Process Modeling, Process Management, Integrated CASE Environments, Process-centered Software Engineering Environments)
- Software Quality Tools (Inspection, Static Analysis)
- Software Configuration Management Tools (Defect & Problem Tracking, Version Management, Release & Build)
- Software Engineering Management Tools (Project Planning & Tracking, Risk Management Measurement)
- Infrastructure Support Tools (Interpersonal Communication, Information Retrieval, System Administrative & Support)
- Miscellaneous Tools Issues (Meta Tools, Tool Evaluation)

In order to get most value out of CASE tools they should be integrated. Pressman states that "Most CASE tools in use today have not been constructed using all building blocks of CASE tools which are integration framework, portability services, operating system, hardware platformn environmental architecture. In fact, some CASE tools remain 'point solutions'. That is, a tool is used to assist in a particular software engineering activity, but does not directly communicate with other tools, is not tied into a project database, is not part of an integrated CASE environment" [13]. Pressman defines four levels of CASE integration:

Individual Tool

Tool Bridges & Partnerships: The integration is improved slightly when individual tools provides faciulities for data exchange.

Single-source Integration: Occurs when a single CASE vendor integrates a number of different tools and sells them as a package. This precludes easy addition of tools from other vendors.

Integrated Project Support Environment: Standards for CASE building blocks are created. CASE tool vendors use these standards to build tools taht will be compatible with one another.

As far as our knowledge, there is not any tool that can be classified as a software development information system. At first sight, generic Enterprise Resource Planning (ERP) packages were thought to be customized for covering the needs of software development organizations, but then the impossibility of this alternative became evident, due to its enormous implementation cost.

3. Insight for the Proposed Software Development Information System

The proposed Software Development Information System pursues the realization of the following goals for software organizations:

Standardization of the activities regarding development and management of software projects,

To serve as a software system which facilitates the operation of the company according to the software development related standards such as CMM (Level 3), IEEE/EIA 12207 (Standard for Information Technology), IEEE 1490 (Adoption of PMI Standard, A Guide to the Project Management Body of Knowledge),

To maximize the integration among the functions of the company and to enhance the information sharing among the people,

To provide cost savings by enabling more efficient and effective performance of the operations,

To supply inputs about performances of resources and processes which enable the initiation of improvement actions,

As a result of the above items, to increase the quality of the software delivered.

Figure 2 and Figure 3 depicts the flow, which software development projects usually follow for project initiation, planning, execution and monitoring from the commencement to the closing of the project. This flow represents the requirements that are pursued by the software development information system solution we propose in this paper.

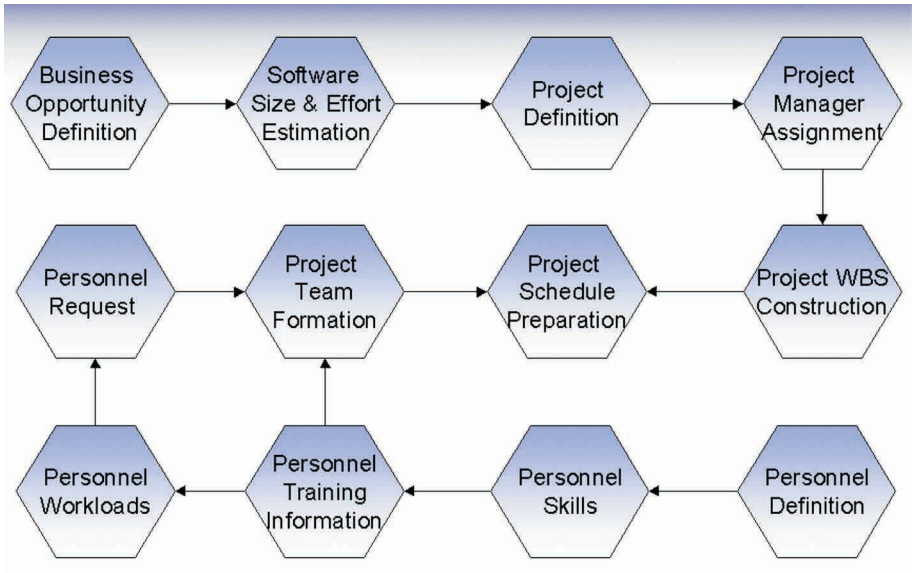

Figure 2. Initiation and Planning of Software Development Projects.

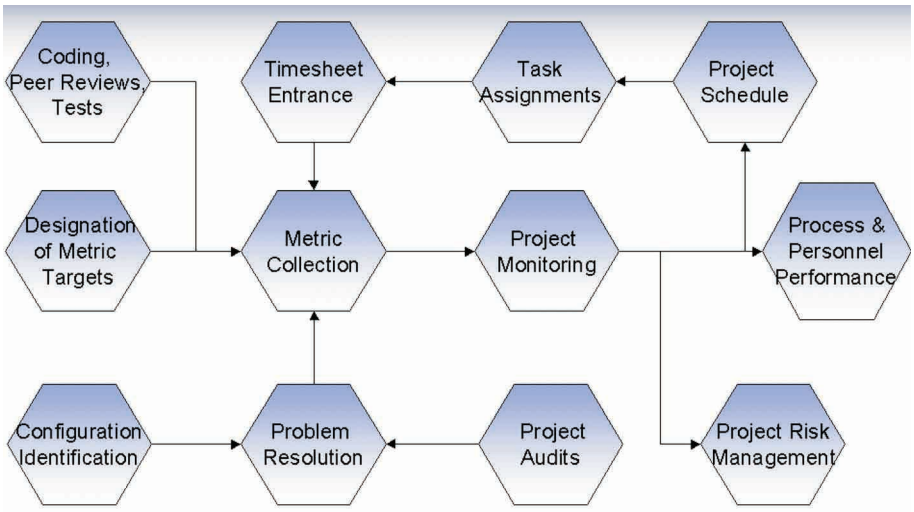

Figure 3. Execution and monitoring of software development projects.

4. Functions of the Proposed Software Development Information Systems

Main functions that are included in the proposed system are explained below. These describe the group of operations that can be performed by the user which enable establishing the high level operational information needed during the software development process in order to accomplish a propoer management with respect to quantitative objectives.

Project Risk Management: Project Risk Management function is employed to identify, assess and track the risks (risk items) of software projects. This is accomplished via a risk matrix (composed of risk items), which is available for each project in the system. Furthermore, the versions of risk matrix are stored in the system.

Software Estimation: The purpose of this function is to provide the necessary capabilities to the user in order to estimate, as accurately as possible, the size, effort, cost and schedule of a software project. The estimation of a software project are performed according to three different methodologies which are; function point estimation, object point estimation and COCOMO II (Constructive Cost Modeling). The system also gives the opportunity of performing estimations for different phases and components of the project.

Metric Data Management: The user tracks project metrics by the help of this function, in order to take the necessary actions for completing the project in the allowed budget and time. For each metric, a datasheet is maintained, which contains the measurement records for the corresponding metric. Each measurement point is composed of data required for the metric. These data are obtained generally from the related parts of the system and seldom from the user. By utilizing the information in the datasheets, metric charts are generated. Thus, a visual analysis can be carried out for identifying the potential problems regarding the software project, more easily. The followings are the examples of the metrics which can be included in the system: Earned Value, Rework Effort, Defect Density, Software Productivity, Requirement Status, Problem Status.

Construction of Project WBS and Schedule: By using this function, the user can form the project's WBS (Work Breakdown Structure). During this operation the activities defined in the system are employed. However, the user has the opportunity of adding detailed activities under these, so the project specific activities may be embraced. The task assignments of the projects are done by specifying the corresponding item in the WBS. The WBSs constituted are product based, i.e. the configuration items of the project (such as modules, software units, documents etc.) can be included in the WBS. The system allows the creation of WBS templates, which can be utilized while constructing a project's WBS. For each item in the WBS, start date, end date, duration, effort, assigned personnel, dependencies with other items, related cash flows are stored for coming up with project schedule. The schedule is also represented as a Gantt chart. Finally, different versions of project WBS and schedule can be kept in the system.

Timesheet: Each personnel enters the time s/he expanded for the tasks s/he performed via Timesheet function. Task assignments made by work group managers result in the formation of related timesheet entries in related personnel's timesheet.

Work Group (Team) Management: A personnel group, in which several people participates for accomplishing a certain goal, is called a work group. Consequently, departments and projects are also work groups. However, other workgroups can also be constructed, e.g. a work group which involves personnel from different department can be formed in order to complete the business planning of the company for

the next year. The operations concerning this work group mechanism are handled via Work Group (Team) Management function. A corresponding work group is created automatically, when a new department or project is added to the system. A hierarchical structure can be constructed for the work groups, i.e. a work group can be defined under another one.

Task Assignment: The managers of the work groups specify the activities that they want to be completed by their personnel by using Task Assignment function. The activities included in the project's Work Breakdown Structure (WBS) are utilized when performing task assignments for a project work group. The following main data is entered to the system for each task assignment: Personnel, Activity, Start Date, Due Date, Duration, Action Type (as creation or rework). By using the duration of the assignment, the workload of the personnel available for the work group and the other task assignments created for the same work group, the feasibility of the task assignment is checked. In fact, this feature is crucial for a software organization in order to manage its human resources properly.

Training Management: In this function, the related capabilities, regarding the in-house and out-sourced trainings, are involved. In this context, training courses are defined. While defining the courses, the personnel skills (entered via Personnel Information Management function) acquired as a result of taking a specific course are also stored. Furthermore, the related records about planned and performed trainings are maintained. The requisition of trainings are provided. Another important feature is the automatic updating of the skills for a personnel as s/he takes a particular training. In this situation, the skills corresponding to the attended training are appended to the skill list of this personnel. Also, the list of the trainings that the personnel attended is updated automatically.

Problem Definition & Resolution: Problem resolution process is implemented by the forms that record and track the status of different types of problems. These include document, software, deviation from standards, process inconsistency problems which originate from different activities such as peer reviews, audits, tests etc.

Audit Operations: This function enables the planning of project audits. Also, audit results are recorded along with the non-conformances detected during audits.

Configuration Identification and Control: Through this function, project configuration items (software work products) are defined and the current status of each configuration item is tracked. Furthermore, it enables the realization of changes in configuration items after necessary control mechanisms are completed successfully.

In addition to the functions described the above, the proposed model also includes the following functions (although not explained here in detail): Project Definition and Initiation, Project Costing, Activity Definition, Personnel Information Management, Personnel Performance Evaluation, Budgeting.

5. An Illustrative Software Development Information System and its Relationship with Software Capability Maturity Model (CMM)

As mentioned before, we will employ PACE system as a representative Software Development Information System for the proposed model. PACE is a web-based software development information system developed to be used in software development enterprises. PACE aims the integration of different facets of a software company via the embracement of project management, budgeting, quality assurance, configuration management and human resources functions in a software setting. PACE system is composed of five modules as shown in Figure 4 along with the functions contained in each of them. PACE System is developed via the

use of Java 2 Enterprise Edition (J2EE) Technology and with an N-tier architecture[2]. Also, independence in terms of platform, application server and database are considered during the design and implementation of the system. PACE system is currently in pilot use, thus we can not present actual evaluation results here.

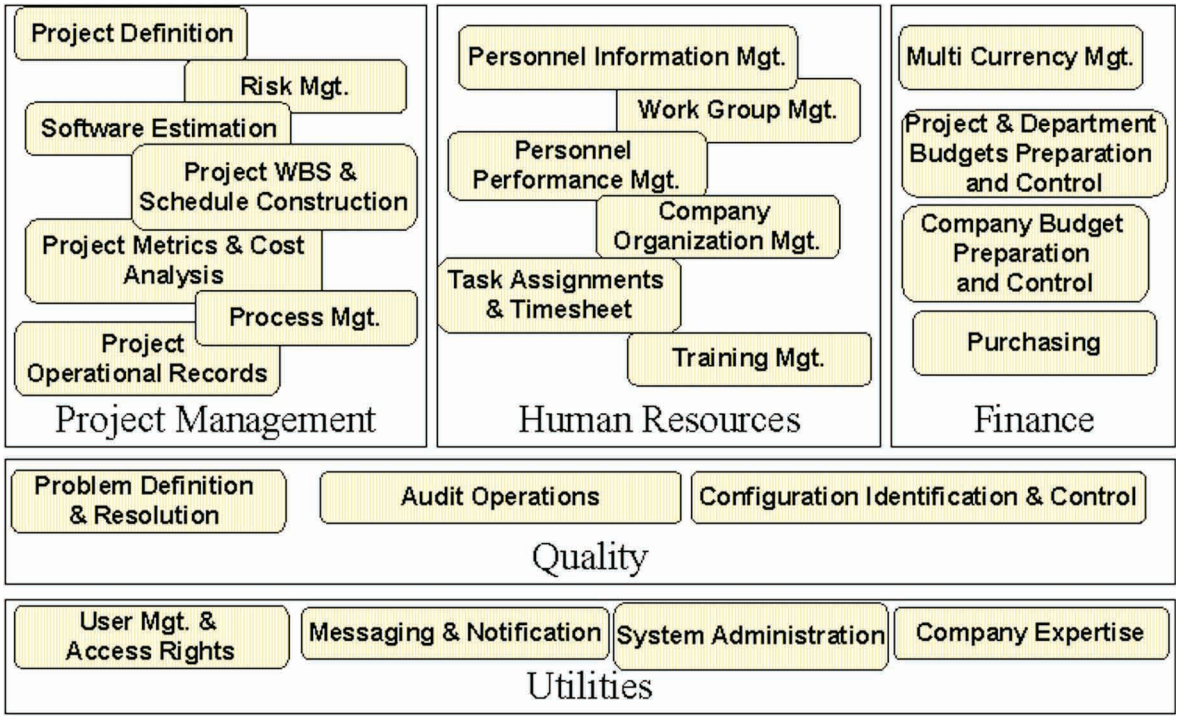

Figure 4. Modules of PACE.

CMM (Level 3) is the primary standard that PACE intends to comply, when employed in a software company. The reason for having such an aim is the high recognition of this standard throughout the software world. Furthermore, SEI describes level 3 as the one through which organization establishes an infrastructure that institutionalizes effective software engineering and management processes across all projects [14]. Each level of CMM is decomposed into Key Process Areas (KPA). SEI defines KPA as a cluster of related activities that, when performed collectively, achieve a set of goals considered important for establishing process capability [1]. The key process areas of CMM Levels 2 and 3 are:

1. Requirements Management (L2)	2. Software Project Planning (L2)
3. Software Project Tracking and Oversight (L2)	4. Software Subcontract Management (L2)
5. Software Quality Assurance (L2)	6. Software Configuration Management (L2)
7. Organization Process Focus (L3)	8. Organization Process Definition (L3)
9. Training Program (L3)	10. Integrated Software Management (L3)
11. Software Product Engineering (L3)	12. Intergroup Coordination (L3)
13. Peer Reviews (L3)	

The handling of each relevant KPA by PACE system is as follows;

Requirements Management: Different types of requirements are defined and they are tracked via *Requirements Status* and *Requirements Stability* metrics. The changes in the requirements are performed according to formal change management mechanisms.

[2]The further details regarding the product architecture cen not be given here due to confidentiality reasons.

Software Project Planning: Size, effort and schedule estimations are performed via Function Point, Object Point, COCOMO II methods. Furthermore, Work Breakdown Structure (WBS) and schedule of the project are constructed. Software and Hardware inventory of software development projects are maintained. Project risks are identified.

Software Project Tracking and Oversight: Various project metrics regarding the quality, effort, schedule, productivity perspective are tracked. Project schedules are updated as needed. Also, the risks of the project are tracked via risk matrices. Finally, the lessons learned from the projects can be maintained in order to feed back the gained expertise.

Software Quality Assurance: Quality activities are planned and tracked, audit results are maintained, resolution of non-compliant issues is carried out.

Software Configuration Management: Identification of configuration items (software work products) and informing the affected groups about status of Configuration Items through the use Configuration Status Account List are provided. Additionally, the changes in these configuration items are performed by filling in the necessary forms which are subject to approval of configuration control boards.

Organization Process Focus and **Organization Process Definition:** Software development process is maintained and improved through the necessary requisition and approval mechanisms. Furthermore, the processes and activities carried out in the course of software projects are defined, which in turn employed in generating project WBS and assigning tasks to personnel. The effort and cost information gathered from the completed tasks across all projects in the organization are reported. These reports are utilized for evaluating the performance of activities and processes. Organizational standards are made available online to project individuals in accordance with the project's selected software process.

Integrated Software Management: Project schedules are constructed in accordance with software development plan by selecting among organization-wide process database. Historical metric data and estimation results are made available for further estimation by the related functions implemented in Project Management module.

Intergroup Coordination: Dependencies between different groups are identified through the WBS construction and scheduling functions. Automatic messages are generated for notifying the related parties when certain events are occurred.

Peer Reviews: The planning of peer reviews may be recorded in project schedule. The results of peer reviews and tracking the related problem reports or action items can be traced through the use Peer Review Summary Report forms. Additionally, peer reviews are analyzed via the use of review status metric.

Training Program: Training courses are defined, training plans are prepared, records of completed training are kept, the skills corresponding to the attended training are added to the skill list of personnel automatically, training waivers are handled.

6. Conclusion

For continual existence in competitive markets, software development organizations shall have the capability of coping with complex software projects which execute simultaneously. The simultaneous execution further complicates the situation and makes the proper achievement of these projects a more difficult task. Consequently, software development organizations should streamline their management and development processes by incorporating software development information systems into their system for accomplishing the organizational objectives. In this paper, a proposal for such systems is presented, which has the capability of delivering real benefits to software developing organizations by unifying the high level operational knowlegde

and providing an integrated project execution and management environment.

References

[1] T.G. Olson, N.R. Reizer, J.W. Over, *Handbook CMU/SEI-94-HB-01: A Software Process Framework for the SEI Capability Maturity Model*, SEI, September 1994.

[2] TickIT Guide, Guide to Software Quality Management System Construction and Certification Using EN29001, Issue 2.0, DISC TickIT Office, 1992.

[3] ISO 9001: 2000, Quality management systems - Requirements.

[4] Similä J., Kuvaja. P., Krzanik L., "BOOTSTRAP: A Software Process Assessment And Improvement Methodology", International Journal of Software Engineering and Knowledge Engineering, 1995, vol. 5(4) pp. 559-584, 1995.

[5] IEEE/EIA Standard: Industry Implementation of International Standard ISO/IEC 12207:1995 Standard for Information Technology–Software Life Cycle Processes.

[6] ISO/IEC (1998a) 15504-2 Information technology - Software process assessment – Part 2: A reference model for processes and process capability. ISO/IEC TR 15504-2: 1998(E).

[7] S. J. Prowell, C.J. Trammell, R.C. Linger, J.H. Poore, *Cleanroom Software Engineering*, Addison Wesley, Addison Wesley Longman Inc., Massachusetts, 1999.

[8] I. Graham, B. Henderson-Sellers, H. Younessi, *The OPEN Process Specification*, Press, Addison Wesley Longman Limited, England, 1997.

[9] C. Ghezzi, M. Jazayeri, D. Mandrioli, *Fundamentals of Software Engineering*, Prentice Hall, Pearson Education Inc., New Jersey, 2003.

[10] Forte G., "In Search of the Integrated Environment", CASE Outlook, March-April 1989.

[11] Sharon D., Bell R., "Tools That Bind: Creating Integrated Environments", *IEEE Software*, March 1995.

[12] IEEE Computer Society, *SWEBOK: Guide to the Software Engineering Body of Knowledge*, Trial Version 1.0, SEI, May 2001.

[13] R.C. Pressman, *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 2001.

[14] M.C. Paulk, B. Curtis, M.B. Chrissis, C.V. Weber, *Technical Report CMU/SEI93-TR-024: Capability Maturity Model for Software, Version 1.1*, SEI, February 1993.