Capturing Software Project Management Knowledge with PROMOL: Project Management Modeling Language

Kadir Alpaslan Demir
Department of Computer Engineering
Turkish Naval Academy
Istanbul,Turkey

ABSTRACT
The reasons for software project failures are mostly project management related issues. Without analyzing these issues in failed projects or capturing the best practices in successful ones, repeating mistakes is inevitable. We developed a project management modeling language (PROMOL) that helps us to plan, understand, analyze and document management of software projects. PROMOL is a formal and visual modeling language. In this paper, we present an overview of PROMOL, provide a simple example and discuss how PROMOL can be used to analyze and capture functional and dysfunctional behavior in software projects.

Categories and Subject Descriptors
K.6.1 [**Project and People Management**]: Life cycles, Management techniques, Staffing, System analysis and design, systems development K.6.3 [**Software Management**]: Software development, Software process

General Terms
Management.

Keywords
Project management modeling language, Software project management modeling, Knowledge engineering, PROMOL

1. INTRODUCTION
Researchers emphasize that management related issues rather than technical issues are the determining factor in the success and failure of software projects [1,2,3,4]. Therefore, if we want to improve our odds of success in software projects, it is essential to identify the causes of failures and understand the best practices applied in successful projects.

According to Jones [5], the difference between companies succeeding in projects and those failing is that the former employ effective project management tool suites while the latter rely on manual methods. Gannt charts, work breakdown structure (WBS), project/program evaluation and review technique (PERT), critical path method (CPM), decision trees are only a few of the classic project management tools [6]. Most of these tools are developed decades ago and automated tools only present these classic tools in various forms and interfaces.

Conceptual modeling of knowledge is a key study area in knowledge and software engineering. Some of the conceptual modeling tools in software engineering are UML, DFD, state transition, entity-relation and i* diagrams etc. Most of these and other tools only address technical aspects. Other tools in knowledge engineering are CML, KARL, PROTÉGÉ etc. (For an overview [7]). These tools are generally domain specific and require extra effort on the part of practitioners. Simplicity is also important [7].

For quite some time, researchers have focused on developing software development life-cycle methodologies. There are many examples of methodologies such as waterfall, spiral, rapid prototyping, agile development, SCRUM etc. There is another line of research area called software process research within the software engineering discipline. Software process research started back in 1980's through a series of workshops and events. Due to many software application failures, researchers focused on improving the software process. The assumption is that there is a direct correlation between the quality of the software process and the quality of the software application developed. A good example in the software process research is the development of the CMM series models. An area of software process research is software process modeling. There are a number of Process Modeling Languages (PMLs) developed [8]. Some examples are Process Interchange Format (PIF) [9,10], Process Specification Language (PSL) [11], Unified Process Model (UPM) [12], Core Plan Representation (CPR) [13], Workflow Management Coalition Process Definition (WfMC) [14], Architecture of Integrated Information Systems (ARIS) [15]. A review of these PMLs can be found in [16]. In June 2005, Business Process Management Initiative (BPMI) and Object Management Group (OMG) merged their activities and formed the Business Modeling & Integration (BMI) Domain Task Force (DTF). They have developed various standard proposals for different views of process management such as Business Motivation Model (BMM) specification [17], Business Process Definition Metamodel (BPDM) [18]. Even Gannt Charts, PERT, and CPM charts are process models and development of Gannt Charts dates back to 1910s. However, there is a significant difference between the PMLs mentioned above and the process models. While the process models (such as Gannt, PERT, CPM) got wide acceptance in industry, as Fuggetta [8] pointed out few (if any) of the proposed PMLs and related Process-centered Software Engineering Environments (PSEE) have been transferred into industrial practice. Fuggetta states that the goal should be to ease the adoption of PMLs. Most of the PMLs are heavily technical and formal. The wide adoption of Gantt, PERT and CPM charts tell us what the practitioners would like to see in these types of process modeling languages: It is simplicity. Since these PMLs were not widely adopted, we do not have actual project data based on models developed with these languages.

Managing software projects has many aspects. Pinto stresses the importance of modeling the business, technical, financial, environmental, and other dimensions of the project before committing any significant sources or even before the go-ahead [19]. Jaafari provides a simplified highest-level representation of a project model and lists the ideal requirements for a project model [20]. He stresses that we still have a long way to go in realizing such sophisticated modeling systems.

We view project managers and management teams as knowledge engineers. Currently, the primary method for capturing their experiences is either interviewing them or analyzing project plans and lessons learned documents (if they are produced). These documents are mostly textual. A project management modeling language, that is simple, visual and intuitive, will help project managers in planning and documenting the projects. At the same time, a formal language helps us to capture and analyze their experiences through project planning. PROMOL is a visual and formal project management modeling language [21,22,23]. Its goal is to aid the project managers in planning, analyzing, and documenting managements of software projects. Even though its starting point is software projects, it can also be used in other types of projects. Using PROMOL, it is possible to conduct formal analyses on projects to check for best and worst practices. A significant capability of PROMOL is the ability to capture and represent various project management aspects at the same time. Especially with its simplicity, PROMOL tries to overcome the shortcomings of previously proposed PMLs and PSEEs. PROMOL is designed in such a way that even project managers with no technical background can develop visual models with PROMOL. User-centric approach is followed in the design of PROMOL.

In this paper, we present an overview of PROMOL and show how it can be used for analyzing and understanding knowledge and best practices of project managers.

2. PROMOL: A VISUAL AND FORMAL PROJECT MANAGEMENT LANGUAGE

In the core of PROMOL, there are two concepts: Activity and entity. An *activity* is a named process, function or task that occurs over a limited time. An *entity* is something that has a distinct, separate existence, though it does not need to be a material existence. Notice that these are commonly used terms defined in the broadest sense. Examples of activities include requirements analysis, hiring and staffing, meetings, code reviews, testing etc. Even social events such as parties may be considered as activities if they are believed to have a positive effect on the project development. Examples of entities include problem statement, project plan document, stakeholders, design, code, prototype, test criteria etc. Even abstract concepts such as teamwork, leadership, communication etc. may be considered as entities. Management of projects is complex. It has many aspects. Not including mechanisms to represent such influential aspects of managing projects limits our ability in understanding and analyzing project dynamics. Therefore, we believe such broad use of these terms increases the representation capability of the modeling language.

PROMOL consists of relations defined on activities and entities. *a()* represents an activity while *e* represents an entity. Activity and entity names can also be used such as *testing()*, *prototype*. Activities are followed by parenthesis distinguishing them from entities.

Create: When an entity is created as the result of an activity "create" relation is used. For example, project staff may be created as the result of a hiring activity. The mathematical representation of this relation is:

$$e_y = a_x(\ )$$

Transform: Most activities in project management take entities and transform them into other entities. This relation is represented with a "transform" relation. For example, a testing activity takes the code and test oracle and transforms them into test results.

$$e_y = a_x(e_1, e_2, e_3, \ldots, e_n)$$

Divide: An activity or an entity may be divided into its smaller parts. Such relation is represented with the "divide" relation. For example, a design activity may be divided into high-level and low-level design.

$$DIVIDE(a(\ )) = \{a_1(\ ), a_2(\ ), a_3(\ ), \ldots, a_m(\ )\}$$

$$DIVIDE(e) = \{e_1, e_2, e_3, \ldots, e_n\}$$

Aggregate: Activities or entities may be aggregated to represent a certain activity or an entity.

$$a(\ )=AGGREGATE\left(a_1(\ ),a_2(\ ),a_3(\ ),...,a_m(\ )\right)$$

$$e=AGGREGATE\left(e_1,e_2,e_3,...,e_n\right)$$

Next: Arrangement of various activities and entities in a time or logical sequence is of particular interest to project managers. In order to represent ordering relation between activities and entities, the "next" relation is used and depicted with a right arrow.

$$a_1(\ )\rightarrow a_2(\ ) \quad e_1\rightarrow e_2$$

Previous: When an activity or an entity has to precede another entity or activity, relation "previous" is used. It is represented with a left arrow.

$$a_2(\ )\leftarrow a_1(\ )\ e_2\leftarrow e_1$$

Require: The relation "require" is used when an activity or an entity requires other activities or entities to happen or exist. This relation helps us to model various dependencies among different aspects of project management. Such one important aspect is the relation of different stakeholders to different activities. For example, a project sponsor may be interested in the planning phase of a project while the end user is interested in participating in the requirements or testing phase of a project.

$$REQUIRE(a())=\{a_1(),a_2(),a_3(),...,a_m(),e_1,e_2,e_3,...,e_n\}$$

Decision: Decision-making is another important aspect of project management. Documenting the rationale and activities following the decision is essential. To capture such issues, the relation "decision" is used. The decision takes various entities (one of them is the decision criteria) as inputs and it provides the next activity to go based on a particular decision (which is another entity).

$$DECISION\left(e_1,e_2,e_3,...,e_m\right)=\left\{\begin{array}{l}\{decision_1,a_1(\ )\},\\ \{decision_2,a_2(\ )\},\\ \{decision_3,a_3(\ )\},\\ ...,\\ \{decision_n,a_n(\ )\}\end{array}\right\}$$

Exist: The relation "exist" provides us a mechanism to conduct formal analyses. In order to question a model for a certain practice, structure or even the existence of an activity or an entity, we use "exist".

$$EXIST\left(a_1(\ )\right)=true/false$$

$$EXIST\left(e_1\right)=true/false$$

$$EXIST\left(a_1(\ )\rightarrow a_2(\ )\right)=true/false$$

The relation "exist" takes an input and check the model for that input. If there is such an instance, it outputs true otherwise false.

There are also some reserved constructs. START and END denote the beginning or an end of a model. When they appear in small letters, they illustrate the start or an end of a specific activity under a hierarchy, which is detailed with the "divide" relation.

We put special effort and emphasis to limit the number of relations introduced. According to Miller's law, short-term memory is limited to 7±2 chunks of information. Even though, it is possible to introduce many different concepts and specialized relations for specific purposes, the ideal is to make it simple, usable, and as optimal as possible while maintaining the capability to be applicable and scalable. Therefore, the modeling language has a limited number of relations.

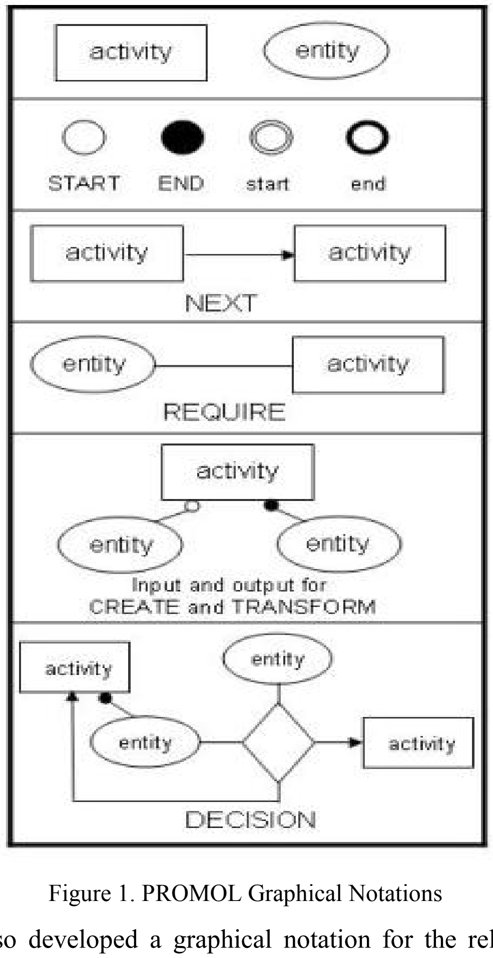

Figure 1. PROMOL Graphical Notations

We also developed a graphical notation for the relations. Figure 1 shows some of the notations used in PROMOL. We put special effort to make the graphical notation intuitive. For example, an activity is represented with a rectangle, which is quite common in the literature. Graphical notation for activities, entities, inputs, outputs and various relations can be found in Figure 1. The relation "previous" is not shown in figure 1 because it is just the opposite of the relation "next". Furthermore, the relations "aggregate", "divide", and "exist" do not have graphical notations. The relations "aggregate" and "divide" are used in hierarchical modeling, while the relation "exist" is only used in the analysis of models.

To illustrate the concepts, a small portion of one of the project management models we have developed is provided [22]. We only show the scope definition of the project, developed with a waterfall life cycle approach. Scope definition ($a_2()$) is an activity at a higher-level model in the hierarchy. The activity ($a_2(\ )$) is detailed using the "DIVIDE" relation. The graphical model is presented in Figure 2.

A portion of the model from one of the case studies is as follows [22]:

<u>Activities</u>
$a_9()$: Identify Problems and Opportunities
$a_{10}()$: Negotiate Scope
$a_{11}()$: Develop Schedule and Budget
$a_{12}()$: Present Project Plan
<u>Entities</u>
e_3 : Project Team
e_4 : System Owners
e_8 : Project Request
e_9 : Problem Statement
e_{10} : Project Vision
e_{11} : Project Budget and Schedule
e_{12} : Department Managers

$$DIVIDE\left(a_2(\ )\right) = \{start, a_9(\ ), a_{10}(\ ), a_{11}(\ ), a_{12}(\ ), e_3, e_4, e_8, e_9, e_{10}, e_{11}, e_{12}, end\}$$
$$start_{a_2} \rightarrow a_9(\ )$$
$$a_9(\ ) \rightarrow a_{10}(\ )$$
$$a_{10}(\ ) \rightarrow a_{11}(\ )$$
$$a_{11}(\ ) \rightarrow a_{12}(\ )$$
$$a_{12}(\ ) \rightarrow end_{a_2}$$
$$e_9 = a_9(e_8)$$
$$e_{10} = a_{10}(e_9)$$
$$e_{11} = a_{11}(e_{10})$$
$$REQUIRE\left(a_9(\ )\right) = \{e_3, e_4, e_{12}\}$$
$$REQUIRE\left(a_{10}(\ )\right) = \{e_3, e_4, e_{12}\}$$
$$REQUIRE\left(a_{11}(\ )\right) = \{e_3\}$$
$$REQUIRE\left(a_{12}(\ )\right) = \{e_3, e_{11}\}$$

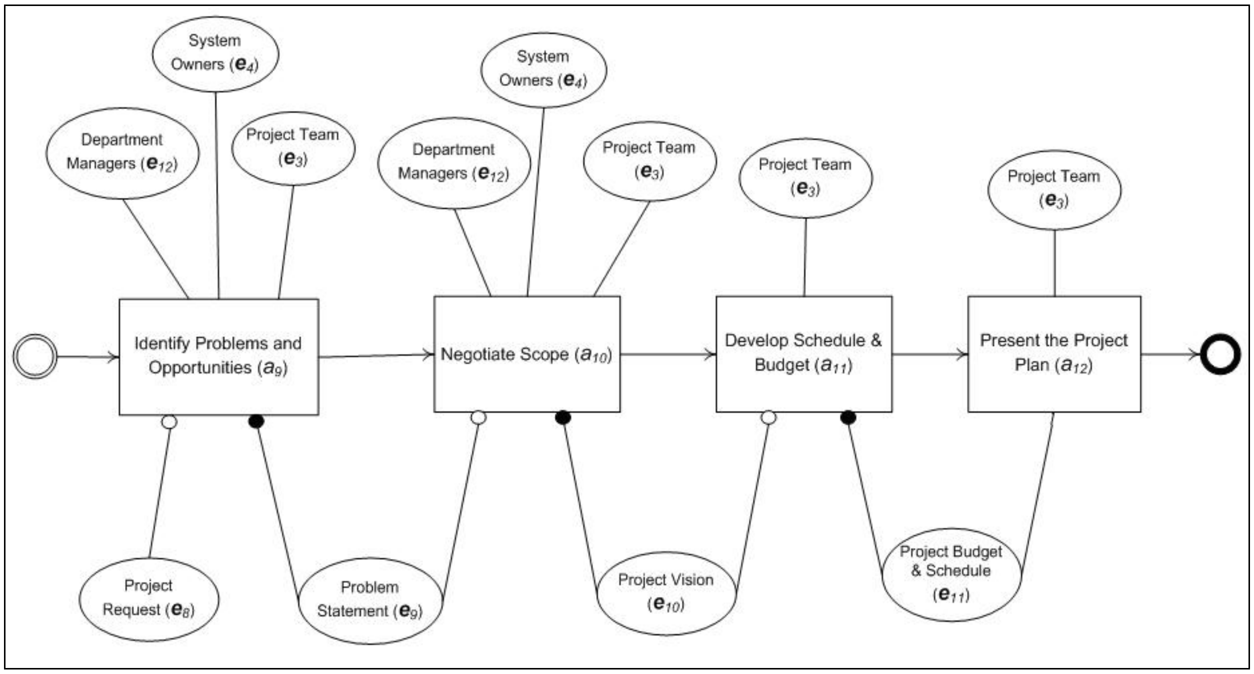
System Owners (e_4)
Department Managers (e_{12})
Project Team (e_3)
System Owners (e_4)
Department Managers (e_{12})
Project Team (e_3)
Project Team (e_3)
Project Team (e_3)
Identify Problems and Opportunities (a_9)
Negotiate Scope (a_{10})
Develop Schedule & Budget (a_{11})
Present the Project Plan (a_{12})
Project Request (e_8)
Problem Statement (e_9)
Project Vision (e_{10})
Project Budget & Schedule (e_{11})

Figure 2. Graphical Model of Scope Definition Activity (Taken from [22])

3. ANALYZING GOOD AND BAD PRACTICES WITH PROMOL

Visual aspect of the modeling language is for the use of practitioners, while mathematical aspect of it is for the development of automated tools and conducting formal analyses on project managements. For example, the relation "exist" does not have a visual counterpart. Since it is only needed for conducting analysis. Analysis of certain best practices in projects particularly is of interest for researchers. It is possible to formally describe some of the best practices with PROMOL and investigate them in the models. Assuming that, there is a model developed by the project manager during project development, it is possible to verify the mathematical model for the existence of certain practices. For example:

"Are requirements development conducted before design?"

$$EXIST\left(design(\ ) \leftarrow requirements\_development(\ )\right) = ?$$

"Is there a prototype before low-level design?"

$$EXIST\left(prototype \leftarrow low\_level\_design\right) = ?$$

"Are code reviews conducted after module coding activities?"

$$EXIST\begin{pmatrix} coding\_moduleA() \rightarrow code\_reviewA() \\ \rightarrow coding\_moduleB() \rightarrow code\_reviewB() \end{pmatrix} = ?$$

Signatures of best practices may be developed using the constructs and relations in PROMOL. Then these best practices may be investigated in a project management model described with PROMOL. The quality and effectiveness measures may be developed based on the existence of sets of best practices and on the lack of bad practices. Since the same technique may be applied to investigate bad project management practices.

4. CONCLUSIONS AND FUTURE WORK

Project management is a complex endeavor and building models helps us to simplify it. Pinto stresses out the importance of modeling the business, technical, financial, environmental, and other dimensions of the project before committing any significant resources or even before the go-ahead [8]. Using PROMOL, we modeled various project managements including a fighter airplane, F-16, multistage improvement project. We acquired a limited set of documents explaining the processes and management aspects of the F-16 multistage improvement project. Using the documents, we build PROMOL models and investigate whether these models ease our understanding of project processes or not. We conclude that the PROMOL models are able present a significant amount of project management information in a concise manner. Furthermore, our experiences indicate that PROMOL is applicable and scalable to real-world projects. In addition, it may be used with any project life-cycle development models.

Using the models, a project manager is able to plan the activities in the project and investigate alternative solutions for the project. Most project management tools only address one aspect of the management. PROMOL triumphs those with its capability to represent multiple important aspects at the same time. PROMOL helps with project activity planning, tracking the inputs and outputs of activities, managing the project artifacts and deliverables, stakeholder management through relating stakeholders to activities and entities, communicating project plans through formality and ease of understanding with visual models. Project plans always change during execution and it is important to reflect this inherent aspect. The PROMOL models easily accommodate changes. It is possible to build high-level models at the beginning of the project and later add detail to the model. The tool allows hierarchy and dynamism in planning. Change in some part of the model does not necessitate change in other parts. Because it is formal and visual, the models eliminate the ambiguity in communicating the project plan to different stakeholders. It is possible to create templates to ensure that certain practices are followed. Achieving a certain level of quality in different projects is important for many organizations.

With being visual, simple, applicable, scalable and able to address more than one aspect, PROMOL is a good candidate for software project management modeling purposes. When project managers use PROMOL for their project planning and documenting needs, the models capture their experiences and knowledge formally. Without extra effort from project managers, researchers are able to analyze the functional and dysfunctional behavior in project management models. Therefore, capturing and analyzing project management knowledge with PROMOL is promising.

Project cost estimation, project scheduling, resource and risk management are among other key responsibilities of project managers and currently, PROMOL lacks the capability to illustrate these aspects. Our investigation showed that it is feasible to include these aspects and we are working on methods to integrate these aspects into the modeling tool. Currently, PROMOL also does not have an automated support. We used various document-processing tools in our case studies. Therefore, we are working on an automated tool that enables practitioners just to work with visual models.

5. ACKNOWLEDGMENTS AND DISCLAIMER

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any affiliated organization or government.

6. REFERENCES

[1] DeMarco, T., and T. Lister,.1999. Peopleware: Productive Projects and Teams, 2nd Edition, Dorset House Publishing Company, New York, NY.

[2] Robertson, S., and Robertson, J., 2005. Requirements-Led Project Management, Pearson Edu. Inc., Boston, MA.

[3] Jones, C., 2004. Software Project Management Practices: Failure versus Success, Crosstalk –The Journal of Defense Software Engineering, Vol. 17, No.10 (October 2004).

[4] Weinberg, G., 1994. Quality Software Management: Volume 3 Congruent Action, Dorset House, New York.

[5] Jones, C., 1998. Project Management Tools and Software Failures and Successes, Crosstalk –The Journal of Defense Software Engineering, (July 1998)

[6] Project Management Institute, 2004. A Guide to the Project Management Body of Knowledge, 3rd Ed.

[7] Dieste, O., et. al., 2001. Conceptual modeling in software engineering and knowledge engineering: Concepts, techniques and trends, Handbook of Software Engineering& Knowledge Engineering, Vol I: Fundamentals, Editor S.K. Chang, World Scientific Publishing Company.

[8] Fuggetta, A., 2000. Software Process: A Roadmap, In Proceedings of the Conference on the Future of Software Engineering, (Limerick, Ireland), 25-34.

[9] Lee, J., Grunninger, M., Jin, Y., Malone, T., Tate, A., Gost, G., and other members of the PIF Working Group, 1996. The PIF Process Interchange Format and Framework Version 1.1 (May 1996).

[10] Lee, J., Grunninger, M., Jin, Y., Malone, T., Tate, A., Gost, G., and other members of the PIF Working Group, 1998. The PIF Process Interchange Format and Framework Version 1.2, The Knowledge Engineering Review, Vol. 13, No. 1, pp. 91-120, Cambridge University Press, (March 1998).

[11] Schlenoff, C., Knutilla, A., Ray, S., 1998. A Robust Process Ontology for Manufacturing Systems Integration, Proc. of the 2nd Int. Conference on Engineering Design and Automation, (Maui, Hawai, August 7-14, 1998).

[12] Kruchten, P., 1999. Unified Process Model (UMP)- A Model of the Rational Unified Process, Proc. of the International Process Technology Workshop, (Villard de Lans, France, September 1999).

[13] Pease, A., 1998. Core Plan Representation, Version 4, (November 1998).

[14] Workflow Management Coalition, 1998. Interface 1: Process Definition Interchange Process Model, WfMC TC-1016-P, (November 1998).

[15] Scheer, A.-W., 1999. ARIS-Business Process Frameworks, 3rd Edition, Springer-Verlag, Berlin.

[16] Breton, E., and Bezivin, J., 2000. An Overview of Industrial Process Meta-Models, 13th Int. Conf. on Software & Systems Engineering and their Applications, (Paris, France, December 5-8, 2000).

[17] Object Management Group, 2007. Business Motivation Model Specification, version 1.0, (September 2007).

[18] Object Management Group, 2007. Business Process Definition Metamodel, Beta 1, (July 2007).

[19] Pinto, J.K., 1998. Project Management Handbook, Chapter 1, Project Management Institute, Jossey-Bass Inc., San Francisco, California, pp. 11.

[20] Jaafari, A., 2004. Modeling of Large Projects, Chapter 13 in The Wiley Guide to Managing Projects, Edited by P. W.G. Morris, and J. K. Pinto, John Wiley and Sons, Inc., Hoboken, New Jersey, pp. 301-302.

[21] Demir, K.A., and Osmundson, J., 2008. A Theory of Software Project Management and PROMOL: A Project Management Modeling Language, Technical Report, NPS-IS-08-006, Naval Postgraduate School, Monterey, CA, USA, (March 2008).

[22] Erguner, A. 2008. Applicability of a novel project management theory and modeling language, M.S. Thesis in Information Systems, Naval Postgraduate School, Monterey, CA, (March 2008).

[23] Demir, K.A., and Erguner, A., Software Project Management Modeling with PROMOL, (in Turkish), Software Quality and Software Development Tools Symposium 2008, Istanbul, Turkey (9-10 October 2008).

[24] Pinto, J.K., 1998. Project Management Handbook*,* PMI, Jossey Bass Inc., San Francisco, CA.

[25] Jorgensen, M., and Molokken-Ostvold, K. 2006. How large are software cost overruns? A review of the 1994 CHAOS report, Information and Software Technology 48 (2006), 297-301.