NPS-IS-08-006

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

A Theory of Software Project Management and PROMOL: A Project Management Modeling Language

by

Kadir Alpaslan Demir and John S. Osmundson

01 March 2008

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Daniel T. Oliver
President

Leonard A. Ferrari
Provost

This report was prepared for the Information Sciences Department.

This report was prepared by:

_____
Kadir Alpaslan Demir
PhD Candidate in Software Engineering

_____
John S. Osmundson
Assoc. Professor of Information Science

Reviewed by:

_____
Dan C. Boger
Chairman
Department of Information Sciences

Released By:

_____
Dan C. Boger
Interim Associate Provost and
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2008	**3. REPORT TYPE AND DATES COVERED** Technical Report	
4. TITLE AND SUBTITLE: Title (Mix case letters) A Theory of Software Project Management and PROMOL: A Project Management Modeling Language			**5. FUNDING NUMBERS**
6. AUTHOR(S) Kadir Alpaslan Demir and John S. Osmundson			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES Replacement of ADB# 337242			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Effective software project management is the key to successful completion of IT software projects. A positive theory of software project management helps to illuminate the path to effective management. Here, we introduce a simple, yet powerful, software project management theory that helps us to understand the conditions and drivers that lead to functional and dysfunctional project behavior. We identify a set of criteria for assessing current and future modeling tools. Finally, we introduce a formal and visual modeling language for management of software projects.

14. SUBJECT TERMS Software Project Management, Project Management Theory, Software Project Management Tools, Software Project Management Modeling, Criteria for Assessment of Project Management Tools	**15. NUMBER OF PAGES** 37
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Effective software project management is the key to successful completion of IT software projects. A positive theory of software project management helps to illuminate the path to effective management. Here, we introduce a simple, yet powerful, software project management theory that helps us to understand the conditions and drivers that lead to functional and dysfunctional project behavior. We identify a set of criteria for assessing current and future modeling tools. Finally, we introduce a formal and visual modeling language for management of software projects.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. STATEMENT OF THE PROBLEM

Software project management is a complex task by its nature. The project manager and the management team have to effectively deal with the complexities of project management. One way to overcome or reduce this complexity is to enable them with tools to make this endeavor simpler and visual. Therefore, our goal is to develop a project management modeling language. However, the software project management discipline lacks a general applicable theory. Therefore, we also introduce such a theory to guide us in developing the modeling language.

B. SIGNIFICANCE OF THE PROBLEM

IT software projects are still suffering from lack of delivering successful results. According to a recent IT projects survey results scheduled to appear in IEEE Software, the most-up-date numbers indicate that 26%-34% of IT software project fail [8]. This study argues some of the results that has been widely become known as CHAOS study, in which the IT projects completed on time and on budget only goes up to 16% [2]. There are also some other studies that report various success and failure rates [11,14,15,23]. Even with the lowest failure rates reported in these studies, software projects are failing significantly when compared to projects in other fields. In [25], current project management issues in leading project-based industries are listed. Among nine industries, in only software industry column, overruns and poor performance is explicitly listed as an issue among others. The average software project is likely to be six to 12 months behind schedule and 50 to 100 percent over budget [31]. One would expect that our record in software projects should have been much better with all the advancements in software engineering. However, we believe relying merely on technological advances would be misleading. We also need significant advances in software project management field to achieve better results in software projects. Therefore, proposals and discussions for applicable and viable theories, models, tools and practices in software project management are important steps in this direction.

DeMarco and Lister state that "*For overwhelming majority of the bankrupt projects we studied, there was not a single technological issue to explain the failure*." in their seminal book Peopleware [6]. Robertson et. al. emphasize that "*In several decades of project experience, we have never seen a project fail for technical reasons. It has always been human failures that have caused otherwise good projects to grind to a halt*." [22]. Defense Science's Board report on 2000, indicates that management is one of the hardest part of software task [5]. According to Weinberg, the three causes of software project failures are "people, people, and people." [30]. It is possible to increase the number of references [7,19,26]. Simply put, project management related issues are the determining factors for our failures and successes in software projects. We need better methods to address the challenges of management side of software projects if we want to improve our odds of success in IT software projects. 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 [20]. The dimensions or aspects of the project should be modeled as far as possible and as cost-effectively as possible. Therefore, we focus on this need in software project management and propose a modeling tool to enable practitioners to model and visualize various aspects of the project management at the same time.

C. LITERATURE REVIEW

Project Management Body of Knowledge (PMBOK) is a significant guideline in project management literature [21]. It outlines the field of project management and provides guidelines in nine management knowledge areas listed in the document. Capability Maturity Model Integration (CMMI) ver 1.2 is the result of years of work by Software Engineering Institute [4]. It is a process improvement approach for organizations wanting to improve their processes to achieve better project results. CMMI identifies five maturity levels for organizations based on the processes they employ in the development of projects. It establishes requirements for process areas in order to achieve higher maturity levels. It puts a certain amount of emphasis on project management

related issues and includes process areas such as integrated project management, project planning, project monitoring and control, risk management, quantitative project management etc. IEEE's Software Engineering Body of Knowledge (SWEBOK) 2004 version is a first baseline for the body of knowledge for the field of software engineering [13]. The guideline employs ten knowledge areas for software engineering field and one of them is software engineering management. These guidelines and standards are among the most important documents in the literature and embodies decades of knowledge and experiences. However, none of the documents is able to provide a theory of project management or a formal approach to project management.

In 2006, Rodney Turner, the editor of International Journal of Project Management, wrote a series of editorials. In these editorials, he pointed out that the project management field was not yet recognized as a proper academic discipline. One reason is that we lack a theory of project management. In that and following editorials, he introduced a normative theory of project management. A normative theory merely expresses what the norm should be. With the theory, Turner explained the domain, the nature of project management, its governance and the functions of project management [27,28,29]. One year later, in 2007, Sauer and Reich wrote a response as guest editorials [24]. While promoting the idea of having a normative theory, they expressed the need for a theory that helps us to understand the conditions, constraints, and drivers leading to functional and dysfunctional behaviors. Therefore, we can influence such behavior to reach intended results.

Even though there have been many advances in software engineering, there have been far fewer advances in software project management. Most of the tools that are widely used such as project evaluation and review technique (PERT) [16], critical path method (CPM) [16], GANTT charts [10], work breakdown structure (WBS) [3], decision tree diagrams for risk assessment [21], network planning models [12] etc. were developed decades ago. Microsoft Project, Oracle Projects, OpenProj, Attask, Project.net, Daptiv, and Celoxis are just a few examples among many automated project management tools. These automated tools from different vendors only offer the classic tools in various forms

and interfaces. Some of them combine a few classic tools. Brief discussions about the most widely used tools are provided.

Work Breakdown Structure (WBS): A work breakdown structure (WBS) is a tool to hierarchically decompose the project into phases, activities or tasks. A WBS may be graphical or textual. It is used for planning purposes. Using WBS, we divide the project into smaller manageable pieces. Developing a work breakdown structure is about identifying what needs to be done. They are often used as the basis for project cost and effort estimation.

Project/Program Evaluation and Review Technique (PERT): It is a statistical approach to develop the schedule of a project. PERT charts use three different estimations for identifying the duration of activities. These are optimistic, pessimistic and most likely durations. Therefore, it is also referred as probabilistic approach. A specific graphical notation is used while developing PERT charts. Before developing a PERT chart for a project, the activities and their dependencies have to be identified.

Critical Path Method (CPM): Critical path method is similar with PERT. It was developed in 1958. CPM is also used to develop a project schedule and has a specific graphical notation. The difference between CPM and PERT is that CPM only uses one estimate while PERT uses three. A project may include simultaneous activities. However, there is always a critical path in the project such that the delay in one of the activities in the path will delay the project. The focus of the CPM is this critical path.

Gantt Charts: Gantt charts take its name after Henry Gantt who designed and published his chart in 1910. Gantt charts graphically present the sequence of activities on a timeline. They are used in project planning and project monitoring. A Gantt chart is a simple but very useful tool in project and program management. On a Gantt chart, there is a timeline on the horizontal axis and there is a list of tasks on the vertical axis. The bars on the chart represent the length of the task on the calendar. The main purpose of a Gantt chart is the scheduling a project based on the start and end dates of each tasks comprising the project.

Organizational Charts: Organizational charts are widely used in project management to communicate the hierarchical structure of the project organization. The hierarchical dependence of people, roles, teams, or departments is represented in a graphical form. In organizations, they reduce the occurrence of unnecessary communications and miscommunications.

A software project manager or the management team has to combine the tools mentioned above and other tools not mentioned here in order to manage the complexities of managing IT software projects. A tool that addresses various aspects at the same time is an added benefit. This is important since the nature of project management is complex and requires multitasking. According to Jones, successful projects employ effective project management tool suites [14] while unsuccessful ones generally do not. He also provides statistics on the kind of tools the leading, average and lagging projects employ. In terms of number of project management tools deployed, there is a 6-to-1 ratio between the leading and lagging projects. Jones emphasize that managers on failing projects are naïve to assume that project planning and estimating are simple enough to be conducted with rules of thumb and manual methods. The modeling tool provided here addresses various project management aspects at the same time. Jaafari provides a very simplified highest-level representation of a project model and lists the ideal requirements for a project model [18]. He stresses that we still have a long way to go in realizing such sophisticated modeling systems. We believe the modeling language proposed is a step in this direction with its capability to address multiple aspects.

THIS PAGE INTENTIONALLY LEFT BLANK

II. A THEORY OF SOFTWARE PROJECT MANAGEMENT

We believe that the project management field would benefit from a simple, yet powerful theory. The theory should be able to capture the essence of vast nature of projects. It should employ common concepts and guide us in developing tools to understand, plan, analyze, and document the projects.

The project management theory employs the following definitions: *A project is an output of the project management function, which is limited over a specific time. The inputs for this function are a limited number of activities and entities related to any part of the project. An activity is a named process, function or task that occurs over limited time. An entity is something that has a distinct, a separate existence, though it does not need to be a material existence.*

We treat the project management as a function. The output of this function is a project. The project consists of all the deliverables, services and products. This project management function takes some inputs. These inputs are in two categories: Activities and entities. We categorize everything related to project management into these two simple concepts. Examples of activities include requirements analysis, design, hiring staff, project status meetings, code reviews, testing, risk identification and even parties. Examples of entities include business need, requirements, system architecture, stakeholders, documents, project manager, code, use cases, customer etc.

Notice a couple of important issues resulting from the definition. The project is limited in time. No project continues forever. Some projects end with the intended product, service or results. Some other projects are cancelled or, among the ones that are completed, their results are not the intended ones. Thus, in the theory, the project definition takes into account different project results. In addition, since the project management is defined as a function, various different inputs results in different project results. Consider a software project management function that takes the business need, some staff and coding activity as its only inputs. The resulting project will likely to be different from the project management function that takes the business need, all

stakeholders, requirements management, design, coding, a proven architecture, testing, independent verification and identification etc. as its inputs. We define the activities and entities in the broadest sense. Even abstract concepts such as leadership, teamwork, communication may be entities.

We formulate a mathematical function using the definition:

$$P = PM(a_1(), a_2(), a_3(), ..., a_m(), e_1, e_2, e_3, ..., e_n)$$

P denotes the project. PM is the project management function. $a_1(), a_2(), a_3(), ..., a_m()$ are various activities conducted during the project. $e_1, e_2, e_3, ..., e_n$ are various entities related to the project. Here, activities are presented as functions. This is the basic difference from an entity. To signify the differences between different activities and entities, a subscript notation is used. These activities and entities are used as basic building blocks of a project management modeling language.

III. PROMOL: PROJECT MANAGEMENT MODELING LANGUAGE

A. CRITERIA FOR ASSESSMENT OF PROJECT MANAGEMENT TOOLS

There are various tools used in project management. Most of these tools are used for project planning, monitoring and analysis purposes and there are various commercial applications automating these tools. These tools include *Work Breakdown Structure (WBS)*, *Project/Program Evaluation and Review Technique (PERT)*, *Critical Path Method (CPM)*, *Gantt Charts*, and *Organizational Charts*.

It is important to know the strengths, weaknesses and capabilities of project management tools in order to maximize the benefit from them. To the best of our knowledge, there has not been an established set of criteria for assessing project management tools. Here, we introduce such criteria. Having a set of criteria will enable us to evaluate existing project management tools while guiding the development of new tools.

1. Simplicity: A quick overview of the tools mentioned previously would yield that all of them have a common aspect. They are simple in concept and easy to apply. A project management tool should be simple. It should be easy to understand, learn, and apply. Simplicity increases the likelihood of wide acceptance from practitioners. While being simple, the tool should be comprehensive and scalable enough for its intended purpose.

2. The tool reasonably addresses at least one aspect of project management: Obviously, the project management tool should provide a means to reasonably address at least one concern regarding the project management issues. Project managers' responsibility is to complete the project within defined scope, expected quality, supplied budget and estimated schedule while dealing with stakeholders and the project development team. The tool should help the management in dealing with one or more of these and other aspects. For example, it may provide alternatives for project scheduling under different constraints; list alternative cost estimates under various conditions; help to manage or synchronize stakeholder involvements.

3. The tool allows for formal analysis: Formality may be simply defined as following a set of prescribed rules. A formal tool does not necessarily have to include complex mathematical concepts. Even though the tool may have very few rules, but a well-defined set of rules, it is still a formal tool. Therefore, it enables us to conduct formal analysis on projects. A formal tool lays the foundation for common understanding and misinterpretations due to different views are eliminated.

4. The tool is based on the concepts extracted from the project management environment: The project management environment has certain common concepts such as stakeholders, deliverables, inputs, outputs, tasks, activities, work items etc. The project management tool should be based on these concepts and be suitable for this specific environment. Even though it may be possible to use the tools developed from other domains, it will be intuitive for project managers when the tool utilizes the same concepts and terms.

B. PROMOL (PROJECT MANAGEMENT MODELING LANGUAGE): A VISUAL AND FORMAL MODELING TOOL FOR MANAGEMENT OF SOFTWARE PROJECTS

Using the project management theory and criteria for assessment of project management tools described above, we developed a formal, visual modeling language (PROMOL) for management of software projects. PROMOL simply consists of a predefined set of relations between the concepts derived from the definitions in the project management theory. The relations are used to build models of management of software projects. First, we will define the mathematical relations, and then we will explain its visual counterparts.

1. Relation "CREATE"

During a project, we create an entity as a result of an activity. Such relation between an entity and an activity is depicted via the "create" relation. For example, an entity such as project staff is created as a result of a hiring activity. This relation is defined as follows:

$$e_y = a_x(\ )$$

Where “e” represents an entity whereas “a” represents the activity. Subscripts are used to distinguish different entities and activities. Specific entity or activity names may also be used.

staff = hiring ( )

An activity is followed by a parenthesis. This differentiates the representation of an entity from an activity. Activities may be considered as mathematical functions. In the “hiring” example inputs to the activity may be a group of possible candidates or a set of criteria for the job positions. In some cases it is not necessary to identify the inputs to activities. The necessity of an input or inputs for most cases leads to the second relation, “transform”.

2. Relation “TRANSFORM”

The relation “transform” is one of the basic relations. Whenever an activity takes an input and results in an output, such relation is represented with a “transform” relation. It is defined as follows:

$$e_y = a_x\left(e_1, e_2, e_3, ..., e_n\right)$$

For example, an entity like product specification may be transformed into an entity, product design documentation, as the result of the design activity.

product design documentation = design (product specification)

It is possible to have multiple inputs. In cases, where there are multiple outputs, for every output it is necessary to provide another transform relation. This is especially required to ensure traceability of various outputs or deliverables. An example for a “transform” relation with multiple input is as follows:

product design documentation = design (product specification, specific design patterns)

In the example above, the project manager specifically requires the development team to use specific design patterns to ensure good practices.

It is important to note the similarity with the “transform” relation and the project management function defined in the project management theory. In fact, the project management function is defined as a “transform” relation at the highest level.

3. Relation “DIVIDE”

An activity or an entity may be divided into its smaller activities or entities. Such a relation between activities or entities is represented with the “divide” relation. This relation is denoted with the word “DIVIDE” in capital letters.

$$DIVIDE\left(a(\ )\right) = \left\{a_1(\ ), a_2(\ ), a_3(\ ), ..., a_m(\ )\right\}$$

$$DIVIDE\left(e\right) = \left\{e_1, e_2, e_3, ..., e_n\right\}$$

For example, a testing activity may be divided as component testing and integration testing. An entity such as design may be divided as high-level design and low-level design.

This relation helps us to achieve different levels of granularities when modeling project managements. It provides a hierarchical structure among some activities and entities.

4. Relation “AGGREGATE”

Some activities and entities may be aggregated to depict a certain activity or entity. The “aggregate” relation is used for such purpose. This relation is denoted with the word “AGGREGATE” in capital letters.

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

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

For example, entities such as project schedule, cost estimate, staffing requirement, and risk analysis may be aggregated to the entity project plan.

project plan = AGGREGATE(project schedule, cost estimation, staffing requirement, risk analysis)

5. Relation "NEXT"

Arrangement of various activities is of particular interest for project managers. Some activities occur in a sequence while others occur in parallel. In order to represent the ordering between the activities and entities, the relation "next" is used. For activities, this relation is based on the order of occurrence. For entities, this relation is based on a logical ordering between entities. This logical ordering may also be the time of occurrence whereas it may be a hierarchical arrangement of entities. The relation "next" is depicted with a right arrow between activities or entities.

$$a_1(\ ) \rightarrow a_2(\ )$$

$$e_1 \rightarrow e_2$$

For example, a design activity is followed by a coding activity. An entity such as design may be followed by an entity code. Another example of the relation "next" between entities is the management relation between a technical lead and his developers.

$$leader \rightarrow developerA$$

$$leader \rightarrow developerB$$

6. Relation "PREVIOUS"

The relation "previous" is similar with the relation "next", except it is based on the relation of what precedes what. It is depicted with a left arrow.

$$a_2(\ ) \leftarrow a_1(\ )$$

$$e_2 \leftarrow e_1$$

7. Relation "REQUIRE"

The relation "require" is used when an activity or an entity requires other activities or entities in order to occur or exist. This relation helps us to understand various dependencies among entities and activities. Stakeholder management is one of the key issues for project managers. It is particularly important to track the concerns of various stakeholders for different specific activities. For example, the project sponsor may be required to be included in the loop during project planning activities. However, the end

user may need to be involved during requirements definition and product specification while the end user may have no interest in planning activities. Therefore, the requirements definition requires the end user and project planning requires the project sponsor. Such inherent aspect of project management is included in PROMOL via the "require" relation. This relation is depicted with the word "REQUIRE" in capital letters.

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

8. Relation "DECISION"

Decision-making is another important aspect of project management. It is important for the project decision makers to choose a particular activity path based on various conditions. Such issue is represented with the relation "decision". As a decision maker, a project manager may come to a certain milestone in the project where he/she may need to choose between going forward and repeating a set of activities due to unsatisfying results. The relation is depicted with the word "DECISION" in capital letters.

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

The relation "decision" takes various entities as inputs. At least one of the entities is the condition or the criteria to guide the decision-making. The others are the outputs of activities leading to this decision point. In order to decide, we use the criteria to assess the other entities, which are the outputs of previous activities. The output of this function is a set of tuples. The first part of the tuple is the specific decision and the second part of the tuple is the activity to go after the decision. This relation helps us to follow different activity paths based on various decisions.

For example, we conduct a testing activity in our project. The testing activity will produce the entity, test results. We also have a condition that says in order to move

forward, we have to achieve above a testing coverage threshold. This threshold entity is our criterion. If we achieved less than our threshold, we repeat the testing activity.

9. Relation "EXIST"

This relation is for the purpose of formal analysis. It questions the model for the existence of its input. It checks whether certain activities, entities or model components exists in the model. In real life, this may correspond to analyzing a project for certain best practices in project management. Its output is either true or false. It is defined as follows:

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

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

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

For example, it may be required to check the model to see if a requirement definition activity exists before a design activity.

10. Reserved Definitions

There are also some reserved definitions such as start and end. When they are specified in capital letters (START, END), they represent the beginning and end of a project. When they appear in small letters, they illustrate the start and end of a specific activity or an entity under a hierarchy.

11. Visual Aspect

We put special effort and emphasis on limiting the number of relations introduced. According to Miller's Law [17] 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 still maintaining the capability to be applicable and scalable. Thus, the modeling language has very few number of relations.

Mathematical models enable formality. However, they are less preferable to work on than visual models. The argument for visual models is obvious for software professionals [31]. The success of UML is an example of the achievements with visual models. Therefore, we also introduce the visual counterparts of the defined relations.

An activity is depicted with a rectangle. The name of the activity is written inside this rectangle. An entity is depicted with an ellipse and its name appears inside this ellipse. The next and previous relations are represented with arrows. The require relation is shown with a line between the activities and entities. The input for an activity is represented with a line ending with an empty small circle, while the output is represented with a line ending with a full small circle. Decisions are illustrated with a diamond. The start of a project is shown with an empty circle, while the end is shown with a full circle. Figure 1 presents the corresponding diagrams for the relations.

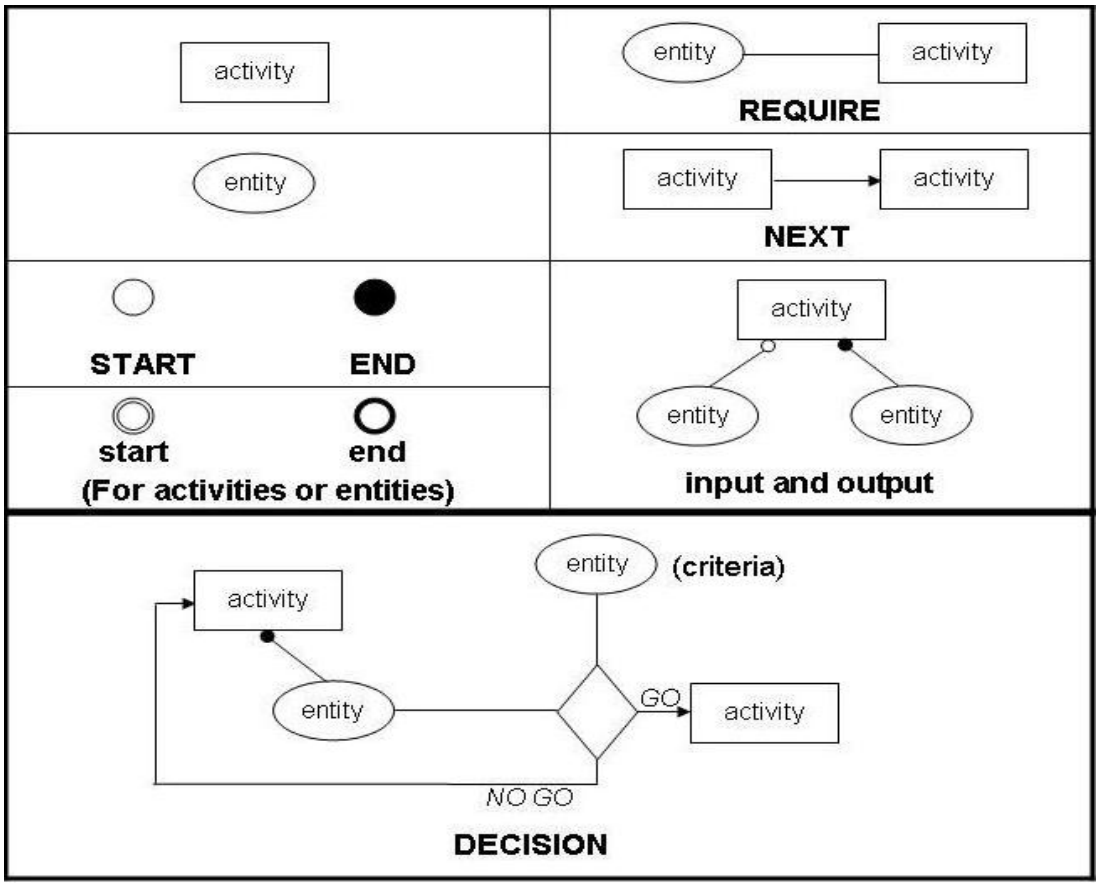

Figure 1. Diagrams and Corresponding Relations

It is important to note a specific notation for the modeling language. Even though, an activity or an entity is referred more than once in different places, they point to the same activity or entity. We made this choice during the design. It eliminates the problems in referring to the same activity or entity among different levels of hierarchy. It also

simplifies the model diagrams by eliminating the difficulty of trying to draw lines to just one activity or entity representation.

C. STEP-BY-STEP MODEL DEVELOPMENT

Below, we present a step-by-step method for model development:

1) Identify the main activities in the project.
2) Identify the stakeholders, deliverables, inputs and outputs for activities and other entities in the project.
3) Write down the project management function as described in the project management theory. This is the highest-level model of the project management.
4) Specify the relations between these activities and entities using the definitions from the modeling language. Also, specify the decision points.
5) Detail the project management model by dividing the activities and entities.
6) Repeat the fifth step until it satisfies the planning needs in a particular project phase.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

There are many advantages in having a modeling language for project management. Project management is complex in nature and building models helps us to simplify the complexities.

- PROMOL helps with the project planning activities. Project planning is one of the major responsibilities of the project manager. Using the models, a project manager is able to plan the activities in the project and to investigate alternative solutions under different resource constraints during project planning phase.
- It is possible to track the inputs and outputs of activities. Thus, it helps with managing the project artifacts and deliverables.
- PROMOL helps with stakeholder management. Stakeholder management is a key issue in project management. The modeling tool allows us to relate various stakeholders with activities.
- The models developed using PROMOL easily adapts to changes. The project plans change during execution. Therefore, it is important that a project management tool reflect this inherent aspect of projects. Using PROMOL, it is possible to build high-level models at the beginning of the project and later add detail to the model using the “divide” relation. The language allows hierarchy and dynamism in planning. Change in some part of the model does not necessitate change in other parts.
- It is formal. Communicating the project plan is important. PROMOL eliminates the ambiguity in communicating the project plan to different stakeholders. Everybody understands the same thing. It allows for development of automated tools and formal analysis of management of projects. We can investigate whether certain best practices are being followed. It is also possible to develop project management metrics.

- It is visual. Diagrams are easy to understand. Lots of information can be presented with just one diagram. It also helps with documenting and communicating project plans.
- PROMOL allows visualizing various aspects of project management at once. The models include activities, developers, inputs, outputs, stakeholders, and the relations among them.
- It is possible to create templates. Process standardization is an issue in big organizations. Achieving a certain level of quality in different projects is important in these organizations. It is possible to create model templates to ensure that certain practices are followed.
- PROMOL allows for extensions. As long as it does not conflict with the preexisting relation definitions, it is possible to extend the language. However, there is a trade-off. Over-specification may defeat the purpose of keeping it simple.

The project management theory introduced here, and PROMOL, a visual modeling language, provide software project managers with a method for planning and monitoring software projects that includes many of the most important project aspects. PROMOL helps to manage the project artifacts and deliverables, helps to relate stakeholders with activities, incorporates project dynamics, eliminates ambiguity through formality, and provides ease of understanding through visual modeling.

B. FUTURE WORK

Project cost estimation, project scheduling, resource and risk management are among other key responsibilities of project managers or management teams [1,9,19] and currently, the modeling language lacks the capability to illustrate these aspects. Our investigation showed that it is possible to include these aspects and currently we are working on methods to integrate these aspects into the language. The language also does not have automated support. Therefore, we are working on an automated tool that will enable practitioners to work just with visual models.

LIST OF REFERENCES

[1] E. M. Bennatan, *On Time Within Budget*, 3rd Edition, John Wiley and Sons, Inc., 2000

[2] Chaos, in: The Standish Group Report, Standish Group, 1994.

[3] D. I. Cleland, *Field Guide to Project Management*, John Wiley and Sons, Inc., 1998, Chapter 7 by P Warner, pp. 74-89.

[4] CMMI Product Team, *Capability Maturity Model Integration Version 1.2*, Software Engineering Institute, Carnegie Mellon University, August 2006

[5] Defense Science Board, Report of the Defense Science Board Task Force on Defense Software, November 2000.

[6] Tom DeMarco, Timothy Lister, *Peopleware: Productive Projects and Teams*, 2nd Edition, Dorset House Publishing Company, New York, NY, 1999

[7] P. J. Denning, "The Field of Programmers Myth", *Communications of the ACM*, Vol. 47, No. 7, July 2004

[8] K. E. Emam, and G. Koru, "A Replicated Survey of IT Software Project Failure Rates", IEEE Software, accepted to appear

[9] K. Forsberg, H. Mooz, H. Cotterman, *Visualizing Project Management, Modeling and Frameworks for Mastering Complex Systems*, 3rd Edition, John Wiley & Sons, Inc. Hoboken, New Jersey, 2005

[10] H. L. Gantt, *Work, Wages, and Profit: Their Influence on the Cost of Living*, The Engineering Magazine, 1910

[11] Contracting for computer software development, FGMSD-80.4, US General Accounting Office, Washington DC, 1979

[12] B. Hughes, and M. Cotterell, *Software Project Management*, 3rd Edition, McGraw-Hill, Berkshire, England, 2002

[13] IEEE Computer Society, *A Guide to Software Engineering Body of Knowledge*, 2004 Version

[14] C. Jones, “Project Management Tools and Software Failures and Successes”, *Crosstalk, The Journal of Defense Software Engineering,* July, 1998

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

[16] H. Kerzner, *Project Management: A Systems Approach to Planning, Scheduling and Controlling*, 4th Edition, Van Nostrand Reinhold, NY, 1992

[17] G.A. Miller, “The Magical Number Seven, Plus or Minus Two: Some Limitations of Our Capability of Information Processing”, *Psychological Review*, Vol. 63, Iss 2, 1956, pp. 81-97

[18] P. W. G. Morris, J. K. Pinto, The Wiley Guide to Managing Projects, John Wiley and Sons, Inc. Hoboken, New Jersey, 2004, Chapter 13 by Ali Jaafari, pp. 301-302.

[19] D. Philips, The Software Project Manager’s Handbook, Principles That Work at Work, IEEE Computer Society, Los Alamitos, CA, 2000

[20] J. Pinto, Project Management Handbook, Project Management Institute, Jossey Bass, Inc., San Fransisco, CA, 1998

[21] Project Management Institute, *A Guide to the Project Management Body of Knowledge*, Third Edition, 2004

[22] S. Robertson, J. Robertson, *Requirements-Led Project Management*, Pearson Education Inc., Boston, MA, 2005

[23] C. Sauer and C. Cuthbertson, "The State of IT project Management in the UK 2002-2003," Computer Weekly 2003.

[24] C. Sauer, and B.H. Reich, “What do we want from a theory of project management? A response to Rodney Turner”, *International Journal of Project Management*, Vol. 25, 2007, pp. 1-2.

[25] D. P. Slevin, D. I. Cleland, J. K. Pinto, *The Frontiers of Project Management Research*, Project Management Institute, Newton Square, Pennsylvania, 2002, pp. 35.

[26] R. Thomsett, “Project Pathology: A Study of Project Failures”, *American Programmer*, July 1995, pp. 8-16.

[27] J. R. Turner, “Towards a theory of project management: The nature of the project”, *International Journal of Project Management*, Vol. 24, 2006, pp. 1-3.

[28] J. R. Turner, “Towards a theory of project management: The nature of the project governance and project management”, *International Journal of Project Management*, Vol. 24, 2006, pp. 93-95.

[29] J. R. Turner, “Towards a theory of project management: The function of project management”, *International Journal of Project Management*, Vol. 24, 2006, pp. 187-189.

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

[31] E. Yourdon, *Death March*, 2nd Edition, Yourdon Press, Pearson Education Inc., Publishing as Prentice Hall Professional Technical Reference, Upper Saddle River, NJ, 2004

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Kadir Alpaslan Demir
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

4. John S. Osmundson
   Department of Information Sciences
   Naval Postgraduate School
   Monterey, California

5. Deniz Kuvvetleri Komutanligi (Turkish Navy Headquarters)
   Ankara, Turkey

6. Deniz Harp Okulu Komutanligi (Turkish Naval Academy)
   Istanbul, Turkey

7. Arastirma Merkezi Komutanligi
   Yazilim Gelistirme Grup Baskanligi
   (Turkish Navy Software Development Center)
   Istanbul, Turkey

8. Deniz Bilimleri Enstitusu ve Muhendisligi Mudurlugu
   (Turkish Naval Science Institution and Engineering School)
   Istanbul, Turkey