2003

Software Project Management: The Role of Modeling

Omar F. El-Gayar

Sreedhar Thota

Software Project Management: The Role of Modeling

Omar F. El-Gayar and Sreedhar Thota

**College of Business and Information Systems, Dakota State University,
Madison, South Dakota, U.S.A.**

ABSTRACT

With the ubiquity of software and in the quest towards improving software development activities, the software engineering research community engaged in an active research agenda on modeling software processes. The modeling objectives ranged from process understanding, improvement, and management, to project management. In spite of the significant achievements in the field, few (if any) of the research results are adopted by the industry.

This paper presents a brief review and a critical evaluation of the status of software process modeling practices with particular emphasis on their applications to project management from a practitioner's point of view as well as recommendations for future work.

Keywords: Software process modeling, project management

1. INTRODUCTION

Since its inception in the late 60's, software engineering focuses on the improving the quality of software as well as the software development process. The underlying premise is that quality processes result in a quality product. Initial effort relied on informal description of software processes. Until the late 1980's when focus turned to more formal description resulting in a number of modeling paradigms and numerous modeling techniques.

Accordingly, software process modeling and in particular process modeling languages (PML) emerged as a technique for defining and analyzing significant aspects of software process with the aim of facilitating human understanding, process management, process improvement, process guidance and project management [1]. However, after more than a decade of active research in the area, few (if any) of the approaches have been transferred into industrial practices. [2,3,4]

It is thus the objective of this article to present a brief review and an evaluation of software modeling practices with particular emphasis on project management from a practitioner's point of view as well as recommendations for future work. The article is organized as follows; section 2 defines software process modeling and its objectives as well as a brief review of software modeling paradigms and techniques. Section 3 defines software project management and develops an assessment framework for evaluating different modeling techniques from a project management perspective. Section 4 reviews a representative set of modeling techniques, while section 5 highlights recommendations for future work.

2. SOFTWARE PROCESS MODELING

A software process is comprised of a set of policies, organizational structure, technologies, procedures, and artifacts needed to conceive, develop, deploy, and maintain a software product [2]. As such, software process modeling seeks to define and analyze significant aspects of these processes [1]. While there is not a wide-ranged consensus on the essential constructs of a process model, most frequently mentioned constructs include agents, roles and artifacts [5] as well as activities and tools [2] as shown in Table 1.

Table 1. Software process modeling constructs

Modeling construct	Description
Agents	An actor who performs an activity (process element)
Roles	Set of responsibilities assigned to an agent
Artifacts	A product created or maintained by the activities
Activities	Steps that need to achieve process objectives
Tools	Are to be utilizes by the process

There are a number of objectives driving software process modeling, most notably [5,6,7]:

- Facilitating human understanding and communication
- Supporting process improvement
- Supporting process management
- Automating process guidance and execution
- Supporting project management
- Facilitating training and learning

To that effect, and since the late 1980's, a number of process modeling paradigms and numerous techniques evolved for modeling software processes. Examples of which include programming language, Petri-net, object-oriented, and quantitative, e.g., systems dynamics. Furthermore, most recently, process-centered software engineering environments (PSEE) have been developed specifically for supporting software development activities both as research projects and as commercial products [4].

While a number of reviews and evaluations of modeling techniques and PSEEs exist in the literature [3,4,5,8,9] these reviews are often general and not relevant from a project management practitioner's perspective. Accordingly, the following section seeks such classification where the support for project management activities is emphasized.

3. PROJECT MANAGEMENT

Project management is the "application of knowledge skills, and techniques to project activities in order to met project requirements" [11]. In engineering projects, activities can be characterized along two dimensions. The first dimension refers to engineering activities that are performed to build whatever the project aims to accomplish, i.e., the product. In software engineering, the product is the software under consideration, and the engineering processes are carried with the concept of a system development life cycle (SDLC), which is the framework for describing the phases involved in developing and maintaining information systems. Popular process models for SDLC include the waterfall model, the spiral model, the incremental release model, and the prototyping model.

On the other hand, the second dimension refers to these activities within project management. In effect, while the associated engineering activities are executed by people and are viewed as a product life cycle processes, the entire project execution can be considered as a process that is followed to build the software. Effectively managing the process is paramount to success [10].

Accordingly, "the set of activities needed to manage the process for a project is specified in the project management process" [10]. The five project management process groups are [11]:

- Project initiation. This includes actions to commit or begin a project.
- Project planning. This includes a wide range of activities including cost and schedule estimation, work plans, staffing, work break down structures, quality management planning, and risk management planning.
- Project execution. This involves coordinating people and other resources to carry out the plan.
- Project control. This ensures that project objectives with respect to cost, schedule, quality, and functionality are met and identifies corrective measure in case the project deviates from the established plan.
- Project closure. This includes formalizing acceptance of the project and the creation of closing document thereby bringing the project to an orderly end.

While each company/project may have its own project management methods, Table 2 identifies a number of project management activities with respect to each of the project process groups. A detailed description of these activities is provided in the PMBOK guide [11].

4. EVALUATION

This section of the paper deals with the evaluation of software process models and environments with respect to software project management. Clearly, a detailed review of all PML, PSEE and other software process modeling approaches is beyond the scope of any one article. Accordingly, this article seeks to demonstrate the applicability of the evaluation grid on a selected subset of modeling approaches. To that effect, we have employed the following criteria to limit the scope of the article:

- A PML has to have a supporting PSEE. The rationale is that from a project management perspective, a stand-alone PML can be very intimidating to use.
- Sufficient background material exists. These include published articles, reports and reviews as well as supporting web sites.
- The approach has to have potential for project management support, as opposed to strict process guidance (from a product-cycle perspective).

As a disclaimer, it should be noted that the evaluation is based on the cited literature and information obtained from their providers (through their web site).

MILOS

Developed by the University of Calgary and the University of Kaiserlautern, MILOS (Minimally Invasive Long-term Organizational Support) is a system that has both process modeling and project planning and enactment features [12]. According to Maurer et. al. [13] MILOS extends MS-Project tool with the means to describe information flow (a feature supported by Workflow Systems).

Table 2. Modeling environments and software project management.

Criteria for Evaluation	MILOS	Endeavors	Little- JIL	SoftPM	System Dynamics
Planning Support					
Scope planning and definition	X	X	X	X	X
Activity Definition	X	X	X	X	X
Activity Sequencing	X	X	X	X	X
Activity Duration Estimating					X
Schedule Development	X		X	X	X
Resource Planning	X	X	X	X	X
Cost Estimating					X
Cost Budgeting					
Quality Planning					X
Organizational Planning					
Communication Planning					
Quantitative Risk analysis					X
Execution Support					
Information Distribution	X	X		X	X
Support for Control					
Schedule Control	X	X		X	
Cost Control					
Quality Control	X	X			X
Performance Reporting				X	
User Interface & Ease of Use	X	X	X	X	
Integration with Project Management tools	X	X			
Support over distributed environments	X	X		X	

This is done via definition of task input and output parameters. The output parameter of a task is mapped to an input parameter of another task. These mappings constitute the information flow. It also allows for a virtual team approach to software development essentially meaning that the members of a software project are not confined to one physical location. This is possible as it is a web based process support system and thus dynamic coordination of distributed software development teams is achieved in MILOS.

To support project management MILOS has three components namely a resource pool component, a project plan management component and a workflow management component. The resource pool component helps managing the agents, roles and agent properties. Scheduling of tasks is possible by querying this component for agents that meet certain criteria.

The project manager can plan and customize the project using the Project Plan Management component. This is done through an interface with COTS tools like MS-Project [14]. The initial project plan is developed in MS-Project that has the start and end dates. The project planner imports the project plan into MILOS. MILOS then helps in the enactment of the plan and during execution the project plan can be refined.

Finally the Workflow management component executes the project plan and manages the products. It can react dynamically to the project plan changes during execution. The three components thus help in project planning, plan enactment and dynamically react to a plan change. Using a connection to other metric tools, MILOS can also achieve quality management.

Endeavors

Endeavors, developed at the University of California, Irvine is a part of a larger scale project on open technology for system evaluation. To that effect, Endeavors is an open, distributed (web-based) environment that builds on the Teamware process modeling language to provide process modeling and execution infrastructure that addresses communication, coordination and control issues. Process modeling capabilities are provided through visual implementation of process constructs (activities, artifacts and resources). On the other hand, support for process execution is supported by providing the team members with summary of assignments, coordinating the creation of artifacts, and tracking execution progress [15].

While Endeavors provide strong process support from a product-cycle (engineering) perspective, there is no explicit support for project management aside from allowing managers (and team members) to track execution progress through information distribution. Alternatively, project management support is indirectly provided through interfacing with project management software, e.g., MS-Project. In that regard, Endeavor, through process modeling, provides for scope planning and definition, activity definition, and activity sequencing. Scope planning and definition is supported through the identification of process artifacts. Explicit representation of process artifacts in the form of work breakdown structure is not provided. Activity sequencing is supported through activity networks that define the

inter-relationship between activities, artifacts and resources.

Endeavors supports user interfaces for visually creating activity networks, assigning resources, attaching artifacts, and browsing, creating, and changing category objects and their attributes.

Little-JIL/Juliette:

Little-JIL, developed at The Laboratory for Advanced Software Engineering Research (LASER), University of Massachusetts Amherst, is a process language focusing on the coordination aspects of processes. Little-JIL is deeply rooted in earlier work on process programming, namely APPL/A [16] and JIL [17]. The primary feature differentiating Little-JIL from earlier research at LASER is that Little-JIL is a graphical language [19]. In this language, the central abstraction is the "step". Steps are organized into a static hierarchy with a dynamic execution structure (defined through control and exception flow mechanisms). Each step is assigned to an agent and is attached to a list of resources required to (human or machine) for execution. When an agent starts executing a step, resource binding in which specific resource instances managed by the resource manager are reserved for the step [18]. It should be noted, however, that the resource manager while a separate entity, is integrated into Little-JIL [19]. This follows one of two underlying hypothesis stating that the coordination structure is separate from other process language issue. The driving motivation is to support and promote process reuse. The second hypothesis emphasizes Little-JIL's focus on coordination as "processes are executed by agents that know how o perform their tasks but benefit from coordination support [19]. While Little-JIL is used to specify step coordination, Juliette, a runtime environment based on Little-JIL provides implementation support to allow for execution of process programs [19].

From a project management perspective, Little-JIL/Juliette provides strong process modeling support for scope definition, activity definition, sequencing and planning, as well as schedule development and resource planning. Scope definition is provided through modeling of artifacts, albeit through a separate artifact management system. On the other hand, activity definition and sequencing is provided through Little-JIL hierarchical modeling structure in which each activity is modeled as a step that is decomposed into sub-steps in a tree like structure. Sequencing, although not provided in the form of network diagrams is embedded in the form of rich control structures provided by Little-JIL. By providing activity duration and estimates and resource requirements and availability, Little-JIL/Juliette with its integrated resource manager provides decision support for a variety of resource planning, coordination and scheduling issues. Examples of which include identifying resource contention issues, as well as evaluating various schedule reduction alternatives. While Little-JIL provides some project management functionality as indicated, the current version does not integrate with existing project management tools. However, the environment is supported by a GUI visual editor thereby facilitating its use.

SoftPM

Developed by the Korea Advanced Institute of Science and Technology, SoftPM (Software Process Management System) is a software process design and management system. The system supports modeling, analyzing and enacting processes [20].

The SoftPM toolset consists of three subsystems. The main system supports process modeling, which is based on the Petri-net paradigm, while the enactment system executes the process and monitors its progress. Enactment of a particular process activity include signaling the agent responsible for carrying the activity, allowing the agent to access the activity applet using a web-browser, allowing the user to download the input artifacts, and uploading the output artifacts. The client system provides agents with the functionality required for artifact exchange through a java-based web interface.

From a project management perspective, the system provides various analysis techniques to aid managerial decision-making in efficiently conducting process activities. Specifically, SoftPM can aid in schedule development and resource planning through calculating the cumulative time consumption for each process activity, spotting agent conflicts, identifying concurrent process activities, and calculating minimum manpower requirements. Other facilities include evaluating the influence of agent conflicts, calculating agent utilization and artifact idle time.

System Dynamics

System dynamics, pioneered by Jay Forrester in the late fifties, refers to a simulation methodology in which elements of control theory are applied to model complex continuous system in social and industrial setting. The emphasis is on capturing information feedback and circular causality among key variables. Such variables are represented as stocks (levels), flows (rates) and the links among variables representing feedback loops.

In project management, system dynamics models are used in domains including large-scale projects in shipbuilding, defense, aerospace, civil construction, power plants, as well as software development [21]. Abdel-Hamid and Madnick [22] pioneered the application of system dynamics to software process modeling in general, and to software project management in particular. They present a generic model of software development and use the model to demonstrate various phenomena encountered in software

projects, e.g., the 90% syndrome and Brooks' Law. The model also illustrates the potential for cost and schedule estimation under different management scenarios as well as the economics of quality assurance.

The use of system dynamics in software project management has also been applied to quality, risk and human resource management. Madachy [23] used system dynamics to evaluate the impact of performing formal inspections on project cost, schedule and quality. The model captured interrelated flows of tasks, errors inspection activities and personnel through the development process and was calibrated to industrial data. Rus et.al. [24] also used system dynamics to quality planning by evaluating various software reliability engineering strategies. In contrast to static reliability models, the model presented can be used to track the quality and reliability of the software throughout the development process by tracking project metrics and adjusts the model accordingly. Another example for the quality planning is a model developed by Tvedt [25] for an incremental software development process. The objective of the model is to evaluate the impact of various process improvement initiatives on development cycle time.

With respect to human resource project management, Collofello et al. [26] uses a system dynamics model to assess the effect of managerial staffing decision on project's budget, schedule and quality. Particular attention is paid to evaluating process to integrate the effects of staff attrition. Abdel-Hamid [27] also used system dynamics to answer "what-if" questions regarding various staffing practices with particular emphasis on the interchangeability of persons and months in a software project.

On project risk management, system dynamics, as a simulation methodology and as presented earlier, is particularly valuable for quantitative risk analysis. In summary, evaluating the impact of various management policies on project cost, schedule and quality.

5. CONCLUSIONS AND RECOMMENDATIONS

The ubiquity of software in our everyday life coupled with its growing complexity is a driving force for software engineering research, in general, and software process research in particular. Nowadays, and after more than two decades of active research resulting in numerous modeling languages and paradigms, few (if any) of these approaches have been adopted by the industry. Moreover, in so far, the focus of existing PML and PSEE is on engineering processes, in particular, providing guidance and process support.

On the other hand, aside from activity coordination and sequencing, this paper indicates little (if any) support is provided to project management processes. In that regard, we note that a project is a temporary endeavor undertaken to accomplish a unique purpose. A project is also characterized as requiring resources, involving uncertainty, and having stakeholders. Incidentally, executing (enacting) a software process (or part thereof) have identical characteristics and thus are projects that need to be managed subject to the triple constraint, i.e., scope, time, and cost. In other words, executing software process is supported by (and thus intricately related to) project management processes.

While integration with existing project management tools is a first step in that direction, the nature of the software development process may require a tighter integration, or even complete assimilation of project management functionality into the PSEE. Specifically, software development is characterized as a creative activity where inconsistency is the rule rather than the exception [28]. While such characteristics impose requirements on PSEE to tolerate and manage inconsistencies and deviations [2], it also emphasizes the need for tight integration (possibly to the extent of complete assimilation) of project management functionality into PSEE. From a software project manager perspective, this alleviates the need to manage two environments, namely, the project management and the PSEE, thereby facilitating adoption. In conclusion, while the literature [2,3,4] identifies several reasons for such slow (or even lack) of adoption, most notably, complexity and inflexibility, we argue that lack of explicit and integrated project management functionality is also a contributing factor.

6. REFERENCES

[1] H. Krasner, J. Terrel, A. Linehan, P. Arnold, and W.H. Ett, Lessons learned from a software process modeling system, Communications of the ACM, Vol. 35, No. 9, September 1992, pp. 91- 100.

[2] A. Fuggetta, Software Process: A Roadmap, Proceedings of the conference on the future of software engineering, Limerick, Ireland, 2000.

[3] G. Cugola and C. Ghezzi , Software Processes:a retrospective and a Path to the Future, Software Process- Improvement and Practice, Vol. 4, June 1998 , pp. 101-123.

[4] V. Ambriola, R. Conradi, and A. Fuggetta, Assesing Process-Centered Software Engineering Environments, ACM Transactions on Software Engineering and Methodology, Vol. 6, No. 3, July 1997, pp. 283-328.

[5] B.Curtis, M.I. Kellner, and J.Over, Process Modeling, Communications of the ACM, Vol. 35, No. 9, September 1992, pp. 73-90.

[6] M.M. Lehman and Department of Computing, Imperial College of Science and Technology, London, England, IEEE, 1990, pp. 91- 94.

[7] M. Kellner, R.J. Madachy, and D.M. Raffo, Software Process Simulation Modeling: Why? What? How?, Journal of Systems and Software, Vol. 46, No. 2/3 , April 1999, pp.1-18.

[8] K. Huff, Software process modeling, In Trends in Software: Software Process, A. Fuggetta and W. Wolf, Eds. John Wiley and Sons, New York, 1996.

[9] P, Garg and M. Jazajeri, PSEEs: A grand tour, In Trends in Software: Software Process, A Fuggetta and W. Wolf, Eds. John Wiley and Sons, New York, 1996.

[10] P. Jalote, CMM in Practice: Processes for executing software projects in InfoSys, Massachusetts: Addison-Wesley, 2000.

[11] Project Management Institute (PMI), A Guide to the Project Management Body of Knowledge, PMI: Pennsylvania, 2000.

[12] F. Maurer, G. Succi, H. Holz, B. Kötting, S. Goldmann, B. Dellen, Software Process Support over the Internet, ICSE 1999,pp. 642-645.

[13] F. Maurer, B. Dellen, F. Bendeck, S. Goldmann, H. Holz, B. Kotting, and M. Schaaf, Merging Project Planning and Web-Enabled Dynamic Workflow Technologies, IEEE Internet Computing, May/June 2000, pp. 65-74.

[14] F. Maurer, H. Holz, Process-Oriented Knowledge Management For Learning Software Organizations, Proceedings of the 12th Knowledge Acquisition Workshop (KAW '99), Banff, Canada, 1999.

[15] G. Bolcer and R. Taylor, Endeavors: A Process System Integration Infrastructure, International Conference on Software Process (ICSP4), Brighton, U.K, December 1996.

[16] S.M. Sutton, Jr., P.L. Tarr, L.J. Osterweil, An Analysis of Process Languages, Department of Computer Science, University of Massachusetts, Amherst, MA 01003, August 1995.

[17] S. M. Sutton, Jr., L. J. Osterweil, The Design of a Next-Generation Process Language, In Proceedings of the Sixth European Software Engineering Conference held jointly with the Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering, Springer-Verlag, 1997 Zurich, Switzerland, pp. 142-158.

[18] B. S. Lerner, A. G. Ninan, L. J. Osterweil, R. M. Podorozhny, Modeling and Managing Resource Utilization in Process, Workflow, and Activity Coordination, Department of Computer Science, University of Massachusetts, Amherst, MA 01003, August 2000.

[19] A.G. Cass, B. S. Lerner, E. K. McCall, L. J. Osterweil, S. M. Sutton, Jr., A. Wise, Little-JIL/Juliette: A Process Definition Language and Interpreter, Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000), June 2000, Limerick, Ireland, pp. 754-757.

[20] S.Min, H.Lee, and D. Bae, SoftPM: A Software Process Management System Reconciling Formalism with Easiness, Information and Software Technology, 42(1), 2000, pp. 1-16.

[21] J. Sterman, System dynamics modeling for project management, 1992 http://web.mit.edu/jsterman/www/SDG/project.html

[22] T. Abdel-Hamid and S. Madnick, Software Project Dynamics: An Integrated Approach, New Jersey: Prentice Hall, 1991.

[23] R. Madachy, System Dynamics Modeling of an Inspection-based Process, Proc. ICSE 96, Berlin, Germany, March 25 - 29, 1996, pp 376 – 386.

[24] I. Rus, J. Collofello, A Decision Support System for Software Reliability Strategy Selection , submitted to the 13th International Conference on Automated Software Engineering, ASE98, 1998.

[25] J.D. Tvedt and J.S. Collofello, Evaluating the Effectiveness of Process Improvements on Software Development Cycle Time via System Dynamics Modeling, Computer Software and Applications Conference (CompSAC'95), 1995.

[26] J. Collofello, I. Rus, A. Chauhan, D. Houston, D. Sycamore and D. Smith-Daniels, A System Dynamics Process Simulator for Staffing Policies Decision Support, Hawaii International Conference on System Sciences (HICSS), January 1998.

[27] T.K. Abdel-Hamid, The dynamics of software project staffing: A system dynamics based simulation approach. IEEE Trans. Software Engineering. 15, 2, 1989.

[28] B. Balzer, "Tolerating inconsistencies," presented at International Conference on Software Engineering (ICSE 13), Austin (TX), 1991.