ADAPTATION OF SOFTWARE PROCESS USING GROUP SUPPORT SYSTEMS AND EVOLUTIONARY MODEL CONSTRUCTS

Mike Godfrey, California State University - Long Beach, mgodfrey@csulb.edu
Thang Nguyen, California State University - Long Beach, tnnguyen@csulb.edu

ABSTRACT

Successful applications of software engineering model constructs requires tailoring software process to existing project development environments. This paper investigates the potential for Group Support Systems (GSS) technologies to aid in software development organization process adaptation for active projects. The role of interactive groupware facilitation is emphasized during the transformation of an organization's standardized software engineering and management processes to a defined set of processes appropriate to the particular project case.

Keywords: group support systems, groupware, software engineering, project management, methods assessment, software process visibility

INTRODUCTION

Software development organizations seeking to overcome their experiences of wide variations in cost, schedule and product quality are increasingly turning to continuous process improvement approaches that enable them to better manage information systems development projects (12). Noteworthy among the emerging approaches for improving organizational software process is the Capability Maturity Model or CMM developed under the auspices of the Software Engineering Institute (SEI) at Carnegie Mellon University (5). This particular software engineering model uses a five-level scale to grade organizations on how successful they are likely to be in creating software systems that meet their customer's needs.

Briefly described, the following five levels of the model provide organizations with a structured approach to assessing their software engineering capabilities as well as a set of guidelines for planning continuous improvement initiatives (1).

> Initial Level #1: "The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics."
>
> Repeatable Level #2: "Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications."
>
> Defined Level #3: "The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for

> the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software."
>
> Managed Level #4: "Detailed measures of the software process and product quality are collected. Both the software process and products are qualitatively understood and controlled."
>
> Optimizing Level #5: "Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies."

The software engineering maturity levels and processes developed in the CMM are primarily directed at improving software capability at the organization level. The term *process* refers here to a "sequence of steps ... using procedures, methods, tools and equipment, to transform raw material (inputs) into a product (output) that is of value to customers (11)". This attention to software process reflects the model's emphasis on methods and practices of value *throughout* an enterprise. Project planners must, therefore, demonstrate care when using the framework of guidelines of the model to help them assign priorities and action items for a specific project. The uniqueness and particular constraints and risk factors influencing an active project may or may not benefit directly from the priorities inherent in the model's structure (7). The quality of decisions related to model feature adaptation necessarily depends upon the experience, knowledge and judgment of a project's software developers.

This paper investigates the potential for Group Support Systems (GSS) technologies to aid software development organizations in the successful adaptation of CMM's standardized information systems processes to active projects. The first part of this paper outlines the internal structure of the CMM software process for Level 3 organizations. This capability level represents the *defined* level of software engineering organizational maturity and the next evolutionary goal for organizations operating at the *repeatable* level. Software process descriptions with key practices can be found here. In the second section, Group Support Systems interactive applications are discussed for their potential in helping develop operational adaptations of specific standardized practices for the Integrated Software Management key process area (KPA) within Level 3 organizations. The paper concludes with a more general commentary on GSS technology support for key process goals and related practices within the SEI's Capability Maturity Model.

Software Process Structure for *Defined* Level Organizations

Organizations operating at Level 3, the *Defined* level, are continually challenged with transforming documented, standardized software processes into an integrated set of tasks and activities that satisfy the unique requirements of each software development project. These same organizations draw upon their various software process assets to achieve the successful transformation of the general *standardized* processes to the *defined* software processes needed for a particular project.

The sources and variety of organizational software process assets include (11): the organization's standard software process including the software process architecture and software

process elements; descriptions of software life cycles approved for use; the guidelines and criteria for tailoring organization's standard software process; the organization's software process database; the library of software process-related documentation.

Defined software processes represent the actual operational software processes (i.e., standards, procedures, tools and methods) an organization uses to match the requirements and characteristics of a project. These processes guide the execution of the activities and tasks of both the managers and technical staff and support the longitudinal evolution of organizational software engineering capabilities. Significant inputs into the development of a project's defined software processes include the organizational software process assets, the system requirements allocated to software, and the selection of the project's software life cycle. The software development plan grows out of the project's defined software process and specific details about how the project will be performed including individuals' roles and software work products. Together, the defined software process and the software development plan enable the execution of the project activities and tasks.

Seven (7) key process areas have been identified in the evolutionary capability maturity model for organizations operating at Level 3. These process areas include (14):

organization process focus - "involves developing and maintaining an understanding of the organization's and project's software processes and coordinating the activities to assess, develop, maintain, and improve these processes..."

organization process definition - "involves developing and maintaining the organization's standard software process, along with related process assets, such as descriptions of software life cycles, process tailoring guidelines and criteria, the organization's software process database, and a library of software process-related documentation..."

training program - "involves first identifying the training needed by the organization, projects, and individuals, then developing or procuring training to address the identified needs..."

integrated software management - "involves developing the project's defined software process and managing the software project using this defined software process. The project's defined software process is tailored from the organization's standard software process to address the specific characteristics of the project..."

software product engineering - "involves performing the engineering tasks to build and maintain the software using the project's defined software process and appropriate methods and tools... The software engineering tasks include analyzing the system requirements allocated to software, developing the software requirements, developing the software architecture, designing the software, and testing the software to verify that it satisfies the specified requirements..."

intergroup coordination - "involves the software engineering group's participation with other project engineering groups to address system-level requirements, objectives, and issues... These requirements, objectives, and plans become the basis for all engineering activity."

peer reviews - "involves a methodical examination of software work products by the producers' peers to identify defects and areas where changes are needed. The specific products that will undergo a peer review are identified in the project's defined software process and scheduled as part of the software project planning activities..."

Within each of the key process areas or KPA's are sets of goals, common features and key practices. Common features are attributes that "indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting." Key practices are further organized by common feature category. Again, the key practices describe software engineering principles for guiding a wide range of information systems projects and they do not specify implementation details but rather leave that to the judgment of the project's management and technical staff.

The five *common features* used to group key practices within the key process areas are (14):

Commitment to Perform: "... the actions the organization must take to ensure that the process is established and will endure... typically involves establishing organizational policy and leadership."

Ability to Perform: "... describes the preconditions that must exist in the project or organization to implement the software process competently... typically involves resources, organizational structures, and training."

Activities Performed: "... describes the activities, roles, and procedures necessary to implement a key process area... typically involves establishing plans and procedures, performing the work, tracking it, and taking corrective actions as necessary."

Measurement and Analysis: ".... describes the basic measurement practices that are necessary to determine status related to the process... typically includes examples of the measurements that could be taken."

Verifying Implementation: "... describes the steps to ensure that the activities are performed in compliance with the process that has been established... typically encompasses reviews and audits by management and software quality assurance."

While the model discussed here provides a framework for organizational software process it does so without specifying how an organization's standard software process is to be implemented for a particular project. In the next section, the focus shifts from structural descriptions of the evolutionary model at the *defined* level to applications of GSS technologies used in developing defined software processes from an organization's standard software process and related process assets.

Groupware-Mediated Software Process Interventions

Software developers in Level 3 organizations explicitly acknowledge the *interdependent* nature of transforming an organization's standard software processes into a set of defined software

processes appropriate for the particular project case. This mutuality extends well beyond simply coordinating the use of a technology resource to defining and relying upon the quality and timeliness of others' work. Indeed these information systems developers must consciously seek ways of articulating and coordinating their distributed and individual work efforts by engaging in activities which may not always be perceived as contributing directly to the software system product or service. Applications of Group Support Systems (GSS) technologies are very appropriate for these kinds of software process interventions (2, 4).

In accordance with the evolutionary capability maturity model the essential inputs to the development of a defined software process include an organization's standard software process, available software process assets, and a selected software project life cycle. The software developers, while employing these input resources, also act as semi-autonomous decision making agents due to the unique project situations and contingencies they face. The quality of defined software processes developed for a project is largely determined by the quality of the communications and judgments of the developers working in often the dispersed and dynamic work settings familiar to many software engineering project ensembles.

Systems developers employ the standard software processes and available software assets to reduce the complexity and overhead costs of developing the project's defined processes. Software process databases, libraries of software process documentation, software life cycle descriptions, criteria guidelines and standard process architectures all contribute to developer decisions that converge on that set of defined processes that becomes operationalized for a particular project. In the discussion that follows Integrated Software Management, a key process area within Level 3 organizations, is examined for its potential in deploying GSS technologies to advantage when developing defined software processes for each project case.

GSS Support for the Integrated Software Management KPA

The Integrated Software Management (ISM) key process area has as its *goals* the development of the defined software process and the management of the software project using the resulting defined process. This KPA activity tailors the standard software process into a defined software process appropriate for the unique characteristics and constraints of each project. Within the defined software process are process elements that have been further decomposed to that level of granularity necessary to clearly describe the process. Examples of process elements include software estimation, software design, coding and peer review. Developers create the defined process by referring, in part, to standard practices contained in the Organization Process Definition KPA.

A *key practice*, Activity 1, listed under the Activities Performed *common feature* of the ISM key process area in the model is examined next in more detail to demonstrate the usefulness of GSS-mediated interventions when defining software process for an actual project (i.e., rather than for standardized organization processes). Activity 1 of the ISM key practice is described in the model as, "The project's defined software process is developed by tailoring the organization's standard software process according to a documented procedure (14)." This key practice is also composed of *sub-practices* - the first of which refers to software life cycle:

> "This procedure typically specifies that:
>
> 1. A software life cycle is: selected from among those approved by the organization, to satisfy the project's contractual and operational constraints; modified, if necessary, in ways permitted by the organization's tailoring guidelines and criteria; and, documented according to the organization's standards."

When one considers the dynamic and often contradictory demands placed upon developers by the working environment even the selection and design of adaptive modifications to a standardized software life cycle can become quite challenging. And, as software developers evolve in sophistication they further develop skills enabling them to integrate hybrid life cycle stage elements in ways that generate additional project schedule and cost savings (9).

Within the evolutionary capability maturity model framework we're currently (and narrowly) focusing on a specific sub-practice (software life cycle), nested within a key practice (developing a defined software process), associated with a common feature (Activities Performed), in the key process area (Integrated Software Management) of an organization operating at Level 3. Even though we're isolating ourselves to decisions associated with life cycle selection and modification this activity can be anything but simple. Consider, for example, a few of the questions one might ask when choosing an effective project life cycle (8): How well do my customer and I understand the requirements at the beginning of the project? How well do I understand the system architecture? How much reliability do I need? How much risk does this project entail? Do I need to provide my customers and management with visible progress throughout the project?

Group Support Systems technologies are well-suited to helping developers converge on a shared response to such critical questions. And, therefore, in selecting an appropriate project life cycle based on the responses and the knowledge-base present in such organizational software assets as matrices of life cycle strengths and weaknesses, and, customization guidelines and criteria (2, 6). These are decisions that require thoughtful reflection and good professional judgment. Groupware supports the creation of the kind of complex and shared 'information space' with links to software assets that will enable the informed decision-making expected of Level 3 organizations.

CONCLUSIONS

Organizations seeking to improve their effectiveness in developing *defined* software processes from *standardized* processes and related software assets can benefit from the interactive groupware capabilities of asynchronous GSS-facilitated decision sessions. How well an organization defines its software processes for a particular project clearly involves professional judgments about how best to adapt standardized practices to the goals of the corresponding key process areas. These professional judgments require an understanding of the evolutionary capability maturity model, the organization (including developers, clients, and project histories), and, characteristics of the particular software engineering project currently under review. It is in just this kind of complex environment that groupware-mediated decision sessions can serve to

elicit and diffuse valuable project-related tacit knowledge that would otherwise remain unavailable to individual software systems developers.

This study examined sub-practices within a single key practice nested within the Activities Performed common feature associated with the Integrated Software Management key process area for a Level 3 organization. For a complete assessment of the potential role of GSS in defining software process many more Level 3 practices and processes would need to be examined. The capacity of GSS application software to perform divergent and convergent tasks, cluster and prioritize, and self-document appears very consistent with the kinds of tacit knowledge elicitation and analyses required for the kinds of complex software process adaptation decisions experienced by those adopting evolutionary model constructs.

REFERENCES

1. Bate, R. et al, "A Systems Engineering Capability Maturity Model, Version 1.1," Software Engineering Institute, CMU/SEI-95-MM-003, November 1995.
2. Connolly, T. L. Jessup and J. Vallacich "Effects of Anonymity and Evaluative Tone on Idea Generation in Computer-Mediated Groups," *Management Science*, vol. 36, no. 6, June 1990.
3. Degrace, Peter, and Stahl, L.H., *Wicked Problems, Righteous Solutions*. Englewood Cliffs, N.J.: Yourdon Press, 1990.
4. Galegher, J. and Kraut, R.E. "Computer-mediated Communication for Intellectual Teamwork: An Experiment in Group Writing," *Information Systems Research*, vol. 5, no. 2, 1994, pp. 110-138.
5. Humphery, W. S. "Characterizing the Software Process: A Maturity Framework," *IEEE Software*, vol. 5, no. 2, March 1988, pp. 73-79.
6. Jones, C. *Assessment and Control of Software Risks.* Englewood Cliffs, N.J.: Yourdon Press, 1994.
7. Jones, C. *Applied Software Measurement: Assuring Productivity and Quality, 2nd Edition*, New York, McGraw-Hill, 1997.
8. McConnell, S. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, 1996.
9. McConnell, S. *Software Project Survival Guide*. Microsoft Press, 1998.
10. NASA/Software Engineering Laboratory, "Software Development History," *Recommended Approach to Software Development, Revision 3*, SEL-81-305, NASA/SEL, GSFC, Greenbelt, MD, 1992.
11. Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V., "The Capability Maturity Model for Software," *Software Engineering*, M. Dorfman and R.H. Thayer, eds., The Institute for Electrical and Electronics Engineers, 1997.
12. Pfleeger, S.L., Software Engineering: Theory and Practice, Prentice-Hall, Inc., 1998.
13. Pressman, R. *Software Engineering A Practitioner's Approach* (4th Edition), New York: McGraw-Hill, 1997.
14. Software Engineering Institute (SEI) *The Capability Maturity Model: Guidelines for Improving the Software Process*, Reading, Mass: Addison-Wesley Longman, 1994.