Assessing the Process Maturity Utilized in Software Engineering Team Project Courses*

JAMES S. COLLOFELLO
Department of Computer Science and Engineering
Arizona State University

CHI HENG NG
Department of Computer Science and Engineering
Arizona State University

ABSTRACT

Many computer science departments offer an introductory software engineering course, which normally provides an introduction to software engineering topics in conjunction with a semester long team project. To ensure students acquire the correct lessons from this project experience, it is essential that the teams utilize well-defined software development processes similar to those practiced by leading software development organizations. Since its inception, the Software Engineering Institute Capability Maturity Model (CMM) has served as a guide for organizations seeking to improve their development practices, through a self-assessment questionnaire. In an effort to assess the maturity of development practices utilized in software engineering courses, an "academic" version of the CMM questionnaire was developed. This questionnaire was distributed to a sample of software engineering instructors in an effort to assess the maturity of academic software engineering course projects. The questionnaire and the survey results are presented and discussed.

I. INTRODUCTION

Many universities/colleges across the country offer an introductory course in software engineering principles at either the undergraduate or graduate level. These courses also usually offer an opportunity for students to work on a team project that typically lasts at least a semester. Since instructors generally adopt their own development approach for course projects, the resulting outcome of success or failure varies widely and is often dependent on the experience of the instructor. To increase the probability of successful projects, instructors should ensure that their recommended development process for teams to follow is appropriate for the course project. This development process should provide software engineering students with exposure to mature practices comparable to those utilized in the best software development organizations. A good model to follow for assessing the development practices to be followed by the teams is the Software Engineering Institute Capability Maturity Model (CMM).

Since its introduction in 1987, CMM has become a de facto standard for assessing and improving software processes. The overall goal is to help an organization identify the key practices to increase the maturity of its software development process. The first step toward improving an organization's software development process is to assess its software process maturity. This can be accomplished through the maturity questionnaire and the accompanying techniques that map a process into one of five maturity levels. The five levels of CMM are summarized as follows:[1]

Level-1: Initial. The software development process is often referred to as ad hoc or chaotic. Few processes are documented. The software development environment is unstable, and sound management practices are absent. Outcomes of the process are unpredictable and success is highly dependent upon individual efforts or heroics.

Level-2: Repeatable. At this level, underlying software project management policies and procedures are in place to track software costs, schedule, and functionality. The software development environment is stable and earlier success on projects with similar applications can be repeated.

Level-3: Defined. The software development process is documented, standardized, and integrated into a standard process for the organization. Both engineering and management activities are stable and repeatable. Foundation is now established for further process improvement.

Level-4: Quantitatively Managed. Quantitative quality goals for both software processes and products are set. An organization-wide database of measurements is collected and analyzed to predict trends in process and product quality. Software process and products are quantitatively understood, and statistically managed.

Level-5: Optimizing. The organization now emphasizes continuous process improvement and defect prevention. It is capable of identifying weaknesses and strengths in its software process by collecting quantitative feedback from the existing process and from innovative ideas and technologies.

Each level of CMM, except the first one, is decomposed into several key process areas (KPAs, 19 in total).[2,3] Each KPA contains issues and activities that must be addressed to achieve a maturity level. A KPA is satisfied when most of the related activities are in place. An organization is designated an overall maturity level once all the corresponding KPAs are satisfied.

This study is an extension of prior published work.[4] In the remainder of this paper, we will describe our approach to assessing the process maturity of software engineering project courses. This approach is modeled after CMM and is tailored to the university environment, as an attempt to assist instructors to improve their

*Based on "Assessing the Process Maturity Utilized in Software Engineering Team Project Courses" by James S. Collofello and Chi Heng Ng, which appeared in the Proceedings of the 1999 Frontiers in Education Conference, San Juan, Puerto Rico, 10–14 November 1999, IEEE Catalog No. 99CH37011, pp. 12a9-5 through 12a9-9, © 1999 IEEE.

courses. The software engineering course maturity questionnaire and its evaluation will be presented, followed by a description of the methodology of our survey and its results. The results will identify the level of discipline followed on projects in software engineering courses. The survey was not a rigorous scientific study, but rather an inexpensive estimation of the maturity level of software engineering courses.

II. Software Engineering Course Maturity Levels

Patterned after CMM, but in a simplified manner, the software engineering course maturity levels provide a means to assess course maturity levels. As with CMM, each level contains the same number of key process areas, customized for the academic environment. The five levels are:

Level-1: Initial. The course does not apply or integrate software engineering principles and practices into the project. Students are taught software engineering principles but overall, guidelines to adopting these principles are not defined. Project instructions given to teams are unclear or vaguely documented. Team assignments are often conducted on an ad hoc basis. Project outcome is largely unpredictable; performance is dependent on group dynamics, and personal abilities.

Level-2: Repeatable. Students are required to perform basic software project planning and tracking activities. The instructor maintains a stable process to ensure that most of the student projects can be successful. Based on the instructor's informal framework, project successes from past semesters can be repeated.

Level-3: Defined. Class projects are undertaken based on a defined, documented, and standardized process. The instructor relies on an established process that serves as a foundation for further process improvement. Course practices can be consistently implemented, regardless of the presence or absence of certain key instructors.

Level-4: Quantitatively Managed. The instructor maintains comprehensive process and project measurements. Course projects are quantitatively measured using statistical methods. Measurements on student projects are gathered and analyzed for further improvement.

Level-5: Optimizing. Emphasis is placed on continuous process improvement and defect prevention. Weaknesses and strengths in development process can be identified by obtaining quantitative feedback from current software process practices, and from testing innovative ideas and technologies. Changes are monitored closely, and the process is refined on a continuous basis.

III. The Questionnaire and its Evaluation

To evaluate process maturity of software engineering courses, a maturity questionnaire similar to the SEI questionnaire[5] was developed. The software engineering course questionnaire[8] was tailored to the university environment. In designing the questionnaire, we borrowed the structure adopted in the SEI questionnaire by categorizing the 55 questions into 4 sections and 19 subsections highlighted by their associated maturity levels and KPAs, respectively. The objective was to enable a smooth transition from one section to the next and to ensure that the questionnaire was easy-to-follow. In addition to the four main sections, a section contained 8 background questions designed to gather information about the respondent and course. This information was evaluated but not considered in assigning the maturity level of the course.

As a brief summary, Level-2 questions (16 questions) pertained to activities for the Repeatable level; Level-3 questions (22 questions) concerned activities associated with the Defined level; Level-4 questions (9 questions) were designed to assess the Quantitatively Managed level; and Level-5 questions (8 questions) dealt with requirements for achieving the Optimizing level. With exception to the background questions, all other questions required merely a 'Yes' or 'No' response. The questionnaire was designed to take approximately 10 minutes to complete.

The questionnaire was evaluated in two simple ways. First, we compared the total number of "Yes" responses for each level to the total number of questions allocated to that level and calculated the percentage. If the percentage reached 80%, the course was deemed to satisfy the requirement of that particular level. In addition, a course had to satisfy the requirement of all lower levels in order to be considered as a candidate for a higher level. In other words, skipping levels was not allowed in this evaluating scheme. Each level serves as a "prerequisite" for the next level, and is a necessary foundation to achieving a higher level. Alternatively, the calculated percentage for each level can also be used to indicate the degree to which a course supports each level. This scheme does not assign a level to a course; rather it provides respondents with an overall understanding of where their course stands in terms of the maturity levels.

IV. Survey Methodology

The survey was posted on the Web for a month, from mid-February to mid-March 1999. The survey was designed to minimize end-user overhead, making it convenient for potential respondents to complete and submit. By combining HTML forms and Perl script, the results were logged automatically. The survey was designed to ensure that only one survey entry was submitted per instructor. Although our design accepted partially completed surveys, almost all surveys submitted were complete.

Survey responses were solicited via an online newsletter to software engineering educators, as well as via direct e-mail invitation. Most responses were the results of our e-mail invitations. E-mail addresses were extracted from several sources. Some of the e-mail addresses came from SEI's published software engineering education directories.[6,7] In most cases, however, e-mail addresses were extracted from instructors' homepages using the readily available

Maturity	Count	Percentage
Level-1	43	81.1%
Level-2	8	15.1%
Level-3	1	1.9%
Level-4	0	0.0%
Level-5	1	1.9%

Table 1. Maturity profile of courses satisfying 80% of requirements.

Internet search engines. The e-mailed invitation in the form of a cover letter was sent to each potential respondent inviting him/her to participate in the survey. The invitation was individually addressed but sent out in bulk, and included the URL of the survey. Approximately 150 universities/colleges were invited to participate. In most cases, no follow-up action was taken to increase the response rate.

Maturity	Count	Percentage
Level-1	39	73.6%
Level-2	7	13.2%
Level-3	6	11.3%
Level-4	0	0.0%
Level-5	1	1.9%

Table 2. Maturity profile of courses satisfying 75% of requirements.

Maturity	Percentage
Level-1	55.2%
Level-2	25.8%
Level-3	15.6%
Level-4	2.7%
Level-5	0.6%

Table 3. 1998 maturity profile of 697 software development organizations.

Maturity	Percentage
Level-2	58.6%
Level-3	65.7%
Level-4	16.6%
Level-5	34.2%

Table 4. Maturity level supported by courses.

Key Process Area	Percentage
Level 2: Repeatable	
Requirements Management	53.8%
Software Project Planning	56.0%
Software Project Control	54.1%
Software Acquisition Management	47.2%
Software Quality Assurance	67.9%
Software Configuration Management	67.3%
Level 3: Defined	
Organization Process Focus	37.7%
Organization Process Definition	29.2%
Organization Training Program	83.0%
Integrated Software Management	61.3%
Software Product Engineering	67.6%
Project Interface Coordination	78.6%
Peer Reviews	71.7%
Level-4: Quantitatively Managed	
Organization Software Asset Commonality	16.4%
Organization Process Performance	23.9%
Statistical Process Management	9.4%
Level 5: Optimizing	
Defect Prevention	32.7%
Organization Process & Technology Innovation	50.9%
Organization Improvement Deployment	24.5%

Table 5. Key process areas supported by courses.

V. Results

There were 53 responses to our survey. Only rarely did more than one instructor respond from the same college; most were the sole participant from their college. Most of the surveys were submitted from the United States. The overall results, in terms of the maturity levels is shown in Table 1. If the percentage threshold was reduced to 75%, more courses reached level 3, but the status of level 4 and 5 remained the same, as shown in Table 2. In either case, a significant gap existed between the levels achieved in software engineering courses and the software industry,[8] as shown in Table 3. Specifically, only approximately 20% of the courses achieved Level-2 or higher, verses 45% in industry. From these results, it was evident that software engineering courses performed at far lower process maturity levels than their industry counterparts.

Alternatively, Table 4 indicates the degree to which courses support each level. Collectively, the sample software engineering courses fell mainly in the Level-1 category. Further, the courses scored better in Level-3 than Level-2, and again in Level-5 than Level-4. This "head heavier than body" phenomenon implied majority of the courses did not have a solid foundation for further process improvement.

Specifically, as shown in Table 5, several KPAs scored better than others: Organization Training Program (83.0%), Project Interface Coordination (78.6%), and Peer Reviews (71.7%). Some KPAs, however, had significantly low results: eight out of the 19 KPAs scored below 50%. The lowest scored KPA was Statistical Process Management (9.4%). We may deduce that particular attention should be given to project planning, tracking, and configuration management. These Level-2 activities seemed to have prevented most Level-1 courses from being promoted to a higher level.

In analyzing the 55 questions individually, a pattern appeared to emerge. In most cases, the underlying activities or training for KPAs were in place for most courses. However, automated tools were usually not available to students for most Level-1 courses. This may be due to a lack of funding as is common in many colleges. In other words, the Level-1 courses appeared ill-equipped in terms of automated tools compared to their non-Level-1 peers.

Table 6 summarizes the background information for Level-1 and non-Level-1 courses. In general, the non-Level-1 courses were taught by more experienced instructors to students with higher industry experience. Also, these students seemed to be more productive—their course duration was slightly longer, but they were able to complete projects almost twice as large.

Background Questions	Level-1	Non-Level-1
Times instructor taught course	6.4	8.2
Instructor's industry experience in years	7.3	9.9
% of students with industry experience	22.0%	29.1%
Project size in SLOC	4060	8071
Size of class	41	34
Size of team	5	5
Course length in months	4.2	5
Course is required	66.7%	92.9%

Table 6. Averages for level-1 and non-level-1 courses.

VI. Conclusions

As suggested by the results, the sample software engineering courses lagged behind their industry counterparts in terms of the maturity levels achieved. The survey results indicate that academic projects suffer from many of the same "schedule pressure" and "not relevant" issues as industry projects with most instructors and students unwilling to invest the time in more mature practices. This is unfortunate, however, since the academic projects provide the ideal opportunity for students to learn and appreciate the value of good development practices. We hope that this survey and its results will both guide and motivate instructors to improve their development processes utilized in software engineering courses as they move up the maturity levels. We have already received e-mail from some respondents that this is indeed happening.

References

1. Paulk, M.C., et al., "Capability Maturity Model for Software, Version 1.1," Software Engineering Institute, CMU/SEI-93-TR-024, February, 1993.

2. Paulk, M.C., et al., "Key Practices of the Capability Maturity Model, Version 1.1," Software Engineering Institute, CMU/SEI-93-TR-025, February, 1993.

3. "SW-CMM V2.0 Draft C," Software Engineering Institute, http://www.sei.cmu.edu/cmm/draft-c/c.html, October, 1997.

4. Collofello, J., et al., "Assessing the Software Process Maturity of Software Engineering Courses," *Proceedings of the Twenty-Fifth SIGCSE Technical Symposium on Computer Science Education*, 1994, pp. 16–20.

5. Zubrow, D., et al., "Maturity Questionnaire," Software Engineering Institute, CMU/SEI-94-SR-7, June, 1994.

6. "Software Engineering Education Directory," Software Engineering Institute, CMU/SEI-91-TR-9, May, 1991.

7. Beckman, K., "Directory of Industry and University Collaborations with a Focus on Software Engineering Education and Training, Version 6," Software Engineering Institute, CMU/SEI-97-SR-018, November, 1997.

8. "Process Maturity Profile of the Software Community," Software Engineering Institute, http://www.sei.cmu.edu/activities/sema/pdf/1998dec.pdf, December, 1998.