Students Managing the Software Development Process: A Meta-Level Retrospective Evaluation.

Anne Comer and Helen M. Edwards
School of Computing and Technology
University of Sunderland, UK
{anne.comer, helen.edwards}@sunderland.ac.uk

Abstract

A fifteen week module on software engineering management exposed graduate students to practical aspects of the management of software development: emphasizing the recording and analysis of factual data during a group project. Data generated within the six projects are analyzed to reveal the extent to which the students demonstrated the required management and monitoring competencies. These results showed the student groups on a spectrum from full competence to no evidence of competence. The most interesting results were for the three groups in the "partially competent" range: and it is these that are discussed in most detail. The group work was retrospectively evaluated against SWEBOK knowledge areas to analyze the extent to which they were covered. Finally the issues that need to be considered in an explicit use of SWEBOK, and the need for support from automated data collection and analysis are discussed.

1. Introduction

Since 1999 we have run a one-year full time MSc in Software Engineering at the University of Sunderland. In 2004 it was reviewed and as a result a new module was introduced "Software Engineering Evolution and Management" since it was perceived that, although many technically demanding topics were covered in the course, there had been little emphasis on the project management and team working aspects of software engineering. The overriding aim of this module was to ensure that the students adopted an experiential and reflective learning approach to the topic area. To enable this, a major feature of the module was the requirement to work upon a specific project within a team. There was nothing novel in this idea, as many previous papers in the CSEE&T conference proceedings attest (for example [1-5]). However, the primary objective for the teams was the delivery of a software product (as always) as the result of a visible, and documented, appropriate project management process (both at the team and personal level). At an individual level we dictated that the students needed to evaluate the product and process in a rigorous, objective manner based on factual data that had been recorded during the project lifespan. Through these mechanisms we sought to use the module to hone and synthesize the software engineering competencies that we believe each masters student should be able to demonstrate on graduating from the course. These competencies can be summarized as the ability to chose and use effectively: software development approaches, configuration management, project monitoring and measurement, project management (at a personal and team level), evaluation, testing, and post-mortem strategies.

The evidence that can be used to assess our success in realizing our aims, and the extent to which we can use the data to judge the students' success is considered in this paper. The analysis of evidence was undertaken some three months after the module ran (and the marking of the assignment had been completed). One driver for this analysis was "process improvement" for the module, in that we wished to reflect on the experience before running

Proceedings of the 19th Conference on Software Engineering Education & Training (CSEET'06)
1093-0175/06 $20.00 © 2006 IEEE

the module for a second time. We knew that there had been an implicit mapping between the activities in the module and the SWEBOK [6], we intended to explore what that had revealed in practice and to consider how to make this more explicit in the subsequent run of the module.

To provide the context for this assignment the characteristics of the student cohort and the module operation are outlined. The core competencies that were to be demonstrated are given. The types of data that were gathered are then identified and the decisions made in order to synthesize and summarize the data presented here are explained and justified. The synthesized data is then used to evaluate the achievement of the students in a practical software engineering team project and compared against the SWEBOK [6]. Finally, the key issues arising from this retrospective evaluation are delineated.

2. The Operational Context

The module ran from February to June 2005, with 48 class contact hours (150 learning hours in total). The class cohort size was 22 students, divided into six teams, all students had first degrees in computing and competency in object oriented development (requirements for entry onto the MSc course). There were 17 men and 5 women. In terms of nationality there was a wide range of students from the UK, northern and southern Europe, the Indian Sub-continent, Arabic states, and the Far East. The students had encountered different pedagogical styles: from rote learning and acceptance of the lecturer's statements, to student-centered and experiential learning. In delivering the module we used conventional lectures for "content delivery" (eleven session of between one and two hours length): tutorial sessions were used as project workshops. In addition to the class contact sessions students were expected to meet as a team as well as work on their individual activities.

The teams were created by the module tutors: these teams were not engineered in terms of group working or skill-level considerations. The primary aim of the assignment was not to evaluate how successfully the students could produce a piece of software, but to have them experience team work in developing a software project (in terms of its process and management). Thus the major requirement was for them to gather quantitative data about their project, during its execution, and to reflect upon it in an objective manner against the factual data thus acquired. In this context, engineering a team for group working abilities or skills-levels was unnecessary: both in the software development and educational context. Moreover, such team formation perhaps more accurately reflects the industry experience, where project managers have to work with whom they have – not whom they would like to have.

Each team was provided with the information in the project document as given in Figure 1. This text explicitly requires a visible, and documented, managed process throughout the project. For this run of the module the deliverable was a simple estimation model, deriving effort and time from function points. This was to use Albrecht's Function Points, (version 1) and Boehm's COCOMO '81 models [7].

Once the project had been issued and individuals allocated to teams the management and progress of the each project was left to each team. However, to provide some initial structure a template was provided for what should be recorded in team meeting minutes (see Figure 2).

In addition an MS-Excel template was provided for use in creating and monitoring individual activity logs (see Figure 3). These captured task specific data including links to group tasks, start and end dates, and duration (in minutes).

Project – Implementing A Small Software Product

Objective: To define the requirements, specify, design, develop, test, manage and document a small software product for a 'client'.

Elaboration: You are expected to meet the client's requirements for a software product. You must work within a documented development process (plans) and record both team and personal activities (records). A priority is visibility of decision making, of activity and of records.

You may develop the model by whatever process, and in whatever language, you decide – your decisions must be discussed in your final evaluation. Responsibility and activity must be allocated clearly and equally – note, all work products (specification, design, development and test products) must be independently reviewed/tested.

You must hold weekly group meetings during the module tutorial sessions, and you will be expected to informally present a review of progress at regular intervals. A project portfolio (incorporating all project documentation and records), and individual portfolios, must be kept up to date. They will be reviewed, and feedback given, at regular intervals.

Project, and individual time and task management is a priority. It is expected that the management, the development, portfolios and project evaluation will be completed within a project total of 160 hours – the client is 'paying' 160 hours only. At the 160 hours milestone a forma1 project review with the client must be held; the project may either be terminated at this stage or be extended by a maximum of a further 40 hours. Termination may be ordered earlier than 160 hours, when the client has expressed dissatisfaction with product progress at two or more earlier reviews.

Upon completion or termination of the project the group must (1) include a project evaluation within the project portfolio, (2) submit the product and the portfolios (both project and individual portfolios).

Figure 1: Group project contract

Project Monitoring
- Task allocation since last meeting
- Progress on each task since last meeting
- Time on each task since last meeting: actual vs predicted.
- Deliverables "delivered" since last meeting: actual vs predicted.
- Deliverables expected for next meeting.
- Rework items.
- Task allocation until next meeting.
- AOB.

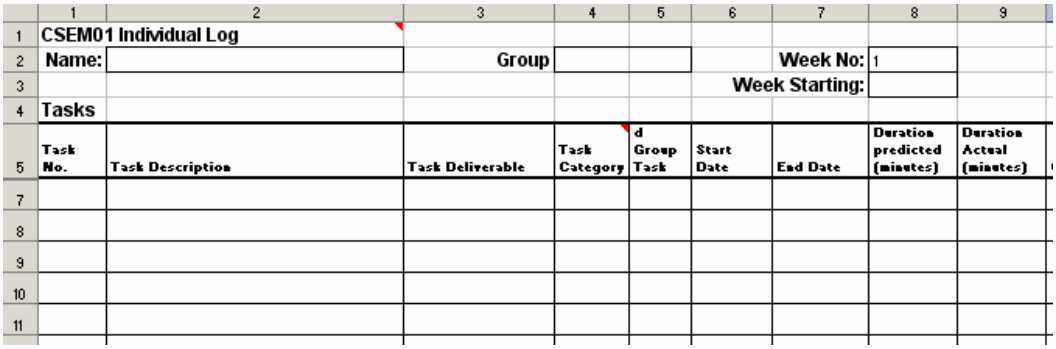
Group Project Decisions
- Are all tasks clearly defined until next meeting?
- Are all tasks clearly allocated until next meeting?
- Are all deliverables defined?
- AO questions?

Project Evaluation
- Rate of progress: by group, by individual.
- Communication with the client.
- Demonstration to, or "validation" involving client.

AOB

Figure 2: Template for team meeting minutes

Tasks were also expected to be categorized as: 1: Planning, 2: Research, 3: Requirements and Specification, 4: Design, 5: Coding, 6: V&V, 7: Project management and administration. Finally, there was a column (not shown) for recording comments, suggested comment types identified: clarity of task, rate of progress, communication with different stakeholders, task dependencies, and task evaluation (e.g. in PDCA/ISO9001 context).



	1	2	3	4	5	6	7	8	9
1	CSEM01 Individual Log								
2	Name:			Group				Week No:	1
3								Week Starting:	
4	Tasks								
5	Task No.	Task Description	Task Deliverable	Task Category	d Group Task	Start Date	End Date	Duration predicted (minutes)	Duration Actual (minutes)
7									
8									
9									
10									
11									

Figure 3: Template for individual activity logs

These two document sets provide the minimum necessary objective data of the project to enable factual evaluation of the project and its outcomes. Teams were encouraged to include additional documentation to support their development activity and project evaluation.

3. Data Acquisition and Synthesis.

The students' team and individual portfolios were marked against the defined assessment criteria and produced individual student marks. However, it is not this assessment that is reported here, but a meta-level evaluation which we undertook three months after the module had been completed.

Within the project work we expected the teams to demonstrate the following specific competencies (these were a subset of the module's full competency set):

- Use of development methods (including testing)
- Project monitoring/measurement
- Project management at a team level
- Project management at a personal level
- Configuration management
- Completion of the project
- Product delivery
- Product quality (tested ness)
- Evaluation/post-mortem

Moreover, we also evaluated the portfolios' evidence and mapped these to the SWEBOK [6] knowledge areas (Kats) which are discussed further in section 4.

The process of analysis was as follows:

- First pass: a simple three point ordinal scale for competency evaluation was devised: fully, partially, or not demonstrated. Each team was rated on this scale by the authors by taking each competency in turn and examining the quantity and quality of the supporting evidence in the portfolio.
- Second pass: the rankings were refined by undertaking pair-wise comparisons of team portfolios for individual competencies. This led to a subdivision of each value into a five point scale, to show ordinal differentiation of the teams.

The resultant evaluation of the six teams (identified as; R, B, P, Y, G, and g) is shown in Table 1, using this 15-point scale.

The recorded data for two teams, G and g, showed little success and had no scope for further analysis. Moreover, teams either reached completion or did not. Therefore, these two teams and one competency were removed: at the same time, the order of

competencies was changed to group together management and monitoring competencies (see Table 2).

	demonstrated fully					partially									not
devt methods (inc. testing)					R	B	P	Y						G	g
proj. monitoring/measurement	R					Y	B	P					G		g
proj. management (team)		R				B		Y			P			G	g
proj. management (personal)		R				Y			B		P			G	g
config. management	R							Y	B				P	G	g
completion	R	B	P	Y	G										g
product delivery	R		YBP											G	g
product quality (testedness)	R	Y					B	P						G	g
evaluation/postmortem	R	Y			B				P					G	g

Table 1: Results of Six Groups vs. Competencies

	demonstrated fully					partially									not
proj. management (team)		R				B		Y			P				
proj. management (personal)		R				Y			B		P				
proj. monitoring/measurement	R					Y	B	P							
config. management	R							Y	B				P		
devt methods (inc. testing)					R	B	P	Y							
product delivery	R		YBP												
product quality (testedness)	R	Y					B	P							
evaluation/post-mortem	R	Y			B				P						

Table 2: As Table 1, Minus: 'not demonstrated' Groups and 'completion' competency.

This visual grouping was useful for our analysis. It clearly shows that one of the four teams, R, was able to fully demonstrate management and monitoring competencies. Whereas, another of the four teams, P, has significantly failed to demonstrate competence in three out of the four management and monitoring competencies.

We had expected that, given the focus of the module, the students would have achieved higher results than they did in the management and monitoring competencies: team R is the obvious exception to this. One explanation is that there may not have been enough time to absorb ideas from the module (it takes time for the penny to drop!). However, outside the scope of the module we have see 'feed-forward' in terms of good management practice within the individual masters (capstone) project – especially for those students from higher scoring teams. It seems it does take time, but the penny does drop.

The weakest management and monitoring competency of the teams was configuration management. This was disappointing and surprising since the module specifically dedicated lecture time to change and configuration management. In practice, despite prompting from the tutors, these concepts seem to have been poorly understood, and teams failed to perform basic configuration management practices.

The strongest and most consistent management and monitoring competency was monitoring/measurement. We assign this to the insistence of the module tutors on the rigorous use of team meeting minutes and individual activity logs, as discussed in section 2. Maybe standards pay-off.

It might be considered that performance against project management, at the team level, might have been influenced by the composition of the team or the team size. However, the

composition was determined randomly and no correlation between size and performance can be seen. in the same order as the assessed competence in (team) project management, team sizes were R:4,B:5,Y:2 and P:4.

Beyond the management and monitoring competencies, what can be said? With respect to product delivery, it was hard to distinguish between the teams (again, with the exception of team R); the three remaining teams all delivered functional and easily usable products.

Three competencies, the bottom of Table 2, stand out from the remaining five: product delivery, product quality (testedness), and evaluation/postmortem. Why? Throughout the duration of the project the teams had been reminded of the assessment parameters for the project – particularly, *"visibility of decision making, of activity and of records"*. Nevertheless, teams consistently discussed the "final product" in meetings with the 'client' (module tutors). From the weekly meetings between the 'client' and each team, it was clear that all teams (with the exception of team R) remained fixed upon delivering a product before turning their attention to product quality and evaluation/postmortem. The end product - whether the software, its quality, or the evaluation/postmortem – seem to matter to the teams more than the process of arriving there.

4. Evaluation of the Group Work Project in the Context of SWEBOK

The module tutors expected that the approach used within the module would provide an opportunity for the students to synthesize software engineering concepts: both those for which they had pre-existing knowledge, and those to which they were introduced during the module. Following the first run of the module, hindsight showed that we had made an two implicit assumptions (i) that these concepts would map onto SWEBOK [6] and, (ii) since the students were computing graduates they would have some experience of, and hence some competence in, the majority of areas in SWEBOK: although not the depth of experience and knowledge expected of the intended SWEBOK audience (graduates with 4 years relevant work experience).

We compared the module competencies and those from SWEBOK: furthermore, we examined the minimum evidence sets to establish a match with the module competencies. Table 3 shows this mapping (where appropriate, to highlight specific concepts the SWEBOK mapping is show at the level of sub areas). Table 3 highlights a good match against the SWEBOK areas. There are two areas where there is an incomplete match: firstly "software maintenance" which was not applicable within the group work, and secondly "software engineering methods and tools" which again, from the group work point of view was not considered, however, apart from the personal logs, each student was required in their individual evaluation of the project to evaluate this aspect.

Has the mapping against SWEBOK been valuable? From a module viewpoint we wanted the students to gain competence in certain topics (such as data gathering, data analysis). The focus of the module is on competencies, and SWEBOK is itself about defining competence (at "graduate + 4"), therefore, it is useful to map from the module to this existing structure allowing others in the software engineering community to understand the focus of this module. This should provide opportunities to make it repeatable and potentially comparable with other similar software engineering courses. If mapping SWEBOK areas to the group work activities is valuable then it would seem sensible to strengthen all the areas (including software maintenance). However, there

exists a counterbalancing argument that says that the depth contained in any one area of SWEBOK mitigates against a comprehensive mapping of SWEBOK to group projects (for evidence of this see [8]). Therefore, in practice, a trade-off needs to be made between breadth, such as in our module and depth of coverage (as in [9]). In our context the breadth of coverage in one module is useful as the depth in specific areas is covered in a number of additional modules within our masters course.

SWEBOK	**MODULE**		
Knowledge Areas (relevant *sub areas* highlighted)	**Expected Competencies** (from assessment criteria and module specification*)*	**Nature of Explicit Evidence (minimum required to be collected)** In personal logs	In group minutes
Software requirements including *Requirements Validation and Practical Considerations (…change management, requirements tracing. requirements measurement).*		Requirements & Specification. V&V. Project management and administration.	Communication with client? Demonstration to, or 'validation' involving, client?
Software design		Design	Deliverables
Software construction		Coding	Deliverables
Software testing	Product quality (testedness)	V&V	Rework items.
Software maintenance			
Software configuration management	Configuration management	Project management and administration.	
Software engineering management	Project management at (i) a team level, (ii) a personal level.		Project Monitoring
Software Project Planning including *Review and Evaluation,*	Evaluation and post-mortem.	Planning. V&V	Task allocation Time on each task (actual vs. predicted) Progress on each task Deliverables 'delivered' (actual vs. predicted).
Software engineering process Including *Process Implementation and Change. Process Definition. Process Assessment. Process and Product Measurements.*	Use of development methods (including testing) Project monitoring and measurement		Progress on each task Deliverables 'delivered' (actual vs. predicted). Project Monitoring. Rate of progress, by group and individual.
Software engineering tools and methods	Use of development methods (including testing)		
Software quality Including *Software Quality Management Processes, Software Quality Assurance, Verification And Validation, Reviews and Audits*	Product delivery. Evaluation and post-mortem.	Project management and administration V&V	Rework items. Demonstration to, or 'validation' involving, client.

Table 3: Mapping of SWEBOK areas to Competencies and Evidence

5. Conclusions

So what have we learnt? The lessons can be summarized under the headings of: SWEBOK, and data collection and analysis.

The SWEBOK knowledge areas (KAs) can provide a suitable framework for assessing student competencies in the module. However if SWEBOK is to be used then we need to inform the students about SWEBOK: its usage and purpose. We could also endeavor to assess not only module outcomes (for institutional purposes) but also "added-value" (for educational learning). In this case entry-level and exit–level competencies would need to be explicitly and rigorously evaluated: requiring the development of an appropriate assessment instrument. Finally, the inclusion of the missing KA of "maintenance" needs to be considered. This can be addressed simply by setting a maintenance style project. However, it will be important to ensure the scope of such a project is manageable within the time frame, scoping is easier to achieve for new developments.

In terms of data collection and analysis it is apparent that both aspects are time-consuming, even with small numbers of teams. However, this aspect is amenable to automation. Thus a software tool for recording and analyzing the contents of meeting minutes and logs has been developed and is currently being evaluated. This aspect also offers scope for more precise data collection and analysis methods, to give explicit correlations with (i) software engineering (project management) competencies and (ii) SWEBOK KAs.

7. References

[1] O.P. Brereton, S. Lees, R. Bedson, C. Boldyreff, S. Drummond, P. Layzell, L. Macaulay, R. Young. "Student Collaboration across Universities: A Case Study in Software Engineering,", Thirteenth Conference on Software Engineering Education & Training, p. 76, 2000.

[2] Maria Isabel Alfonso, Francisco Mora. "Learning Software Engineering with Group Work," 16th Conference on Software Engineering Education and Training (CSEE&T 2003), p. 309, 2003.

[3] A. Inkeri Verkamo, Juha Taina, Turjo Tuohiniemi, Yury Bogoyavlenskiy, Dimitry Korzun. "Distributed Cross-Cultural Student Software Project: A Case Study,", 18th Conference on Software Engineering Education & Training (CSEET'05), pp. 207-214, 2005.

[4] Guttorm Sindre, Tor Stalhane, Gunnar Brataas, Reidar Conradi. "The Cross-Course Software Engineering Project at the NTNU: Four Years of Experience,",16th Conference on Software Engineering Education and Training (CSEE&T 2003), p. 251, 2003.

[5] Ivica Crnkovic, Rikard Land, Andreas Sjogren. "Is Software Engineering Training Enough for Software Engineers?," cseet, p. 140, 16th Conference on Software Engineering Education and Training (CSEE&T 2003), 2003.

[6] A. Abran, J.W. Moore (Executive Editors), P. Bourque and R. Dupuis (Editors) Guide to the Software Engineering Body of Knowledge: 2004 Edition. IEEE Computer Society Press, April 2005. ISBN 0-7695-2330-7.

[7] Fenton, N.& S. L. Pfleeger. Software Metrics: A Rigorous and Practical Approach, Second Edition. International Thomson Computer Press, 1996. pp 258-265, 415, 438-441.

[8] M. Daniels, X. Faulkner, I. Newman, "Open Ended Group Projects, Motivating Students and Preparing them for the 'Real World'".Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET.02).

[9] S. Ramakrishnan. Student Handbook, Software Engineering Studio Project CSE4002. School of Computer Science and Software Engineering, Monash University, Australia, 2004. available from http://www.csse.monash.edu.au/courseware/cse4002/handbook/handbook.pdf.