Using a Mini-Project to Identify Intervention Criteria for a Capstone Project

Stacey Omeleze, Vreda Pieterse and Fritz Solms
Dept. of Computer Science
University of Pretoria
South Africa
vpieterse@cs.up.ac.za

ABSTRACT

In order to prepare computer science students for their six-month capstone industry project, we have introduced an intense six-week mini-project prior to the capstone project. The mini-project exposes students to many of the best practices and challenges for industrial software engineering projects. We have found that an analysis of the team dynamics, activities, peer reviews and self-assessments done during the mini-project can reveal information which can be used to identify intervention indicators for the capstone project, i.e. indicators which predict problems teams may be likely to experience during the execution of their capstone project. It is hoped that identifying intervention indicators may assist to trigger early intervention in order to improve the pedagogical value of the course and reduce the failure rate experienced in capstone projects. This paper reports on the identification of intervention criteria and the correlation of these criteria with pathological aspects observed in subsequent the capstone project.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Programming Teams*; K.3.2 [**Computers and Education**]: Computers and Information Science Education; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management—*Staffing, Training*

General Terms

Management, Human Factors

Keywords

Teaching software engineering, Teaching Design and Integration, Teaching Teamwork, Soft Skills

1. INTRODUCTION

In order to prepare students for the work environment, it is customary for final year computer science a and software engineering students to complete an industrial scale software development project [9, 11]. We have, however, found that students lack essential understanding of software engineering practices and tools as well as experience to work in teams leading to an excessively high failure rate [].

In order to improve the pedagogical value of the final year software engineering module and reduce the probability of failure, we have inserted an intense six-week mini-project before the six-month capstone project[**?**, 5]. The mini-project aims to introduce the use of configuration management and testing practices and tools, expose students to the complexities of team dynamics and the challenges around integrating independently developed modules into a single product. The latter is an area which is not typically included in the scope of a capstone project. To increase the experience students acquire in team dynamics, different teams with different challenges are formed for each phase of the project[**?**].

Within the mini-project, students are aimed to retrospectively analyze their strengths and weaknesses in both, technical domains and soft-skills[11]. The latter is aimed to assist them when forming complementing teams for the capstone project. We have found that it is valuable to relax the aim of mini-project success in favour of optimizing the skills and insights acquired during the mini-project. In addition to assigning group marks for artifacts produced during the mini-project, students are also continuously assessed through peer reviews and self assessments [3]. Individual contribution marks are calculated from an assessment of the contributions to the version control system and from peer reviews [6].

For the main project teams students form new teams of three students each. They submit tenders for projects with actual clients from either industry or academia, specifying the order of preference they have for those projects. Clients then select the teams they would prefer for their project in order of preference. Project allocations are based on both, team and client preferences. Teams are expected to follow a defined software development process of their choice in order to elicit the detailed requirements from the client and deliver the product to the client.

In the mean time the assessments (including peer reviews and self-assessments) are analyzed for interception indicators in order to proactively identify potential pathological teams which can be expected to experience difficulties during their project execution. In future these interception indi-

cators will be used to proactively intervene in teams which are identified as potential pathological teams in order to make them aware of the challenges they might be facing and to assist them with introducing mitigating measures. This is expected to not only reduce the failure rate in capstone projects, but also to increase the pedagogical value of the course. This paper, however, focuses on identifying correlations between interception indicators deduced from mini-project observations and pathological aspects.

2. RELATED WORK

- [5] introduces 5'th semester mini-project with the aim to address
  - majority of cap-stone projects ended up with poor designs and no testing (all focus on churning out code)
  - high-failure rate and
  - difficulty in guiding many different cap-stone projects with focus on how it is done, not what is done, significantly improved student's understanding, project management (e.g. the number of capstone projects which were completed in time), and the quality of the capstone projects. Also found that ability to work in teams is improved.

  Introducing 5'th semester mini-project

- [7] introduce a software engineering mini project as part of a first year breadth-first introduction to computer science. Gave benefits to teams who completed task successfully and where all team members contributed in some major documented way. used time sheets. Teams of 25 including team manager, team architect, syadmin and team librarian (maintain and integrate code). Customer available to answer questions.

- Tadayon [13] discusses using a semi-realistic industrial SE project to teach SE. XP with peer programming.

- Basholli et al. [1] discusses peer assessment within group SE projects together with validation through questionaires as well as student feedback on the assessment. Clark et al. [3] also discuss assessment through peer reviews and self-assessment, individual contribution reports, and quantitative contribution assessment of group members by group members. Hayes et al. [6] validate./aigment this with pop questionaires for individuals about the project details.

- Collofello and Hart [4] discuss monitoring team progress and individual contributions through regular weekly progress reports, team meeting reports, frequent deliverables, and metrics.

- Yu and Zhang [16] show that using a failure case study is an effective way of teaching for a computer networking course (need better references for teaching through failure).

- Madsen and Desai [8] argue organizations learn more effectively from failue than from success

- Varol and Bayrak [15] discuss the necessity of includinf a real-life client-sponsored project in a software engineering curriculum, particularly to covey the importance of SRS, analysis and design.

- Stein [12] discusses using large versus small groups in pedagogical SE projects

- van der Duim, Andersson and Sinnema [14] discuss some good practices for educational software engineering projects including the use of reciprocity (giving and takling) and cooperation amongst students, encourage contacts between students and faculty, providing prompt feedback, emphasizing time on task, active learning, communicating high expectations, and respecting diverse talents and different ways of learning (not a very useful reference, but might still be worth it)

- [2] discuss expanding software engineering project to include the wider business context and business case.

- [10] perform defect analysis → pattern identification → improvement definition as pedagogical process.

3. THE METHOD

Write introduction paragraphs

3.1 Mini-project design

Discuss design of mini-project here. This should be compact with reference to Vreda's previous papers. Mention waterfall, the constructed team challenges and also that only in the mini-project teams are challenged to have to integrate modules developed by other teams.

3.2 The design of the capstone project

Write section:

3.3 Intervention indicators, pathological project symptoms

We identified a range of information sources from which we could extract information about the individual personalities and their performance during the mini-project. We then extracted measures from these sources which could potentially contribute to intervention indicators which represent some probability of the capstone project experiencing pathological symptoms.

Note that the intervention indicators are not the measures. For example, measuring 3 diligent isolates in a team could be an intervention indicator. We need to think about this . . .

Similarly we identified a set observed pathological symptoms of the capstone project. This information was also obtained from a variety of sources.

We then went ahead and performed a simple correlation analysis between each of the potential indicators and each of the pathological symptoms.

Waffle a bit more ...

We used a variety of sources to extract information from the mini-project which could potentially be used as intervention indicators for the capstone project. The sources include peer reviews, self-assessments, historical academic records, enrollment records, the version control system (git) and the teaching assistant who observed and assisted the teams. From this we were able to extract the following measures:

- the academic strength (from the academic records),
- the work load the student had in the current academic year (from the enrollment records),
- the level of involvement (from the git repository analysis, the peer reviews and the self-assessments),
- the quality of work (from the git repository analysis, the peer reviews and the self-assessments),
- personality traits of the student (from peer reviews, self-assessments and observations made by the teaching assistant)

For each of the above measures we need to provide a little detail on how the corresponding value in the spreadsheet determined.

3.4 Pathological project symptoms

In order to identify pathological project symptoms we used various assessments, information obtained from the teaching assistant and client as well as self-assessments and peer reviews. The assessments included documentation assessments, code reviews, demo assessments, assessments made by the department's lecturers during the annual project day and client assessments.

Very valuable information was obtained by integrating the teaching assistant into the various capstone teams. To this end, the teaching assistant requested to be invited to with each team's chat group, took part in team meetings (e.g. scrum meetings), and had extensive discussions with individual team members,

Add any other mentioning of your hard work, Stacey

.

This enabled us to identify the following pathological symptoms in capstone projects

- teams starting late with their capstone project (observed from the git repositories),
- team members working independently (observed by teaching assistant and peer reviews),

  Not sure about this, but if we do not currently ask how often they work together, we should do so next year.

  ,
- certain team members slacking, i.e. contributing only very little to the project (observed from git repositories and peer reviews),
- certain members taking over key elements of the project to the exclusion of others,

  I am sure this can be formulated better.
- teams do not follow a defined software development process (observed from documentation assessments and observations made by teaching assistant),
- low quality products (observed from demos, code reviews, continuous assessments, project day assessments and client feedback),
- weak quality assurance including weak or no unit testing or integration testing, no peer reviews, ... (observed from git repository reviews and discussions during demos),
- weak communication with the client (observed from peer reviews, client feedback and information obtained from group by teaching assistant),
- team tensions (observed from peer reviews and observations made by teaching assistant).

Do we want to state that these symptoms are simply based on past experience or what justification do we want to provide for these symptoms?

3.5 Intervention indicators

This section is the most difficult and critical. How are we constructing intervention indicators from the measures. Hmmmm ...

4. RESULTS

Write section:

5. CONCLUSIONS AND FUTURE WORK

Write section:

6. REFERENCES

[1] A. Basholli, F. Baxhaku, D. Dranidis, and T. Hatziapostolou. Fair assessment in software engineering capstone projects. In *Proceedings of the 6th Balkan Conference in Informatics*, BCI '13, pages 244–250, New York, NY, USA, 2013. ACM.

[2] J. Bolinger, K. Yackovich, R. Ramnath, J. Ramanathan, and N. Soundarajan. From student to teacher: Transforming industry sponsored student projects into relevant, engaging, and practical curricular materials. In *Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments, 2010 IEEE*, pages 1–21, April 2010.

[3] N. Clark, P. Davies, and R. Skeers. Self and peer assessment in software engineering projects. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*, ACE '05, pages 91–100, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[4] J. Collofello and M. Hart. Monitoring team progress in a software engineering project class. In *Frontiers in Education Conference, 1999. FIE '99. 29th Annual*, volume 1, pages 11B4/7–11B410 vol.1, Nov 1999.

[5] P. Desai, G. H. Joshi, and M. Vijayalaskhmi. A novel approach to carrying out mini project in computer science amp; engineering. In *Engineering Education: Innovative Practices and Future Trends (AICERA), 2012 IEEE International Conference on*, pages 1–4, July 2012.

[6] J. H. Hayes, T. C. Lethbridge, and D. Port. Evaluating individual contribution toward group software engineering projects. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 622–627, Washington, DC, USA, 2003. IEEE Computer Society.

[7] P. C. Isaacson. A mini software engineering project for cs0. *J. Comput. Sci. Coll.*, 19(1):169–178, Oct. 2003.

[8] D. Madsen and V. Desai. Failing to learn? the effects of failure and success on organizational learning in the global orbital launch vehicle industry. *Academy of Management Journal*, 52:451–476, 2010.

[9] M. Oudshoorn and K. Maciunas. Experience with a project-based approach to teaching software engineering. In *Software Education Conference, 1994. Proceedings.*, pages 220–225, Nov 1994.

[10] D. Peixoto, V. Batista, R. Resende, and C. PaÌĄdua. Learning from students' mistakes in software engineering courses. In *Frontiers in Education Conference (FIE), 2010 IEEE*, pages F1J–1–F1J–6, Oct 2010.

[11] S. Roach. Retrospectives in a software engineering project course: Getting students to get the most from a project experience. In *Software Engineering Education and Training (CSEE T), 2011 24th IEEE-CS Conference on*, pages 467–471, May 2011.

[12] M. V. Stein. Using large vs. small group projects in capstone and software engineering courses. *J. Comput. Sci. Coll.*, 17(4):1–6, Mar. 2002.

[13] N. Tadayon. Software engineering based on the team software process with a real world project. *J. Comput. Sci. Coll.*, 19(4):133–142, Apr. 2004.

[14] L. van der Duim and J. Andersson. Good practices for educational software engineering projects. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, pages 698–707, May 2007.

[15] C. Varol and C. Bayrak. Applied software engineering education. In *Information Technology Based Higher Education and Training, 2005. ITHET 2005. 6th International Conference on*, pages T3C/25–T3C/29, July 2005.

[16] L. Yu and W. Zhang. Failure case study: An instructive method for teaching computer network engineering. In *Computer Science and Education (ICCSE), 2010 5th International Conference on*, pages 296–299, Aug 2010.