Relating Student, Teacher and Third-Party Assessments in a Bachelor Capstone Project

Vincent Ribaud, Vincent Leilde

Relating Student, Teacher and Third-Party Assessments in a Bachelor Capstone Project (short paper)

Vincent Ribaud[1][0000-0002-5710-6469] and Vincent Leilde[2][0000-0002-4153-4012]

[1] Université de Brest, Lab-STICC, team MOCS, Brest, France
[2] ENSTA Bretagne, Lab-STICC, team MOCS, 3 rue François Verny, Brest, France
ribaud@univ-brest.fr

Abstract. The capstone is arguably the most important course in any engineering program because it provides a culminating experience and is often the only course intended to develop non-technical, but essential skills. In a software development, the capstone runs from requirements to qualification testing. Indeed, the project progress is sustained by software processes. This paper yields different settings where students, teachers and third-party assessors performed [self-] assessment and the paper analyses corresponding correlation coefficients. The paper presents also some aspects of the bachelor capstone. A research question aims to seek if an external process assessment can be replaced or completed with students' self-assessment. Our initial findings were presented at the International Workshop on Software Process Education Training and Professionalism (IWSPETP) 2015 in Gothenburg, Sweden and we aimed to improve the assessment using teacher and third-party assessments. Revised findings show that, if they are related to curriculum topics, students and teacher assessments are correlated but that external assessment is not suitable in an academic context.

Keywords: process assessment, competencies model, capstone project.

1 Introduction

Project experience for graduates of computer science programs has the following characteristic in the ACM Computer Science Curricula [1]: "*To ensure that graduates can successfully apply the knowledge they have gained, all graduates of computer science programs should have been involved in at least one substantial project. […] Such projects should challenge students by being integrative, requiring evaluation of potential solutions, and requiring work on a larger scale than typical course projects. Students should have opportunities to develop their interpersonal communication skills as part of their project experience.*" The capstone is arguably the most important course in any engineering program because it provides a culminating experience and is often the only course used to develop non-technical, but essential skills [2]. Many programs run capstone projects in different settings [3-8]. The capstone project is intended to provide students with a learning by doing approach about software development, from requirements to qualification testing. Indeed, the project progress is sustained by software processes.

Within the ISO/IEC 15504 series and the ISO/IEC 330xx family of standards, process assessment is used for process improvement and/or process capability determination. Process assessment helps students to be conscious about and improve what they are doing. Hence, a capstone teacher's activity is to assist students with appreciation and guidance, a task that relies on the assessment of students' practices and students' products. This paper yields different settings where students, teachers and third-party assessors performed [self-] assessment and analyses correlation coefficients. Incidentally, the paper presents some aspects of the bachelor capstone project at Brest University. Data collection started 3 years ago. Initial findings were presented in [9].

The paper structure is: section 2 overviews process assessment, section 3 presents different settings we carried process assessments; we finish with a conclusion.

2 Process assessment

2.1 Process Reference Models

Most software engineering educators will agree that the main goal of the capstone project is to learn by doing a simplified cycle of software development through a somewhat realistic project. For instance, Dascalu et al. use a "streamlined" version of a traditional software development process [3]. Umphress et al. state that using software processes in the classroom helps in three ways: 1 - processes describe the tasks that students must accomplish to build software; 2 - processes can give the instructor visibility into the project; 3 - processes can provide continuity and corporate memory across academic terms [4]. Consequently, the exposition to some kind of process assessment is considered as a side-effect goal of the capstone project. It is a conventional assertion that assessment drives learning [10]; hence process assessment drives processes learning. Conventionally, a process is seen as a set of activities or tasks, converting inputs into outputs [11]. This definition is not suited for process assessment. Rout states that "*it is of more value to explore the purpose for which the process is employed. Implementing a process results in the achievement of a number of observable outcomes, which together demonstrate achievement of the process purpose* [12]." This approach is used to specify processes in a Process Reference Model (PRM). We use a small subset of the ISO/IEC 15504-5:2012 Exemplar Process Assessment Model that includes a PRM (replicated from the ISO/IEC 12207:2008), mainly the Software Processes of the ENG Process Group [13]: ENG.3 System architectural design, ENG.4 Software requirements analysis, ENG.5 Software design, ENG.6 Software construction, ENG.7 Software integration, ENG.8 Software testing. Process Purpose, Process Outcomes, Base Practices (BP) have been kept without any modification; Input and Outputs Work Products (WP) have been set to main products.

2.2 Ability model

From an individual perspective, the ISO/IEC 15504 Exemplar Process Assessment Model (PAM) is seen as a competencies model related to the knowledge, skills and attitudes involved in a software project. A competencies model defines and organizes

the elements of a curriculum (or a professional baseline) and their relationships. During the capstone project, all the students use the model and self-assess their progress.
A hierarchical model is easy to manage and use. We kept the hierarchical decomposition issued from the ISO/IEC 15504 Exemplar PAM: process groups – process – base practices and products. A competency model is decomposed into competency areas (mapping to process groups); each area corresponding to one of the main division of the profession or of a curriculum. Each area organizes the competencies into families (mapping to processes). A family corresponds to main activities of the area. Each family is made of a set of knowledge and abilities (mapping to base practices), called competencies; each of these entities is represented by a designation and a description. The ability model and its associated tool eCompas have been presented in [14].

2.3 Process assessment

The technique of process assessment is essentially a measurement activity. Within ISO/IEC 15504, process assessment has been applied to a characteristic termed process capability, defined as "*a characterization of the ability of a process to meet current or projected business goals*" [13]. It is now replaced in the 330xx family of standards by the larger concept of process quality, defined as "*ability of a process to satisfy stated and implied stakeholders needs when used in a specific context* [15]. In ISO/IEC 33020:2015, process capability is defined on a six point ordinal scale that enables capability to be assessed from the bottom of the scale, *Incomplete*, through the top end of the scale, *Innovating* [16]. We see Capability Level 1, *Performed*, as an achievement: through the performance of necessary actions and the presence of appropriate input and output work products, the process achieves its process purpose and outcomes. Hence, Capability Level 1 will be the goal and the assessment focus.
If students are able to perform a process, it denotes a successful learning of software processes, and teachers' assessments rate this capability. Because we believe that learning is sustained by continuous, self-directed assessment, done by teachers or a third-party, the research question aims to state how students' self-assessment and teacher's assessment are correlated and if self-assessment of BPs and WPs is an alternative to external assessment about ISO/IEC 15504 Capability Level 1. Obviously, the main goal of assessment is students' ability to perform the selected processes set.

3 The Capstone Project

3.1 Overview

Schedule

The curriculum is a 3-year Bachelor of Computer Science. The project is performed during two periods. The first period is dispatched all the semester along and homework is required. The second period (2 weeks) happens after the final exams and before students' internship. Students are familiar with the Author-Reader cycle: each deliverable can be reviewed as much as needed by the teacher that provides students with comments and suggestions. It is called Continuous Assessment in [5, 6].

System Architecture

The system is made of 2 sub-systems: PocketAgenda (PA) for address books and agenda management and interface with a central directory; WhoIsWho (WIW) for managing the directory and a social network. PocketAgenda is implemented with Java, JSF relying on an Oracle RDBMS. WhoIsWho is implemented in Java using a RDBMS. Both sub-systems communicate with a protocol to establish using UDP. The system is delivered in 3 batches. Batch 0 established and analyzed requirements. Batch 1 performed collaborative architectural design, separate client and server development, integration. Batch 2 is focused on information system development.

Students consent

Students were advised that they can freely participate to the experiment described in this paper. The class contains 29 students, all agreed to participate; 4 did not complete the project and do not take part to the study. Students have to regularly update the competencies model consisting in the ENG process group, the 6 processes above and their Base Practices and main Work Products and self-assess on an achievement scale: Not - Partially - Largely - Full. There will be also teacher and third-party assessments that will be anonymously joined to self-assessments by volunteer students.

3.2 Batch 0 : writing and analyzing requirements

Batch 0 is intended to capture, write and manage requirements through use cases. It is a non-technical task not familiar to students. In [7], requirements are discussed as one of the four challenges for capstone projects. Students use an iterative process of writing and reviewing by the teacher. Usually, 3 cycles are required to achieve the task. Table 1 presents the correlation coefficient r between student and teacher assessment for the ENG.4 Software requirements analysis. It relies on 3 BPs and 2 WPs. Table 2 presents also the average assessment for each assessed item. The overall correlation coefficient relates 25 * 6 = 150 self-assessment measures with the corresponding teacher assessment measures, its value `r = 0.64` indicates a correlation.

Table 1: ENG.4 assessment (self and teacher)

	Stud. avg	*Tch. avg*	*r*
BP1: Specify software requirements	2.12	1.84	0.31
BP3: Develop criteria for software testing	1.76	1.76	1.00
BP4: Ensure consistency	1.92	0.88	0.29
17-8 Interface requirements	1.88	1.88	1.00
17-11 Software requirements	2.08	2.08	1.00

Thanks to the Author-Reader cycle, specification writing iterates several time during the semester and the final mark given to almost 17-8 Interface requirements and 17-11 Software requirement documents was Fully Achieved. Hence correlation between students and teacher assessments is complete. However, students mistake documents assessment for the BP1: Specify software requirements. Documents were improved through the author-reader cycle, but only reflective students improve their

practices accordingly. Also, students did not understand the ENG.4. BP4: Ensure consistency and failed the self-assessment. Most students did not take any interest in traceability and self-assessed at a much higher level that the teacher did.

A special set of values can bias a correlation coefficient; if we remove the BP4: Ensure consistency assessment, we get `r = 0.89`, indicating an effective correlation. However, a bias still exists because students are mostly self-assessing using the continuous feedback they got from the teacher during the Author-Reader cycle.

Students reported that they wrote use cases from a statement of work for the first time and that they could not have succeeded without the Author-Reader cycle.

3.3 Batch 1 : a client-server endeavor

For the batch 1, students have to work closely in pairs, to produce architectural design and interface specification and to integrate the client and server sub-systems, each sub-system being designed, developed and tested by one student. Defining the high-level architecture, producing the medium and low-level design are typical activities of the design phase [3]. 4 pairs failed to work together and split, consequently lonesome students worked alone and have to develop both sub-systems.

We were aware of two biases: 1 - students interpret the teacher's feedback to self-assess accordingly; 2 - relationship issues might prevent teachers to assess students to their effective level. Hence, for ENG.3 System architectural design process and ENG.7 Software integration process, in addition to teachers' assessment, another teacher, experienced in ISO/IEC 15504 assessments, acted as a third-party assessor.

Architectural design

For the ENG.3 System architectural design, table 2 presents the correlation coefficient between student and teacher assessments and the correlation coefficient between student and third-party assessments. Assessment relies on 3 BPs and 2 WPs. Table 2 presents also the average assessment for each assessed item. The correlation coefficient between self-assessment and teacher assessment measures is `r1 = 0.28` and the correlation coefficient between self-assessment and third-party assessment measures is `r2 = 0.24`. There is no real indication for a correlation.

Table 2: ENG.3 (self, teacher and third-party)

	Stud. avg	*Tch. avg*	*3-party avg.*	*r Std-Tch*	*r Std-3party*
BP1: Describe system architecture	2.24	2.02	1.68	-0.22	0.18
BP3. Define interfaces	1.96	2.16	1.56	0.48	0.36
BP4. Ensure consistency	2	1.72	0.88	0	0.44
04-01 Database design	2.48	2.2	1.88	0.49	0.35
04-04 High level design	2.12	1.84	1.64	0.37	-0.11

Detailed correlation is poor, except maybe for database design and interface design, but these technical topics are deeply addressed in the curriculum. An half of students perform a very superficial architectural work because they are eager to jump

to the code. They believe that the work is fair enough but teachers do not. The BP4. Ensure consistency is a traceability matter that suffers the same problem described above. A similar concern to requirements arose: most students took Work Products (Design Documents) assessment as an indication of their achievement.

Students reported that requirement analysis greatly helped to figure out the system behavior and facilitated the design phase and interface specification. However, students had never really learnt architectural design and interface between sub-systems, indeed it explains the low third-party assessment average for BPS and WPs.

Integration

ENG.7 Software integration is assessed with 6 main Base Practices and 2 Work Products. The correlation coefficient between self and teacher assessments is `r1 = -0.03` and the correlation coefficient between self and third-party assessments is `r2 = 0.31`. However, several BPs or WPs were assessed by the third-party assessor with the same mark for all students (N or P): the standard deviation is zero and the correlation coefficient is biased and was not used. Table 3 presents the assessment average for the third types of assessment.

Table 3: ENG.7 indicators

	Stud. avg	*Tch. avg*	*3-party avg.*
BP1: Develop software integration strategy	1.56	1.20	0.40
BP2: Develop tests for integrated software items	2.08	1.08	0.52
BP3: Integrate software item	2.00	2.12	1.76
BP4: Test integrated software items	2.00	1.80	1.16
BP5. Ensure consistency	1.76	1.20	0.72
BP6: Regression test integrated software items	1.64	0.52	0.2
08-10 Software integration test plan	1.44	0.88	0.00
11-01 Software product	2.04	2.12	1.48

All BPs and WPs related to integration and test are weakly third-party assessed, indicating that students are not really aware of these topics, a common hole in a Bachelor curriculum. Some students were aware of the poor maturity of the integrated product, partly due to the lack of testing. Although the Junit framework has been taught during the first semester, some students did not see the point to use it while some others did not see how to use it for the project. As mentioned by [4], we came to doubt the veracity of process data we collected. Students reported that they appreciated the high-level discipline that the capstone imposed, but they balked at the details.

3.4 Batch 2 : information system development

For the batch 2, students have to work loosely in pairs; each of the two has developed different components of the information system and has been assessed individually.

Table 4 presents the correlation coefficient `r` between student and teacher assessment for the ENG.6 Software construction process. It relies on 4 Base Practices and 2

Work Products. Table 4 presents also the average assessment for each assessed item. The correlation coefficient is `r = 0.10` and there is no indication for a correlation.

However, BPs and WPs related to unit testing were assessed by the teacher with almost the same mark for all students (N or P), biasing the correlation coefficient. If we remove BPs and WPs related to unit testing (17-14 Test cases specification; 15-10 Test incidents report; BP1: Develop unit verification procedures), we get `r = 0.49`, indicating a possible correlation.

Table 4: ENG.6 assessment (self and teacher)

	Stud. avg	*Tch. avg*	*r*
BP1: Develop unit verification procedures	1.84	0.40	0.05
BP2: Develop software units	1.92	1.84	0.37
BP3: Ensure consistency	1.92	0.92	0.25
BP4: Verify software units	1.96	1.00	-0.2
17-14 Test cases specification	1.80	0.36	0.07
15-10 Test incidents report	1.52	0.12	-0.45

Our bachelor students have little awareness of the importance of testing, including test specification and bugs reporting. This issue has been raised by professional tutors many times during the internships but no effective solution has been found until yet.

Students reported that the ENG.6 Software construction process raised a certain anxiety because students had doubt about their ability to develop a stand-alone server interoperating with a JDeveloper application and two databases but most students succeeded. For some students, a poor Java literacy compromised the project progress. It is one problem reported by Goold: the lack of technical skills in some teams [5].

4 Conclusion

The research question aims to see how students' self-assessment and external assessment [by a teacher or a third-party] are correlated. This is not true for topics not addressed in the curriculum or unknown by students. For well-known topics, assessments are correlated roughly for the half of the study population. It might indicate that in a professional setting, where employees are skilled for the required tasks, self-assessment might be a good replacement to external assessment.

Using a third-party assessment instead of coaches' assessment was not convincing. Third-party assessment is too harsh and tends to assess almost all students with the same mark. Self-knowledge or teacher's understanding tempers this rough assessment towards a finer appreciation.

The interest of a competencies model (process/BPs/WPs) is to supply a reference framework for doing the job. Software professionals may benefit from self-assessment using a competencies model in order to record abilities gained through different projects, to store annotations related to new skills, to establish snapshots in order to evaluate and recognize knowledge, skills and experience gained over long periods and in diverse contexts, including in non-formal and informal settings.

5 Acknowledgements

We thank all the students of the 2016-2017 final year of Bachelor in Computer Science for their agreement to participate to this study, and especially Maxens Manach and Killian Monot who collected and anonymized the assessments. We thank Laurence Duval, a teacher that coached and assessed half of the students during batch 1.

References

1. ACM: 2013 Computer Science Curricula - Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. http://www.acm.org/education/CS2013-final-report.pdf (last accessed 2017/16/6).
2. Capstone project. In S. Abbott (Ed.) The glossary of education reform. http://edglossary.org/capstone-project (last accessed 2016/23/3).
3. Dascalu, S. M., Varol, Y. L., Harris, F. C., Westphal, B. T.: Computer science capstone course senior projects: from project idea to prototype implementation. In Proceedings 35th Conference on Frontiers in Education, pp. S3J-1. IEEE, Indianapolis, USA (2005).
4. Umphress D. A., Hendrix T. D., Cross J. H.: Software process in the classroom: the Capstone project experience. IEEE Software, 19(5), pp. 78-81 (2002).
5. Karunasekera, S., Bedse, K.: Preparing software engineering graduates for an industry career. In Proceedings of the 30th Conference on Software Engineering Education & Training (CSEE&T), pp. 97-106. IEEE, Dublin, Ireland (2007).
6. Vasilevskaya M., Broman D., Sandahl K.: Assessing Large-Project Courses: Model, Activities, and Lessons Learned. Transactions on Computing Education 15(4), 30 p. (2015).
7. Bloomfield A., Sherriff M., Williams K.: A service learning practicum capstone. In 45th technical symposium on Computer science education (SIGCSE), pp. 265-270 (2014).
8. Goold A.: Providing process for projects in capstone courses. In Proceedings of the 8th conference on Innovation and technology in computer science education (ITiCSE), pp. 26-29. ACM, Thessaloniki, Greece (2003).
9. Ribaud V. et al.: Process Assessment Issues in a Bachelor Capstone Project. In International Workshop on Software Process Education, Training and Professionalism (IWSPETP), pp. 25-33. CEUR 1368, Gothenburg, Sweden (2015).
10. Dollard J., Miller N.E.: Personality and psychotherapy; an analysis in terms of learning, thinking, and culture. McGraw-Hill, New York (1950).
11. ISO/IEC 12207:2008, Systems and software engineering -- Software life cycle processes. ISO, Geneva (2008).
12. Rout, T.: The evolving picture of standardisation and certification for process assessment. In Proceedings of the 7th Conference on Quality of Information and Communications Technology (QUATIC), pp. 63-72. IEEE, Portugal (2010).
13. ISO/IEC 15504-5:2012. Information technology -- Process assessment -- Part 5: An exemplar software life cycle process assessment model. ISO, Geneva (2012).
14. Ribaud V., Saliou P.: Towards an ability model for SE apprenticeship. Innovation in Teaching and Learning in Information and Computer Science 6(3), pp. 7-107 (2007).
15. ISO/IEC 33001:2015. Information technology -- Process assessment -- Concepts and terminology. ISO, Geneva (2015).
16. ISO/IEC 33020:2015. Information technology -- Process assessment -- Process measurement framework for assessment of process capability. ISO, Geneva (2015).