Use and evaluation of simulation for software process education: a case study

Nauman bin Ali and Michael Unterkalmsteiner

Blekinge Institute of Technology
{nauman.ali,michael.unterkalmsteiner}@bth.se

Abstract. Software Engineering is an applied discipline and concepts are difficult to grasp only at a theoretical level alone. In the context of a project management course, we introduced and evaluated the use of software process simulation (SPS) based games for improving students' understanding of software development processes. The effects of the intervention were measured by evaluating the students' arguments for choosing a particular development process. The arguments were assessed with the Evidence-Based Reasoning framework, which was extended to assess the strength of an argument. The results indicate that students generally have difficulty providing strong arguments for their choice of process models. Nevertheless, the assessment indicates that the intervention of the SPS game had a positive impact on the students' arguments. Even though the illustrated argument assessment approach can be used to provide formative feedback to students, its use is rather costly and cannot be considered a replacement for traditional assessments.

Keywords: Software process simulation, project management, argument evaluation

1 Introduction

The Software Engineering (SE) discipline spans from technical aspects, such as developing techniques for automated software testing, over defining new processes for software development improvement, to people-related and organizational aspects, such as team management and leadership. This is evident in the software development process, which is *"the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product"* [11]. This breadth of topics encompassed here makes education in SE challenging as the interaction of the different disciplines cannot be exclusively taught on a theoretical level, but must also be experienced in practice. As such, SE education needs to identify means to prepare students better for their tasks in industry [16].

However, the complexity and dynamism of software processes makes it difficult to illustrate the implications of the chosen development process on the outcomes of a project. Students will have to undertake multiple iterations of developing the same project using different software development processes to

understand the various processes and their implication on the project attributes [26]. Such repetitions are however impractical because of the time and cost involved. To overcome this shortcoming software process simulation (SPS) has been proposed as a means of SE education. SPS is the numerical evaluation of a computerized-mathematical model that imitates the real-world software development process behavior [13]. It has been found to be useful in SE education as a complement to other teaching methods e.g. in combination with lectures, lab sessions and projects [17, 26].

In this paper we motivate, illustrate and evaluate how a targeted change was introduced in the graduate-level *Applied Software Project Management (ASPM)* course. The course aims to convey to students in a hands-on manner how to prepare, execute and finalize a software project. In previous instances of the course, we have observed that students encounter difficulties in choosing an appropriate software development process and in motivating their choice. We hypothesize that the students lack experience of different software development processes, and lack therefore the analytical insight required to choose a process appropriate for the characteristics of the course project. We study our hypothesis by exposing students to software process simulations (SPS) and by evaluating thereafter the argumentative strength for choosing/discarding a particular process.

There are three major contributions in this paper. First, a review of frameworks for evaluating argumentative reasoning was updated to cover more recent research. Secondly the framework relevant for evaluating arguments in the context of SE was selected and adapted. Thirdly, independent of the creators of SimSE, we used it in the context of an active course instead of a purely experimental setting, and evaluated its effect indirectly, in terms of students' understanding of software development processes.

The remainder of the paper is structured as follows: Section 2 summarizes the relevant work on the topic of SPS in SE education. Section 3 presents the context of the study, research questions, data collection and analysis methods. Section 4 presents the results, Section 5 revisits the research questions based on the findings and Section 6 concludes the paper.

2 Background and Related Work

In this section, we briefly discuss the two overarching themes in this study: SPS based education and evaluation on scientific argumentation.

2.1 SPS in SE education

SPS provides an alternative to manipulation of the actual software process by providing a test-bed for experimentation with realistic considerations. Compared to static and analytical models, SPS achieves this because of its ability to capture the underlying complexity in software development by representing uncertainty, dynamic behavior and feedback/feed-forward mechanisms [13].

Since the initial proposal of SPS its potential as a means of education and training was recognized [13]. Some of the claimed benefits of SPS for SE education include: increased interest in SE project management [18], motivation of students [8], and effective learning [20]. It can facilitate understanding by experiencing different processes with certain roles (e.g. as a as a software manager making decisions in software development, which would not have been possible in an academic context without SPS [27]).

Navarro and Hoek [17] evaluated the experience of students playing SPS based games for SE education. They found that the SPS based teaching is applicable for various types of learners as it aligns well with objectives of a multitude of learning theories. For example, it encourages exploratory learning by experimenting, emphasizes learning by doing and through failure, and by embedding in a context that resembles the real-world use of the phenomenon of interest.

Wangenheim and Shull [26], in a systematic literature review of studies using SPS for SE education, found that the two most frequent aims in such studies are *"SE Project Management"* and *"SE process"* knowledge [26]. They also found that in most of the existing research, subjective feedback was collected after the students had used the game [26]. Similarly, they reported that it was difficult to evaluate the effectiveness of SPS interventions because a majority of the articles do not report the *"expected learning outcome and the environment in which students used the game"* [26].

These findings motivated our choice to have a simulation based intervention in the course as the two major learning objectives for the course are related to project and process management. The context is described in Section 3.1. Furthermore, adhering to the recommendation that is based on empirical studies [26], we used SPS to target a *"specific learning need"* of the students, i.e. to improve the understanding and implications of a software development lifecycle process. SimSE was the chosen platform due to a stable release, good graphical user-interface and good feedback from earlier evaluations [17]. Unlike the existing evaluations of SimSE, in this study, we took an indirect approach to see if the simulation based intervention had the desired impact. We looked at the quality of arguments for the choice of the lifecycle process in the student reports without explicitly asking them to reflect on the SPS game.

2.2 Evaluating scientific argumentation

Argumentation is a fundamental driver of the scientific discourse, through which theories are constructed, justified, challenged and refuted [10]. However, scientific argumentation has also cognitive values in education, as the process of externalizing one's thinking fosters the development of knowledge [10]. As students mature and develop competence in a subject, they pass through the levels of understanding described in the SOLO taxonomy [5]. In the taxonomy's hierarchy, the quantitative phase (unistructural and multistructural levels) is where students increase their knowledge, whereas in the qualitative phase (relational and extended abstract levels) students deepen their knowledge [4]. The quality of scientific argumentation, which comprises skills residing in higher levels of the

SOLO taxonomy, is therefore a reflection of the degree of understanding and competence in a subject.

As argumentation capability and subject competence are intrinsically related, it is important to find means by which scientific argumentation in the context of education can be evaluated. Sampson and Clark [21] provide a review of frameworks developed for the purpose of assessing the nature and quality of arguments. They analyze the studied frameworks along three dimensions of an argument [21]:

1. Structure (i.e., the components of an argument)
2. Content (i.e., the accuracy/adequacy of an arguments components when evaluated from a scientific perspective)
3. Justification (i.e., how ideas/claims are supported/validated within an argument)

We used the same criteria to update their review with newer frameworks for argument evaluation. This analysis was used to select the framework appropriate for use in this study.

3 Research design

3.1 Context

The objective of the Applied Software Project Management (ASPM) course is to provide students with an opportunity to apply and sharpen their project management skills in a sheltered but still realistic environment. Students participating in ASPM typically[1] have completed a theory-building course on software project management, which includes an introduction to product management, practical project management guided by the Project Management Body of Knowledge [1], and an excursion to leadership in project teams [12].

Figure 1 shows the student characteristics of the two course instances that were studied. In 2012, without SPS intervention, 16 students participated in total, having accumulated on average 18 ECTS points at the start of the course. In 2013, with the SPS intervention, 15 students participated in total, having accumulated on average 84 ECTS points at the start of the course. In both course instances, three students did not take the theory course on software project management (Advanced SPM). The major difference between the two student groups is that in 2013, considerably more students did not successfully complete the Advanced SPM course. The higher ECTS average in 2013 can be explained by the participation of three Civil Engineering students who chose Applied SPM at the end of their study career while SE and Computer Science students chose the course early in their studies.

The course follows the three months schedule shown in Figure 2, which illustrates also the planned interactions between students and instructors. The introduced modifications are shown in italics and further discussed in Section 3.3.

[1] ASPM is also an optional course in the curriculum for students from computer science and civil engineering programs

Program	2012	2013
Master in Software Engineering	12	12
Master in Computer Science	4	0
Civil Engineer in Industrial Economy	0	3
ECTS (average)	18	84

Advanced SPM grade	2012	2013
A	0	0
B	2	0
C	5	3
D	2	3
E	3	1
Not finished	1	5
Not taken	3	3

Fig. 1. Student demographics from 2012 (without intervention) and 2013 (with SPS intervention) of the Applied SPM course

Students are expected to work 200 hours for this course, corresponding to a 20 hours/week commitment.

The course has five assignments but Assignment 1 and 5 are important for this study (see Figure 2). Assignment 1 consists of delivering a project management plan (PMP) where students also report the choice and rationale for a software process they will use. The teams receive oral feedback and clarifications on the PMP during the same week. The course concludes with a presentation where project teams demo their developed products. In Assignment 5, the students are asked to individually answer a set of questions that, among other aspects, inquiry their experience with the used software process in the project.

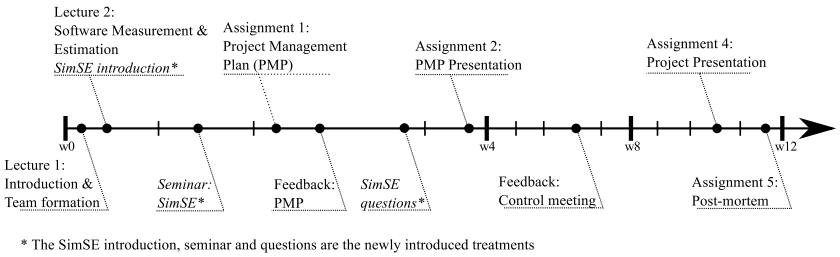

Fig. 2. ASPM course time-line with events

3.2 Research questions

The posed research questions in this study are:

RQ1: How can improvement in software development process understanding be assessed?

RQ2: To what extent can process simulation improve students' understanding of software development processes?

With RQ2, we investigate whether process simulation has a positive impact on students' understanding of development processes. Even though studies with

a similar aim have already been conducted (c.f. [18]), experiments in general are prone to the Hawthorne effect [7], where subjects under study modify their behavior knowing that they are being observed. Similar limitations can be observed in earlier evaluations of SimSE where *"the students were given the assignment to play three SimSE models and answer a set of questions concerning the concepts the models are designed to teach"* [17]. Hence we introduce process simulation as an additional teaching and learning activity into a course whose main purpose is *not* to teach software development processes. Furthermore, we do not modify the requirements for the graded deliverables. Formally, we stated the following hypotheses:

H_0: There is no difference in the students' understanding of process models in course instances 2012 and 2013.

H_a: There is a difference in the students' understanding of process models in course instances 2012 and 2013.

Due to the subtle modifications in the course, we needed to identify new means to evaluate the intervention, measuring the impact of introducing process simulation on students' understanding of development processes. In order to answer RQ1, we update the review by Sampson and Clark [21] with two more recent frameworks proposed by Reznitskaya et al. [19] and Brown et al. [6], select the framework that provides the strongest argument evaluation capabilities, and adapt it to the software engineering context.

In order to answer RQ2, we apply the chosen argument evaluation framework on artifacts delivered by project teams and individual students which did receive the treatments shown in Figure 2 and on artifacts delivered in the previous year.

3.3 Instrumentation and data collection

Assignment 5: Post mortem, as shown in Figure 2, is an individual assignment where students had to motivate and reflect on their choice of software process model selected in their projects. This assignment is used to evaluate the influence of SimSE on the student's understanding of the software processes. A baseline for typical process understanding of students from the course was established by evaluating *Assignment 5* from year 2012 and it was compared to the evaluation results of *Assignment 5* from year 2013. To supplement the analysis we also used *Assignment 1: Project Management Plan (PMP)* (which is a group assignment) from both years. The design for the study is shown in Figure 3. Where deltas *'a'* and *'b'* are changes in understanding between the Assignments 1 and 5 within a year. While deltas *'c'* and *'d'* represent changes across the years for Assignment 1 and 5 respectively.

For the evaluation of assignments, we used the EBR framework [6]. Other frameworks considered and the reasons for this choice are summarised in Section 4.2. Once the framework had been adapted, first it was applied on one assignment using *"Think-aloud protocol"* where the authors expressed their thought process while applying the evaluation instrument. This helped to identify ambiguities in the instrument and also helped to develop a shared understanding

of it. A pilot of the instrument was done on two assignments where the authors applied it separately and then compared and discussed the results. Both authors individually coded all the assignments and then the codes were consolidated with consensus. The results of this process are presented in Section 4.3.

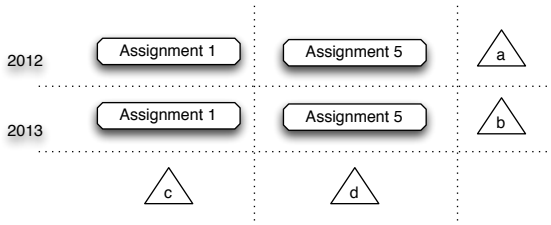

Fig. 3. Design to evaluate the impact of the SPS intervention

3.4 Limitations

The assignments were retrieved from the Learning Management System and personal identification of students was replaced with a unique identifier to hide their identity from the authors. This was done to avoid any personal bias that the authors may have towards the students as their teachers, in this and other courses. Furthermore, to ensure an unbiased assessment both the overall grades of students and their grades in the assignments were hidden from the authors when the assessment instrument was applied in this study.

To avoid any bias introduced by asking questions directly about the intervention of process simulation, and to have a relative baseline for assignments from 2012, we did not change the assignment descriptors for the year 2013. Thus we tried to measure the effect of the intervention indirectly by observing the quality of argumentation without explicitly asking students to reflect on the experience from simulation based games.

The intervention was applied in a post-graduate course, rendering experiment-like control of settings and variables impossible. Among other factors, any difference in results could purely be because of the different set of students taking the course in the years 2012 and 2013. However, as discussed in Section 3.1 the groups of students were fairly similar thus the results are comparable. Small number of students is also a limitation of this study.

Similarly, by having the students fill out questions about the various simulation based games we tried to ensure that students have indeed played the games. However, we have no way of ensuring that the students did indeed play the games individually and not share the answers with each other. This limitation could weaken the observable effect of the intervention.

4 Results

In this section we report the two main results of our study. In Section 4.1 we review two argument evaluation approaches and classify them according to the

framework presented in Section 2.2. Then we choose one argument evaluation approach and adapt it to our goals (Section 4.2), and apply it to students arguments on choosing a particular process model for their project (Section 4.3). The data (argument coding and quantification) is available in the supplementary material to this paper, available at http://www.bth.se/com/mun.nsf/pages/simeval.

4.1 Review update

In Table 1 we summarize Sampson and Clark's [21] analysis w.r.t. the support various frameworks provide to assess structure, content and justification of an argument. In the rest of the section we report the classification of two newer frameworks as an extension to their review.

Table 1. Strengths and weaknesses of argument assessment frameworks

Framework	Structure	Content	Justification
Domain-general			
Toulmin [25]	strong	weak	weak
Schwarz et al. [23]	strong	moderate	moderate
Domain-specific			
Zohar and Nemet [28]	weak	moderate	strong
Kelly and Takao [14]	strong	weak	strong
Lawson [15]	strong	weak	strong
Sandoval [22]	weak	strong	strong

Brown et al. [6] propose with the Evidence-based Reasoning (EBR) framework an approach to evaluate scientific reasoning that combines Toulmin's argumentation pattern [25] with Duschl's framework of scientific inquiry [9]. This combination proposes scientific reasoning as a two-step process in which a scientific approach to gather and interpret data results in rules that are applied within a general framework of argumentation [6].

Figure 4a shows the structure of the framework, consisting of components that should be present in a strong scientific argument. The strength of an argument can be characterized by the components present in the argument. For example, in an unsupported claim (Figure 4b), there are no rules, evidences or data that objectively support the claim. An analogy (Figure 4c) limits the argumentation on supporting a claim only with instances of data, without analyzing and interpreting the data. In an overgeneralization (Figure 4d), analysis and formation of a body of evidence is omitted and data is interpreted directly to formulate rules (theories, relationships) that are not supported by evidence.

The EBR was not designed for a specific scientific context [6] and we classify it therefore as a domain-general framework. It provides strong support for evaluating arguments along structure (i.e. the components of an argument) and justification (i.e. how claims are supported within an argument) dimensions.

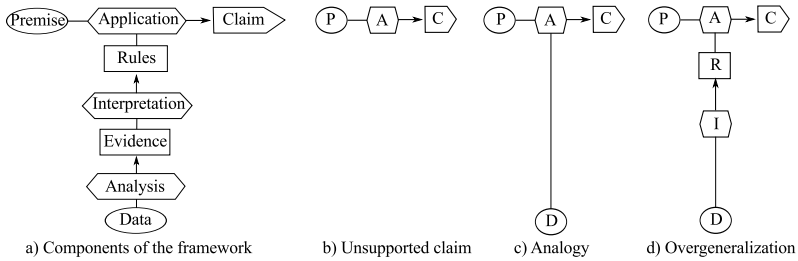

Fig. 4. The Evidence-Based Reasoning framework (a) and different degrees of argument sophistication (b-d) (adapted from Brown et al. [6])

However, solely identifying rules and evidences components in an argument does not provide an assessment of the arguments content, i.e. the adequacy of argument components. As such, the framework provides the tools to identify components of content (rules and evidences), but no direct means to assess the content's quality. Therefore, we rate the framework's support for the content dimension as moderate.

Reznitskaya et al. [19] propose a domain-specific framework for evaluation of written argumentation. They proposed and compared two methods to implement this framework:

1. Analytical method, which is a data driven approach where individual statements are coded and their relevance to the main topic is judged, categories are derived from these codes (deciding about these categories will be based on the theoretical and practical significance in the domain). Next the report is evaluated on five sub-scales which cover aspects from: number of arguments made, types of previously identified categories of arguments covered in the report, opposing perspectives considered, number of irrelevant arguments and the use of formal aspects of discourse.
2. Holistic method takes a rubric based approach attempting to provide a macro level assessment of the arguments.

In essence, the framework has no explicit focus on the components of an argument and only indirectly covers the aspects of structure while creating the instrument. The fundamental building block of the evaluation framework is the analytical coding process where both the content and justification are considered. Content (accuracy and adequacy) is only assessed by identifying the relevance of the argument to the topic. Justification is covered indirectly in the variety of argument categories identified in the reports. However, all argument categories are given equal weight in scoring. Therefore, we rate the framework support for the structure as weak, and for content and justification as moderate.

4.2 Selection and adaptation of an argument evaluation framework

Based on the analysis of the reviewed frameworks, summarized in Table 1, and the updated review presented in Section 4.1, we decided to use the EBR frame-

work [6]. We chose a domain-general over a domain-specific framework due to our preference of customizing generic principles to a specific context. The alternative, to construct an assessment instrument completely inductively from the particular domain and data, as for example in Reznitskaya et al. [19], would imply a relative assessment approach, weakening the evaluation of the intervention. Furthermore, a domain-generic approach allows us to re-use the assessment instrument with minor adaptations, lowering the application cost by keeping the general approach intact.

Table 2 shows an example argument with the EBR components one can identify in written statements. This example illustrates what we would expect in a strong argument: a general rule on a process model is applied on a premise that refers to the specific circumstances of the students' project, justifying their claim that the selected model was the best choice. The rule is supported by evidence (a reference to a scientific study) and by an experience from the project that creates a relationship between short development cycles and late changes. The evidence is supported by data from the project.

Table 2. Example application of the EBR on an ideal argument

Component	Statement
Premise	The requirements for our product are not that clear and likely to change.
Claim	eXtreme Programming (XP) was the best choice for our project.
Rule	XP embraces continuous change by having multiple short development cycles.
Evidence	Reference to Beck [3]; customer changed user interaction requirements six weeks before the delivery deadline but we still delivered a working base product;
Data	Seven change requests to initially stated requirements; four requirements were dropped since customer was satisfied already;

The EBR framework enables a fine-grained deconstruction of arguments into components. The price for this strength on the structural and justification dimension is a moderate support for assessing content (see Section 4.1). Even though the overall argument content can be judged to some extent by the interplay between the argument components, domain-specific knowledge is nevertheless required to judge whether individual statements are accurate. Looking at Table 2, the rule component in particular requires knowledge on XP in order to decide whether the statement is accurate or not. Hence we assess the argument content by qualifying a rule as sound/unsound, given the stated premise, claim, evidence and data, based on our domain knowledge on process models. Concretely, we declare an argument for a claim as:

- Sound
  - If the stated rule is backed by evidence/data *and* is pertinent for the premise (strong).

- In arguments with no corroborative evidence (weak): If the stated rule is in compliance with literature and/or the assessors understanding of the topic *and* is pertinent for the premise.

– Unsound
  - If an argument does not fulfill either of the above two criteria.

4.3 Application of the chosen framework

In this section we illustrate the results of applying the EBR framework on the students' arguments for choosing/rejecting a particular process model for their project. We coded statements according to the EBR frameworks' components of an argument: a premise (P), rule (R), evidence (E), data (D). We also noted when components are used atomically or are combined into pairs or triples of components to form a coherent argument. Based on this codification, we evaluated the overall content of the argument (unsound / weak / strong) by following the rules established in Section 4.2. The claim of the argument, in principle constant and only changing in sign, was that a particular process model is / is not the best choice for the project.

Table 3 shows the results in terms of the argument component frequencies encountered in the students' project plans from 2012 (without intervention) and 2013 (with SPS intervention). For example, in Plan #1 we identified 8 premises (P), 4 premise-rule (PR) pairs and 1 premise-evidence (PE) pair. We expected to find some premise-rule-evidence (PRE) triples as they would indicate that students can motivate their choice by examples, e.g. by referring to scientific literature, to experience from previous projects or from playing SimSE. However, the results clearly indicate a tendency for students to create overgeneralizing arguments (premise-rule pairs). Looking at the argument content, we identified no strong arguments (lack of evidence component) and, in proportion to weak arguments, a rather large number of unsound arguments, indicating a lack of understanding of process models and their properties.

Table 3. Frequencies of identified argument components and argument content strength in project plans for choosing a particular process model

Year	Plan#	P	PR	PE	PRE	RE	R	Unsound	Weak	Strong
	1	8	4	1	0	0	0	2	2	0
2013	2	9	7	0	0	0	3	4	6	0
	3	3	1	0	0	1	0	1	1	0
	Sum	**20**	**12**	**1**	**0**	**1**	**3**	**7**	**9**	**0**
	4	10	3	0	0	0	1	1	3	0
	5	5	7	0	0	1	1	3	6	0
2012	6	6	3	0	0	0	0	0	4	0
	7	2	1	0	0	0	2	1	1	0
	Sum	**23**	**14**	**0**	**0**	**1**	**4**	**5**	**14**	**0**

After assessing the project plans, which were a group assignment, we applied the EBR framework on the project post-mortems that were handed in individually by the students (13 in 2013 and 12 in 2012). Table 4 illustrates the results, showing the frequencies of identified argument components and component combinations. Observe that, in contrast to the post-mortem, we identified more combinations of components, e.g. premise-data (PD) or premise-evidence-data (PED) triples. For both years it is evident that students reported more justification components (evidence and data) in the post-mortem than in the project plan. This is expected as we explicitly asked to provide supporting experience from the conducted project.

Table 4. Frequencies of identified argument components and argument content strength in project post-mortems for choosing a particular process model.

Year	Premise	Premise-Rule	Premise-Evidence	Premise-Rule-Evidence	Rule-Evidence	Rule	Evidence	Data	Evidence-Data	Premise-Data	Rule-Data	Rule-Evidence-Data	Premise-Rule-Data	Premise-Evidence-Data	Premise-Rule-Evidence-Data	**Unsound Argument**	**Weak Argument**	**Strong Argument**
2012	5	2	3	4	14	11	16	4	8	4	2	3	0	1	2	17	14	7
2013	30	5	5	9	17	20	9	16	9	3	9	6	2	1	0	12	42	14

5 Analysis and revisiting research questions

5.1 Assessing software development process understanding (RQ1)

With support from the EBR framework we decomposed students' arguments to a degree that allowed us to pinpoint the weaknesses of their development process model understanding. Looking at the frequencies in Table 4, we can observe that:

- For both years, a relatively large number of standalone argument components were identified (e.g. single premise, rule and data components in 2013 and evidence components in 2012). A standalone argument component indicates a lack of a coherent discussion, a concatenation of information pieces that does not create a valid argument for a specific claim. There are exceptions, e.g. PM#8 and PM#24 (see supplementary material), which is also expressed in a strong argument content rating.
- Looking at the argument component combinations that indicate a strong argument (i.e. that contain a premise, a rule, and evidence or data), we can observe that assignments containing weak arguments outnumber assignments with strong arguments in both years.

The detailed analysis of arguments with the EBR framework could also help to create formative feedback to students. For example, in PM#5, the student reported 7 data points on the use of SCRUM but failed to analyze this data, creating evidence (describing relationships of observations) and connecting them to rules. On the other hand, we identified several assignments with the premise that the requirements in the project are unknown or change constantly (using this premise to motivate the selection of an Agile process). However, none of the assignment reports data on change frequency or volatility of requirements, weakening therefore the argument for choosing an Agile process.

Given these detailed observations one can make by applying the EBR framework we think it is a valuable tool for both assessing arguments and to provide feedback to students. However, this power comes at a price. We recorded coding times between 10 and 30 minutes per question, excluding any feedback formulation that the student could use for improvement. Even if coding and feedback formulation efficiency could be increased by tools and routine, one has to consider the larger effort this type of assessment requires compared to other means, e.g. rubrics [2].

5.2 Impact of SPS on students' understanding of software development processes (RQ2)

The only difference in how the course was conducted in 2013 compared to 2012 was the use of SimSE simulation based software process games. Besides the limitation of this study (as discussed in Section 3.4) improvements in the students' understanding can be seen as indications of usefulness of SimSE based games for software process education.

In order to evaluate students' understanding, we measured the quality of argumentation for the choice of the software process. Concretely, we evaluated the content of the student reports by using the strength (classified as strong, weak and unsound) of an argument as an indicator. To test the hypotheses stated in Section 3.2, we used the chi-square test of independence [24] and rejected H_0 at a confidence level of $\alpha < 0.05$.

For the project management plan (Table 3), which was a group assignment and was delivered at the beginning of the course, the observed frequency of strong, weak and unsound arguments did not differ significantly between 2012 and 2013. Hence we cannot reject H_0 for the project management plan.

For the project post-mortem (Table 4), which was an individual assignment and was delivered at the end of the course, the observed frequency of strong, weak and unsound arguments did differ significantly between 2012 and 2013 ($chi - squared = 8.608$, $df = 2$, $p - value = 0.009$). Hence we can reject H_0 for the project post-mortem and accept that there is a difference in process model understanding of students in course instances 2012 and 2013. However, since this difference only materialized at the end of the course, after the project has been conducted, the improved understanding cannot be attributed to the software process simulation game alone as discussed in the limitations of this study (Section 3.4).

Another indirect observation that shows a better understanding of the process model is reflected in the choice of the process model in the context of the course project (with small collocated teams, short fixed duration for project etc.). Compared to 2012 where most of the groups took plan driven, document intensive process models (two groups chose an incremental development model, one group chose Waterfall and only one chose Scrum), in the year 2013 all groups chose a light-weight, people centric process model that is more pertinent to the given context.

6 Conclusion

The EBR framework enabled decomposition of arguments into distinct parts, which ensured an objective evaluation of the strength of the arguments in student reports. This assessment allowed us to gauge students' understanding of software development processes.

The indications reported in this study (from use of software process simulation in an active course) adds to the confidence in evidence reported in earlier empirical studies in controlled settings. Given the potential gains as seen in this study, and relative maturity, user interface and decent documentation of SimSE, the minor additional cost of including it in a course to reinforce concepts already learned was well justified.

As future work, we intend to do a longitudinal study where more data is collected over the next few instances of the course.

References

1. *A Guide to the Project Management Body of Knowledge: PMBOK Guide*. PMBOK Guides. Project Management Institute, 3rd edition, 2004.
2. S. Barney, M. Khurum, K. Petersen, M. Unterkalmsteiner, and R. Jabangwe. Improving students with rubric-based self-assessment and oral feedback. *IEEE Transactions on Education*, 55(3):319 –325, Aug. 2012.
3. K. Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, Oct 1999.
4. J. Biggs and C. Tang. *Teaching for Quality Learning at University: What the Student does*. Mcgraw-Hill Publ.Comp., 3rd edition. edition, Nov. 2007.
5. J. B. Biggs and K. F. Collis. *Evaluating the quality of learning: the SOLO taxonomy (structure of the observed learning outcome)*. Academic Press, 1982.
6. N. J. S. Brown, E. M. Furtak, M. Timms, S. O. Nagashima, and M. Wilson. The evidence-based reasoning framework: Assessing scientific reasoning. *Educational Assessment*, 15(3-4):123–141, 2010.
7. J. P. Campbell, V. A. Maxey, and W. A. Watson. Hawthorne effect: Implications for prehospital research. *Annals of Emergency Medicine*, 26(5):590–594, Nov. 1995.
8. A. Drappa and J. Ludewig. Simulation in software engineering training. *Proc. of the 22nd Int. Conf. on Software Engineering - ICSE '00*, pages 199–208, 2000.
9. R. A. Duschl. Assessment of inquiry. *Everyday assessment in the science classroom*, pages 41–59, 2003.

10. S. Erduran, S. Simon, and J. Osborne. TAPping into argumentation: Developments in the application of toulmin's argument pattern for studying science discourse. *Science Education*, 88(6):915–933, 2004.
11. A. Fuggetta. Software process: a roadmap. In *Proc. of the Conf. on the Future of Software Engineering*, pages 25–34. ACM, 2000.
12. P. Hersey, K. H. Blanchard, and D. E. Johnson. *Management of organizational behavior: leading human resources*. Prentice Hall, Upper Saddle River, N.J., 2001.
13. M. I. Kellner, R. J. Madachy, and D. M. Raffo. Software process simulation modeling : Why ? What ? How ? *J. of Syst. Software*, 46(2-3):91–105, 1999.
14. G. J. Kelly and A. Takao. Epistemic levels in argument: An analysis of university oceanography students' use of evidence in writing. *Science Education*, 86(3):314–342, 2002.
15. A. Lawson. The nature and development of hypotheticopredictive argumentation with implications for science teaching. *International Journal of Science Education*, 25(11):1387–1408, 2003.
16. T. Lethbridge, J. Diaz-Herrera, J. LeBlanc, R.J., and J. Thompson. Improving software practice through education: Challenges and future trends. In *Future of Software Engineering, FOSE '07*, pages 12–28, 2007.
17. E. Navarro and A. van der Hoek. Comprehensive evaluation of an educational software engineering simulation environment. In *Proc. of the 20th Conf. on Software Engineering Education Training, CSEET '07*, pages 195–202, 2007.
18. D. Pfahl, O. Laitenberger, J. Dorsch, and G. Ruhe. An externally replicated experiment for evaluating the learning effectiveness of using simulations in software project management education. *Empirical Software Engineering*, 8(4):367–395, 2003.
19. A. Reznitskaya, L.-j. Kuo, M. Glina, and R. C. Anderson. Measuring argumentative reasoning: What's behind the numbers? *Learning and Individual Differences*, 19(2):219–224, June 2009.
20. D. Rodriguez. e-Learning in project management using simulation models: A case study based on the replication of an experiment. *IEEE Transactions on Education*, 49(4):451–463, 2006.
21. V. Sampson and D. B. Clark. Assessment of the ways students generate arguments in science education: Current perspectives and recommendations for future directions. *Science Education*, 92(3):447–472, 2008.
22. W. A. Sandoval. Conceptual and epistemic aspects of students' scientific explanations. *Journal of the Learning Sciences*, 12(1):5–51, 2003.
23. B. B. Schwarz, Y. Neuman, J. Gil, and M. Ilya. Construction of collective and individual knowledge in argumentative activity. *Journal of the Learning Sciences*, 12(2):219–256, 2003.
24. D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures, Second Edition*. Chapman and Hall/CRC, Boca Raton, 2 edition edition, Feb. 2000.
25. S. E. Toulmin. *The uses of argument*. Cambridge University Press, Cambridge, U.K.; New York, 1958.
26. C. von Wangenheim and F. Shull. To game or not to game? *IEEE Software*, pages 92–94, 2009.
27. A. Zapalska, D. Brozik, and D. Rudd. Development of Active Learning with Simulations and Games. *US-China Education Review*, 2:164–169, 2012.
28. A. Zohar and F. Nemet. Fostering students' knowledge and argumentation skills through dilemmas in human genetics. *Journal of Research in Science Teaching*, 39(1):35–62, 2002.