Use of simulation for software process education: a case study

Nauman bin Ali and Michael Unterkalmsteiner

Blekinge Institute of Technology,
37179 Karlskrona, Sweden
{nauman.ali,michael.unterkalmsteiner}@bth.se
http://www.bth.se

Abstract. Teaching Software Engineering concepts is often accomplished by practising them in a sandbox environment. The objective of this study is to introduce and evaluate the use of software process simulation (SPS) based games for improving students' understanding of software development processes. In the context of an applied software project management course, we measure the effects of the intervention by evaluating the students' arguments for choosing a particular software development process model. The arguments are assessed with the Evidence-Based Reasoning framework, which we extended in order to assess the strength of an argument. The results indicate that students have generally difficulties 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 therefore should not 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. SE shares common boundaries with many other disciplines, such as computer science, mathematics, systems engineering, quality management and others [2]. This breadth of topics renders education in SE challenging as the interaction of the different disciplines cannot be exclusively thought on a theoretical level, but must be also experienced in practice. As such, SE education needs to identify means to prepare students better for their tasks in industry [23].

In this paper we motivate, illustrate and evaluate how a targeted change in teaching and learning activities was introduced in the graduate-level *Applied*

Software Project Management (ASPM) course. The objective of this course is 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 on and exposure to different software development processes, and have therefore deficiencies in the sensitivity and 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.

Software development processes are highly dynamic and complex, which involve interaction between people, tools, technology, and organization [14]. This makes it difficult to illustrate the implications of the chosen development process on the success of a project. Students will have to undertake multiple iterations of developing the same project using different software development processes to understand the various phases, the sequence of phases and their implication on the project attributes like cost and time taken for development, and the quality of the developed product [40]. 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 [24, 19]. 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 [29, 40].

There are four 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 we used SimSE in the context of an active course ASPM instead of a purely experimental setting. Lastly the effect of simulation, in terms of improved understanding of software development processes, was evaluated.

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 discusses the findings and Section 6 concludes the study.

2 Background and Related Work

2.1 Software development processes

A software development process can be defined as: *"the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product"* [14]. Some of the more frequently used software development processes include *"Royce model* [34], iterative and incremental model [21] and the Spiral model [7]. Not only do each of these models have a divergent organization of activities but they also advocate a different philosophy altogether. For example, one dimension of variation among these processes is the emphasis they put on planning, another dimension is that of people or document centric nature.

2.2 Process simulation

SPS is numerical evaluation of a mathematical model that imitates a real-world software development process. SPS provides an alternative to manipulation of the actual software process that may be too risky. Contrary to static and analytical models, SPS can capture behavior of the software development process more realistically by [19] [24]:

- Representing the uncertainty in software development induced from variation in measurements, human factors (e.g. motivation) and other confounding factors.
- Capturing dynamic variables like productivity and defect detection rates.
- Representing how behavior and decisions taken at one stage affect others in complex and indirect ways that need to be considered to understand the process behavior.

SPS serves as a test-bed for experimentation with realistic considerations (because of the ability of SPS models to capture the underlying complexity in software development in terms of representing uncertainty, dynamic behavior and feedback/feed-forward mechanisms) [19]. It provides an inexpensive [26, 19] means to experiment with software development (the development of SPS models will be a onetime cost). It further facilitates understanding and experiencing different processes by participating with a certain role [41]. For example, as a software manager making decisions in software development, which would not have been possible in an academic context without SPS.

2.3 Learning theories

Navarro and Hoek [29] 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, including:

- Discovery learning [31] that encourages exploratory learning by experimenting, questioning and seeking answers to *"discover"* a piece of knowledge.
- Learning through failure [37] that is similar to discovery learning however it is based on the assumption that failure yields a more memorable lesson and motivates the student to try again to succeed.
- Learning by doing [31] that emphasizes to provide opportunities for students *"to do"* what they are learning.

– Situated Learning [8] that emphasizes to embed the *"learning by doing"* activities in a context and culture that resemble its real-world use.
– Kellers attention, relevance, confidence, and satisfaction (ARCS) motivation theory [18] that focuses on learner's feeling (in terms of ARCS) instead of the physical environment to motivate students to learn.

For a more detailed discussion of these learning theories and their use in SE education please see [28] [27].

2.4 Simulation in Software Engineering education

Since the initial proposal of SPS [25] its potential as a means of education and training was recognized [19]. Some of the claimed benefits of SPS for SE education include: increased interest in SE project management [30], motivation of students [11], and effective learning [33].

Wangenheim and Shull [40] evaluated the evidence to validate such claims in a systematic literature review. They found that the two most often aims of primary studies were teaching *"SE Project Management"* and *"SE process"* knowledge [40]. This also motivated our choice to have a simulation based intervention in this course given that two major ILO are related to project and process management.

Wangenheim and Shull [40] also found that in most of the existing research subjective feedback was collected after the students used the game. In this study, we have taken a more 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.

Wangenheim and Shull [40] also 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"* [40]. In this study we report the context in detail and adhering to their recommendation, which is based on empirical studies, use SPS to target a *"specific learning need"* of the students [40], i.e. improving the understanding and implication of a development lifecycle process.

2.5 Evaluating scientific argumentation

Argumentation is a fundamental driver of the scientific discourse, through which theories are constructed, justified, challenged and refuted [13]. However, scientific argumentation has also cognitive values in education, as the process of externalizing one's thinking fosters the development of knowledge [13]. As students mature and develop competence in a subject, they pass through the levels of understanding described in the SOLO taxonomy [6]. 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 [5]. The quality of scientific argumentation, which comprises skills residing in higher levels of the

Table 1. Strengths and weaknesses of argument assessment frameworks

Framework	Structure	Content	Justification
Domain-general			
Toulmin [39]	strong	weak	weak
Schwarz et al. [38]	strong	moderate	moderate
Domain-specific			
Zohar and Nemet [42]	weak	moderate	strong
Kelly and Takao [20]	strong	weak	strong
Lawson [22]	strong	weak	strong
Sandoval [36]	weak	strong	strong

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 [35] 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 [35]:

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)

Sampson and Clark apply the reviewed frameworks on a sample argument, used throughout their analysis, and illustrate to what extent argument assessment is supported. In Table 1 we summarize their analysis w.r.t. the support a framework provides to assess structure, content and justification of an argument.

3 Research design

3.1 Context

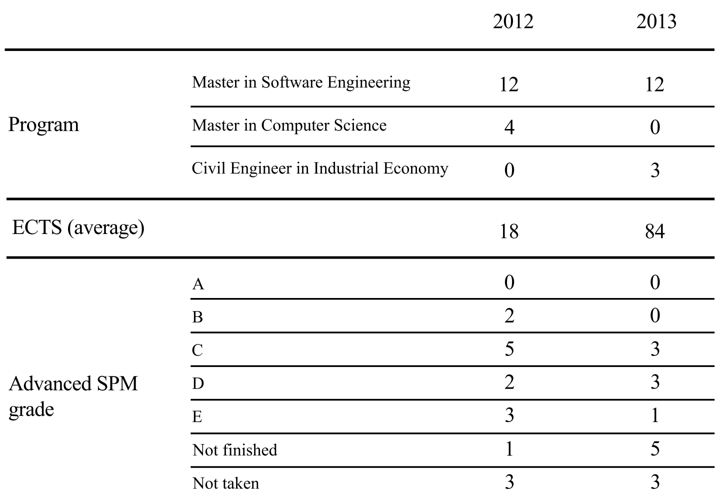
The objective of the Applied Software Project Management (ASPM) course is to provide students with an opportunity to apply and sharpen their project management skills in a sheltered but still realistic environment by "steering and administrating a project from start to finish, applying methods and techniques for making sure the project ends in a successful manner and, additionally, understanding and learning how to interpret stakeholders interest in a typical project.[1]" Students participating in ASPM typically[2] have completed a theory-building course

[1] Objective taken from the course descriptor PA2407, available from the authors.

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

on software project management, including 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 [16].

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.



		2012	2013
Program	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	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

The course follows the three months schedule shown in Figure 2, which illustrates also the planned events and interactions between students and instructors. The introduced modifications are shown in italics and further discussed in Section 3.3. Students are expected to work 200 hours for this course, corresponding to a 20 hours/week commitment.

During the introductory lecture, the instructors form the project teams. This introduces some additional challenges for the students as they would be expected in a real-life project in a development company. The teams are formed to be

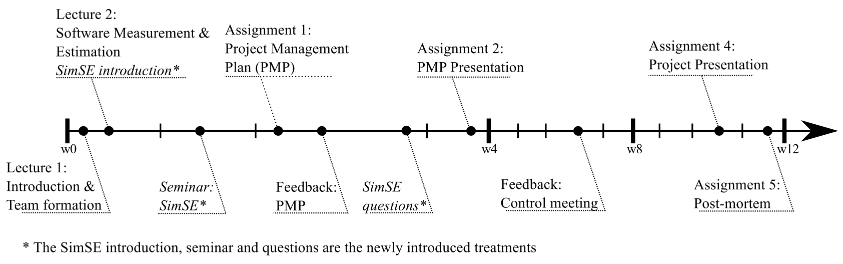

Lecture 2:
Software Measurement & Estimation
*SimSE introduction**
Assignment 1:
Project Management Plan (PMP)
Assignment 2:
PMP Presentation
Assignment 4:
Project Presentation
w0
w4
w8
w12
Lecture 1:
Introduction & Team formation
*Seminar: SimSE**
Feedback: PMP
*SimSE questions**
Feedback: Control meeting
Assignment 5: Post-mortem
* The SimSE introduction, seminar and questions are the newly introduced treatments

Fig. 2. ASPM course time-line with events

heterogeneous in terms of the members' personality traits and homogeneous in terms of their technical skills. The latter is achieved by surveying the students' capabilities, e.g. their knowledge in programming languages and problems domains. The second aspect considered in team formation is individual personality traits, assessed with the whole-brain model [15]. The goal is to achieve a constellation that represents all four thinking types (logical, sequential, conceptual, emotional), leading to an effective team. In order to kick-start collaboration and get the students acquainted with each other, several team building activities are conducted, e.g. a strength, weaknesses, opportunities, threats (SWOT) analysis [17, 43].

In the second lecture, an introduction to software measurement and estimation is given. The goal is to provide the teams with the practical means to estimate software size (using function point analysis) and estimate effort (using COCOMO or simple linear regression). The first assignment consists of delivering a project management plan (PMP) which is assessed by a dedicated rubric [3]. The teams receive oral feedback and clarifications on the PMP during the same week, together with assessment criteria for assignment two (see Figure 2). The PMP presentations should focus on how the team has addressed the received feedback and report on the progress. During week seven, a control meeting is held with individual teams, again focusing on how the project progresses.

The course concludes with a presentation where project teams demo their developed products. In assignment five, the students are asked to individually answer a set of questions that inquiry their experience during the project and proposals for improvement.

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 , experiments in general are prone to the Hawthorne effect [10], where subjects under study modify their behavior knowing that they are being observed. 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. However, due to the subtle modifications of the course, new means to evaluate the impact are needed. Hence, with RQ1 we aim to identify the means by which we can measure 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 [35] with two more recent frameworks proposed by Reznitskaya et al. [32] and Brown et al. [9], select the framework that provides the strongest argument evaluation capabilities, and, if necessary, 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 previous years. Instrumentation and data collection are illustrated in Section 3.3.

3.3 Instrumentation and data collection

Assignment 5: Post mortem, as shown in Figure 2, is an individual assignment where students had to motivate 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 years for Assignment 1 and 5 respectively.

For the evaluation of assignments, we used the EBR framework [9]. 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 assignments using "Think-aloud protocol" where the author expressed their thought process while applying the evaluation instrument. This helped to identify ambiguities in the instrument and also helped 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.

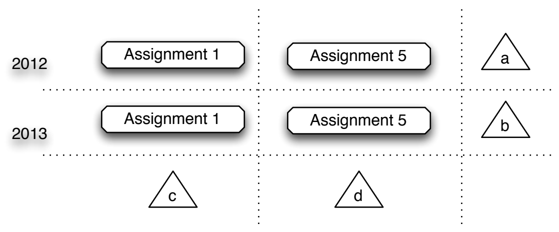


Fig. 3. Study design used in this study.

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 based on the experience from simulation based games.

The intervention was applied in a post-graduate course therefore experiment like control of settings and variables was not possible. 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.

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 just shared 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 2.5. 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).

4.1 Review update

Brown et al. [9] propose with the Evidence-based Reasoning (EBR) framework an approach to evaluate scientific reasoning that combines Toulmin's argumentation pattern [39] with Duschl's framework of scientific inquiry [12]. 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 [9].

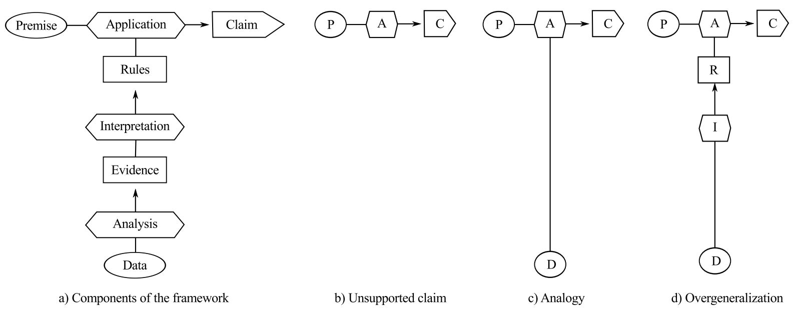


Fig. 4. The Evidence-Based Reasoning framework (a) and different degrees of argument sophistication (b-d) (adapted from Brown et al. [9])

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 [9] 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. 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 frameworks support for the content dimension as moderate.

Reznitskaya et al. [32] 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 subscales 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 framework [9]. 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. [32], 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 adaptions, 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.

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 of 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 required

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 [4]; 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;

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
	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

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 fulfil 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 the SPS. 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.

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.

5 Discussion

5.1 Assessing software development process understanding

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 could be identified (e.g. premise (P), rule (R) and data (D) in 2013 and evidence (E) 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 Table 4), which is also expressed in a strong argument content rating.

Table 4. Frequencies of identified argument components and argument content strength in project post-mortems for choosing a particular process model. Premise(P), Rule(R), Evidence(E), Data(D).

Year	PM#	P	PR	PE	PRE	RE	R	E	D	ED	PD	RD	RED	PRD	PED	PRED	Unsound	Weak	Strong
	1	3	3	0	2	0	2	1	2	1	0	0	0	0	0	0	2	5	1
	2	0	0	0	0	0	3	0	2	1	0	0	0	0	0	0	0	3	0
	3	5	1	0	2	2	1	1	2	1	1	1	0	0	0	0	1	4	2
	4	1	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	2	1
	5	0	1	1	0	0	4	0	7	0	0	0	0	1	0	0	0	5	1
	6	0	0	3	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0
2013	7	14	0	0	1	0	1	1	0	0	0	1	0	1	0	0	1	1	2
	8	0	0	0	2	1	1	0	1	0	1	0	1	0	0	0	0	2	3
	9	0	0	0	1	0	0	0	1	1	0	3	0	0	0	0	1	3	0
	10	7	0	0	0	0	5	0	1	0	0	3	0	0	0	0	5	3	0
	11	0	0	0	0	4	2	2	0	2	0	0	1	0	0	0	1	6	1
	12	0	0	0	0	9	0	0	0	0	0	0	2	0	0	0	1	8	2
	13	0	0	0	0	1	0	4	0	2	0	1	1	0	0	0	1	1	1
	Sum	**30**	**5**	**5**	**9**	**17**	**20**	**9**	**16**	**9**	**3**	**9**	**6**	**2**	**1**	**0**	**13**	**42**	**14**
	14	0	0	0	0	0	0	0	1	1	3	0	0	0	0	1	0	0	1
	15	0	0	0	1	2	0	1	3	3	1	1	1	0	0	0	0	4	1
	16	0	0	0	0	2	0	1	0	1	0	0	1	0	0	1	2	3	0
	17	0	2	0	0	2	0	0	0	1	0	1	0	0	0	0	3	2	0
	18	4	0	1	1	3	0	0	0	0	0	0	0	0	1	0	0	3	1
2012	19	1	0	0	0	1	0	2	0	1	0	0	0	0	0	0	0	1	0
	20	0	0	0	0	3	0	3	0	0	0	0	0	0	0	0	1	1	1
	21	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	22	0	0	0	1	0	0	6	0	0	0	0	0	0	0	0	1	0	0
	23	0	0	1	0	0	8	1	0	1	0	0	0	0	0	0	8	0	0
	24	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	3
	25	0	0	0	0	0	3	1	0	0	0	0	0	0	0	0	3	0	0
	Sum	**5**	**2**	**3**	**4**	**14**	**11**	**16**	**4**	**8**	**4**	**2**	**3**	**0**	**1**	**2**	**17**	**14**	**7**

– 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 assignment 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 [3].

5.2 Impact of SPS on students' understanding of software development processes

We used the quality of argumentation for the choice of the software process as a measure of students' understanding. The only difference in how the course was conducted in 2012 was that of 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.

We focused on evaluating the content of the student reports and used both the strength (classified as strong, weak and unsound) and frequency of arguments as an indicator. No discernible differences are visible in the group assignments (Table 3) but when we compare the results of individual project post-mortem reports for the years 2012 and 2013 (as shown in Table 4) some clear indications appear. Results for the year 2013 project post-mortem show twice as many strong arguments as the previous year. Similarly, the number of relatively weaker arguments has increased by three times while at the same-time the number of *'unsound'* arguments has improved only slightly.

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 incremental development model, one group chose Waterfall and only one chose Scrum), in 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 helps the decomposition of student arguments into distinct parts that enabled an assessment of students' understanding of software development processes. Even though it was appropriate for the goal of the study and provided higher quality of assessment its use as a tool for assessing assignments in general cannot be motivated due to the time taken for such assessment.

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. With a relatively mature simulation based game like SimSE, with a good GUI and decent documentation the cost of including it in a course to reinforce concepts already learned is justified given the potential gains as seen in this study.

Future work could do a longitudinal study where more data is collected over the next few instances of the course. Another alternative could be to consider applying the same approach, asking students however explicitly to use experience from simulation to motivate their choice in PMP. This would allow us to evaluate whether SPS has an impact, i.e. if experiencing the process in a simulation improves their argument content (soundness).

References

1. *A Guide to the Project Management Body of Knowledge: PMBOK Guide*. PMBOK Guides. Project Management Institute, 3rd edition, 2004.
2. A. Abran, P. Bourque, R. Dupuis, and J. W. Moore, editors. *Guide to the Software Engineering Body of Knowledge - SWEBOK*. IEEE Press, Piscataway, NJ, USA, 2004.
3. 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.
4. K. Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, Oct 1999.
5. J. Biggs and C. Tang. *Teaching for Quality Learning at University: What the Student does*. Mcgraw-Hill Publ.Comp., 3rd edition. edition, Nov. 2007.
6. J. B. Biggs and K. F. Collis. *Evaluating the quality of learning: the SOLO taxonomy (structure of the observed learning outcome)*. Academic Press, 1982.
7. B. W. Boehm. A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–72, 1988.
8. J. S. Brown, A. Collins, and P. Duguid. Situated cognition and the culture of learning. *Educational researcher*, 18(1):32–42, 1989.
9. 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.

10. 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.
11. A. Drappa and J. Ludewig. Simulation in software engineering training. *Proceedings of the 22nd international conference on Software engineering - ICSE '00*, pages 199–208, 2000.
12. R. A. Duschl. Assessment of inquiry. *Everyday assessment in the science classroom*, pages 41–59, 2003.
13. 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):915933, 2004.
14. A. Fuggetta. Software process: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 25–34. ACM, 2000.
15. N. Herrmann. The creative brain. *The Journal of Creative Behavior*, 25(4):275295, 1991.
16. P. Hersey, K. H. Blanchard, and D. E. Johnson. *Management of organizational behavior: leading human resources.* Prentice Hall, Upper Saddle River, N.J., 2001.
17. T. Hill and R. Westbrook. SWOT analysis: It's time for a product recall. *Long Range Planning*, 30(1):46–52, Feb. 1997.
18. J. M. Keller and K. Suzuki. Use of the arcs motivation model in courseware design. 1988.
19. M. I. Kellner, R. J. Madachy, and D. M. Raffo. Software process simulation modeling : Why ? What ? How ? *Journal of Systems and Software*, 46, 1999.
20. 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):314342, 2002.
21. C. Larman and V. R. Basili. Iterative and incremental development: A brief history. *Computer*, 36(6):47–56, 2003.
22. A. Lawson. The nature and development of hypotheticopredictive argumentation with implications for science teaching. *International Journal of Science Education*, 25(11):1387–1408, 2003.
23. 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.
24. R. J. Madachy. *Simulation.* John Wiley & Sons, Inc., 2002.
25. J. A. McCall, G. Wong, and A. Stone. A simulation modeling approach to understanding the software development process. In *Fourth Annual Software Engineering Workshop, Goddard Space Flight Center Greenbelt, Maryland*, 1979.
26. M. Melis, I. Turnu, A. Cau, and G. Concas. Evaluating the impact of test-first programming and pair programming through software process simulation. *Software Process: Improvement and Practice*, 11(4):345–360, 2006.
27. E. Navarro. *Simse: A Software Engineering Simulation Environment for Software Process Education.* PhD thesis, Long Beach, CA, USA, 2006. AAI3243955.
28. E. O. Navarro. A survey of software engineering educational delivery methods and associated learning theories. 2005.
29. E. O. Navarro and A. van der Hoek. Comprehensive evaluation of an educational software engineering simulation environment. In *Software Engineering Education Training, 2007. CSEET '07. 20th Conference on*, pages 195–202, 2007.
30. 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 . . .*, pages 367–395, 2003.

31. M. Prensky and M. Prensky. *Digital Game-Based Learning*. Mcgraw Hill, 2001.
32. 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.
33. D. Rodriguez. e-Learning in project management using simulation models: A case study based on the replication of an experiment. *Education, IEEE ...*, 49(4):451–463, 2006.
34. W. W. Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.
35. 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):447472, 2008.
36. W. A. Sandoval. Conceptual and epistemic aspects of students' scientific explanations. *Journal of the Learning Sciences*, 12(1):5–51, 2003.
37. R. Schank and H. Saunders. *Virtual learning: A revolutionary approach to building a highly skilled workforce*. Wiley Online Library, 2001.
38. 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.
39. S. E. Toulmin. *The uses of argument*. Cambridge University Press, Cambridge, U.K.; New York, 1958.
40. C. von Wangenheim and F. Shull. To game or not to game? *Software, IEEE*, pages 92–94, 2009.
41. A. Zapalska, D. Brozik, and D. Rudd. Development of Active Learning with Simulations and Games. *US-China Education Review*, 2:164–169, 2012.
42. A. Zohar and F. Nemet. Fostering students' knowledge and argumentation skills through dilemmas in human genetics. *Journal of Research in Science Teaching*, 39(1):3562, 2002.
43. O. Zuber-Skerritt. A model of values and actions for personal knowledge management. *Journal of Workplace Learning*, 17(1/2):49–64, Jan. 2005.