Specification By Example for Educational Purposes

Isabelle Blasquez, Hervé Leblanc

Specification By Example for Educational Purposes

Isabelle Blasquez
Limoges University
33 rue François Mitterrand
Limoges, France
isabelle.blasquez@unilim.fr

Hervé Leblanc
IRIT
118 Route de Narbonne
Toulouse, France
leblanc@irit.fr

ABSTRACT

The Specification By Example (SBE) is a guideline for building the *right software*, a software that meets customer requirements. It is based on seven process patterns and enhances communication and collaboration and it usually is used in agile software development. The connection between education and agile software development sounds actually as an emergent topic. In this paper, we propose to structure a teaching approach in analogy to an agile software developement by transposing each process pattern of SBE to a corresponding one in the teaching domain. Moreover, we show that thanks to the emergence of a collective intelligence process, the students are more confident and more responsible. Such a course offers the opportunity to learn not only technical skills, but also some values in a new mindset.

Keywords

Agile Software Development; Specification by Example; Agile Teaching

1. INTRODUCTION

"Building the product right and building the right product are two different things. We need both in order to succeed". From this observation, Godjko Adzick builds a collective knowledge by studying over 50 software projects [1]. The related process consists of gathering examples to clarify requirements, deriving tests and automating them. Specification By Example (SBE) is also defined as an approach of software development based on seven process patterns that help teams to build the right software product by writing just enough documentation to facilitate change effectively in short iterations or in flow-based development [1]. It also enhances communication and collaboration and it usually is used in agile software development.

The connection between agile software development and education sounds actually as an emergent topic. Some transpositions are developped concerning agile manifesto[1], agile values[2], the Scrum framework[3] and more generally agile approaches[4]. These propositions suggest the idea that education today needs a "refactoring". While the agile manifesto proposes to evolve from a traditional plan-driven paradigm to a value-driven paradigm, the agile school manifesto suggests an evolution from traditional teaching-approaches to new learning approaches.

We propose a teaching approach in analogy to an agile software developement by transposing each process pattern of SBE to a corresponding one in the teaching domain. The result of the transposition is a guideline usable by teachers to produce the *right course* as the agile methodologies are used to produce the *right software*.

As example, we use a software product methods course that we gave to eighty two-year french undergraduates. This course is a part of French National Pedagogical Program (PPN), a common program to all technical colleges specialized in Computer Technology. The objective of this course is to present software development processes. We have chosen to teach agile software development.

This paper presents a transposition of the seven process patterns of Specification By Example to improve the design of a course. This is one section per process pattern with three paragraphs each: an overview of the original pattern in SBE, the corresponding pattern in teaching-domain, and an example on the software development processes course. The next section gives some considerations about the concept of right course and gives a toolkit of our proposal.

2. BACKGROUND AND CONTEXT

Traditional teaching approach is teacher-centered and often relies only on lectures and small exercises. Nowadays, we remark that students become more quickly bored and inattentive due to mismatches between students learning and teaching styles [8]. To improve the quality of courses and to help both teacher and students for a better alignment, we have designed a guideline based on SBE.

A definition of quality in the context of educational game is suggested in [21] as *if it provides a positive learning effect, motivates students to study and provides a pleasant and engaging learning experience.* We choose to use this definition to the *right* course whose the expectations in terms of

DOI: http://dx.doi.org/10.1145/3059009.3059039

[1]https://www.infoq.com/articles/agile-schools-education
[2]https://pedagogieagile.com/2012/05/12/les-valeurs
[3]http://eduscrum.nl/en
[4]http://approchealpes.info

learning, collaboration, commitment and happiness to work are similar. The purpose of the *right* course is to enhance the individual student's capability to participate in and contribute to collaborative learning process based on collective intelligence process. According to Lévy, Collective Intelligence (CI) is the capacity of human collectives to engage in intellectual cooperation in order to create, innovate and invent [15].

The guideline is presented in the table 1. The first column provides original patterns of SBE. The second column provides corresponding teaching-domain patterns. And the last column provides practices used to implement the patterns. Teachers could design the right course by determining the right teaching resources as well as posssible. For students, this guideline gives the opportunity to learn not only technical skills, but also some values in a new mindset.

Throughout this paper, we propose an analogy between the agile software development terminology and the teaching terminology. As we have already suggested to translate *product* into *course*, we also suggest to translate *Product Manager* into *Course designer* and *requirements* into *syllabus* of the course. *Examples* from SBE will be transposed to *teaching resources*.

3. DERIVING SCOPE FROM GOALS

Original pattern.

A best practice for this pattern is to start with a customer's business goal and then to collaborate with the business to derive the scope. This pattern also first focuses on *why* something is needed and *who* needs it. For effectively collaborating, it recommends to ask *how* something would be useful before *what* you need to build it. "Impact Mapping" is a technique for deriving scope from goals which propose to build a minmap around four aspects of the software: goal (why), actors (who), impacts (how), and deliverables (what). [2]

Corresponding teaching-domain pattern.

This process pattern focuses on the course's vision by answering *why*, *who*, *how*, and *what*. The *why* helps to define the objective of the course also called learning goals. The *who* depicts actors, usually students. The *how* identifies impacts to help actors to achieve the objective. The impacts of the teaching domain are learning outcomes (knowledge and competencies acquired or improved by students). The *what* helps to outline deliverables needed to support the impacts. For teaching, they are *topics* which are the transposition from *features*.

Example.

Our objective is to propose an introduction to agile software development in an agile mindset: *Doing agile* and *Being agile* with respect to the first levels of Bloom's taxonomy suitable to undergradates (knowledge, comprehension and application) [3]. The actors are students. There are three learning outcomes relative to agile practices: delivering the right product, delivering the product right, and delivering fast and regularly. There is one learning outcome relative to agile values and principles. The scope is also determined by higher-level topics which are: introduction to agile software development, overview of collaborative tools, introduction to Scrum, from traditional to agile software development, business model, agile requirements, user story, Scrum in practice, tests, and retrospective.

As the students were novices, this course's vision has been suggested by the teacher. But in another context such as an advanced course, the course's vision could be defined collaboratively by an open backlog and a collective vote.

4. SPECIFYING COLLABORATIVELY

Original pattern.

Specifying collaboratively enables to harness the knowledge and the experience of the whole team. It also creates a collective ownership of specifications, making everyone more engaged in the delivery process [1]. No document should be written in isolation. To collaborate effectively, most popular models are introduced: workshops (all-team or smaller), pair-writing, and even informal conversations. All-team workshops are useful to discover and learn about the business model. They are a good way to build a shared understanding of the requirements and produce a set of examples that illustrate a feature. The smaller workshops help to clarify or complete the specification. Pairing to write specifications allows to mature products by getting several different perspectives on a example. Mostly, collaboration needs a preparation phase to be efficient. It is easier to initiate a collective discussion if some examples have been prepared before.

Corresponding teaching-domain pattern.

Specifying collaboratively transposes to Teaching collaboratively. In SBE, collaboration models are classified according to the size of the team. In teaching domain, collaboration models will be classified according to the teaching approach.

We propose two kinds of collaboration models: inductive and deductive workshops. Induction is a reasoning progression that proceeds from particulars (observations, measurements, data) to generalities (governing rules, laws, theories). Deduction proceeds in the opposite direction [8]. Traditional teaching approach is deductive. The inductive approach includes problem-based learning, discovery learning, inquiry learning, or some variation on those themes which are characterized by [19] as constructivist based approaches. These approaches impose more responsibility on students for their own learning. They almost always involve students discussing questions and solving problems (active learning) and the work is done in groups (collaborative or cooperative learning).

The Participatory Action Research (PAR) is a collaborative process of research, education and action explicitly oriented towards social transformation [12]. To promote the emergence of a collective intelligence process, we propose to structure the workshops around a AAA-PAR strategy that consists in three ordered steps. The first step is the Arrangement time: the preparation phase where teacher prepares materials provided to the students to start the workshop. The second step is the Action time: the core of the workshop. During this step, the students collaborate to achieve the required learning goals. The self-organized *development team* of an agile software development also transposes to an autonomous *student team*. The behavior of the stakeholders is a crucial factor for the success of these workshop. The main teacher's responsibility is to engage students in learn-

Table 1: Guideline for delivering the right course

Process Patterns	Teaching Patterns	Practices
Deriving scope from goals	Course's vision	goal, skateholders, skills, main topics
Specifying collaboratively	Teaching collaboratively	inductive workshop, deductive workshop structure of the workshop
Illustrating using examples	Illustrating with teaching resources (teaching resources support)	learning style teaching style
Refining the specification	Refining teaching resources (teaching resources details)	competency, timeboxing, application domain outcome, starter kit
Automating validation without changing specification	Teaching resources management	schedule, repetition
Validating frequently	Knowledge evaluation process	with or without grades, individual or collective material, frequency, ...
Evolving a documentation system	Evolving a documentation academic	access, material, architecture

ing: he becomes the facilitator of the workshop [19]. He must ensure trust behaviors. Thirteen trust behaviors have been identified by [5], some of them are: demonstrate respect, create transparency, listen first, keep commitments, and extend trust. The third step is the Assertion time: a kind of workshop review. The teachers also encourages students to explicitly reflect on the events of the Action time and to examine the lessons learned about the workshop. An agile retrospective [6] is well-adapted at this time and the three following questions could be asked: What did you learn in this workshop ? What is amazing in this workshop ? Why will you reuse or not the workshop later ?

Example.

Both inductive and deductive workshops are used in our course. Inductive workshops was preferred to respect the CI process and to promote the emergence of agile values. The teams are composed by six or seven students to respect the ideal size of an agile team.

5. ILLUSTRATING USING EXAMPLES

Original pattern.

Examples are used to clarify meaning in everyday conversation: they are concrete and less unambiguous. Illustrating requirements using examples is a way to specify with enough details that we can be checked by assertion. Using examples will ensure that the delivery teams focus on the right product and that they have a shared understandings of what the business users expect out of the system [1]. To be used from requirement analysis to testing, examples should be small, precise, realistic and easy to understand.

Corresponding teaching-domain pattern.

Illustrating with examples tranposes to illustrating with teaching resources. We define a *teaching resource* as an activity used by a teacher to engage students in learning to achieve required learning goals.

Learning is presented as a two-step process involving the reception and processing of information [8]. The processing step may involve different learning's models: simple memorization, inductive or deductive reasoning, reflection, action, and introspection or interaction with others. As students learn in many ways and as teaching methods are also various, learning styles and teaching styles are identified by [8]. A learning style should consider: perception (sensory or intuitive), input (visual or auditory), organization (inductive or deductive), processing (active or reactive), and understanding (sequential or global). A teaching style should consider: content (concrete or abstract), presentation (visual or verbal), organization (inductive or deductive), student participation (active or passive), and perspective (sequential or global).

When mismatches exist between learning and teaching styles, students become bored and inattentive in class, do poorly on tests, and get discouraged about the courses [8]. To engage students in learning, a best practice is to choose the right teaching resources support by finding the better alignment between learning and teaching style. Sometimes, lectures are necessary to introduce or clarify a concept. For a better alignment of the right course, we suggest two others potential teaching resources.

Gamification is the process of using game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning, and solve problems [11]. Game can also offer some moment of serendipity. The Assertion time is crucial to have benefits from games. A link between games, culture, happiness, learning, and productivity is shown in [16] and studied on an educational game in [20]. Innovation Games [10] or Game storming [9] give usefull examples of gamification.

Project Based Learning (PBL) is perceived to be a student-centered approach to learn [4, 18]. The students need to produce a solution to solve a problem and an outcome in the form of a report. PBL focuses on large, open-ended problems, like many real-world problems [17]. It is based on five principles: students work together in groups; a real world problem that affects the life of the students is presented for investigation; students discuss findings and consult the teacher for guidance, input, and feedback; the maturity level of students skills determines the degree of guidance provided by the teacher; resulting products can be shared with the community.

Example.

Various teaching resources have been used : traditional lectures, games (lego-based approaches as Lego4Scrum or TDDLego[14]), collaborative workshops (story-writing workshop and Coding dojo). A PBL approach has also been adopted including collaborative workshops (product vision statements, story mapping, impact mapping) and innovation games (Product Box, Speed Boat). Gamification has been preferred to introduce concepts whenever possible. Videos from professionnal conferences have been watched in some lectures. To introduce the visual management, visual information has often been used with pictures, animations and sketchnotes.

6. REFINING THE SPECIFICATION

Original pattern.

This pattern brings further informations about the specification. To be unambiguous, a good specification should be precise and testable, and concerns only business functionality. To be useful as long-term documentation, it should be self-explanatory, focused, and in domain language.

Corresponding teaching-domain pattern.

This pattern focuses on teaching resources details. It helps the teacher to refine the teaching resources and to improve it. A teaching resource should be focused about a specific competency and be well time-boxed to respect the duration of the workshop. To be precise, the application domain must be carefully chosen to promote the commitment of students. To be testable, an outcome must be defined. An outcome describes a way to verify and validate that the required learning goals are well-achieved. The *tests* in software development transposes to the *outcome* in teaching domain. Collective discussions induced by the teaching resource will be efficient only if everyone has a common understanding of it. A starter kit could prevent misunderstanding and promote self-organization of the activities.

Example.

Each teaching resource has been focused on a specific competency and time-boxed to respect the duration of a session. Each student team has choosen by collective vote his own application domain for the project used for the PBL workshops. At least, two major benefits can be highlighted with this kind of project. First, the product manager (a student) is always avalaible. Then most of students feel involved because they could be quickly become the users of this application. A starter kit has been provided for each new teaching resources. It includes a roadmap, a brief summary of the concept taught, a description of required deliverable, a glossary, and bibliographical references. The outcome have been various as oral feedback, photos, and specific artifact or summary.

7. AUTOMATING VALIDATION WITHOUT CHANGING SPECIFICATIONS

Original pattern.

After refining the specification, the examples can be used as a target for the implementation and the validation of the software. The tests should often be run during the development to ensure quality of the product and to reduce delays of the feedback. This pattern focuses on automation, as a solution of a quick feedback. The automation has long-term benefits: having an objective measurement of when the job is finished, checking more frequently, and getting a living documentation [1].

Corresponding teaching-domain pattern.

Thinking about automation is asking about repetition of a learning resource. The repetition is a well-known best practice in teaching. By using different types of teaching resources on a same competency, we prevent the disalignment between teaching and learning styles, we respect the learning time of each student, and we offer different opportunities to apply the competency. This pattern focuses on teaching resources management. It helps to select suitable teaching resources by considering learning goals, learning styles and competency. A first sequence of teaching resources can be scheduled according to the sequence of topics. To ensure repetition of some competencies, new teaching resources can be introduced. During the course, a sequence can be changed by the teacher according to the feedback on student behaviors and feelings (motivation, comprehension, outcome validation, ...). The teacher should also apply some values to his behavior, as responding to change in his course's design.

Example.

The course run over a period of 10 weeks with two 2-hours sessions per week. Lectures, games, collaborative workshops and PBL have been alternatively used. As students have a mentored software development project to lead at the same time, some workshop of PBL have been repeated in this context to help students to find user-stories from the vision. These workshops were self-organized by the student teams.

8. VALIDATING FREQUENTLY

Original pattern.

A continuous integration system builds the product and runs the tests. It ensures that once the product is built right, it stays right. To satisfy this point, this pattern suggests to validate executable specifications frequently to keep them reliable. The best practices insist on reducing unreliability and on looking for ways to get faster feedback.

Corresponding teaching-domain pattern.

Mostly, the actors build their own knowledge during the right course. The teacher is responsible for the reliability of the knowledge. The students are responsible to get faster feedback. This pattern suggests to determine the knowledge evaluation process. This process must be defined by considering some questions as: is it an individual or collective evaluation ? What is the material for the evaluation ? How often should we evaluate ? Are grades really required as a measure of academic performance ? and so on.

Example.

An evaluation has been planned at the end of each workshop. This evaluation has taken place during the Assertion time with an oral presentation of the outcomes. No grade is attributed for this. For each workshop of the PBL, a summary has been required per team as material for the evaluation. This summary was based on a template provided by the teacher: presentation of the workshop, deliverables, comments, and retrospective. It allows to detail knowledge, comprehension, and application of the competency taught. Such a summary could also be used as a cookbook to apply easily the competency again. At the end of the course, a report including a presentation of the PBL project and all the summaries has been delivered by each team. These reports have been graded by one grade by team to respect the collective intelligence process.

9. EVOLVING A DOCUMENTATION SYSTEM

Original pattern.

Living documentation is an artifact and the end-product of SBE. It is a reliable and authoritative source of information on system functionality, which anyone can easily ac-

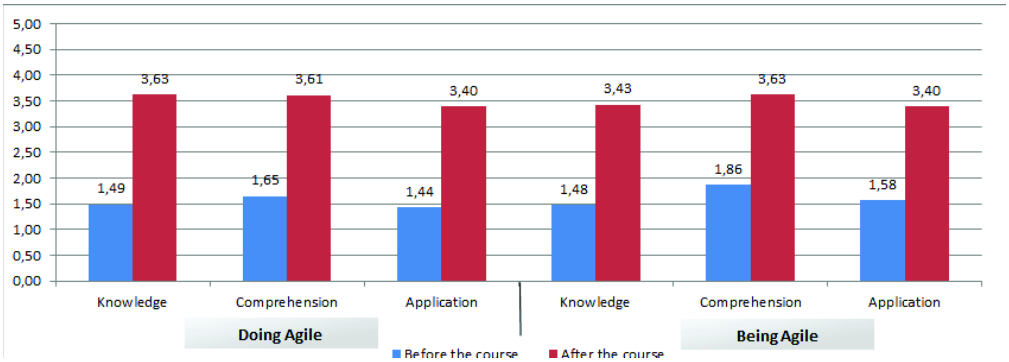

Figure 1: Frequency diagram of answers with respect to the feedback on learning.

cess [1]. Best practices dedicated to living documentation are: easy to access, easy to understand, and each change in the system needs to be reflected.

Corresponding teaching-domain pattern.

System documentation transposes to academic documentation in teaching domain. To easily access to the documentation, material courses (lectures, exercices, tutorials, references) can be deposit in repositories managed by a version control system. Each stakeholder (teacher and students) can consult or tell about changes to update material courses. The role of the teacher is to encourage students to share and update the materials to have more understable course notes. The documentation can be alive by setting up an automatic notification system to alert all the stakeholder when a new document is added or updated.

Example.

Github is used to share on-line public material courses[5]. It provides collaboration features such as pull request or wikis. A Slack[6] team has been created for this course to facilitate discussions between skateholders. The web-service IFTTT[7] connects Github with Slack to automatically notify all the stakeholder when a change in the documentation is pushed.

10. VALIDATION

We focus now on the evaluation of the *quality* (as defined in section 2) of the agile software project management course presented as the example. This course has been designed in 2015 and delivered in the fall 2015 and 2016.

The validation has been adapted from a specific framework [20] which is based on two questionnaries related to Bloom's taxonomy of educational objectives [3] and Kirkpatrick's levels of evaluation [13]. It has already been applied for educational games and coding dojo session [7, 21]. We adapted the terminology from game to course and we deleted some items from original questionnaires to only focus a set of teaching resources. Moreover, the evaluation is concerned by student perception in terms of motivation, user experience, and learning process. Questionnaries are given to the 80 students at the end of the course.

Results of the first questionnaire are presented in Fig. 1. It is based on [3] and evaluates the perception of the evolution of learning in the competencies taught before and after the course. It focuses on the learning goals (*doing agile* and *being agile*) with respect to the perceived impacts rated on a scale from 1 to 5. The perception of the Agile Software Development (the heart of the course) was multiplied by more than two. It is the same for the *being agile* posture that students can reused for other courses.

Results of the second questionnaire are presented in Fig. 2. It is based on [13] consists in 21 items asking motivation, user experience, and learning on a Likert scale with response alternatives ranging from strongly disagree (-2) to strongly agree (2). The majority of the students agreed strongly that the course promotes moments of cooperation. They also confirmed that they had fun while interacting with other students. The social interaction has also been the highest rated dimension. Overall, results are positive in terms of fun, challenge, and social interactions.

11. CONCLUSION

This paper presents a guideline for delivering a *right* course by structuring a teaching approach as an agile software development. We defend a new teaching way based on collective intelligence process which aims to align a course with the needs of students by designing the right teaching resources. The role of the teacher as a facilitator has also been presented as a key of success. By maintaining trust behaviors and by encouraging the communication, he helps students to collaborate efficiently and to be more commitment, more creative and more responsible. In agile software development, the most popular methodologies used are: Scrum and eXtreme Programming. We showed that SBE can be transposed to specify teaching resources. Scrum has already been transposed to manage learning experience where the responsibility for the learning process is delegated to students. No transposition has already been proposed for eXtreme Programming. We plan to study their principles, especially the transposition from a user story to a teaching story.

12. ACKNOWLEDGMENTS

This same transposition was used for a specific course during 3 days on agile requirements at the University of Toulouse for a professional Bachelor dedicated to Development and Software Quality.

[5]https://github.com/iblasquez
[6]https://slack.com: a cloud-based team collaboration tool
[7]https://ifttt.com: *If This Then That*

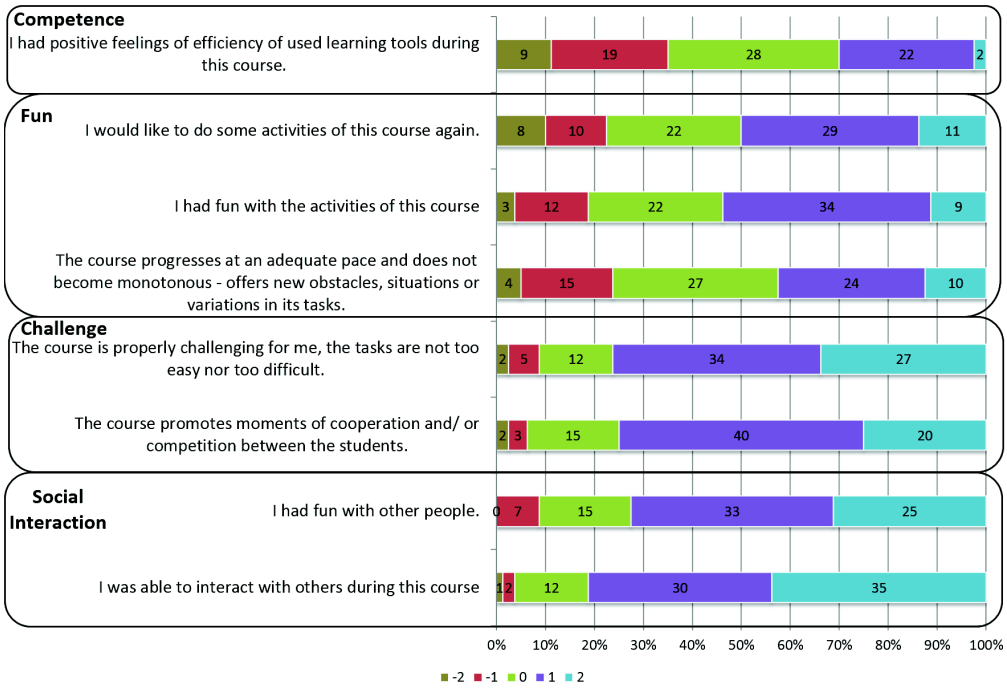

Figure 2: Frequency diagram of answers with respect to the sub-component user experience.

13. REFERENCES

[1] G. Adzic. *Specification by Example: How Successful Teams Deliver the Right Software*. Manning Publications Co., Greenwich, CT, USA, 2011.

[2] G. Adzic and M. Bisset. *Impact Mapping*. Provoking Thoughts, 2012.

[3] B. S. Bloom, M. B. Engelhart, E. J. Furst, W. H. Hill, and D. R. Krathwohl. *Taxonomy of educational objectives. The classification of educational goals.* Longmans Green, 1956.

[4] S. Chandrasekaran, A. Stojcevski, G. Littlefair, and M. Joordens. Learning through projects in engineering education. In *Proceedings of SEFI Conference*, 2012.

[5] S. Covey and R. Merrill. *The SPEED of Trust: The One Thing that Changes Everything*. Free Press, 2008.

[6] E. Derby and D. Larsen. *Agile Retrospectives: Making Good Teams Great*. Pragmatic Bookshelf, 2006.

[7] B. Estácio, N. Valentim, L. Rivero, T. Conte, and R. Prikladnicki. Evaluating the use of pair programming and coding dojo in teaching mockups development: An empirical study. In *HICSS*, pages 5084–5093. IEEE Computer Society, 2015.

[8] R. M. Felder and L. K. Silverman. Learning and teaching styles in engineering education. *engineering education*, 78(7):674–681, 1988.

[9] D. Gray, S. Brown, and J. Macanufo. *Gamestorming: A Playbook for Innovators, Rulebreakers, and Changemakers*. O'Reilly Media, 2010.

[10] L. Hohmann. *Innovation Games*. Pearson Education, 2006.

[11] K. Kapp. *The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education*. Wiley, 2012.

[12] S. Kindon, R. Pain, and M. Kesby. *Participatory Action Research Approaches and Methods: Connecting People, Participation and Place*. Routledge Studies in Human Geography. Taylor & Francis, 2007.

[13] D. L. Kirkpatrick and J. D. Kirkpatrick. *Evaluating training programs : the four levels*. Berrett-Koehler Publishers, 2006.

[14] S. Kurkovsky. A lego-based approach to introducing test-driven development. In *ACM Conference on ITiCSE*, pages 246–247, New York, NY, USA, 2016.

[15] P. Lévy. From social computing to reflexive collective intelligence: The ieml research program. *Information Sciences*, 180(1):71–94, 2010.

[16] D. Mezick. *The Culture Game: Tools for the Agile Manager*. FreeStanding Press, 2012.

[17] E. D. Ragan, S. Frezza, and J. Cannell. Product-based learning in software engineering education. In *IEEE Frontiers in Education Conference*, pages 1–6, 2009.

[18] J. R. Savery. Overview of problem-based learning: definition and distinctions, the interdisciplinary. *Journal of Problem-based Learning*, pages 9–20, 2006.

[19] K. Smith, S. Sheppard, D. Johnson, and R. Johnson. Pedagogies of engagement: Classroom-based practices. *Journal of Engineering Education*, 94(1):87–100, 2005.

[20] C. G. von Wangenheim, R. Savi, and A. F. Borgatto. Deliver! - an educational game for teaching earned value management in computing courses. *Inf. Softw. Technol.*, 54(3):286–298, 2012.

[21] C. G. von Wangenheim, R. Savi, and A. F. Borgatto. Scrumia : An educational game for teaching scrum in computing courses. *Journal of Systems and Software*, 86(10):2675–2687, 2013.