An Experimental Card Game for Software Testing

Development, Design and Evaluation of a Physical Card Game to Deepen the Knowledge of Students in Academic Software Testing Education

Alexander Soska, Jürgen Mottok
LaS³ - Laboratory for Safe and Secure Systems
OTH Regensburg
Regensburg, Germany
{alexander.soska; juergen.mottok}@oth-regensburg.de

Christian Wolff
Lehrstuhl für Medieninformatik
Universität Regensburg
Regensburg, Germany
christian.wolff@sprachlit.uni-regensburg.de

Abstract—**Teaching software testing is a challenging task. Especially if you want to impart more in-depth and practical knowledge to the students. Therefore, most lectures still teach in a classic lecture format despite the fact that this way of instruction is in any case the optimal way of instruction for today's requirements anymore. In this paper we present our implementation of an active learning method to deepen the knowledge in academic software test education. We describe a card game for advanced learning that promotes students' collaboration and knowledge exchange in a playful and competitive manner. The design of the game is based on constructive and cooperative theories. A subsequent evaluation shows that the use of this card game for teaching software testing is a suitable method.**

Keywords – software testing; playful learning; game-based learning; card game

I. *Introduction*

Software engineering is a field that is rapidly growing in size. The complexity and amount of this profession is increasingly difficult to mediate in academic education. On the one hand, the topics are too widespread and abstract that learners instantly understand the contents. On the other hand, practical experience is crucial to comprehend the methods and procedures that are applied in this profession. Industry has recognized this problem and over time has requested academia to better prepare its students for their future occupation in this field. Students still lack of required skills, technical as well as soft skills [1]. At that point, industry also criticizes that students have no deeper understanding.

In response to this, the academic world tries to bridge this gap and adjusts the learning methods to fulfill the demands of the industry. One of the most common approaches is to increase the amount of self-determined learning elements. One popular method is active learning. Active learning is a process whereby students engage in activities, such as reading, writing, discussion, or problem solving that promote analysis, synthesis, and evaluation of class content. Cooperative learning, problem-based learning, and the use of case methods and simulations are some approaches that promote active learning. This focuses the learner's self-responsibility to their own learning process. This engages learners in two aspects – doing things and thinking about things they are doing [2]. There is a diverse range of implementations for this learning method, for example learning through play, group work, or project method.

In this paper we present a prototypical implementation of active learning to deepen the knowledge in academic software testing education. We describe a card game that promotes students' collaboration and knowledge exchange in a playful and competitive manner. By integrating this game into the lecture, we propose that

- students are exchanging their experiences and knowledge with each other,
- students adopt the gameplay to meet their individual learning progress,
- students get motivated to play again and increase their knowledge level
- and students deepen their knowledge by explaining their knowledge to fellow students.

After a short introduction into game-based learning in chapter II, we outline the current state of the art of card games in software engineering education in chapter III. Chapter IV describes the theoretical and didactical fundamentals for designing our game. The actual description of the game design and its rules as well as its gameplay follows in the chapter V. Furthermore, we present the results of an experiment with a group of students in chapter VI before giving a short conclusion in chapter VII.

II. *Game-Based Learning*

Learning occurs with repetition and practical application. Learners need to be engaged with the learning content and fill their lack of knowledge in a self-active manner. Therefore, the lecturer should create environments in which learning can take place. To get the students involved and motivated, a common approach is to combine education with entertaining elements [3]. *Game-based Learning* (GBL) describes a method where students explore relevant aspects of learning in a gamified context, designed by the lecturer. Evidence exists suggesting that educators should consider using GBL as part of their courses, also in software engineering [4]. In addition, based on subjective feedback, many studies report that learners prefer gamified activities compared to traditional instructional methods [5]. Students get engaged into the context-specific environment and have feelings of engagement, discovery and fun while acquiring knowledge about the subject matter. Therefore, GBL uses competitive exercises, either pitting the students against each other or getting them to challenge themselves in order to motivate them to learn better. Games are designed to create a compelling complex problem space or world, which players come to understand through self-directed exploration. They are scaffolded to deliver just-in-time learning and to use data to help players understand how they are doing, what they need to work on and where to go next. Games can create a compelling need to know, a need to ask, examine, assimilate and master certain skills and content areas. Furthermore, there are other attributes of games that facilitate learning. One of these is the state of being known as *play* (gameplay experience) [6]. Much of the activity of play consists in failing to reach the goal established by a game's rules. And yet players rarely experience this failure as an obstacle to trying again and again, as they work toward mastery. There is something in play that gives players permission to take risks considered outlandish or impossible in "real life." There is something in play that activates the tenacity and persistence required for effective active learning.

However, there is much debate about the long-term effect of games. Some believe that they have no lasting effect — that they just serve as a momentary distraction. Others believe that games are so good for us that they will become the cornerstone of the 21st century education [7]. Some experts argue that games are, first and foremost, learning systems, and that this accounts for the sense of engagement and entertainment players experience. In order to create a truly educational game, the instructor needs to make sure that mastering the content is essential to scoring and winning. Therefore, a big challenge is how to integrate the learning objectives into the game. Especially the way games are designed and integrated into the lecture. Thereby we need to admit that games alone are not sufficient pedagogical devices to teach domain specific content and must be supplemented by additional methods. We suggest that games in education are good methods to strengthen or deepen knowledge and not to acquire knowledge. This is supported by researchers that found that the use of games as a complement to traditional teaching is much more efficient than as a single method of teaching [8,9,10].

III. Current Games in Software Engineering Education

Several examples for the implementation of games in software engineering education already exist. Over the last few years, several games were proposed as a method for learning software engineering. Games like *SIMSE*, *PlayScrum*, and *Problems and Programmers* were created in order to improve the quality of the learning process in software engineering related topics [11,12,13]. Research shows that overwhelmingly most of the games are created in the areas of project management or requirements engineering. A similar example for software testing does not exist, yet. Furthermore, the majority are implemented as digital simulation games or role playing games. There are just a couple of card games, for example *Problems and Programmers* or *PlayScrum*. These are competitive card games in which each student plays the role of for example a project manager. Here the player tries to use given tasks and processes in order to finalize his or hers project quicker than the other players. A main criticism of this approach is that these games do not involve all participating learners. Most of these games only have single player elements which reduces the development of learning in collaborative ways. Since the goal of these games is to be the first to win, the players respectively learners do not try integrate the other players into their progress. Furthermore, there is no approach that has a systematic adoption of knowledge levels to the tasks and activities in a game [14].
In the following, we present two current games more in detail.

A. Problems and Programmers

Problems and Programmers [13] is a competitive card game, in which each student plays the role of a project manager with the same project goals. The player who finishes first is the winner. To finish the project, the players manage their budget, meet customer requirements and produce high quality software. Basically, they should follow the best practices of software engineering in order to avoid any obstacle. The software development method followed in the game is based on exchangeable process models. The gameplay goes through the different stages of this model (analysis, design, development, integration and test). Players begin to create a column with analysis cards, then building a column of design cards. They continue by playing code cards. While some players will place great emphasis on the analysis and design phases and carefully inspect the code others will rush through all these stages in order to deliver the project as soon as possible.

B. PlayScrum

The card game *PlayScrum* [12] is a competition card game. Every player has the role of a Scrum Master in a software project development. PlayScrum can be played by two to five players. The game is divided into sprints that differ from project to project. During each sprint, each player must develop a number of tasks defined at the start of the game. *PlayScrum* includes one board for each player, product backlog cards that determine the characteristics of the project, problem cards launched by opponents of a player, concept cards used by the player, as a remedy to problems launched by his opponents, developer cards corresponding to the development members of the team of a player, artefacts that represent the tasks performed by the team during the project and a die. The winner is the player who first performs all tasks defined for the project without errors or the player who has the highest percentage of tasks without errors, after the end of the last iteration.

IV. Theoretical and Didactical Fundamentals

Despite the current approaches, the design and use of games in software engineering education is expendable. As highlighted in chapter III, there some experiments in developing and integrating games in academic software engineering education have taken place. The usual development of these games starts with an idea about the field of application and a breakdown of learning objectives to game challenges. But especially this breakdown and conversion to game elements is a difficult task. Most of the games lack an adequate realization and implementation of didactical fundamentals.

A. Shift from Teaching to Learning

Students have to learn large amounts of knowledge and construct this knowledge from the retrieved information. Shifting this emphasis from teaching to learning can create more interactive and engaging learning environments. Thereby the role of both teachers and learners change. The role of the teacher changes from *knowledge transmitter* to that of *learning facilitator*. The new role does not diminish the importance of the teacher but requires new knowledge and skills. Students will have greater responsibility for their own learning in this environment as they seek out, find, synthesize, and share their knowledge with others. For this, classical learning concepts must be overcome and complemented or even replaced by appropriate methods. This shift from teaching to learning [15] enables students to acquire knowledge in engaging and more motivated forms.

B. Constructivism

To support the shift to students' learning, fundamental models of learning must be examined. One possible learning approach is constructivism. This theory is based on the belief that learning occurs as learners are actively involved in a process of meaning and knowledge *construction*. Mascolol and Fischer state that "Constructivism is the philosophical and scientific position that knowledge arises through a process of active construction" [16]. Learning is integrated in an active and constructive process. Learners construct their meaning and knowledge. Constructivist teaching fosters critical thinking and active learning. Because all learning is filtered through pre-existing knowledge, constructivists suggest that learning is more effective when a student is actively engaged in the learning process rather than attempting to receive knowledge passively. This meets with our suggestions made in chapter II as we stated that games are most suitable when used for more in-depth knowledge.
A wide variety of methods claim to be based on constructivist learning theory. Most of these methods rely on some form of guided discovery where the teacher avoids most direct instruction and attempts to lead the student through questions and activities to discover, discuss, appreciate, and verbalize the new knowledge. This knowledge is constructed based on personal experiences and hypothesis of the environment. Learners continuously test these hypotheses through social negotiation. Thereby each student has a different interpretation and develops his or her knowledge in exchange with others. The design of appropriate methods that are applicable is a crucial task for the knowledge acquisition.

According to Audrey [17] the characteristics of a constructivist classroom are that

- the learners are actively involved,
- the environment is democratic,
- the activities are interactive and student-centered and
- the teacher facilitates a process of learning in which students are encouraged to be responsible and autonomous.

For instance, in group investigations they need to choose and research a topic from a limited area. They are then held responsible for researching the topic and presenting their findings to the class. More generally, group learning should be seen as a process of peer interaction that is mediated and structured by the teacher. This methodical element can be promoted in games by the possibility to work together on specific tasks. This is guided by means of effectively directed questions, the introduction and clarification of concepts and information, and references to previously learned material.

C. Constructivistic Framework for the Gameplay of Card Games

For the development of our card game, we need to develop a framework that implements constructivistic attributes. To identify which game elements correspond to constructivism, we identified relevant *game design patterns* [18] and sorted them. Therefor we suggest the card game representations in Table I.

TABLE I: CONSTRUCTIVISTIC FRAMEWORK FOR CARD GAMES

Constructivist Characteristics	**Game Design Pattern Scope**	**Realization Examples**
Active Involvement	Goals of Arrangement, Persistance, Information and Knowledge	Rescue, Collection, Guard, Gain Information, Gain Competence
Democratic environment	Common decisions	Player Decided Results;
Student-centered interactive activities	Collaboration; Group Activities; Stimulated Social Interaction	Cooperation; Team Play; Alliances, Social Interaction; Trading; Bidding;
Autonomous and self-responsibility	Creative Control; Narrative Structures	Varied Gameplay; Freedom of Choice; Character Development; Identification; Progresses

Active involvement lures the learner out of his role as an observer and forces him or her to actively deal with learning content. To get learners involved, they need to identify with the goals presented in the game. Thus they get motivated to pursue the tasks and activities presented. In order to achieve this, we need to design goal structures that are accomplishable. Goals that promote conflicts or any dispute between the players need to be avoided. Goals like *Collection* (*of Rewards*) keep the players out of these social dilemmas.
In *democratic environments* all participants are valuable for common decisions. The participants solve conflicts together and decide in collaboration about specific elements. Thereby they often need to change views and actively discuss to solve conflicts. Therefore the different point of views must be explained and approaches for a common agreement must be

made. In a card game this can be realized by deciding about the reward points between all players. The decision about the acceptance of an answer has to be made together with the other non-players and the answer needs to be accepted.

Collaborative tasks promote learners to develop teamwork skills and individual participation as essentially related to the success of group learning. Providing learners with opportunities to use and share their previous knowledge as a basis for the construction of new knowledge is considered crucial. It is important to consider the design of appropriate help, feedback and hint structures [19]. This can be done in various ways, including asking questions, explaining or augmenting the game [20]. Cooperative learning is the instructional use of small groups so that students work together to maximize their own and each other's learning. Carefully structured cooperative learning involves people working in teams to accomplish a common goal, under conditions that involve both positive interdependence and individualism. For a card game, an exemplary realization can be to integrate team play. Thereby players need to solve tasks and overcome the challenges presented together.

In addition, the learner needs to develop self-awareness for his or her learning. The student needs to feel self-responsibility and accountability for their own learning process. The learner must autonomously decide, filter and construct knowledge from all the available information. This decision-making ability is crucial for the acquirement of knowledge. An example for this is to give the player freedom of choice. This promotes the ability of the learner to choose between different strategies or paths.

The elements described in Table I constitute a constructive framework for card games emphasizing participation, challenge and interactivity. The main goal still is to win the game, but to reach this the players need to examine which strategy, partnership and information fits best. In addition they need to collaborate with each other to solve in-game activities.

D. *Scaffolding*

According to constructivism, the lecturer needs to prepare suitable environments in which the learner can acquire knowledge. This scaffolding or instructional scaffolding to guide students' learning has been widely studied in the past [21].

Instructional scaffolding [22,23] or simply known as scaffolding in education is defined as a guidance or support from teachers, instructors or other knowledgeable persons that facilitate students to achieve their goals in learning. Conceptually, scaffolding means providing students with instructions during the early stage of learning before slowly shifting the responsibility to them as they develop their own understanding and skills. As technology extends learning from classroom to learning communities, same goes to the concept of scaffolding.

An instructor can provide a variety of scaffolds to accommodate different levels of learning. The following enumeration [24] outlines a few common scaffolding strategies, which can also be used for GBL:

1. Advanced Organizers: Tools that present new information or concepts to learners (diagrams, flow charts, …).
2. Modeling: Instructors demonstrate desired behavior, knowledge or task to students (Provide step-by-step instructions, …).
3. Worked Examples: A worked example is a step-by-step demonstration of a complex problem or task.
4. Concept Maps: Graphical tools for organizing, representing and displaying the relationships between knowledge and concepts
5. Explanations: Ways in which instructors present and explain new content to learners (Different constructivist learning arrangements).
6. Handouts: A supplementary resource used to support teaching and learning.
7. Prompts: A physical or verbal cue to aid recall of prior or assumed knowledge (E.g. instructional questions with class room response systems).

The enumerated scaffolding strategies in Table II can be used in GBL design for a physical card game.

TABLE II: SCAFFOLDING LINKED TO GBL DESIGN FOR A PHYSICAL CARD GAME

Scaffolding Strategy	***Physical Card Game***
Advanced Organizers	Guidelines, Recipes, …
Modelling	Guidelines, Recipes, …
Worked Examples	UML Diagrams
Concept Maps	UML Diagrams
Explanations	Guidelines, Recipes, …
Handouts	Guidelines, Recipes, …
Prompts	Repetition

E. *Integration of learning objectives into card games*

The integration of the learning objectives into a game must be well designed, appropriate, and well implemented. In fact, the educational content must be at the heart of game play, so that learners get engaged in the activities or tasks as they play the game [19]. The learning objectives should also be contextual to the game, in the sense that they must be perceived by the player to be an element of the game play. Thereby the constructivist design also emphasizes the fundamental concepts of the learning subject [25]. Methods for the design of educational card games have already been described [26]. Nevertheless, the systematic integration of different knowledge levels into the gamy by identifying game elements or mechanics is still expendable. Furthermore, while nowadays the use and application of learning content is crucial for the acquirement of knowledge and practical experience, research shows that the learning levels of current software engineering games mainly pitch at the first level of Bloom's taxonomy learning objectives, i. e. *knowledge* [4]. Only a marginal amount of games exists for deeper understanding (at the levels of, e. g., *analysis* or even *synthesis*. We want to overcome this deficit by proposing a systematic method of mapping learning objectives to card game activities and tasks.

a) EVELIN Classification Scheme

To solve typical problems in software engineering, the students need to learn different cognitive, technical and practical skills. Teaching in software engineering must therefore promote all necessary elements to approach these requirements. For this purpose, it is necessary that teaching and learning bases on suitable methods that help to identify and integrate learning objectives. In this paper, we focus on the cognitive domain which involves knowledge and the development of intellectual skills.

The cognitive domain involves knowledge and the development of intellectual skills. This includes the recall or recognition of specific facts, procedural patterns, and concepts that serve in the development of intellectual abilities and skills. One of the most cited and popular scheme is bloom's taxonomy (see above) which also plays a major role in modeling and describing competence orientation in current academic curricula following the European Bologna process. There are six major categories of cognitive processes, starting from the simplest to the most complex. At the same time, the taxonomy is expandable for new types of learning objectives. In our research project EVELIN (Experimental Improving of learning in Software Engineering), we developed a scheme that has been reworked based on bloom's taxonomy. The project EVELIN has set itself the goal to optimize lectures in software engineering in different types of IT and CS curricula. The EVELIN classification scheme is a proposal to describe competences for technical content [27]. It consists of the following six classes:

- *Remembering*: Remember and reproduce Information verbatim.
- *Understanding*: Understand the meaning and significance of information.
- *Explanation*: Recognize interrelationships, dependencies and similarities between information and explain in own words.
- *Using*: Use information in a defined or limited context or under instruction.
- *Apply*: Apply information and select solutions independently even in difficult environments.
- *Develop Further*: Using of existing information to develop new solutions and approaches.

b) In-game tasks and activities

We have adopted knowledge levels and identified indicators for advanced learning. Consequently, we have mapped them to card game design patterns to elaborate the fundamentals of our card game. The goal is to have different card game tasks and activities for every knowledge level. In addition to this, constraints and conditions must be identified and added to every level in order to fulfill constructivistic requirements. Table II illustrates game activities, and tasks as well as their collaboration constraints.

TABLE III: KNOWLEDGE LEVELS AND THEIR CORRESPONDING IN GAME ACTIVITIES AND TASKS

Knowledge Level	**Task Description**	**Collaboration**
Remembering	State the terms that are demanded in the task!	Only Single Player
Understanding	Describe the solution of the task!	Only Single Player
Explanation	Describe the solution your own words! Draw additional material like images to express your explanation!	Only Single Player
Using	Use the hints on the card to solve the task!	Single or Multi Player
Applying	Solve the given task without additional material!	Only Multi Player
Develop Further	X	X

V. *Game Design*

As already stated in chapter IV, research shows that most of the games for software engineering concentrate on software engineering management practices or software engineering processes. Only a few number of games exist for other parts of software engineering. Therefor we want to give a contribution to one of the minor highlighted areas, software testing.

Our card game builds upon the theoretical content of the international software testing qualifications board (ISTQB). The ISTQB builds upon three levels; foundation, advanced and expert [28]. For this first approach, the activities and questions within our game are built upon the foundation level. This level is aimed for first contacts with the topic of software testing. The content of this level are

- Fundamentals of testing,
- Testing throughout the Software Life Cycle,
- Static Techniques,
- Testing Design Techniques,
- Test Management and
- Tool Support for Testing.

The foundation level is appropriate for everyone who needs a basic understanding of software testing.

After identifying the technical content, the next step is to integrate it into the game. For a systematic approach on our game design, we analyzed the learning objectives and mapped them to the card game activities and tasks. The first step was to sort the content according to the corresponding knowledge level of the EVELIN Taxonomy. The ISTQB learning objectives are structured in four sections and classified as follows:

1. Remember, recognize, recall
2. Understand, explain, give reasons, compare, classify, categorize, give examples, summarize
3. Apply, use
4. Analyze

We mapped them to the EVELIN taxonomy based on the indicators presented in chapter IV. Table III gives an overview of the EVELIN knowledge level and the corresponding learning objective indication.

TABLE IV: MAPPING OF ISTQB LEARNING OBJECTIVES TO EVELIN TAXONOMY

EVELIN Knowledge Level	Corresponding ISTQB Learning Objectives
Remembering	Remember, Recall, Recognize, State
Understanding	Understand, Classify, Compare, Categorize, Summarize
Explaining	Explain, Give Reasons, Give Examples,
Using	Use
Applying	Apply, Analyze
Develop Further	---

The last level, *develop further*, is not part of the learning objectives in the ISTQB. Table IV shows which knowledge levels are addressed by the learning content of the ISTQB.

TABLE V: OVERVIEW OF LEARNING OBJECTIVES OF ISTQB FOUNDATION LEVEL (R=REMEMBERING, U=UNDERSTANDING, E=EXPLANATION, U=USING, A=APPLYING, D=DEVELOPFURTHER; A=FUNDAMENTALS OF TESTNG, B=TESTNG THOUGHOUT THE SOFTWARE LIFE CYCLE, C=STATIC TECHNIQUES, D=TESTING DESIGN TECHNIQUES, E=TEST MANAGEMEN, F=TOOL SUPPORT FOR TESTING)

		ISTQB Foundation Level					
		A	B	C	D	E	F
EVELIN taxonomy	R	X	X	X	X	X	X
	U	X	X	X	X	X	X
	E	X	X	X	X	X	
	U			X	X	X	
	A			X	X	X	
	D						

The learning objectives have different levels of activities and tasks. In the areas of *Static Techniques*, *Testing Design Techniques* and *Test Management*, students need to actively conduct activities corresponding to the taxonomy levels using and applying to deepen their knowledge in these areas. Hence, most activity cards are listed within these areas.

A. *Card Illustration*

The design of the card is inspired by *Magic: The Gathering (M:TG)* [29]. Our adopted design can be seen in Fig. 1.

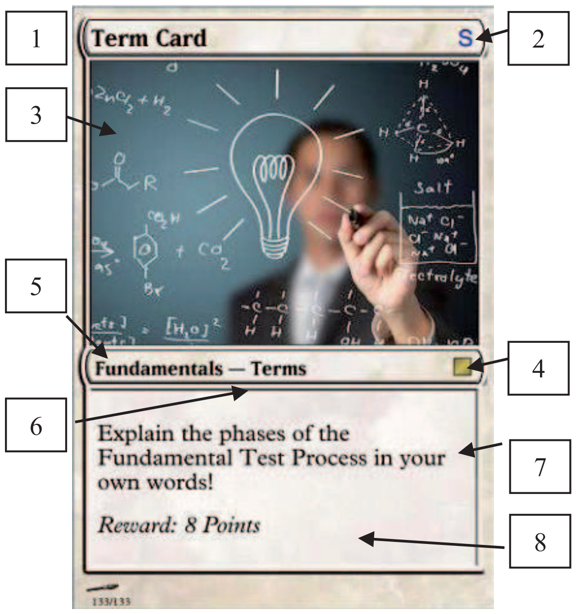
1
2
Term Card
S
3
5
4
Fundamentals — Terms
6
Explain the phases of the Fundamental Test Process in your own words!
7
Reward: 8 Points
8
133/133

Figure 1: Card Design

There are different elements, which are displayed on each card. The listing number corresponds to the identification number in Fig. 1:

1. At the top left corner of the card, the card label is tagged.
2. On the top right corner, either Single Player or Multi Player is given as condition.
3. In the middle of the card, additional information to solve the task is displayed if required. If there is no additional information available, the image remains blank.
4. The difficulty of the task is displayed with this square. There are three difficulty levels: green (easy), yellow (moderate) and red (tough).
5. The corresponding content level of the ISTQB is displayed here if the students wants to look up.
6. Here, additional information on the content is given.
7. In this field, the actual task or activity is described.
8. The reward is the number of points the player can achieve maximal when solving the task on his or hers own. The amount of reward points is adjusted to the corresponding difficulty level.

The solution of the task or activity is encrypted on the back of the card. The solutions are only readable, when a red transparent film is held over the text. To decrypt, the players need an extra paper sheet. Thus, we want to prevent that players read the solution before giving the answer.

B. Gameplay

The game is set up as a competitive game, in which students can chose to play alone or together with some of the group members, depending on the game situation. The cards have different difficulty levels and reward scores and can be solved solely or in collaboration. The tasks on the cards to achieve the points are based on the theoretical content of the ISTQB Foundation Level. Depending on their choices, the points will be divided between them. Nevertheless, the player who scores the limit, is the winner. The game is designed for up to four players. More players are possible, but the flow of play is reduced the more people are playing. To play the game, pen and paper and a red transparent film is needed. We want to promote communication and exchange of experience between the students and suggest the gameplay to be adoptable to their knowledge. Furthermore, we designed the gameplay to engage exchange experience and support within each other. The player can choose his or hers playing strategy based on his or hers knowledge level.
Turns are taken according to the following description:
At the beginning of the game, the card deck is shuffled. Afterwards, every player gets four cards. At the beginning of every move, the player who has the turn draws a card from the card deck. The youngest player begins with the first round. Then he or she chooses one of the cards on his hand to play. He puts the card in the middle of the table and reads the task out loud. Next, he or she chooses, depending on the card, if he or she wants to solve the card on his own, with a partner or with the rest of the group. The solutions for the tasks and activities are written on the back of the cards, defaced with red colors. After the player has tried to solve the task, the group decides for the given solution if the player earns the reward or not.
The tasks are designed to be solved in a few minutes so that a fluent gameplay can be ensured. The reward points are noted on a sheet of paper. The winner is the player who reaches 30 Points at first.

VI. Software Engineering Curriculum

The card game was integrated as a learning arrangement into the software engineering lecture in the fourth semester of the *Mechatronics Engineering* bachelor degree program at the Ostbayerische Technische Hochschule (OTH Regensburg). This bachelor comprises seven semesters. In the first two semesters, an introductory C programing course is set with a 90 minute lecture each week. A voluntary tutorial is offered parallel in the first semester and a mandatory practical course with 2 ECTS (European credit transfer system) credits is scheduled in the second semester. The course closes in the second semester with an exam and is rewarded with 4 ECTS credits. In the third semester basics of micro-controllers and assembler programming are presented in a course (5 ECTS credits) with two 90 minute lectures per week. In the fourth semester, this lecture is followed by a practical course (2 ECTS credits) and 90 minute lectures per week in C++ and software engineering. Each of these courses is worth 3 ECTS credits. Parallel to the C++ lecture, a practical course (2 ECTS credits) is scheduled. The curriculum closes in the fifth semester with a practical software engineering course (5 ECTS credits). The fifth semester is the practical semester in which our students are working for 18 weeks in real life engineering environments either in industry or in the research facilities at our university. The lecture software engineering closes with an exam, while the practical course ends with a course-related certificate of performance. The content of the software engineering curriculum consists of the following topics:

- Models of Software Design
- Requirements Engineering
- Review
- Design and Architecture of Software
- Configuration Management
- Design Pattern
- Software Testing
- Documentation and Maintenance of Software

VII. *Experiment*

A. Experiment Design

In order to perform a qualitative and exploratory validation of our card game, we asked software engineering bachelor students of our curriculum at the OTH Regensburg to play the game and to submit feedback in the form of a questionnaire. Therefore, we recruited 30 students who had already visited the software testing course and attended the software engineering exam. The students formed five groups to play the game, six persons in each group. Based on our experiences and supervision of the students in previous lectures, we mixed the students randomly while trying to avoid group formations on their own. Thereby we wanted to encourage the students to exchange with different fellow students. The experiment was designed to validate the hypothesis stated in chapter I, so we didn’t include any evaluation about the comparison of the performance of the different groups. In the future, we plan to conduct further evaluation experiments as well as studies about differences between students who played the game and those who did not.
At the beginning of the experiment, we gave a short introduction into the basics of the gameplay and assisted them in the first turns of the game. Afterwards, we let them independently play on their own until the winner was identified. The students played about 1 ½ hours till the last game activity had finished. Subsequent to this, we handed out the questionnaire. The design of the evaluation consisted of questions with numerical answers on a six point Likert scale. The questions were designed as statements for which the students could choose from fully agree (6) to fully deny (1).

The content design of the evaluation was divided into four categories.

- Gameplay and personal opinions about the game
- Social Interaction during the game
- Software testing content in a playful manner
- Personal data

B. Results

We have tried to match our results with the objectives stated in chapter I. In the following, we present major results of this experiment.

First we wanted to know whether students didn't only play the game but also had the possibility to discuss solution approaches and knowledge with each other. Hence, the first statements aimed at the communicational aspects during the gameplay.
The first statement *"I could exchange views with other players about practical software testing content"* had an average score of 5.3. This strong agreement lets us assume that during the game the students discussed the approach on solving the demanded card tasks and activities, particularly when they chose to work in groups.
If they mostly tried to solve single player tasks or not, we stated *„I didn't solve the tasks and activities on my own*. " The average score of 2.7 indicates that the students mostly tried to solve the tasks together with other players.
We suggested that *"I could comprehend the solutions respectively the answers of the other players"*. The average score of 4.83 indicates that most of the players agree with this statement. This gives us that the students could follow up with the solutions made by the other players.
Furthermore, we wanted to examine if the students' explanations were useful to understand the solutions. With an average score of 4.67, the statement *"The explanations of the other players for the solutions were comprehensible"* is agreed upon.
Altogether, we can assume that during the play of the game the students are actively exchanging and discussing approaches on solving tasks. Thereby the students gain new insights on how to approach on software testing problems.

Furthermore, we wanted to investigate if students can adopt the gameplay in a way that meets their individual learning style. Thus, students should have the possibility to gather new methods on how to approach different tasks and activities.
The statement *"I could control the gameplay in a way that I could ideally contribute my knowledge to my own learning needs"* earned an average score of 4.4. Thus we can assume that the players could choose individual gameplay strategies. These helped them to find their ideal way on how to approach on solutions based on their own learning styles. They could choose to solve the tasks on their own or combine their knowledge with the other group members.
Finally, we wanted to know if students did change their strategy because strategies employed by their peers worked better for solving the tasks. In the questionnaire, we stated, "*I tried to imitate the gameplay of the other players to solve the tasks and activities"*. With an average score of 4.17 students' affirmed this statement.

In addition, we stated that *"The difficulty level of the game corresponded to my abilities"*. With this statement the students agreed with an average score of 4.6. Thus we can say that the design of the tasks meets the typical level of knowledge, despite the heterogeneity of the students.

Another more fundamental question is, whether students get motivated to play again and get aware to increase their knowledge level by playing the game. First, we stated *"The game was a waste of time"*. This statement didn't meet the opinions of the students with an average score of 1.61. Quite to the contrary, with an average score of 4.64, they said that *"I would like to repeat the game"*.
Above that, the combination of playing a card game with software testing content gained acceptance with an average score of 4.5. The corresponding statement was *"I enjoyed the playful engagement with software testing!"*.

By explaining their approaches, students need to describe their sights in their own words to their teammates. Thereby they gain new insights on how to express and points of view on their knowledge. They deepen their knowledge.
This was confirmed by two statements "*Through the exchange with the other players, I could gain new insights and findings about software testing content"*. This statement earned an average score of 5.5 and thus was highly confirmed by the students.
The second statement instead aimed at the gained insights during the gameplay. *"By playing this game I could gain new insights and findings about software testing content"*. This was also confirmed with an average score of 4.67.
Most likely the third statement aimed at the promotion of knowledge by playing the game. The statement *"I could deepen my knowledge in software testing"* has an average score of 4.5

Summing up, the experiment generally confirms the suggestions made in chapter I. The students seemed to like the combination of software testing content within a card game. Our approach in the design of the game seems to be adjustable to the learning needs of the students. While playing the game, students changed their group members and the size of their group. This had two reasons: First, after a short period they figured out the individual players' strengths and weaknesses and used this information for their own good. Second, if one player gets too many points, they needed to work together with other players to catch up. Overall, the students seemed to enjoy playing our card game and would like to play it again. Furthermore, the results of our experiments show that students will embrace the use of the game, as most of our test subjects felt that playing the game was both a useful lesson and an enjoyable experience. They stated that by playing this game they could consolidate their knowledge in software testing by adjusting the gameplay and cooperation during the game.

VIII. Conclusion

This was a first experiment to use a card game in a software engineering course to strengthen the content of software testing. It addresses many of the weaknesses of more traditional techniques and brings benefits in the form of group learning and enjoyable learning. In combination with lectures, we believe that the integration of game-based learning elements like our card game builds an appropriate environment for the learning needs of our software engineering students. Although the first results of our experiment are quite satisfactory, still some work needs to be done. Especially balancing of bonus points of cards and gameplay is a task that needs to be revised. In future work, we want to examine and survey the effects of the card game to the knowledge development of the learners in detail. But the first results show that using games, respectively card games, in academic education of software engineering is motivating, enjoyable and helps the students to promote deeper knowledge.

ACKNOWLEDGMENTS

This work is supported by the Federal Republic of Germany, Federal Ministry of Education and Research, BMBF grant EVELIN "Experimentelle Verbesserung des Lernens von Software Engineering", grant identity 01PL12022F, project sponsor DLR. The network project EVELIN is part of the „Bund-Länder-Programm für bessere Studienbedingungen und mehr Qualität in der Lehre". The Universities of Applied Sciences Aschaffenburg, Coburg, Kempten, Landshut, Neu-Ulm and Regensburg are the network partners. More information under http://www.las3.de or http://www.qualitätspakt-lehre.de. Solely the authors are responsible for the content of this publication.

REFERENCES

[1] N. E. A. M. Almi, N. A. Rahman, D. Purusothaman, and S. Sulaiman, "Software engineering education: The gap between industry's requirements and graduates' readiness," in Informatics (ISCI), pp. 542–547.

[2] C. Bonwell and J. Eison, "Active Learning: Creating Excitement in the Classroom", in AEHE-ERIC Higher Education Report, vol. 1, Washington D.C.: Jossey-Bass, 1991.

[3] N. J. Maushak, H. Chen and H. Lai, "Utilizing edutainment to actively engage k-12 learners and promote students' learning: An emergent phenomenon.", in Association for Educational Communications and Technology Annual Conference Proceedings, Atlanta, October, 2001.

[4] C. Caulfield, J. Xia, D. Veal, and S. P. Maj, "A Systematic Survey of Games Used for Software Engineering Education," in MAS, vol. 5, no. 6, pp. 28–43, 2011.

[5] C.G. von Wangenheim and F. Shull, "To Game or Not to Game?," in Software, IEEE , vol.26, no.2, pp.92-94, March-April 2009.

[6] L. Ermi and F. Mäyrä, "Fundamental components of the gameplay experience: Analysing immersion.", in Worlds in play: International perspectives on digital games research, p. 37, 2005.

[7] J. Schell, "The Art of Game Design: A book of lenses.", CRC Press, 2014.

[8] R. Sandford, "Teaching with games.", in Computer Education Stafford-Computer Education Group, vol 112, p. 12, 2006.

[9] L. Griffin and J. Butler, "Teaching games for understanding: Theory, research, and practice.", Human Kinetics Publishers, 2005.

[10] T. Hopper, "Teaching games for understanding: The importance of student emphasis over content emphasis.", in Journal of Physical Education, Recreation & Dance, p. 44-48, 2002.

[11] E. O. Navarro and A. van der Hoek, "SimSE: an educational simulation game for teaching the Software engineering process.", in ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, New York: ACM Press, p.233-233, 2004.

[12] J. M. Fernandes and S. M. Sousa, "PlayScrum - A Card Game to Learn the Scrum Agile Method," in 2010 2nd International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES 2010), pp. 52–59, 2010.

[13] A. Baker, E. O. Navarro, and A. van der Hoek, "Problems and Programmers: an educational software engineering card game," in 25th International Conference on Software Engineering Proceedings, pp. 614–619, 2003.

[14] D. Ismailovic, J. Haladjian, B. Kohler, D. Pagano and B. Brugge, "Adaptive serious game development.", in 2nd International Workshop on Games and Software Engineering (GAS), Zurich, Switzerland, S. 23–26, 2012.

[15] J. H. Sandholtz, C. Ringstaff, and D.C. Dwyer, "Teaching with technology: Creating student-centered classrooms.", Teachers College Press, New York, 1997.

[16] M.F. Mascolo, M. F., "The coactive construction of selves in culture.", in M. F. Mascolo and J. Li (Eds.),Cultureand self: Beyond dichotomization. New directions in child and adolescent development series, San Francisco: CA:Jossey-Bass., 2004

[17] A. Gray, "Contructivist teaching and learning.", in SSTA Research Centre Report, p. 97-07, 1997.

[18] S. Bjork, and J. Holopainen, "Patterns in game design", Charles River Media Game Development, 2004.

[19] S. M. Fisch, "Making educational computer games educational.", in proceedings of the 2005 conference on Interaction design and children, ACM, 2005.

[20] R. T. Hays, "The effectiveness of instructional games: a literature review and discussion.", in Technical Report 2005–2004 for the Naval Air Center Training Systems Division, Orlando, FL, 2005.

[21] I. Verenikina, "Understanding scaffolding and the ZPD in educational research," Paper presented at the Australian Association of Educational Research Conference, Auckland: New Zealand, Nov. 30 – Dec. 3, 2003.

[22] N. F. Jumaat and Z. Tasir, "Instructional Scaffolding in Online Learning Environment: A Meta-Analysis", in IEEE International Conference on Teaching and Learning in Computing and Engineering, 2015.

[23] S. N. A. Rabu, B. Aris, and Z. Tasir, "Instructor Scaffolding and Students' Critical Thinking through Asynchronous Online Discussion Forum", in IEEE Learning and Teaching in Computing and Engineering, 2013.

[24] Northern Illinois University, Faculty Development and Instructional Design Center, Instructional Scaffolding to Improve Learning, found at: http://www.niu.edu/facdev/resources/guide/strategies/instructional_scaffolding_to_improve_learning.pdf

[25] D H. Jonassen, "Designing constructivist learning environments.", in Instructional-design theories and models: A new paradigm of instructional theory, vol.2, Mahwah, NJ: Lawrence Erlbaum Associates, p.215-239, 1999.

[26] M. Kordaki, "A computer card game for the learning of basic aspects of the binary system in primary education: Design and pilot evaluation.", in Education and Information Technologies, Vol. 16.4, p.395-421, 2011.

[27] S. Claren and Y. Sedelmaier, „Ein Kompetenzrahmenmodell für Software Engineering. Ein Schema zur Beschreibung von Kompetenzen.", in Embedded Software Engineering (ESE), Sindelfingen, S. 647–652, 2012.

[28] D. Graham, E. van Veenendaal and I. Evans, "Foundations of software testing: ISTQB certification.", Cengage Learning, EMEA, 2008.

[29] Magic: The Gathering, Wizards of the Coast, [Online]. Available: http://www.magicthegathering.com