Association for Information Systems
AIS Electronic Library (AISeL)

BLED 2017 Proceedings | BLED Proceedings

2017

The Playful Learning Approach for Learning How to Program: A Structured Lesson Plan

Robert Heininger
Technical University of Munich, robert.heininger@in.tum.de

Victor Seifert
Technical University of Munich, victor.seifert@tum.de

Loina Prifti
Technical University of Munich, prifti@in.tum.de

Matthias Utesch
Staatliche Fachober- und Berufsoberschule Technik München, utesch@in.tum.de

Helmut Krcmar
Technische Universität München,, krcmar@in.tum.de

30TH BLED ECONFERENCE: DIGITAL TRANSORMATION – FROM CONNECTING THINGS TO TRANSFORMING OUR LIVES (JUNE 18 – 21, 2017, BLED, SLOVENIA)
A. Pucihar, M. Kljajić Borštnar, C. Kittl, P. Ravesteijn, R. Clarke & R. Bons

University of Maribor Press

The Playful Learning Approach for Learning How to Program: A Structured Lesson Plan

ROBERT HEININGER, VICTOR SEIFERT, LOINA PRIFTI, MATTHIAS UTESCH & HELMUT KRCMAR

Abstract IT-based learning has proven to be a useful approach to educate people regardless of their age or other characteristics. However, the developments in IT and its socio-economic implications have a high influence on education with new approaches and methods, such as the playful learning approach (PLA). This approach has been widely researched and can be applied to teach programming as one of the core digital skills. However, scientifically developed and validated structures for PLA units in programming are rare. In this paper, we offer a lesson structure for a PLA to programming by addressing the five core success factors of playful learning. Our structure includes six units and follows an iterative and agile procedure by combining game features with the educational content. Educators and teachers can use the presented results to design the lesson structure in their classes. Furthermore it offers a basis for further research in the area of PLA and can be used as a starting point for the development of educational games and concepts in teaching how to program.

Keywords: • IT-based learning • programming • playful approach • best practice • lesson structure •

CORRESPONDENCE ADDRESS: Robert Heininger, Technical University of Munich (TUM), Arcisstraße 21, 80333 München, Germany, e-mail: robert.heininger@in.tum.de. Victor Seifert, Technical University of Munich (TUM), Arcisstraße 21, 80333 München, Germany, e-mail: victor.seifert@tum.de. Loina Prifti, Technical University of Munich (TUM), Arcisstraße 21, 80333 München, Germany, e-mail: prifti@in.tum.de. Matthias Utesch, Staatliche Fachober- und Berufsoberschule Technik München, Orleansstraße 44, 81667 München, Germany, e-mail: utesch@in.tum.de. Helmut Krcmar, Technical University of Munich (TUM), Arcisstraße 21, 80333 München, Germany, e-mail: krcmar@in.tum.de.

https://doi.org/10.18690/978-961-286-043-1.16 ISBN 978-961-286-043-1

Available at: http://press.um.si.

1 Introduction

The term *Educational Technology* is defined as "the study and ethical practice of facilitating learning and improving performance by creating, using, and managing appropriate technological processes and resources" (Robinson, Molenda, & Rezabek, 2008). E-learning, web-based learning, online learning, and IT-based learning are just a few examples for technological processes and resources based on Information Technology (IT). However, IT is a constantly evolving sector influencing many areas of all our lives. Thus, education is changing since IT allows for many new possibilities; especially in combination with the playful approach to learning. Playful learning, often also called *edutainment*, is a recent trend in academic education which focuses on the *hands-on* practice of learning instead of on the *sit-and-listen* approach, spanning between free play (in which the students play independently), and guided play (where an overseer directs their play)(Lillard, 2013).

In the digital economy, there is a huge demand for skilled IT workers, not only in the IT departments but across entire organizations (Capgemini Consulting, 2013; Prifti, Knigge, Kienegger, & Krcmar, 2017). This demand can be satisfied by a broad introduction of students to the digital world and further education in IT skills early on. Additionally, in today's society, it has become a standard for students to have constant access to the internet and the immediate on-the-spot knowledge provision opportunities that are provided by it. Students use digital means to learn, communicate, access knowledge, and for other professional application (Baggia et al., 2016). A case study involving undergraduate students in Slovenia, stated that 37,5% of the students reported to use the web frequently in more than half of their courses, and 50% even stated that they always use web resources (Kljajić Borštnar, 2012). This trend in students' behaviour can be observed growing rapidly. In 2002, a case study, involving 2054 students at 27 institutions of higher education across the United Stated, stated that 73% of respondents claimed to use the internet more than the library, while only 9% said they use the library more than the internet for information searching (Jones, 2002). Thus, our nowadays educational systems are facing both, students with highly developed digital skills and the necessity to evolve the digital skills of students in order to support self-regulated personalized learning (cf. Zimmerman, 2002).

IT-based learning approaches gain momentum in our digital economy as countless numbers of software projects, for computers as well as for smartphones, are realized with the goal to educate students on different topics. The importance of digital mediums as a form of education is increasing (Blamire, 2010), and many academic researchers, such as McGonigal (2011), Prensky (2005), Gee (2003), Fabricatore (2000), and Pivec and Moretti (2008), put increasing effort into the exploration and validation of teaching techniques using the playful approach as a core concept. Many of these playful approaches use hands-on and self-regulated or personalized learning as a medium to transfer knowledge to students and the usage of the self-regulatory processes has shown strong correlations with high academic achievement (Everson, n.d.)

Case studies implementing the playful learning approach (PLA) in all kinds of fields of education are widespread, although the majority of them supplies no scientific results concerning a structure on how to implement such a playful teaching model. Thus, this paper aims to fill this gap and hence to answer the following research question:

What would be a best practice for programming lessons by using a playful approach?

Fullan (2007) asserted that the moral purpose of education is to equip students with the skills that enable them to be productive citizens when they finish their studies. Thus, in the age of the digital economy improving the digital skills should be one of the topics students ought to learn in school. Trilling and Fadel (2009) called education in information, media, and technology skills vital for the 21st century job market, differentiating between the traditional education subjects and upcoming subjects such as information transformation and coding skills which are much needed in today's economy.

As the structure of a teaching approach is always to some level dependent on the subject and because of the dire state of the computer science education throughout Europe, we focus on the area of computer science, with the special topic of learning how to program, as a framework in which the structure is developed. However, the structure is easily transferrable to other school subjects as the core composition stays the same.

Next to the success factors identified for teaching with a playful approach in Heininger, Prifti, Seifert, Utesch, and Krcmar (2017), other factors also need to be taken into account concerning a successful implementation of a study course. Examples are the relevant curriculum (in our case technology and computer science), and specific environment factors of the respective school or university (the weekly hours planned for computer science, the duration of the course, and the chosen programming language). The development of the structure will manipulate certain environmental factors as well as provide opportunities for the students to become motivated, interact in class, and engage with the content of the curriculum on a deeper level than in a "traditional" course in which the lecturer just transfers knowledge by lecturing.

The overall goal is to promote continuous passion and motivation to look beyond the box and to become interested in programming. The ideal outcome would manifest in students wanting to code even if there is no extrinsic reason to, such as good grades in school or any other rewards from the outside. The development of the curricular structure and the structure of implementation targets this endeavor.

2 State of the Art

Driven by rapidly advancing information technology, many academic researchers have lately delved into the topic of different approaches to education. The benefits and theoretical as well as practical outcomes of playful approaches in education have been

put to test in many settings, e.g. in undergraduate and postgraduate programs (Connolly, Stansfield, & McLellan, 2006), at high school (Papastergiou, 2009), universities (Colace et al., 2008; Iglesias & Gálvez, 2008; Jantke & Woelfert, 2012), and even at an US Air Force Academy (Fagin, Merkle, & Eggers, 2001). However, scientifically developed and afterwards validated structures for implementation are rare and often not documented. Only Simionescu and Marian (2016) present a proposal for a change of paradigm, based on a playful approach in course structures for efficient student evaluation.

In a previous research paper (Heininger et al., 2017) the success factors concerning the playful teaching approach were discussed and evaluated by conducting a literature review of the current state of the art. With this contribution, we aim at answering our research question, by using these success factors as the theoretic part of the basis for our structure. Table 1 shows the five key success factors influencing the outcome of such an endeavor: motivation, integration and involvement in class, the audience-centred focus, giving feedback and enhancing interaction, and the fluent integration of the educational content into the gameplay (Heininger et al., 2017).

Table 1: Success factors for teaching with a playful approach, based on Heininger et al. (2017)

Success factor	*Description*
Motivation	The basic reason why people do anything; The variation of individual differences; The attitude, behaviour and study practices
Integration and involvement	Self-involvement and active participation by students; The shift from the passive recipient to an active integrated partner
Adaption to the audience	Specification of the target audience; The adaption to the curriculum and environment; The game shall be easy to understand and to play
Interaction and feedback	Formal as well as informal; The individual response to the student's difficulties; The examination of different ideas and multiple perspectives
Integration of educational content into gameplay	Finding an application of the curriculum inside of a gameplay; The balance of gameplay and educational content in a game

3 Method

Curriculum decision making is generally an iterative and lengthy process, carried out by a broad range of participants and influenced by a wide variety of stakeholders (Van den Akker, 2007). Cho (1998) noted on the same subject that the implementation of curricula is not an event, but a longer change process. Following Biggs and Tang (2007), there are

three levels of learning outcomes at major institutions – at the institutional level, at the degree programme level, and at the course level. As we concentrate on the curriculum of a school, our focus is on the course level learning outcome, while the degree programme level is less distinctive. Furthermore, we limit the computer science curriculum to the subject of learning how to program in order to be able to provide a clear and comprehensive structure.

Van den Akker and Voogt (1994) noted that the curriculum should offer materials, which provide concrete and illustrative elaborations of the general program, as well as a framework for broad categories. Biggs and Tang (2007) further remarked the importance and analysed the 'intended learning outcome' for the student as it defines the goals of the curriculum. Based on this outcome-based learning concept by Biggs and Tang (2007) we will describe the current state of computer science curricula and point out some major design decisions for our curriculum in this chapter.

3.1 Linking to a School Curriculum

Many countries have adjusted their curriculum in the last years, adapting to the changes of the digital economy, which requires new skills (Capgemini Consulting, 2013; Prifti et al., 2017). However, specific curricula are often set at state or city level, not at national level. An example outside of Europe is New York City where a computer science course will be a graduation requirement by 2018 (Taylor & Miller, 2015). In Europe, many countries have begun to adjust their curricula as well, but major changes are still awaited by both, the industry and the academia. Austria for example requires a mandatory course devoted solely to computer science only in grade 9, teaching students about software, hardware, operating systems, and data privacy (Guerra, Kuhnt, & Blöchliger, 2012). Slovenia offers a mandatory computer science course in grade 10 (high school and gymnasium), while offering elective courses in the grades 7-9 and 11-13, teaching students about processing data, computer networks, and programming as well as algorithmic thinking and problem solving (Guerra et al., 2012). However, in 2016, Mori and Lokar (2016) criticized the outdated Slovenian computer science curriculum, dating back to 1998 with minor updates in 2008 as well as the short duration of the mandatory course of only one year. In Germany, the education system is mainly the responsibility of the federal states. In 2012 only two out of sixteen German federal states had mandatory computer science courses – Free State of Saxony in grades 7 and 8; and Free State of Bavaria in the grades 6 to 12, including topics such as introduction to software and hardware, terminology, basic concepts of information technology, computer networks, algorithms, and data modelling (Guerra et al., 2012). However, the curricula for the different types of schools in Germany, respective in each individual state, are often vastly different and some of them do not teach about computer science at all (Deutscher Bildungsserver, 2017). In 2015 a survey with 1002 German parents stated that the majority (56%) considered the children's preparation by the schools for the digital job market 'bad', 5% even considered it 'very bad', while only 32% considered it 'good' and 3% considered it 'very good' (Netzwerk Digitale Bildung, 2015). Overall, the changes in

the education systems of European countries seem to be far behind the demands of the digital industry. It is hence our goal to develop a structure that gives incentive for students to learn about computer science (in our specific case: how to program) and changes the approach from the traditional path to a playful one.

The structure, developed in this contribution, is based on the Bavarian curriculum for vocational schools in introductory computer science courses, which divides the topic of learning how to program into three parts:

basics of modern programming languages,
programming techniques and data structures, and
object-oriented-programming.

The basics of modern programming languages include the types of data, arithmetic and logical operators, and output of data. School students learn about basic programming structures such as sequences, single-branching, and multi-branching. Furthermore, this includes how identifiers are structured, type conversion, and how statements are coded by differentiating between logical lines. Advanced programming techniques and data structures are supposed to give the students the ability to create software that is more advanced.

"*Students create programming-oriented diagrams and use repetition commands and subroutines. They know the necessity of further programming techniques and structures to solve complex problems using varying data structures. Thereby they recognize how to write more effective and clearly arranged software.*"

(Bayerisches Staatsministerium für Unterricht und Kultus, 2006)

The emphasis lies on students learning to independently break a problem down into smaller parts and create advanced pieces of software. The modulation of problems is explained with the use of procedures, functions, giving parameters, and multiple usage of blocks of code. Knowledge about array data structures as well as objects of pre-defined classes are part of the curriculum as well. The overall goal is to teach students about complex structures like loops and functions to give them the ability to differentiate between the use of procedural structures and functions. Object oriented programming is the last part of the computer science curriculum of the vocational schools in Bavaria.

"Students ought to gain insight to how coding and handling objects work. Their ability to generate a solution creating functions based on problem is in focus."

(Bayerisches Staatsministerium für Unterricht und Kultus, 2006)

The curriculum for object-oriented programming contains knowledge about classes and objects, their attributes and methods, as well as class instantiation. The focus lies on the

definition of classes and calling functions and attributes of the instances, as well as handling the transfer of parameters. This includes calling functionality in other functions as well as setting optional return values. The goal is to teach students about well and systematically designed structures of classes and functionality to ensure their ability to solve complex problems and give them the tools to code on their own in the future.
At this point of our contribution, we deem it important to note that the curriculum of vocational schools in Bavaria will be changed in the near future – computer science will soon not be a mandatory subject anymore at all, but rather a voluntary subject in grade 12 and 13 (Staatsinstitut für Schulqualität und Bildungsforschung München, 2017). The content of the voluntary subject will also consist of multiple optional topics from which only a few can be taught as the course will probably not have the required length in order to cover all of them thoroughly.

3.2 Choosing the Programming Language Python

The focus of an introduction to programming in secondary schools lies on giving insights into writing software and understanding the basics of programming. The usage of languages developed for children (e.g. Scratch or Snap!), which are based on graphical drag-and-drop handling, seems to be missing the point in this endeavor.

Choosing a programming language for beginners, means not only considering technical aspects, for instance run time efficiency, memory consumption, and reliability (Prechelt, 2000), but must be evaluated even more concerning the difficulty of learning the language without prior coding experience. Furthermore, there is the possibility of teaching an industry relevant language like Java, Python, and C++, or a language specifically designed to introduce programming to beginners. In a case study Ivanović, Budimac, Radovanović, and Savić (2015) note that the choice of programming language does indeed not affect the success of students in the course. They also note that "the main goal of the introductory programming course is to teach students essential programming concepts in order to develop their ability to think and solve problems by algorithms, and to acquire new/other programming languages and techniques efficiently" (Ivanović et al., 2015). This allows students to change to other, differently structured programming languages later on, if desired.

The language should have a simple syntax, provide easy I/O handling as well as output formatting, use meaningful keywords, and give immediate feedback (Grandell, Peltomäki, Back, & Salakoski, 2006). Python offers many advantages concerning grammar and writing of code: Time wasting matters of style are avoided and indentation not only structures the code grammatically but also visually helps beginners to easily understand which block of code belongs to which functionality. Another advantage of Python is its compactness and that unfamiliar code in Python is usually easy to read.

Choosing Python as a programming language affects the curriculum concerning the time, which must be allocated for specific concepts within the language. As an example,

compared to other languages, writing and structuring Python code happens mainly based on indents, which needs less time to explain to the students than the introduction of keywords like 'static' or 'void' would take in Java.

Python is a language that most people consider easy to learn though it still finds use in 'big' software companies like Google: "Python where we can, C++ where we must" (Holderness, 2016) and in games like EVE Online, Battlefield, Mount&Blade, or Civilization IV (Boddie, 2013). Thus, we consider Python as an industry-relevant language. Unlike other languages proposed for teaching to novices, Python is not just a teaching language, it is a language that is suitable for developing real-world applications (Radenski, 2006). An important goal of the Python developers is making Python fun to use, as also can be noticed by the origin of its name, while focusing on code readability, extensibility, and clear error message handling (The Python Software Foundation, 2017). Focusing on gaining insight into what it is really like to code and at the same time considering the shallow learning curve of Python, leaves us at the conclusion that Python is a good start for beginners without prior coding experience.

4 Results

In order to clarify the developed structure we divide the topic into its three major components in this chapter. First, we will present the organization of the lectures, which consist of four parts: plan, do, check, and act. Secondly, we will draw up a timeline of the progress of the lessons, tackling the curriculum apportionment and the general concept of introducing students to new knowledge. In the last part of this chapter, we will introduce the educational content, the means of the playful approach, and the main design principles for an educational game.

4.1 Organization of the Lectures

The lecture optimally is held by an experienced teacher with previous experience in the playful teaching approach, as well as extensive knowledge in computer science and coding. Computers with the pre-installed software are provided by the school, but the students should still be encouraged to bring their own laptops in order to be able to take notes, and to code in their own environment, as they would back home on their own. Every school student should play and code independently.

In the first five minutes of each unit, the teacher repeats the knowledge learned in the last unit, combined with an outlook to what students are required to learn in the current unit, and how it is connected to previous knowledge. As the students will have progressed at different speeds and hence be at different levels of knowledge, the repetition is based on their remarks and input. The underlying idea is to not set a common learning goal for the students for each unit, but to clear up eventual difficulties with previous contents. The students themselves set interim learning goals (**Plan**).

During the course of each unit the teacher helps and coaches the students if needed. The teacher does not need to approach students but let them work independently in order to give them the freedom to solve problems on their own. The aim behind this endeavor is fostering their motivation, self-value, and to increase their confidence in their own skills. A solution for a problem should not merely be offered by the teacher, but be worked towards together (**Do**).

The last five minutes of the units are for a short repetition of the current content in order to gather more information on the difficulties the students experienced and how they felt about the learning process while playing the game. The students should use the time for self-reflection (**Check**). The teacher provides handouts for each stage of the game to give the students something tangible to study and base their learning effort on before the exam. In order to keep track of the school student's progress, homework will be assigned to the students. To ensure quality, one of the underlying concepts of this approach is the Plan-Do-Check-Act Cycle (PDCA)[1] of W. Deming (Deming, 1986) as illustrated in Figure 1, showing the general lesson structure.

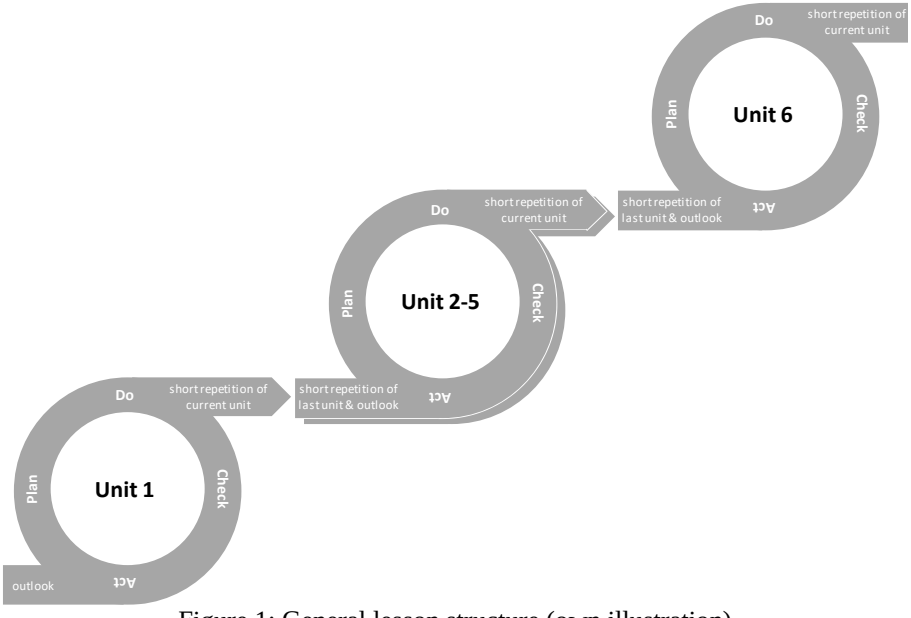

Figure 1: General lesson structure (own illustration)

The students will receive instant feedback on their code by the game. The teacher should help if difficulties come up and the students cannot solve a problem based on the feedback of the game or do not understand the error messages. By observing the students play, the teacher will also be able to identify complications in the game-play or during the execution of coding tasks (**Act**).

4.2 Lesson Structure

The length of each lesson is 45 minutes as it is custom in the German school system. The time planned for computer science, and more specifically to teach how to program, are six lessons, spread over the course of six weeks. In the Bavarian technical upper vocational schools, the course is mandatory for the students, most of which have no prior experience in coding. In this chapter, we explain the specific content of each lesson.

Whenever possible, concepts should be introduced completely and at once, as an exception some concepts are initially introduced in a limited fashion and then revisited in a later topic for full coverage (Radenski, 2006). The lesson structure introduces the students to different concepts and knowledge about coding with a gradually advancing learning curve. The content is structured to build on previous knowledge learned in earlier units and allows for a start without any prior coding experience. The lesson structure is adapted to the curriculum set by the Bavarian Education Ministry and the pre-defined situational and environmental circumstances in the school. The course is divided into six individual units, which are presented in the following and summarized in Table 2.

1st Unit

Students are introduced to coding and its use in our daily lives. The educational game will be implemented and its controls will be explained. Beginning with small pieces of code the school student gains knowledge on how to handle the IDLE Python GUI editor to be able to solve tasks and experiment at home on his own. Basic knowledge about assignment of values to variables, mathematical operators, and output, as well as basic data types including integers, floats, and strings.

2nd Unit

The students learn about boolean values and conditional structures to give software the ability to react on different types of input. The students learn basic commands to handle arrays and to calculate with boolean operators. The goal is to deepen the knowledge about dealing with different types of output and data while introducing branching structures. Manipulation of arrays and decision structures based on boolean variables, or by analyzing the input, complete this chapter.

3rd Unit

The third lesson will give a deeper understanding and further foster the ability to handle more complex data structures. More commands in order to handle arrays are given, such as sorting, reversing, inserting, indexing, and popping elements in arrays. This will be combined with handling the “for”-loop functionality. Iterating over arrays and building decision structures to connect the new knowledge to the 2nd Unit are at the core of this

unit. Students learn how to break down a problem into smaller parts and separate problems in order to be able to find solutions for a bigger task.

Table 2: The six units of the lesson structure

Unit	Content	Goal
1	Introduction to coding in the IDLE Python GUI and to the educational game, output, operators and types of variables such as string, integer and float	Ability to handle the coding tools, first introduction to basic coding and the educational game
2	Introduction to boolean values and arrays, conditional structures	Ability to deal with different types of output/input and connecting basic knowledge of mathematics to programming structures
3	Complex data structures, for loop and array handling	Advancing complex code structures, iterating and building extensive decision structures
4	Repetitive use of code by learning about methods and classes	Object oriented programming, describing virtual objects by setting attributes and implementing functionality
5	Use of classes, instances, parameter transfer, return values in functions	Systematic design of software, readability and transparency of the code
6	Repetition by connecting previously learned knowledge with complex tasks	Discover and close gaps in knowledge, deepen the comprehension of the content

4th Unit

In this unit, the students learn about repetitive use of code using methods and classes. Methods are introduced in a functional coding style and afterwards connected to classes. Classes will be used as a way to describe objects in a compact way and give functionality to them in order to change their attributes based on input and parameters as well as handling the interaction between classes. Examples of their use will be given and students are expected to add functionality and attributes to an existing class before creating a class on their own.

5th Unit

This unit focuses on knowledge about classes while introducing new features involving multiple instances and parameter transfer as well as optional return values. Furthermore, the while-operator in connection with boolean operators and values will be introduced.

The emphasis of this unit lies on good systematic design of software to ensure readability and transparency of the code.

6th Unit

The last unit holds no more additional new knowledge but means to give students the tools to code on their own in the future. It combines previous knowledge of the first six units. The students are given complex tasks in order to recognize potential in coding. The goal of the unit is to discover knowledge gaps in order to help students repeat or deepen specific units in order to achieve full understanding of given code.

4.3 Game, Features, and Educational Content

An educational video game is being developed with the single goal of connecting the curriculum (of how to program) with a game that is interesting and fun to play; and simultaneously puts learning into perspective and gives incentive to learn more. The students will be able to see that their own learning environment was created by using the same knowledge they will have at the end of the course. Furthermore, by creating an educational game for learning how to code in a gameplay setting, which resembles widely used and popular interactive development environments (IDE), could incite the motivation, self-esteem, and knowledge in students to carry on coding on their own. Necessary features in an educational game can easily be derived by approaches to satisfy the success factors for the PLA (Heininger et al., 2017): motivation, integration and involvement in class, the audience-centred focus, giving feedback and enhancing interaction, and the fluent integration of the educational content into the gameplay.

Main design principles for an educational game can also be derived from the iterative processes of conducting case studies by other authors and evolving and improving an educational game: interaction and iteration of those interactions, adaptive and personalized feedback, clear winning criterion, and no or few opportunities to cheat the game (Tillmann, De Halleux, Xie, Gulwani, & Bishop, 2013). Those criteria have been confirmed by another often cited study of educational games proposing practical steps for designing educational games (Linehan, Kirman, Lawson, & Chan, 2011). A good game is easy to learn but hard to master (Prensky, 2002) meaning the handling of the basic game is easy, but to be good at it becomes more and more difficult the further you advance. The playful approach motivates the students to become more interested in their own education and willingly sacrifice time for it. The game achieves this by students having to work for their knowledge by beating the game, defeating numerous enemies and mastering obstacles.

A basic rule of good game-play is to provide the player always with clear short-term goals (Prensky, 2002). In an educational game, this may mean partly beating enemies, collecting coins and jumping over obstacles, but also means solving coding tasks to advance in the gameplay. Beating enemies and collecting coins may obviously not have

a direct impact on the students learning success, but very well can create the joy and motivation to keep on playing, and thus to keep on learning. Examples of collectibles for the sake of their own can be found in numerous games: coins in the Mario series, spraying graffiti in GTA San Andreas, or visiting viewpoints in the Assassins Creed series. Players often do not try to find all collectibles for the purpose of getting a reward but for the sheer pleasure of the treasure hunt. After all, games are supposed to be fun and not purely work.

5 Conclusion

Based on former research, concerning the success factors for teaching with a PLA (Heininger et al., 2017) we were able to develop a lesson structure which shows the structures used for a playful approach to teaching. The content is based on the curriculum for technology classes of the Bavarian technical upper vocational schools, as well as Bavarian school norms. However, it can be transferred to other types of schools and curricula as well by customizing the proposed structure to fit different circumstances, such as timeframe of a lesson, length of course, programming language, and previous skill level of students. The lesson plan can be especially useful to be implemented in different German vocational schools, but also in other cultures as the structure's core is based on outcome-based learning, which was analysed and evaluated by Biggs and Tang (2007) and is used worldwide as a framework for good teaching and assessment. The curriculum was adapted to Python and its specific language constructs. The development of an educational game as an exemplary playful learning approach would be based on the structure created and paired with the educational content of the curriculum. The educational game as an electronic learning platform brings together practice (learning how to program), teachers (as a supporting entity, rather than lecturer), scholars (providing the theoretic foundation and practical knowledge based on previous case studies), and students (as active participants in their own education) in one environment.

We chose python as the favourable programming language based on its compactness and intuitiveness, which makes it easy to learn and to use for beginners. Whenever possible, concepts are introduced completely and at once, as an exception some concepts are introduced in a limited fashion and later revisited for full coverage (Radenski, 2006). A professional teacher, who is experienced in the topic of computer science, optimally accompanies the lecture. Five minutes in the beginning as well as at the end of each lesson are used to repeat the knowledge of the last lesson, respectively to recap the knowledge learned in the current lesson.

The use of an educational game as shown in this paper and the implementation of a playful teaching approach in a case study at a Bavarian upper vocational school will be part of the research in the future in order to validate the lesson structure.

Acknowledgement

The authors would like to thank the anonymous reviewers for their very helpful suggestions and comments. Additionally, the authors would like to express special thanks to the upper vocational schools at Bavaria, Germany represented by Günter Liebl, Werner Maul, Konrad Maurer, and Thomas Ondak.

Notes

1 Originally introduced as *Plan – Do – Study – Act* (PDSA)

References

Baggia, A., Žnidaršič, A., Borštnar, M. K., Pucihar, A., Šorgo, A., Bartol, T., . . . Dolničar, D. (2016). *Factors influencing the Information Literacy of Students: Preliminary Analysis.* Paper presented at the 29th Bled eConference – Digital Economy, Bled, Slovenia.

Bayerisches Staatsministerium für Unterricht und Kultus. (2006). Lehrpläne für die Fachoberschule und Berufsoberschule; Unterrichtsfach: Technologie/Informatik Retrieved from https://www.isb.bayern.de/download/9223/lp_fos_bos_technologie.pdf

Biggs, J., & Tang, C. (2007). *Teaching for quality learning at university* (Society for research into higher education Ed. 3 ed.): Open University Press.

Blamire, R. (2010). Digital Games for Learning: Conclusions and recommendations from the IMAGINE project. *European Schoolnet*.

Boddie, P. (2013). Python Games. Retrieved from https://wiki.python.org/moin/PythonGames

Capgemini Consulting. (2013). *The Digital Talent Gap* Retrieved from Digital Transformation Research Institute - Capgemini https://www.capgemini.com/resource-file-access/resource/pdf/the_digital_talent_gap27-09_0.pdf

Cho, J. (1998). *Rethinking Curriculum Implementation: Paradigms, Models, and Teachers' Work*. Paper presented at the Annual Meeting of the American Educational Research Association, San Diego, CA, USA.

Colace, Francesco, Santo, D., Massimo, Gagliardi, & Nicoletta. (2008). *Multimedia Learning in Advanced Computer based Contexts: 'Discovering Trier'*. Paper presented at the 3rd International Conference on Information and Communication Technologies: From Theory to Applications, Damascus, Syria.

Connolly, T., Stansfield, M., & McLellan, E. (2006). Using an online games-based learning approach to teach database design concepts. *The Electronic Journal of e-Learning, 4*(1), 103-110.

Deming, W. E. (1986). *Out of the Crisis*. Cambridge, MA: MIT Press.

Deutscher Bildungsserver. (2017). Bildungspläne / Lehrpläne der Bundesländer für allgemeinbildende Schulen. Retrieved from http://www.bildungsserver.de/Bildungsplaene-Lehrplaene-der-Bundeslaender-fuer-allgemeinbildende-Schulen-400.html

Everson, H. (n.d.). Barry Zimmerman. Retrieved from http://learningandtheadolescentmind.org/people_04.html

Fabricatore, C. (2000). *Learning and videogames: An unexploited synergy.* Paper presented at the Annual Convention of the Association for Educational Communications and Technology (AECT), Long Beach, CA, USA.

Fagin, B. S., Merkle, L. D., & Eggers, T. W. (2001). *Teaching computer science with robotics using Ada/Mindstorms 2.0.* Paper presented at the Annual ACM SIGAda International Conference on Ada, Bloomington, Minnesota.

Fullan, M. (2007). *The new meaning of educational change* (4 ed.). New York, NY: Teachers College Press.

Gee, J. P. (2003). What video games have to teach us about learning and literacy. *Computers in Entertainment (CIE), 1*(1), 20-20.

Grandell, L., Peltomäki, M., Back, R.-J., & Salakoski, T. (2006). *Why complicate things? Introducing programming in high school using Python.* Paper presented at the 8th Australasian Conference on Computing Education, Hobart, Australia.

Guerra, V., Kuhnt, B., & Blöchliger, I. (2012). *Informatics at school - Worldwide*. Retrieved from https://fit-in-it.ch/sites/default/files/small_box/Study%20Informatics%20at%20school%20-%20Worldwide.pdf

Heininger, R., Prifti, L., Seifert, V., Utesch, M., & Krcmar, H. (2017). *Teaching How to Program With a Playful Approach: A Review of Success Factors.* Paper presented at the IEEE Global Engineering Education Conference (EDUCON2017), Athens, Greece.

Holderness, S. (2016, 27.01.2016). Python at Google. *Why Python?* Retrieved from https://www.codeschool.com/blog/2016/01/27/why-python/

Iglesias, A., & Gálvez, A. (2008). *Effective BD-Binding Edutainment Approach for Powering Students' Engagement at University through Videogames and VR Technology.* Paper presented at the Third International Conference on Convergence and Hybrid Information Technology (ICCIT'08), Washington, DC, USA.

Ivanović, M., Budimac, Z., Radovanović, M., & Savić, M. (2015). *Does the choice of the first programming language influence students' grades?* Paper presented at the 16th International Conference on Computer Systems and Technologies, Dublin, Ireland.

Jantke, K. P., & Woelfert, C. (2012). *A Trojan research tool invading private homes: Concepts and implementations of playful learning*. Paper presented at the 1st Global Conference on Consumer Electronics (GCCE), Tokyo, Japan.

Jones, S. (2002). *The Internet Goes to College: How Students Are Living in the Future with Today's Technology*. Retrieved from https://eric.ed.gov/?id=ED472669

Kljajić Borštnar, M. (2012). Towards understanding collaborative learning in the social media environment. *Organizacija, 45*(3), 100-107.

Lillard, A. S. (2013). Playful learning and Montessori education. *American journal of play, 5*(2), 157.

Linehan, C., Kirman, B., Lawson, S., & Chan, G. (2011). *Practical, appropriate, empirically-validated guidelines for designing educational games.* Paper presented at the SIGCHI Conference on Human Factors in Computing Systems (CHI'11), New York, NY, USA.

McGonigal, J. (2011). *Reality is broken: Why games make us better and how they can change the world*. New York: The Penguin Press.

Mori, N., & Lokar, M. (Producer). (2016, 02.02.2017). A New Interactive Computer Science Textbook in Slovenia. [Presentation] Retrieved from http://issep2016.ens-cachan.fr/talks/ISSEP2016_Mori_Lokar_presentation.pdf

Netzwerk Digitale Bildung. (2015, 1st December 2015). Umfrage: Eltern erwarten, dass Schulen in Deutschland Digitale Bildung vorantreiben. Retrieved from http://www.netzwerk-digitale-bildung.de/presse/umfrage-eltern-erwarten-dass-schulen-in-deutschland-digitale-bildung-vorantreiben/

Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education, 52*(1), 1-12.

Pivec, M., & Moretti, M. (2008). *Game Based Learning: Discover the Pleasure of Learning*. Lengerich:: Pabst:Science: Publishers.

Prechelt, L. (2000). An empirical comparison of C, C++, Java, Perl, Python, Rexx and Tcl. *IEEE Computer, 33*(10), 23-29.

Prensky, M. (2002). The Motivation of Gameplay or, the REAL 21st century learning revolution. *On The Horizon, 10*(1).

Prensky, M. (2005). Engage Me or Enrage Me: What Today's Learners Demand. *Educause review, 40*(5), 60.

Prifti, L., Knigge, M., Kienegger, H., & Krcmar, H. (2017). *A Competency Model for "Industrie 4.0" Employees.* Paper presented at the 13. Internationalen Tagung Wirtschaftsinformatik (WI 2017), St. Gallen.

Radenski, A. (2006). *Python First: A lab-based digital introduction to computer science.* Paper presented at the Eleventh Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 06, Bologna, Italy.

Robinson, R., Molenda, M., & Rezabek, L. (2008). Facilitating Learning. In A. Januszewski & M. Moleda (Eds.), *Educational Technology: A Definition with Commentary*. New York: Lawrence Erlbaum Associates.

Simionescu, S., & Marian, M. (2016). *A playful approach in course structuring for an effective student evaluation.* Paper presented at the 20th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania.

Staatsinstitut für Schulqualität und Bildungsforschung München. (2017). LehrplanPLUS. Retrieved from http://www.lehrplanplus.bayern.de/fachlehrplan/fos/13/informatik/wahl-s-abu-g-w

Taylor, K., & Miller, C. C. (Producer). (2015, 20.1.2016). De Blasio to Announce 10-Year Deadline to Offer Computer Science to All Students. *The New York Times*. Retrieved from http://www.nytimes.com/2015/09/16/nyregion/de-blasio-to-announce-10-year-deadline-to-offer-computer-science-to-all-students.html?_r=0

The Python Software Foundation. (2017, Feb 14, 2017). Python 3.6.0 Documentation. Retrieved from https://docs.python.org/3/

Tillmann, N., De Halleux, J., Xie, T., Gulwani, S., & Bishop, J. (2013). *Teaching and learning programming and software engineering via interactive gaming.* Paper presented at the 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA.

Trilling, B., & Fadel, C. (2009). *21st century skills: Learning for life in our times*. San Francisco, CA: John Wiley & Sons.

Van den Akker, J. (2007). *Curriculum design research.* Paper presented at the An introduction to educational design research, Shanghai (PR China).

Van den Akker, J., & Voogt, J. (1994). The use of innovation and practice profiles in the evaluation of curriculum implementation. *Studies in Educational Evaluation, 20*(4), 503-512.

Zimmerman, B. J. (2002). Becoming a self-regulated learner: An overview. *Theory into practice, 41*(2), 64-70.