Developing Complex Problem-Solving Skills: An Engineering Perspective

https://doi.org/10.3991/ijac.v12i3.11067

Sigrid Schefer-Wenzl (✉)
University of Applied Sciences Campus Vienna, Austria
sigrid.schefer-wenzl@fh-campuswien.ac.at

Igor Miladinovic
University of Applied Sciences Campus Vienna, Austria

Abstract—While technical skills remain the core foundation for engineering graduates, professional competencies, including communication, teamwork, and complex problem solving, are increasingly important to succeed in work environments. To enhance the employability of our students, we have integrated several professional skills courses into our "Software Design and Engineering" master study curriculum, and combined each of them with a technical course. In this paper, we present a blended learning course design including didactic methods for teaching complex problem solving. The course is intertwined with a lecture on software integration topics, which enables the students to apply their complex problem-solving skills on real projects in a domain-specific context.

Keywords—Professional skills, complex problem solving, blended learning, engineering education

1 Introduction

Over the last years, trends in the industry have shown a strong shift from hardware towards software, resulting in an increased need for highly skilled software engineers, with both technical and professional skills. Technical skills refer to domain-specific knowledge, such as programming, mastering computer networks, and developing electronic components. Professional skills (also referred to as soft skills) include competencies needed for a profession beyond technical skills, such as complex problem solving, teamwork, and communication skills [1, 2]. They can be applied in different domains, jobs, and situations.

In the curricula of engineering studies, the main focus is usually on developing technical skills. Therefore, employers around the world frequently state a noticeable lack of professional skills among graduates of engineering studies (see, e.g., [3, 4, 5]). To address this issue, we have integrated several professional skills courses into the curriculum of our master study program "Software Design and Engineering". Moreo-

ver, we combined each professional skills course with a technical course, where students need to apply professional skills on a concrete domain problem.

In this paper we present our blended e-learning course design for "Complex Problem Solving" focusing on didactic methods. Complex problem solving is one of the most requested professional skills from employers (see, e.g., [2]). The main goal of our course was to create a motivating learning environment, where students would be able to individually identify and continuously improve their development potential by targeted learning. Additionally, the students should be able to apply a structured approach for solving complex problems in a real software project in virtual teams.

We designed the course based on Goldratt's "Theory of Constraints" [6] and a set of logical tools presented by Dettmer in [7]. We combined it with the "Software Integration" course, where students have to apply complex problem-solving methods on problems in a real software integration project. In a mix of in-class and e-learning units, the students work on defined tasks, supported by their lecturers. They start with the definition of a Goal Tree of a real software integration project. Thereafter they analyze the current project circumstances and create a Current Reality Tree (CRT). Driven by the Goal Tree, the students identify the Future Reality Tree (FRT) as the desirable behavior of the system. Conflicts between CRT and FRT are resolved by applying the Evaporating Clouds approach.

This paper is structured as follows. Section 2 explains the term complex problem solving and introduces associated challenges. Section 3 presents the course design for teaching complex problem solving in an engineering degree program. Section 4 contrasts our work with related approaches. Finally, Section 5 concludes the paper.

2 Complex Problem Solving

The ability for complex problem solving is one of the central competencies for succeeding in the 21st century [8, 9, 10, 11]. Complex problems can be referred to as ill-defined problems. In contrast to well-defined problems, they have no clear problem or goal definition, and it is not clear how to progress towards the goal. Typical attributes of complex systems are (see, e.g. [12]):

- Complexity of the problem, often represented by a huge number of relevant and interrelated variables
- Connectivity and dependencies between involved variables
- Dynamics reflecting changing time and developments within a system
- Intransparency of involved variables
- Many goals leading to goal conflicts on different levels

Due to these factors, solving complex problems is often associated with failure and lack of progression. In addition, several psychological phenomena further impede complex problem solving. When faced with a complex problem, typical psychological effects are [13]: a reduced intellectual level due to decreased self-reflection, a reduced realization intention, stereotyping, a tendency for fast actions, including a higher readiness for risks, violation of rules, and an increased tendency to escape the situation.

Cross-cultural differences and failures during the planning and acting stages also exist.

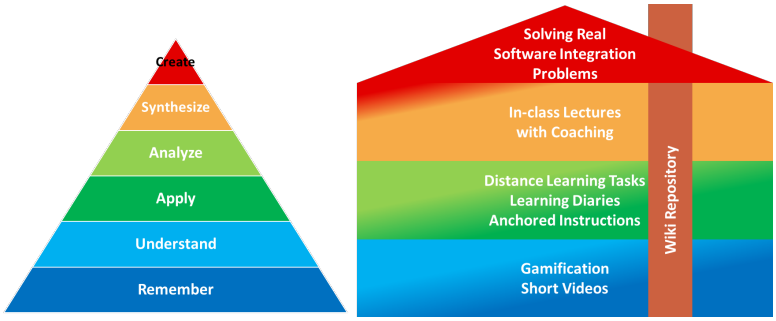

Fig. 1. Learning Stages Compared to Bloom's Taxonomy

The majority of software engineering and integration problems is of complex nature. Such problems have a number of interacting systems, require many different competencies and involve multiple stakeholders, each with different goals that can change over time. Therefore, teaching strategies and techniques for solving complex problems is a key objective in most engineering studies. We chose to integrate a dedicated course teaching methods and tools for complex problem solving into our master program "Software Design and Engineering" and combined it with a technical course on "Software Integration". The "Complex Problem Solving" course is based on a set of logical tools, which are supposed to convert complex real-world situations into easy-to-read and absorb cause-and-effect diagrams [6, 7].

3 Method Mix

The "Complex Problem Solving" course is structured in five modules. Each module starts with a distance-learning phase followed by an in-class unit. Figure 1 shows the methods of the course and their mapping to the knowledge levels of Bloom's taxonomy [14]. For each level we defined clear tasks students need to accomplish. The first levels are Remember and Understand according to Bloom's taxonomy. These levels are addressed by introductory course games. The Applied and Analyze levels are driven by the distance-learning exercises and the learning diaries. In-class hours focus on the Synthesize level. Finally, the solution of real complex problems supported by the methods of the course enable the students to move to the Create level.

Throughout the complete course, students are working on a Wiki knowledge repository [15]. All students are able to contribute to the knowledge base by collaboratively creating a repository with examples from their own experiences about applying CPS methodology to software problems. This is important, as the literature on applying the methods presented in [6, 7] on software problems is very limited. The majority of currently available examples is from production and marketing areas. From year to

year selected inputs will remain in the repository forming a solid learning basis for the future students.

3.1 First learning stage

We use gamification elements and short videos to introduce and motivate new topics throughout the course (see Figure 1). Using gamification in different educational contexts is known to be one way to enhance learner motivation and to improve learning outcomes by capturing the interest of learners and inspiring them to continue learning [16, 17].

The course starts with gamification elements to demonstrate the basic idea of Goldratt's "Theory of Constraints". In a distance-learning phase, students play two types of an online dice game showing how a system's output depends on the output of the subsystems. Students have to reflect on their observations and to link them with their knowledge on software projects. In the in-class hours, students are playing a real dice game with one group of students simulating a system with a limited capacity. Each student represents a dedicated subsystem in a system chain. Other students are observing, working on optimizing this system, and reflecting the results. Using this gamification element students are able to remember and understand the core idea of solving complex problems. Similarly, other gamification elements are used for different topics in the lecture to leverage lessons learnt in distance and in-class units.

Short videos, such as on the Cynefin Framework [18, 19], introduce students to the topics concerning the complexity levels of problems at the beginning of the course. In further learning stages, short videos are used to enable ubiquitous learning.

3.2 Second learning stage

In the distance-learning phase, students are confronted with a new course topic and complete exercises supported by online learning material, including text, videos, and games. To complete the exercises, students need to learn a new topic and apply it to a certain software problem. As our master students already have a strong software engineering background, they need to connect new topics with problems familiar to them. The purpose in this learning stage is to anchor the new topics with their existing knowledge. The findings are stored in the Wiki repository.

Students reflect their insights and progress in a learning diary, which is reviewed by the lecturers before the next in-class unit. Inputs from the learning diary are used to identify which topics require further explanation. Additionally, the learning diary documents the individual learning progress and supports motivation for further learning.

3.3 Third learning stage

In-class time is devoted to deepening the topics from the distance-learning phase and relating them to software engineering problems. Students discuss their findings from the distance-learning units supported by coaching of the lecturers.

Furthermore, students are encouraged in group discussions to generate new ideas for problem solving. The students' distance-learning submissions are peer-reviewed, and the reviews are used to exchange different approaches to solve a problem. All exercises are stored in the Wiki repository.

3.4 Final learning stage

The last module is intertwined with projects from the "Software Integration" course. In teamwork, the students have to apply the methods learned in "Complex Problem Solving" systematically in a real complex software integration project. According to the methods from [6, 7], the students define a Goal Tree for their software integration project. Subsequently, they perform a gap analysis between the current project circumstances and the desired state of the project and create a Current Reality Tree (CRT). Driven by the Goal Tree, the students identify the Future Reality Tree (FRT) to verify that actions taken will lead to the desirable results. Conflicts between CRT and FRT are resolved by applying the Evaporating Clouds approach. The lecturers of both courses, "Complex Problem Solving" and "Software Integration", mentor the whole project. Finally, the outcome of each team is peer-reviewed by two other teams.

4 Related Work

To the best of our knowledge, we only found very little research outlining didactic methods or course concepts for teaching complex problem solving. However, several approaches emphasize the importance of integrating the development of complex problem-solving skills into different kinds of academic curricula. For instance, Bautista [20] investigated the outcomes of teaching two different complex problem-solving models to physics students and concluded that students being taught a certain kind of constructive instruction model were performing significantly better in solving complex problems in other subjects.

Kirkley [21] describes the development of teaching different approaches on complex problem solving. In early years students had to transfer rather abstract problem-solving models to certain topics on their own. Later on, more domain-specific models were taught. He concludes with a set of principles for teaching problem-solving skills.

In [22], the benefits and drawbacks of integrating gamification elements into engineering classrooms are discussed. They especially point out the motivational factors as well as the development of certain professional skills, such as leadership, teamwork, goal-setting and critical thinking, that are associated with this kind of learning material. Other authors, who analyzed gamified engineering courses, state that students report improved content comprehension, retention and recap (see, e.g. [23]).

5 Conclusion

Several surveys among employers show that graduates from engineering studies frequently lack professional skills, including complex problem solving. Hence, we integrated several different kinds of soft skills into the curriculum of our master study program "Software Design and Engineering". In this paper, we show the design of our blended learning course "Complex Problem Solving", which is intertwined with a lecture on software integration topics. Thus, students have to apply and extend their knowledge gained on complex problem solving in a real software integration project. By applying a selected mix of different teaching methods, our course design follows a four-layered approach of consecutive learning stages. At the time of writing this paper, the course was running for the first time. In future work, we will present evaluation results as well as insights from implementing our course.

6 References

[1] S. Gardelliano, "UNIDO competencies. Strengthening organizational core values and managerial capabilities," Technical report, United Nations Industrial Development Organization UNIDO, 2002.

[2] F. Sanchez Carracedo at al., "Competency Maps: An Effective Model to Integrate Professional Competencies Across a STEM Curriculum," In Journal of Science Education and Technology vol. 27, pp. 448–468, 2018. https://doi.org/10.1007/s10956-018-9735-3

[3] H. Jang, "Identifying 21st Century STEM competencies using workplace data," In Journal of Science Education and Technology, Vol. 25, Nr. 2, pp. 284–301, 2016.

[4] Hart Research Associates, "Falling short? College learning and career success," Association of American Colleges and Universities, Washington, DC, 2015

[5] S. A. Hajkowicz, A. Reeson, L. Rudd, A. Bratanova, L. Hodgers, C. Mason, and N. Boughen, "Tomorrow's digitally enabled workforce: Megatrends and scenarios for jobs and employment in Australia over the coming twenty years," In Commonwealth Scientific and Industrial Research, Canberra, Australia, 2016.

[6] E. M. Goldratt, "Theory of Constraints," North River Press, 1990.

[7] H. W. Dettmer, "The Logical Thinking Process. A Systems Approach to Complex Problem Solving," American Society for Quality, 2007.

[8] D. Dörner J. and Funke, "Complex Problem Solving: What It Is and What It Is Not," In Frontiers in Psychology, vol. 8, article 115. https://doi.org/10.3389/fpsyg.2017.01153

[9] P. Griffin and E. Care, "The ATC21S method," In Assessment and Teaching of 21st Century Skills, eds. P. Griffin and E. Care (Dordrecht, NL: Springer), pp. 3–33, 2015. https://doi.org/10.1007/978-94-017-9395-7_1

[10] National Research Council, "Assessing 21st Century Skills: Summary of a Workshop," Washington, DC: The National Academies Press, 2011.

[11] M. Demaria, Y. Hodgsona and D. Czecha, "Perceptions of Transferable Skills among Biomedical Science Students in the Final-Year of Their Degree: What are the Implications for Graduate Employability?" In International Journal of Innovation in Science and Mathematics Education, Vol. 26, Nr. 7, pp. 11–24, 2018.

[12] J. Funke, "Complex problem solving," in Encyclopedia of the Sciences of Learning, Vol. 38, ed. N. M. Seel (Heidelberg: Springer), pp. 682–685, 2012. https://doi.org/10.1007/978-1-4419-1428-6_685

[13] D. Dörner, "On the difficulties people have in dealing with complexity," In Simulat. Gam. vol. 11, pp. 87–106, 1980. https://doi.org/10.1177/104687818001100108

[14] B. S. Bloom, "Taxonomy of educational objectives, handbook I:
[15] Cognitive domain," New York: David McKay, 1956.
[16] K. Parker, J. Chao, "Wiki as a Teaching Tool," In *Interdisciplinary Journal of E-Learning and Learning Objects, Vol. 3, Nr. 1*, pp. 57-72, 2007. https://doi.org/10.28945/386
[17] G. Barata, S. Gama, J. Jorge, D. Goncalves, "Improving Participation and Learning with Gamification," In Proc. of the 1st International Conference on Gameful Design, Research, and Applications, Toronto, Ontario, Canada, 2013. https://doi.org/10.1145/2583008.2583010
[18] M. Kosa, M. Yilmaz, R. O'Connor, P. Clarke, "Software Engineering Education and Games: A Systematic Literature Review," In Journal of Universal Computer Science, Vol. 22, Nr. 12, pp. 1558-1574, 2016.
[19] D. J. Snowden, "Liberating Knowledge", In Liberating Knowledge. CBI Business Guide. London: Caspian Publishing, 1999.
[20] D. J. Snowden and M. E. Boone, "A Leader's Framework for Decision Making," In Harvard Business Review, vol. 85, nr. 11, pp. 69–76, November 2007.
[21] R. G. Bautista, "The Convergence of Mayer's Model and Constructivist Model towards Problem solving in Physics," In Journal of Education and Practice, Vol. 3, Nr. 10, pp. 33-41, 2012
[22] Kirkley, Jamie, "Principles for Teaching Problem Solving," Plato Learning, Inc., 2003.
[23] M. Nino and M. A. Evans, "Fostering 21st Century Skills in Constructivist Engineering Classrooms with Digital Game-Based Learning", In: IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, Vol. 10, Nr. 3, August 2015. https://doi.org/10.1109/RITA.2015.2452673
[24] P. G. F. Matsubara, C. L. Correa da Silva, "Game elements in a software engineering study group: a case study", In: Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track, ICSE-SEET '17, pp. 160-169, Buenos Aires, Argentina, IEEE Press, 2017. https://doi.org/10.1109/ICSE-SEET.2017.8

7 Authors

Sigrid Schefer-Wenzl is a senior researcher and lecturer at the University of Applied Sciences "FH Campus Wien", Favoritenstraße 226, 1100 Wien, Austria. She also works as a lecturer at the WU Vienna and the University of Salzburg. Her current research and teaching activities focus on the fields of software engineering and IT-security. Sigrid has published the results of her work in top ranked journals and presented her work at various international conferences.

Igor Miladinovic is head of the degree program "Computer Science and Digital Communications" and "Software Design and Engineering" at the University of Applied Sciences "FH Campus Wien", Favoritenstraße 226, 1100 Wien, Austria. He worked for more than 10 years on leading positions at Alcatel-Lucent (later Nokia) in the area of telecommunication software and in parallel as a lecturer at two universities. His research interests cover telecommunication networks, software engineering and IoT, with over 40 publications in international journals, conferences and as book chapters. (email: igor.miladinovic@fh-campuswien.ac.at).

This article is a revised version of a paper presented at the International Conference on E-Learning in the Workplace 2019 (ICELW 2019), held in June 2019, at Columbia University in New York, NY, USA. Article submitted 2019-06-19. Resubmitted 2019-08-05. Final acceptance 2019-08-23. Final version published as submitted by the authors.