Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 154 (2019) 137–145

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Congress of Information and Communication Technology, ICICT 2019

Deep-Learning-Based Agile Teaching Framework of Software Development Courses in Computer Science Education

Jian Yang[a]*, Xiao Ling Zhang[a], Peng Su[a]

[a]*Department of Computer Science, Dali University, Dali, Yunnan, China, 671003*

Abstract

Compared with other courses, especially non-engineering ones, software development courses teaching (SDCT) has many unique characteristics, such as higher requirement to students' logical reasoning, the need for students to have comprehensive problem-solving capabilities, good communication and teamwork skills, etc. The practices of Agile Methodology can be compatible with these characteristics. Therefore, it is a matter of course to integrate Agile Methodology with SDCT. This paper analyses the problems and difficulties in the practices of applying agile teaching in SDCT. In addition, it uses an agile teaching project graph instead of linear teaching schedule to control teaching progress. Moreover, a deep generative model is used to obtain the abstract representation of students' implicit skill status so that the construction and implementation of agile teaching project graph can be carried out. Finally, a deep-learning-based agile teaching framework for SDCT is proposed, which has the potential to positively transform SDCT's future.

Keywords: Agile Teaching, Software Development Course Teaching, Deep Learning, Deep Generative Model, Implicit Skill Status

1. Introduction

The programming ability is one of the core objectives of undergraduate teaching of computer specialty. It directly determines the practical capabilities of the students, and affects their further graduate study and employment

* Corresponding author. Tel.: +86-0872-2219600; fax: +86-0872-2219600.
E-mail address: yangjian@dali.edu.cn

10.1016/j.procs.2019.06.021

prospects. With the continuous progress of the research in computer education, varieties of effective teaching methods to improve programming ability have been developed and applied, such as using games as a means of expanding computer science education [1]. Compared with other courses teaching, software development courses teaching (SDCT) has its own characteristics. It is difficult to match these characteristics with the traditional instructional lecture-based teaching, so the improvement of students' abilities is limited in the traditional environment.

Widely used in computer science and engineering, Agile Methodology requires incremental progress in rapid iterations. In every iteration step, continuous communication and mutual learning promote ongoing progress among members of the team, in which everyone can learn how to avoid software bugs as soon as possible. Such characteristics are very suitable for SDCT. We believe that the major differences in the application of agile methods between in real-world software engineering and in education lie in the goals of agile projects, the project reviews and feedbacks in each iteration and the basic skills of team members. Based on this observation, it was found that one of the most difficult problems in integrating agile practices into SDCT is students' skill quantitative evaluation. Using deep representative learning, implicit evaluation points, which are difficult to be clearly measured, are represented by Deep Learning model.

Integrating Agile Methodology with SDCT and making full use of modern AI technology to help to accurately monitoring and adjusting the teaching progress will greatly improve the effectiveness and efficiency of SDCT. Be different from the practices of agile teaching in other courses, this paper discusses the agile practices according to the features of SDCT. It is also different from applying agile methods to software development industry, because we are talking about agile teaching in SDCT. The structure of the following sections is as follows: Firstly, the research progress of agile teaching and Deep Learning is summarized in Section 2, and then the practices of agile teaching in SDCT are proposed in Section 3. In Section 4, aiming at the technical problems introduced in Section 3, a teaching framework based on Deep Learning model is proposed. At last, a conclusion is drawn in section 5.

2. Related work

2.1 Agile teaching

Since Agile Methodology was put forward in 2002[2], its applications in education area has been studied and approached. Chun (2004) explained the goals and architecture of Agile Learning/Teaching Methodology (ALTM), and proposed that the teaching and learning aspects of ALTM should be agility with effective communication and process management [3]. Maria Jakovljevic (2012) studied agile group dynamics and knowledge sharing in IT related classroom teaching, and analyzed the impact of group dynamics on promoting students' abilities [4]. Paolo Maresca [5] (2013) studied the impact of group consonance and resonance in agile teaching and found that collaborative learning could be more effective than independent learning, but an effective coordinated resonance in agile teams was needed before class beginning. Based on the idea of Extreme Programming, a student-centered teaching conceptual framework, Extreme Pedagogy, was proposed in 2015[6]. Many new teaching softwares and frameworks have also been developed and applied [13-14].

No longer guided by a linear teaching plan, in agile teaching, students learn through the gradual realizations of the iterative projects and the information exchanges between team members. It can be seen as a kind of self-regulated learning. In literature [7], with well-designed experiments, a comparison between task-based learning (TBL) in self-regulated education and traditional instruction-based learning was made, and the author found that there were higher levels of intrinsic goal orientation, task value, using of elaboration learning strategies, critical thinking, metacognitive self-regulation, effort regulation and peer learning in TBL. This observation enlightens the core research content of this paper. However, agile methods in SDCT still need some changes in practice.

2.2 Deep learning

In 2006, Hinton et al. proposed using a layer-wised pre-training method to train the network parameters of each layer in a Multi-Layer Perception, which solved the problem that the objective function of traditional multi-layer neural network is difficult to optimize [8]. From then on, as a new area of machine learning, Deep Learning emerged

and developed rapidly. In addition to Restricted Boltzmann Machines (RBM), the basic component of DBN, the typical structures in Deep Learning include auto-encoder, Convolution Neural Network, Deep Stacking Network, and Recurrent Neural Network and so on. Since 2006, Deep Learning has attracted a lot of attention and has been applied successfully in many areas, such as speech & image recognition [16], machine health monitoring [17], etc. However, the applications of Deep Learning in teaching technology are relatively rare, at least so far.

As a kind of machine learning, through hierarchical information process, Deep Learning aims at feature extraction, pattern recognition and classification, and is also used in teaching [15]. An important goal of using Deep Learning is to extract effective features from data by supervised or unsupervised process. For applying agile practices in SDCT, Deep Learning provides the capabilities of expressing and learning implicit features, and it is responsible for integrating these features into specific teaching practices, such as the construction of agile projects, teaching assessment and evaluation of students' skill status.

3. Agile practices in SDCT

3.1. Characteristics of SDCT

Compared with other courses teaching, SDCT has many unique characteristics:

The requirements of logical reasoning and practical abilities are higher, so it is necessary to accumulate experiences and knowledge in continuous practices and promote the capability of solving problems with different development tools.

Although software development ability can be evaluated by using knowledge points as main benchmark, the familiarity degree of knowledge point is a vague and abstract concept, so its evaluation is uncertain. It is difficult to judge whether the students have mastered certain knowledge points and related skills by a specific question.

The software development ability is closely related to not only the language itself, but also many other basic and prep courses. It remains to be discussed whether the evaluation based on knowledge points can well reflect students' comprehensive application capabilities under the background of multi-course integration.

The cultivation of programming skills involves a large number of "tacit knowledge" transformations, which cannot be simply promoted by huge problem-solving;

The improvement of students' programming skills is closely related to their cognitive, emotional and creative abilities [11]. It is also directly related to students' logical reasoning and personal interests. Therefore, it is not suitable to use traditional teaching methods as well as general teaching pattern and linear teaching schedule.

In software development, to solve the same problem maybe there are many different ways. However, which one is elegant, scalable, conforms to software development specifications, and reflects the tacit abilities of developers? To do such a judgment requires relatively accurate characteristic representations of different solutions, which are extremely complicated and difficult to gain.

Traditional classroom teaching is instructional, lecture-based, in which students' learning schedule is pre-arranged and regulated according to teaching plan. Because of the above characteristics, the traditional teaching practices cannot get good results in SDCT. Students fight individually, and the fixed teaching plan slows down the excellent students, but for the students with poor foundation, such teaching pattern cannot arouse their interests, which results in gradually giving up learning. Therefore, taking different measures according to different situations, teaching students in accordance with their aptitude and embracing changes with agile practices, are necessary for meeting students' needs as much as possible and improving teaching quality.

3.2. Agile practices in SDCT

It is necessary to transform teaching practices into corresponding agile ones. However, the goal of SDCT is not exactly the same as that of real-world software engineering project development. Therefore, some innovations of agile teaching are required in SDCT. Next, we will discuss these improved practices in SDCT from several aspects.

3.2.1 Teachers' adaption.

In industrial software projects, the managers are required as to provide guidance and the necessary information to the teams and clear the work barriers. In SDCT, teachers are the "super-manager" of all agile project units. Similar to industrial projects, the role of teachers is weakened because agile projects are self-organized and self-managed. However, in order to achieve the ultimate teaching goals, they still have to undertake some necessary and special works. These works include:

making adaptable changes in specific teaching objectives from the perspective of agile teaching needs,
exploring the relationship between knowledge points and agile project units,
providing support for the design and assessment of agile projects from the data perspective,
designing a series of agile teaching project units aiming at knowledge points and learning objectives,
replacing the fixed schedule with a combination of nonlinear progressive project units,
formulating schedule specifications and other relevant high-level agile management specifications,
using IT methods to dynamically meet the difference of students' skill status,
Designing evaluation methods to adapt to changes and truly reflect students' skill status.

In SDCT, the teaching process is no longer linear. The non-linear process helps to embrace the change, to adapt to different students' skill status, to satisfy students' needs, and to stimulate students' interests in programming. A comparison between linear teaching plan and nonlinear agile teaching project graph is shown in Fig. 1. The graph consists of a number of agile project units forming a directed acyclic graph. The jump between the intermediate units is associated with the evaluation results of the completed previous units, which forms a framework similar to a multi-dimensional Markov model.

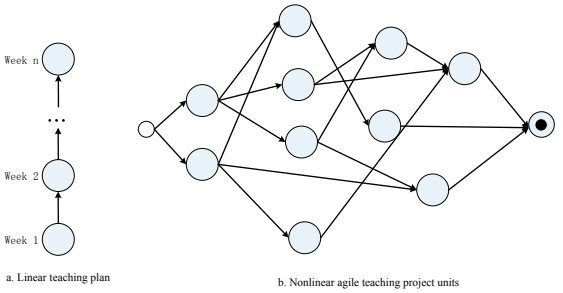

Fig. 1. comparison between linear teaching plan and nonlinear agile teaching project graph.

It is an obvious difference between industrial software projects and SDCT in which there is no customer to put forward and control functional requirements. Therefore, designing good requirements and suitable progressive relationships between project units by teachers are very important to promote teaching and learning in agile SDCT.

3.2.2 Composition and communication of project team.

In XP (Extreme Programming), pair-working is one of the best twelve practices and used in the teaching of IT courses [9]. The pairs and project teams should have been carefully planned before the beginning of the course. It is feasible that a more competent student is paired with a less competent student who takes on the responsibility of writing code (the driver in XP). The number of people in a project team should vary from three to four and the personnel pairing rules are similar to pair programming. In each team, a student acts as a communicator responsible for facilitating the smooth progress of the project iterations, organizing regular reviews and seminars, communicating with teachers, and providing incentives to finish the projects.

Before the beginning of a course, teachers should carefully prepare an agile project implementation plan, in which includes the iteration cycle and the delivery time of each project unit, the discussion criteria of project teams, and so on. The agile project implementation standards should be explained to all students. Whether a project is finished

according to the specifications should be included in the checkpoint as one of the final evaluation results.

3.2.3 Self-management, guidance and monitoring of agile teaching project.

After the implementation plan of agile project units is specified, learning-aimed projects can be approached and self-managed. When a project starts, the corresponding team is led by its communicator to decompose the project objectives, assign pairs and tasks, implement functions, and call the summary and feedback meetings regularly. When a team encounters problems that cannot be solved by the team itself, the teacher should help to deal with it. In this process, the teacher must pay close attention to the progress of each project unit, and guide the direction at the appropriate time. When a team completes a project unit, its achievement appraisal is carried on by the teacher, after that the team enters the next agile project unit according to the appraisal result.

3.2.4 Quick iteration of agile project.

The most important practices of Agile Methodology are iterative delivery and quickly achieving goals. These practices should also be included in agile SDCT. During the whole course cycle, each of agile teams should complete 4 to 6 project units, which cover different knowledge points of different learning stages of the course. Aiming at finishing a project unit, the students master related knowledge points by information query, reading books, discussion, thinking collision and other communication forms, instead of by lecture-based learning. With these elaborate project units, the learning of knowledge and skills has become incremental and iterative.

Agile project units in SDCT should be coherent for the teaching goal. Based on students' assessment results, moving from one project to the next reflects the sustainability of Agile Methodology. The jump between projects is optional, which reflects that the whole framework can respond to different students' skill status. In addition, each project unit should not be set up too complex and fits the needs to master the core knowledge of different stages, as well as taking the prior knowledge into account.

3.2.5 Evaluation and assessment.

In agile SDCT, teaching is driven by project units, and the assessment method of learning results should be changed accordingly instead of evaluating students' skills based on final exams. The evaluation includes the ability of full control of the project process by the team, the completion of a single project unit, the comprehensive completion of all units, the self-evaluation and mutual evaluation of project team members, and so on.

3.3 Problems and difficulties

There are several problems that still have to be solved in agile SDCT:

Feature extraction of the abstract implicit skill status. The knowledge points and topics of a course are explicit, but students' skill status is implicit and abstract. The correlation degree between questions and knowledge points is also difficult to quantify accurately. It is difficult that the teacher evaluates the promotion degrees of students' skills driven by the agile project unit with clear and quantitative means.

In the above-mentioned agile teaching practices, the building of agile project units and the construction of the project graph are the keys to success in SDCT. How do we evaluate the coherent relationship between different agile projects? How do the teachers construct a non-linear teaching schedule according to these projects? How do the teachers implement the teaching schedule according to the students' skill status?

The solutions of these problems involve the presentation and evaluation of abstract concepts such as students' skill status. The selection and jumping of agile project units involves the evaluation of the effectiveness of students' skills improvement given by finishing a project, and the evaluation can also be handled with some discriminant models in Deep Learning. In addition, the position of every project unit in the agile teaching process graph needs to be determined according to the relationship of the knowledge points included in these project units.

4. Agile SDCT under the framework of deep learning

To build an effective graph, it is necessary to quantify the students' skill status, which is a difficult problem. The students' situations obtained after projects completed can partly reflect their skill status. These situations include the errors or bugs ranked by severity, the ways to project objectives - such as arrays expected but several variables being used, the projects' completion time, the level of formal code writing, the situations of project progress, task splitting and seminar implementation, documents referring, pair programming, execution of personal tasks, etc.

In the implementation of agile SDCT, the acquisition of transfer probability in sequences of project units is also a difficult problem. It is necessary but difficult to quantify the improvement level of students' skill after project finished. This paper attempts to apply an unsupervised representative learning method of Deep Learning to some of these abstract features.

4.1 Build the graph of project units

Before approaching an agile SDCT, the teacher assembles a number of project units according to the content of the course, and then uses them to build and initialize the project unit graph, in which each unit is related to several knowledge points.

Let $\mathbb{K}=\{k_i\}$错误！未找到引用源。, i.e. $1 \le i \le n$错误！未找到引用源。, be the set of n knowledge points of a course. A vector $\boldsymbol{w}=(w_1, w_2, \ldots, w_n)$ defines the required degree of knowledge points understanding, in which 错误！未找到引用源。w_i is interpreted as the upper bound of the required degree of knowledge k_i.

Let $\mathbb{P}=\{\mathbf{p}^{(j)}\}$ be a vector set of m project units generated by the teacher, and denoted by $\mathbf{p}^{(j)}=(p_1^{(j)}, p_2^{(j)}, \ldots, p_n^{(j)})$错误！未找到引用源。 the j-th project unit. Then $p_i^{(j)}$ 错误！未找到引用源。 is interpreted as the weight of the i-th knowledge point in the j-th project unit. So for each project unit $\mathbf{p}^{(j)}$ 错误！未找到引用源。, we have:

$$\sum_i p_i^{(j)} = 1 \tag{1}$$

This representation reflects the explicit curriculum learning objectives contained in a project. According to the syllabus of the course, there are inherence relations between different knowledge points included in different project units. The initial graph of agile project units can be constructed according to these relations.

4.2 The quantification of students' skill status

The students' skill status is difficult to observe and quantify, but it is also composed of the most important features in agile SDCT. In this paper, we use an unsupervised representative learning method to explore the quantitative representation of skill status.

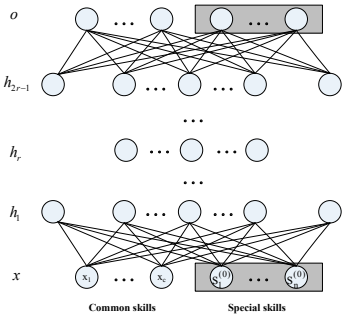

Fig. 2. the abstract student's skill status

A Deep Auto-Encoder model was constructed as shown in the Fig. 2, to quantify the abstract skill status. The inputs to the model are made up of two parts. One is the student common skills, which can be expressed qualitatively or quantitatively, including the abilities of code writing and annotation, program debugging, IDE usage, usage of framework and library function, communication in the process of development, and so on. These skills can be obtained by issuing questionnaires before the project starts, by conducting targeted tests, or by evaluating the completion of a project unit. The other part is the familiar degree of the course special skills, which can be calculated according to the evaluation of the completion degree of the project.

Let $\mathbf{s}^{(i)}$错误！未找到引用源。 be a student's skill status after the *i*-th project unit finished, and $\mathbf{s}^{(0)}=(0)_n$. Then the change of skill status can be denoted as:

$$\Delta\mathbf{s}^{(i)} \approx \mathbf{p}_{result} \text{ e } \mathbf{p}^{(i)} \tag{2}$$

The vector $\mathbf{p}_{result} = (p_1, p_2, \ldots, p_n)$错误！未找到引用源。 defines the scores of each knowledge points in the *i*-th project. Then, the student's current skill status $\mathbf{s}^{(i)}$ can be defined as:

$$\mathbf{s}^{(i)} = \mathrm{Max}\left(\boldsymbol{w},\ \Delta\mathbf{s}^{(i)} + \mathbf{s}^{(i-1)}\right) \tag{3}$$

In which the function Max is a pairwise maximum function and w is the required degree of knowledge points understanding.

After model learning, the output h_r of the middle hidden layer can be extracted as the expression of students' current skill status. Unlike traditional teaching evaluation, which aims at mastering knowledge points, the evaluation of students' skill status in our model includes common skills. Combining the common skill status with the special skill status obtained after a project unit finished, the completion situation of a project can reflect the promotion and enhancements of student's skills exerted by the project, and reflect the implicit relationship between the two kind skill statuses. Therefore, the representation of skill status acquired in this way can be used in the project unit graph in agile SDCT.

4.1. The practices of Agile teaching in SDCT

A Recurrent Neural Network model was constructed as shown in the Fig. 3, to determine how to select the next project unit after a team completes a project unit.

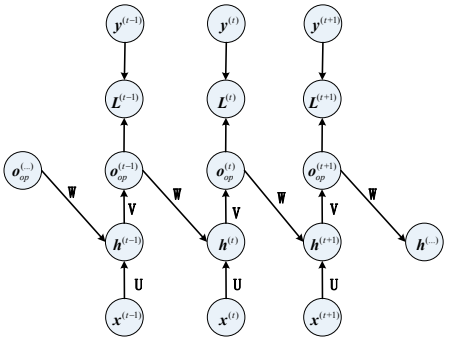

Fig. 3. a RNN model for project unit selection[12]

In Fig. 3, the input of the *t*-th iteration is composed of two parts:

The abstract expression (h_t in Fig. 2) of the student's skill at time t.

A vector $\boldsymbol{o}_{op}^{(t-1)} = (\boldsymbol{o}_1^{(t-1)}, \boldsymbol{o}_2^{(t-1)}, \ldots, \boldsymbol{o}_m^{(t-1)})$, which indicates the probabilities of each project unit chosen by the student at time t-1.

When the network is initializing, the probabilities of all candidate items at same step are divided equally. The model's optimization can be set to a deformation form of the function $\boldsymbol{L}^* = \arg\max(\Delta \mathbf{s}^{(i-1)})$. After learning, the hidden relationship between students' skill status and the probabilities of each project unit selected can be obtained. Here the RNN model used as a discriminant model for project unit selection is due to that:

It can reflect the nonlinear relationships between the selection of project units and the enhancement of students' abilities.

It can reflect the progressive superposition relationships among different project units.

It can reflect the change and the growth degree of students' skills.

According to the student's current skill status and the selected previous project unit that has just finished, the above model can be used to judge which project unit should be more likely selected as the next one to obtain the greatest skill enhancement and achieve the teaching objectives in agile SDCT.

5. Conclusion

Because of the special features of SDCT, traditional instructional, lecture-based teaching is not suitable, and the experimental teaching guided by linear schedule cannot appropriately meet students' different needs. Meanwhile, many practices of agile teaching are fit to be spontaneously integrated with SDCT. This paper analyzes the special features of SDCT, puts forward some improved agile teaching practices, and proposes to construct an agile project unit graph instead of using linear teaching schedule. Different project units are chosen according to the students' current skill status to match different needs. There are some problems need to be solved in the application of agile teaching methods in SDCT, especially evaluating implicit students' skill status and the selection of agile project units. In order to solve these problems, this paper proposes to use Deep Auto-Encoder to map explicit qualitative and quantitative features into the abstract representation of skill status, and then construct a Recurrent Neural Network model to reveal the nonlinear relationship between students' skill status and its improvement stimulated by the project units. The proposed framework provides a novel option to practice agile teaching in SDCT.

References

[1] Mcgill M, Johnson C, Atlas J, et al. Game Development for Computer Science Education[C]// ACM Conference on Innovation and Technology in Computer Science Education. ACM, 2016: 23-44.

[2] Ambler, S. (2002). Introduction to agile modelling. Denver: Ronin International Inc.

[3] Chun A H W, Chun W. The Agile Teaching/Learning Methodology and Its e-Learning Platform[C]// International Conference on Web-Based Learning. 2004: 11-18.

[4] M Jakovljevic. Knowledge management in an ICT classroom an agile group dynamics for innovative teaching and learning[C]// 2012 ISTE International Conference on Mathematics, Science and Technology Education. UNISA, 2012:364-378.

[5] Maresca P, Guercio A. Increasing Consonance and Resonance in Agile Teaching Methodologies[J]. Gstf Journal on Computing, 2014, 3(1):82-92.

[6] D'Souza M J, Rodrigues P. Extreme Pedagogy: An Agile Teaching-Learning Methodology for Engineering Education[J]. Indian Journal of Science & Technology, 2015, 8(9): 828-833.

[7] Janagam D, Suresh B, Nagarathinam S. Efficiency of task based learning and traditional teaching on self-regulated education[J]. Indian Journal of Science & Technology, 2011, 4(3): 308-312.

[8] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural Computation, 2006, 18(7): 1527-1554.

[9] Kuanchin Chen, Alan Rea. Do Pair Programming Approaches Transcend Coding? Measuring Agile Attitudes in Diverse Information Systems Courses[J]. Journal of Information Systems Education, 2018, 29(2): 53-64.

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning[M]. USA:MIT press, 2016: 230-232.

[11] Karl Royle, Jasmina Nikolic. A modern mixture, Agency, Capability, Technology and 'Scrum': Agile Work Practices for Learning and Teaching in Schools[J]. Journal of Education and Social Policy, 2016, 3(3): 37-47.

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning[M]. Posts & Telecom Press, 2017.

[13] Deeb F A, Hickey T. Flipping introductory programming classes using spinoza and agile pedagogy[C]// IEEE Frontiers in Education Conference. IEEE Computer Society, 2017:1-9.

[14] Elahe Javadi, Season Tanner. Design and implementation of an agile teaching framework[C]// Americas Conference on Information Systems 2018. Association for Information Systems, 2018:1-9.

[15] Jian-Min Liu; Yu Xiang, Min-Hua Yang. Construction and application of new intelligent MOOC teaching system based on deep learning neural network in remote sensing course[C]// Proceedings of the 8th International Conference on Information Technology in Medicine and Education. IEEE, 2016:459-462.
[16] Schmidhuber J. Deep Learning in neural networks: An overview [J]. Neural Networks, 2015, 61:85-117.
[17] Zhao R, Yan R, Chen Z, et al. Deep learning and its applications to machine health monitoring[J]. Mechanical Systems & Signal Processing, 2019, 115:213-237.