Blended Learning in Software Engineering Education: the Application Lifecycle Management Experience with Computer-Supported Collaborative Learning

Maksimenkova Olga
School of Software Engineering,
Faculty of Computer Science
National Research University Higher School of Economics
Moscow, Russian Federation
omaksimenkova@hse.ru

Neznanov Alexey
School of Data Analysis and Artificial Intelligence,
Faculty of Computer Science
National Research University Higher School of Economics
Moscow, Russian Federation
aneznanov@hse.ru

***Abstract* — Software engineering education (SEE) process simulates the main professional software lifecycle processes such as analysis, design, construction and maintenance (see SWEBoK, ITIL, etc.). The necessity of meeting both educational needs and requirements from industry explains that using Supported Collaborative Learning (CSCL) techniques in software engineering (SE) should be based on professional tools or on similar to them. The main purpose of this work is to fill the gap between the SEE needs and the current trends in CSCL development. We generalize world experience and suggest the framework of using industry approved methods and tools. We compare CSCL tools and the other collaborative services; analyze the teaching experience of several SE courses supported by different collaborative methods and collaborative web-services. Special attention is paid to formative feedback implementation. Following achieved result we suppose that using best practices from SE will enrich CSCL methodology and tools not only for SE field, but also for other areas of knowledge.**

Keywords — blended learning; collaborative learning; software engineering; education; SE; SEE; CSCL

I. INTRODUCTION

Plenty of *software engineering education* (SEE) processes correlate with the main professional software lifecycle processes such as analysis, design, construction and maintenance. This caused partly by that, *software engineering* (SE) daily routines and activities should be agreed to the requirements to these processes [1]. Therefore, graduates of each level (bachelor or master) in SE should demonstrate skills connected with professional software tools such as integrated development environments (IDE), configuration management systems (CMS), and continuous integration (CI) systems. For this reason, current syllabuses in SE correspond to *software engineering curriculum* 2004 (SEC2004) [2], which define the using of tools as one of the obligatory learning outcomes.

Thus, the deep practical orientation of SEE makes it the area full of collaborative software. Such software like various application lifecycle management (ALM) systems, IDEs, concurrent versions systems (CVS) are applied in educational process not only as a tool but as the subject of study as well. Consequently, the necessity of meeting both educational needs and requirements from industry explains that SEE should be based on professional tools or on similar to them.

The significant question is: "What do all above mean for instructional design?" The answer seems to be understandable intuitively: *computer-based learning* (CAL) and *computer-supported collaborative learning* (CSCL) technics are natural for SEE courses design.

What is about existing collaborative professional and educational software? Both of them bring several troubles in their direct application to SEE. On the one hand, the professional tools of SE are not suitable to the educational purposes. They simply have the other goal and are intended for experienced developers. On the other, the current generation of *learning management systems* (LMS) do not meet the needs of SEE. Typical LMS (like Moodle [3] or eFront [4]) does not contain modules convenient to support software lifecycle artefacts, processes and collaborative activities. Hence, professional SE tools support collaboration in terms of SE as a solution for multi-role development teams. In they turn, LMSs support common instructional collaborative processes and work in terms of education (e.g. in-class co-working and WebQuests [5]).

Specific ways of collaboration in SE settle following educational matters:

- How to conceive SEE by CSCL technics and tools taking into account SE industry needs?
- How to adopt professional software to educational purposes?
- How to hold down significant characteristics of real life SE context?
- How to optimize the learning curve [6] of SE tools?

978-1-4799-8706-1/15/$31.00 ©2015 IEEE 20-24 September 2015, Florence, Italy
Proceedings of 2015 International Conference on Interactive Collaborative Learning (ICL)

The main purpose of this work is filling the gap between the SEE needs and the current trend in CSCL development by generalizing experience of using industry approved methods, tools, and suggesting the relevant educational framework.

II. Stratification and Classification of Collaborative Tools

Collaborative tools themselves are the popular subject of research papers. Area state in the beginning of 2000s was described in [7]. Stocktaking and analysis of current state are not the aim of this report. Moreover, everyone can search Wikipedia for suitable lists of popular collaborative software by license, application field and functionality.

Actually, we do not need consider any other technological basis to collaborative tools except the Internet. This describes by that *Internet* (as global telecommunication network) form is a key infrastructure for almost all collaborative tools since 2000.

Basic classification of collaborative software by "time" and "place" is obvious: time/place of participants of collaboration may be the same or different. Of course, any modern collaborative tool should be scalable and should allow distant communication with notification abilities. Real-time communication has special meaning for the people but also has special busyness and technological requirements.

For the planning and structuring our educational framework, it is useful to distinguish four levels of collaborating tools.

1. Informing tools like news feeds and blogs. This class of tools focuses on conveying the news to the audience.
2. Real-time communicating tools like chats, pagers and messengers. This class of tools supports personal communication of two or several people in form of text messages, audio and video.
3. Discussing tools like Internet groups and forums. This class of tools attend to structured longitude multiuser conversation with full history tracking and search ability.
4. Co-authoring tools like collaborative editors. This class of tools focuses on multiuser creating of some artefacts: UI mockup, source code, etc.

As a rule, tools of higher level include functionality of lower levels in form of additional modules. For example, Microsoft Office Online [8] has embedded Skype functionality [9].

Collaborative platform is a toolset that supports different types of artefacts and the different roles of participants of teamwork. Such platforms necessarily support organizational tasks on some level of complexity: from simple scheduling component up to project management system of full value.

III. Collaborating in Software Engineering

History of *modern software engineering* begun in late 1980s after so called "Software Crisis" of 1970s. It is important for the paper that one of cornerstone of modern SE is figuring out that collaborating in SE is very special in various aspects. In 1975 it was first mentioned by Brooks [10]. Problems in analysis and design was deeply expounded by Jackson in [11]. Problems in organization and collaboration was clarified by DeMarco and Lister in [12].

The central concept of SE is Application Life Cycle Management. It is treated simultaneously as methodology and technological framework based on specific Software Development Life Cycle (SDLC) Model. SDLC is also an abbreviation of a broader term System Development Life Cycle whose extent includes many other phases and approaches. There are a plenty of SDLC models in SE – from the naive Waterfall model up to Agile Development (see [13] for a brief introduction).

To date, software engineering society has the result of several iterations of the Software Engineering Body of Knowledge (SWEBoK) [1] with well-defined formal tasks, processes and roles in SDLC. SEC2004 and draft of its next version [14] clearly fit SWEBoK.

According to SWEBoK software engineers threat SE collaborative platform as extension of configuration management system (CMS) (see short description of significance of CMS in [15]). Other characteristics of industry leading SE collaborative platform are:

1. Supporting concrete SDLC or a set of SDLCs.
2. Supporting majority of the roles in ALM (system analyst, business analysts, project manager, team leader, software developers, etc.).
3. Supporting most of the processes in ALM, including:
   - primary (development, operation, and maintenance of software),
   - supporting (configuration management, quality assurance, verification and validation),
   - organizational (training, process measurement analysis, infrastructure management, portfolio and reuse management, organizational process improvement, management of software life cycle models)
   - optional cross-project (reuse, software product line and domain engineering).
4. Ability to integrate smoothly with other ALM products.
5. High durability and scalability.

IV. Two Sides of CSCL. Education and Technology Interaction

This section contains a brief review of CSCL researches and practices in computer science education (CSE) and SEE. Here and after we will address both CSE and SEE, because investigations in both of them often deals with similar questions. According to the duality of our primary theme "CSCL based on professional software in SEE" in following review we will consider on current state of CSCL techniques applied to studying educational area and on software using in it.

First of all the need of real-life software using in CSE is dictated by tentative educational results. Therefore, turning back to the learning outcomes, we would say that the latest version of computer science curriculum 2013 (CSC2013) [16] defines project experience among the others characteristics of graduates. Actually, the matter began to be discussed a bit earlier. Thus, already in 2009 Joel Spolski in [17] claimed that it had been amazing how easy it had been to sail through a Computer Science degree from a top university without ever learning the basic tools of software developers, without ever working on a team, and without ever taking a course for which you did not get an automatic F for collaborating.

Of course, in this paper we do not have a necessity of discussing the fact of growing all-around interest to CSCL technological artifacts [18]. It was caused mostly by the highly decreased transaction cost and appearance of electronic collaborative tools. Term "collaborative" in the context of electronic tools for teamwork was first used by Douglas Engelbart at Augmentation Research Center and it was back in late 1970s [19]. Routine collaborative technologies in lots of areas quickly grew their popularity and began wide spreading. One of the last additions to this spread is "collaborative manufacturing" [20]. Hence, the reaction of the sphere of education is predictable and is not the exception.

Educational researchers and practitioners concerned on the specificity of CSCL artefacts [18] to particular educational field and to the different types of educational process [21, 22]. For example, Forte in [23] described integrating such open collaboration systems as Wikipedia and OpenStreetMap into the classes of high school and college students. Another example, which presents organizing collaboration while teaching programming in elementary school, may be find in [24]. This article studies interaction ability of TurtleArt to be used for collaborative needs. In spite, the fact that authors presented useful experience it little suits the needs of CSE and SEE.

Nevertheless, to date, specialists have been studying and combining educational methods and software tools suitable for SEE needs more than for a decade [25, 26]. There are several approaches of blended learning and CSCL application to CSE and SEE, which were been described in the literature.

Some researchers focus on such aspects of these educational designs implementation as problems on different stages of CSE [24].

The others, for example, Bati and colleagues, [27] study the influence of students' group size on educational process. In this work authors concerned on the course for beginner programmers designed according to blended learning approach. The paper signed out problems of novice programmers. Nevertheless, it touched only course design and took into consideration two questions: prerequisites (knowledge and skills) required to study programming and psychological aspects of programming. The influence of professional SE software on educational process was not been studied.

At last, some publications describes experience of integrating blended learning, active learning, specialized developers tools and the other methods and activities into SEE [25, 26, 28].

Indeed high technology of collaborative software tools and integrated suites has produced a specific branch of CSCL studies. These studies deals with educational software, which supports different CSCL procedures and processes. However, most of these studies lay outside of classical LMS methodology.

Thus, for example, Sondergaard and colleagues in [29] examined calibrated peer review in technical disciplines. Authors signed out software engineering processes in connection of peers. They also claimed the special requirements to the tools for programming source code review. Unfortunately, this paper addressed only special tools for peer-to-peer assessment and authors did not touch the problem of adoption real-life tools. Peer-review design tacking into account SE problematics is concerned earlier in [30]. The idea of peer review for the needs of SEE was extended by Rajapakse in [31].

Initiative "OneNote for Teachers" [32] build universal instructional collaborative framework using OneNote Class Notebook Creator on top of Microsoft Office 365. Another example of special software can be find in [33]. Authors presented special framework, which supports multiplayer game educational technique. However, despite those framework undoubtedly applied to CSCL method, it also does not support any specific SE process, and has been tested on physics classes of high school.

However, the most remarkable example is the new level of educational collaboration, which is implemented in the Coursefork platform [34]. Coursefork is a platform for collaborating on educational material. Authors also call it "GitHub for course creation". For today, the same initiative group have announced an "open teaching" platform called Trinket, for details see [35].

The reviewed examples demonstrates much researchers' interest in the different aspects of CSCL. Besides, they confirm the existence of a gap in studies, analysis and informative descriptions of professional collaborative software application to the needs of CSCL techniques in SEE. The paper singed out significance of the technological artifacts that support collaboration. In our paper, we consider the artifacts of this type for the needs of SEE.

V. CSCL IN SEE

Highlighting of communication and organizational skills on the one hand and accentuation of the teamwork techniques in ALM on the other explain significance of collaborative tools in SEE. Consideration of these characteristics along with specific subjects and forms of educational process lead us to blended learning style. Previous section shows that CSCL seems to be ideal for supporting SEE. To resolve stated above problems, we suggest combining methods of face-to-face learning with CAL and enriched them with mature SE collaborative technologies.

We strongly believe that the symbiosis of mature ALM infrastructure and breakthrough in CSCL of last decade. Nevertheless, there are several stumbling stones in the current state of academic and educational collaborative tools. We have compared properties of modern CSCL tools and other popular collaborative web-services, using SE industry standards and

curricula requirements as a source of criteria. The examples of evident stumbling stones are:

- source code decoration (it is necessary to use external JS libraries and additional command for code syntax highlight in most collaborating tools);
- versioning considering formal grammar of programming languages and special objects types;
- interpretation of diagrams (ERD, UML, BPMN, etc.) not as pictures but as formal object ready to conversion and simulation;
- access to building and testing environments (for program execution).

SEE have a huge set of concepts in fields of programming, analysis, design, management, etc. Teacher does not discuss those concepts all together but student should have systematic knowledge of SE [36]. To provide student with it we need methodology support and preliminary clarification of ALM processes. Hence, collaborating in SEE should be process-oriented.

A. Minimal SEE requirements

At least following special SE artefacts must be considered:

- source code (with syntax highlighting and folding);
- design diagrams (of ER Models, UML, BPMN, etc.);
- datasets in popular formats (CSV, XML, JSON, etc.);
- references to changesets (with "diffs" and changes history);
- API description (WSDL, JSON Schema, etc.);
- tickets in bug tracking and feature requesting systems.

At least following special SE processes should be considered:

- application field analysis;
- requirements management;
- software design;
- source code construction;
- code reviewing;
- software testing (including unit testing and functional testing).

Some universal project management tasks and processes have specific properties in SE. Firstly planning and scheduling should have direct association with SDLC model used in current SEE track.

B. Educational software requirements

We have analyzed the teaching experience of several SE courses supported by different collaborative methods and different collaborative web-services. Special attention is paid to formative feedback implementation, which is claimed to be the significant part of every constructivist instructional design model.

In addition to listed SE requirements special instructional needs (e.g. teacher-student interaction, instructional design modelling, tasks assignment, and formative feedback) should be satisfied.

C. Adoptation of professional ALM software

The adoption of professional software for the needs of education aims the following:

- simplifying information security requirements and rules;
- optimizing processes definitions and constraints for small groups and multiphase assignments;
- integrating ALM software with LMS via API.

Anyway, *tutoring* is becoming more and more important part of the instructional process. Supportive information in form of *tutorials* are increasing popularity in academy, echoing the way of the industry.

D. Optimizing learning curve of SEE tools

Information technology industry makes big effort to simplify first steps in immanently complex and difficult creative tasks. It has been noted that consuming information products already simplified for everyday life.

Tools intended primarily for scientific tasks, as Authorea [37], ShareLaTeX [38] and others, already had exerted influence on research collaboration in form of co-authoring.

Nowadays, we can see downgrade of document semantic markup complexity: SGML, LaTeX, XML, wiki, markdown. For example, GitHub [39] uses variant of markdown syntax in descriptions (e.g. readme.md) and comments. At the same time, one can take a notice that there are mature "babel-type" projects (like PanDoc [40]) for converting various formats into each other with ease.

Therefore, there is a convergence between technologies for the usability of end-user solutions. One of the most exciting examples – Microsoft OneNote [41], very addictive, cross platform, integrated with other Office and OneDrive solutions. Other cases are services like GitBook [42], which uses CVS Git [43], wiki or markdown as a simple semantic markup language and a bunch of additional tools for simple but powerful collaborative creating of books.

VI. Experience of Building Modern Software Engineering Educational Collaborative Platforms

We dedicate this section to practical requirements and recommendations for building SEE collaborative platform according to current state of SE Curricula, Software Engineering Education Knowledge (SEEK) and SWEBOK Knowledge areas with complementary CS Curricula.

For introductory SE courses we recommend using platforms consisted of:

- Distributed source code version control system.

- Automatic program testing environment.
- Simple web-hosted configuration management system with advanced communication services and the semantic markup of messages.
- Advanced grading tools and assessment tools for educational testing.

Senior courses should be accompanied with real ALM system. In the instructional process such a system can have reduced functionality and can be integrated with LMS for automatic assessment.

There is a big difference between open source and proprietary software in construction of instructional process. In the report, we will discuss pros and cons of several platforms. Illustrative examples of approved platforms are the following.

1. Open source solution: Edgewall Software TRAC [44] with plugins + Git [43] / Mercurial [45] + any programming IDE (for example, Eclipse, but other good choice is educational edition of JetBrains IDE like IntelliJ IDEA or PyCharm).
2. Proprietary solution: Microsoft Foundation Server [46] + Microsoft SharePoint [47] with additional Web Parts + Microsoft Visual Studio [48].

In the trend of "IDEs Shifting to the Web" Microsoft has been recently announced Visual Studio Online [49] (VS Online) as "… a set of cloud-powered collaboration tools that work with your existing IDE or editor, so your team can work effectively on software projects of all shapes and sizes." However, the great advantage from the student point of view is ability to use VS Online from any tablet without local Visual Studio (Fig. 1). It is also interesting that VS Online can use not only Team Foundation Version Control (TFVC) but also Git for source code management. Fig. 2 illustrates extra collaborative components of VS Online.

The mentioned earlier independent project Trinket [35] is the latest attempt of creation a handy open CSCL environment with straightforward ideology. This project has been achieved a new level of educational tools integration in the field of formal language learning. Such a promotion is explained by the effect of combining Open Education with fundamental engineering principal of reusing artefacts and components. As far as we can evaluate, these principals are implemented by using of mature versioning technology transparently to end user. We note, that the project is suitable not only for programming languages (Fig. 3), but for musical staves notation of simple music compositions and the others.

The main problem of such tools in the context of SEE is obvious barriers to the integration with professional SE tools and standards. This is a challenge of modern SEE to bridge the gap between common CSE competences and specific SEE competences.

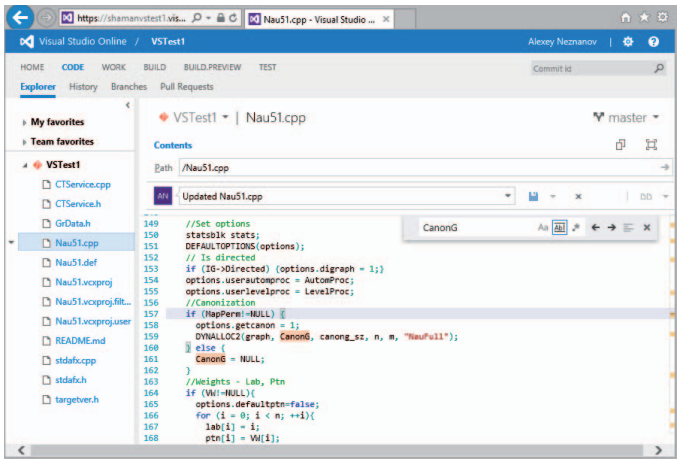

Fig. 1. Microsoft Visual Studio Online Code Editor

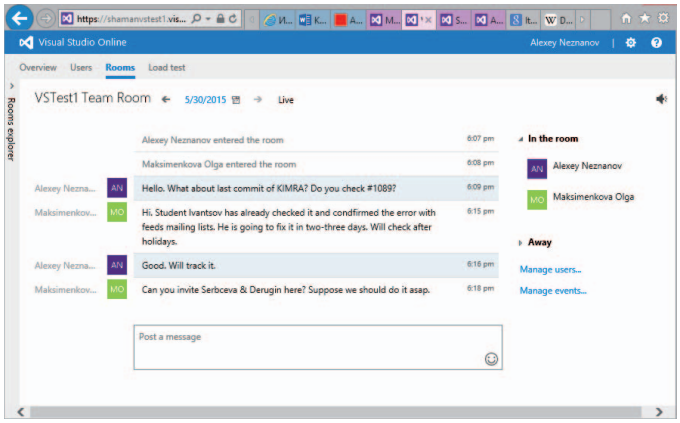

Fig. 2. Discussions in Microsoft Visual Studio Online, structured by Rooms and Topics

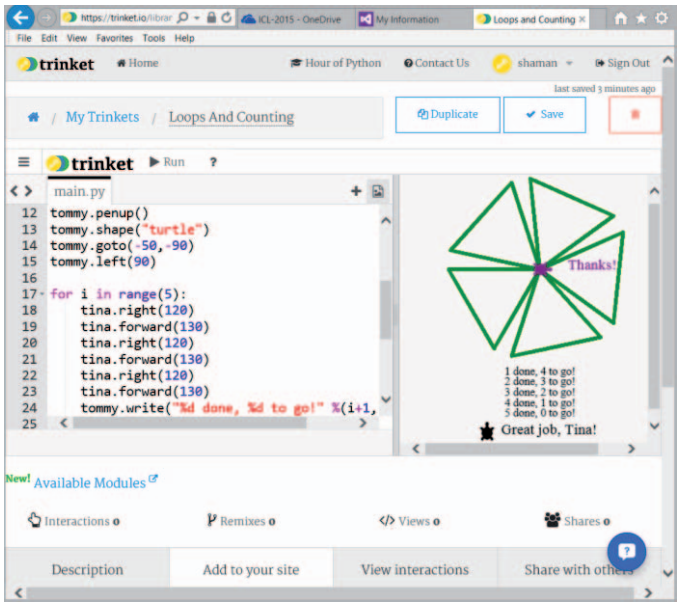

Fig. 3. One of the "Trinkets": Loops and Counting interactive exercise

VII. EDUCATIONAL CASES AND SOLUTIONS

Whereas the SEE processes, which may be supported by collaborative systems, may run on the different stages of education, we should say about significant difference between students' prerequisites, tasks and supportive information. Actually, a typical first-year bachelor student knows less about SE processes then a final-year student or a master-student who can even has a considerable industrial experience. Therefore, tutorials, guidelines, and links' collections should be planned according to students' level and contain extra information about studying tool and methods.

Let us give examples. These examples introduce a number of cases for the first-year SE bachelor students. Reviewing course is "Programming", studying language C#, the tool is Microsoft Visual Studio 20xx. Students' population mostly contains high schools graduates who are familiar with basic algorithms and their implementation on such programming language as Standard C, Pascal or Logo.

The first case is "Supporting students' work under embracing Visual Studio Tools". The accidental and essential complexity [10] of tools brings extra difficulty into education of program construction. Beginners deal not only with language concepts, data structures and algorithms, but with IDE and software construction as a part of ALM as well.

Our solution for this case was to make educational process maximum agreeable with 4C-ID [50, 51] and to extend it with formative assessment. Supportive information in our case was presented by the links' collections and the extra presentations, which were distributed by means of current LMS of NRU HSE. The links' collection of every topic consists of a list of URLs targeted to open tutorials, standards and articles. Presentations concerned on particular cases such as editor settings, code decoration, and debugging, using code snippets (Fig. 4), data structures (Fig. 5), etc. Formative assessment was carried out as brief (10-15 minutes) individual problem solving. The main purpose of this type of assessment was to coerce students to start transforming their knowledge to skills by using them on practice and to provide them with meaningful feedback (see, Example 1).

Example 1. Brief problem solving example

Task example. Declare the **Seq1222** class. The **_seq1222** field of the class is a reference to an array of real numbers. Class constructor with an integer parameter **N** (> 0) initializes the **_seq1222** field with a reference to an array with **N** items. Array items are the following sequence elements: (**1**, **1/3**, **1/5**, **1/7**, …).

In the main program, create the **Seq1222** instance with the sequence of **8** items. Save this instance to a local file using binary serialization. Afterwards give a code of deserialization data from the file to a new instance. Output the result of deserialization to the console (precise of real numbers: three decimal digits).

Feedback 1. (-) Code documentation is absent. (-) The **for** loop in **main()** is infinite. (-) Task does not require getting the number of elements from a user in **main()**.

Feedback 2. (-) Code documentation is absent (+) Iterator is implemented well.

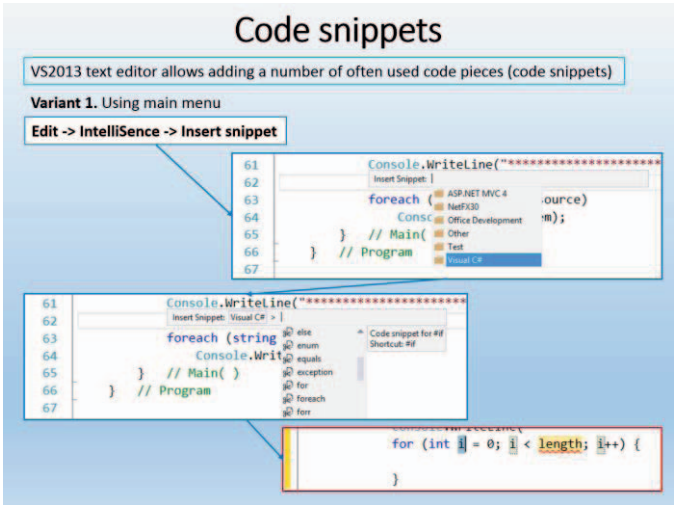

Fig. 4. "Code snippets" slide from supporting presentation

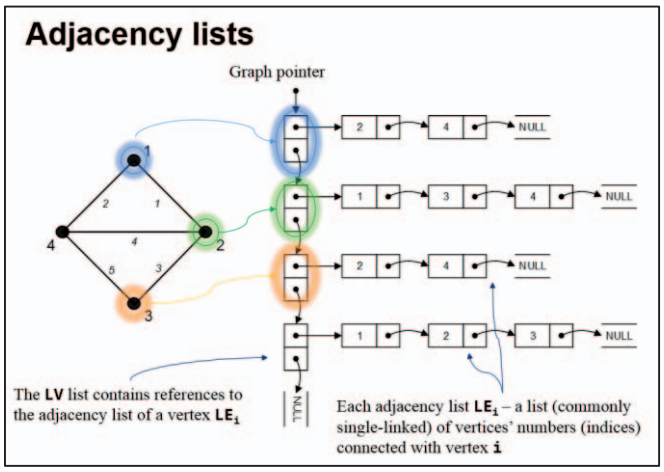

Fig. 5. Graph representation in memory by linked lists, supportive slide

The second case is "Associating VS Project with GitHub". Certainly, qualified reader has already known that Visual Studio 2013 enables developers to manage their code in Git repositories through the IDE. Let us reiterate that CVSs lay on the dividing line between code construction and the other processes of application lifecycle.

Unfortunately, the majority of first-year SE students have not a priory met with professional team development systems and CVSs. Thus, in spite the fact that it is easy to connect GitHub through IDE, we have prepared a brief list of URLs to open tutorials and guides. It touches as GitHub + VS2013 using as related topics like Git version of markdown. This list has been used during working under the Course project and have been delivered through the teachers' blog.

VIII. DISCUSSION AND CONCLUSION

The main goal of this paper was to fill the gap between collaborative technologies used in software industry and CSCL tools. The reviewing of relevant literature demonstrates a great

interest as to CSCL techniques as to collaborative technologies and their application in teaching and education.

Another one more private, but remarkable effect of using such collaboration tools as CVSs (e.g. GitHub) for educational needs is the intergenerational continuity of students' projects. Teacher could plan prolonged projects, which may be maintained as by exactly the same students, as by new students.

In particular, we have several conclusions:

1. Educational collaborative software for SEE should meet requirements of the configuration management systems and ALM methodology. SWEBoK has been established a common framework for such SE requirements and provide an opportunity to use ontology modelling [52] in educational tasks.
2. There is a gap between needs of SEE and, for example, common research activities (e.g. Authorea, Zotero, GitBook, PanDoc). It is seems to be strange because there are the plenty of mature solutions with different forms of collaboration between deferent roles in software life cycle management.
3. Communication subsystems (e.g., chats, forums, blogs) of SEE framework should support all mentioned SE artefact types with adequate decoration and have to provide formative feedback abilities.
4. Different stages of SEE require different abilities which can be achieved in the context of some collaborative platform.

Summarizing given in the paper cases we should say that including professional (collaborative or not) software into educational process leads to increasing the number of using learning technologies. Fortunately, their careful application approximates us to declared outcome, giving real-life experience during the instruction. In other words, educational technologies enriched with analyzed methods and tools for the needs of SEE may have a great impact on bachelors and masters SE programs. It would be interesting to assess the proportion of this impaction. This seems to be a rich direction to the further work.

We also believe that the results of our study as useful for the SEE design as for educational software construction.

ACKNOWLEDGMENT

Authors are very grateful to prof. V.V. Podbelskiy for fruitful discussions and sharing more than 30-year experience of working with software construction tools in education.

REFERENCES

[1] IEEE Computer Society, Guide to the Software Engineering Body of Knowledge, Version 3.0, P. Bourque and R. Fairley, Eds., 2014.

[2] ACM/IEEE-CS Joint Task Force on Computing Curricula, "Software Engineering Curricula 2004," 2004.

[3] "Moodle - Open-source learning platform | Moodle.org," [Online]. Available: http://moodle.org. [Accessed 01 05 2015].

[4] "Enterprise Learning Management System Software - eFront LMS," [Online]. Available: http://www.efrontlearning.net. [Accessed 01 05 2015].

[5] B. Dodge, "WebQuest.Org," [Online]. Available: http://webquest.org. [Accessed 19 04 2015].

[6] A. Bills, "The curve of learning," in General experimental psychology, New York, Longmans, Green and Co, 1934, pp. 192-215.

[7] G. Bafoutsou and G. Mentzas, "Review and functional classification of collaborative systems," International Journal of Information Management, vol. 22, pp. 281-305, 2002.

[8] "Microsoft Office Online - Word, Excel, and PowerPoint on the web," [Online]. Available: https://office.live.com/start/default.aspx. [Accessed 01 05 2015].

[9] K. Tong, "Skype is Stepping Up Productive Communication," 23 12 2014. [Online]. Available: http://blogs.skype.com/2014/12/23/skype-is-stepping-up-productive-communication-2/. [Accessed 01 05 2015].

[10] F. Brooks, The mythical man-month: essays on software engineering, Addison-Wesley, 1975, p. 195.

[11] M. Jackson, Problem frames: analyzing and structuring software development problems, Addison-Wesley, 2001, p. 390.

[12] T. DeMarco and T. Lister, Peopleware: Productive Projects and Teams, 3 ed., Addison-Wesley, p. 272.

[13] N. Ruparelia, "Software development lifecycle models," vol. 35, no. 3, pp. 8-13, 05 2011.

[14] G. Hislop, M. Ardis, D. S. M. Budgen, J. Offutt and W. Visser, "44th ACM Technical symposium on computer science education (SIGCSE'13)," in Revision of the SE2004 curriculum model, 2013.

[15] L. Murta, C. Werner and J. Estublier, "The configuration management role in collaborative software engineering," in Collaborative Software Engineering, Springer, 2010, pp. 179-194.

[16] ACM/IEEE-CS Joint Task Force on Computin Curricula, "Computer Science Curricula 2013," ACM Press and IEEE Computer Society Press, 2013.

[17] J. Spolsky, "Capstone projects and time management," 2009. [Online]. Available: http://joelonsoftware.com/items/2009/10/26.html. [Accessed 01 05 2015].

[18] G. Stahl, S. Ludvigsen, N. Law and U. Cress, "CSCL artifacts," Intern. J. Comput.-Support. Collab. Learn, vol. 9, pp. 237-245, 2014.

[19] D. Engelbart, "Proceedings of the 1984 AFIPS Office Automation Conference," in Collaboration Support Provisions in Augment, Los Angeles, 1984.

[20] The Economist, "A third industrial revolution: collaborative manufacturing," 21 04 2014. [Online]. Available: http://www.economist.com/node/21552902. [Accessed 01 05 2015].

[21] S. Goggins, I. Jahnke and V. Wulf, Eds., Computer-Supported Collaborative Learning at the Workplace, New York: Springer Science+Business Media, 2013.

[22] S. Wood, "A new approach to interactive tutorial software for engineering education," Education, IEEE Transactions on, vol. 39, no. 3, pp. 399-408, 1996.

[23] A. Forte, "The new information literate: Open collaboration and information production in schools," Intern. J. Comput.-Support. Collab. Learn, vol. 10, pp. 35-51, 2014.

[24] D. Harlow and A. Leak, "Mapping students' ideas to understand learning in a collaborative programming environment," Computer Science Education, vol. 24, no. 2-3, pp. 229-247, 2014.

[25] L. Jonhua, Y. Zhang, J. Ruths, D. Moreno, D. Jensen and K. Wood, "120th ASEE annual conference & exposition," in Innovations in software engineering education: An experimental study of integrating active learning and design-based learning, 2013.

[26] W. Jang and H. Lim, "Integration of enterprise resource planning systems into a production and operations analysis course," Int.J. Engng Ed., vol. 20, no. 6, pp. 1065-1073, 2004.

[27] T. Bati, H. Gelderbrom and J. van Biljon, "A blended learning approach for teaching computer programming: design for large classes in Sub-Saharah Africa," Computer Science Education, vol. 24, no. 1, pp. 71-99, 2014.

[28] S.-W. Hwang, "ITiCSE'09," in Blended learning for teaching operating systems with Windows, Paris, 2009.

[29] H. Sondergaard and R. Mulder, "Collaborative learning through formative peer review: pedagogy, programs and potential," Computer Science Education, vol. 22, no. 4, pp. 343-367, 2012.

[30] V. Garousi, "Applying peer reviews in software engineering education: an experiment and lessons learned," IEEE Transactions on Education, vol. 53, no. 2, pp. 182-193, 2010.

[31] D. Rajapakse, "Peer feedback in software engineering cources," in Overcoming challenges in software engineering education: delivering non-technical knowledge and skills, IGI Global, 2014, pp. 111-121.

[32] Microsoft, "OneNote for Teachers - Interactive Guides," [Online]. Available: http://www.onenoteforteachers.com. [Accessed 01 05 2015].

[33] A. Echeverria, C. Garcia-Campo, M. Nussbaum, F. Gil, M. Villalta, M. Amestica and S. Echeverria, "A framework for the design and integration of collaborative classroom games," Computers & Education, vol. 57, pp. 1127-1136, 2011.

[34] "Coursefork," CourseFork team, [Online]. Available: http://coursefork.org. [Accessed 25 05 2015].

[35] "Trinket. The key to open teaching," [Online]. Available: http://blog.trinket.io/a-new-brand-for-open-teaching/. [Accessed 25 05 2015].

[36] I. Alsmadi, "Contributions to ICL2010," in An Ontological Teaching Of Software Engineering; Separation of Concerns, 2010.

[37] "Write research documents online, together. | Authorea," [Online]. Available: http://www.authorea.com. [Accessed 01 05 2015].

[38] H. Oswald, J. Allen and B. Gough, "ShareLaTeX, the Online LaTeX Editor," [Online]. Available: https://www.sharelatex.com. [Accessed 01 05 2015].

[39] "GitHub · Build software better, together.," [Online]. Available: http://github.com). [Accessed 01 05 2015].

[40] "Pandoc - a universal document converter," [Online]. Available: http://pandoc.org. [Accessed 01 05 2015].

[41] T. Maurer, "This is why OneNote is awesome," 03 02 2014. [Online]. Available: http://www.thomasmaurer.ch/2014/02/this-is-why-onenote-is-awesome. [Accessed 01 05 2015].

[42] "GitBook · Write & Publish Books," [Online]. Available: http://www.gitbook.com. [Accessed 01 05 2015].

[43] "Git," [Online]. Available: http://git-scm.com. [Accessed 01 05 2015].

[44] Edgewall Software, "The Trac Project," [Online]. Available: http://trac.edgewall.org. [Accessed 01 05 2015].

[45] "Mercurial SCM," [Online]. Available: https://mercurial.selenic.com. [Accessed 01 05 2015].

[46] "Team Foundation Server Overview| Visual Studio," [Online]. Available: https://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx. [Accessed 01 05 2015].

[47] Microsoft, "What is SharePoint 2013 – Overview of Features," [Online]. Available: https://products.office.com/en-us/SharePoint/sharepoint-2013-overview-collaboration-software-features. [Accessed 01 05 2015].

[48] Microsoft, "Application Lifecycle Management | Visual Studio," [Online]. Available: http://www.visualstudio.com/explore/app-lifecycle-management-vs. [Accessed 01 05 2015].

[49] Microsoft, "What is Visual Studio Online?," [Online]. Available: http://www.visualstudio.com/en-us/products/what-is-visual-studio-online-vs.aspx. [Accessed 01 05 2015].

[50] J. van Merrienbour, Traning complex cognitive skills: a four-component instructional design model for technical training, Englewood Cliffs, NJ: Educational technology publications, 1997.

[51] J. van Merienboer, R. Clark and M. de Croock, "Blueprints for complex learning: The 4C/ID-Model," Educational technology research and development, vol. 3, no. 50, pp. 39-64, 2002.

[52] E. Mena, M. Pareja, M. Sicilia and E. García-barriocanal, "2nd International Workshop on Ontology, Conceptualization and Epistemology for Software and System Engineering," in Deriving competencies from the SWEBOK ontology, Milan, 2007.