Gaining Hands-on Experience via Collaborative Learning: Interactive Computer Science Courses

http://dx.doi.org/10.3991/ijet.v10i3.4510

A. Danielewicz-Betz and T. Kawaguchi
University of Aizu, Aizuwakamatsu, Japan

Abstract—**In this paper we report on the practical outcomes of Software Studio (SS) undergraduate course, but also on a graduate Software Engineering for Internet Applications (SEIA) course, both of which are taught collaboratively by IT and non-IT faculty members. In the latter, students are assigned to projects proposed by actual customers and work together in teams to deliver quality results under time and resource constraints. We are interested in the learning results, such as skills acquired, e.g. by analysing the interaction between students and customers to determine how and to what degree the students transform through project based collaborative learning. As for the SEIA course, the primary goal is to allow students to manage a relatively large number of tools with little prior knowledge and having to work out how to obtain detailed information about given features, when required. In other words, students have to understand the key ideas of web application development in order to be able not only to apply technical knowledge, but also to successfully interact with all the stakeholders involved. In the process, we look for the added value of collaborative teaching, aiming at equipping the participants with both technical and non-technical skills required for their prospective jobs.**

Index Terms—**collaborative teaching and learning, social learning, software engineering, soft skills, Transactional Analysis**

I. INTRODUCTION

This paper is concerned with the practical outcomes of two courses taught in a Computer Science and Engineering university in Japan. We focus on Software Studio (SS) - an undergraduate course, taught collaboratively for the last three years, but also discuss some benefits of a graduate course in Software Engineering for Web Applications (SEIA), taught collaboratively for a year so far.

One of the reasons for collaborative teaching in our case is adding the communicative aspect – the interaction and interconnectivity between the stakeholders involved and the sociocultural context in which they act and interact sharing experiences (cf. Vygotsky's focus, as described by [1]).

Generally, learning becomes a reciprocal experience for the students and teachers and when courses are taught over a number of years, they evolve, transform with the differing students and other stakeholders, as well as with the growing experience gained by the instructors. Following Vygotsky [2], we assume that since knowledge construction takes place within social context, participation in a given course involves student-student and 'expert'-student collaboration in solving real-world problems or tasks that are shaped by each individual's culture and relate to each person's language, skills, and experience.

II. METHODOLOGY, THEORETICAL FRAMEWORK, AND RESEARCH QUESTIONS

The pedagogical method applied is that of collaborative teaching/learning, with the main characteristics being a common task or activity, small learning group (5-14 students), co-operative behaviour, interdependence, and individual responsibility and accountability [3]. The instructor team consists of one IT instructor and one non-IT instructor, with the latter responsible for the 'soft' and interactional aspects of the course, including negotiation and change management. We are interested in finding out collaborative learning outcomes and analysing interactions between students and customers, which are then directly reflected in the changing course content and progress reports. For this we carried out pre-and post-course assessment of the SS course participants, applying certain predefined characteristics and drawing on so-called transactional analysis (TA) [4]. In particular, in student assessment, we resorted to the Ego-State Parent-Adult-Child (PAC) model [5] in order to determine how and to what degree the students transform through project based- and collaborative learning, especially when interacting with real customers.

To encourage peer evaluation and team spirit development, we make use of team evaluation forms (work progress, individual member evaluation, etc.) developed by [6] Oakley et al (2004). Drawing on Vygotsky's [7] theory of social learning we also reflect on the challenges related to team building and team work in the Japanese culture, arising mainly due to lack of criticism and confrontation in customer relations or honest evaluation of team co-members, peer-rating, and the like. Vygotsky [7] stated that we learn through interactions with others. In the case of students, learning takes place through interactions with their peers, teachers, and other experts in their field. A teacher, or a coach, but also an external expert (or a customer in our case) plays a role of a learning facilitator, creating the necessary environment for directed and guided interactions to take place. According to Vygotsky [2], social interaction plays a fundamental role in the process of cognitive development. Vygotsky introduced the concept of the More Knowledgeable Other (MKO) as anyone who has a better understanding or a higher ability level than the learner with respect to a particular task, process, or concept (in our case an instructor, coach, customer, or a fellow student). The Zone of Proximal Development (ZPD) is the distance between a student's ability to perform a task under guidance and/or peer collaboration and that student's ability to solve a problem at hand independently. Vygotsky claims that learning occurs in this zone (see Fig. 1 for Vigotsky's model as applied to the SS course under consideration).

The ego-state model applied is based on Berne's [4] Transactional Analysis in psychotherapy and includes three states, namely:

a. **Adult ego state** (A) that refers to one's thinking, feeling and behaving in the here and now mode, appropriately to any stimulus. When one is in the adult ego state, one is in full contact with and is responding to the here and now in an adult way.
b. **Child ego state** (C) that is judged by the way a person responds to the here and now as if it were an event from the past (acting in the same/similar way as when one was a child).
c. **Parent ego state** (P) is demonstrated in situations when one responds as if one were the parent figure rather than reacting directly and accordingly to the situation. One is said to be borrowing one's parents' (old) way of being [4] [5].

We conducted interviews with the two technical course instructors, and carried out post-course student self-evaluations. Our research questions were the following:

d. What are the benefits of collaborative teaching and learning?
e. What are the challenges of multi-directional communication in software development?
f. What are the tangible course outcomes both from the student and instructor perspectives? What skills emerge as crucial for software development?
g. What are the team building and teamwork skills required and acquired?

A. Software Studio course – An overview

In the SS course, students are assigned to one of the several projects proposed by actual customers (e.g. bus companies and other external business organisations) and work together in teams. Each team is guided by a coach (an expert from a given industry) who provides advice during the development phase. Teams report to the instructors every week, hold meetings with the customers as necessary, and report on their progress and results at two (interim and final) presentation meetings. Also, specific documents produced at each step of the development process are submitted. Since each project team works for an actual customer, results must be produced and delivered to tight deadlines. All students in a given team have to be prepared to actively and responsibly participate in their project. The main objective is to understand the challenges related to developing software that satisfies specific functionality and quality. Students gain first hand experience of project management in software development, conducted under certain constraints, such as limited resources (people, time, equipment). Unlike in regular, theoretically oriented courses, student experience the necessity to timely respond to uncertain circumstances, including identifying cross-functional roles and communicating information correctly. Students working in teams have an opportunity to apply knowledge and skills gained also in other courses to solving realistic and practical problems, guided by instructors and coaches. Moreover, they analyse a given problem and corresponding application domains, making sure to understand the requirements and design

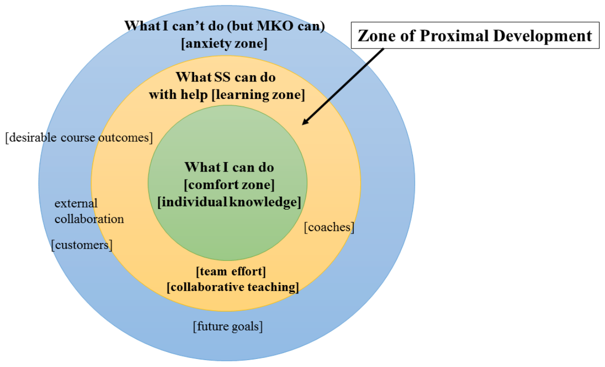


Figure 1. Zone of Proximal Development for students in Software Studio collaborative course

software architecture in order to be able to develop the necessary methods for task/software implementation and to conduct usability test for design optimisation. Each student should also understand the surrounding environment of a given problem, including the network, hardware, and software features. Their activities are then oriented toward gaining project management skills and overall understanding of organisational issues.

As mentioned before, the project work is typically carried out for an external customer or a potential customer. Technical advisors assist the instructors in evaluating student work for content and correctness. Another focus is placed on activating students' multi-directional communication with teachers, fellow students, coaches, and customers, as well as their ability to seek solutions to and solve problems on their own initiative and learn in the process. There are currently two coaches from industry; and their role relates to the customer order and its content and the steps in software development. They are not responsible for the educational aspects and management of students; however, although they do get involved to a certain extent in educational issues, while taking into account the individual characteristics of participating students. The instructors focus primarily on student education and overall project management.

B. Software Engineering for Internet Applications – A course overview

Industry-oriented IT education - especially at the master level – has to ensure that university graduates can assume the role of systems architects and lead system development projects in domestic and international settings. Since software industry is growing very fast, it is practically impossible to study all the tools in detail that are implementable in new software projects. The primary goal is, therefore, to allow students to manage a relatively large number of tools without much prior knowledge and having to work out how to obtain detailed information about software features, as and when required. In other words, students mainly have to understand key ideas in order to be able to apply them in practice.

One of the main tendencies in the software development field is priority given to Web applications. This results in an ever-increasing demand for professionals who can design large Web systems. The aim of this course is, therefore, to study state-of-the-art concepts and methods for Internet application engineering. The course is taught in English (English as the lingua franca as 3 nationalities are involved) by one technical instructor (a computer scientist, head of the software engineering lab), one non-technical instructor (an expert in language, communication, and business discourse who focuses on soft skills, interaction with stakeholders, negotiation process in the context of software engineering, change management, aesthetic design and the user perspective), as well as one substitute technical instructor (with different IT background).

C. Multi-directional interaction: challanges

As mentioned before, the SS course offers the opportunity to interact with real customers (see Fig. 2). The 'contract' with the customer expresses agreement to show understanding for the fact that a given project serves an educational purpose; therefore it is not a genuine project per se (which may not sometimes be entirely comprehended by the customer when placing expectations in the final product). Students interact with their customers in diverse ways, whereby customers differ from year to year and represent different lines of business, which implies differing types of interaction as one of the intentional course aims. The course started with the local businesses (public and private) and is now expanding to include customers from other parts of Japan. As companies favour continual interaction, students have to adjust accordingly. Although they are guided by both instructors in terms of modes of interaction to be used, they are mostly able to find out themselves what method is best for them. Feedback from the customers (by email) comes directly to the mailing list and the interactions are monitored by the instructors. Although face-to-face meetings are preferred and desired, because of time constraints and logistics reasons they may not take place regularly. Usually, local companies visit every week, but if they are based in Tokyo, then regular face-to-face contact becomes difficult. Moreover, the students are not allowed to initiate personal interaction with customers.

In every class students communicate with both instructors and submit weekly progress reports, identify progress made, problems occurring, and ways of approaching problems. The instructors intervene, when necessary, giving instructions on how to proceed, and checking documents prior to passing them on to the customers. This offers yet another chance to guide students through the process, as they often do not know how to produce business documents.

Since students are reluctant to speak about problems or are unable to express what is happening in words and only at the last moment admit that there is something wrong, a close watch with the help of some trustworthy students (or friends of the participating students) is necessary. In this way the instructors can indirectly collect information about the problems encountered. A project management tool is used to share files and course related online resources, with the content monitored by the instructors. It is for the students to use for important project deliverables, not necessarily for the team meetings.

The relationship among 2 instructors (as co-managers of the project), 2 coaches (as technical and business advisers), and students (as software engineers-to-be) is structured in such a way as to put the external customers (such as a bus company or farmers, with no IT specialist on board) and their needs (i.e. also issues and problems) first and to solve them professionally using ICT skills. But in

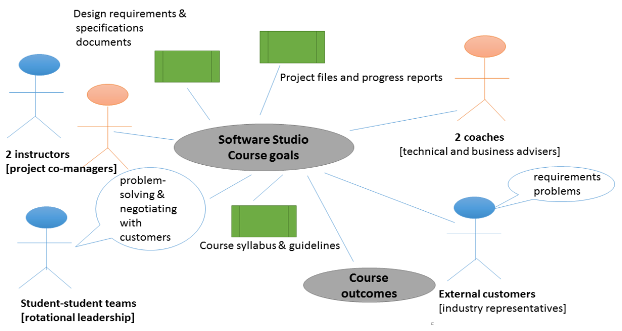

Design requirements & specifications documents
Project files and progress reports
2 instructors [project co-managers]
Software Studio Course goals
2 coaches [technical and business advisers]
requirements problems
problem-solving & negotiating with customers
Course syllabus & guidelines
Course outcomes
Student-student teams [rotational leadership]
External customers [industry representatives]

Figure 2. Multi-directional communication in Software Studio course

order to relay problems in software terms, considerable technical and professional advice is required from the two coaches. From the educational perspective, too much advice, however, might change the direction of software development to be negotiated about with the respective customers. In this process, students are supposed to collaborate to make sure that external requirements/requests are met and problems solved in a desired and, simultaneously, doable way in terms of ICT skills. Sometimes, there is a mismatch between the external customers and the students regarding their understanding of the requirements, which demonstrates that collaborative work requires a high degree of human interaction management, as well as management of project milestones.

D. Advantages of collaborative teaching: SS Course

We have found out that collaborative teaching is mostly beneficial as it provides a wider vision, taking student diversity and the dynamic nature of the course into consideration. Our previous experience in collaborative teaching shows, however, that in some cases it may prove difficult: there may exist different visions since the focus varies. In that sense, it is easier to start and conduct a course individually since once others join in, they have to be guided, adjustments made, etc., which may sometimes negatively affect students and naturally takes time to yield positive results. In the case of the SS course, the added value of non-technical expertise of the co-instructor lies mainly in (inter)personal development, consultation about interaction with customers and customer relations (management aspects), as well as in overall course management support. This results in a more reliable course content, with mutual instructor consultations regarding decisions (and timing of decisions), e.g. about the best ways of explaining certain constraints to customers, monitoring customers, the degree of intervention in students' decisions when, for instance, being aware that they are heading in the wrong direction. The question then is whether to let the students learn from their own mistakes or take over control of the situation by contacting the customer in question on their behalf in order to prevent mistakes or redress certain course of action. Such decisions are best taken collaboratively.

E. Advantages of collaborative teaching: SEIA Course

In this course, both instructors complement each other as they pay attention to different aspects of the course content and learning outcomes. They ask different types of questions, and their comments differ. In this way, various points of view are taken into account, alternative discussion points identified and verbalised. The instructors also employ differing styles of teaching, presenting information, involving students in interaction, and giving feedback. Consequently, students are exposed to alternatives and gain a broader picture of the software engineering field. They also have an opportunity to seek individual consultations and discuss respective assignments with both instructors. In class, technical exercises submitted by students are anonymously discussed, together with the prototypes and solutions suggested. In a follow-up step, students work on improvements that they can again discuss individually with the instructors.

F. Advantages of collaborative teaching: SS course

Fig. 3 shows the overall average of the TA pre- and post-assessments. In this particular class, 14 students were divided into 2 groups, and worked on different projects requested by real customers. The TA surveys were conducted both prior to taking the course and upon its completion to observe the behavioural characteristics of the students related to collaboration awareness. In Fig. 3, the highest average of all characteristics coincides with the adult stage that indicates any reality-bound adult-like behaviour as a response to here and now. At this stage students were considerate and respected others, and tried to gain clarity by such means as open discussions; but, on the other hand, they sometimes tended to use rational debate to achieve the right outcome and hence became critical. By contrast, the lowest average was that of engaging behaviour from childhood that paid no attention to parental rules or limits.

The students tended to express their ideas and thoughts based on their experience and knowledge gained and then made suggestions to the customer. Sometimes, however, the ideas did not match what the customer expected since the students tried to put forward the best solution or idea from their own perspective or understanding of what the customer explained to them. Prior to learning how to negotiate with the customer, the students tended to be very agreeable, but as Fig. 3 indicates, they would change over to the adult stage once comfortable at seeking clarification and suggest preferable outcome.

Fig. 4 depicts the comparison between Controlling Parent (CP) and Nurturing Parent (NP) behaviours. In this case, 13 students' characteristics (from the pre- and post assessments) show opposite results. It appears that the team course assignments made students realise the value of collaboration and looking after each other's interests as based on a genuine regard for diversity in ability and skill. Initially, they would rather passively imitate others or copy from the previous year's student assignments than show any initiative. In fact, every student demonstrated different abilities and ways of reaching out to collaborate with others, believing that their approach was best to complete the task at hand. Sometimes, the customers would not know precisely what they were looking for and that caused communication problems with the students, making some of them lose interest in the project, accompanied by time constraints and necessity to demand action in order to complete a given task. The post-assessment shows that it is easier for the students to follow what others have agreed on, and that they do not realise that demanding and caring are perceived differently. Yet, they learn balancing communication with customers and teammates through this activity.

As presented in Fig. 5, when we focus on both FC and AC, the pre- and post-assessment results do not exhibit

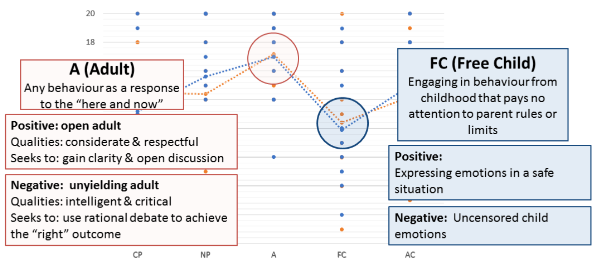


Figure 3. Transactional Analysis: pre- and post-assessment of student team collaboration (A) vs. (FC)

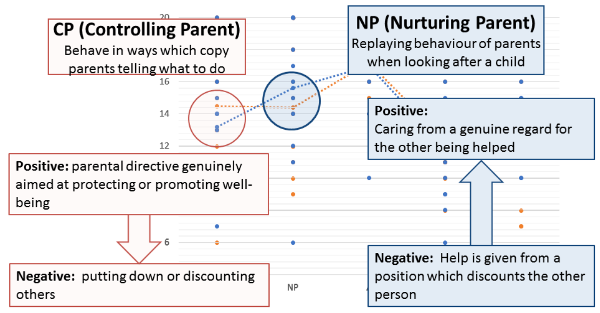


Figure 4. TA pre- and post-assessment: student perceptions in parental state

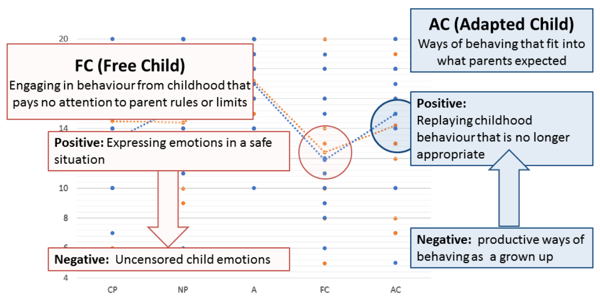


Figure 5. TA pre- and post-assessment: student perceptions in child state

any big differences, apart from AC behaving in such a way as to meet other students' expectations. This demonstrates that in order to work collaboratively, students tend to care for others. In the course of 3 months, students' collaborative behaviour makes them see others in a more professional way. The biggest challenges are peer interaction (due to differences in knowledge and skills), negotiating, and solving problems encountered in working with the real customers (who also look for solutions, but from their (non-IT) perspective). Students also have to work together outside of class and they also use software-mediated communication to share each other's progress results. It seems easier for them to express themselves in writing, but contradiction may occasionally occur when talking face to face. As we mentioned above, the additional challenge is posed by the fact that the students attending the SS course exhibit differences in terms of knowledge and skills, with some completing all the tasks without any problems and not realising that high demands are placed on the others. Gradually, they learn how to collaborate as a team in such a setting.

G. Student self-evaluaton (skills acquired in SS)

The final self-reflection took place after handing over the completed software product to the customer. On the whole, students claimed that they improved their overall project management, communication, presentation, writing, business, and software development skills (for the self-evaluation breakdown, see Fig. 6).

Specifically, the following skills were self-reported as improved or newly acquired:

h. **Project Management skills:** delegating tasks to others according to their differing knowledge/skills, time management, overview of the project and its tasks, communicating with other project members, understanding progress, team management;

i. **Communication skills:** informing others politely, conveying a message in a positive way, communicating intention (pragmatic rules of communication), expressing an honest opinion and sharing opinions, understanding others, dealing with various types of people (with differing personal characteristics), listening, logical thinking, explaining ambiguity, virtual communication;

j. **Presentation skills:** presenting facts (previously very difficult), representing information visually, giving an overview of the project, presenting ad hoc (no 'speech script'), explaining complex technical content in an easy to understand way, summarising information - preserving the intended message, detecting what the other party wishes to know and meeting their expectations, explaining in a logical order;

k. **Written skills:** improving accuracy of expression, summarising, creating presentation materials, clarifying ambiguity and reducing the possibility of misunderstanding, writing in non-technical language and in an easy to comprehend way, establishing sequential order of the written material, presenting facts in writing (difficult in the beginning – both in oral and written form);

l. **Business skills:** familiarity with the ChatWork task management tool, creating reports, business interaction, consulting skills, facilitating a meeting based on an agenda, managing meetings, note and minute taking, precise information retrieval and regular updates, assessing the weight of oral versus written information, chairing a meeting efficiently, reporting on tasks to team members;

m. **Software development skills:** understanding the processes and programming aspects of software development, familiarity with software development tools, e.g. Super Agile Struts (SAStruts), API mapping; reading programs written by others and understanding their structure, understanding customer needs (request proposal), program testing, web design, HTML, web system development tools – e.g. Seasar2, Ajax library, Java script, mathematical data analysis, and development of written documents.

H. Teamwork (instructor perspective)

The primary SS course objective is not team building per se, but understanding software development; teamwork is a tool not an objective in this case and functions as a part of project management in software engineering. But as teamwork is required, students naturally build teams, more or less successfully, but some 'teams' do fail at

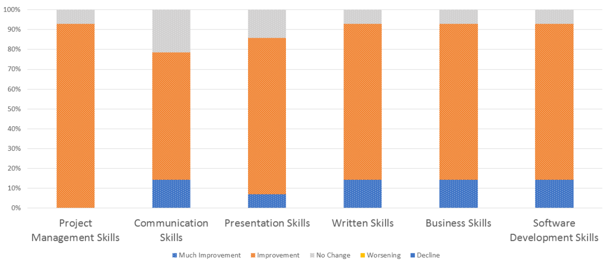


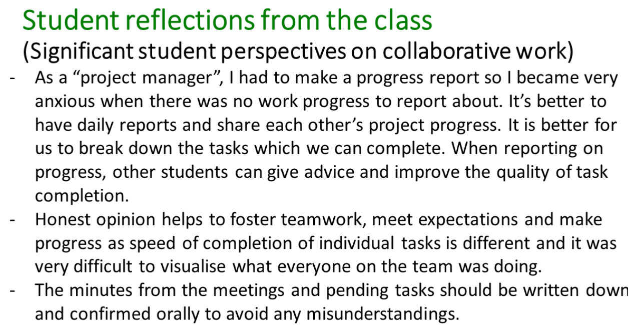
Figure 6. Student self-evaluation: final report

teambuilding and teamwork. Nevertheless, they learn some valuable lessons in the process and those students who try hard acquire some teamwork related skills (see student self-reflections in Fig. 7).

The system of rotational leadership is applied so that the learning process is, to a certain degree, equal for everyone involved. Too frequent a change in team leadership is not desirable, however, as this may confuse customers. Therefore, as a compromise, not all but a few (usually most motivated) students become leaders. On the other hand, a given team evaluation does not always depend on the leader (the leader may in fact contribute the least). Students are taught team membership, assume different roles, and learn how to contribute to the team. The drawbacks include uneven distribution of tasks and disproportional contributions (with freeloaders taking credits for others' work), which makes individual assessment rather difficult. In addition, since once students perceive difficulty and tension in the team they may stop communicating with others, it is the instructors' responsibility to point out individual capabilities to the team members and to encourage development of different points of view in students.

I. Teamwork (student perspective)

The SS course participants' reflections regarding collaboration in teams are summarised in Fig. 7. Students have to report their progress to the customers, as well as their couches and instructors every week. At this point team leaders have to ensure that the final summary of each meeting be confirmed by all the team members, eliminating any gaps and/or preventing loss of overview regarding team activities. Such realisation of mutual collaboration is a very significant tool for gaining leadership skills.


Student reflections from the class

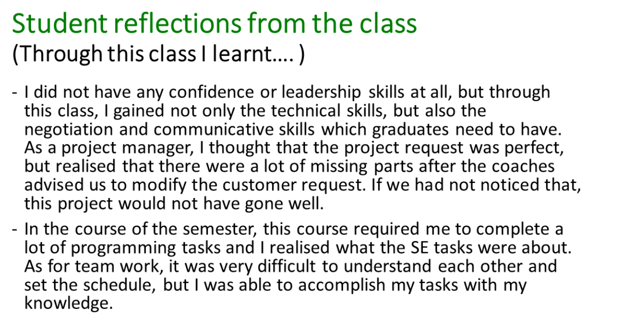
(Significant student perspectives on collaborative work)

- As a "project manager", I had to make a progress report so I became very anxious when there was no work progress to report about. It's better to have daily reports and share each other's project progress. It is better for us to break down the tasks which we can complete. When reporting on progress, other students can give advice and improve the quality of task completion.
- Honest opinion helps to foster teamwork, meet expectations and make progress as speed of completion of individual tasks is different and it was very difficult to visualise what everyone on the team was doing.
- The minutes from the meetings and pending tasks should be written down and confirmed orally to avoid any misunderstandings.

Figure 7. Student post-course self-reflection: collaboration

J. SS course benefits

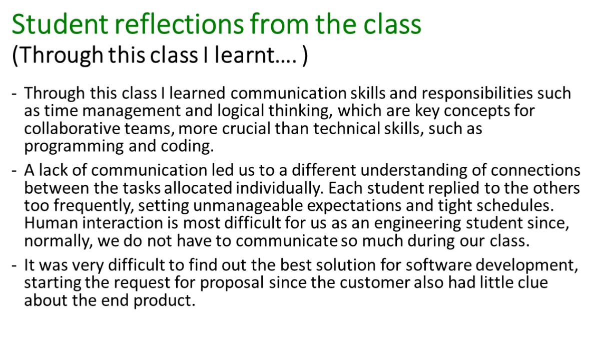
As discussed above, students submit reports on what they have done as well as self-evaluation reports ("self-reflections") on personal development, that is the acquired business skills, communicative skills, teamwork, management skills. Almost all participating students say that they have gained valuable skills, with some, more open, admitting that although the course is difficult, they have gained a great deal from it and, overall, it has been a rewarding experience (see Fig. 8 and Fig. 9).


Student reflections from the class
(Through this class I learnt....)

- I did not have any confidence or leadership skills at all, but through this class, I gained not only the technical skills, but also the negotiation and communicative skills which graduates need to have. As a project manager, I thought that the project request was perfect, but realised that there were a lot of missing parts after the coaches advised us to modify the customer request. If we had not noticed that, this project would not have gone well.
- In the course of the semester, this course required me to complete a lot of programming tasks and I realised what the SE tasks were about. As for team work, it was very difficult to understand each other and set the schedule, but I was able to accomplish my tasks with my knowledge.

Figure 8. Student post-course self-reflection: learning outcomes (1)

The SS course is embedded in the software engineering track, so it is mandatory to attend. Yet, some students fail to complete it. Naturally, it is easier to conduct the course when only the motivated students attend it. But, on the other hand, working with less motivated students makes the situation much more real. This is a unique realistic environment, comparable to a workplace, where one may not be inclined to work with certain people but has to do so. The course is taught in Japanese and the customers are Japanese. Although student are taught in their native language, they still need to learn to express themselves, identify and articulate problems (in a timely manner), and then – in graduate school – they move on to multicultural working environment. In this sense, the course offers a good preparation for graduate courses. Furthermore, we have observed that students are capable of answering questions when asked directly. Internal understanding of the processes involved in software development is crucial at this stage, which they seem to acquire. Additionally, students may use the software development skills gained in their part-time jobs in software companies, applying not only their programming skills, but also communication skills.


Student reflections from the class
(Through this class I learnt....)

- Through this class I learned communication skills and responsibilities such as time management and logical thinking, which are key concepts for collaborative teams, more crucial than technical skills, such as programming and coding.
- A lack of communication led us to a different understanding of connections between the tasks allocated individually. Each student replied to the others too frequently, setting unmanageable expectations and tight schedules. Human interaction is most difficult for us as an engineering student since, normally, we do not have to communicate so much during our class.
- It was very difficult to find out the best solution for software development, starting the request for proposal since the customer also had little clue about the end product.

Figure 9. Student post-course self-reflection: learning outcomes (2)

K. SEIA course benefits

Students in the Software Engineering for Internet Applications course become familiar with the soft (non-technical) factors affecting the performance of software development teams, which include team climate, team diversity, team innovation, team member competencies and characteristics, top management support and team leader behaviour, which all have a considerable effect on software development team performance, also in pair programming. Mutual trust and communication effectiveness are found to be the prioritised factors affecting software development teams' performance.

In this course, work in teams has been, so far, only theoretically discussed and based on case examples rather than hands-on experience (e.g. the concept of "two-pizza" teams in SaaS – software as service – and Scrum). This is due to the small number of participants and the short duration of the course (10 weeks), which makes team building a rather futile task.

One point made clear is that desirable software capabilities cannot be achieved without developers demonstrating those characteristics themselves. These include, among others, availability, correctness, reliability, integrity, efficiency, user-orientation, flexibility, and creativity [8], [9]. Students draw parallels between desirable software features and the skills required to create such software. They learn that communication in software engineering is multi-fold and complex, as many stakeholders are involved [10] (cf. Fig. 10).

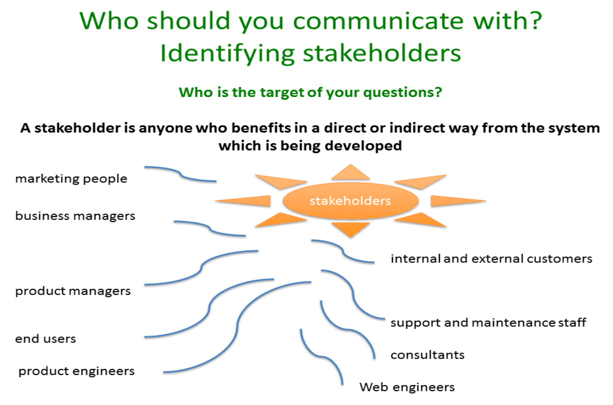

Who should you communicate with?
Identifying stakeholders
Who is the target of your questions?
A stakeholder is anyone who benefits in a direct or indirect way from the system which is being developed
stakeholders
marketing people
business managers
internal and external customers
product managers
support and maintenance staff
end users
consultants
product engineers
Web engineers

Figure 10. Stakeholders in software development

Change management as "the only constant in software development" is also discussed. It involves planned software requirements, bug fixes found during testing, correcting unanticipated consequences, and fixing customer complaints. Within IT, change control is a component of change management. The change control process is usually conducted as a sequence of steps proceeding from the submission of a change request. Typical IT change requests include addition of features to software applications, installation of patches and upgrades to network equipment.

In both courses students are exposed to negotiation (in SS hands-on; in SEIA more theoretically), which is a key skill in software engineering. Typically during negotiation, the software engineer reconciles conflicts between what the customer wants and what can be achieved given limited business resources. Requirements are ranked (i.e. prioritised) by customers, users, and other stakeholders. Risks associated with each requirement are identified and analysed. Rough guesses of development effort are made and used to assess the impact of each requirement on project cost and delivery time. Using an iterative approach, requirements are then eliminated, combined and/or modified so that each party achieves some measure of satisfaction.

A mild critical feedback is given in class without pointing to a particular participant (thus taking the Japanese cultural factors into consideration). Students are also praised whenever possible to facilitate motivation (e.g. "You've done your best", "Great").

The technical instructor guides students toward optimised solutions from their points of view rather than imposing his own idea (e.g. regarding design). The typical questions include: "How would you improve your own solution?", "What were the points of difficulty?", "How long did a given task take to complete?", "What was interesting/difficult about working with the tool?", or "What about applicability to multi-screens?". The key assessment of students' performance is carried out during the final project presentations (the task being creation of the Software Engineering Lab website). This is done individually; there is no teamwork involved, as previously mentioned. Each presentation is followed by a Q&A session.

III. CONCLUSION

Based on our investigative study and first-hand experience teaching the courses in question as non-technical instructors, we have observed that course participants clearly profit from collaborative learning: they become more confident and willing to communicate with others in complex business multi-channel situations (which otherwise pose a definite challenge to Japanese CS students), and may eventually reach the stage where they actually work productively as a team. Collaborative teaching makes it possible for students to gain a more complete picture of multi-fold characteristics of their scope of responsibilities as future software developers and the expectations placed on them by various stakeholders. The instructors also profit in the process, for instance from sharing of ideas and materials, exposure to differing teaching styles, and peer feedback. We have also observed that technically minded students tend to have difficulty comprehending and identifying the soft aspects of software development and the non-technical requirements of software engineering-related job positions, not to speak of fostering those skills in themselves. The same applies to the aesthetic aspects of software design. This leads us to conclude that such factors should also be incorporated in other engineering course syllabi and dealt with in detail.

ACKNOWLEDGMENT

We wish to thank Dr Rentaro Yoshioka for his valuable insights about the Software Studio course and for sharing student self-evaluation forms with us.

REFERENCES

[1] K. Crawford, "Vygotskian approaches in human development in the information era," *Educational Stud. in Math.*, Vol. 31, pp. 43-62, 1996. http://dx.doi.org/10.1007/BF00143926

[2] L.S. Vygotsky, Mind in Society: The development of higher psychological processes. Cambridge, MA: Harvard University Press, 1978.

[3] N. LeJeune, "Critical components for successful collaborative learning in CS1," *J. of Computing and Sci. in Colleges*, Vol. 19, pp. 275-285, 2003.

[4] E. Berne, Transactional Analysis in Psychotherapy. Grove Press, Inc., New York, 1961.

[5] R. G. Erskine, "Ego structure, intrapsychic function, and defense mechanisms: A commentary on Eric Berne's original theoretical concepts", *Trans.Anal. J.,* Vol. 18, pp. 15-19, 1988.

[6] B. Oakley, R. M. Felder, R. Brent and I. Elhajj, "Turning student groups into effective teams", *The J. of Student Centered Learning,* Vol. 2, pp. 9-34, 2004.

[7] L.S. Vygotsky, Thought and Language. Cambridge, MA: MIT Press, 1962. http://dx.doi.org/10.1037/11193-000

[8] A. Fox and D. Patterson, *Engineering Long-Lasting Software: An Agile Approach Using SaaS And Cloud Computing.* San Francisco: Strawberry Canyong LLC, 2012.

[9] R. S. Pressman, and D. B. Lowe, Web engineering: a practitioner's approach. New York: McGraw-Hill Higher Education, 2009.

[10] K. A. Saleh, Software Engineering. Fort Lauderdale, FL: J. Ross Publishing, 2009.

AUTHORS

A. Danielewicz-Betz, PhD, is an associate professor in the Centre for Language Research, University of Aizu (e-mail: abetz@u-aizu.ac.jp). Her research interests include Pragmatics, Critical Discourse Analysis, Interdisciplinary Studies (e.g. Forensic Linguistics, Corporate Discourse), Business Ethics, and Management of Higher Education. Previously, she worked for Prince Sultan University, Riyadh, Saudi Arabia and Ludwig-Maximilian University of Munich, Germany. She also is an experienced business consultant and trainer offering customised in-house business English courses and coaching services for multinational companies.

T. Kawaguchi is an associate professor in the Centre for Strategy of International Programs, University of Aizu (e-mail: kawaguch@u-aizu.ac.jp). He is responsible for international affairs and international student support and services. His research interests focus on designing and integrating the study abroad programs with an aim to develop effective educational activities by adopting an intercultural approach, as well as ensuring successful preparation for professional training and development of global engineers. This includes placing study abroad program administration within the local community and management systems for enhancing international student satisfaction, language assessment, and enrolment management.

This article is an extended and modified version of a paper presented at the International Conference on Interactive Collaborative Learning (ICL2014), held 03-06 December 2014, in Dubai, UAE. Manuscript received 08 March 2015. Published as resubmitted by the authors 11 May 2015.