Learning Object-Orientation through ICT-mediated Apprenticeship

Annita Fjuk
InterMedia, University of Oslo /
Telenor R&D, Norway
annita.fjuk@telenor.com

Ola Berge
InterMedia, University of Oslo /
Telenor R&D, Norway
ola.berge@telenor.com

Jens Bennedsen
it university west, Denmark
jbb@it-vest.dk

Michael E. Caspersen
Department of Computer Science,
University of Aarhus, Denmark
mec@daimi.au.dk

Abstract

This paper gives insights into how socio-cultural theories are applied in a course design directed towards introduction to object-oriented programming. The particular focus is on the apprenticeship between learners (apprentice) and more experienced peers (co-learners and teacher).

1. Introduction

It is widely accepted that activities directed towards object-oriented programming and modeling require different ways of thinking and different approaches to a phenomenon than procedural development activities. In agreement with this line of thought, there is a long tradition in discussing the challenges around teaching and instructional design connected to introductory courses in programming (SIGCSE; OOPSLA, etc). We argue, however, that there has been a lack of explicit foundation in learning theories underlying past research and course design. The field tends to focus on the technology rather than on the learning theories, or didactics of computer science [8]. As a consequence, the important analytical issue of how programming as a knowledge domain is created by the individual as well as what aspects are considered as critical for the individual's understanding, are both missing. Many computer science educators have no formal training in education [8, 3] or do not have capacity to do research on this area besides their own research area in computer science.

This paper is concerned with how socio-cultural theories about learning can inform design of a model-based introductory programming course (Introduction to Obejct-Oriented Programming, IOOP), at Aarhus University, Denmark. In addition to the particular model-based philosophy, the characteristics of the target group have essential impacts on how the learning theory is incorporated into the course design. The target group of the course is adult part-time students, committed to different work organizations, families, geographical places, etc. Given this situation, the learning activities need to be provided in a flexible manner. A conscious combination of new information- and communication technologies (ICTs) facilitate for organizing learning activities across individual constraints such as technical infrastructure, profession and experiences as well as preferences with regard to learning style.

2. Method

The research is based on a case study carried out during the fall semester of 2003 on IOOP. The unit of analysis for the case study is *the learning activity*. This means that the focus of our analysis is on the constellation of learning resources and what effect each of them have on the learning activity. We believe that the relationships between these resources are rather interwoven, implying that it is complicated, sometimes even impossible, to consider which ones that are *most critical* with respect to the learners' understanding of object-oriented programming.

The data gathering has been carried out by observation of the online activities and weekend seminars, and by in-depth interviews with 9 students, the instructor, and the teaching assistant. Finally, this study is supplemented by a survey carried out among the students as a part of the course evaluation.

3. Theoretical foundations

To illustrate the importance of using learning theories in the course design, we will focus on a core set of concepts that we, first, considered as important for understanding the particular case of IOOP and, then, for understanding the role of ICTs herein (e.g., Vygotsky [15, 16], Leont'ev [12] and Davydov, Zinchenco & Talyzina [4])).

The core argument for using socio-cultural theories is the focus on the social and contextual dimension of knowledge construction and the rich approach of understanding the inseparable role of artifacts. The learning theoretical heritage from Vygotsky [15,16] is that knowledge construction is social by its very nature, and that intellectual development takes place on two levels: First it appears on a social level, through interpersonal interaction. Then it appears on an individual level through intrapersonal interactions. These interactions take place through a range of actions that are directed towards conscious objectives. The actions have operational aspects, i.e., the way the action is actually carried out.

The notion of artifact mediation becomes fundamental in this respect. Artifacts are incorporated parts of the actions, they carry with them a particular culture and history, and – as such – influence how human actions are operationalised. Because of this nature of artifacts, they should not be considered as given, but be viewed inseparable from every human activity. Many computer-based artifacts occupy interrelated roles as both means for thought and reflection and as tools for operationalising the same action. A text-based communication system (e.g., chat, e-mail, etc) is one typical example and the object-oriented language Java is another example. Java can be regarded as an artifact for operationalising an object-oriented way of thinking into program code. It may serve as a means for thinking into this perspective on programming, but at the same time providing a communication language for communities of programmers. Moreover, object-oriented programming languages contain some fundamental different principles than e.g., procedural programming languages, implying that a comparable task will be performed completely different by these two types of artifacts.

The socio-cultural groundwork has received vastly differing interpretations, under which knowledge construction (and intrapersonal processes) play different roles. One widely known interpretation is how the student internalizes the scaffolding and guidance of more capable peers. The pedagogical approaches seek to provide instructional support for performance of tasks and are often conceptually tied to the pedagogical intentions of teachers and other caregivers. Without explicitly referring to this theoretical foundation, we argue that the CS1 area is dominated by this instructional view, however, without any clear interpretation of how learning takes place. For example, Bergin [3] has developed pedagogical patterns that are generally aimed at providing – in a uniform way - solutions to common problems in teaching object-oriented programming. These patterns serve as artifacts for mediating the teaching activities, but do not in any strong sense include basic theoretical principles of learning [2]. The work of Kölling and colleagues [10] on the BlueJ environment is another example. The background for developing BlueJ is the challenges around teaching object-oriented programming and, the pedagogy behind BlueJ is reflected upon such a view by describing instructional guidelines [10].

Another and "cultural" interpretation (as Lave & Wenger [11] term it) comes along with Vygotsky's distinction between scientific (specialized language such as e.g., programming languages) and everyday concepts, and on his argument that a mature concept is created when the scientific and everyday versions have merged. Holmboe [8] uses this theoretical foundation when demonstrating how data modeling incorporates both of these concepts simultaneously. Holmboe's study indicates that students tend to confuse natural language and formal elements of the programming language. The challenge of the individual student, Holmboe [8] argues, is to use the language to describe the world in a context-independent and stringent manner so that the computer can understand it.

Recent interpretations of socio-cultural perspectives take collective and societal perspectives rather thoroughly into considerations. The works of Engeström [7] and Lave & Wenger [11] have significant positions in this respect. These interpretations extend the study of learning beyond the context of pedagogical structuring and schooling, and focus on the contradictory nature of social practice. According to Engeström, learning is the mastery of expansion from everyday actions of individual to new activity collectively generated as a solution to so-called double-bind situations. The work of Engeström has influenced a variety of studies within the computer science field.

Lave & Wenger [11] share the focus on social processes of learning with Engeström, but place more emphasis on connecting issues to socio-cultural transformation with the changing relations between newcomers and old-timers in the context of a changing shared practice [11]. With an absence of what we traditionally know as formal teaching in apprenticeship, crucial issues are what promote the learning process, what actions must be focused and how to structure the social interactions. In the context of IOOP, one important learning objective is the *processes of programming*. This means that it is regarded as important that the students gain insights into how programmers develop their solutions from the initial problems, e.g. how one frequently compile code, use documentation and test partial solutions. One way of attaining this goal is to expose the students to how an expert programmer works. Another is to consider the student as an active participant in a community of co-students. Concerning the former, it is close to what Nielsen & Kvale [13] term a *person-centered approach*. The master reflects and thinks aloud of the particular action, making them visible and as a source of identification (Ibid.). As such, the apprentice learns from observing the master (teacher) performing the actions embedded in the profession (e.g. coding, testing, etc). From this particular position, the role of language (oral and written) becomes important. Furthermore, the master's comments to the student's practice have an important position in the student's reflection in action (cf. [18]). Concerning the latter, it is described as a *decentered approach* by Nielsen & Kvale [13]. Knowledge construction is considered as legitimate peripheral participation, i.e., the attention is on the student's inevitably participation in communities of practitioners where the old-timers legitimate the skills and knowledge of the individual newcomer. The student is the apprentice and the teacher (or more capable peer) is the expert in the social interactions. Mastery does not reside on the master, but on the organization of the community (of which the master is a part) and on the structuring of a community's learning resources (ibid.).

The next section is concerned with how the two approaches is combined in the course design, as well as what ICTs and learning resources that were selected to operationalise the actions in the interactional processes.

4. The design

One important aspect in introductory programming courses is the role of the programming language. In IOOP three perspectives are identified [9]: Instructing the computer, managing the program description, and conceptual modeling. A central issue pertaining to the design is the decision to maintain a balanced view on these three aspects in the course design. The primary outcome of this choice of balancing are expected to be that the students: 1) learn a systematic approach to programming; 2) obtain a deeper understanding of the programming process; and 3) focus on general programming concepts instead of language constructs in a particular programming language. The rationale for this choice is described in more detail in [1].

4.1. The person-centered approach

Apprenticeship learning with respect to the three above interrelated aspects of learning object-oriented programming implies a definition of what actions that are needed to be performed by the teacher and the student. This implies not just as in the traditional form where topics are listed, but also in a broader sense including working patterns, traditions and habits. When the teacher's role is to legitimate the skills and knowledge of the student, the teacher needs a fairly deep understanding of the level of skills - otherwise it is very difficult to legitimate anything.

One central artifact is the weekly assignment. An assignment is designed as programming exercises, and is based on the readings and exercises scheduled for that week. The assignment is a means for thinking and for understanding the practice of programmers as well as to engage in the process of creating object-oriented computer programs. Together with Java, the Blue J environment is used by the individual to operationalise these sets of actions. Moreover, the assignment is a fundamental means for interaction between student and teacher, and thus for legitimating the student's actions towards the problem. As such, the apprenticeship approach implies a change *from* viewing the assignments as control / evaluation mechanisms *to* a communicative artifact between the master and the apprentice. The assignments are therefore not part of the final grade but used with the communicative purpose and as a way of structuring the student's time.

In order to enrich the pedagogical philosophy of the assignment, a corresponding weekly online meeting is conducted. The topics treated in these meetings are based on the individual student's experiences in solving the assignment, combined with her/his request posted in an asynchronous discussion forum beforehand. This approach denotes a particular mode of engagement and participant control, at the same time as the teacher legitimates and shows how programming / modeling processes associated with the weekly problem areas can be approached.

The online meetings are mediated by real-time video streaming of the teacher's PC screen, where his use of the various programming and modeling tools are shown. There is a corresponding audio stream, where the students can hear how the teacher reason and think aloud about the problem. In some theories of apprenticeship, the use of language is considered crucial in the master-apprentice relationship. This is pertinent for the apprentice's learning while the master is performing the actions of the craft of programming. But it is also important in order for the teacher to get a feeling of the skills and knowledge of the students, and in particular in situations where the teacher and the students are geographically separated. In order to support interactions amongst students and between student and teacher during the online meetings, a text-based chat conference in conjunction with the real-time audio- and video streams are organized.

Another artifact organized to facilitate the students' knowledge construction while working on the weekly assignments, is a collection of short demonstrations of how to approach specific issues. These are made in a similar manner to the online meetings; there is a video stream showing how the teacher approaches the problem together with an audio stream where the teacher thinks aloud. The difference from the online-meetings is that these demonstrations are available at any time, and that there is no opportunity for interaction. The basic principle, however, is that the teacher legitimates an approach to solving the problem. Such a demonstration typically involves modeling and programming in the Blue J environment, including frequent compilations that sometimes reveal (intentional) syntax errors, tests of parts of the solution, and consultation of Java SDK documentation. In this way there is a focus on the programming process as well as the conceptual understanding of object orientation

4.2. The decentered approach

The idea behind the design is to create opportunities for the students to participate in an actual practice of programming experts so as they gradually learn through legitimate peripheral participation. This can be further operationalised by utilizing the different backgrounds of the adult students so as they become each other's experts and legitimates in the shared learning community. Theoretically, individual knowledge is mediated by the apprentices' shared interests in learning object-oriented programming and by the ICTs and other resources s(he) has available.

This important social aspect of learning is taken into the pedagogical design, and operationalised through both technological and organizational elements. On the *technological side*, the course design facilitates for collaboration by offering the students tools for text-based communication. All the students installed the chat client *Yahoo! Messenger*, and registered all the course participants as "friends" – enabling them to see who of their peer students that are available for interaction at any given time. Thus, in addition to acting as a tool for planned collaborative events, Yahoo! Messenger also gives the students opportunities for more spontaneous interaction. Additionally, there is a web-based discussion forum available for the students. This tool is aimed at mediating the student's dialogues with peer-students where time is not a critical factor.

On the *organizational side*, there are two important mechanisms for supporting the social interactions amongst students. During the course, the students meet physically three times. One central aspect of these weekend seminars is to stimulate collaborative activities while the student works distributed. Experience from net-based learning points to the importance of such face-to-face meetings for online collaboration [5]. The other mechanism is that the students are divided into groups. These groups are put together based on where the students live, in order to make physical meetings outside the weekend seminars easier. The student groups are given tasks during the weekend seminars, and they are encouraged to work together during the full length of the course.

5. Concluding remarks

In this paper, we have shown how socio-cultural theories have informed the design of a course resting on a model-based philosophy of object-orientation. An essential learning objective within this philosophy is the programming processes *as such*. This means that the individual student should construct knowledge on how programmers develop their solutions from the initial problems to the final code. To move toward this learning goal, the IOOP course design has incorporated a combination of the so-called person-centered and decentered approaches to apprenticeship learning.

By using these socio-culturally inspired approaches, the design focus is the *individual learning activity* rather than instructional techniques and guidelines. Furthermore, the aim of designs is to organize for a *combination of artifacts* that each has embedded characteristics and conditions for operationalising the individual actions. As such, the theory is strong for analyzing what artifact (learning resources, the instructor's guides, text book, etc.) that is important for the individual learning processes of object-oriented programming. Analysis of the qualitative interviews (in forthcoming papers) will hopefully provide insights on this important issue of the field of learning object-oriented programming.

Our study so far, however, indicates that the person-centered approach to apprentice learning has been very successful, while the decentered approach to apprentice learning is found to be more problematic when it comes to practice. There are at least three aspects that make the decentered approach to apprenticeship problematic for our target group. First, the students are novices in object-oriented programming and may as such be too immature to play a role as experts for co-students. Second, ICT-mediated collaboration requires a well-orchestrated interdependence amongst the students (requires sharing of meaning, certain division of labor, etc.) [6], and that a certain level of regulation and tutor guidance are often desired to succeed [5]. Third, and certainly in line with the second argument: due to the life situation of many of the students (committed to family and work besides their study), individual study – which allows for greater flexibility – was preferred to collaboration with peer students.

Our preliminary analysis indicates that the aspects of the course modeled on the person-centered approach to apprenticeship, were more successful. There are, however, issues to be addressed in this design of the online meetings too. In the beginning of the course, the interactions during the online meetings were mediated by text chat, enabling the students to ask questions when they had problems. The outcome of this technical design was silence! This lead to a change in the use of the chat application, i.e., much more interaction was initiated by the teacher where he raised questions etc. to the students. Further elaboration on this part of the design is needed.

6. Acknowledgments

This research is conducted as a part of project COOL[1] (Comprehensive Object-Oriented Learning), financed by the Norwegian Council of Research; the development of the course was done as a part of the Flexnet[2] project funded by it university west. A special thanks to Sten Ludvigsen for giving valuable comments on drafts of this paper.

7. References

[1] Bennedsen, J., Caspersen, M. *A Model-First Approach to Teaching Introductory Object-Orientation*. Workshop on Learning and Teaching Object-Orientation – Scandinavian Perspectives. Oslo, 2003.

[2] Bennedsen, J., Eriksen O. “Applying and Developing Patterns in Teaching”. In: *Procedings of Frontiers in Education* 2003.

[3] Bergin, J. *Fourteen pedagogical patterns*, http://csis.pace.edu/~bergin/PedPat1.3.html, 2003

[4] Davydov, V. V., Zinchenko, V. P., Talyzina, N. F., *The problem of activity in the works of Leont'ev*. Soviet Psychology, Vol 24, No 4, 31-42, 1983.

[5] Fjuk, A. *Computer Support for Distributed Collaborative Learning. Exploring a Complex Problem Area*. Dr. Scient Thesis, Department of Informatics. Oslo, University of Oslo, 1998.

[6] Fjuk, A. and Ludvigsen, S. “The Complexity of Distributed Collaborative Learning: Unit of Analysis”. In: *Proceedings of Euro CSCL '01. First European Conference of Computer-supported Collaborative Learning*. Dillenbourg, P., Eurelings, A. and Hakkarainen, K. (Eds.) Maastricht, 2001.

[7] Engeström, Y. *Learning by Expanding: An Activity-Theoretical Approach to Developmental Research*. Helsinki, Orienta-Konsultit, 1987.

[8] Holmboe, C. *Language, and the learning of data modelling*. Dr. Scient Thesis, Department of Teacher training and School development, University of Oslo, Norway, 2003

[9] Knudsen, J.L., and Madsen, O.L., *Teaching Object-Oriented Programming is more than Teaching Object-Oriented Programming Languages*, DAIMI-PB 251, Department of Computer Science, University of Aarhus, Denmark, 1990.

[10] Kölling, M., Quig, B., Patterson, A. and Rosenberg, J. “The BlueJ System and its Pedagogy”. In *Computer Science Education*, Vol.13, No.4, pp. 249-268, 2003.

[11] Lave, J. and Wenger, E. *Situated Learning: Legitimate Peripheral Participation*. Cambridge, Cambridge University Press, 1991.

[12] Leont'ev, A. N. *Activity, Consciousness, and Personality*. Englewood Cliffs, Prentice-Hall, 1978

[13] Nielsen, K., Kvale, S. “Current issues of apprenticeship”. In: *Nordisk Pedagogik*, Vol 17, pp. 130-139, 1997.

[14] Schön, D. *The Reflective Practitioner: How Professionals Think in Action*. Harper Collins Publishers, 1983.

[15] Vygotsky, L., *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, 1978.

[16] Vygotsky, L., *Thought and language*. (Ed. by Kozulin, A.). The MIT Press, Cambridge, 1994.

[1] http://www.intermedia.uio.no/cool/
[2] http://www.it-vest.dk/aktiviteter/index.shtml