

Adopting Exploratory + Collaborative Learning in an Adaptive CSCL Environment for Introductory Programming

Agoritsa GOGOULOU, Evangelia GOULI, Maria GRIGORIADOU
*Department of Informatics & Telecommunications, University of Athens,
TYPA Buildings, Panepistimiopolis, GR 15784, Athens, Greece*

Abstract. The work, presented in this paper, examines and discusses issues concerning the adoption of the ECLiP (Exploratory + Collaborative Learning in Programming) framework in the SCALE (Supporting Collaboration and Adaptation in a Learning Environment) environment in order to support and enhance learning in Introductory Programming courses. ECLiP provides guidelines for the design of an integrated set of learning activities, in the context of any subject matter, comprising a three-step process - Acquiring knowledge, Constructing knowledge by Exploring+Collaborating, and Applying-Refining Knowledge. SCALE is a web-based adaptive collaborative environment, which enables students to work on collaborative learning activities concerning various subject matters of different orientation, guides and helps them through intelligent agents, supports alternative models of collaboration between the group members, and facilitates the collaboration/communication by adapting the “communication-scaffolding” tools (i.e. sentence openers and communication acts) according to the learning outcomes addressed by the learning activity and/or the model of collaboration followed by the group members. SCALE supports the elaboration and the accomplishment of the “programming” learning activities, designed on the basis of the ECLiP framework, by enabling the utilization of the appropriate educational tools, activating the assessment and the feedback modules, guiding the students during the activity elaboration, enabling the students’ collaboration in groups acting equivalently or according to specific roles and facilitating the students’ communication.

Introduction

In Introductory Programming courses, the students are expected to acquire knowledge about the main programming concepts/constructs and to cultivate basic skills concerning the development of simple programs. A lot of research effort is devoted to the improvement of the educational setting concerning the teaching and the learning processes. To this end, new teaching approaches are proposed and evaluated in real-classroom environments exploiting characteristics from contemporary theories of learning such as collaborative learning (e.g. the approach of “pair-programming” [1]), exploratory learning (e.g. the “Black-Box” method [2], the “explorations” [3]), etc. Additionally, educational tools are developed and used in the teaching and the learning processes, such as (i) microworlds/mini-languages [4] aiming to support the students with simple programming languages, (ii) visual programming/algorithmic design environments [5] aiming to reduce the focus on the syntax and facilitate the processes of the solution design and the program development, (iii) program visualization environments [6], [7] aiming to help the students to visualize the program execution and to debug their programs, etc. Also, research is devoted to the development of collaborative learning environments,

incorporating intelligent agents and/or providing adaptive capabilities, focusing on the development of good habits in programming [8], the support of programming projects [9], etc.

The work, presented in this paper, examines and discusses issues concerning the adoption of the ECLiP (Exploratory + Collaborative Learning in Programming) framework [10] in the SCALE (Supporting Collaboration and Adaptation in a Learning Environment) environment [11] in order to support and enhance learning in Introductory Programming courses. The ECLiP framework forms a basis for the design of an integrated set of learning activities, in the context of any subject matter, while SCALE is a web-based adaptive collaborative learning environment, which provides specific capabilities in terms of the collaborative learning activities that are supported and the collaborative facilities provided to the students. More specifically, the work focuses on issues concerning the elaboration and the accomplishment of various alternative “programming” learning activities, designed on the basis of the ECLiP framework, in the SCALE environment.

The rest of the paper is structured as follows: In Section 1, we describe the ECLiP framework, which adopts characteristics from the exploratory and collaborative learning. In Section 2, we present the main functionality of the SCALE environment. In Section 3, issues concerning the adoption of the ECLiP framework in the SCALE environment are examined and discussed. The paper ends with the main points of our work and our near future plans.

1. The ECLiP Framework for the Design of “Programming” Learning Activities

Following the key idea of the LfU model [12] as well as on the “programming” learning activities, which we developed based on a number of alternative teaching approaches taking into account the students’ difficulties on specific programming concepts/constructs [13], we proposed the ECLiP framework [10], which provides guidelines for the design of an integrated set of learning activities, in the context of any subject matter. The framework comprises a three-step process in order to (i) motivate the students to acquire new knowledge, (ii) support the construction of new knowledge by engaging the students in exploratory and collaborative learning activities, and (iii) enable the application and refinement of the new knowledge. The elaboration of the framework in the design of “programming” learning activities for Introductory Programming courses indicates the following:

Acquiring knowledge: Learning is more effective if the students participate in learning activities that are perceived to be meaningful [14] and the new knowledge is assimilated when the students require the acquisition of the knowledge [15]. Therefore, it is important to set up conditions that (i) are likeable/meaningful to the students, and/or (ii) are related to a goal that is challenging, and/or (iii) give the students the opportunity to express their beliefs/opinions. In Introductory Programming courses, it is essential to engage the students in learning activities concerning simple authentic problems that are close to their experience and show the usefulness of the programming process beyond the specific course. Instead of asking the students to solve numeric problems [4], the engagement in problems that makes them to think of “mapping” an “every-day” process to a “programming” process, may stimulate them to become curious and seek for new knowledge. Besides, the learning activities should address explicitly the learning goals to be achieved and may be enriched with various forms of scaffolding (e.g. questions, case studies) in order to create the appropriate conditions for the students to become aware of their own difficulties/incomplete understanding.

Constructing knowledge by Exploring+Collaborating: Knowledge construction is supported through observation and/or communication with others [12]. The learning activities should guide the students towards the activation and revision of their existing mental model. The students’ engagement in learning activities, which support the exploration of the particular functional properties of the programming constructs and provide guidance through suitably

designed questions and additional scaffolding tasks, enables them to understand the functional characteristics of the programming constructs and revise appropriately their mental model in case of preconceived misconceptions. Moreover, the students' involvement in collaborative activities enhances learning since they are forced to externalize/negotiate on their thoughts/ideas, to argue on their actions or on their points of view and to articulate their reasoning [14]. The collaboration may take place at different stages of the learning activity, depending on the learning goal/outcomes and the underlying content, in the form of groups where the students act (i) equivalently by discussing and exchanging ideas, or (ii) according to specific roles, such as the case of "pair-programming" [1].

Applying-Refining knowledge: The processes of reflection and application supports knowledge refinement and contributes to its retention, future retrieval and use [12]. Reflection in programming may be achieved (i) by asking the students to check their thinking [14], and/or reason their decisions, and/or (ii) by engaging the students in collaborative activities in which they examine and discuss their ideas with others and/or evaluate their peers' statements/solutions. The learning activities concerning the application of knowledge, may ask the students (i) to develop/modify a simple program, and/or (ii) to check the correctness of a program and modify it according to the problem definition, and/or (iii) to act as evaluators of their peers' work. During this step, the students may collaborate from the beginning or at different stages of the learning activity (e.g. at the beginning for exchanging ideas and designing a problem solution and/or at the end for completing the development of a program).

The application and the evaluation of the ECLiP framework in the context of an Introductory Programming course in a real classroom environment, has proved its effectiveness in enhancing learning [10]. The "programming" learning activities, which were designed on the basis of the framework taking into account the students' difficulties on specific programming concepts/constructs [10], [13], enabled the subjects of the study to gain deeper understanding of the programming constructs under consideration and apply them more effectively in solving "programming" problems [10].

2. The SCALE Environment

The SCALE environment aims to enable the students to work on collaborative learning activities concerning various subject matters of different orientation (i.e. practical and theoretical), to guide the students at the communication and at the learning level through intelligent agents, to support alternative models of collaboration between the group members, and to provide adaptive capabilities as far as the communication tool is concerned according to the learning outcomes addressed by the learning activity and/or the model of collaboration. More specifically, the main functionality of the SCALE environment includes [11]:

Working on collaborative activities: A library of learning activities is available to the students. The learning activity includes one or more sub-activities and/or question items, addressing specific learning outcomes. The learning outcomes are classified to the following four levels of cognitive processes [16]: (i) *Comprehension level*, (ii) *Application level*, (iii) *Checking-Critiquing level*, and (iv) *Creation level*. The elaboration of each question item may imply the use of different support educational tools such as concept mapping tools, simulation programs, educational software, etc or navigation and information retrieval from the web.

Collaborating in groups: Depending on the context and the nature of the learning activity, the students may collaborate in groups (up to four members) acting equivalently or according to specific roles.

Coaching and guiding collaboration and learning: SCALE fosters collaboration and learning through coaching/guiding. More specifically, SCALE supports a Collaborative Coach agent (CCa), which is responsible for the determination of the appropriate intervention actions

at the communication level (e.g. provides motivational prompts by encouraging/motivating the students to participate in case they seem to be passive). During the activity elaboration, the Learner Assistant agent (LAa) guides the students, either in an unsolicited form by giving hints/making proposals or in a solicited form by meeting the students' help requests. Apart from this form of guidance, SCALE incorporates a feedback module, which is responsible to provide the appropriate form of feedback to the students upon the completion of the learning activity (e.g. correct answer, hints/guidelines or relevant examples/material in order to guide the students to think of the correct answer).

Facilitating and supporting the group communication by providing adaptive capabilities: SCALE supports adaptivity of the provided "communication-scaffolding" tools (i.e. sentence openers and communication acts). According to the level of the learning outcomes, SCALE supports sentence openers when the learning outcomes concern the Comprehension, the Application and the Checking-Critiquing level. In the case of the Creation level, the environment supports communication acts since for higher order cognitive skills, it suffices to guide the students in terms of their intention/action. Regarding the model of collaboration, SCALE supports communication acts when the students collaborate under specific roles, since they serve more effectively the cultivation of the underlying cognitive/communication skills. The sentence openers/communication acts interface is adapted according to the level of the learning outcomes or the roles assigned by the model of collaboration.

In case learning activities do not explicitly address learning outcomes of one out of the four aforementioned levels of cognitive processes/skills, but they rather aim to cultivate to the students skills in communication, and/or to enable them to discuss/exchange ideas on a specific topic, or on the subject/solution of an activity, etc, the communication between the students is carried out through communication acts enabling them to express their opinion, to agree/disagree, etc.

SCALE is in the initial phase of development. At this stage, we finish the implementation of the modules/interface for the activity presentation and the group communication. Also, we have determined initial sets of sentence openers [16] and communication acts based on the results of a series of experiments that we conducted.

3. Adopting ECLiP in SCALE

Taking into consideration the effectiveness of the ECLiP framework in enhancing learning [10] and the capabilities of the SCALE environment in supporting collaborative learning, we aim to support the elaboration and the accomplishment of the learning activities designed on the basis of the ECLiP three-step process in the SCALE environment. Although, SCALE supports students' collaboration on learning activities addressing various subject matters of different orientation and various levels of learning outcomes as well as alternative models of group collaboration, it is necessary to examine whether it can support effectively the learning process in the context of the ECLiP framework for an Introductory Programming course. Therefore, we developed scenarios of applying the ECLiP framework in the design of "programming" learning activities. In the following, we present indicative scenarios and discuss whether and how SCALE supports the elaboration of such learning activities in terms of (i) the "communication-scaffolding" tools that are provided, (ii) the model of group collaboration that is followed, (iii) the educational tools that are needed, and (iv) the guidance/help provided by the environment.

3.1 Scenarios of applying the ECLiP three-step process in SCALE

Learning activities for acquiring knowledge: According to the first step of the ECLiP framework, the learning activities aim to motivate the students and create demand for knowledge and/or elicit curiosity. The motivation to acquire content understanding and/or to cultivate specific skills may concern a completely unknown subject/concept to the students or a subject/concept that has been misconceived or presents significant learning difficulties.

In case of a completely unknown subject/concept, the learning activity may require that students think and design the solution of a simple given problem. For example, in order to introduce the students to the concept of looping constructs, the learning activity may situate the students in an authentic and challenging context asking them to play the role of a program designer and to design the solution of a given problem requiring the use of a looping construct. The students are expected to become keen on how solving the problem and to find out that their existing “programming” knowledge is inadequate. The activity may include a number of question items, having the form of free response questions, asking the students to express their opinion of (i) whether the problem’s solution requires any kind of repetition, (ii) how they can apply what they already know in the context of the specific problem, (iii) whether they need a “new” programming construct to be available and how it may function, etc. Additionally, the learning activity may ask the students to collaborate in groups and exchange their ideas on the given problem. Upon the completion of the activity, the students may be provided with the appropriate feedback having the form of hints about the solution or examples addressing similar problems.

The SCALE environment supports the aforementioned activity quite well. More specifically, in case each student elaborates on the learning activity on his own, the environment does not support any facility for communication. The modules/mechanisms that are activated, are those related to the composition/submission of the student’s answer, the provision of the appropriate feedback and the support of the learning process through the LAa. On the other hand, if the activity asks the students to collaborate, then the appropriate “communication-scaffolding” tools are provided on the basis of the addressed learning outcomes or the model of collaboration. In particular, in the context of the aforementioned activity, which urges the students to think of/reason/discuss in groups the solution of a given problem, the interface of the communication tool is adapted to the communication acts, making available to the students all those keywords that enable them to express their beliefs, to agree/disagree to a proposal, to ask for clarification, etc. Apart from the activity response area where the students compose/submit their common answers, all the students have at their disposal a personal working area to write down their own ideas and their personal answers. Also, the CCa is activated in order to prompt/activate the passive students.

In case of a subject/concept that has been misconceived or presents significant learning difficulties, the learning activity may ask the students to predict the execution results of a given program, to compare the actual results to the predicted ones and to explain any differences between them. For example, in the case of an infinite while loop, which presents difficulties in its identification, the learning activity may focus on a program addressing an infinite loop, asking the students to predict the execution results and the number of times of loop execution, to compare the actual/correct results/answers to the predicted ones and to explain any differences (addressing learning outcomes of the Comprehension level). The actual execution results are available to the students either in a written form, or by running the corresponding executable program. This kind of activity aims to bring to the surface the students misconceptions and to make them wonder about the correctness of their answers. During the elaboration of such an activity in the SCALE environment, the students have at their disposal the appropriate tools (e.g. the programming environment) in order to run the executable form of the program. In case they collaborate in groups, the provided

“communication-scaffolding” tool is adapted to a sentence openers interface with respect to the addressed learning outcomes.

Learning activities for constructing knowledge by exploring+collaborating: According to the second step of the ECLiP framework, the learning activities aim to support the knowledge construction process by enabling the students to explore on their own the functional properties of the programming constructs under consideration and/or by collaborating with other students. More specifically, the learning activities may include a number of sub-activities aiming to introduce the students to the functional properties of a programming construct gradually, presenting different and alternative aspects of its function and application in the context of various “programming” problems. Also, the learning activities may focus on specific learning difficulties aiming to help the students to overcome their misconceptions. In any case, each learning activity (or sub-activity) may include a number of question-items having the form of multiple-choice/true-false/ordering/fill-in the blank questions, in order to enable the automatic assessment of the students’ answer and the provision of the appropriate feedback in the SCALE environment. Also, the students may collaborate at various stages of the learning activity following a specific model of collaboration.

For example, taking into account the students’ difficulty in identifying an infinite loop and the non-existence of the “update” statement [10], [13], the learning activity may consist of three sub-activities: (i) the first one asks the students to run two slightly different programs (their difference concerns the existence/non-existence of the “update” statement) for a pre-specified set of input values and to answer to a series of questions related to the messages appeared and to the difference of the two programs as far as the execution results are concerned, etc, (ii) the second sub-activity asks the students to study the code of the two corresponding programs and to answer to a number of questions aiming to guide them to observe the existence/non-existence of the “update” statement in the two programs, to understand its role/function and to relate it to the infinite execution of the “while” loop in the case of the second program, and (iii) the third sub-activity asks the students to collaborate, to discuss their answers to the questions of the two preceding sub-activities, and to give a common answer about the difference of the two programs regarding the execution results.

The SCALE environment supports the elaboration of such activities, by enabling the utilization of the appropriate tools (e.g. the programming environment), required to run the executable form of the programs and by activating the assessment and the feedback modules in order to assess the students’ answers and provide immediate feedback. Also, the environment enhances the support and guidance to the students during the elaboration of the activity, by enabling the personal LAa to provide hints and help to the students. In case of collaboration, the “communication-scaffolding” tool is adapted either to the level of the addressed learning outcomes of each activity/sub-activity (i.e. the aforementioned sub-activities address learning outcomes of the Comprehension level) or to the model of collaboration.

Learning activities for applying-refining knowledge: The third step of the ECLiP framework addresses learning activities, which enable students to reorganize their knowledge and connect the new knowledge to the existing, by applying and reflecting on the new knowledge. More specifically, the learning activities may ask the students to apply their knowledge (i) by determining and implementing specific statements given a problem and an incomplete solution to the problem, and/or (ii) by evaluating the correctness and the completeness of a given program and proceeding with the appropriate modifications, and/or (iii) by modifying an existing program in the context of a slightly different problem, and/or (iv) by developing a program in the context of a specific problem. The first case addresses learning outcomes of

the Application level, the second one of the Checking-Critiquing and the Application level while the last two cases address learning outcomes of the Creation level.

As far as the question items included in the learning activity (or sub-activity) are concerned, these have the form of fill-in the blank/multiple-choice/true-false/short-answer questions (first and second case) or the form of free-response questions – construction of a program (third and fourth case). The form of the question item affects the capability of the environment to provide automatic assessment and feedback. In particular, the question items that require the construction of a program from scratch are difficult to be automatically assessed since it is necessary to have the appropriate evaluation systems that enable the judgment of the program correctness and the provision of advice to the students.

During the activity elaboration, the students may collaborate by playing specific roles or acting equivalently, facilitating in this way the reflection process. For example, an indicative activity may be consisted of two sub-activities, like: (i) the first one asks the students to develop a simple program in the context of a specific problem, and (ii) the second one requires the evaluation of a pre-specified solution to the same problem. The students may collaborate in pairs by playing specific roles (“driver”/“observer” [1]) and interchanging these roles in the context of the two sub-activities.

The SCALE environment supports effectively the accomplishment of the above learning activities. As already mentioned, SCALE enables the utilization of the appropriate programming environments for the composition and the execution of a program and provides the appropriate mechanisms for automatic assessment and feedback to the students’ answers (in case of question items having the form of fill-in the blank/multiple-choice/etc questions) and for guidance/help through the intelligent agents. For example, in case of an activity, which requires of the students to fill in the missing statements, the SCALE environment presents to the students the problem and the incomplete solution and enables them to fill in the required statements through the available text fields. Also, in such an activity, the environment automatically assesses the submitted answer and provides the appropriate available feedback to the students. Regarding the “communication-scaffolding” tool, SCALE adapts the sentence openers/communication acts interface according to the level of the addressed learning outcomes or the model of collaboration. That is, in case the students collaborate according to specific roles (as in the above mentioned example), the level of the learning outcomes is not taken into account, and the “communication-scaffolding” tool is aligned to a communication acts interface, which is adapted, for each student, according to the underlying roles. In case students collaborate equivalently, the “communication-scaffolding” tool is adapted to a sentence openers interface or a communication acts interface, according to the underlying level of the addressed learning outcomes.

4. Conclusions

In this paper, we discuss issues concerning the adoption of the ECLiP framework, which forms a basis for the design of an integrated set of learning activities, in the SCALE environment, in order to support and enhance learning in Introductory Programming courses. The “programming” learning activities, which are designed on the basis of the ECLiP framework, may (i) include question items having the form of various types of questions such as multiple-choice questions, free-response/short answer questions, aiming to motivate the students to acquire knowledge, help/guide the students in the knowledge construction process, and help/support the students in the knowledge application/refinement process by enabling them to construct/modify/evaluate/correct a program, (ii) require the utilization of appropriate educational tools for the composition and the execution of the addressed programs, and (iii) ask the students to collaborate in groups acting equivalently or according to specific roles.

These requirements seem to be quite well supported by the SCALE environment taking into account its capabilities and tools. In the near future, upon the completion of the modules/interface for the activity presentation and the group communication of the SCALE environment, we plan to investigate/evaluate whether SCALE supports effectively alternative “programming” learning activities with respect to the provided “communication-scaffolding” tools and the models of collaboration.

References

- [1] Williams, L., Upchurch, R.L., “In Support of Student Pair-Programming”, *In Proceedings of SIGCSE '01 Conference*, ACM Press, Charlotte, USA, 2001, pp. 327-331.
- [2] Haberman, B., Kolikant, Y.B.D., “Activating «Black Boxes» instead of opening «Zippers» - a method of teaching novices basic CS concepts”, *In Proceedings of ITiCSE '01 Conference*, ACM Press, Canterbury, UK, 2001, pp. 41-44.
- [3] Lischner, R., “Explorations: Structured Labs for First-Time Programmers”, *In Proceedings of SIGCSE '01 Conference*, ACM Press, Charlotte, USA, 2001, pp. 154-158.
- [4] Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., Miller, P., “Mini-languages: A Way to Learn Programming Principles”, *Education and Information Technologies*, 2(1), 1997, pp. 65-83.
- [5] Crews, T., Ziegler, U., “The Flowchart Interpreter for Introductory Programming Courses”, *In Proceedings of FIE '98 Conference*, Tempe, Arizona, USA, 1998, pp. 307-312.
- [6] Satratzemi, M., Dagdilelis, V., Evangelidis, G., “A system for program visualization and problem-solving path assessment of novice programmers”, *In Proceedings of ITiCSE '01 Conference*, ACM Press, Canterbury, UK, 2001, pp. 137-140.
- [7] Dann, W., Cooper, S., Pausch, R., “Making the Connection: Programming With Animated Small World”, *In Proceedings of ITiCSE '00 Conference*, ACM Press, Helsinki, Finland, 2000, pp. 41-44.
- [8] Vizcaino, A., Contreras, J., Favela, J., Prieto, M., “An adaptive collaborative environment to develop good habits in programming”, *In ITS 2000*, LNCS 1839, Springer, 2000, pp. 262-271.
- [9] Lopez, N., Numez, M., Rodriguez, I., Rubio, F., “Including Malicious Agents into a Collaborative Learning Environment”, *In ITS 2002*, LNCS 2363, Springer, 2002, pp. 51-60.
- [10] Gogoulou, A., Gouli, E., Grigoriadou, M., Samarakou, M., “Exploratory + Collaborative Learning in Programming: A Framework for the Design of Learning Activities”, accepted in *IEEE ICALT 2003 Conference*, Athens, Greece, 2003.
- [11] Gogoulou, A., Gouli, E., Grigoriadou, M., Samarakou, M., “Supporting Collaboration and Adaptation in a CSCL Environment”, accepted in *IEEE ICALT 2003 Conference*, Athens, Greece, 2003.
- [12] Edelson, D., “Learning-for-Use: A Framework for the Design of Technology-Supported Inquiry Activities”, *Journal of Research in Science Teaching*, 38(3), 2001, pp. 355-385.
- [13] Grigoriadou, M., Gogoulou, A., Gouli, E., “Alternative teaching approaches in introductory programming courses: teaching proposals”, *In Proceedings of the 3rd Hellenic Conference, with international participation, on ICT in Education*, Rhodes, Greece, 2002, pp. 239-248.
- [14] Vosniadou, S., “How children learn”, *Educational Practices Series*, n°7, <http://www.ibe.unesco.org/International/Publications/EducationalPractices/prachome.htm>
- [15] Schank R., Berman T., Macpherson K., “Learning by Doing”, In Charles M. Reigeluth (Eds.), *Instructional-design Theories and Models, A New paradigm of Instructional Theory*, Volume II, Lawrence Erlbaum Associates, 1999, pp. 161-184.
- [16] Gouli, E., Gogoulou, A., Grigoriadou, M., Samarakou, M., “Towards the development of an Adaptive Communication Tool promoting Cognitive and Communication Skills”, accepted in *PEG 2003 Conference*, St. Petersburg, Russia, 2003.