Visualizing the Complex Process for Deep Learning with an Authentic Programming Project

Jun Peng[1], Minhong Wang[1*] and Demetrios Sampson[2,3]
[1]KM&EL Lab, Faculty of Education, The University of Hong Kong, Hong Kong // [2]School of Education, Curtin University, Australia // [3]Department of Digital Systems, University of Piraeus, Greece // u3002116@connect.hku.hk // magwang@hku.hk // demetrios.sampson@curtin.edu.au, sampson@unipi.gr
[*]Corresponding author

ABSTRACT
Project-based learning (PjBL) has been increasingly used to connect abstract knowledge and authentic tasks in educational practice, including computer programming education. Despite its promising effects on improving learning in multiple aspects, PjBL remains a struggle due to its complexity. Completing an authentic programming project involves a complex process of applying programming strategies to design and develop artifacts. Programming strategies are often implicit and hard to capture, but critical for programming performance. This study proposes a visualization-based learning environment that externalizes the complex process of applying programming strategies to the design and development of solutions to an authentic programming project. It aims to make the complex process accessible, trackable, and attainable with the support of technology. Twenty-nine senior college students participated in this study, using the proposed learning environment to complete a PjBL module of ASP.NET. The proposed approach improved students' programming performance and subject knowledge and activated their intrinsic motivation to learn programming.

Keywords
Visualization, Project-based learning, Complex process, Computer programming, Authentic learning, Computer-based learning environment

Introduction

The learning of computer programming has received increased attention with the rapid growth of industry demand for programmers and student interest in programming. However, computer programming is a hard subject for many learners (Govender & Grayson, 2008; Lahtinen, Ala-Mutka, & Järvinen, 2005). A programmer must master both programming knowledge (e.g., concepts, syntax, and semantics) and programming skills and strategies (Robins, Rountree, & Rountree, 2003). The latter refers to the general and specific skills and strategies for integrating abstract programming knowledge into programming tasks by planning, designing, and implementing solutions. Different from programming knowledge, programming skills and strategies are more implicit and harder to capture, yet are critical for programming performance (Soloway, 1986).

Traditional programming education has focused on elaborating abstract knowledge rather than on linking knowledge to specific contexts, which can be labeled as "knowledge driven" or "teacher-centered" education (Robins, Rountree, & Rountree, 2003). As a result, many students demonstrate fragile knowledge (i.e., missing, inert, and misplaced knowledge) and unsatisfactory programming skills and strategies despite passing their programming courses with decent grades. To address the gap, project-based learning (PjBL) has been increasingly promoted by encouraging learners to work with authentic programming tasks and develop artifacts, such as computer programs or design plans, that are realistic products closer to professional reality (Blumenfeld, Soloway, Marx, Krajcik, Guzdial, & Palincsar, 2011).

PjBL contributes to the meaningful learning of abstract knowledge, the improvement of student motivation and confidence, and to the development of communication and problem-solving skills (Kay et al., 2000; Thomas, 2000). Nevertheless, the implementation of PjBL in programming courses, mainly in advanced courses, remains a struggle for many educators (Perrenet, Bouhuijs, & Smits, 2000). PjBL involves a wide range of complex problem-solving activities and extensive hands-on practice, which challenge teachers' ability to design PjBL curricula and support students throughout their projects. A major concern is that programming tasks involve complex cognitive processes that are inaccessible to learners and instructors. Such complexity can overwhelm learners, making them unable to engage in deeper learning experiences and achieve desired learning outcomes (Helle, Tynjälä, & Olkinuora, 2006; Pucher & Lehner, 2011; Thomas, 2000). Moreover, these complex processes are not always made evident to instructors due to the inherent difficulty of tracking and capturing them, thereby limiting instructors' capacity to facilitate and improve student performance. Additionally, the resource-consuming nature of PjBL makes it difficult to implement without sufficient time, manpower, and resources (Pucher & Lehner, 2011; Thomas, 2000).

This study attempts to address this challenge by designing a visualization-based learning environment that makes the complex process of carrying out an authentic project accessible, trackable, and attainable throughout the learning process. It uses a web-based learning environment to support PjBL learning with more flexibility in the delivery of learning activities and provision of feedback and support to learners without time and space constraints. This learning environment also has the potential to save costs by reusing learning resources and saving manpower. The study uses ASP.NET as the learning subject, as it is a popular programming course and a mainstream language in the software industry. This study aims to explore the feasibility and effects of the proposed approach in supporting computer programming PjBL. The research questions (RQs) of the study are stated below.
RQ1: How can a visualization-based learning environment be designed to make the complex process of carrying out an authentic project accessible to learners in a programming course?
RQ2: What are the effects of the proposed approach on project-based learning in a programming course?

Literature review

Project-based learning (PjBL)

PjBL is a student-centered pedagogy that encourages students to learn by exploring solutions to real-world problems, mainly by creating artifacts (David, 2008). It highlights the integration of knowing and doing based on the belief that students acquire deep knowledge through active exploration of real-world problems. PjBL is closely related to inquiry learning, problem-based learning, and other learning approaches that aim to fill the gap between learning in a formal instructional environment and the application of knowledge in realistic settings. These learning approaches are underpinned by situated cognition and situated learning theories (Brown, Collins, & Duguid, 1989; Lave & Wenger, 1991), which claim that knowledge is rooted in physical and social contexts and that meaningful learning only occurs when knowledge is created and applied.

PjBL distinguishes itself by placing more attention on the development of realistic products closer to professional reality and the assessment of product quality (Adderley et al., 1975; Nuutila, Törmä, & Malmi, 2005). In PjBL contexts, projects tend to involve complex tasks grounded in challenging problems that require students to devise solutions and formulate tangible outcomes by applying their knowledge; learning with a project often takes an extended period of time. Although PjBL has been increasingly promoted in educational practice and has shown promising effects on meaningful learning, it is often not fully implemented (e.g., in programming education) due to its complex and resource-consuming nature (Pucher & Lehner, 2011). Furthermore, the literature has reported inconsistent and inconclusive findings on the effects of PjBL (Blumenfeld et al., 2011; David, 2008).

Learning through complex tasks

Learning through an authentic project is characterized by carrying out a complex task or solving a sophisticated problem, which usually involves complex, implicit processes. The complexity of such processes may generate heavy cognitive loads for learners (Kirschner, Sweller, & Clark, 2006), which is often underestimated by instructors or experts, for whom many of the requisite processes have become largely subconscious or automated. With limited capability to capture the complex processes of authentic tasks, many learners tend to engage in surface instead of deep learning and are unable to achieve the desired learning outcomes (Wang, Kirschner, & Bridges, 2016).

Providing learners with a scaffold or the necessary support to complete a complex task is crucial to learning through ill-structured problems or authentic tasks (Belland, Walker, Kim, & Lefler, 2016; Hmelo-Silver, Duncan, & Chinn, 2007). Scaffolding aligns with the cognitive apprenticeship model, which claims that performing a complex task involves implicit processes. It is critical to make such processes visible for novices to observe, enact, and practice with expert help (Collins, Brown, & Holum, 1991). Approaches to scaffolding for complex tasks have focused on structuring or decomposing a complex task into a set of main actions or key questions to help learners recognize the important goals to pursue in their exploration (Reiser, 2004). The four-component instructional design model presents a guiding framework for systematic learning through complex tasks (van Merriënboer & Kirshner, 2007). The model involves four interrelated components: learning tasks, supportive information, procedural information, and part-task practice. In addition to decomposing a complex learning task, guidance, supportive information, and feedback should be provided for non-recurrent aspects of the tasks; procedural information can be offered as a just-in-time alert for recurrent aspects of the task; and part-task practice can be used to improve routine skills to reach a required level of automaticity. These suggestions have been incorporated into the design of this study's proposed learning environment.

Visual representations

Recent research has explored the use of visual representations to externalize and facilitate complex cognitive processes in complex tasks, and has shown improvements in knowledge and task performance (Gijlers & de Jong, 2013; Suthers, Vatrapu, Medina, Joseph, & Dwyer, 2008; Wang, Wu, Kinshuk, Chen, & Spector, 2013; Wu & Wang, 2012). Visual representations or graphic forms (e.g., diagrams, maps, tables, and pictures) have advantages in representing and communicating complex thinking and cognition in flexible ways (Alexander, Bresciani, & Eppler, 2015). By representing information or knowledge both verbally and spatially, such forms can afford more efficient cognitive processing and meaningful communication of complex issues than text messages alone. In recent decades, computer-based visual representations such as concept maps, causal maps, and system models have been increasingly used to extend and augment human cognition, and they have shown their advantages in fostering high-order thinking and meaningful learning in various contexts (Jonassen, 2005).

In computer programming, visual representations and simulation-based visualization tools have been used to visualize the complex structures and algorithms of software programs, explore the development and evolution of programs, and demonstrate the run-time behavior of programs to discover their defects (Koschke, 2003; Roels, Meştereagă, & Signer, 2016). In relation to this, visualization-based learning facilities (e.g., diagrams, pictures, and animations) have been used in programming education to engage learners and assist in their understanding of the abstract concepts and complicated behavior of programs (Eisenberg, Basman, & Hsi, 2014; Hundhausen & Brown, 2007; Naps et al., 2003; Sorva, Karavirta, & Malmi, 2013). They have also been used to help learners integrate separate pieces of knowledge into a coherent whole for meaningful understanding and effective application by allowing them to see a big picture of knowledge on a visual map (Wang, Peng, Cheng, Zhou, & Liu, 2011). Although visualization has shown positive effects on engaging programming learners, research has reported inconclusive findings on its effects on programming learning outcomes, with a major concern for how visualization technology can effectively be integrated with learning and instructional design (Rajala, Laakso, Kaila, & Salakoski, 2008; Sorva et al., 2013).

Methods

This study adopts a design-based research approach, a systematic methodology that creates, builds, and evaluates innovative artifacts or interventions to solve identified problems (Amiel & Reeves, 2008). Design-based research features iterative analysis, design, development, and implementation and close collaboration among researchers and practitioners in real-world settings, leading to contextually sensitive design principles and theories. It is particularly suitable when complex and ambitious educational reform policies are ill specified and the implementation process is uncertain (Wang, Vogel, & Ran, 2011).

Based on relevant learning theories and models, a visualization-based learning environment was designed by decomposing a complex project into a set of main actions, visualizing the process of the project and the process and/or structure of individual actions, and providing specific guidance or strategies for individual actions to make the implicit aspects more accessible. The students used the proposed visualization-based learning environment to complete a PjBL module of ASP.NET. The students' learning outcomes were measured in terms of subject knowledge, programming performance, and motivation to learn programming. Learner perceptions of the learning environment were also examined, as such perceptions have significant effects on learning in technology-mediated learning environments. Unless a system is properly designed and implemented to the extent that learners find it acceptable, further investigation into the effect of the approach may not produce reliable and meaningful results.

Proposed learning environment

Making the complex implicit process visible

To make the complex and implicit aspects of a programming project accessible, a project is decomposed into a set of main actions based on heuristics or disciplinary strategies. According to the literature on and practice of computer programming (Bassil, 2012; Deek & McHugh, 2002), a programming project typically comprises a number of key steps, namely problem understanding (or problem formulation), modular design (to design a plan of the solution), process design (to design a detailed solution), coding (to implement a solution), and evaluation and reflection (to test and deliver a solution). Accordingly, the process of completing a programming project is represented in a visual format, as shown in Figure 1. By clicking on the icon of each action, learners can enter the action space for learning and practice. After completing an action, learners can review and refine their outputs.

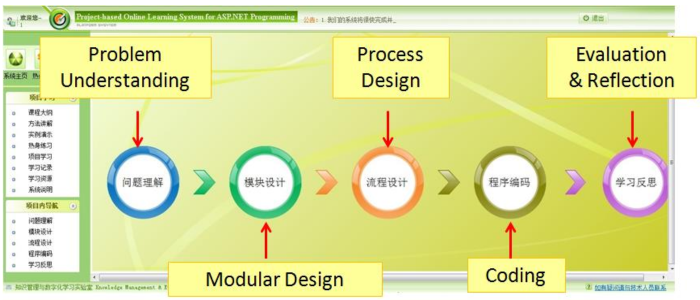
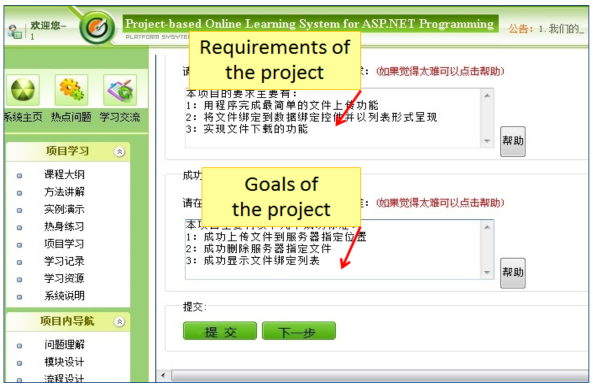


Figure 1. The process of completing a programming project

- Problem understanding
  The first step of a programming project is to formulate a clear understanding of the problem by identifying its goals, givens, and expected results. A structured form is provided to learners to state their understanding of the problem by specifying the requirements and goals of the programming project, as shown in Figure 2.



(如果觉得太难可以点击帮助)
本项目的要求主要有：
1：用程序完成最简单的文件上传功能
2：将文件绑定到数据绑定控件并以列表形式呈现
3：实现文件下载的功能

(如果觉得太难可以点击帮助)
1：成功上传文件到服务器指定位置
2：成功删除服务器指定文件
3：成功显示文件绑定列表

提交：

Figure 2. Problem understanding

- Modular design
  A computer program is often organized as a set of functions or modules to be developed independently and then combined to solve the problem. Based on the understanding of the problem, a solution plan can be generated by decomposing the main goals into sub-goals, identifying modular functions to accomplish each sub-goal, and specifying the relationships between the functions. A diagramming tool is provided to learners to build a functional block diagram to outline the plan of the solution, as shown in Figure 3. Moreover, structural analysis and design strategies are offered with a focus on the independence and completeness of the modules.

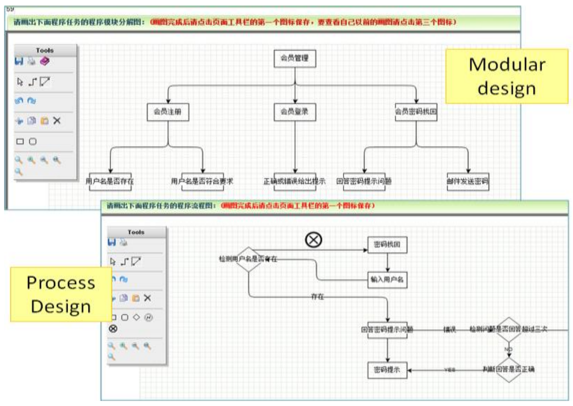
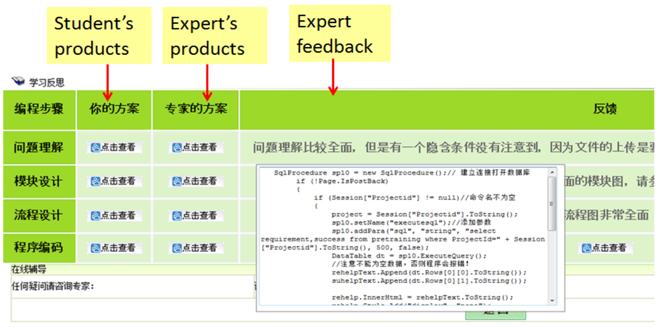


Figure 3. Modular design and process design

- Process design
  The process within and across the functions must be outlined to illustrate the solution to or algorithm of a given problem, mainly by showing the steps and connections between them. Learners can use the diagramming tool to build a flowchart for the software program, which demonstrates a detailed design of the solution (Figure 3). Learners are also provided with process design strategies, which a focus on priority analysis and critical analysis when designing a complex flowchart involving a number of interactive modules.

- Coding
  The modular design and process design can then be translated into program code as a solution to the project. Learners can upload their program codes, which can be reviewed and revised throughout their projects. More importantly, coding strategies are offered to learners with a focus on top-down gradual refinement in addition to their data structures and algorithms.

- Evaluation and reflection
  After completing their codes, learners must evaluate their programs by testing and debugging them. Moreover, they can reflect on their performance and areas for possible improvement by reviewing the artifacts generated in each action along with the comments and feedback of their instructor (Figure 4). They can also update their artifacts or solutions and receive further instructor feedback.



编程步骤	你的方案	专家的方案	反馈
问题理解	点击查看	点击查看	问题理解比较全面，但是有一个隐含条件没有注意到，因为文件的上传是
模块设计	点击查看	点击查看	面的模块图，请
流程设计	点击查看	点击查看	流程图非常全面
程序编码	点击查看	点击查看	点击查看

在线辅导
任何疑问请咨询专家：

```
    SqlProcedure sp10 = new SqlProcedure();// 建立连接打开数据库
        if (!Page.IsPostBack)
        {
            if (Session["Projectid"] != null)//命令名不为空
            {
                project = Session["Projectid"].ToString();
                sp10.setName("executesql");//添加参数
                sp10.addPara("sql", "string", "select
requirement,success from pretraining where ProjectId=" + Session
["Projectid"].ToString(), 500, false);
                DataTable dt = sp10.ExecuteQuery();
                //注意不能为空数据，否则程序会报错！
                rehelpText.Append(dt.Rows[0][0].ToString());
                suhelpText.Append(dt.Rows[0][1].ToString());

                rehelp.InnerHtml = rehelpText.ToString();
```

Figure 4. Reflection with feedback

Providing instructional guidance

In addition to decomposing a programming project into a set of actions, relevant instructions and guidance on how to apply relevant programming skills and strategies to perform each action are provided to learners as learning materials in the system along with a sample project for illustration. Programming skills and strategies are often implicit and hard to capture, yet are critical for programming performance (Soloway, 1986). Learners need clear guidelines on how to formulate a clear understanding of a problem by identifying its goals and requirements, how to generate a solution plan by organizing a set of modular functions and establishing their relationships, and how to design a logical and appropriate program flow to implement the solution.

Providing adaptive feedback to individuals

During the project, the instructor can observe and analyze the artifacts that students generate. Moreover, the instructor can use the system to provide feedback to each student by giving specific comments on his/her problem statement, modular design, program flowchart, and program code. The purpose is to enhance student engagement and in-depth reflection and eventually improve students' learning outcomes. It is important to make a formative assessment of learning performance based on the artifacts generated not only at the end but also during the project. Visual representations of the learning artifacts play an important role in making previously undetected learning processes accessible for self-reflection by the student and for analysis and feedback by the instructor.

Participants

Twenty-nine Year Three college students (12 males and 17 females) participated in this study by using the proposed learning environment to complete a PjBL module of ASP.NET.

Learning task

The participants were asked to complete a representative authentic programming project (membership management) using the proposed learning system. To complete the project, the students went through the main actions of problem understanding, modular design, process design, coding, and evaluation and reflection. The students completed each action by submitting the relevant learning artifacts (i.e., the problem statement, modular design, program flowchart, and code). During the project, the instructor reviewed the students' learning artifacts and provided individual feedback and comments via the online system.

Procedure

The learning module lasted for 6 weeks. In the first week, the students signed the consent forms for their participation in the study. A questionnaire was then administered to the participants to collect their demographic data. A knowledge test and a programming task were arranged to assess their subject knowledge and programming performance before the study. Next, the participants were given a 30-minute face-to-face instruction session on how to use the proposed system. Relevant information and guidance were also available in the system for flexible access. During the instruction session, a sample project was used for demonstration by the instructor and for self-practice by the students, allowing them to become familiar with the learning environment.

The students started their self-directed learning in the second week. They were asked to complete a project in their free time over a 4-week period. They were also asked to pace themselves and spend 4 hours per week on the project. They could use the online forums for flexible discussion and communication with other students. Based on the log data, most of the students spent approximately 3 hours per week with the system. For each project, most of the students received two to three comments on problem understanding and one to two comments on each of the other parts (i.e., modular design, process design, and coding).

In the sixth week, a knowledge test and a programming task were arranged to assess the students' subject knowledge and programming performance, respectively. In addition, questionnaires and semi-structured interviews were administered to assess student perceptions and collect feedback.

Measures

Pre-test questionnaire

The pre-test questionnaire was used to collect students' demographic information and as a self-assessment of their computer skills (very poor, poor, intermediate, good, very good) and intentions to use online learning applications (from strongly disagree to strongly agree).

Post-test questionnaire

The post-test questionnaire was used to collect students' perceptions of the learning system and intrinsic motivation to learn programming using the proposed system. It used a 5-point Likert scale ranging from 1 (strongly disagree) to 5 (strongly agree). Internationally validated and widely adopted measurements from the IT acceptance literature (Arbaugh, 2000; Davis, 1989; Gattiker & Hlavka, 1992) were used to measure student perceptions of the system in terms of usefulness, ease of use, intention to use, and overall satisfaction. Examples of the items include, "The system is useful for my learning," "The system is easy for me to use," "I intend to use the system," and "I am satisfied with the learning system."

The intrinsic motivation inventory model (Ryan & Deci, 2000) was adapted to measure the students' intrinsic motivation to learn programming using the proposed system. The motivation measurements involved five scales, including interest/enjoyment, effort/importance, value/usefulness, perceived competence, and pressure/tension. Examples of the items include, "I enjoy attending the learning module," "I feel confident during the learning module," and "I get nervous while studying."

Knowledge tests

Knowledge achievement was assessed before and after the study using two traditional knowledge tests (i.e., a pre-test and a post-test). Different questions of similar difficulty were asked in both tests, which each comprised single-choice, fill-in-the-blanks questions and a short program. All of the test questions were designed by an experienced programming instructor and a programming expert. The total score of each knowledge test was 100.

Programming tasks

Programming performance was assessed before and after the study using two programming tasks (i.e., a pre-task and a post-task). Different tasks of similar difficulty were used in both tests, which were designed by an experienced programming instructor and a programming expert. Both tasks were practical and moderately difficult. For example, in a program, the students were asked to create a class of students, store the name and grades of five courses for each student, calculate the average grade for each student, and display the results of all students.

Based on the literature on programming performance assessment (Deek et al., 1999), the instrument for assessing programming performance consisted of two distinct categories: the programming process and the programming product. The programming process was measured by three subscales: problem understanding, solution planning (i.e., modular design), and solution design (i.e., process design). The programming product (i.e., code) was measured in terms of correctness, efficiency, reliability, and readability. Accordingly, the assessment rubrics for programming performance in this study accounted for four aspects: problem understanding, modular design, process design, and coding. Based on the literature and common practice, the weighting was 40% for coding and 20% for each of the other aspects. The full programming performance score was 20 (8 points for coding and 4 points for each of the other aspects). The detailed assessment rubric with respect to the description, weight, rating criteria, and score range of each aspect is outlined in Table 1.

Table 1. Assessment rubric for programming tasks

Aspect	Weight	Description		Score range
Problem understanding	20%	4 – Problem is clearly and correctly stated. All goals, givens, and results are identified. 3 – Problem is correctly stated. Most goals, givens, and results are identified. 2 – Problem is partially stated and/or some facts are identified. 1 – Problem statement is incorrect and meaningless facts are identified. 0 – No problem representation/fact identification attempted or completely irrelevant work.		0 to 4
Modular design	20%	4 – Detailed and clear planning, with complete goal refinement and task identification. 3 – Adequate planning, with sufficient goal refinement and task identification. 2 – Partially correct planning, with some goal refinement and task identification. 1 – Incorrect planning and meaningless goal refinement. 0 – No planning/refinement attempted or completely irrelevant work.		0 to 4
Process design	20%	4 – Complete module decomposition, organization, and detailed specifications. 3 – Sufficient module decomposition, organization, and sufficient specifications. 2 – Partial design and/or some module specifications. 1 – Improper module decomposition, organization, and specifications. 0 – No design/specifications attempted or completely irrelevant work.		0 to 4
Coding	10%	Correctness	2 – Correct solution specifications/program code and results consistent with problem requirements. 1 – Partial solution specifications/program code and/or some results 0 – No solution specifications/program code, or results inconsistent with problem requirements.	0 to 2

	10%	Efficiency	2 – Most algorithms, data structures, control structures, and language constructs for this problem situation are appropriate. 1 – Program accomplishes its task, but lacks coherence in choice of either data and/or control structures. 0 – Program solution lacks coherence in choice of both data and control structures.	0 to 2
	10%	Reliability	2 – Program functions properly under all test cases. Works for and responds to all valid inputs. 1 – Program functions under limited test cases. Only works for valid inputs, but fails to respond to invalid inputs. 0 – Program fails under most test cases.	0 to 2
	10%	Readability	2 – Program includes commented code, meaningful identifiers, indentation to clarify logical structure, and user instructions. 1 – Program lacks clear documentation and/or user instructions. 0 – Program is totally incoherent.	0 to 2

Semi-structured interview

The interviews collected the students' comments and feedback on the learning module by requiring students to write the responses on the paper. Each interview addressed two open-ended topics: (1) the strengths and weaknesses of the learning module and (2) suggestions for improving the learning module.

Results

Twenty-nine students completed the study. Most of them had an intermediate level of computer skills and a neutral intention to use online learning applications.

Questionnaire

Perceptions of the learning environment

The participants reported positive perceptions of the learning system in terms of its usefulness (Mean = 4.12, *SD* = .46), their intentions of using it (Mean = 4.22, *SD* = .54), and their overall satisfaction (Mean = 4.14, *SD* = .57). However, their perceptions of the ease of use of the system were weakly positive (Mean = 3.66, *SD* = .58). An internal consistency analysis using Cronbach's alpha confirmed that all of the subscales were reliable (.75 for usefulness, .74 for ease of use, .70 for intention to use, and .74 for overall satisfaction).

Motivation to learn programming

The participants reported having a clear motivation to learn programming using the proposed learning environment. The means were 4.09 (*SD* = .46) for interest/enjoyment, 4.39 (*SD* = .44) for value/usefulness, 3.89 (*SD* = .57) for effort/importance, 3.63 (*SD* = .55) for perceived competence, and 2.95 (*SD* = .86) for pressure/tension. As shown by the data, the students had a strong interest in learning using the proposed system and found the learning module highly useful. Moreover, they reported that they had made efforts in the learning task and felt competent in completing the project. Their perceived pressure during the learning process was slightly lower than neutral. An internal consistency analysis using Cronbach's alpha confirmed that all of the subscales were reliable (.76 for interest/enjoyment, .77 for effort/importance, .75 for value/usefulness, .79 for perceived competence, and .86 for pressure/tension).

Knowledge tests

The descriptive statistics for the knowledge-test scores and the paired sample t-test results for the difference between the pre-test and post-test are presented in Table 2. The students demonstrated significantly improved programming knowledge after completing the learning module.

Table 2. Descriptive statistics and paired sample t-test for knowledge achievement (*n* = 29)

Pre-test		Post-test		Paired sample *t*-test		
Mean	*SD*	Mean	*SD*	*df*	*t*	Sig. (two-tailed)
46.34	17.29	53.31	15.38	28	-3.780	.001***

Note. $^{***}p < .001$.

Programming tasks

Two domain experts assessed the students' performance on the programming tasks, and their scores were averaged. The two raters were blind to student identification and test information (i.e., whether the test was pre-test or post-test). The inter-rater reliability was computed using Cohen's kappa. The results exceeding .8 indicated perfect agreement, while .6 indicated substantial agreement. The results were .872 for problem understanding, .866 for modular design, .815 for process design, and .675 for coding (all significant at the .001 level), reflecting perfect agreement and consistency between the raters on the first three scales and substantial agreement on the last scale.

The descriptive statistics for the students' programming performance before and after the study and Wilcoxon signed-rank test for the difference between the two are presented in Table 3. The students showed significant progress on all of the programming performance scales.

Table 3. Descriptive statistics and Wilcoxon signed-rank test for programming performance (*n* = 29)

Programming performance	Pre-test		Post-test		Paired sample *t*-test	
	Mean	*SD*	Mean	*SD*	*Z*	Sig. (two-tailed)
Problem understanding	2.71	.921	3.03	.801	-2.140	.032*
Modular design	2.57	.904	3.345	.6139	-3.679	.000***
Process design	2.41	.814	3.293	.6199	-4.080	.000***
Coding	5.67	1.611	6.29	1.154	-2.676	.007**
Total score	13.36	3.076	15.97	2.787	-4.488	.000***

Note. $^{*}p < .05$; $^{**}p < .01$; $^{***}p < .001$.

Semi-structured interview

The comments shared by the students in their responses are presented in Table 4. During the interview, many students reported that the proposed learning module was highly useful in improving their programming thinking and problem-solving skills and in supporting their self-directed learning. Some of the students mentioned that they enjoyed the practical and meaningful learning experience and felt strongly motivated and interested in learning computer programming. Meanwhile, the students reported experiencing difficulties completing the learning tasks and suggested having more interactions with other students during the learning process. This is achievable not only through online forums but also through group-based learning with face-to-face communication. Some of the students suggested adding more learning resources and programming tasks to the system for self-study. The students also suggested improving some interfaces or functions of the system to make the system more attractive and easy to use.

Table 4. Students' responses to semi-structured interview questions

Students' comments	Count
Strengths of the learning module	
Improving my programming thinking	21
Supporting self-directed learning or improving my self-directed learning capability	21
Enhancing my programming problem-solving skills	16
Offering practical and meaningful learning experiences	8
Insightful and useful support afforded by the learning system	8
Activating my interest and motivation for learning programming	6

Weaknesses of the learning module	
Insufficient interactions with peers	16
Inadequate learning resources	9
Learning tasks are too difficult	8
Not comfortable with self-directed learning	7
Some interfaces or functions of the learning system are not attractive or easy to use	6
Suggestions for improvement	
Allowing more interactions between students, probably via group-based learning	20
Including more learning materials and programming tasks in the system	18
Adjusting the difficulty level of the learning projects	10
Making some interfaces or functions of the learning system more user-friendly	5

Discussion

RQ1: How can a visualization-based learning environment be designed to make the complex process of carrying out an authentic project accessible to learners in a programming course?

The proposed learning environment was designed by decomposing a complex project into a set of main actions, visualizing the process of the project and the process and/or structure of individual actions, and providing relevant guidance or strategies for individual actions to make the implicit aspects more accessible. This made the complex process of PjBL visible and accessible to learners and instructors, empowered the students to engage in deep learning and reflection, and enabled the instructors to observe individual performance and provide adaptive feedback and support throughout the project.

RQ2: What are the effects of the proposed approach on project-based learning in a programming course?

The results show the promising effects of the proposed learning environment as reflected by students' achievements in subject knowledge and programming performance and by their perceptions and motivations to learn.

First, the participants made significant improvements on programming task performance and subject knowledge after completing the learning module. Their progress in programming performance was significant in all aspects (i.e., problem understanding, modular design, process design, and coding).

Second, the students reported positive perceptions of the learning system in terms of its usefulness, their intentions of using it, and their overall satisfaction. However, their perceptions regarding the ease of use of the system were weakly positive. The participants also reported having a clear motivation to learn programming using the proposed learning environment. In particular, they enjoyed the learning experience and found it highly useful. They made efforts on the learning task and felt competent in completing the project, although they perceived some pressure during the learning process.

Third, the interview results confirm the students' perceptions and motivations to learn in addition to their comments on the strengths of the learning environment in facilitating self-directed learning. The students also provided suggestions for improving the learning module by enabling more peer interaction during the learning process, providing more online learning resources, and making the system more user-friendly. These issues will be addressed in a future study following the design-based research paradigm.

The findings align with those of related studies on PjBL (Blumenfeld et al., 2011) and visual representations in programming education (Sorva et al., 2013; Wang, Peng, Cheng, Zhou, & Liu, 2011). It should be noted that the visual representations used in previous studies have mainly targeted the learning of programming knowledge (e.g., basic concepts, syntax, and semantics) rather than programming skills and strategies, which are harder to capture yet are more critical for programming performance. Although specialized visual representations, such as functional block diagrams and flowcharts, are more applicable to learning programming skills and strategies (Blackwell, Whitley, Good, & Petre, 2001), it is not easy for students to use such tools to generate a satisfactory solution to a program. Research is needed to explore how visualization technology can effectively be integrated with programming learning and instruction (Rajala et al., 2008; Sorva et al., 2013). The findings of this study contribute to the efforts to address the concern by visualizing the complex process of performing an authentic programming project using relevant strategies.

Conclusion

PjBL is a promising approach to learning that connects abstract knowledge to authentic tasks. It has been increasingly promoted in educational practices, including computer programming education. Although written exams and program codes have been widely used for summative assessments in programming courses, grades may not reflect students' actual programming performance, as many students with good grades still have many problems completing actual programming tasks. PjBL has the potential to address this gap. Nevertheless, making the complex process of authentic projects accessible for effective learning and instruction is a pressing issue that makes it difficult to fully implement PjBL and to achieve the desired learning outcomes. Although research has shown the promising effects of visualization technology in programming education, the literature has reported inconsistent findings concerning how visualization technology can effectively be integrated with learning and instruction (Rajala et al., 2008; Sorva et al., 2013).

In an attempt to address this challenge, this study proposes the design of a visualization-based learning environment that externalizes the complex process of applying programming strategies to the design and development of solutions to an authentic programming project. It aims to make the complex process accessible, trackable, and attainable with the support of technology. In particular, it engages students in deep learning and reflection on the process and enable instructors to observe individual performance and provide adaptive feedback throughout the project. A learning environment focused on decomposing a complex project into a set of main actions, visualizing the project process and the process and/or structure of individual actions, and including relevant guidance or strategies for individual actions was designed. After using the proposed system to complete a PjBL module of computer programming, the students showed significant pre-post improvements in subject knowledge and programming performance. They also reported positive perceptions of the learning environment and favorable motivation and confidence to learn programming using the proposed approach. Following the design-based research paradigm, the proposed learning environment will be refined based on the results of the study, especially the comments and feedback of the participants. A control group design will be carefully implemented in a future study for further analysis.

The findings contribute to the knowledge of how complex PjBL can be adequately implemented through the effective design of technology-enhanced environments, particularly by making the complex aspects of PiBL visible and accessible for deep learning and reflection and by making the learning artifacts generated in the process available for timely analysis and feedback. Capturing and facilitating the complex process of carrying out an authentic project is a challenging issue in PjBL. The proposed visualization approach has the potential to gain holistic understandings of learner activity, cognitive processes, and performance in learning with authentic projects or complex tasks to improve learning and performance in such contexts. Although this study pertains to the domain of computer programming, the proposed approach has the potential to be transferable to PjBL in other disciplines.

Acknowledgements

This research is supported by the General Research Fund (Project No. 17201415) from the Hong Kong SAR Government, and the Seeding Fund for Basic Research (Project No. 201611159071) and the Faculty Research Travel Support for Full-time MPhil and PhD students from the University of Hong Kong. It was partially supported by the Greek General Secretariat for Research and Technology under the Matching Funds 2014-2016 (Project No. 325123). The authors would thank Professor Haijing Jiang for his valuable support for this project.

References

Adderley, K., Ashwin, C., Bradbury, P., Freeman, J., Goodlad, S., Greene, J., Jenkins, D., & Uren, O. (1975). *Project methods in higher education*. London, UK: Society for Research into Higher Education.

Alexander, E., Bresciani, S., & Eppler, M. J. (2015). Knowledge scaffolding visualizations: A Guiding framework. *Knowledge Management & E-Learning, 7*(2), 179–198.

Amiel, T., & Reeves, T. C. (2008). Design-based research and educational technology: Rethinking technology and the research agenda. *Educational Technology & Society, 11*(4), 29–40.

Arbaugh, J. B. (2000). Virtual classroom characteristics and student satisfaction with Internet-based MBA courses. *Journal of Management Education, 24*, 32-54.

Bassil, Y. (2012). A Simulation model for the waterfall software development life cycle. *International Journal of Engineering & Technology, 2*(5), 742-749.

Belland, B. R., Walker, A. E., Kim, N. J., & Lefler, M. (2016). Synthesizing results from empirical research on computer-based scaffolding in STEM education: A Meta-analysis. *Review of Educational Research*, *87*(2), 309-344.

Blackwell, A. F., Whitley, K. N., Good, J., & Petre, M. (2001). Cognitive factors in programming with diagrams. *Artificial Intelligence Review, 15*(1/2), 95-114.

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (2011). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist, 26*(3/4), 369-398.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18*, 32–42.

Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator, 15*(3), 6–11.

David, J. L. (2008). What research says about project-based learning. *Educational Leadership, 65*, 80-82.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly, 13*(3), 319-340.

Deek, F. P., Hiltz, S. R., Kimmel, H., & Rotter, N. (1999). Cognitive assessment of students' problem solving and program development skills. *Journal of Engineering Education, 88*(3), 317-326.

Deek, F. P., & McHugh, J. (2002). SOLVEIT: An Experimental environment for problem solving and program development. *Journal of Applied Systems Studies, 2*(2), 376-396.

Eisenberg, M., Basman, A., & Hsi, S. (2014). Math on a sphere: Making use of public displays in mathematics and programming education. *Knowledge Management & E-Learning, 6*(2), 140–155.

Gattiker, U. E., & Hlavka, A. (1992). Computer attitudes and learning performance: Issues for management education and training. *Journal of Organizational Behavior, 13*, 89-101.

Gijlers, H., & de Jong, T. (2013). Using concept maps to facilitate collaborative simulation-based inquiry learning. *Journal of the Learning Sciences, 22*(3), 340-374.

Govender, I., & Grayson, D. J. (2008). Pre-service and in-service teachers' experiences of learning to program in an object-oriented language. *Computers & Education, 51*(2), 874–885.

Helle, L., Tynjälä, P., & Olkinuora, E. (2006). Project-based learning in post-secondary education–theory, practice and rubber sling shots. *Higher Education, 51*(2), 287-314.

Hmelo-Silver, C. E., Duncan, R. G., & Chinn, C. A. (2007). Scaffolding and achievement in problem-based and inquiry learning: A Response to Kirschner, Sweller, and Clark (2006). *Educational Psychologist, 42*(2), 99–107.

Hundhausen, C. D., & Brown, J. L. (2007). What you see is what you code: A "live" algorithm development and visualization environment for novice learners. *Journal of Visual Languages & Computing, 18*(1), 22-47.

Jonassen, D. H. (2005). Tools for representing problems and the knowledge required to solve them. In S. O. Tergan, & T. Keller (Eds.), *Knowledge and Information Visualization* (pp. 82-94). Berlin, Germany: Springer.

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H., & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education, 10*(2), 109-128.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An Analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*(2), 75–86.

Koschke, R. (2003). Software visualization in software maintenance, reverse engineering, and re-engineering: A Research survey. *Journal of Software Maintenance and Evolution: Research and Practice, 15*(2), 87–109.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A Study of the difficulties of novice programmers. In *Proceedings of ACM SIGCSE Bulletin* (Vol. 37, pp. 14-18). New York, NY: ACM.

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. New York, NY: Cambridge University Press.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., & Velázquez-Iturbide, J. A. (2003). Exploring the role of visualization and engagement in computer science education. In *Proceedings of ACM SIGCSE Bulletin* (Vol. 35, No. 2, pp. 131-152). New York, NY: ACM.

Nuutila, E., Törmä, S., & Malmi, L. (2005). PBL and computer programming — The Seven steps method with adaptations. *Computer Science Education, 15*(2), 123-142.

Perrenet, J. C., Bouhuijs, P. A. J., & Smits, J. G. M. M. (2000). The Suitability of problem-based learning for engineering education: Theory and practice. *Teaching in Higher Education, 5*(3), 345-358.

Pucher, R., & Lehner, M. (2011). Project based learning in computer science–A Review of more than 500 projects. *Procedia-Social and Behavioral Sciences, 29*, 1561-1566.

Rajala, T., Laakso, M. J., Kaila, E., & Salakoski, T. (2008). Effectiveness of program visualization: A Case study with the ViLLE tool. *Journal of Information Technology Education, 7*, 15-32.

Reiser, B. J. (2004). Scaffolding complex learning: The Mechanisms of structuring and problematizing student work. *Journal of the Learning Sciences, 13*(3), 273–304.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A Review and discussion. *Computer Science Education, 13*(2), 137-172.

Roels, R., Meştereagă, P., & Signer, B. (2016). An Interactive source code visualisation plug-in for the MindXpres presentation platform. In *Proceedings of the International Conference on Computer Supported Education (CSEDU 2015)* (pp. 169-188). doi:10.1007/978-3-319-29585-5_10

Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist, 55*, 68–78.

Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 850-858.

Sorva, J., Karavirta, V., & Malmi, L. (2013). A Review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education, 13*(4), 15. doi:10.1145/2490822

Suthers, D. D., Vatrapu, R., Medina, R., Joseph, S., & Dwyer, N. (2008). Beyond threaded discussion: Representational guidance in asynchronous collaborative learning environments. *Computers & Education, 50*(4), 1103-1127.

Thomas, J. W. (2000). *A Review of research on project-based learning*. San Rafael, CA: Autodesk Foundation.

van Merriënboer, J. J. G., & Kirshner, P. (2007). *Ten steps to complex learning: A Systematic approach to four-component instructional design*. Mahwah, NJ: Lawrence Erlbaum Associates.

Wang, M., Kirschner, P. A., & Bridges, S. M. (2016). Computer-based learning environments for deep learning in inquiry and problem-solving contexts. In *Proceedings of the 12th International Conference of the Learning Sciences* (pp. 1356-1360). Singapore: International Society of the Learning Sciences.

Wang, M., Peng, J., Cheng, B., Zhou, H., & Liu, J. (2011). Knowledge visualization for self-regulated learning. *Educational Technology & Society, 14*(3), 28–42.

Wang, M., Vogel, D., & Ran, W. (2011). Creating a performance-oriented e-learning environment: A Design science approach. *Information & Management, 48*(7), 260–269.

Wang, M., Wu, B., Kinshuk, Chen, N. S., & Spector, J. M. (2013). Connecting problem-solving and knowledge-construction processes in a visualization-based learning environment. *Computers & Education, 68*, 293-306.

Wu, B., & Wang, M. (2012). Integrating problem solving and knowledge construction through dual mapping. *Knowledge Management & E-Learning, 4*(3), 248–257.