# Using a visualisation-based and progressive learning environment as a cognitive tool for learning computer programming

**Jun Peng**
School of Education, City University of Macau, China

**Minhong Wang**
KM&EL Lab, Faculty of Education, University of Hong Kong, China
Department of Educational Information Technology, East China Normal University, China

**Demetrios Sampson**
Department of Digital Systems, University of Piraeus, Greece; School of Education, Curtin University, Australia

**Jeroen J. G. van Merriënboer**
School of Health Professions Education, Maastricht University, The Netherlands

Project-based learning (PjBL) has been widely promoted in educational practice, for example, computer programming education. While PiBL may help learners to connect abstract knowledge with authentic practice, the complexity of completing an authentic project may overwhelm learners, making them unable to achieve the desired learning outcomes. This study proposes a visualisation-based and progressive learning environment as a cognitive tool to support PjBL of programming. The cognitive tool is designed to externalise the complex process of completing a realistic programming project. It aims to scaffold the complex project process, foster effective thinking and reflection, and allow the teacher to track and give feedback on individual performance throughout the project process. Moreover, simple-to-complex sequencing of whole-task projects is incorporated into the cognitive tool to support progressive learning with authentic projects. Senior college students participated in this study by completing a project-based programming learning module using the proposed cognitive tool. The results showed that after incorporating the simple-to-complex approach, the visualisation-based cognitive tool is more effective in improving students' programming performance and better perceived by students in terms of its support for scaffolding and articulating the complex project process.

## Introduction

Project-based learning (PjBL) is a student-centred pedagogy that encourages students to learn by working with authentic whole-tasks or projects. 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 distinguishes itself from a related pedagogy called problem-based learning (PBL) by placing more attention on the development of realistic products closer to professional reality and on the assessment of product quality. By creating artefacts as solutions to real-world problems, PjBL helps students to connect abstract knowledge with real-world practice (Blumenfeld et al., 2011).

PjBL has been widely promoted in educational practice especially in senior-year curricula in higher education. One big concern in higher education is about the graduates' ability to apply knowledge to solve industry problems. As reported in prior studies, competency gaps were found between graduates' professional attributes and the expectations of their employees in areas such as problem-solving and communication skills (Jollands, Jolly, & Molyneaux, 2012). Researchers also discussed the relationship between school learning and workplace learning and transitioning graduates from novices to experts through practice (Tynjälä, 2008; Wang, Yuan, Kirschner, Kushniruk, & Peng, 2018). In these studies, PjBL was recognised as a promising approach to addressing the gap between knowledge and practice and to pushing professional readiness.

Nevertheless, PjBL is much more easily advocated than accomplished. Completing an authentic project often involves complex cognitive processes, which are difficult for learners to capture and for teachers to

facilitate. On the other hand, research reveals the advantages of computer-based cognitive tools in allowing people to construct, recall, and modify their understanding of complex issues by reflecting complex cognition on the screen (Jonassen, 1996). While computer-based cognitive tools have the potential to facilitate complex learning with authentic projects (Peng, Wang, & Sampson, 2017), there is inadequate research examining whether and how PjBL can be supported by computer-based cognitive tools. To this end, this study explored the design and effects of a visualisation-based learning environment as a computer-based cognitive tool to support PjBL.

Computer programming, a learning subject in engineering education, was selected for this study. PjBL is particularly suitable for engineering education because almost every task undertaken in professional practice by an engineer is related to a project. Computer programming is an important subject in engineering education. It is also considered a hard subject to learn. A programmer needs to master programming knowledge (e.g., concepts, syntax, semantics) and apply it to programming tasks. The strategies and skills for applying programming to realistic tasks are often implicit and hard to capture, yet are critical for programming performance (Robins, Rountree, & Rountree, 2003; Soloway, 1986). PjBL has therefore been increasingly promoted in programming education by encouraging students to work with realistic programming projects and developing artefacts, such as computer programs or design plans, which are realistic products closer to professional reality (Blumenfeld et al., 2011). However, the implementation of PjBL in programming courses remains a struggle for many educators to facilitate the complex process of analysis, design, and development of computer programs (Pucher & Lehner, 2011). Such complexity can overwhelm learners, making them unable to engage in effective learning experiences and achieve the desired learning outcomes (Helle, Tynjälä, & Olkinuora, 2006; Pucher & Lehner, 2011).

This study aimed to address the challenge of PjBL in programming education by (a) proposing a visualisation-based cognitive tool for PjBL and (b) applying a simple-to-complex progressive approach to learning with complex projects. Firstly, a visualisation-based learning environment was proposed by externalising the complex process of completing a realistic programming project in visual formats. It aimed to scaffold complex learning, foster effective thinking and reflection, and enable the teacher to track and give feedback on individual performance. Secondly, considering that completing a realistic whole-task project might be too challenging for novices initially, the simple-to-complex sequencing of whole-task projects was incorporated into the learning environment. A realistic project was deconstructed into a set of sub-projects arrayed in a simple-to-complex order; each sub-project comprised all central elements of a project.

This study may contribute to knowledge of how effective learning with complex authentic projects can be realised through a visualisation-based and progressive learning environment as a computer-based cognitive tool. While research has shown the promising effects of visualisation-based cognitive tools on improving learning with authentic projects by externalising the complex cognitive processes (Peng et al., 2017), novices may have problems in completing an authentic project even though its complex process has been visualised. This study explored whether and how learning with complex projects can be supported not only by visualising the cognitive processes, but also by simple-to-complex sequencing of whole-task projects. Both are important elements of a computer-based cognitive tool for learning with complex authentic projects.

## Literature review

### Scaffolding learning with complex tasks

Learning by working with a realistic project is characterised by performing a complex task or solving a sophisticated problem, which usually involves complex, implicit processes. The complexity of the process may generate heavy cognitive loads for learners, making them unable to achieve the desired learning outcomes (Kirschner, Sweller, & Clark, 2006). Providing learners with a scaffold or necessary support is important, if not essential, to learning with complex tasks or problems (Belland, Walker, Kim, & Lefler, 2016; Hmelo-Silver, Duncan, & Chinn, 2007). The commonly used approaches to scaffolding complex learning involve structuring a complex task into a set of main actions or using key questions to help learners recognise the important goals to pursue in their task (Reiser, 2004), in addition to using prompts to bring learners' attention to the important issues of complex tasks (Ge & Land, 2003).

Scaffolding learning with complex tasks echoes to a certain extent the cognitive apprenticeship model, which claims that performing a complex task involves implicit processes, and it is critical to make such processes visible for novices to observe, enact, and practise with the necessary help (Collins, Brown, & Holum, 1991). The cognitive apprenticeship model suggests a set of cognitive strategies (namely *exploration*, *scaffolding*, *modelling*, *coaching, articulation*, and *reflection*) to support learning in complex task situations. Scaffolding and articulation focus on the externalisation of complex processes; modelling and coaching highlight the provision of instructional support and feedback; exploration and reflection encourage learners to work with realistic problems or tasks and reflect on the task experience.

### Computer-based visual representations as cognitive tools

To support the externalisation of complex processes in learning with complex tasks, visual representations have been increasingly employed to represent the complex cognitive process in visual forms. Visual representations like diagrams, maps, tables, and pictures, if used appropriately, can reduce human cognitive load by utilising the human brain's capacity to rapidly process visual images (Scaife & Rogers, 1996) and by meaningful representation of complex ideas (e.g., representing information verbally and spatially, reducing ambiguous expression, grouping together relevant information). They can work as cognitive tools to extend mental capability and afford efficient cognitive processing.

In recent decades, computer-based visual representation tools such as concept maps, procedural flowcharts, causal maps, and integrated cognitive maps have been increasingly applied to educational practices and incorporated in computer-based learning environments to foster higher-order thinking and self-directed learning (Lajoie & Derry, 1993; Lee, Pradhan, & Dalgarno, 2008; Spector & Anderson, 2000; Wang, Derry, & Ge, 2017). Such visualisation-based cognitive tools (Jonassen, Carr, & Yueh, 1998) can help represent complex, abstract issues and processes that are difficult to convey in traditional formats. Research has shown the promising effects of such tools in improving students' knowledge and task performance in various contexts (Gijlers & de Jong, 2013; Slof, Erkens, Kirschner, Janssen, & Jaspers, 2012; Suthers, Vatrapu, Medina, Joseph, & Dwyer, 2008; Wang, Cheng, Chen, Mercer, & Kirschner, 2017; Wang, Wu, Kirschner, & Spector, 2018).

In programming education, visual representations (e.g., diagrams, pictures, animations) with relevant tools have been used to visualise the complex structures and algorithms of software programs and demonstrate the run-time behaviour of programs (Koschke, 2003; Sorva, Karavirta, & Malmi, 2013). For example, the instructor using the tool may start by demonstrating a program segment with learned elements, and then make some change (e.g., adding a *for loop*) to the segment to introduce a new language element to students. During the process, the added code lines are highlighted, and the output is visually presented to show how the new element is used to achieve the goal (Hooper et al., 2007). Other tools, such as BlueJ, can visualise the class structure to help students understand classes and objects and their relationships, which are important issues in object-oriented programming, but difficult to explain to students (Kölling, Quig, Patterson, & Rosenberg, 2003). Visualising the class structure enables students to see and interact with objects before being confronted with syntax details that bother students most. These approaches have been mainly used to help students to understand the abstract concepts and complicated behaviour of programs and to support the coding process. They are effective in introductory programming courses and predefined programming problems, but inadequate for ill-defined realistic programming projects (Sykes, 2007). The literature has reported the promising advantages of these approaches in engaging programming learners, but their effect on improving learners' programming performance is inconclusive (Rajala, Laakso, Kaila, & Salakoski, 2008; Sorva et al., 2013). In this study, the visualisation-based cognitive tool was designed to externalise the complex cognitive process of completing a realistic programming project, which involves not only coding and debugging but also other stages such as problem formulation, solution planning, and solution design. Visualisation of such kind of mental images for problem-solving is crucial to programming (Gómez-Albarrán, 2005), especially for ill-defined realistic programming projects.

### Simple-to-complex sequencing of whole-tasks for learning with complex tasks

Completing a realistic task or project might be too challenging for novices at the initial stage. They may have problems in completing the project even though the process has been demonstrated or externalised in visible forms. In relation to this issue, the four-component instructional design (4C/ID) model presents a framework for systematic learning with complex tasks (van Merriënboer & Kirschner, 2017). In the 4C/ID

model, learners are asked to perform meaningful whole tasks in authentic situations, where the tasks comprise all key aspects of the complete task. In addition to providing learners with procedural information and supportive guidance to complete complex tasks, authentic whole-tasks need to be organised in a simple-to-complex order. Initial complex tasks can focus on the most fundamental and central elements of the whole task, to help students form a holistic view of the complex task's skeleton that could be enriched by later learning tasks (van Merrienboer & Sweller, 2005). The simple-to-complex progressive learning approach has been empirically explored and has shown its great potential in complex learning domains, including programming learning in high-school classrooms and computer-based learning environments (Marcellis, Barendsen, & van Merrienboer, 2018; van Merriënboer, 1990; van Merriënboer & De Croock, 1992).

## Research questions

This study adopted a pre- and post-test control group design. Both experimental and control groups were asked to complete a PjBL module of ASP.NET. ASP.NET is a popular programming language and a widely-used web application framework for developing dynamic modern web applications and services. The PjBL module was designed for students to develop senior-level programming skills by working on realistic programming projects after they have completed basic computer programming courses. Students were expected to apply basic programming knowledge and skills to create program artefacts as solutions to ill-defined real-world problems.

To achieve the learning goal, students needed instructional support that allows them to capture the complex cognitive process of completing a realistic programming project. Such support was offered to students in both experimental and control groups via using the visualisation-based learning environment. Considering the possible need for simple-to-complex sequencing of whole-task projects, students in the experimental group were asked to work with a realistic project in a progressive way, while those in the control group worked with the same project in a non-progressive way.

The research questions (RQs) were specified as follows:

- RQ1. Is the visualisation-based cognitive tool for PjBL of programming effective for improving students' programming performance after the study?

To answer this question, students' pre-test and post-test scores of programming performance before and after the study were analysed to examine the pre-post difference.

- RQ2. Will the incorporation of the simple-to-complex progressive learning approach influence the effects of the visualisation-based learning environment for PjBL of programming (as reflected in students' programming performance and their perceptions of the learning environment)?

To answer this question, students' perceptions of the learning environment (in terms of its cognitive strategies for support of learning with complex tasks) were collected and compared between those using the progressive approach and others using the non-progressive approach. The programming performance was also analysed and compared between the two groups of students.

## Method

The study received the ethical approval from the Human Research Ethics Committee of the researchers' university. The participants gave informed consent to participate in this study.

### Participants

There were 69 year-three undergraduates (44 males and 25 females) in computer science participating in this study. They had completed basic computer programming courses before the study. Each student was randomly assigned to either the experimental condition using the progressive learning approach or to the control condition using the non-progressive learning approach. All the 34 students (20 males and 14 females) assigned to the experimental group completed the entire study. Among the 35 students assigned

to the control group, 33 of them (22 males and 11 females) completed the entire study, and other 2 failed to take the post-test. The data of 69 students completing the entire study were used for analysis.

### Visualisation-based learning environment

In this study, the visualisation-based cognitive tool was designed in response to the aforementioned need for externalising the complex cognitive process of completing a realistic programming project, with particular attention to problem formulation, solution planning, and solution design, rather than coding and debugging only. Based on the literature and practice of computer programming (Bassil, 2012; Deek & McHugh, 2002), the cognitive process of completing a programming project was visualised as a set of key actions (listed below). The tacit knowledge or key strategies underlying these actions were also highlighted. Visualisation of such kind of mental images for problem-solving is crucial to computer programming (Gómez-Albarrán, 2005), especially for ill-defined realistic programming projects (Peng et al., 2017). It can scaffold student learning with a complex project, foster effective thinking and reflection during the project, and enable the teacher to track and give feedback on individual performance.

- Problem understanding (to formulate a problem). The first step of a programming project is to formulate a clear understanding of the problem. The problem understanding should highlight the requirements and goals of the project. Learners can present their understanding by specifying the requirements and goals of the project in a structured form.
- Modular design (to design a plan of the solution). A computer program is often organised 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. The modular design strategies highlight the independence and completeness of the modules. A diagramming tool is offered for learners to build a functional block diagram to outline the plan of the solution.
- Process design (to design a detailed solution). 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. The process design strategies focus on priority analysis and critical analysis when designing a complex flowchart involving a number of interactive modules. Learners can use the diagramming tool to build a flowchart, which demonstrates a detailed design of the solution.
- Coding (to implement a solution). 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. The coding strategies focus on top-down gradual refinement in addition to the data structures and algorithms.
- Evaluation and reflection. After completing their codes, learners need to evaluate their programs by testing and debugging them. Moreover, they can reflect on their performance and areas for possible improvement by reviewing the artefacts generated in each action along with the comments and feedback from the teacher. They can update their artefacts or solutions and receive further feedback.

As shown in Figure 1, the process of completing a programming project is externalised in a visual format. By clicking on the icon of each action, learners can enter the action space, view the key strategies underlying the action, and present the output of the action for effective thinking and reflection.


*Figure 1.* Visualisation-based learning environment

As shown in Figure 2, a diagramming tool is provided for learners to build a functional block diagram to outline the plan of the solution. Moreover, learners can use the diagramming tool to build a flowchart for the software program, which demonstrates a detailed design of the solution.

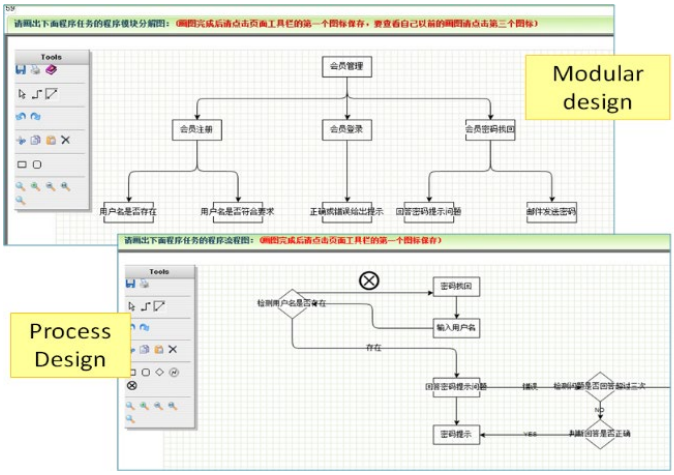


*Figure 2.* Modular design and process design

After completing an action, learners can review and refine their outputs. Moreover, the teacher can use the system to observe students' artefacts and provide feedback to individuals by giving specific comments on their outputs on problem statement, modular design, program flowchart, and program code, as shown in Figure 3.

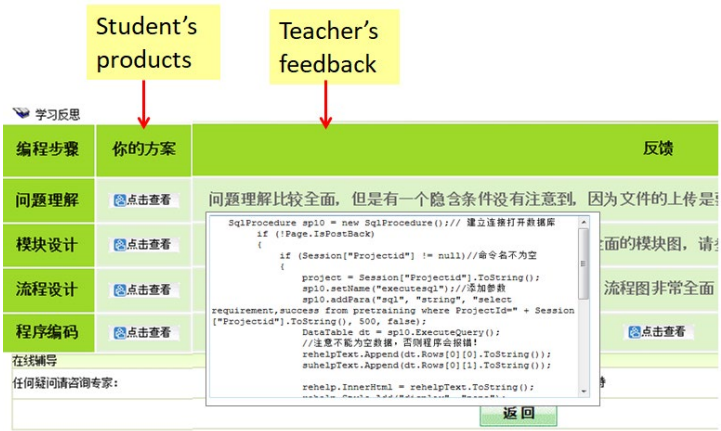


*Figure 3.* Reflection with feedback

The overall design of the learning environment was aligned with the six strategies proposed in the cognitive apprenticeship model, namely exploration, scaffolding, modelling, coaching, articulation, and reflection.

- Exploration: providing learners with opportunities to work with realistic projects
- Scaffolding: making the complex process accessible to learners by visualising the process of completing a programming project into a set of main actions (problem understanding, modular design, process design, coding, and evaluation and reflection) and externalising the key strategies or tacit knowledge underlying the actions
- Modelling: the teacher's demonstration of the process of completing a sample project using the system
- Coaching: providing feedback on individual performance by the teacher via the system
- Articulation: enabling learners to present their project process in visible forms
- Reflection: enabling learners to review and reflect on their project process and identify the gap in their knowledge and performance.

## Learning task

During the study, students in both groups used the proposed visualisation-based cognitive tool to work on a realistic programming project – *membership management*. The control group worked with the project in a non-progressive way, that is, without deconstructing the project. The experimental group worked with the project in a progressive way. According to the 4C/ID model (van Merriënboer & Kirschner, 2017), authentic whole-tasks need to be organised in a simple-to-complex order; initial complex tasks can focus on the most fundamental and central elements of the whole task, helping students to form a holistic view of the complex task's skeleton that could be enriched by later learning tasks. In this study, the membership management project was deconstructed into a set of sub-projects arrayed in a simple-to-complex order. As outlined in Table 1, the requirement of each sub-project was based on the function of the prior sub-project, with extended functions in the variant. For each sub-project, students need to go through all key aspects of the task – problem understanding, modular design, process design, coding, and evaluation and reflection. In this way, students worked with a realistic whole project in a progressive way, instead of being exposed to its full complexity from the outset.

Table 1
*Learning with a programming project in a progressive and non-progressive way*

| Learning approach | Project | Main requirement |
|---|---|---|
| Progressive | Sub-project 1 | Develop a program that allows for member registration and user login |
| | Sub-project 2 | Add functions for login validation and update of member information |
| | Sub-project 3 | Add further functions for password setting and resetting |
| Non-progressive | Project | Develop an integrated program that includes all the functions mentioned above |

To complete a project or sub-project, students went through the main actions – problem understanding, modular design, process design, coding, and evaluation and reflection. They completed each action by submitting relevant learning artefacts (i.e., problem statement, modular design, program flowchart, and code). During the project, the teacher, who was also a programming expert, reviewed the students' learning artefacts and provided comments on individual performance via the learning system. Students could view the teacher's comments for improvement and work on a project more than once for practice. They could also use online forums for flexible discussion and communication among peers in the same group.

## Measures and instruments

*Programming performance*
Students' programming performance concerning the programming process and programming product was assessed before and after the study using a programming task. An experienced programming teacher and a programming expert worked together to design two programming tasks, one for the pre-test and the other for the post-test. The two tasks were different in content but at the same level of difficulty as validated by the domain experts. 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. The two tasks were randomly assigned to the two tests.

The assessment of students' programming performance was informed by the programming assessment model proposed by Deek, Hiltz, Kimmel, and Rotter (1999), which consists of two distinct categories: programming process and programming product. The programming process was measured by three sub-scales: 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 of programming performance in this study consisted of four items: problem understanding, modular design, process design, and coding. Based on the literature and common practice, the rubric weighting was 40% for coding and 20% for each of the other items. The assessment rubrics with respect to the description, weight, rating criteria, and score range of each item are outlined in Table 2. The full programming performance score was 20 (8 points for coding and 4 points for each of the three other aspects).

Table 2
*Assessment rubrics for programming performance*

| Category | Item | Weight | Description | | Score range |
|---|---|---|---|---|---|
| Programming process | Problem understanding | 20% | 4 – Problem is clearly and correctly stated. All requirements and goals are identified.<br>3 – Problem is correctly stated. Most requirements and goals are identified.<br>2 – Problem is partially stated and/or some facts are identified.<br>1 – Problem statement is incorrect and meaningless facts are identified.<br>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.<br>3 – Adequate planning, with sufficient goal refinement and task identification.<br>2 – Partially correct planning, with some goal refinement and task identification.<br>1 – Incorrect planning and meaningless goal refinement.<br>0 – No planning/refinement attempted or completely irrelevant work. | | 0 to 4 |
| | Process design | 20% | 4 – Complete module decomposition, organisation, and detailed specifications.<br>3 – Sufficient module decomposition, organisation, and sufficient specifications.<br>2 – Partial design and/or some module specifications.<br>1 – Improper module decomposition, organisation, and specifications.<br>0 – No design/specifications attempted or completely irrelevant work. | | 0 to 4 |
| Programming product | Coding | 10% | Correctness | 2 – Correct solution specifications/program code and results consistent with problem requirements.<br>1 – Partial solution specifications/program code and/or some results<br>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.<br>1 – Program accomplishes its task, but lacks coherence in choice of either data and/or control structures.<br>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.<br>1 – Program functions under limited test cases. Only works for valid inputs, but fails to respond to invalid inputs.<br>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.<br>1 – Program lacks clear documentation and/or user instructions.<br>0 – Program is totally incoherent. | 0 to 2 |

Two programming experts graded the students' programming tests blindly and independently, and their scores were averaged. The inter-rater reliability analysis using Cohen's kappa coefficients confirmed the inter-rater reliability for students' programming tests (.86 for problem understanding, .83 for modular design, .88 for process design, .89 for coding).

*Learner perceptions*

The post-test questionnaire was used to collect students' perceptions of the learning environment in terms of its cognitive strategies for support of learning with complex tasks. It used a 5-point Likert scale ranging from 1 (*strongly disagree*) to 5 (*strongly agree*). Fifteen items were developed from the instrument proposed by Stalmeijer, Dolmans, Wolfhagen, Muijtjens, and Scherpbier (2008) on the basis of the cognitive apprenticeship model. They involve six sub-scales of cognitive strategies: exploration (working with realistic problems or tasks), scaffolding (providing support to learners for the tasks that learners are unable to complete without help), modelling (providing learners with examples or models of desired performance), articulation (learners' articulation of their thinking and understanding), coaching (observing learners' performance and offering feedback), and reflection (learners' reflection on their performance and comparison with the performance of others).

The validity and reliability of the instrument have been well established (Stalmeijer et al., 2008, 2010). Examples of the questions in the survey included: "The learning system helped me to complete a task that is beyond my level of competence," "The learning system allowed me to articulate my task process," "The learning system provided me useful feedback during the study," and "The learning system stimulated me to thinking about how to improve my task performance." In this study, an internal consistency analysis using Cronbach's alpha coefficients confirmed that all of the sub-scales were reliable (.82 for exploration, .70 for scaffolding, .77 for modelling, .84 for articulation, .74 for coaching, .84 for reflection).

## Procedure

The learning module lasted for 6 weeks. In the first week, the participants were given 30-minute face-to-face instruction on how to use the visualisation-based learning environment and how to apply relevant programming skills and strategies to perform each action in completing a programming project. Relevant information and guidance were also available in the system for flexible access. During the instruction, a sample project was used for demonstration by the teacher. Students could use the sample project to practise and become familiar with the learning environment. In addition, a programming task was used to assess their programming performance before the study.

The students started their independent learning in the second week. They were asked to complete the membership management project in their free time over a 4-week period. They were asked to pace themselves and spend 4 hours per week on the project. Based on the log data, most of them spent approximately 3 hours per week with the system. For each project, most of the students received two or three comments on problem understanding and one or two comments on each of the other parts, including

modular design, process design, and coding. In the sixth week, a questionnaire survey was administered to collect students' perceptions, and the post-test of programming performance was arranged to assess their programming performance after the study.

### Data analysis

The collected data were analysed using the following methods:

(1) Paired samples *t* tests were used to determine whether there were differences between the pre-test and post-test scores of all participants.
(2) Independent samples *t* tests were conducted on the pre-test scores to identify whether students in the two groups differed in their programming performance before the study.
(3) One-way analysis of covariance (ANCOVA) was used to examine group differences in the post-test scores, whereby the students' pre-test score was used as covariate.
(4) ANOVA was conducted on the survey data to determine the differences in student perceptions of the learning environment between the two groups.
(5) A two-way ANOVA was conducted on programming performance to examine whether there was an interaction between the two factors (learning time (pre-test vs. post-test) as a within-subject factor; group condition (progressive vs. non-progressive) as a between-subject factor).
(6) Cohen's *d* effect size was calculated for the effects on programming performance.

## Results

### RQ1. Is the visualisation-based cognitive tool for PjBL of programming effective for improving students' programming performance after the study?

As shown in Table 3, the participants made a significant pre-post improvement on all scales of the programming performance (problem understanding, modular design, process design, coding). The effect size (Cohen's $d$ = 0.97 for problem understanding, 0.97 for modular design, 1.15 for process design, 1.27 for coding, 1.29 for total score) indicated a large effect of the visualisation-based learning environment in improving students' programming performance after the study.

Table 3
*Descriptive statistics and t tests on pre-test and post-test scores of programming performance*

| Aspect | Test | *N* | Mean | *SD* | *t* | *df* | *p* | Cohen's *d* |
|---|---|---|---|---|---|---|---|---|
| Problem understanding | a | 67 | 14.142 | 4.187 | -5.627 | 123.359 | 0.000*** | -0.97 |
| | b | 67 | 17.761 | 3.193 | | | | |
| Modular design | a | 67 | 13.769 | 3.803 | -5.527 | 127.103 | 0.000*** | -0.97 |
| | b | 67 | 17.149 | 3.117 | | | | |
| Process design | a | 67 | 13.396 | 3.446 | -6.673 | 132 | 0.000*** | -1.15 |
| | b | 67 | 17.313 | 3.349 | | | | |
| Coding | a | 67 | 19.030 | 5.589 | -7.322 | 115.451 | 0.000*** | -1.27 |
| | b | 67 | 28.000 | 8.325 | | | | |
| Total | a | 67 | 60.336 | 15.069 | -7.466 | 132 | 0.000*** | -1.29 |
| | b | 67 | 80.224 | 15.758 | | | | |

*Notes.* a = pre-test. b = post-test. *** $p < 0.001$

**RQ2. Will the incorporation of the simple-to-complex progressive learning approach influence the effects to the visualisation-based learning environment for PjBL of programming (as reflected in students' programming performance and their perceptions of the learning environment)?**

*Programming performance*

As shown in Table 4, there were no significant differences between the experimental and control groups in the pre-test scores of programming performance. The results indicated that the two groups had comparable programming performance before the study.

Table 4

*Descriptive statistics and t tests on the pre-test scores of programming performance of both groups*

| Aspect | Condition | *N* | Mean | *SD* | *t* | *df* | *p* |
|---|---|---|---|---|---|---|---|
| Problem understanding | E | 34 | 13.82 | 4.09 | -0.629 | 65 | 0.532 |
| | C | 33 | 14.47 | 4.32 | | | |
| Modular design | E | 34 | 13.53 | 3.59 | -0.520 | 65 | 0.605 |
| | C | 33 | 14.02 | 4.05 | | | |
| Process design | E | 34 | 13.38 | 3.42 | -0.032 | 65 | 0.975 |
| | C | 33 | 13.41 | 3.53 | | | |
| Coding | E | 34 | 19.12 | 5.70 | 0.130 | 65 | 0.897 |
| | C | 33 | 18.94 | 5.56 | | | |
| Total | E | 34 | 59.85 | 14.27 | -0.264 | 65 | 0.792 |
| | C | 33 | 60.83 | 16.05 | | | |

*Notes*. E = Experimental group. C = Control group.

Based on the ANCOVA results shown in Table 5, students in the experimental group outperformed those in the control group in all scales of programming performance. The large effect size (Cohen's $d = 1.01$ for problem understanding, 0.78 for modular design, 0.73 for process design, 0.86 for coding, 0.09 for total score) indicated that the progressive learning approach made the visualisation-based learning environment more effective in improving students' programming performance after the study.

Table 5

*Descriptive statistics and ANCOVA on programming performance of both groups after the study*

| Aspect | Condition | *N* | Mean | *SD* | Adjusted mean | *F*(1, 65) | *p* | Cohen's *d* |
|---|---|---|---|---|---|---|---|---|
| Problem understanding | E | 34 | 19.19 | 1.56 | 19.25 | 20.566 | 0.000*** | 1.01 |
| | C | 33 | 16.29 | 3.76 | 16.23 | | | |
| Modular design | E | 34 | 18.27 | 1.98 | 18.33 | 12.024 | 0.001** | 0.78 |
| | C | 33 | 15.99 | 3.64 | 15.94 | | | |
| Process design | E | 34 | 18.46 | 2.53 | 18.46 | 8.872 | 0.004** | 0.73 |
| | C | 33 | 16.14 | 3.70 | 16.14 | | | |
| Coding | E | 34 | 31.28 | 3.82 | 31.26 | 12.691 | 0.001** | 0.86 |
| | C | 33 | 24.62 | 10.23 | 24.65 | | | |
| Total | E | 34 | 87.20 | 5.29 | 87.32 | 18.481 | 0.000*** | 0.99 |
| | C | 33 | 73.04 | 19.45 | 72.91 | | | |

*Notes*. E = Experimental group. C = Control group. *** $p < 0.001$; ** $p < 0.01$

The two-way ANOVA results showed that students in the experimental group outperformed their counterparts in the overall programming performance ($F(1, 130) = 6.753$, $p < 0.05$), that is, group condition influenced the performance. In addition, students in both groups had improved their programming performance after the study; in other words, learning time had an effect on the programming performance of both groups ($F(1, 130) = 60.806$, $p < 0.001$). The interaction between learning time and group condition reached statistical significance ($F(1, 130) = 8.912$, $p < 0.01$), suggesting that the effects of the progressive learning approach interacted with the effects of learning time.

*Learner perceptions*

As shown in Table 5, both groups reported having positive perceptions of the learning environment in terms of its cognitive strategies that support the learning with complex tasks. The experimental group recorded higher scores than the control group on two scales – scaffolding and articulation, but no significant differences in the other four scales – modelling, coaching, reflection, and exploration.

Table 6

*Descriptive statistics and ANOVA on students' perceptions of the learning environment*

| | Condition | *N* | Mean | *SD* | *p* value of Levene's test | *F* | *p* |
|---|---|---|---|---|---|---|---|
| Modelling | E | 34 | 4.17 | 0.59 | 0.250 | 0.120 | 0.730 |
| | C | 33 | 4.12 | 0.47 | | | |
| Coaching | E | 34 | 4.25 | 0.62 | 0.742 | 0.132 | 0.718 |
| | C | 33 | 4.19 | 0.58 | | | |
| Scaffolding | E | 34 | 4.24 | 0.57 | 0.337 | 4.448 | 0.039* |
| | C | 33 | 3.92 | 0.64 | | | |
| Articulation | E | 34 | 4.37 | 0.59 | 0.320 | 5.781 | 0.019* |
| | C | 33 | 4.02 | 0.61 | | | |
| Reflection | E | 34 | 4.25 | 0.61 | 0.800 | 0.525 | 0.471 |
| | C | 33 | 4.14 | 0.68 | | | |
| Exploration | E | 34 | 4.24 | 0.68 | 0.444 | 0.008 | 0.930 |
| | C | 33 | 4.22 | 0.51 | | | |

*Notes*. E = Experimental group. C = Control group. * $p < 0.05$

## Discussion

This study proposed a visualisation-based cognitive tool that externalised the complex process of completing a realistic programming project in visual formats. It aimed to scaffold students' learning with projects, facilitate their thinking and reflection, and enable the teacher to track and give feedback on student performance during the project. Considering that completing a realistic whole-task project might be too challenging for novices initially, the simple-to-complex sequencing of whole-task projects was incorporated into the learning environment. It allowed students to start from relatively simple but authentic whole-tasks and then progress to practice that increasingly approximates the reality of professional practice.

The proposed visualisation-based progressive learning environment as a computer-based cognitive tool for project-based learning extends the literature in two aspects. Firstly, while existing visualisation-based tools for programming education have focused on helping students to understand the abstract concepts and complicated behaviour of programs and supporting the coding process, the visualisation-based cognitive tool proposed in this study focuses on externalising the complex cognitive process of completing a realistic programming project, which includes not only coding and debugging but also problem formulation, solution planning, and solution design. Visualisation of such kind of mental images for problem solving is crucial to programming, especially for ill-defined realistic programming projects (Peng et al., 2017). Secondly, this study demonstrates that learning with complex projects may need instructional support in multiple aspects. In addition to visualising the complex cognitive process, simple-to-complex sequencing of whole-task projects needs to be taken into account since completing a realistic project might be too challenging for students at the initial stage even though its process has been visualised. The findings of the study reveal that both visualising the complex cognitive process and simple-to-complex sequencing of whole-task projects are important elements of a computer-based cognitive tool for learning with complex authentic projects. The detailed findings of the study are discussed as follows.

### RQ1. Is the visualisation-based cognitive tool for PjBL of programming effective for improving students' programming performance after the study?

After completing the PjBL module using the visualisation-based learning environment, the students were found to make a significant improvement on all scales of the programming performance concerning programming process (problem understanding, modular design, and process design) and programming product (coding). The results have shown the promising effects of the visualisation-based learning

environment for PjBL of programming. The finding is consistent with a previous study (Peng et al., 2017). The finding also supports the claimed advantages of computer-based cognitive tools in representing and manipulating complex cognition so as to improve student thinking and performance (Chen, Wang, Grotzer, & Dede, 2018; Toth, Suthers, & Lesgold, 2002; Wu & Wang, 2012) and expertise development (Wang, Yuan, et al., 2018).

The improvement made by the students in programming performance after the study was in alignment with their positive perceptions of the learning environment. The participants, either in the experimental group or in the control group, had positive perceptions of the visualisation-based learning environment in terms of its cognitive strategies for support of learning with complex tasks (exploration, scaffolding, modelling, articulation, coaching, and reflection). Students perceived its affordances in scaffolding the complex task process and enabling them to articulate and reflect on their task process; moreover, learners found the learning environment helpful in encouraging learning with authentic projects and enabling the teacher to demonstrate the project process and provide useful feedback to individuals during the project.

### RQ2. Will the incorporation of the simple-to-complex progressive learning approach influence the effects to the visualisation-based learning environment for PjBL of programming (as reflected in students' programming performance and their perceptions of the learning environment)?

*Programming performance*
Students in the experimental group were found to outperform those in the control group in all aspects of their programming performance after the study. By incorporating a progressive approach to learning with complex projects, the visualisation-based cognitive tool for PjBL of programming helped students to achieve a better performance in programming. In using the visualisation-based cognitive tool for PjBL of programming, both learning time and the progressive learning approach played a role in improving students' programming performance. The two factors interacted in their effects on improving student learning of programming using the visualisation-based learning environment.

*Learner perceptions*
Compared to students in the control group, those in the experimental group reported having more positive perceptions of the visualisation-based cognitive tool in terms of two strategies – scaffolding and articulation – but no significant differences in the other four strategies (modelling, coaching, reflection, exploration). The results indicated that the learning environment was better received by learners using the progressive approach to work with a complex project. They found the learning environment more helpful in enabling them to capture the complex process and articulate their individual process of working on a project. By deconstructing a complex project into a set of sub-projects arrayed in a simple-to-complex order, the visualisation-based learning environment may better help students to capture the project process, thus helping them to achieve better performance through practice. Meanwhile, by applying the progressive learning approach, the visualisation-based learning environment may better help students to articulate their individual process for reflection and improvement without being overwhelmed by its full complexity from the outset. The result is in alignment with the findings of previous studies which reported the promising effects of the simple-to-complex progressive approach to learning in programming education (van Merriënboer, 1990; van Merriënboer & De Croock, 1992) and other complex domains (e.g., Tjiam et al., 2012; Vandewaetere et al., 2015).

The results showed no significant differences between the experimental and control groups in student perceptions of the learning environment in terms of its other four strategies (exploration, modelling, coaching, reflection). A plausible reason is that the learning environments for both groups were similar in providing learners with opportunities to work with realistic projects, demonstrating the process of completing a sample project, providing feedback to individual performance, and enabling learners to review and reflect on their individual performance.

## Limitations

The study has some limitations. First, it was conducted in the domain of computer programming education. The findings from this domain are not necessarily generalisable to other domains. Second, while project-based learning can be either individual- or group-based in practice, the present study focused on individual

learning. Third, while an authentic project in this study was deconstructed into three sub-projects in a simple-to-complex order to support student learning, the deconstruction method is not fixed for all kinds of projects. The limitations can be addressed in further research by exploring the effects of the proposed approach in collaborative learning contexts, extending the investigation in other domains, and investigating methods for organising authentic whole-task projects in a simple-to-complex order.

## Conclusion

PjBL has been increasingly used to connect abstract knowledge with authentic practice in educational practice, including programming education. Nevertheless, completing the complex process of a realistic programming project is a pressing issue. This study proposed a visualisation-based cognitive tool to address the challenge by making the complex process of completing a realistic programming project visible to learners. It attempted to scaffold complex learning, enable effective thinking and reflection by learners, and enable the teacher to track and provide feedback on individual learning process. Moreover, a simple-to-complex progressive approach was incorporated into the learning environment to facilitate student learning with complex projects without sacrificing authenticity.

After using the visualisation-based cognitive tool to complete a PjBL module of computer programming, students made a significant pre-post improvement in their programming performance and reported having positive perceptions of the learning environment in terms of its cognitive strategies for support of learning with complex tasks. Further, students found the learning environment more helpful when it was incorporated with the progressive approach to learning with complex projects. The visualisation-based progressive learning environment helped students to achieve better programming performance after the study with projects.

Project-based learning is much more easily advocated than accomplished. Although emerging learning technologies have substantially expanded the opportunities for working with authentic tasks or realistic projects, there is a great need to investigate the challenges experienced by learners in such contexts and examine how such challenges can be resolved by effective design of instructional scaffolds and cognitive tools with the support of technology. The findings of the study may contribute to knowledge of how effective learning with complex realistic projects can be realised through a visualisation-based progressive learning environment as a sort of computer-based cognitive tool.

The implications of the findings are twofold. Firstly, while visualisation-based cognitive tools have been mainly used to help learners to understand abstract concepts and complicated behaviour of computer programs and supporting the coding process, it is important to visualise the complex cognitive process of programming problem-solving, which includes not only coding and debugging but also problem formulation, solution planning, and solution design. Secondly, while visualising the complex cognitive process has demonstrated its salient affordances in supporting learning with realistic projects, deconstructing a complex realistic project into a set of simple-to-complex sub-projects made the visualisation-based cognitive tool more effective in supporting learning with complex projects.

## Acknowledgements

This project was supported by the General Research Fund from the Research Grants Council of the Hong Kong SAR Government (Project No. 17201415), the Seeding Fund for Basic Research from the University of Hong Kong (Projects No. 201811159019, No. 201611159071), and the Eastern Scholar Chair Professorship Fund (No. JZ2017005) from the Shanghai Municipal Education Commission of China. The authors thank Professor Haijing Jiang for his valuable support for this study.

## References

Bassil, Y. (2012). A Simulation model for the waterfall software development life cycle. *International Journal of Engineering & Technology, 2*(5), 742–749. https://doi.org/10.12691/ajse-5-1-2

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. https://doi.org/10.3102/0034654316670999

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. https://doi.org/10.1080/00461520.1991.9653139

Chen, J., Wang, M., Grotzer, T. A., & Dede, C. (2018). Using a three-dimensional thinking graph to support inquiry learning. *Journal of Research in Science Teaching, 55*(9), 1239–1263. https://doi.org/10.1002/tea.21450

Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator, 15*(3), 6–11. https://doi.org/10.1.1.124.8616

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. https://doi.org/10.1002/j.2168-9830.1999.tb00453.x

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. https://doi.org/10.1002/j.2168-9830.1997.tb00270.x

Ge, X., & Land, S. (2003). Scaffolding students' problem-solving processes in an ill-structured task using question prompts and peer interactions. *Educational Technology Research and Development, 51*(1), 21–38. https://doi.org/10.1007/BF02504515

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. https://doi.org/10.1080/10508406.2012.748664

Gómez-Albarrán, M. (2005). The teaching and learning of programming: A survey of supporting software tools. *The Computer Journal, 48*(2), 130–144. https://doi.org/10.1093/comjnl/bxh080

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. https://doi.org/10.1007/s10734-004-6386-5

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. https://doi.org/10.1080/00461520701263368

Hooper, C., Carr, L., Davis, H., Millard, D., White, S., & Wills, G. (2007). AnnAnn and AnnAnn. Net: Tools for teaching programming. *Journal of Computers, 2*(5), 9–16. https://doi.org/10.4304/jcp.2.5.9-16

Jollands, M., Jolly, L., & Molyneaux, T. (2012). Project-based learning as a contributing factor to graduates' work readiness. *European Journal of Engineering Education, 37*(2), 143–154. https://doi.org/10.1080/03043797.2012.665848

Jonassen, D. H. (1996). *Computers in the classroom: Mindtools for critical thinking*. Englewood Cliffs, NJ: Prentice Hall.

Jonassen, D. H., Carr, C. & Yueh, H. P. (1998). Computers as mindtools for engaging learners in critical thinking. *TechTrends, 43*(2), 24–32. https://doi.org/10.1007/BF02818172

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. https://doi.org/10.1207/s15326985ep4102_1

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Computer Science Education, 13*(4), 249–268. https://doi.org/10.1076/csed.13.4.249.17496

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. https://doi.org/10.1002/smr.270

Lajoie, S. P., & Derry, S. J. (Eds.). (1993). *Computers as cognitive tools*. Hillsdale, NJ: Lawrence Erlbaum. https://doi.org/10.4324/9780203052594

Lee, M. J. W., Pradhan, S., & Dalgarno, B. (2008). Using screencasting to scaffold exercises and promote cognitive engagement for novice object-oriented programmers. *Journal of Information Technology Education, 7*, 61–80. https://doi.org/10.28945/179

Marcellis, M., Barendsen, E., & van Merrienboer, J. J. G. (2018). Designing a blended course in Android App development using 4C/ID. In *Koli Calling '18. Proceedings of the 18th Koli Calling International Conference on Computing Education Research* (pp. 1–5). New York, NY: ACM Press. https://doi.org/10.1145/3279720.3279739

Peng, J., Wang, M., & Sampson, D. (2017). Visualizing the complex process for deep learning with an authentic programming project. *Journal of Educational Technology & Society, 20*(4), 275–287. Retrieved from https://www.jstor.org/stable/26229223

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. https://doi.org/10.1016/j.sbspro.2011.11.398

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. https://doi.org/10.28945/3237

Reiser, B. J. (2004). Scaffolding complex learning: The mechanisms of structuring and problematizing student work. *Journal of the Learning Sciences, 13*(3), 273–304. https://doi.org/10.1207/s15327809jls1303_2

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Scaife, M., & Rogers, Y. (1996). External cognition: How do graphical representations work? *International Journal of Human-Computer Studies, 45*, 185–213. https://doi.org/10.1006/ijhc.1996.0048

Slof, B., Erkens, G., Kirschner, P. A., Janssen, J., & Jaspers, J. G. M. (2012). Successfully carrying out complex learning-tasks through guiding teams' qualitative and quantitative reasoning. *Instructional Science, 40*(3), 623–643. https://doi.org/10.1007/s11251-011-9185-2

Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 850–858. https://doi.org/10.1145/6592.6594

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. https://doi.org/10.1145/2490822

Spector, J. M., & Anderson, T. M. (Eds.) (2000). *Integrated and holistic perspectives on learning, instruction and technology: Understanding complexity.* Dordrecht, The Netherlands: Kluwer Academic Press. https://doi.org/10.1007/0-306-47584-7

Stalmeijer, R. E., Dolmans, D. H., Wolfhagen, I. H., Muijtjens, A. M., & Scherpbier, A. J. (2008). The development of an instrument for evaluating clinical teachers: involving stakeholders to determine content validity. *Medical Teacher, 30*(8), 272–277. https://doi.org/10.1080/01421590802258904

Stalmeijer, R. E., Dolmans, D. H., Wolfhagen, I. H., Muijtjens, A. M., & Scherpbier, A. J. (2010). The Maastricht Clinical Teaching Questionnaire (MCTQ) as a valid and reliable instrument for the evaluation of clinical teachers. *Academic Medicine, 85*(11), 1732–1738. https://doi.org/10.1097/ACM.0b013e3181f554d6

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. https://doi.org/10.1016/j.compedu.2006.10.007

Sykes, E. R. (2007). Determining the effectiveness of the 3D Alice programming environment at the computer science I level. *Journal of Educational Computing Research, 36*(2), 223–244. https://doi.org/10.2190/J175-Q735-1345-270M

Tjiam, I. M., Schout, B. M., Hendrikx, A. J., Scherpbier, A. J., Witjes, J. A., & van Merriënboer, J. J. (2012). Designing simulator-based training: An approach integrating cognitive task analysis and four-component instructional design. *Medical Teacher, 34*(10), e698–e707. https://doi.org/10.3109/0142159X.2012.687480

Toth, E. E., Suthers, D. D., & Lesgold, A. M. (2002). "Mapping to know": The effects of representational guidance and reflective assessment on scientific inquiry. *Science Education, 86*(2), 264–286. https://doi.org/10.1002/sce.10004

Tynjälä, P. (2008). Perspectives into learning at the workplace. *Educational Research Review, 3*(2), 130–154. https://doi.org/10.1016/j.edurev.2007.12.001

Vandewaetere, M., Manhaeve, D., Aertgeerts, B., Clarebout, G., van Merrienboer, J. J., & Roex, A. (2015). 4C/ID in medical education: How to design an educational program based on whole-task learning: AMEE Guide No. 93. *Medical Teacher, 37*(1), 4–20. https://doi.org/10.3109/0142159X.2014.928407

van Merriënboer, J. J. G. (1990). Strategies for programming instruction in high school: Program completion vs. program generation. *Journal of Educational Computing Research, 6*(3), 265–285. https://doi.org/10.2190/4NK5-17L7-TWQV-1EHL

van Merriënboer, J. J. G., & De Croock, M. B. M. (1992). Strategies for computer-based programming instruction: Program completion vs. program generation. *Journal of Educational Computing Research, 8*(3), 365–394. https://doi.org/10.2190/MJDX-9PP4-KFMT-09PM

van Merriënboer, J. J. G., & Kirschner, P. A. (2017). *Ten steps to complex learning: A systematic approach to four-component instructional design* (3rd ed.). New York, NY: Routledge. https://doi.org/10.4324/9781315113210

van Merrienboer, J. J. G., & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review, 17*(2), 147–177. https://doi.org/10.1007/s10648-005-3951-0

Wang, M., Cheng, B., Chen, J., Mercer, N., & Kirschner, P. A. (2017). The use of web-based collaborative concept mapping to support group learning and interaction in an online environment. *The Internet and Higher Education, 34*, 28–40. https://doi.org/10.1016/j.iheduc.2017.04.003

Wang, M., Derry, S., & Ge, X. (2017). Guest Editorial: Fostering deep learning in problem solving contexts with the support of technology. *Educational Technology & Society, 20*(4), 162–165. Retrieved from https://www.jstor.org/stable/26229214

Wang, M., Wu, B., Kirschner, P. A., & Spector, J. M. (2018). Using cognitive mapping to foster deeper learning with complex problems in a computer-based environment. *Computers in Human Behavior, 87*, 450–458. https://doi.org/10.1016/j.chb.2018.01.024

Wang, M., Yuan, B., Kirschner, P. A., Kushniruk, A. W., & Peng, J. (2018). Reflective learning with complex problems in a visualization-based learning environment with expert support. *Computers in Human Behavior, 87*, 406–415. https://doi.org/10.1016/j.chb.2018.01.025

Wu, B., & Wang, M. (2012). Integrating problem solving and knowledge construction through dual mapping. *Knowledge Management & E-Learning, 4*(3), 248–257. https://doi.org/10.34105/j.kmel.2012.04.021

---

**Corresponding author**: Minhong Wang, magwang@hku.hk

**Please cite as**: Peng. J., Wang, M., Sampson, D., & van Merriënboer, J. J. G. (2019). Using a visualisation-based and progressive learning environment as a cognitive tool for learning computer programming. *Australasian Journal of Educational Technology, 35*(2), 52–68. https://doi.org/10.14742/ajet.4676