A Pedagogical Pattern for Teaching Computer Programming to Non-CS Majors

Zhen Jiang
Dept. of Computer Science
West Chester University
West Chester, PA 19383, USA
E-mail: zjiang@wcupa.edu

Eduardo B. Fernandez
Dept. of Comp. & Elect. Eng. and Comp. Sci.
Florida Atlantic University
Boca Raton, FL 33431, USA
E-mail: ed@cse.fau.edu

Liang Cheng
Dept. of Comp. Sci. & Eng.
Lehigh University
Bethlehem, PA 18015, USA
E-mail: cheng@cse.lehigh.edu

***Abstract*—We introduce a new method for non-computer-science majors to quickly learn computer programming. The challenge here is to help those who lack sufficient background knowledge, to quickly gain the skills and knowledge to develop programs correctly. Traditional computer science programming requires several semesters and many foundation courses. By using our practice of loop program training at West Chester University as an example, we demonstrate how effectively our approach can attract students and can keep them working hard on the materials with a significant technical depth. On one hand, we adopt the method that is commonly used in programming training in China in order to cover all the required materials with many subtasks. On the other hand, we use the traditional method in the American classes and provide a condensed, comprehensive case study, to help students form the abstractions, understand the corresponding materials, gain the appropriate skills, and achieve each intermediate task goal. This pattern provides a solution for a complex education problem in a short time scale.**

***Index Terms*—Course curriculum, pedagogical pattern, teaching programming.**

I. Introduction

Computers have changed our world in many significant ways. Students, especially senior students, from other science majors with various backgrounds require computer skills for their own major study or research work. This creates a demand for a quick training that can prepare them one or two semesters. Existing entry-level courses in CS0, CS1, or minor programs (e.g., [4]–[12]) that are available for those inexperienced non-CS majors usually aim to a long-term, systematic, and broad study for the CS major/career. The need for quick training is usually ignored. On the other hand, most of the computer applications on which they are working are commercial-off-the-shelf and use many advanced techniques in industry that are beyond what we currently teach in the entry-level courses. A new effective learning process is needed to quickly prepare students from widely divergent background to be able to have some of the programming abilities of CS graduates.

Loops [14] are one of the basic program structures. However, as indicated by Elliot Solloway, even experienced programmers have but a 50/50 chance of developing the correct loop when confronted with a do-while (or do-until) choice. To ensure the correctness, the programmers must learn many theoretical aspects such as axiomatic semantics [20] and require a lot of practice. With the pedagogical pattern proposed in this paper, we illustrate our approach to complete the required training within a very limited time frame.

Our approach integrates different teaching styles. Adopting the Chinese training model, the entire learning process is disciplined in such a way that the learning outcomes can be assessed easily in different steps. Reaping the general benefits of using a pattern, i.e., reusability and concept abstraction [18], students easily learn and successfully practice the deeper materials within a limited time in each step using the traditional American teaching/learning model to form abstraction, understand materials, and gain skills. Accumulating achievements in each step helps us reach an ambitious educational goal within a short time scale. A 3-phase solution is presented.

1) We adopt game-like computer equipment, with research activities relevant to the topics discussed in class to give students an overall picture of the entire training process, motivating their study from the beginning and verifying their ultimate learning outcomes at the end.
2) Adopting the <u>American</u> training model, we develop a software tool to encourage students to have sufficient programming practice. In order to help students successfully conquer the obstacles in each step (or subtask), we summarize programming experience into templates. In this way, the advanced materials can be transparent, making them easy to learn and to practice.
3) Organized under the <u>Chinese</u> training model, each step has its own learning outcomes verified by practicing our customized software tool in different levels.

The remainder of this paper is organized as follows: Section 2 describes the pattern, while Section 3 presents some conclusions.

II. Pedagogical Pattern for Teaching Programming to Non-CS majors

This pattern describes a method to attract a broad range of students (e.g., non-CS majors) to a computer training that teaches structural programming, such as loop development. It keeps them working hard on materials with a significant technical depth. The students do not have much background in programming, so we need to introduce concepts without requiring much previous knowledge. This helps taming the complex teaching tasks in a very short time frame and guarantees the quality of their education and the expected outcomes.

A. Context

A course or program with complex education tasks, for instance, industry training in centers that needs to prepare programmers in a short time; computer training in universities that teaches structural programming to non-CS majors, such as loop development for junior students of the Physics department (and other 11 departments such as the Accounting department) at West Chester University (WCU) who had little or limited computer experience but require computer techniques to process the experimental data for their senior research projects.

B. Problem

The key is to draw students' interest and keep this during the entire procedure of our condensed and accelerated training. The challenges are

1) the difficulty of learning with critical time constraints
2) the widely divergent background, and
3) the lack of sufficient technique preparation.

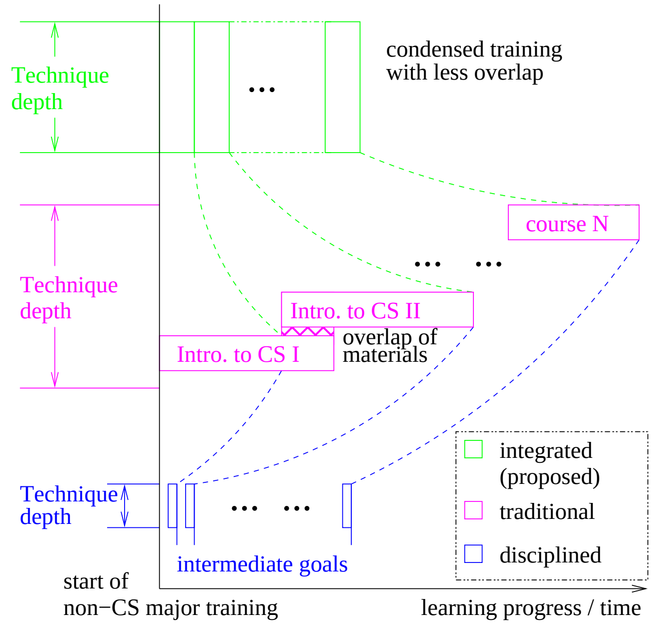


Fig. 1. Comparison of different teaching styles.

C. Forces

Traditional class training

- **Pros**: uses many examples to help students form the concept abstraction and gain programming skills;
- **Cons**: is not feasible in a course with critical time constraints.

The training is not only limited to the class time but also extended to project practice in the succeeding courses. Such training in existing bachelor's study takes several semesters and requires many core courses before any advanced topic course can be taken (i.e., the traditional teaching style shown in Figure 1 in magenta).

Many educators have noticed the advantages of teaching in different cultures (e.g., [17]). In the Chinese teaching style,

- **Pros**: the trainees are disciplined so that they can be guided to finish the work quickly;
- **Cons**: the learning is fragmental and superficial and faces a high failure ratio.

The success of iPhone application development in China proves that it is possible for amateurs to quickly obtain a high level of proficiency. In such a model (i.e. the disciplined teaching style shown in Figure 1 in blue), the trainees accomplish the study by achieving many intermediate goals. However, each sub-task does not have sufficient help to guide the trainee's analysis and thought. It is difficult for the trainee to extend his/her knowledge due to the lack of a complete background on the subject (i.e., fragmental and superficial). Moreover, the ultimate training goal cannot be reached whenever any sub-task becomes an obstacle for the trainee and this person cannot have the required thinking and analysis by him- or herself.

D. Solution

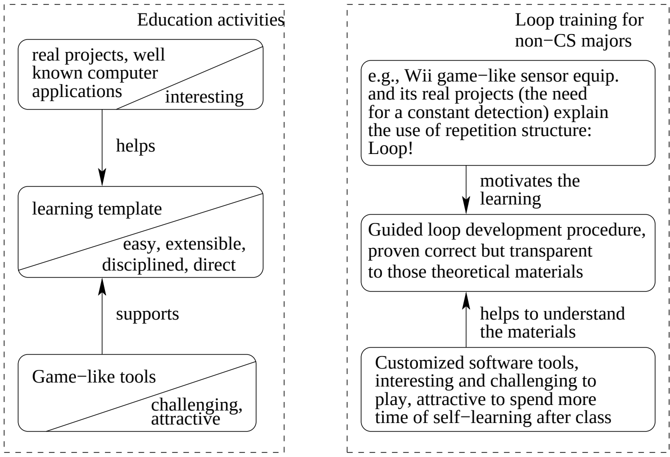
To find an education solution that meets all challenges as the proposed style shown in Figure 1 in green, we need to find a balance of the forces of different teaching styles, taking advantages of both of them.

To schedule student progresses within a very limited time frame, we adopt the Chinese discipline model and design the learning process via multiple steps. Each step has its own intermediate goals that are relatively easy to achieve and has its outcomes quickly verified. The overlap among different steps can be reduced to the minimum, in order to avoid any unnecessary replication. By accumulating all intermediate achievements, the ultimate learning goal can be achieved quickly with this method.

To guarantee the quality of education at each step for inexperienced trainees to obtain the expected outcomes, we integrate the necessary processes of analyzing and thinking into tasks under the traditional American model. The student learning is organized in a pre-set procedure. The advanced materials can be transparent in the description of the success experience in this template. The concept abstraction provides the disciplines that we need to guide the student study. Its extensibility provides the room for students to have a process of analyzing and thinking.

To handle the widely divergent background in training, we use real research projects to motivate the students by giving a big picture of what they can do. Then, to guarantee the success of each step, we develop a game-like tool to help the learning. It balances the superficial education and its high failure rate under the Chinese style in the following aspects:

- The human-computer interaction attracts the students to spend more time after class in a self-learning process,



(a) Pattern design

(b) Pattern application in loop study

Fig. 2. The proposed pedagogical pattern.

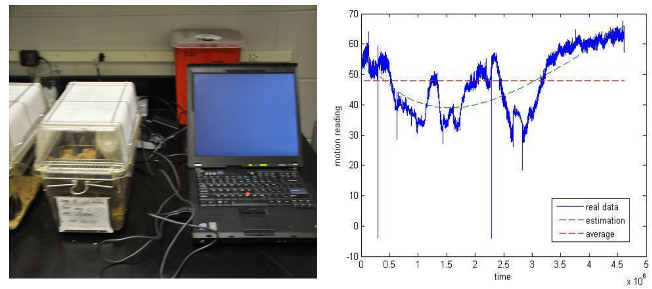


Fig. 3. (a) Research in the biology laboratory at WCU with the iMote2 sensing platform, and (b) the results that are analyzed by a loop program.

guaranteeing enough practice time that we require.

- The computer tool will test the students' learning progress with different difficulty stages, encouraging students to learn the materials gradually and continuously.
- The computer tool uses a comprehensive set of testing cases, which helps instructor(s) to verify the completeness and quality of the training and to synchronize the students' progress.

We make the training at each step fit the disciplines under the Chinese training style by the follows:

- The computer grading in the test will force students to accomplish a quality work, helping instructor(s) to ensure the students' progress in a very limited time frame.
- The tool for the student work after class saves time in class, making the condensed training feasible.

Our solution focuses on realizing the steps that are disciplined by the Chinese training model with the American model. It is implemented in 3 phases as illustrated in Figure 2 (a). In phase one, we adopt game-line computer equipment with research activities relevant to the training to motivate student study at the beginning and use its sample work to verify the ultimate outcomes at the end. In phase two, we develop a software tool to encourage students to have sufficient programming practice. In order to help students successfully conquer the obstacles in each step (or subtask), we adopt templates to train the programming development. In this way, the advanced materials can be transparent, making them easy to learn and to practice. In phase three, at each step during the entire learning process, a software tool is used to verify the student achievement and to encourage students to continue the learning. Figure 2 (b) shows our implementation of this approach for a sample loop training.

E. Example for pattern implementation

We use the loop training at WCU to demonstrate the proposed approach. In the following, we first list our ultimate education goals as the training requirements. Then, we show the challenges in the proposed training. The learning of loop development is organized in 3 subtasks (i.e., steps), following the criteria of national training standard for information systems security professionals [16]. Step 1 introduces the loop syntax in class time. Step 2 discusses a correct development process with templates in class. Step 3 focuses on student practice in and after class with the development template. Each step has its own education goal according to its different role in the entire learning process. The rest of discussion in this part focuses on how practical and effective a learning of loop development template under the American training model helps to realize the above 3 steps that are disciplined by the Chinese training model. Such a method will guarantee the quality of student work of the necessary analyzing and thinking, while meeting all challenges. We will discuss our implementation of that 3-phase solution.

1) Requirements of a loop program training for non-Computer-Science majors: The loop is an essential program structure. Our practice is not only limited to the learning of its syntax, but also has the following highly expected outcomes in order to prepare students for the programming of the research projects in their own major:

- to know when and where a loop is needed,
- to be able to interpret a sequence of events/activities of program execution into a loop program,
- to have knowledge of the correctness of a loop program,
- to be able to verify a loop program and to correct any possible error,
- to know how to maintain the program when the requirements or situation change.

2) Constraints: The challenge is to achieve the above goals as fast as in 3 classes (75 minutes each) or equivalently 10 days. Note that many students in such a class are from other departments such as the Performance & Arts Department. Those students are from 11 different departments and may not have sufficient background knowledge. Imagine that those inexperienced students will be asked to ensure that all code lines are correct since their very first program. An effective education model is needed.

3) 3-phase solution: In phase 1, we use work on real research projects to motivate students. Without the appropriate

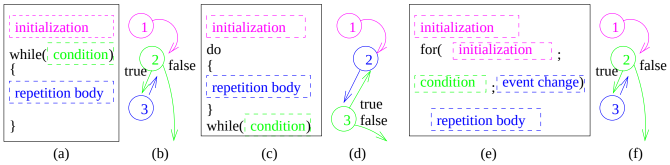


Fig. 4. Statement syntax and operational semantics. (a)-(b) While loop, (c)-(d) do-while loop, and (e)-(f) for loop.

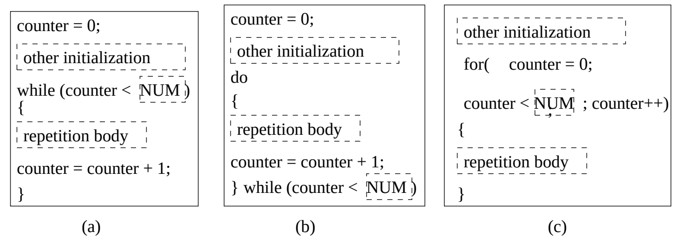


Fig. 5. Counter-control loop template (Dashed line encloses the part to be decided).

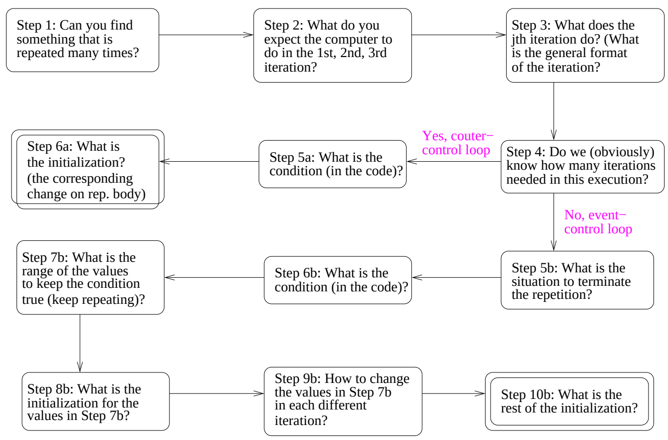


Fig. 6. Disciplines to develop loop programs (development template).

motivation, the non-CS majors will work for credits only to meet the general education requirement for their graduation. They can easily be distracted from the required learning material according to our classroom experience. The planned education will become very difficult. However, due to the variety of backgrounds of those students, such a research result should also have a broad impact that can be recognized by most students.

For the loop study, we introduce our research [2] with the Wii [15] game-like equipment iMote2 [13] (see Figure 3). It is used to detect the change of environment in terms of the temperature and the light, continuously for every second in a 7/24 schedule. To analyze the data collected, even simply to obtain the average, a loop is needed. This introduction of our iMote2 work can bridge the student development in class with their daily life and even the jobs in industry. Unlike existing activity-driven teaching that adopts laboratory materials of wireless sensing for senior project development (e.g. [22]), our use of wireless sensing equipment focuses on the enhancement of students' interest. Once the students finish the loop training, they will have a chance later to apply the programming knowledge to change the control of sensor equipment in the sample code. This will be tested in our customized software (discussed later) and its success ratio will help to verify the ultimate learning outcomes that a CS graduate may have.

In phase 2, our education adopts a learning process that helps students obtain knowledge from practice. First, the trainees will follow a disciplined procedure to practice the correct loop program development. The development follows a standard, to avoid awkward codes that cause a potential hard work in the maintenance of the code. Advanced concepts and advanced techniques such as program correctness are transparent, but effectively guaranteeing the quality of programming. Then, when the trainees practice such a procedure of programming, they repeat the development of correct programs, accumulating the experience to gain the required skills and knowledge. As an abstract module, this procedure can satisfy different purposes for the use of loop statements, meeting the goal of our training. Note that the correctness proof for such a procedure is out of the scope of this paper and is omitted.

Figure 4 shows the statement syntax and the operational semantics of loop development. It is used in class to explain the use of the loop development procedure. The *repetition body* consists of the code lines that are repeated many times in a sequence. The *condition* is used to stop the repetition when it becomes false. The *initialization* consists of all assignments changing the value of variables that are used in the repetition body before the loop starts its 1^{st} iteration.

Figure 6 shows the details of the development template. The development starts from the determination of the repetition body. Then, the termination condition is decided. At the end, the initialization is finished to ensure that the right code can be applied to the right pre-condition. This process is based on the operational semantics of the loop and it is easy to follow. When the number of loop iterations can be known easily, this loop is called counter-control loop; otherwise, it is called event-control loop. To simplify the development, we strongly suggest the use of the counter-control loop because it has a simple, structural regularity (see the template in Figure 5) and require a fewer steps of development (i.e., less work and fewer mistakes). Note that the correctness of the loop now is easy to verify for inexperienced programmers by re-applying the same development procedure for a consistency check. Algorithm 1 summarizes the function of each step. Figures 7 and 8 show the development of a sample loop program in different ways: the counter-control loop and the event-control loop respectively. The development progress at each step is highlighted in red. The event-control loop requires more steps (i.e., steps 7−9) to determine the event descriptor and its value changes.

Our program development introduces the analysis and thinking to the life cycle of software development as a) requirement analysis, b) design & coding, c) verification, and d) maintenance change:

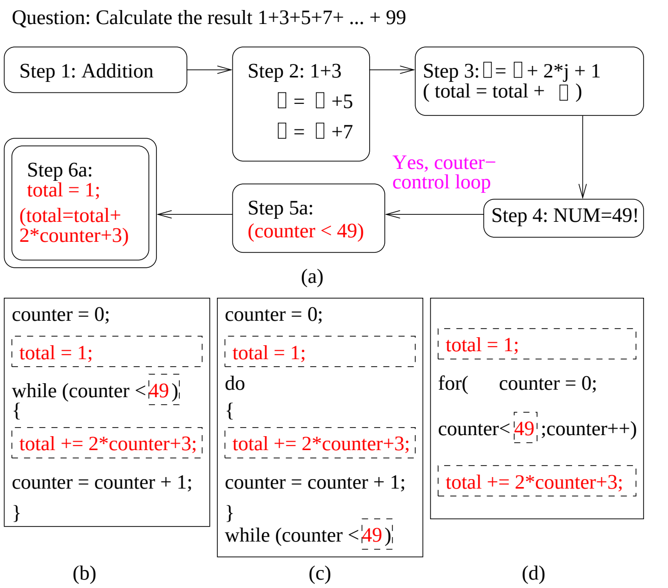


Fig. 7. (a) Development of a counter-control loop, highlighted in red, following the template in Figure 6. (b), (c), and (d) The resulting program in different formats.

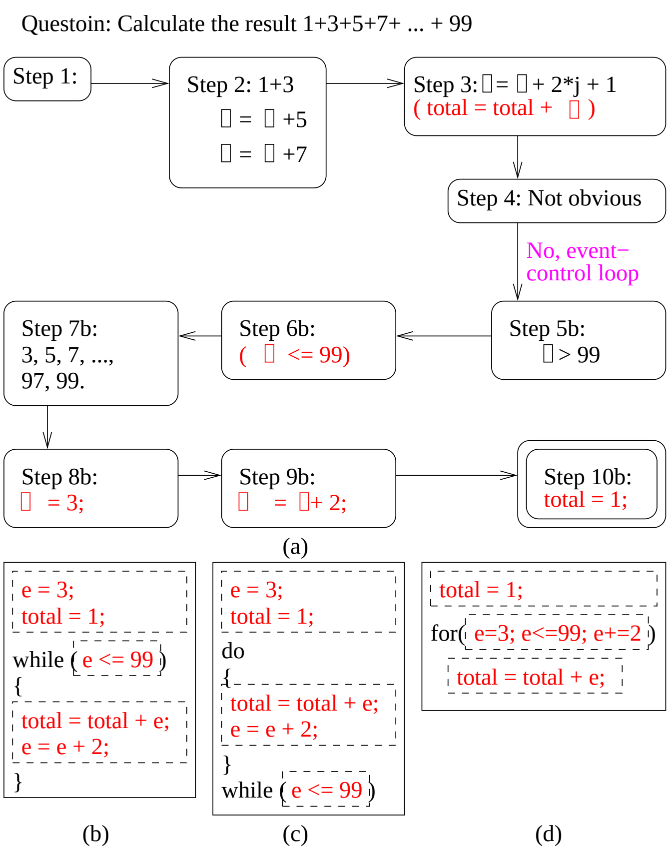


Fig. 8. Development (a) and resultant programs (b), (c), and (d) of an event-control loop under the guide of the template in Figure 6.

Algorithm 1: Loop development template.

1) Determine the repetition body and check the necessity of using a loop statement (Step 1 in Figure 6).
2) Find the general format of the repetition body that is derived from each iteration in the sequence of execution (Steps 2 and 3 in Figure 6).
3) If the number of iterations can be known, the program can be developed in a counter-control loop mode. Otherwise, an event-control loop is needed .
4) For a counter-control loop, provide the condition and initialization (Steps 5a and 6a in Figure 6) for the required work that has been shown in Figure 5.
5) For an event-control loop, decide the condition to terminate the repetition process (Step 5b in Figure 6). Then, determine the event description and its change in the repetition (Steps 6b – 9b in Figure 6). At last, provide the initialization (Step 10b in Figure 6) to finish the work.

- **Analysis:** *When is a loop needed?*
  Answer: Whenever we find that something is repeated many times in a sequence (also called the repetition body).
- **Design & Coding:** *How is a loop developed?*
  Answer: First, we obtain the description of the repetition body. This forms the abstraction from all iterations, i.e., an "everyone" model that can represent all iterations. The process is similar to the development of the general process for each element in an array, but it focuses on the common part of iterations in the time sequence. This procedure also supports nested-loop development by recursively applying the same development process on the internal repetition body. Next, we complete the code (i.e., the terminating condition and the initialization part) in counter-control loop mode or event-control loop mode respectively.
- **Verification:** *How do I know I am doing this correctly?*
  Answer: To avoid any careless mistake, we need to verify each step with the results obtained in all previous steps. For instance, the initialization development in Step 6a must be consistent with the 1^{st} iteration described in Step 2. This verification also identifies those identical programs, in order to avoid unnecessary change.
- **Maintenance:** *What about a change of code?*
  Answer: Simply repeat the above processes until everything is consistent after the check of the verification phase.

In phase 3, even though many advanced matters are clear, the students may still feel that the procedure is too complex to follow because they are not convinced by the necessity of each part/step. Due to the limited practice time allowed in class, a tool is needed to help students learn each programming step by themselves after class. We have developed a customized

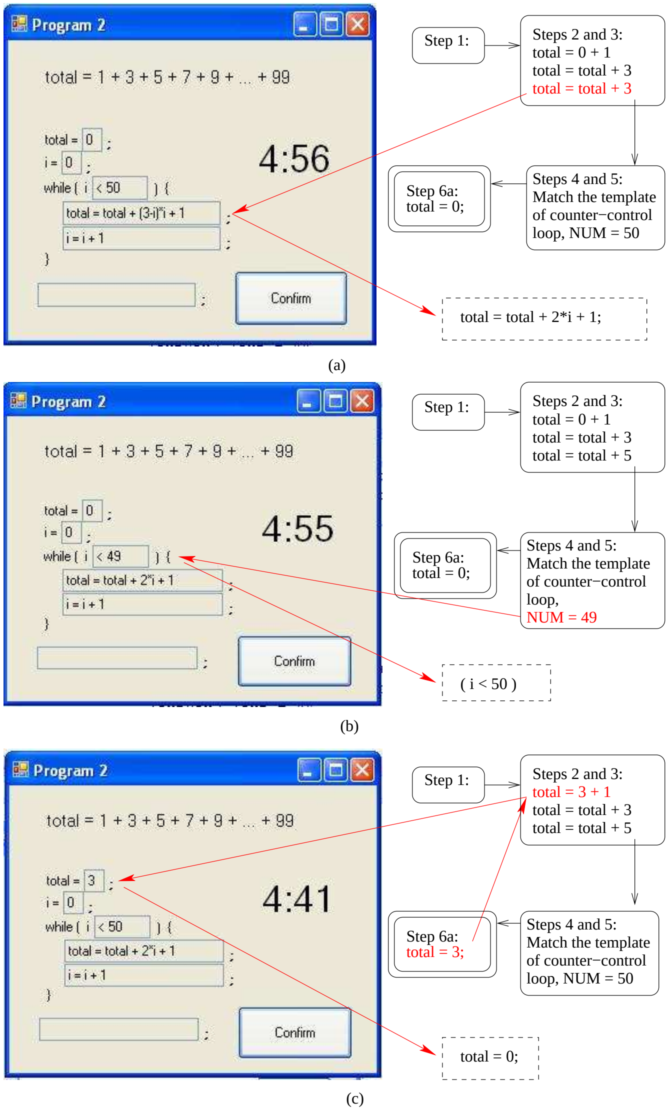


Fig. 9. Test of the counter-control loop development with the computer tool ("one error to correct"). The red arrow indicates the thinking process.

software for students to practice the loop development with the proposed template procedure. This computer game-like software will use one of 10 loop programs according to the difficulty level the student selected. For each selected loop, the computer will randomly create an implementation which has one and only one error among all 12 different steps (in the template). The student will be asked to correct this and only this error, in 5 minutes. This hand-on exercise with up to 67 different cases for each selected loop will force this student to have a deep understanding of the entire development procedure. It also provides certain flexibility for students to control the amount of exercise needed after class (e.g., hours spent on and times of trials) if they are confident with what they have learnt.

This kind of "one error to correct" game helps students to go through the development process of Algorithm 1 and helps

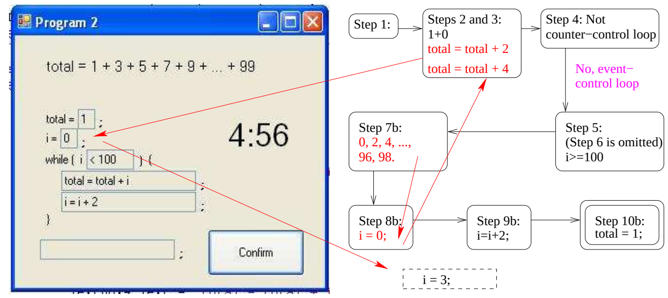


Fig. 10. Test of the event-control loop development with the computer tool.

them understand all the relevant details. Figure 9 illustrates the use of such a tool for the development procedure of the counter-control loop. For instance, in Figure 9 (a), an inconsistency is found when the 3^{rd} iteration has the result of "+3" which is different from the expected "+5", by following the development procedure. Since there is no other error, the body needs a new code to ensure all the iterations have the right addition operation. The required change "total=total+2*i+1" is obvious. In Figure 9 (b), the loop stops one iteration short (i.e., number of iterations $NUM = 49$), which demands for a change of condition ("i<50" or "i<=49"). In the case in Figure 9 (c), we can find two inconsistent places: one is in the execution of the 1^{st} iteration and the other is in the initialization. Considering that only one error incurs both inconsistencies, the initialization must be changed ("total=0").

Figure 10 demonstrates a similar process for the event-control loop. We have 4 inconsistent places: two are in the iterations, one is in the value changes of the event descriptor, and one is in the initialization part. After the event initialization is changed ("i=3"), the code passes the verification and becomes 100% correct.

4) Consequences: Figure 11 summarizes our 9 years' experience in teaching loop development in a general education course for non-CS majors. After the introduction to syntax of loops, the students are asked to do a self-evaluation. Most of them are confident with what they have learned and estimate that they can finish 70% of loop development work in this class. However, when we test with the software tool whether they can think about the subject matter as a CS graduate can (e.g., the analysis and thinking process in Figures 9 and 10), only 4% of the students can really finish the job.

After the introduction to our template (in Figure 6) for the program development in our iMote2 research project, students are willing to use our proposed development procedure in their exercises. After that, up to 40% of the students get the idea and pass the test. However, because the training is limited to one week only, up to 60% of the students still have a difficult time and cannot fully understand the materials. In the past 3 years, we introduced the tool in class and used it in student projects. The improvement is substantial and significant. After the discussion of such a tool in class, students learned how to develop program correctly. 50% of the students can successfully finish the required work in a quiz. Then,

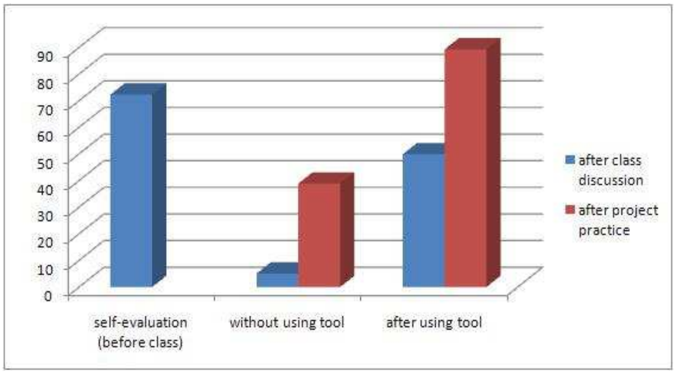


Fig. 11. Outcome assessment.

after their project practice with that tool, up to 90% of the students can pass the computer test, achieving our education goal (as a CS graduate can think in Figures 9 and 10) for such a condensed training.

F. Known uses

While many computer scientists focus on the quality of programming in large-scale and complex systems, the industry requires a more general education of computer programming for inexperienced people, in order for them to fulfill certain programming jobs. Recently, more and more universities have resorted to new training programs (e.g., various minor or certificate programs) to meet such a market demand. One of the NSA certified programs [16] at West Chester University offers undergraduate students a quick information technology (IT) training by taking only 6 courses. "CSC115 - Introduction to Computer Programming" is an entry-level programming course and is the only prerequisite course for non-computer-science majors to take our NSA-certified topic courses. This course aims to structure a programming basis for the development of complex computer application in the succeeding topic courses.

Our CSC115 at WCU is a general education course for all non-Computer-Science majors, with up to 8 sessions each semester, four times the number of sessions of the entry-level programming course that we make available to Computer Science majors. Recently, its curriculum upgrade with our proposed pattern has gained the attention from other departments at WCU and their students. For instance, a joint research has been initiated with the Biology department and the Physics department, in order to have more suitable projects for their students' computer training.

The corresponding education practice has also attracted the interdisciplinary education collaboration from other schools. For instance, our training pattern has been adopted in a newly created summer programming training for CS majors in Shanghai Jiaotong University to enhance their programming ability, which has won the ACM International Collegiate Programming Contest [1] 3 times in the past 8 years.

G. Related patterns

Pedagogical patterns [3], [19], [21], [23] are high-level patterns in documenting good education and training practices and experiences through design patterns. They have been recognized in many areas of training such as group work, software design, human computer interaction, education, and others.

Although many pedagogical patterns such as "Spiral" use the steps (or fragments) in the learning process, they focus on the repetition or additional use to enforce the learning. This requires extra time and defers our learning with critical time constraints.

The lack of sufficient background knowledge brings a new insight to our loop development. Our practice for non-CS majors who haven't obtained systematic training of computer programming provides a pedagogy solution for this complex programming education problem in a short time scale. Patterns such as "Consistent Metaphor", "Lay of the Land", and "Larger than Life" help us to determine the use of real research projects in the introduction to the loop program in class. "Tool Box" helps us to decide the use of the Wii-game like wireless equipment, which motivates the students at the beginning and verifies the outcomes at the end. "Toy Box" and "Fixer Upper" motivate us to create the "one error to correct" software tool, in order to help students learn the proposed development template.

III. Conclusion

Our goal is to make a condensed training practical for inexperienced trainees. As a result, our computer training such as the one for loop development has achieved a good success. In this paper, we have shown the efficiency and effectiveness of our education model to organize the learning in and after class through its three phases. The students are motivated to follow a disciplined procedure and learn to handle the complex aspects of a routine. As an abstract module, our approach can satisfy a variety of education purposes. The Computer Science Department of WCU has adopted it to attract more non-CS majors to enroll in CSC115 course and develop their computer skills. Similar training has been planned in the the interdepartmental and interdisciplinary level, in order to repeat the success of our CSC115 education.

The specific problem in teaching trainees who have little experience is of intrinsic interest because of its economic importance and potential market value. It is clear that general education, certificate programs, and minor programs are used in many places. These programs vary in language, application and teaching method. However, our search did not yield any complete comparison. Based on the discussion above, it is clear that learning programming with our proposed education model works better with practice. More importantly, by reaping the benefits of using patterns, our approach can evolve to a better teaching model for computer education.

ACKNOWLEDGMENT

We especially thank our shepherd, David West, who provided valuable comments which considerably improved this paper. We would also like to thank Ms. Ayesha Begum for her assistance in the software development. This work was supported in part by NSF REU grant CCF 0936942.

REFERENCES

[1] ACM International Collegiate Programming Contest. Information is available at http://en.wikipedia.org/wiki /ACM_International_Collegiate_Programming_Contest.

[2] An accurate measurement of infection on mice with wireless Imote2 sensor equipment, supported by the CAS dean's office at WCU. Information is available at http://www.cs.wcupa.edu/∼zjiang/RA_Spring11.htm.

[3] J. Bergin. *Pedagogical Pattern*. Information is available at http://csis.pace.edu/∼bergin/#pedpat.

[4] CS121−204, undergradate programming courses, Comptuer Science Department, Drexel University. Information is available at http://www.drexel.edu/catalog/ug/coe/cs-index.htm#.

[5] CIS101-Introduction to Computer Science: Principles of Information and Computation, Department of Computer and Information Science, University of Pennsylvania. Information is available at http://www.cis.upenn.edu/ugrad/all-courses.shtml.

[6] CIS1001−68, programming courses for all science majors, College of Science and Technology, Temple University. Information is available at http://www.temple.edu/bulletin/ugradbulletin/ ucd/ucd_cis.html.

[7] CISC101-Computers and Information Systems, University of Delaware. Information is available at http://primus.nss.udel.edu/CourseDesc/ info.action?searchKey=2011%7CCISC101.

[8] CISC103-Introduction to Computer Science with Web Applications, University of Delaware. Information is available at http://primus.nss.udel.edu/CourseDesc/ info.action?searchKey=2011%7CCISC103.

[9] CISC106-General Computer Science for Engineers, University of Delaware. Information is available at http://primus.nss.udel.edu /CourseDesc/info.action?searchKey=2011%7CCISC106.

[10] CMPSC097−397, undergradate programming courses, Penn State University. Information is available at http://bulletins.psu.edu/bulletins/bluebook/ university_course_descriptions.cfm?letter=C&dept=CMPSC

[11] CSC110 & CSC115, general education courses for nonmajors, Computer Science Department, West Chester University. Information is available at http://www.wcupa.edu/ _INFORMATION/OFFICIAL.DOCUMENTS/Undergrad.Catalog/ compsci.htm.

[12] CSE15−17, programming courses for CS major and minor, Computer Science and Engineering Department, Lehigh University. Information is available at http://www3.lehigh.edu/academics/catalog/html/index.html.

[13] iMote2. Information is available at https://docs.tinyos.net/index.php/ Imote2.

[14] Loop. Information is available at https://en.wikipedia.org/ wiki/Control_flow.

[15] Wii. Information is available at https://us.wii.com/.

[16] National security telecommunications and informationa system security (NSTISS). *National Training Standard for Information Systems Security*, June 1994. Document is also available at http://www.cnss.gov/Assets/pdf/nstissi_4011.pdf.

[17] A. Chua. *Battle Hymn of the Tiger Mother*, 2011. Penguin Press HC.

[18] F. Buschmann. *Pattern-Oriented Software Architecture*, 1996. Wiley.

[19] S. Frizell and R. Hubscher. Aligning Theory and Web-based Instructional Design Practice with Design Patterns. *Proceedings of E-Learning 2002: World Conference on E-Learning in Corporate, Government, HealthCare, & Higher Education*. pp. 298-304, 2002.

[20] W. Groesbeck and S. Delaney. Program correctness. Document is available at http://www.cse.unr.edu/∼bebis/CS365/StudentPresentations/ ProgramCorrectness.ppt.

[21] D. Jones and S. Sharonn and L. Power. Patterns: using proven experience to develop online learning. *Proceedings of ASCILITE'99*. pp. 155-162, 1999.

[22] J. Mache and N. Bulusu and D. Tyman. Making sensor networks accessible to undergraduates through activity-based laboratory materials. *Proc. of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08)*. pp. 606-608, San Francisco, CA, June 16- 20, 2008.

[23] Y. Mor. Design for learning: a pattern approach. presented in *the Workshop of the Chairs COnference on Instructional Technolgies Research: "Learning in the Technolgical Era"*. 2007. Document is also available at http://lp.noe-kaleidoscope.org/outcomes/chairs.