A Project-based Learning Approach for Junior Windows Software Programmer

Yu Liu
National Experimental Teaching Demonstration Center
CSE Department, Beihang University
Beijing, China
buaa_liuyu@buaa.edu.cn

Lei Yin
Training Center of Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
yinlei@tianbo.com.cn

***Abstract*—This paper presents an approach to train junior Windows software programmers for making software more appealing to students. This approach proposes to separate the software development technologies into three layers, which are the language layer, the application layer and the project layer. For each layer, based on the idea of PBL, this paper gives 20,12 and 1 projects to make the course interesting and easy to understand. Each student must finish a project named as LMS at the end of the course which contains many main technologies of Windows programming. According to the academic results and student surveys, most of the students can finish this course successfully and make a progress in software programming.**

Keywords-PBL;software programmer; Windows pogramming;

I. INTRODUCTION

Software programming is a branch of computer science and engineering(CSE). With the development of computer application, many people from other areas need to do some programming work and it is hard for the beginner to master the basic skills of software programming. Project-based learning (PBL) [1] appears as one of the most efficient and interesting instructional strategies [2] in the engineering education context. Related research in [3] and [4] tries to engage students in authentic real-world tasks. And the PBL strategy can significantly enhance the learning[5]. In this paper, we will represent a training course for junior software programmers base on PBL.

II. COURSE ARCHITECTURE

For junior software programmers, they always spend much time on programming language syntax. After that period, they often have the question that what can I do? I still cannot construct a simple software such as *Windows Calculator*. At the beginning of our training, we must tell the students that what is a Windows software and what you must know to be a software programmer. Figure 1 shows our course architecture.

We separate our course into three layers, which is shown in Fig.1, they are:

- **Language Layer**. The student must finish this layer at first. And we arrange 42 hours(7 days) at the language layer. For a programmer want to do some Windows applications, there are many languages to choose, such as C++,Java and C#. Here we use C++ for our training language because the C++ has two main features in the most popular programming languages. Firstly, the C++ is a *Middle* OOP language. In fact, C++ is a better C[6], which is the basic programming language. Almost all the student who want to be a software engineer have learned the C language by self or in the previous course. Secondly, the C++ has a more complex syntax then other OOP languages. For example, the friend relationship, the default parameter syntax and the pointer & reference syntax.
- **Application Layer.** This layer occupies 60 hours(10 days) in our course. After familiar with the C++ language, students will enter the application layer. In this period, we will tell them the how to build a featured program, such as DB application, network application and MT application. We choose MFC as our basic programming library since it is an old but still popular programming library and was written in C++.
- **Project Layer**. To be a *real* software engineer, the student should learn some *Software Engineering* issues. We plan a project layer at the end of our course. Because our course is mainly faced to those beginner, the purpose of this layer is only to introduce the basic issue in software engineering. There are 3 days for the student to browse the concepts and 5 days for them to finish the last *big* project.

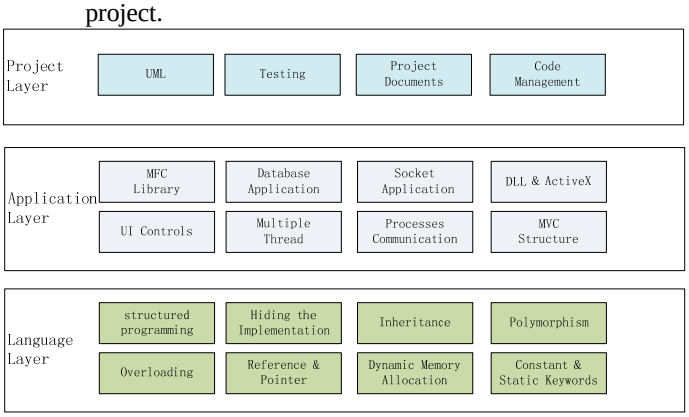

Figure 1 The course architecture for junior Windows programmer

We aimed to teach the student who want to be a software engineer the basic knowledge to build an application. After this course, a student should use the basic C++ syntax practiced and can read the complex code contains extended

syntax. And he should have the idea of how to build a useful Windows application and how to obtain assistance from documents and online references. Finally, he can build a real Windows application by himself.

III. PBL BASED COURSE DESIGN

A. Dummy Examples for Syntax

It is very hard to build a project to represent a proper syntax because the application context is too complex to let the learner focus on a simple syntax. The beginners will be confused with the multiple lines code. We built 20 example to show the C++ key syntax. Each example are based on a virtual background and we call them *dummy* examples. The examples should be easy to understand and as interesting as possible. Meanwhile, the examples should contain the key points of Data Structure and Design Patterns[7]. Data Structure is one of the most importance basic course in CSE and Design Patterns is the high level abstraction of OOP.

Here we show 5 typical examples.

- **List Structure**. This example is based on the *single list* structure and it is an importance bridge to contact the C and the C++ programming language. The student code a C style list firstly, which use the syntax *struct* and *pointer*. Then we replace the *struct* key word with the *class* key word. And we can propose the access control problem and introduce *public*, *private* key words to them. From this example, students may know that the *class* is almost same with the *struct* and familiar with the access control method.
- **Person Class**. We use the *Person* class to explain some syntaxes because this class is very visualized. Everyone knows that a person should have some attributes such as *age* and *name*, which are names as data members in OOP. And a person should have some behaviors such as *Go2School* and *BuyTicket*. We call those behaviors function member in OOP. Besides these concepts, the main purpose of this example is to show the initialization processes of the class. We use a *InitPerson* method at first and tell the student these method is very importance but easily to be ignored by the client programmer. Then we bring out the construction method to fix the pitfall. After this demo students can know the key ideas of *constructor*, *destructor* and *default constructor*.
- **IOEngine Class**. Suppose if we have a hardware device and we can do *Read* and *Write* operation with it. Before we do any operation, we must *Open* this device and *Close* it after the operation. In this context, users can instantiate many object when they want to communication with the hardware. But it is hard to maintain the *Open* and *Close* operations and it is meaningless to build many objects which are essential same. We use the *IOEngine* class to show the idea of *Singleton* design pattern and practice the using of *static* keyword and the
- **Pet Family**. This example is a very interesting and importance demo in the Language Layer. We build a basic class hierarchy contains a base class *Pet* and two derived classes, *Dog* and *Cat*. From these three classes students can learn the ideas of inheritance. Then we write another class which has a method called *Needle*. The *Needle* method need a parameter, both *Dog* and *Cat* may be the parameter so we use *Pet*. Things going on well except that in *Needle* function , the input object loss its type and the argument object acts the same as a *Pet* object, whether it is a *Dog* or a *Cat* instance. At this moment, we can give the polymorphism syntax and introduce the *virtual* key word. Then we draw the blue map of the *latter binding* and tell the student the *v-table* and *v-ptr* mechanisms.
- **MyStack**. We want to show the *template class* to the learner by this example. The stack is a basic type in Data Structure. We build a stack which contains a data pool typed as integer. And after that we will ask the student to build another stack for *double* type. It is a typical context to use the *template*. After these two demos, we show the usage of the classes in *STL* and give the ideas of how to choose a standard *template*.

B. Targeted Projects for Application Layer

From an executable application, a junior programmer can get a great sense of accomplishment. We arrange some simple but complete applications for each Windows application contexts.

Table 1 shows all the projects in the application layer. For details:

- We arrange a complex application in the first 3 days to show the basic ideas of Windows programming and MFC structure. This project is from [8] and we streamlined the application. After this demo, the students can build an application called *Drawing Board*, which is a simple *MSPaint*.
- For the database application context, we set up two projects, one get the DB connection via *ODBC* and the other based on *ADO*. These projects can make the main knowledge of DB application clearly.
- The *Process* and *Thread* are basic concepts in OS. We also set up two projects to introduce the MT application in Windows. In *ThreadClass* application, we use a derived class from the *CWinThread* class to meet our MT requirement. And use self-definition Windows messages for the start and stop controls of the thread.
- The socket programming is a very common field in generic applications. So we spend two days on TCP and UDP sockets. And in the projects *Chatting Room* and *File Trans*, we will use the *MT* knowledge in previous course.
- We give three projects to show the technology in *Process Communication*. They are based on the *Event*, *Share Memory* and *Named Pipe* methods.

- The dynamic linking library is an importance issue in Windows application. We set up three small projects in this field. The student should build a standard *DLL* firstly and call it by dynamic and static approaches.
- The UI design can representation the application directly. We choose some common demos from CodeProject[9] and tell the student how to build a UI with standard and self-definition controls.

TABLE I. TARGETED PROJECTS IN APPLICATION LAYER

Application Context	Targeted Project		
	Name	***Contents***	***Time(day)***
Windows Programming & MFC	Drawing board	Message based application. MVC structure Document-view structure MFC library introduction Usage of MFC classes Windows API	3
DB Application	ODBC App	ODBC concept Database and Recordset Insert, delete, update and select operations	1
	ADO App	ADO concept Exception handling COM application Database transaction	
MT Application	ThreadFun	Callback function Begin & end thread Suspend & resume Thread & UI	1
	ThreadClass	CWinThread class Self-definition msg	
Socket Programming	Chatting Room	UDP process Socket API	2
	File Trans	TCP process Socket API	
Process Communicati-on	App Manager	Event handling	1
	ShowPerson	Share memory	
	PipeDemo	Named pipe	
DLL	DLL	Build static library Build DLL Call DLL dynamic Call DLL by lib	1
UI&Controls	MyCombo	Self-definition controls Use other controls	1

C. *Library Management System*

After the courses of the Application Layer, students can obtain the main skills to build a Windows application. In order to check their ability and give them a review. We arrange a relatively large project called *Library Management System(LMS)*.The reason of choose this topic as our graduation design is everyone is familiar with the library workflow and we need not spend much time to make the business clear.

The requirements for *LMS* are:

- Borrow and return books. Borrowers can borrow and return books in *LMS*. All the user history should be recorded and can be queried. In order to meet this requirement , *LMS* must base on a database.
- User identification. The borrower has a *RFID* card which has already be recorded in the database. When he want to borrow or return a book, he must credit his card to identification. We give the student a *RFID* reader and some cards. And we also give them the manual and the *SDK* of the reader. This requirement need them to operate the *RFID* reader and practice to handle a strange hardware.
- Access control. At the entrance of our library, borrower should use his card to open the checking channels. We give the students the communication protocols with which to control the doors.
- The application should have simple and generous UI..
- The design and testing documents should be written.
- Finish this application in 5 days.

IV. EVALUATION RESULTS

The PBL based training course for junior Windows programmer started in 2008. During the five years, 98 people entered our training in 7 terms. 94 students had finished the course. In the 94 students, 40(44%) persons were majored in CSE or related majors. While others 54 students were from unrelated majors such as Geology, Management and Chemistry. The main reason for them to participate this course was that they need do some coding work in their daily works.

A. *Academic Results*

In order to evaluate the study result of this course. We invited 5 senior software programmers to value the graduation project for each students. Table 2 shows the methods of evaluation. We use 6 scoring point to evaluate the products, and we think the code and documents is the most important feature for junior programmers. Each student was valued respectively and given a mean score.

TABLE II. JUDGING RULES FOR LMS

Rated point	Rules	
	Concerns	***Scores***
Code & Document	Intuitive and succinct. Well structured. Detailed comments	30
Function	Finish all the features to meet with the requirements	15
Technology	Choose the most suitable technology to complete the function	15
UI	Use the proper controls. Professional.	15
UE	Easy to use.	15
Creativity	Self-definition features to make the application more useful	10

According to Fig. 2, only one student did not pass the graduation test. 5 persons in related majored group got the full score. While only 1 person did the same work in unrelated majored group. It means that the related education background is important for software programmer. The

trends of the two groups are almost same, it shows our course can well fit the requirements of junior software programmers.

B. *Student Surveys*

After the training, we asked the students some questions like these:

- [Q1]The course has been interesting.
- [Q2]The course has increased my ability in software programming.
- [Q3]I think I can learn other software programming technologies by myself in the future work.

To answer these questions, every student had to choose between six different answers with a numerical value: I fully agree (5); I agree (4); I partially agree (3); I partially disagree (2); I disagree (1); and I fully disagree (0). For the statistics of 94 students, the average values are [Q1] = 3.8, [Q2] = 4.1 and [Q3] = 3.9. We are very pleasure for that [Q3] get 3.9 scores because a main purpose of our training is to tell the students how to handling the problems in software programming.

V. CONCLUSION

This paper presents an approach to train junior Windows software programmer based on PBL and provides a detailed description of the course structure, the designing of targeted projects, and the evaluations completed in the last five years. The junior Windows software programmer training course has been separated into three phase includes *Language Layer*, *Application Layer* and *Project Layer*. In the Language Layer, 20 dummy examples are designed. In the second phase, the course contains 12 projects to practice the common technologies in Windows programming area. At the end of the training, there is a bigger project named *LMS*, which combines many technologies to check the students' learning result. This paper also given a method to evaluate the last five years training results. The students have increased their interest in software programming, the curriculum characteristics have permitted the students to acquire advanced knowledge and skills to develop more complex software systems and learn new related technologies.

ACKNOWLEDGMENT

The authors would like to thank all members of the National Experimental Teaching Demonstration Center at Beihang University. Especially to thank Prof. Cao Qinghua at Beihang University and Prof. Wang Jianhua at CAS for their comments and fruitful discussions during the past years that made this work possible. This work has been supported by the National Natural Science Funds of China(61202238).

REFERENCES

[1] G. Solomon, "Project-Based learning: A Primer," Technol. Learn., vol.23, no. 6, pp. 20–30, Jan. 2003.

[2] M. Hedley, "An undergraduate microcontroller systems laboratory," IEEE Trans. Educ., vol. 41, no. 4, pp. 345–353, Nov. 1998.

[3] H. Markkanen, G. Donzellini, and D. Ponta, "NetPro: Methodologies and tools for project based learning in internet," in Proc. World Conf. Educational Multimedia, Hypermedia Telecommunications (ED-MEDIA 2001), Chesapeake, VA, 2001, pp. 1230–1235.

[4] D. Ponta, G. Donzellini, and H. Markkanen, "NetPro: Network based project learning in internet," in Proc. Eur. Symp. Intelligent Technologies, Hybrid Systems Their Implementation Smart Adaptive Systems 2002, Albufeira, Portugal, 2002, pp. 703–708.

[5] S. A. Ambrose and C. H. Amon, "Systematic design of a first-year mechanical engineering course at Carnegie-Mellon University," J. Eng. Educ., vol. 86, pp. 173–182, Apr. 1997.

[6] Eckel,B.,Thinking in C++,2nd Ed.,vol.1.NJ: Prentice Hall.2000, pp.71-72.

[7] M. Fayad, D. Schmidt, "Object-Oriented Application Frameworks," Comm. ACM, Oct. 1997, pp. 32-38.

[8] J. J. Hou Dissecting MFC 2nd Edition using Visual C 5.0 & MFC 4.2, Taiwan:Unalis Corporation, 1997, pp.101-150.

[9] http://www. codeproject.com, 2008.

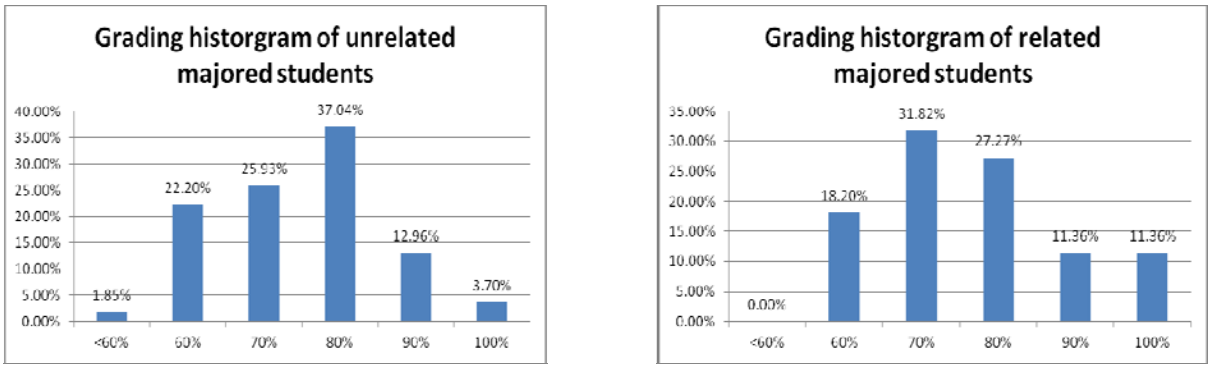

Figure 2. Academic results for : (a) unrelated majored students (b) related majored students.