International Journal of Innovative
Computing, Information and Control
Volume 9, Number 8, August 2013

ICIC International ©2013 ISSN 1349-4198
pp. 3415–3436

BUILDING COMMUNICATION SOFTWARE: A PROJECT-BASED APPROACH FOR TEACHING C++ OBJECT-ORIENTED PROGRAMMING

Yen-Lin Chen[1], Chuan-Ming Liu[1], Chuan-Yen Chiang[2]
Shyan-Ming Yuan[2] and Jenq-Haur Wang[1,*]

[1]Department of Computer Science and Information Engineering
National Taipei University of Technology
No. 1, Sec. 3, Chung-hsiao E. Rd., Taipei 10608, Taiwan
*Corresponding author: jhwang@csie.ntut.edu.tw

[2]Department of Computer Science
National Chiao Tung University
No. 1001, University Road, Hsinchu 30050, Taiwan
smyuan@cs.nctu.edu.tw

Received July 2012; revised November 2012

Abstract. *This paper presents a project-based remedial curriculum for teaching the C++ programming language, as well as object-oriented programming (OOP) skills and concepts. The pedagogical approach of the proposed curriculum comprises of a set of homemade projects to assist students in learning essential C++ and OOP skills quickly by accordingly implementing a large-scale communication software system. Based on constructivist learning technology, the proposed project-based curriculum can effectively enhance the learning effectiveness and interests of students via hands-on, minds-on, and learning-by-doing practices related to their lives. Based on the student survey and grade assessment results, the proposed project-based curriculum and practical homemade projects demonstrate effectiveness and feasibility in motivating student to enhance C++ and OOP skills and incorporate these skills for developing practical, large-scale software.*
Keywords: Computer science education, Project-based approach, Communication software, Object-oriented programming (OOP)

1. **Introduction.** In contemporary information and communication technology (ICT) industries, language programming and object-oriented programming (OOP) have become essential skills for engineers graduating from computer science (CS) related departments. Programming courses are generally offered in the undergraduate curricula of computer science related departments [1,2]. Whereas fundamental OOP features are usually taught in conventional OOP courses, the ability to develop a large-scale software system to solve practical and technical problems is still difficult for students to learn and develop. Therefore, a practical lab-based or project-based curriculum is necessary for developing student abilities to incorporate various OOP programming skills in designing large-scale software [3].

Courses in various object-oriented programming languages, such as C/C++, C#, Java, and Delphi, are widely offered in contemporary computer science and engineering curricula [4]. The conventional OOP courses might adopt the largest share of course time for teaching the essential topics of OOP, including data types, string and I/O processing, class design, inheritance, and generic programming technicalities [5]. To assist students in learning OOP skills and concepts, computer-assisted educational and learning systems are developed to assist course instructors in enhancing teaching performance [6,7]. However,

although students might have chances to practice some of the aforementioned OOP concepts and technicalities in a traditional programming curriculum, they mostly practice the concepts and technicalities via isolated and small-scale programming exercises (no more than hundreds of code lines) throughout the entire curriculum. Moreover, in a typical computer science curriculum in *Computing Curricula CC2001* [1,2], the advanced course for training students to implement relatively large-scale software is generally offered in the Capstone Project or Major Project in the senior or junior year, whereas the basic programming courses are given in the first two to three semesters. Therefore, some gaps exist in the sophomore year of students developing abilities to incorporate programming skills for designing and completing a software application.

According to the constructivist learning technology from epistemological theory [8], the hands-on, minds-on, and learning-by-doing practices can significantly promote the learning effectiveness and interests in some technological courses, such as the autonomous robotic laboratory [9], high performance computer networks [10,11], computer architecture and embedded system courses [12,13], and control system engineering education [14]. To resolve the aforementioned problems in the programming curriculum, project-based teaching approaches based on the constructivist learning technology, which involve practical projects related to students' lives and interests, are presented to improve the students' learning performance and interests in the programming courses [15-17]. Based on the attraction and popularity of computer games among students, Chen and Cheng [15] presented an object-oriented framework for computer game programming, and utilized this programming framework to design a project-based OOP laboratory to instruct the sophomore students to incorporate and enhance their programming skills after enrolling in the fundamental programming courses. In Zhu's OOP teaching approach [16], the financial literacy of students' lives, including student loan concerns, savings and debt management, and mortgage calculation topics, are arranged into a project-based Java OOP course to help students study the programming skills toward overcoming life's obstacles. Pérez and Rosell [17] proposed a goal-directed course for students to design a robotic motion planning software, offering students opportunities to practice and incorporate C++ programming skills, opensource software libraries, and development tools.

Because of the industrial trends of information and communication technology (ICT) in the authors' country, industrial employers primarily require software engineers having the abilities to program communication software, embedded system software, device drivers, and multimedia signal processing applications of consumer electronics products. The C++ programming language is essential for students seeking technical jobs, because C++ language technicalities can provide not only high-level programming features, such as object-orientation paradigms and generic programming, but also low-level programming features, such as bit-wise and device I/O control abilities [18,19]. Although the steep learning curve of the C++ programming language may diminish students' interests and motivations while performing coursework, relevant studies on four-year undergraduate computer science curricula, such as CC2001 [1,2] and project-based programming laboratory courses [15], can offer sufficient training for students to practice C++ programming and OOP skills.

However, in each academic year, many students (including domestic and international students) who hold non-CS bachelor degrees are admitted into the graduate program in the authors' department (i.e., the Department of Computer Science and Information Engineering at National Taipei University of Technology, Taiwan). These non-CS admitted students may enroll in basic programming courses from other engineering-related departments, but those who lack sufficient CS experience did not have qualified and sufficient training in C++ and OOP programming skills for their graduate studies and theses.

Therefore, to satisfy these programming skill requirements, students without CS experience need a supporting and remedial curriculum to strengthen their C++ programming and software design skills quickly and efficiently.

To fulfill the aforementioned demands, this study designed a project-based curriculum to rapidly instruct these students in major C++ and object-oriented programming (OOP) skills, as well as to instruct them in implementing a set of homemade projects, toward eventually constructing a large-scale software system.

The motivation of this paper is based on a concept that the students may more desire to put more efforts on implementing some useful tools that can facilitate their works (i.e., the graduate thesis studies) during their course studies. In this sense, the objective of the proposed curriculum mainly focuses on allowing the students to learn essential OOP concepts and problem-solving skills, rather than simply teaching a programming language. During their thesis studies, the students may need some convenient tools to collect the communicated information, share their discussion records, and research materials with their class and lab mates, or advisors. Therefore, through a set of homemade projects with various programming solutions toward a consistent objective (that is, creating their own multimedia communication application for supporting their thesis studies), students can rapidly develop problem-solving skills and learn how to apply practical solutions using different OOP concepts. Furthermore, based on the cooperative learning concepts via the knowledge-sharing portal, blogging, and social networks [20,21], an interactive course web forum based on Web 2.0 social-network technology is also established. This interactive course approach can assist students in conveniently obtaining the announcements of course resources and project assignments, and in posing questions for seeking help from and discuss with the TAs and instructor, as well as immediately providing feedback and sharing knowledge of programming skills and concepts.

Based on the above-mentioned features, the major innovation and significances of the proposed project-based curriculum and interactive course forum are:

1) Effectively arousing the students' interests in problem-solving skills and paradigms with OOP;
2) Providing high-quality chances for them to learn the advantages of OOP on implementing their own tools that can facilitate their thesis works;
3) Promotion on students' learning efficiency practical software engineering.

According to the student survey and grade assessment results, the proposed project-based curriculum and practical homemade projects demonstrate the effectiveness and feasibility of the proposed curriculum in motivating student to apply and incorporate C++ and OOP skills for developing practical, large-scale software. Moreover, the proposed project-based curriculum not only can apply in teaching C++ OOP, and its project-based concept and pedagogic methodology can widely be adapted for teaching many practical programming courses in different, such as embedded system programming, network programming, and mobile application programming with C++, C#, Java, or Objective-C programming languages.

The rest of this paper is organized as follows. Section 2 introduces the course materials adopted in the proposed project-based curriculum. In Section 3, the proposed project-based curriculum design for the students to develop multimedia communication software is introduced in detail. The results on the proposed project-based curriculum are demonstrated and discussed in Section 4. Finally, Section 5 offers the conclusions to this paper.

2. **Course Materials.** To facilitate the students' learning process in designing large-scale software applications using the C++ programming and object-oriented programming

(OOP) skills, the Unified Modeling Language (UML) [22] proposed by the Object Management Group (http://www.UML.org) is also introduced and applied in the project-based curriculum. The UML modeling process is very useful and feasible for software project management and design in different aspects, such as designing the software components of the configurable system frameworks [23,24], and modeling and implementation for the distributed repositories of learning objects in service-oriented architecture on e-learning systems [25]. Therefore, through UML modeling and analysis, the students can learn how to visualize, specify, construct, and describe the analytical architectures and system modifications while developing software-intensive systems. Thus, the UML provides students with an approach for considering problems using models based on real-world ideas.

In this curriculum, several useful UML diagrams are introduced for students to apply in analyzing, modeling, and developing their own multimedia communication software projects, including class diagrams, use case diagrams, and sequence diagrams. In coordination with the instructive lectures on the OOP, the use case diagrams can assist students in learning how to analyze and plan for the requirements and functional behaviors of a particular software system. Then class diagrams are then introduced to assist students in identifying and organizing the structural and behavioral characteristics of the provided classes of objects, which are combined to construct a large software system. The structural characteristics can describe the attributes and respective associations of the classes, whereas the behavioral characteristics can express their cooperative operations and methods. To express elaborately the interactions and communications among the class objects in a software system, the sequence diagrams are also discussed to help students define and illustrate how interactive events and messages are sequentially transmitted and received for certain class objects in particular use cases. The three aforementioned major UML diagrams are the most frequently used and are capable of describing and modeling most designs and implementations of multimedia communication software in this curriculum. Along with learning the UML diagrams, UML graphical design tools, such as StarUML (http://staruml.sourceforge.net/) or Microsoft Visio (http://office.microsoft.com/en-us/visio/), are extremely useful for students to create some initial software designs by illustrating their ideas according to the aforementioned UML diagrams, as well as to refine and extend the initial designs to those that are more elaborate.

Because concurrent software generally requires a graphical user interface (GUI) to interact with users, it is also essential to teach students how to develop their own GUIs associated with software projects. For this purpose, a GUI programming framework comprising sufficient OOP implementation features and which is released as opensource libraries should be the satisfactory choice for supporting an OOP software development curriculum. The Microsoft Foundation Class (MFC) library in Windows is an effective example of strong OOP features, including encapsulation, inheritance hierarchy, and polymorphism. However, adapting the MFC on different platforms and operating systems is not easy. Because multimedia communication software is usually developed and applied on handheld platforms, cross-platform portability is particularly essential.

Among these requirements, the Qt library [26] (http://qt.nokia.com/) and FLTK tool library (http://www.fltk.org), which are originally offered in Unix/Linux environments and can also be ported and applied on different OSs (such as Windows and Mac OS), can provide not only the desired OOP features but also source-opened features. Thus, the two aforementioned opensource GUI framework libraries are adopted in this curriculum to support students in learning the OOP features and developing the GUI in their software projects. In addition, because a high-quality code project should accompany appropriate documentation and comments, a code-commenting tool is also required. For this purpose,

the Doxygen (http://www.stack.nl/~dimitri/doxygen/index.html), a cross-platform code documentation tool, is introduced for students to practice effective programming styles and habits by commenting their codes. Using the Doxygen tool, students can easily document their software project codes by simply adding their comments within the codes using a specific mark; thus, well-organized code documentation can be automatically produced for review and refinement.

Furthermore, student learning performance can be promoted further via the knowledge-sharing portal, such as blogging and social networks, as demonstrated in previous empirical studies [20,21]. Therefore, for immediate feedback and knowledge sharing, an interactive course web forum is based on the Web 2.0 social-network technology of Facebook (http://www.facebook.com/). The students can conveniently obtain the announcements of course materials and project assignments, as well as pose questions and quickly seek help from and discuss with the TAs and instructor. In addition, the TAs and instructors can also obtain the solutions of most common programming problems and skills, and then announce these solutions and related experiences in the interactive course web forum.

3. **Curriculum Design and Homemade Project Assignments.** This section introduces the proposed project-based curriculum design for students to practice effectively and faithfully all of the general and critical OOP tasks of designing applicable multimedia communication software. This goal can be achieved in class lectures discussing the major elements of OOP, as well as in correlated homemade project assignments. The overall task for developing multimedia communication software is arranged into seven successive project assignments, so that the students can gradually design and implement a large-scale software system by applying their learned OOP programming skills. The students will then create fully functional multimedia communication software at the end

TABLE 1. Curriculum topics and project assignments

Major Topics	**Contents**	**Homemade Project Assignments**
Basic Programming Skills	Basic data types, variables, strings, computations, error handling, functions, source code files	Homemade Project 1
Input and Output Programming	Console mode I/O, file I/O, I/O streams, string processing,	Homemade Project 2
Object-oriented Programming	Class design, UML class diagrams, encapsulation, inheritance hierarchy, subtype polymorphism	Homemade Project 3
Graphical User Interface (GUI) Programming	GUI tool libraries, GUI widgets, GUI design, event-driven based programming	Homemade Project 4
Generic data processing and network programming	Templates, data containers, algorithms, the standard template library (STL), network communication programming	Homemade Project 5
Software Engineering	Software testing and maintenance, UML modeling diagrams for OOP, Basic Design Patterns	Homemade Project 6
Network and Mobile Communication Software Design	Real-time communication software, advanced GUI implementation, software integration	Homemade Project 7

of the curriculum. The proposed curriculum is arranged into seven major lecture topics and seven homemade project assignments, as listed in Table 1.

In the semester, we spend the first month introducing basic C++ programming skills and input and output programming topics, including basic data types, variables, string processing, numerical computations, error handling, functions, managing source code files, I/O streams, and file I/O processing skills [18,19]. We then spend a majority of the class lectures addressing the major topics of object-oriented programming. In addition, topics of OO-based GUI programming libraries and software engineering [22,26] are introduced to the students. This includes details of class design, encapsulation, inheritance hierarchy, polymorphism, and using opensource GUI tool libraries (Qt and FLTK libraries) to facilitate learning large frameworks with OO paradigms, as well as teaching the students to apply software engineering tools and concepts to developing their own software systems. Next, to help the students manage large amounts of data and the reusability of code components in large software systems, we also explain the data structures and generic programming skills using the standard template library (STL) [27]. Finally, the students are asked to apply their learned OOP skills and software engineering concepts to creating their own multimedia communication software systems for their livelihood applications during graduate studies, which is the goal of this course.

The homemade projects assigned in the proposed curriculum are addressed as follows.

Project Assignment 1: Implementing Text-Parsing Application. In this first homemade project, students implement a text-parsing application to practice fundamental C++ programming skills. This text-parsing application must have the functions necessary to record and parse the text contents of a real-time interactive chat, including the names of the members having joined the chat, their corresponding talk message sentences typed in the chat console, and the time records of each talk message given by someone who has joined the chat. To facilitate chat recording, students are also requested to transform and save accordingly the chat records into the XML formatted file, as shown in Figure 1(a). The XML file should also be able to be correctly loaded and parsed for displaying the formatted chat record content on the console mode user interface, as shown in Figure 1(b).

To implement this simple text-parsing application, students can practice the required basic programming skills, including string processing and manipulation, function design, basic error handling, and file and console mode I/O stream processing. Consequently, students will develop a basic software application using file I/O, text string parsing functions, and a console-mode user interface.

Project Assignment 2: Text Searching & Retrieval Functions. In the second homemade project, students implement new fundamental functions of the messenger communication system based on the previous project they had completed, including a search function with simple query syntax. The goal of this project is to provide information searching and retrieval functions of the desired textual data from the XML files of the archived chat records from the previous project. A search syntax must be implemented to perform information retrieval functions within various fields of message records (including message speaker and receiver, message contents, and message timing records) using field tags and logical operators along with search keywords, as illustrated in Figure 2(a). Here, the field tags are utilized to specify the contents of the given record fields to be searched (without case sensitivities) and are defined as follows:

Attendee-Name tag ([name]):

The *attendee-name* tag is utilized for searching a keyword within the field of the chat attendee name records, and its syntax is defined as: [name] "keyword string of an attendee's name".

Attendee-Role tag ([S], [R], [RS] or [SR]):

The combination of the *attendee-role tag* with the *attendee-name* tags can be applied for searching the chat records that contains the users who are attending with the same chat in different roles. Users can add "[S]", "[R]" or "[SR]" (or "[RS]") at the left of a given keyword of an attendee name to indicate who had acted as a message speaker, a receiver, or both, respectively.

Message tag ([message]):

Using the message tag is similar to using the nametag. By specifying the message tag (i.e., [message] "keywords or incomplete sentences"), the user can use some partial keywords or incomplete sentences to search for the recorded messages having the desired keywords or sentences, and the incomplete sentences can comprise multiple fragmented words.

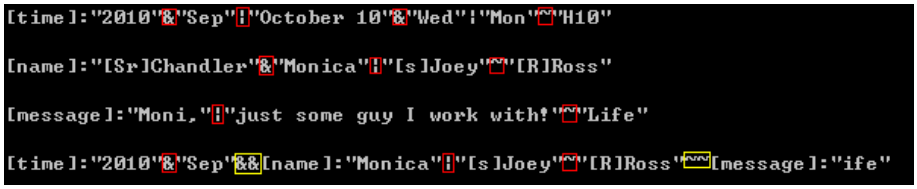
```
- <MSN>
  - <Chat>
      <date>Mon Sep 21 10:21:43 2010</date>
      <From>Chandler</From>
    - <To Friend_Name="2">
        <Friend_Name>Joey</Friend_Name>
        <Friend_Name>Monica</Friend_Name>
      </To>
      <message>All right Joey, be nice. \n So does he have a hump? A hump and a hairpiece?</message>
    </Chat>
  - <Chat>
      <date>Mon Sep 21 10:22:02 2010</date>
      <From>Phoebe</From>
    - <To Friend_Name="3">
        <Friend_Name>Chandler</Friend_Name>
        <Friend_Name>Joey</Friend_Name>
        <Friend_Name>Monica</Friend_Name>
      </To>
      <message>Wait, does he eat chalk?</message>
    </Chat>
  - <Chat>
      <date>Wed Oct 10 10:22:02 2010</date>
      <From>Rachel</From>
    - <To Friend_Name="5">
        <Friend_Name>Chandler</Friend_Name>
        <Friend_Name>Joey</Friend_Name>
        <Friend_Name>Monica</Friend_Name>
        <Friend_Name>Phoebe</Friend_Name>
        <Friend_Name>Ross</Friend_Name>
      </To>
      <message>It's my life. Well maybe I'll just stay here with Monica.</message>
    </Chat>
  - <Chat>
      <date>Wed Oct 10 10:22:41 2010</date>
      <From>Monica</From>
    - <To Friend_Name="5">
        <Friend_Name>Chandler</Friend_Name>
        <Friend_Name>Joey</Friend_Name>
        <Friend_Name>Phoebe</Friend_Name>
        <Friend_Name>Rachel</Friend_Name>
        <Friend_Name>Ross</Friend_Name>
      </To>
      <message>Well, I guess we've established who's staying here with Monica...</message>
    </Chat>
</MSN>
```

(a) Snapshot of the sample chat records in XML format

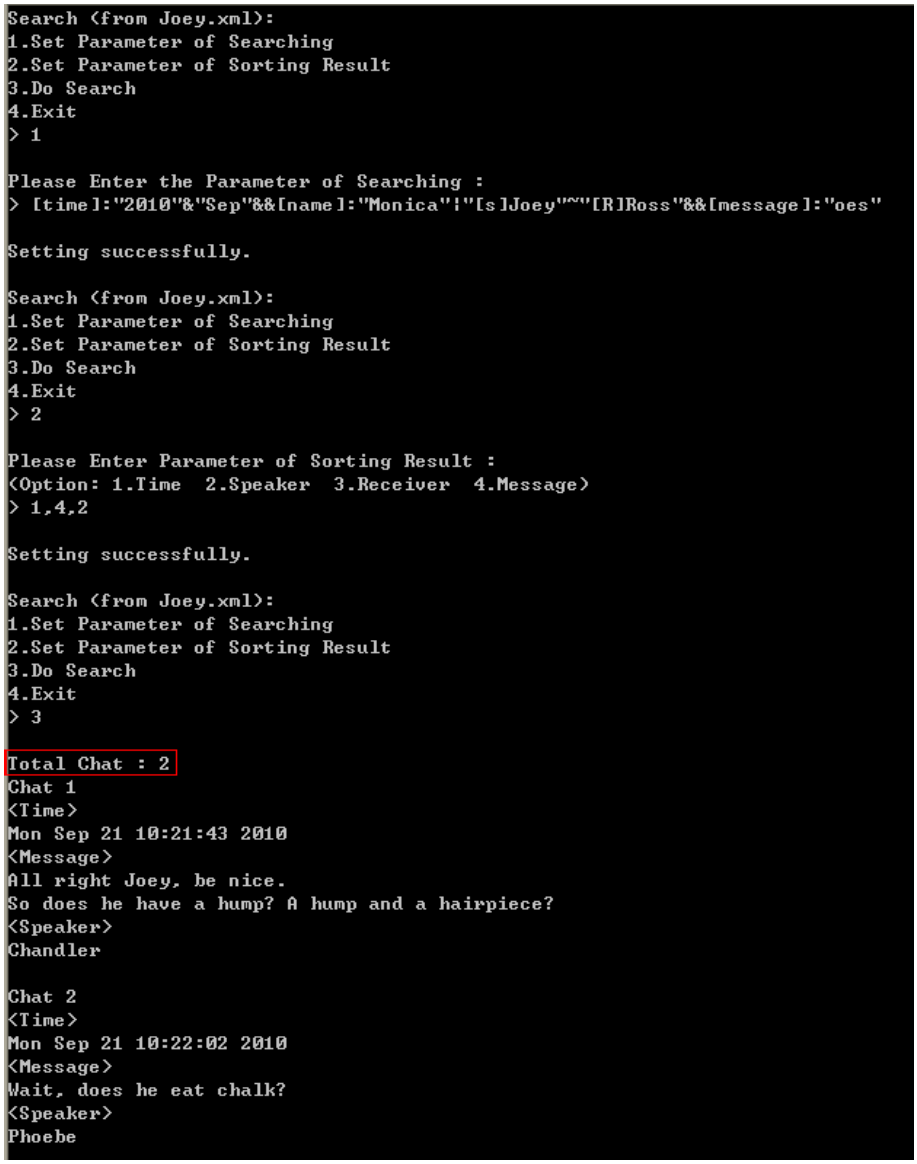
```
MSN Chat:
1.Enter the speaker's name
2.Enter the receiver's name
3.Chat
4.Save
5.Exit
> 3

Please Enter the Chat's content
> There's nothing to tell! \n He's just some guy I work with!
```

(b) Snapshot of the console mode user interface

FIGURE 1. Snapshots of the text-parsing application in the first homemade project



(a) Snapshots of the sample usage of the field tags and logical operators



(b) Snapshots of performing text search and retrieval functions in the second homemade project

FIGURE 2. Snapshots of the text search and retrieval functions requested in the second homemade project

Time tag ([time]):

The time tag is utilized for searching any message records in a particular period. Because the time records are stored in "⟨date⟩" fields of XML files, the students must implement formatting criteria to search for the desired times appropriately. For example,

when the user enters the search command: [time] "2011" & "Jun" & "Sun", the application should search and report any message records on any Sundays in June in 2011 (where "&" is a logical operator that is described later).

The logical operators are then adopted to search more specifically for the conditions of the desired contents, including "And", "Or", and "Not" operators. Students must also allow these three operators to have computing priority in the manner of the fundamental operations of arithmetic; that is, the Not operator should have the highest priority, followed by the And operator and the Or operator, who has the lowest priority. These logical operators are described as follows,

And operator (&, &&):

"**&**": This binary operator reveals that a pair of keywords or sentence fragments to the left and right of the "&" must simultaneously exist in the contents in a given field specified by a field tag.

"**&&**": Using this operator is similar to using "&", but the double of And operators is utilized to associate the specified keywords or sentence fragments in different fields of various tags.

Or operator: (|, ||):

"|": This binary operator means that the search results from the contents in a specified field should contain any of the paired keywords or sentence fragments to the left and right of the "|".

"||": Similarly, the double Or operator "||" is applied to obtaining search results with any of the paired keywords or sentence fragments from different fields of various tags.

Not operator:($\sim$, $\sim\sim$)

"$\sim$": This unary operator requires that the search results should exclude the contents having any keyword or sentence fragment specified by a "$\sim$".

"$\sim\sim$": Similarly, the double Not operator "$\sim\sim$" is used for searching the results excluding those with the specified keyword or sentence fragment in any different fields of various tags.

Accordingly, the students are requested to implement the aforementioned search and retrieval functions by parsing and processing the XML files of chat records, and integrate these functions using an enhanced console user interface to respond to the user command lines and display the processing results, as shown in Figure 2(b).

Project Assignment 3: Using OOP Skills and Exception Handling. In the third homemade project, students have the chance to practice the OOP skills and concepts to rework and re-organize their developed programs from the two previous projects; thus, the students can learn how to solve programming problems using essential OOP skills. Accompanying the course lectures on the basics of object-orientation and Universal Modeling Language (UML) class diagrams, students are asked to convert their procedural codes (i.e., the text parsing and searching functions implemented in the previous two assignments) to OO implementation, as depicted in Figure 3. Moreover, students are also required to implement some exception handling classes in their developed classes to provide error message reporting and handling functions.

Based on the code fragments shown in Figure 4(a), students should implement a class "chat" that integrates the text file processing and XML parsing functions they implemented in the first assignment. The students are then required to design a set of hierarchical classes to provide different ways to apply search functions, such as searching the chat records according to time, name, and message contents, by implementing a base class "DoSearch", as well the sub-classes "DoTimeSearch", "DoNameSearch", and "DoMessageSearch" for conducting specific search functions by practicing the OO skills of inheritance and polymorphism, as depicted in Figure 4(b).

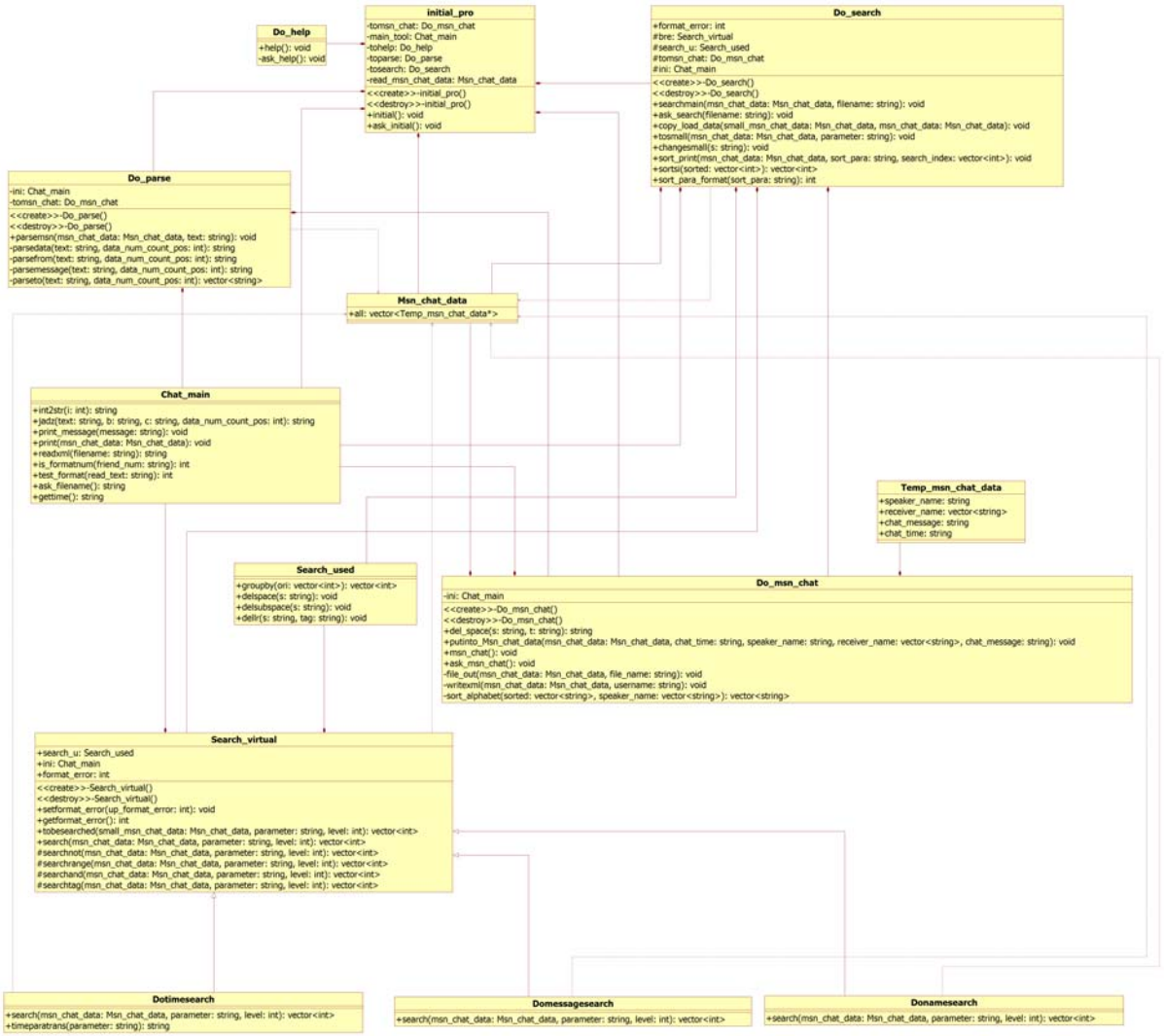


FIGURE 3. Snapshots of the UML class diagram of hierarchical classes of text parsing and searching functions

Regarding the exception handling practices, students must implement two sets of exception classes, and implement corresponding error-recovery functions for XML parsing and text searching functions of the designed classes, thus practicing OO skills in handling program errors. For example, for the XML parsing functions, when parsing the XML file with lost or erroneous tags, the program should raise corresponding exceptions and tell users which tags are lost or erroneous, as well as where they are. For the searching functions, when the user types an incorrect search notation, the program should respond to explain where and what the incorrect logic notation is. In addition, when the user enters an incorrect search range (e.g., inappropriate ranges of "time", "name", or "message"), the program should raise associated exceptions and respond by stating that the search cannot be conducted and inform users of the erroneous range. After raising exceptions and response error messages, the program should recover from the errors and be able to execute functions correctly.

Project Assignment 4: Developing a Graphical User Interface (GUI). Students are often interested in developing GUI applications, which can be conveniently implemented via the Qt opensource GUI tool library using OO programming concepts. In the course lectures, several sample programs are provided for students to become quickly

```
class chat{

    //...

private:
        string time;
        string speaker;
        string reciever;
        string message;

    // ...
};
```

(a) Sample code fragments of the class "chat"

```
class DoSearch{
    //...
public:
    virtual void search() = 0;

protected:
    logic_not();
    logic_and();
    logic_or();
    //...
};

class DoTimeSearch : public DoSearch{
    //...
public:
    void search();
    //...
};

class DoNameSearch : public DoSearch{
    //...
public:
    void search();
    //...
};

class DoMessageSearch : public DoSearch{
    //...
public:
    void search();
    //...
};
```

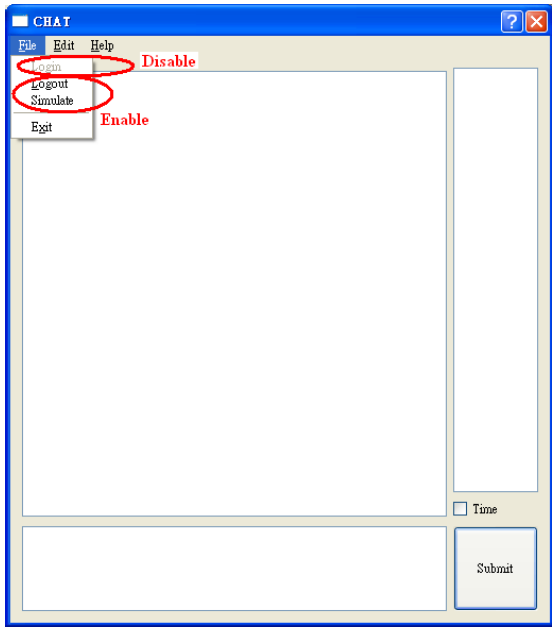
(b) Sample code fragments of the base class "DoSearch" and its sub-classes

FIGURE 4. Snapshots of OO implementation of text parsing and searching functions in the previous homemade projects

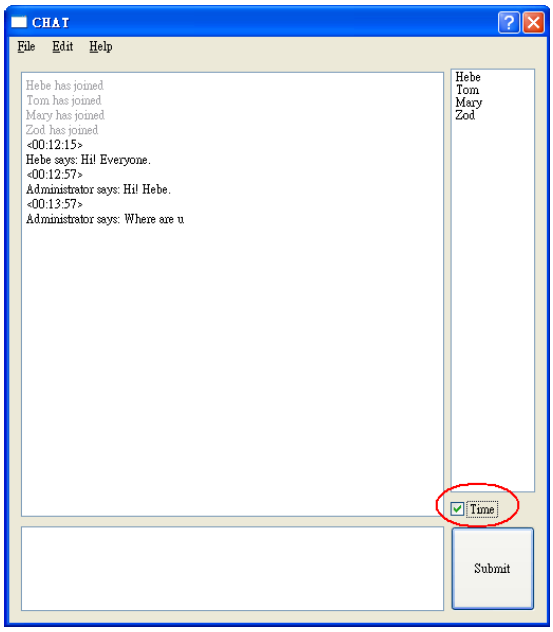
familiar with how to apply the essential Qt classes and APIs for building their own GUI applications. Moreover, the Model-View-Controller (MVC) design pattern is also introduced for students to elaborate their GUI applications. Through this practice, students can have a deeper understanding of the OOP concepts in designing a software system that interacts with various users.

In this homemade project, students are required to implement a set of cooperative windows with corresponding GUI widgets, including chat windows, chat invitation windows that interact with each other, and content search windows, as shown in Figure 5. First, the students must design their main chat windows using text input and output, a "submit" button, and a tool menu with required action items, where the submit button and functional items should be appropriately enabled and disabled under different conditions, as depicted in Figure 5(a) and 5(b). For instance, the submit button can only be enabled if the user has logged in (as Figure 5(b) depicts), and the "Login" and "Logout" functional items should be toggled as disabled or enabled once the user logs in or logs out, respectively (Figure 5(a)).

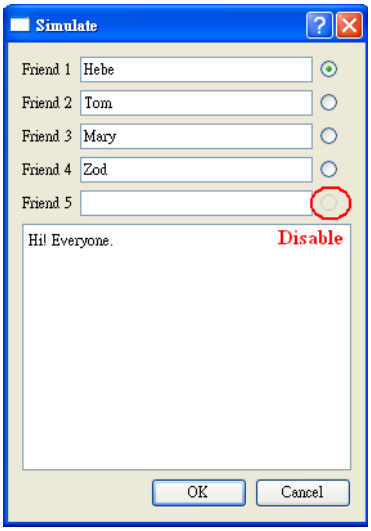
To invite other class friends to a new discussion chat, a new-chat invitation window, which can be invoked using the functional item in the main chat window, should also be implemented. Such a new-chat invitation window should provide some selection functions to allow users to choose the friends whom they want to invite by applying GUI widgets (such as text boxes and radio buttons), as illustrated in Figure 5(c). To practice text data processing skills further, the search functions implemented in the previous assignments



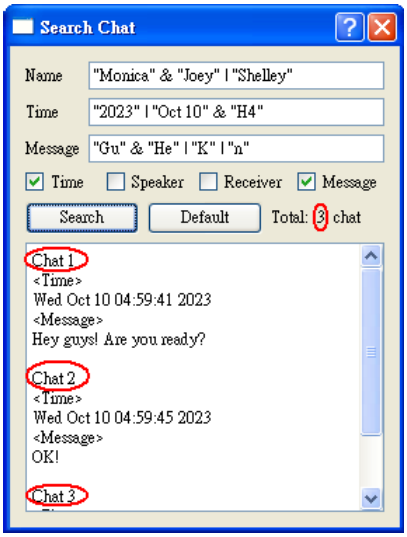
(a) Sample user menu of main chat window



(b) Main chat GUI window used during discussion



(c) New-chat invitation window invoked using the main chat window



(d) Search window invoked using the main chat window

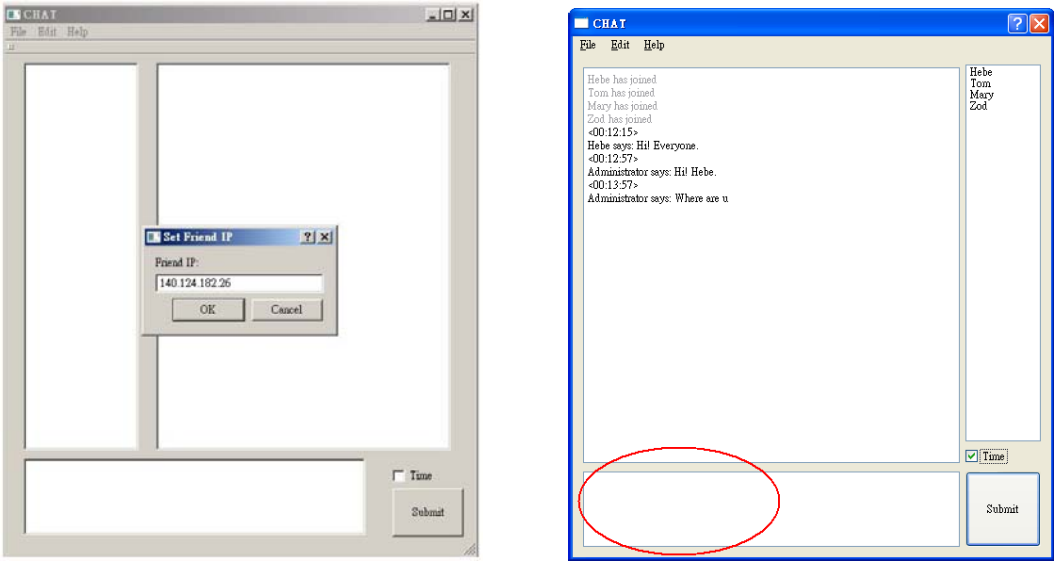
FIGURE 5. Snapshots of GUI implementation of the chat windows using Qt and MVC patterns

must also be integrated into students' chat GUI as a search window, which can be invoked using the menu item of the main chat window, as depicted in Figure 5(d). Here, the search window should offer diverse criteria for users to perform a search action in one or more record fields (including names, times, and message contents) using the searching syntaxes implemented in the second assignment. The search result display of the chat records, which include times, speaker roles, receiver roles, and message contents, can be customized by selecting the corresponding field tags, as illustrated in Figure 5(d).

Project Assignment 5: Implementing Network Communication via Socket Programming. In this project assignment, students have the opportunity to practice

exploiting the socket programming techniques to transfer messages and data between users via the network communication. To implement network communication functions, students may consider integrating the socket classes provided by Qt or MFC class libraries into their projects. Moreover, this project also accompanies the course lectures on the UML diagrams and the standard template library (STL) [27]. Thus, students have opportunities to adopt the use-case and sequence diagrams to determine the requirements and flows of the network communication functions, as well as to apply the appropriate STL containers and algorithms in solving the message queuing problems associated with network communication.

In this sense, students should implement some protocols and processes for making a connection between users. When a connection for transmitting messages between users is begun, the message speaker should send a formatted request string that includes the receiver's IP address and username using the created socket; after receiving the request string, the message receiver should then send a formatted response string that includes the speaker's IP address and username. Thus, a connection between two users is created, and users can begin talking by utilizing formatted strings to specify the speaker, receivers, and message contents transmitted via the connected socket. Figure 6(a) shows the snapshot of the user IP setting window, and Figure 6(b) exhibits the snapshot of the chat message transmission after the connection being created.

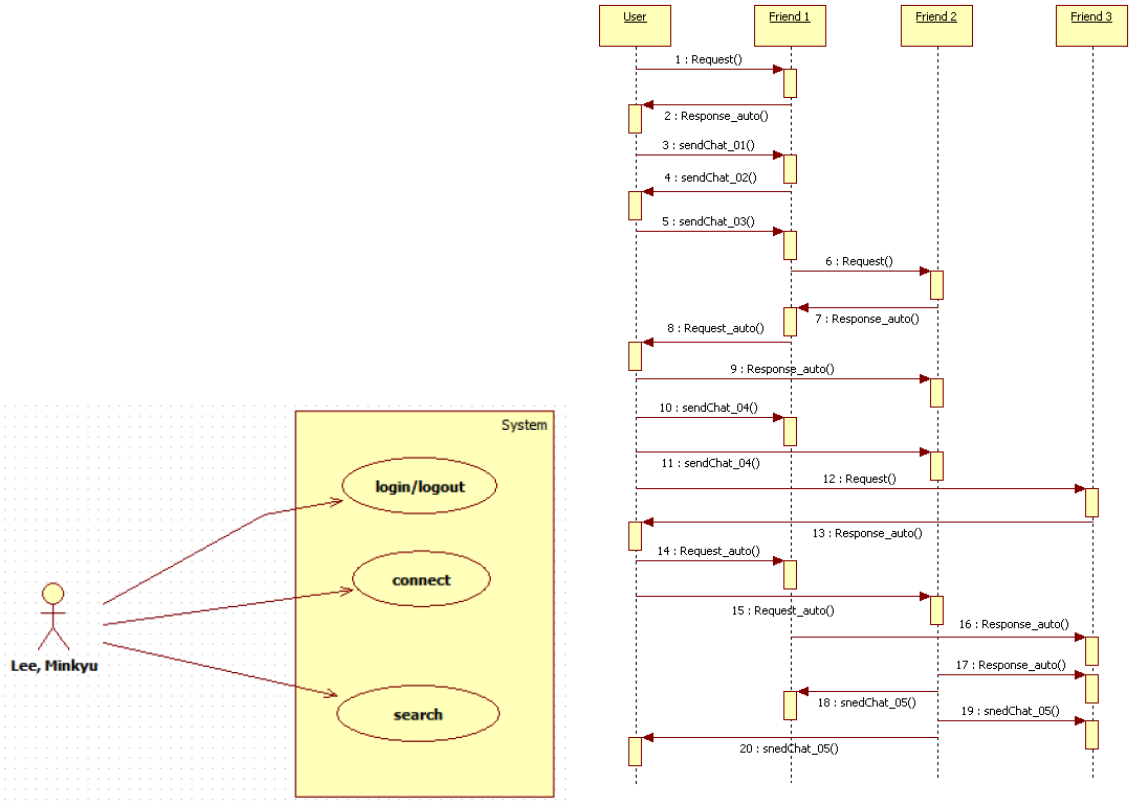


(a) Snapshot of the user IP setting window

(b) Snapshot of the chat message transmission after the connection is created

FIGURE 6. Snapshots of network communication functions

To practice the requirement analysis of a system, students can determine the requirements of the network communication functions of the chat system as use-case diagrams, as illustrated in Figure 7(a). The connection and transmission processes of the message information via the network communication can also be analyzed and implemented according to the sample UML sequence diagram shown in Figure 7(b). To appropriately manage messages to be transmitted to multiple users, students can also practice using the STL containers (such as *dequeue* containers) and implement efficient message queuing and management machineries. This can further benefit the students in learning the data structures and developing problem-solving techniques for network communication.



(a) Use-case diagram of the chat system communication functions

(b) Sample sequence diagram of the network communication process of chatting

FIGURE 7. UML use-case and sequence diagrams of the chat system communication functions

Project Assignment 6: Integrating Software Testing and Improving Information Management. Because of the importance of unit testing in software testing and maintenance, this project asks students to design and practice using the unit testing processes of all crucial functions of their designed classes of communication software functions and previous projects, as well as re-work their codes to ensure that their designed software are testable. Therefore, all functions of content searching and text parsing, network communication, message archiving, and any other facilities implemented in their previous projects must be automatically tested and verified via a systematic unit testing program, which can be implemented using the Qt Unit Test modules provided by the Qt Library, as depicted in Figure 8. When the user executes the testing program, the program should accordingly begin executing each test function, and no additional commands are required for the testing process.

Moreover, in addition to improving the efficiency and reliability of their message communication software, students are also asked to apply some advanced data structures and language features. In this project, they practice adopting the associative containers (i.e., map containers) to rewrite the corresponding classes and algorithmic functions of message record searching and retrieval in their program, so that these functions can provide more efficient message archiving performance. After completing the new program using the improved message searching and retrieval functions, students can again practice the software testing process and learn how to verify effectively the new version of the program when classes and functions are updated. Having the complete testing program experience,

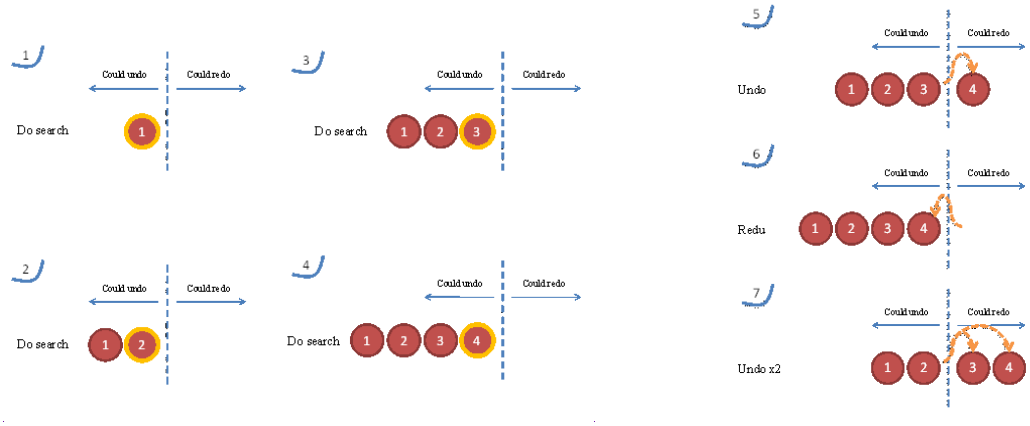
```
void UniTestTest::DoTimeSearch_search_testCase()
{
    DoTimeSearch d;
    string aa="\"h10\"";
    string bb="\"h08\"";
    int b = 3;
    d.getInfo("tue oct 16 h08:50:45 2010");
    d.search(b,aa);
    QVERIFY2(b==3,"DoTimeSearch_search Failure");
    d.search(b,bb);
    QVERIFY2(b==2,"DoTimeSearch_search Failure");
}
void UniTestTest::DoNameSearch_search_testCase()
{
    DoNameSearch dn;
    string aa="ken";
    string bb="panda ken pu gg";

    string a="\"panda\"";
    string bbb="\"qqqq\"";

    dn.getInfo(aa);
    dn.getInfo1(bb);

    int b=3;
    dn.search(b,a);
    QVERIFY2(b==2,"DoNameSearch_search Failure");
    dn.search(b,bbb);
    QVERIFY2(b==2,"DoNameSearch_search Failure");
}
void UniTestTest::DoMessageSearch_search_testCase()
{
    DoMessageSearch dm;
    dm.getInfo("haha i am beach");
    string a="\"beach\"";
    string b="\"ddddd\"";
    int check=3;

    dm.search(check,a);
    QVERIFY2(check==2,"DoMessageSearch_search Failed");
    dm.search(check,b);
    QVERIFY2(check==2,"DoMessageSearch_search Failed");
}
```

```
Starting C:\csdworkspace\uniTest-build-desktop\debug\tst_unitesttest.exe...
FTH: (5752): *** Fault tolerant heap shim applied to current process. This is usually due to previous crashes. ***
********* Start testing of UniTestTest *********
Config: Using QTest library 4.7.0, Qt 4.7.0
PASS   : UniTestTest::initTestCase()
PASS   : UniTestTest::Search_cut_space_testCase()
PASS   : UniTestTest::Search_upcase_to_low_testCase()
PASS   : UniTestTest::DoSearch_add_space_inDate_testCase()
PASS   : UniTestTest::DoTimeSearch_timeFormat_testCase()
PASS   : UniTestTest::DoTimeSearch_search_testCase()
PASS   : UniTestTest::DoNameSearch_search_testCase()
PASS   : UniTestTest::DoMessageSearch_search_testCase()
PASS   : UniTestTest::cleanupTestCase()
Totals: 9 passed, 0 failed, 0 skipped
********* Finished testing of UniTestTest *********
C:\csdworkspace\uniTest-build-desktop\debug\tst_unitesttest.exe exited with code 0
```

FIGURE 8. Snapshots of the unit testing program and results

students can efficiently, skillfully, and elegantly improve and maintain their implemented message communication software.

Project Assignment 7: Completing a Messenger Communication Software. The purpose of the final project assignment is to provide students with a chance to practice integrating all of the C++ programming skills and technicalities to complete a communication software application. At this point, the students should have sufficient understanding of applying and integrating C++ programming skills and OOP technicalities, including file processing, string processing, OOP paradigms, GUI programming, generic data retrieval, software testing, and network communication design, to achieve their own large scale real-time communication applications. Thus, students are encouraged to extend the basic messenger communication application by incorporating additional skills and features, such as multimedia data file transferring, undo/redo functions, and other customized creative designs.



(a) Illustration of basic undo and redo applications of content search functions

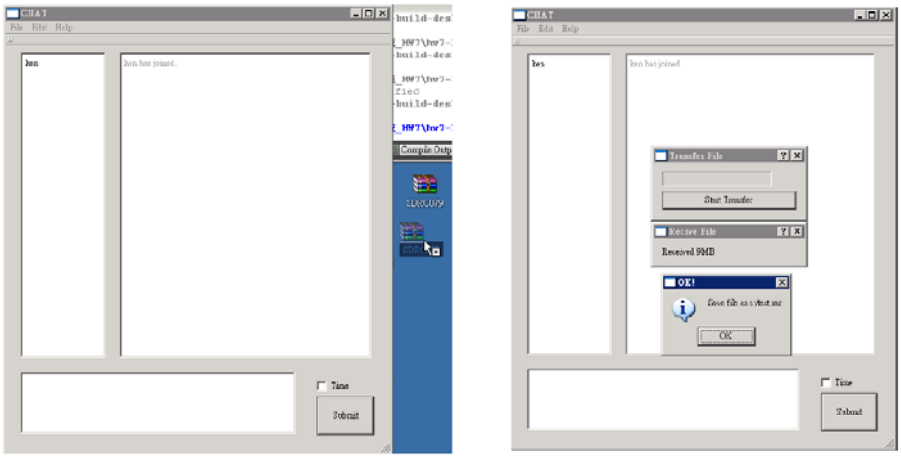
(b) Illustration of additional undo and redo applications of content search functions

FIGURE 9. Illustrations of "Undo" and "Redo" implementations

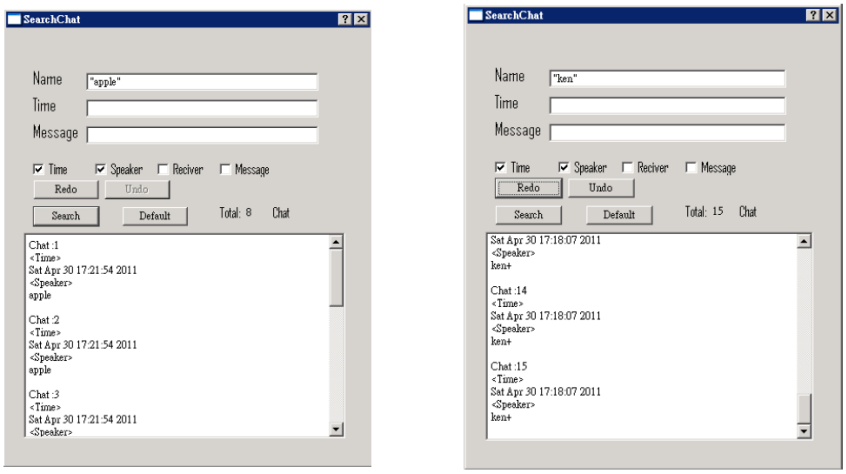
To provide advanced multimedia communication functions, students should implement the network file transferring functions, which should be accompanied by a well-known *drag-and-drop* in their designed messenger GUIs to allow chatting users to transmit and receive multimedia files through the network communications. Thus, students can practice incorporating the programming skills of file processing, generic data retrieval, GUI, and network programming to create a friendly interface of multimedia communication application.

"*Undo*" and "*Redo*" functions also provide high-quality practice for students to incorporate data retrieval and GUI programming skills. Thus, students are required to implement undo and redo functions on the search and retrieval interfaces of chat records, which can be implemented by adopting the C++ technicalities of STL containers and standard algorithms. As illustrated in Figure 9, a circle represents a search or retrieval action of chat records. An action at the left of the dashed line means that the current action can be undone. An action at the right of the dashed line means that the action can be redone. When the user presses the search button first, and the undo button is enabled, the redo button is still disabled because no action has been executed for undoing. Figure 9 illustrates an example of performing a series of subsequent search operations along with undo and redo actions. Notably, the eighth operation (Figure 9(b)) indicates that if the actions that can be redone exist and the user can perform a new search operation, then all of the operations that can be redone are deleted. Through the practice of implementing the undo and redo functions on students' messenger applications, students have the opportunity to practice incorporating the C++ technicalities of STL containers and standard algorithms, as well as the skills of user interface design.

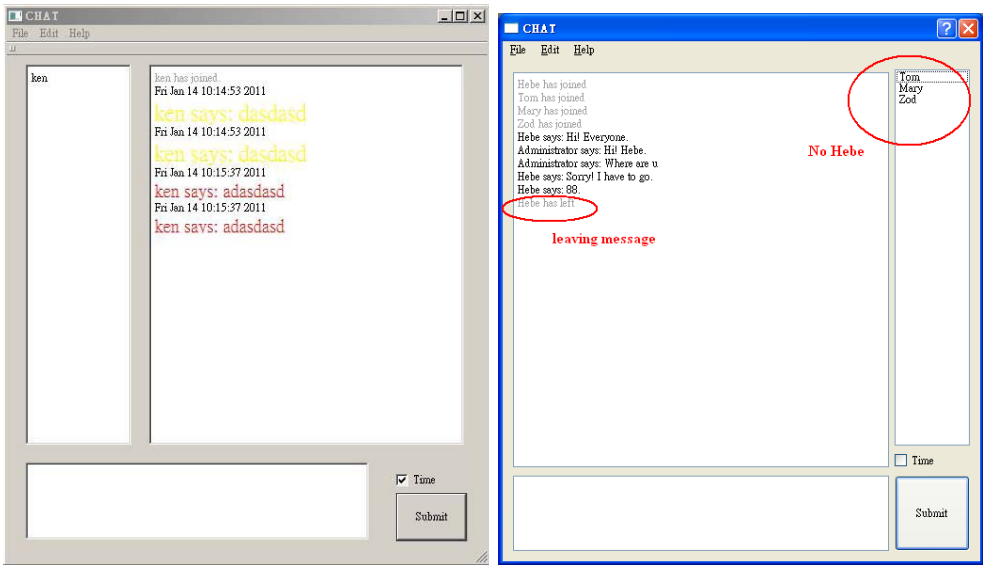
In this final project, students are encouraged to extend their messenger software projects using innovative ideas, including additional elegant human-computer interfaces (such as automatic IP recording, multi-language supporting, customized font color, and size setting interfaces), multimedia chatting interfaces, and message record management functions. The implemented classes and functions of the students' resulting messenger communication software must also undergo the unit testing processes. The snapshots and class diagram of the students' messenger communication software are shown in Figure 10 and Figure 11. Therefore, these extensive practices could help the students in this course apply creative skills and integrate multiple opensource projects with their codes to develop large-scale software projects.



(a) Snapshots of network file transferring applications



(b) Snapshots of undo and redo interfaces of chat record retrieval applications



(c) Snapshots of customized font color and size setting interfaces

(d) Snapshots of improved chat interface with additional leaving notifications

FIGURE 10. Snapshots of the extended messenger communication software in the final project

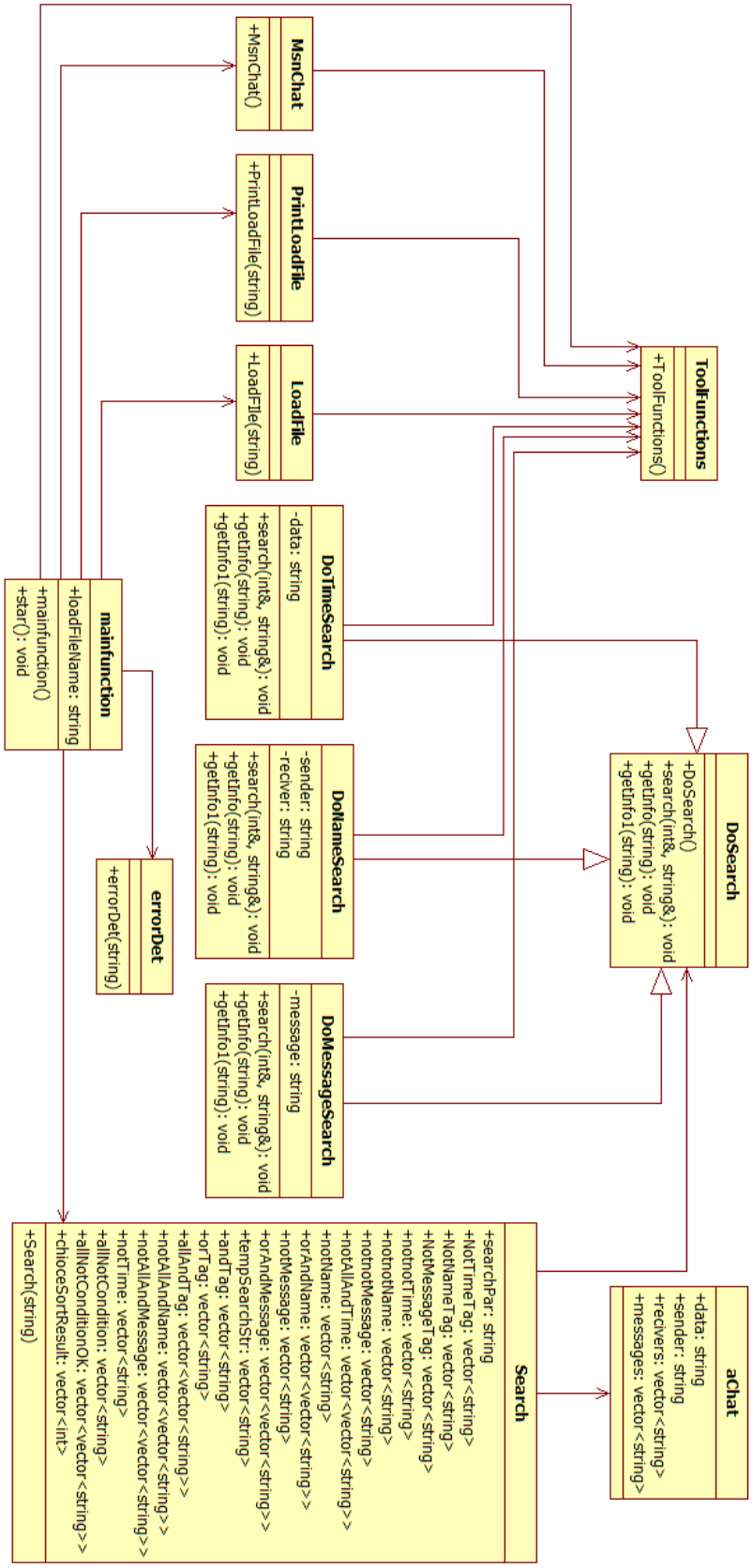


FIGURE 11. Sample UML class diagram of the students' resulting messenger communication software

4. **Results and Discussions.** The remedial C++ object-oriented programming course for native and international students who held non-CS bachelor degrees began in the Fall 2009 semester (when this course was taught by the conventional programming course curriculum [2]) in the graduate school of the Department of Computer Science and Information Engineering at National Taipei University of Technology, Taiwan. In the Fall 2010 semester, the project-based curriculum for implementing a large-scale practical communication software by incorporating C++ programming and OOP skills was introduced in this remedial course. Various forms of student feedback, such as that from discussions via the course Facebook social-network forum and project reports, were accordingly adopted to adjust and improve the course contents. In each project assignment, the students were encouraged to discuss and report their encountered difficulties and required programming skills and knowledge via the course forum and project reports. In this way, the students enrolled in the course with the proposed project-based curriculum in the Fall 2010 and 2011 semesters are adopted as the experimental groups, while the students enrolled in the course without the project-based curriculum in the Fall 2009 semester are adopted as the control group.

As depicted in Figure 12, the students' grade distribution and their GPA trend obtained from the Fall 2009 to the Fall 2011 semesters demonstrate the effectiveness of the proposed project-based curriculum. When the project-based curriculum was introduced in the Fall 2010 semester, although the failed and withdrawn students (i.e., GPA = 0) increased somewhat because of the heavy loading of homemade projects, the average GPA increased to 3.25 in the Fall 2010 semester, and to 3.47 in the Fall 2011 semester. Concurrently, the students' grade distribution also shifted to the higher ends, and the students' dominant GPA became A (4.0) in the Fall 2011 semester. As a comparison, in the Fall 2009 semester, the students' average GPA was approximately 2.91 when they are taught by the conventional programming course curriculum [2].

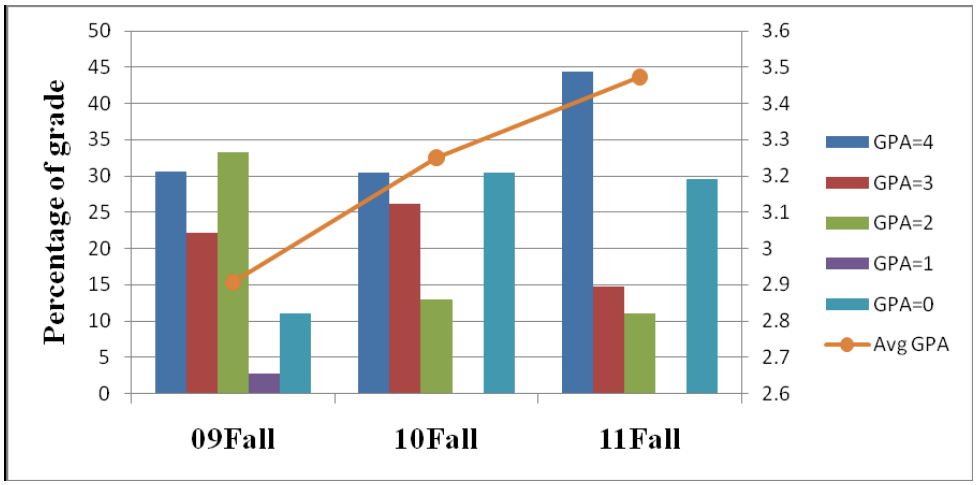


FIGURE 12. Illustrations of students' grade distribution and GPA trend from Fall 2009 to Fall 2011

A course survey was administered at the end of the Fall 2011 semester, and all 27 students (including domestic and international students) enrolled in the course completed the survey questionnaires. The survey results from five questions are listed in Table 2. As depicted in Table 2, most students agreed strongly that applying C++ and OOP concepts and software design skills from the course contents and homemade projects contributed significantly to their understanding of developing communication software (Table 2, 4.46 out of 5 in Question 1, and 4.55 out of 5 in Question 2, respectively).

TABLE 2. Course survey questions and results from students enrolled in Fall 2011

Survey questions (1 = Strongly Disagree to 5 = Strongly Agree)	**Results**
1. I fell that I have a better understanding of the C++ and OOP concepts and software design after taking this course.	4.46/5
2. The homemade projects can effectively help me to learn the C++ and OOP software design	4.55/5
3. I study hard to develop my programming skills and solve the problems of developing large-scale software.	4.64/5
4. The course contents interest me and I would recommend this course to my junior classmates.	4.36/5
5. The course social-network forum can help me learn programming skills and conveniently obtain support from TAs and the instructor.	4.55/5

The students also definitely agreed that they were interested in the course contents and homemade projects, and were encouraged to study hard to enhance their problem solving abilities in developing large-scale software projects (Table 2, 4.64 out of 5 in Question 3, and 4.36 out of 5 in Question 4, respectively). They pay more than 20 hours weekly in average on their homemade projects throughout the whole semester. The remaining question also reflects that the students feel the course social-network forum can efficiently provide intermediate supports and helps from the TAs and the instructor, provide them a feedback way to the instructor to improve the course contents, and share the programming knowledge and experiences (Table 2, 4.55 out of 5 in Question 5). The survey results demonstrate that the proposed project-based curriculum successfully provokes student interests in developing real-time communication software. By applying C++ and OOP skills, the proposed curriculum contributes to students learning the advantages of OOP and practical software engineering.

The students' grade assessment results in Figure 12 reveal that the students' learning performance applying C++ and OOP skills significantly improved, and their interests and motivation in incorporating the programming concepts and skills for constructing a practical large-scale software system related to their lives also effectively improved, as compared with the grade assessment results in the Fall 2009 semester using the conventional programming course curriculum [2]. Accordingly, the student survey and grade assessment results in Table 2 and Figure 12 reveal the efficiency and effectiveness of the proposed project-based curriculum, including the course contents of C++ and OOP features, homemade projects for developing large-scale communication software, and course social-network forum. Most of the students significantly benefited from rapidly developing their problem-solving skills and learning how to apply practical solutions in using different OOP concepts for their thesis studies.

The proposed teaching approach integrates practical applications of communication software design and network social connections on OOP course. After taking this curriculum, the students significantly and strongly develop their sense of accomplishment and their interests on solving practical engineering problems using OOP skills. The course contents and projects can be widely adapted for many practical aspects of students' research studies and their careers in information and communications technology (ICT) industries, e.g., mobile computing software design, social computing, cloud computing, user interface design, and embedded systems.

5. **Conclusions.** This study developed a project-based remedial curriculum for teaching the skills and concepts of the C++ programming language and object-oriented programming (OOP). The proposed curriculum is designed for domestic and international students admitted into CS graduate programs without CS bachelor degrees or sufficient computer programming experience. To assist such students in quickly learning the major C++ and OOP skills for their graduate studies, the pedagogical approach of the proposed curriculum includes a set of homemade projects, whose results are used to create a large-scale communication software system related to their lives and theses. Based on constructivist learning technology, the proposed project-based remedial curriculum can significantly promote the learning effectiveness and interests of students via hands-on, minds-on, and learning-by-doing practices. Furthermore, an interactive course web forum based on the Web 2.0 social-network technology of Facebook was also established. This interactive course forum approach can help students to obtain the announcements of course materials and project assignments conveniently, and to pose questions for assistance from and discussion with the TAs and instructor. In addition, the forum approach facilitates providing immediate feedback and knowledge sharing of programming skills and concepts.

Because the proposed teaching approach incorporates practical applications of communication software design and network social connections on OOP course, the students' interests on software design and problem-solving skills for their graduate studies are efficiently aroused. The student survey and grade assessment results demonstrate that this project-based curriculum can effectively provoke student interests in applying C++ and OOP skills as well as cultivate their problem-solving abilities, thus providing effective chances to learn the advantages of OOP and practical software engineering. Besides, the proposed project-based curriculum not only can apply in teaching C++ OOP, and the proposed course contents and projects can widely be adapted for teaching many practical programming courses in different, such as embedded system programming, network programming, and mobile application programming with C++, C#, Java, or Objective-C programming languages. Thus, the students can widely and effectively adapt their learned concepts and skills for many practical aspects of students' research studies and their careers in ICT industries.

Acknowledgment. This work is supported by the National Science Council of Taiwan under Contract No. NSC-101-2219-E027-002, NSC-101-2219-E-027-006, and NSC-101-2219-E-027-007.

REFERENCES

[1] ACM AIS IEEE-CS, *Curricula Recommendations*, http://www.acm.org/education/curricula-recommendations, 2009.

[2] The Joint Task Force on Computing Curricula, Computing curricula 2001, *J. Educ. Res. Comput.*, vol.1, pp.1-240, 2001.

[3] J. Lang, G. C. Nugent, A. Samal and L.-K. Soh, Implementing CS1 with embedded instructional research design in laboratories, *IEEE Trans. Educ.*, vol.49, pp.157-165, 2006.

[4] Z. Anik and O. F. Baykoc, Comparison of the most popular object-oriented software languages and criterions for introductory programming courses with analytic network process: A pilot study, *Comput. Appl. Eng. Educ.*, vol.19, pp.89-96, 2011.

[5] H. Zhu and M. Zhou, Methodology first and language second: A way to teach object-oriented programming, *Proc. of the 18th Annu. Conf. OOP Syst. Lang. Appl.*, Anaheim, CA, pp.140-147, 2003.

[6] J. Gálvez, E. Guzmán and R. Conejo, A blended E-learning experience in a course of object oriented programming fundamentals, *Knowledge-Based Syst.*, vol.22, pp.279-286, 2009.

[7] G. Licea, R. Juárez-Ramírez, C. Gaxiola, L. Aguilar and L. G. Martínez, Teaching object-oriented programming with AEIOU, *Comput. Appl. Eng. Educ.*, 2011.

[8] B. Hofer and P. Pintrich, The development of epistemological theories: Beliefs about knowledge and knowing and their relation to learning, *Rev. Educ. Res.*, vol.67, pp.88-140, 1997.
[9] T. Inanc and H. Dinh, A low-cost autonomous mobile robotics experiment: Control, vision, sonar, and handy board, *Comput. Appl. Eng. Educ.*, vol.20, pp.203-213, 2012.
[10] M.-H. Chen and T.-L. Li, Construction of a high-performance computing cluster: A curriculum for engineering and science students, *Comput. Appl. Eng. Educ.*, vol.19, pp.678-684, 2011.
[11] C.-Y. Chen and H.-M. Cheng, Open architecture design of embedded controller for industrial communication gateway, *ICIC Express Letters, Part B: Applications*, vol.1, no.1, pp.51-56, 2010.
[12] Y. Tang, L. M. Head, R. P. Ramachandran and L. M. Chatman, Vertical integration of system-on-chip concepts in the digital design curriculum, *IEEE Trans. Educ.*, vol.54, pp.188-196, 2011.
[13] C. Mu, S. Liu and J. Chen, Hardware/Software integrated training on embedded systems, *International Journal of Innovative Computing, Information and Control*, vol.2, no.2, pp.457-464, 2006.
[14] F. Valles-Barajas and W. Schaufelberger, A proposal for the software design of control systems based on the personal software process, *International Journal of Innovative Computing, Information and Control*, vol.6, no.8, pp.3451-3466, 2010.
[15] W.-K. Chen and Y. C. Cheng, Teaching object-oriented programming lab with computer game programming, *IEEE Trans. Educ.*, vol.50, pp.197-203, 2007.
[16] H. Zhu, Teaching OOP with financial literacy, *IEEE Trans. Educ.*, vol.54, pp.328-331, 2011.
[17] A. Pérez and J. Rosell, A roadmap to robot motion planning software development, *Comput. Appl. Eng. Educ.*, vol.18, pp.651-660, 2010.
[18] S. Lippman and J. Lajoie, *C++ Primer*, 3rd Edition, Addison-Wesley, MA, 1998.
[19] B. Stroustrup, *Programming: Principles and Practice Using C++*, Addison Wesley, 2009.
[20] H. S. Du and C. Wagner, Learning with Weblogs: Enhancing cognitive and social knowledge construction, *IEEE Trans. Prof. Commun.*, vol.50, pp.1-16, 2007.
[21] C. Safran, Blogging in higher education programming lectures: An empirical study, *Proc. of the 12th MindTrek*, Tampere, Finland, pp.131-135, 2008.
[22] Unified Modeling Language (UML) Specification, *Object Management Group*, http://www.omg.org/.
[23] J.-R. Chang Chien, A handheld electronic patient record using a new UML component-based architecture, *Biomed. Eng. Appl. Basis. Comm.*, vol.22, pp.437-451, 2010.
[24] Y. L. Chen, C. Y. Chiang, C. W. Yu, S. M. Yuan and Z. W. Hong, A customized and portable component-based framework for intelligent home-care system design with video and physiological monitoring machineries, *Biomed. Eng. Appl. Basis. Comm.*, vol.23, pp.325-348, 2011.
[25] S. Otón, A. Ortiz, J. R. Hilera, R. Barchino, J. M. Gutierrez, J. J. Martinez, J. A. Gutiérrez, L. de Marcos and M. L. Jiménez, Service oriented architecture for the implementation of distributed repositories of learning objects, *International Journal of Innovative Computing, Information and Control*, vol.6, no.3, pp.843-854, 2010.
[26] J. Blanchette and M. Summerfield, *C++ GUI Programming with Qt4, 2/e*, Prentice Hall, 2008.
[27] N. M. Josuttis, *The C++ Standard Library: A Tutorial and Reference*, Addison Wesley, 1999.