Software Requirements Specification of A University Class Scheduler

Deanna M. Needell   Jeff A. Stuart   Tamara C. Thiel   Sergiu M. Dascalu   Frederick C. Harris, Jr.

Department of Computer Science
University of Nevada, Reno
1664 N. Virginia St.
Reno, NV, 89557, USA

Abstract

The University Class Scheduler (UCS) presented in this paper is a novel scheduling tool intended to be used by universities to schedule classes into classrooms. In essence, UCS allows university administrators to enter relevant college and building information, schedule the input classes (courses) into input classrooms, and create web pages that provide detailed schedule information on a semester-by-semester basis. The UCS, which performs the scheduling of classes according to a number of user-selected parameters, can be easily adapted for applications outside the academic realm. This paper presents the main aspects of the University Class Scheduler's UML-based specification, gives details of the UCS' current development status, and points to a series of possible extensions that we intend to investigate in the near future.

Keywords: class scheduler, software tool, requirements specification, use cases, scenarios, class diagram, UML.

1 Introduction

The University Class Scheduler (UCS) that we have recently developed is a software product intended to be used by universities to schedule classes (courses) offered during a semester into available classrooms. The focus of this project has been on academic scheduling for universities, yet due to its flexible design the UCS can be easily adapted for applications outside the academic realm. The UCS program allows university administrators to input relevant college, department, and building information, schedule courses into classrooms, and create web pages linked to appropriate university pages (specifically, college and department web pages). Thus, via web posting, the UCS makes available to the public detailed information on the scheduling of classes.

The UCS performs the scheduling based on certain parameters selected by the user, as explained later in the paper. Besides its scheduling algorithms, UCS incorporates a generic XML parser that allows the manipulation of large data files. In particular, the XML parser is used to translate raw data files into XML-defined storage files, thus reducing or eliminating the user's highly tedious and time consuming task of entering large amounts of data.

There are rather few similar products currently on the market, for instance the Astra Schedule [1] and the Book King Online Scheduling Software [2]. However, these products are more complex than ours and thus require more extensive training of the user. Also, particularly important, these products are rather expensive. We have embarked in the UCS project with the goal of developing a software product that provides a simpler yet very efficient and flexible class scheduler tool that can be made more easily available to universities across the nation.

For development purposes we have followed a software engineering approach along the lines described in [3] and have used extensively the Unified Modeling Language (UML) [4, 5] for representing the model of the UCS software. The specifications of the UCS' requirements as well as the descriptions of use cases and scenarios have been written following the guidelines presented in [6]. We have found that by following a rigorous development process and by relying on the UML as modeling notation both the UCS' functionality and overall architecture have not only been better understood, defined, and represented but also better planned to accommodate future requirements.

This paper, in its remaining part, is structured as follows: Section 2 presents a brief description of the UCS, Section 3 describes the main functional and non-functional requirements of the scheduler, Section 4 provides details on use cases and scenarios, Section 5 presents the class diagram of the UCS, Section 6 reports on the current status of the UCS tool, and Section 7 completes the paper with a number of pointers to future work.

2 General Description

The UCS is intended to provide a means for entering the necessary information for class scheduling, viewing and modifying this information, storing the information into XML-defined files, parsing XML files into ready-to-process data, scheduling classes into classrooms based on user-selected parameters, and creating web pages with scheduling details about courses offered and their assigned classrooms.

From an organizational point of view the UCS software tool has two main components: the *scheduler* itself and the *publisher* of the schedule. The first component, the scheduler, is used by the administrators of each university department to enter, either manually or by loading data from existing files, the department's classes, times, and professors. Information on buildings and classrooms is also entered. As soon as all the necessary information is entered into the system the UCS can schedule the specified classes into available classrooms without overlap, based on certain parameters selected by the user. Specifically, these parameters are classroom size, classroom type (e.g., regular room, distance education room, or "smart" room), and distance from department. Once the scheduling is complete, the second component of the UCS, the publisher, can be invoked to create an HTML file with detailed schedule information and pointers to appropriate web pages. Such pointers include links to the University's online map system, as well as to department and professor home pages.

3 Requirements Specification

Following the guidelines and notations presented in [6], the requirements for the UCS have been specified as detailed below.

3.1 Functional Requirements

The main functional requirements of the UCS are the following:

1. The UCS shall provide a means for entering and storing:
   a. college information;
   b. department information;
   c. professor information;
   d. class information;
   e. building information;
   f. classroom information.
2. The UCS shall schedule, if possible, the classes into classrooms.
3. The UCS shall take into consideration for generating the schedule the following parameters: classroom size, classroom type, and distance from the department. These parameters shall be selected by the user from a list of pre-defined values.
4. The UCS shall save the schedule in a format that can be used to generate a University Semester Catalog that is readable by humans.
5. The UCS shall notify the user if no valid schedule can be generated and shall indicated the causes that prevent the generation of a valid schedule.
6. The UCS shall write the schedule containing class and classroom information to a file that can be read by supplied web pages so that these can be posted onto the university web server.

3.2 Non-functional Requirements

The most important non-functional requirements for the UCS are the following:

1. The UCS shall be written in C++.
2. The interfaces of the UCS shall be implemented using the QT Non-Commercial Toolkit [7].
3. The UCS shall store its files using XML tags [8].
4. The UCS will be platform independent.

4 Use Case Modeling

As part of the modeling process, the functionality of the UCS has been defined using use cases and scenarios. At a high level of abstraction the entire functionality of the class scheduler is captured in the use case diagram shown in Figure 1, a couple of major parts of UCS functionality are presented in use cases shown in Figures 2 and 3, and examples of specific ways of using the software are provided as scenarios in Figures 4 and 5. According to Rumbaugh, a use case is "a specification of sequences of actions, including variant sequences and error sequences, that a system, subsystem, or class can perform by interacting with outside actors" [9]. One the other hand, as pointed out by Booch, "scenarios are to uses cases as instances are to classes, meaning that a scenario is basically one instance of a use case" [5]. We have relied extensively on use cases and scenarios to define the expected functionality of the UCS and better describe its intended behavior.

4.1 Use Case Diagram

The use case diagram shown in Figure 1 depicts the interactions between the actors (the staff, the students, and the professors of the university) and the UCS software system. First, the department staff will enter information about professors, colleges, buildings, departments, classes, and classrooms. Once this is done, a university administrator will use the UCS to generate the classroom schedule. The university catalog creators will then use the generated classroom schedule to create the semester class catalog. The university webmaster will post the web pages and the generated schedule file to the university server. Consequently, the students, the professors and, generally speaking, the public, can view these web pages.

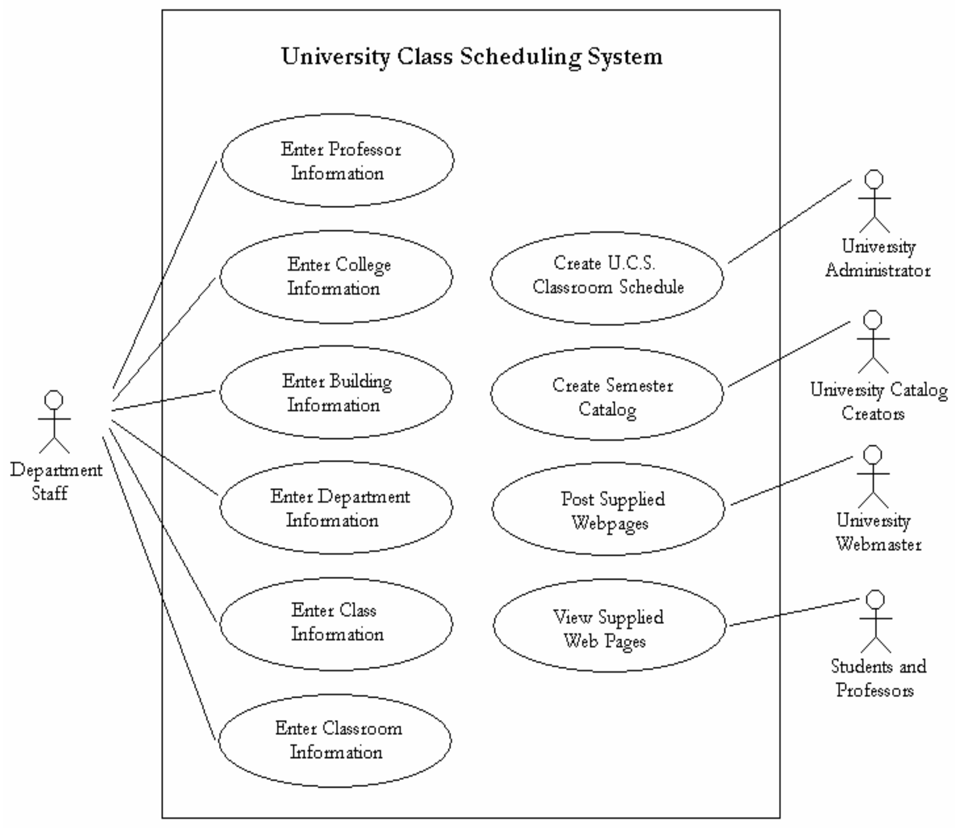
University Class Scheduling System
Enter Professor Information
Enter College Information
Enter Building Information
Enter Department Information
Enter Class Information
Enter Classroom Information
Create U.C.S. Classroom Schedule
Create Semester Catalog
Post Supplied Webpages
View Supplied Web Pages
Department Staff
University Administrator
University Catalog Creators
University Webmaster
Students and Professors

Fig. 1 Use Case Diagram of the UCS

4.2 Detailed Use Cases

Due to space limitations we present in the following only two examples of use cases and two sample scenarios. Specifically, with respect to use cases, Figure 2 depicts the "Enter class information" use case, in which relevant data pertaining to a class (course) offered during the semester is entered, while Figure 3 shows the "Create classroom schedule" scenario. It is worth noting that each use case has a primary scenario which describes the normal, most common way of using the UCS software to perform the respective use case, as well as a number of secondary scenarios, which describe the less likely to occur, less common, or "exceptional" execution paths of the use case.

Use Case: Enter Class Information

ID: UC5

Actor:
Department staff

Precondition:

1. Class information needs to be entered or modified.

Flow of events:

1. The use case starts before scheduling is done.
2. The system prompts the user to enter the following class information: college name, department name, class number, call number, professor name, number of students, days and times for lectures, classroom type, lab information, if any, and cross-list information, if any.
3. The user enters all the above class information.
4. The system validates the entry.
5. The system saves the information entered to file.

Secondary scenarios:
Entered Class Already Exists
Required Information Is Missing
User Cancels Enter Class Information

Postcondition:

1. Entered class information is saved.

Fig. 2 UCS Use Case "Enter Class Information"

Use Case: Create Classroom Schedule

ID: UC7

Actor:
University administrator

Precondition:

1. All semester information has been entered.

Flow of events:

1. The use case starts after all semester information has been entered.
2. The system prompts the user to choose which parameter(s) he or she wishes to schedule by.
3. The system schedules, as much as possible, the classes into classrooms.
4. The system notifies the user whether it has successfully scheduled all classes into classrooms.
5. The system saves the schedule even if it was not successful (in all cases, the portion that is successfully scheduled is saved).

Secondary Scenarios:
No Valid Schedule Solution Is Possible
Ran Out of Memory
User Cancels Schedule Creation

Postconditions:

1. A schedule file is created.
2. The user knows whether the schedule was created successfully or if changes to the information are needed.

Fig. 3 UCS Use Case "Create Classroom Schedule"

4.3 Scenarios

For the "Enter class information" use case described above both the primary scenario and a secondary scenario are presented in the figures that follow. More exactly, Figure 4 presents the normal scenario of the "Enter class information" use case, while Figure 5 shows the "Entered class already exists" scenario of this particular use case.

Scenario for Use Case: Enter Class Information
Primary scenario
Scenario ID: S5.1
Actor: Department staff
Precondition: 1. Class information needs to be entered or modified.
Flow of events: 1. When the user chooses to enter class information, a window is displayed with fields in which to enter or to select for existing values the following class information: college name, department name, class number, call number, professor name, number of students, and classroom type. There are also fields to enter or select the date and time(s) of both lecture and, if there is one, the lab. If there is a lab, fields allowing the user enter or select classroom parameters for the lab are also displayed. 2. The user enters class information by providing all the items indicated in step 1. 3. The user clicks the "Done" button. (Note that if the user clicks the "Cancel" button, the scenario "User Cancels Enter Class Information" takes place.) 4. The system validates the class information entry. (Note that if the entry is invalid, either the scenario "Entered Class Already Exists"or the scenario "Required Information Is Missing" takes place.) 5. The system saves the class information to file.
Postcondition: 1. Entered class information is saved.

Fig. 4 Primary Scenario of the "Enter Class Information" Use Case

Scenario for Use Case: Enter Class Information
Secondary scenario: Entered Class Already Exists
Scenario ID: S5.2
Actor: Department staff
Precondition: 1. Class information needs to be entered or modified.
Flow of events: 1. This scenario starts following step 3 of scenario S5.1, after the user clicks "Done". If the program detects that the class entered already exists, the program will display an error message and prompt the user for a valid class name and department. 2. The program detects that a class with the same class name and same department has previously been entered. 3. An error message is displayed notifying the user that the entered class already exists. The user is asked to enter a different class name or department. 4. The user clicks the "OK" button. 5. The window for the use case "Enter Class Information" is displayed, as indicated in step 1 of scenario S5.1. The user has the option of either updating the required information in the fields provided (and continue as in scenario S5.1) or clicking the "Cancel" button. In the latter case the scenario "User Cancels Enter Class Information" takes place from here.
Postcondition: **1.** Class information remains unchanged.

Fig. 5 A Secondary Scenario of the "Enter Class Information" Use Case

5 Class Diagram

The class diagram of UCS is presented in Figure 6. This diagram has been developed iteratively along the guidelines suggested in [6]. Specifically, we have developed first an analysis class diagram, with the less detailed analysis classes indicated.

Then, the class diagram has evolved into a structural model for the UCS that contains the more refined design classes of the system. The variant we present in Figure 6 is the design class diagram of UCS, showing the complete high level architecture of the UCS software and details on attributes, operations, relationships, and multiplicity constraints.

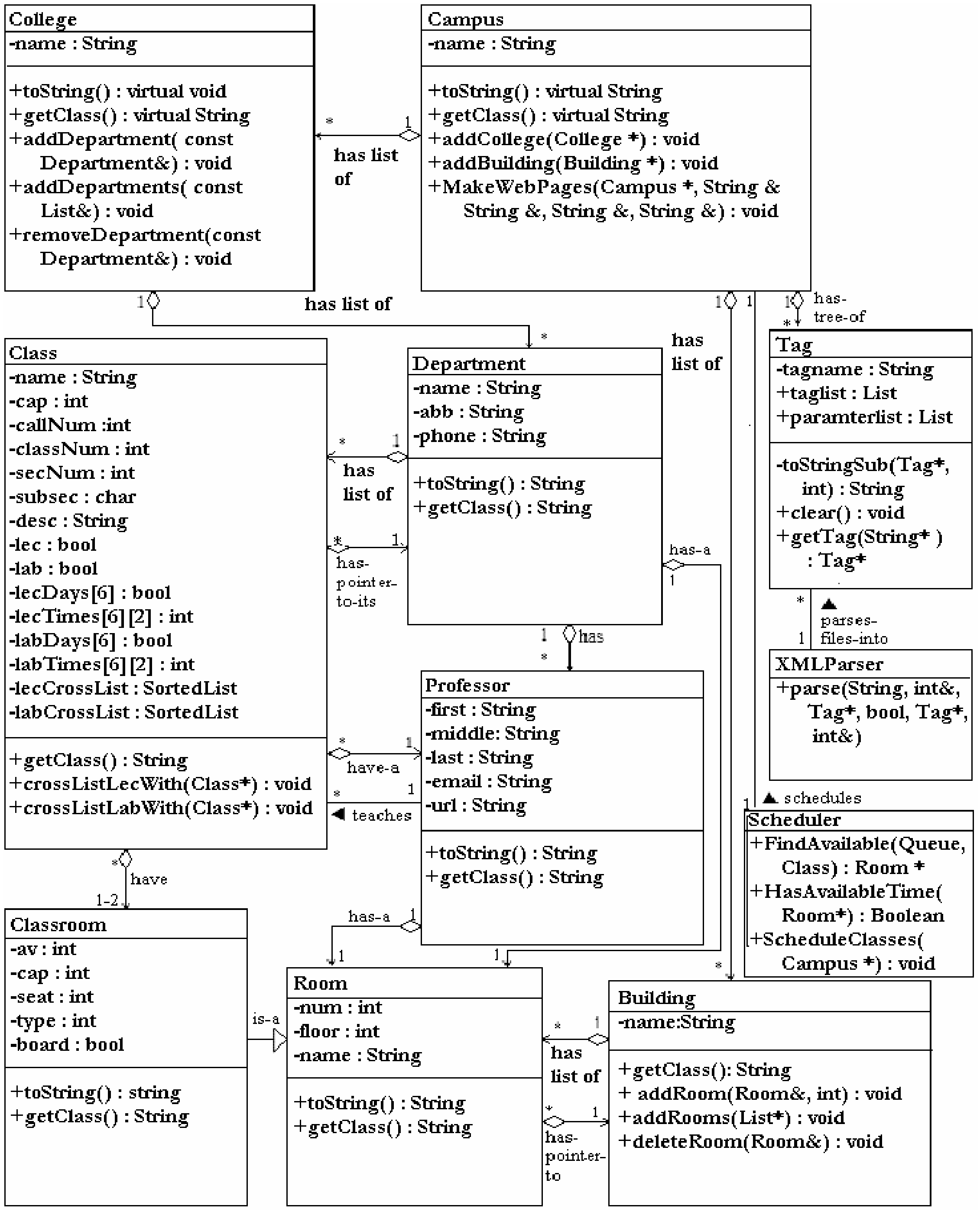

Fig. 6 The Class Diagram of the UCS

6 Current Status

Although we have focused in this paper on the specification of the UCS software, we note that the development of the UCS has already led to a first, fully operational version of the scheduler. This version has been written in C++, and its interfaces were implemented using the QT Non-Commercial Toolkit [7]. In addition, UCS stores all its data files in XML format [8].

The UCS has been successfully tested at the University of Nevada, Reno using information pertaining to the last semester. The results of the complete scheduling were more than encouraging: the UCS has provided a better allocation of classes to classrooms than the one previously used (done manually). In particular, more classes were assigned to classrooms in buildings belonging to the offering departments, thus shortening considerably the distances between the departments that offer the courses and the locations where the courses are delivered.

More details on the design, implementation, integration, and testing of the UCS are beyond the scope of this paper. For more information on the UCS the reader is invited to have a look at [10].

7 Conclusions

The UCS tool whose specification has been presented in this paper provides a simple but efficient means of scheduling university classes into classrooms. We believe that the UCS not only represents a useful, practical tool, but also a research proof of concept based on which many possible extensions to this project can be further investigated.

A very useful addition to the UCS would be the inclusion of time scheduling. Thus, the UCS would schedule classes not only into classrooms but also into time slots. Also, the UCS could suggest optimal courses of actions for resolving scheduling conflicts of various types, including time conflicts. Another possible extension would be to elaborate the web pages associated with UCS so that they could link to campus maps, campus images, and so forth. Enhancement of scheduling algorithms is another potential area of research and development for the UCS. Finally, the UCS could evolve into a generic scheduler that could be used in numerous applications outside the university environment.

References

[1] Ad Astra Information Systems, *Astra Schedule*, accessed May 4, 2003 at http://www.aais.com/as.html

[2] *Book King Online Scheduling Software*, accessed May 4, 2003 at http://www.bookking.ca/index.asp

[3] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 1999.

[4] OMG's UML Resource Page, accessed April 25, 2003 at http://www.omg.org/uml.

[5] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1998.

[6] J. Arlow and I. Neustadt, *UML and the Unified Process: Practical Object-Oriented Analysis & Design*, Addison-Wesley, 2002.

[7] Trolltech AS, *QT Overview*, accessed at www.trolltech.com/products/qt/index.html, May 4, 2003.

[8] E. R. Harold and W. S. Means, *XML in A Nutshell*, 2nd Ed., O'Reilly and Associates, Inc., 2002.

[9] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.

[10] Group 8 UCS Project web-site, accessed at http://www.cs.unr.edu/~needell/cs426/, May 4, 2003.