Teaching Assignment Planner (TAP)

Hui Ying

A Major Report
In
The Department
Of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

June 2002

0-612-72949-4

Abstract

Teaching Assignment Planner

Hui Ying

Each year, the Department of Computer Science at Concordia University needs to assign instructors to about 200 courses. Each course requires an instructor and some of the courses also require one or two coordinators. The instructors must be assigned using various constraints. The *Teaching Assignment Planner (TAP)* program is designed and implemented as computer software to assist the people involved to do the teaching assignment work at Concordia University to fulfill the assignment task easily.

To satisfy the purpose of helping the user do the assignment work, it is very important to make this program very easy to use. A user-friendly interface and comprehensive functionalities are the quality factors to a successful program. As a result, great efforts are focused on them. The TAP program incorporates the Java technology and MS-Access is used as storage database.

This report demonstrates the design and implementation of the TAP program and the results of how to use this program is also provided as an important part. In addition, the future work of this project is presented.

Acknowledgements

First and foremost, I wish to express my sincere gratitude to my supervisor, Dr. Peter Grogono, for all his enthusiastic support, and valuable guidance during the development of the project. His encouragement and help enabled me to understand this project and complete it. Without his kind support, nothing would have been possible.

I also wish to thank to Dr. Ahmed Seffah, for his taking time to review my report and provide valuable opinions. Also, I will thank to Graduate Program Secretary, Halina for her great help.

Furthermore, I wish to take this opportunity to appreciate all my family members, especially my parents, for their unconditional love and moral support. Without their encouragement, I could not finish my graduate study including this project smoothly.

Finally, the unique gratefulness goes to my husband, Chunbo Yang, and I would like to dedicate this project to him for his infinite love, sharing and always being there as in need. Thanks to him for accompanying me during many days and nights of hard working. Without his great patience and full support, it is impossible to complete this project in such a short time.

Table of Contents

List of Figures

1. Introduction

1.1 About Teaching Assignment Planner (TAP)

Teaching Assignment Planner (TAP) is a program designed to assist the user (secretary or other staffs) perform the task of teaching assignment. The program allows the user to:

- View the current teaching assignment state
- Perform teaching assignment,
- Modify the information of courses and instructors
- View the summary statistics

1.2 Choice of Software

The software tools we choose to implement TAP are:

- Java programming language and Java Swing technology
- Borland JBuilder 4.0 as the development tool
- Windows 2000/ME as the development platform
- MS-Access for database management

1.3 Organization of the Report

First, we describe the background of developing the TAP program in the Chapter 2.

Then, Software Requirement Analysis is presented in Chapter 3. In this chapter, both functional and non-functional requirements are discussed. In Chapter 4, the System Design and User Interface Design are described. In Chapter 5, we describe the problems and solutions of implementation of TAP program. Following, usability and results illustrating how the TAP program is being used are also presented in Chapter 6. Finally, Chapter 7 describes the conclusion and future work of the TAP program.

2. Background and Project Overview

2.1 Motivation of TAP Program

Each year, the Department of Computer Science at Concordia University needs to assign instructors to about 200 courses. Each course requires an instructor and some of the courses also require one or two coordinators. [TAPR01]

An instructor must be assigned using various constraints. Some of the constraints are easy to understand and simple to check. For example, an instructor cannot teach two courses at the same time. Also, a course section can be assigned to only one instructor.

But some other constraints are more complicated and must be carefully prioritized. For example, an instructor who does not like teaching in the evening may want to teach a graduate course, but the only available graduate courses are all scheduled in the evening. Another example, instructor A doesn't like to teach COMP248 and wants to teach COMP471. Instructor B agrees to give up COMP471 but prefers teaching COMP352 to COMP248. Consequently, we must find an open section of COMP352 and another instructor for COMP248. Based on such complicated constraints, we should consider them completely when performing the teaching assignment.

It is a very tough task of doing the teaching assignment without any tool. The person who needs to perform the teaching assignment should deal with not only the variable courses and instructors' information, but also all kinds of assignment constraints. Besides, it would be difficult for the assigner to know what degree the teaching assignment has been completed. He might probably omit a very important constraint and finish an assignment which would incur conflict. For example, there is an instructor who has already been assigned several course sections to teach. The assigned assigns another section to the instructor, forgetting to check every possible constraint. As a result, the assigned section has a time conflict with one of the assigned sections before. Both are scheduled on the same time slot on the same day!

The motivation of the *Teaching Assignment Planner (TAP)* is to help the person doing the teaching assignment simplify the task of the assignment. With the help of this TAP program, the assigner would finish the task of teaching assignment very efficiently and with accuracy. [TAPR01]

2.2 Teaching Assignment Planner Program

Since the *Teaching Assignment Planner (TAP)* is aimed to help the person doing the assignment simplify the task of the assignment, that is to say, help the user handle the complicated assignment constraints at ease. The TAP program should be very easy to learn, easy to use, and the program should be very robust in case of all kinds of possible operations the user could perform which would lead to error.

The TAP program involved a 3-tier architecture and was implemented by using the pure Java Language and Java Swing technology. The MS-Access served as the database which stores all the information of the courses, instructors and the assignments. The JDBC worked as the bridge interface between the Java application and the storage database.

TAP has two major features:

- *Cross Platform:* TAP program is implemented by pure Java language in such a way it runs on any platform;
- *Heterogeneous Database:* Since TAP incorporates the JDBC, it can access all the JDBC-enabled databases. Therefore, The database migration will be very easy.

Through TAP, the user can view the current teaching assignment state. The user is not only able to know the instructor who has been assigned to a selected course/section, but also he can know the list of courses and sections assigned to a certain instructor. Obviously, checking whether a course/section has been assigned must be very intuitive.

When the user does the assignments, the TAP also suggests the available assignment candidates and can ensure that no conflict exists.

The user can also modify all the information about the courses, sections and instructors. This makes the teaching assignment program very flexible.

Furthermore, TAP displays the summary of assignment at all times which provides an immediate knowledge of how much of the task has been fulfilled.

3. Software Requirements Analysis

3.1 Functional Requirements

From the user perspective, TAP should provide the functionalities described in the previous section. The user should be able to view the current assignment state, perform the assignment and modify the database. In addition, the user should be able to view the assignment summary.

3.1.1 Use Cases

Based on the functional requirements described above, the major use cases are:

- ***View Courses/ Sections:***

  TAP displays the course/section list. When the user selects a section from the list, he or she can find out all the detailed information about the selected section such as the section name, scheduled time and so on. In case an instructor has been assigned to this section, TAP will show the instructor's name. Coordination is treated as a special section in the list. When a coordination is selected, detailed information such as course title and credits will be shown. The name of the coordinator will also be displayed if it has been assigned. Actually, this case contains three sub use cases according to whether the section has been assigned an instructor:

- *View all the sections*

  The user views both the assigned sections and the unassigned sections.

- *View assigned sections*

  The user views the assigned sections only.

- *View unassigned sections*

  The user views the unassigned sections only.

- ***View Instructors:***

TAP displays the instructor list. When the user selects an instructor from the list, he can know all the detailed information about the selected instructor including the fixed and the variable information. The fixed information includes such as the name, assigned workload and experience while the variable one includes those information such as actual workload and expected workload. If some courses or sections have been assigned to this instructor, a list of assigned courses/sections will be also displayed. According to whether the instructor's actual workload has reached his assigned workload, the following three sub use cases are also defined:

- *View all the instructors*

  The user views both the instructors with complete workload and the instructors with incomplete workload.

- *View the instructors with complete workload*

  The user views the instructors with complete workload only.

- *View the instructors with incomplete workload*

  The user views the instructors with incomplete workload only.

- ***Add Coordination Assignment:***

  There are two cases:

  - The user selects an unassigned course which requires a coordinator from the course list. Using the suggestible list of available instructor candidates provide by TAP, the user then chooses one of the candidates and then assigns the instructor to that course as its coordinator.
  - The user firstly selects an instructor from the list. With the suggestible available course candidates provided by the TAP program, the user chooses one of the candidates and then assigns the course coordination to the instructor's coordinating load.

  In both cases, the TAP should first check that no conflict exists before the assignment and then recalculate the instructor's actual workload and expected workload after the assignment.

- ***Delete Coordination Assignment:***

  There are two cases:

  - The user selects a course with a coordinator assigned to it, and then deletes the coordinator.
  - The user selects an instructor, and deletes one of the coordination

assignments from his working load.

In both cases, the TAP program should recalculate the instructor's actual workload and expected workload.

- ***Add Teaching Assignment:***

  There are two cases:

  - The user selects an unassigned section from the section list. With the suggested available instructor candidates list provide by TAP, the user chooses one of the candidates and then assigns the instructor to the selected section.
  - The user firstly selects an instructor from the list. With the suggested available section candidates provided by TAP, the user chooses one of the candidates and then assigns the teaching section to the instructor's teaching load.

  In both cases, the TAP should check that no conflict exists before the assignment and then recalculate the instructor's actual and expected workload after the assignment.

- ***Delete Teaching Assignment:***

  There are two cases:

  - The user selects a section with an instructor assigned to it, and then deletes the instructor.

- The user selects an instructor, and deletes one of the teaching assignments from his teaching load.

In both cases, TAP should recalculate the instructor's actual and expected workload.

- ***Modify the Information Data:***

  The user can manage the database in the following several ways:

  - Add a course
  - Modify a course
  - Remove a course
  - Add a section
  - Modify a section
  - Remove a section
  - Add an instructor
  - Modify an instructor
  - Remove an instructor

  TAP should provide a effective checking mechanism to ensure that conflict will not occur when the user modifies the data.

- ***View Assignment Summary:***

  TAP displays the summary data at all times so that the user can get a general concept of how the assignment progresses whenever he needs. The summary data includes the following information:

- The total number of the courses
- The total number of the coordinations
- The number of the coordinations with a coordinator assigned
- The total number of the sections
- The number of the sections with an instructor assigned
- The total number of the instructors
- The number of the instructors whose workload is complete

3.1.2 Use Case Diagram

The following use case diagram describes the major use cases presented above. The diagram incorporates the UML notations: (See the next page)

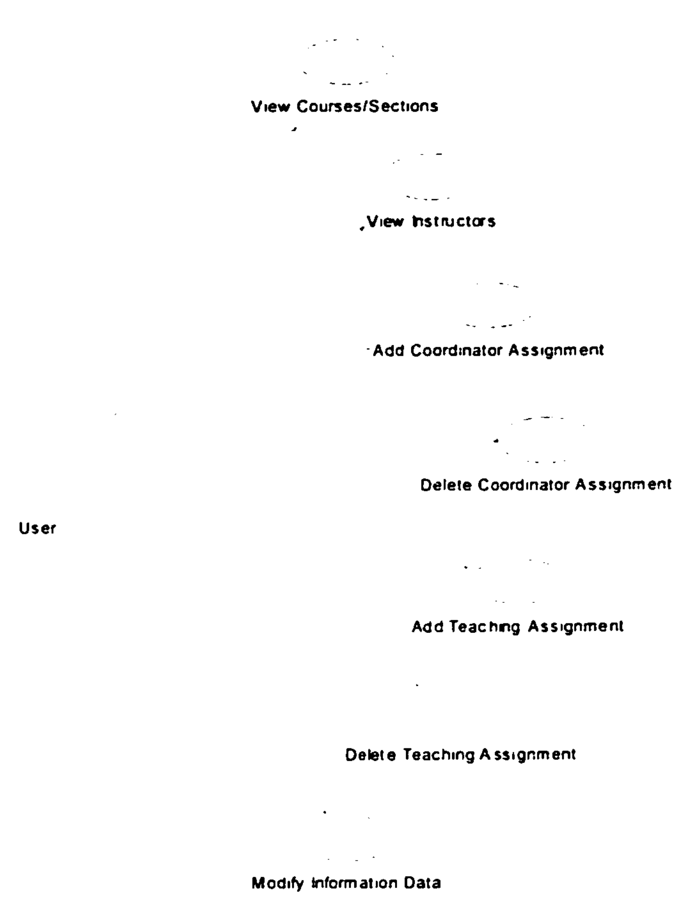


Figure 3-1: Use Cases

3.2 Non-Functional Requirements

3.2.1 Usability Requirements

TAP should be designed to be very easy for the user to learn how to use the program.

A good usability is important. TAP should have a user-friendly interface, fully supporting mouse and keyboard input. The user should be able to sit and use the program without reading the complicated help documents.

3.2.2 Integration and Migration Requirements

TAP should be required to integrate with the legacy system in the school and the storage database may also be required to change correspondingly. For example, TAP may be required to import database from the school system. It's an important issue that the TAP program should be designed to make it easy to integrate with other system or migrate to other database.

3.2.3 Recovery and Robustness Requirements

In the event of disastrous accident on a computer, such as sudden loss of electrical power and system crash, the TAP should not lose any information or data it is processing. The system should maintain the integrity of the information at all times. In addition, the user could perform all kind operations through the TAP program. Some of the operations may lead to error and TAP program should be robust enough to avoid the system to be crashed or accept an illegal assignment state.

4. System and User Interface Design

4.1 System Design

The *Teaching Assignment Planner (TAP)* program is a Java application which can run on any platform. TAP uses the MS-Access database to store the information data.

4.1.1 Design Rationale

A well-designed architecture is the foundation for a changeable and extensible system. There are some important issues that we have to consider during the architecture design of this system.

- **Usability**

  Since the user of TAP may have another background and have only basic computer operation skills, a good user interface is essential. Since a good user interface is often the result of many design iterations, the user interface of TAP will tend to experience changes as the system evolves. Consequently, it is vital to separate the user interface from the application logic so that changes in the user interface can be done easily without too much impact on the application logic. In order to allow new views to be easily connected to an existing domain layer, the domain objects should not have direct knowledge of or be directly coupled to views. Therefore, the MVC model is selected to fulfill these requirements. How should be the UI designed in an easy to use manner will be discussed in the UI design part (See Chapter 4.2).

- **Integration with Legacy Software System or Migration**

  Although the TAP program is currently an independent software, it is supposed to be integrated with the legacy software system in the university to share data in the future version. In so doing, the potentially involved database technology can be very diverse. Therefore, the application logic tier should be decomposed into finer layers so that the domain model is separate from the database interface. In this way, the database interface can be changed to adapt to the changes in the database while leaving the domain model unchanged.

- **Robustness and Security**

  As mentioned in the analysis part (See Chapter 3), the user of the TAP program may not a computer expert who will perform all kinds of operations through the program. Some of the operations will probably incur the system error. We must take into account how to make the TAP program robust and secure enough to prevent the potential mistakes.

4.1.2 System Architecture Overview

TAP is a three-tier architecture. Figure 4-1 illustrates the architectural overview of

TAP program.

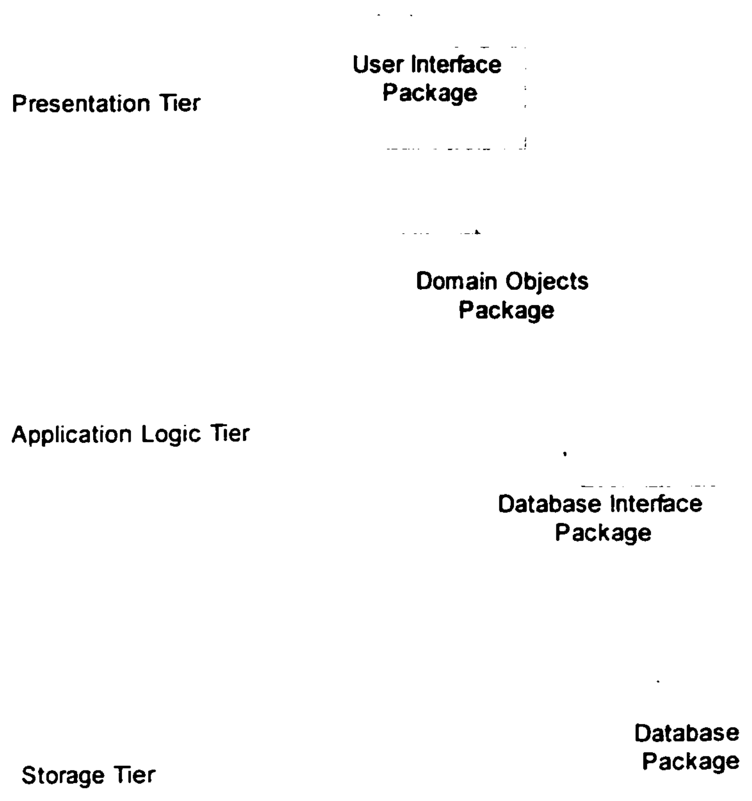


Figure 4-1: Overview of Architecture

The User Interface package represents the presentation tier. The User Interface package includes classes for the entire user interface, which enables the user to enter new data and view data from the system. These classes are based on the Java Swing package, which is a standard library in Java for writing user-interface applications. This package cooperates with the Domain Objects package.

The application logic tier is composed of the Domain Objects package and Database Interface package. The Domain Objects package includes the domain classes from the analysis model such as Course, Section, Instructor, Assignment, etc. The Database Interface package is based on the JDBC, which is a standard library in Java for accessing to various popular commercial database systems.

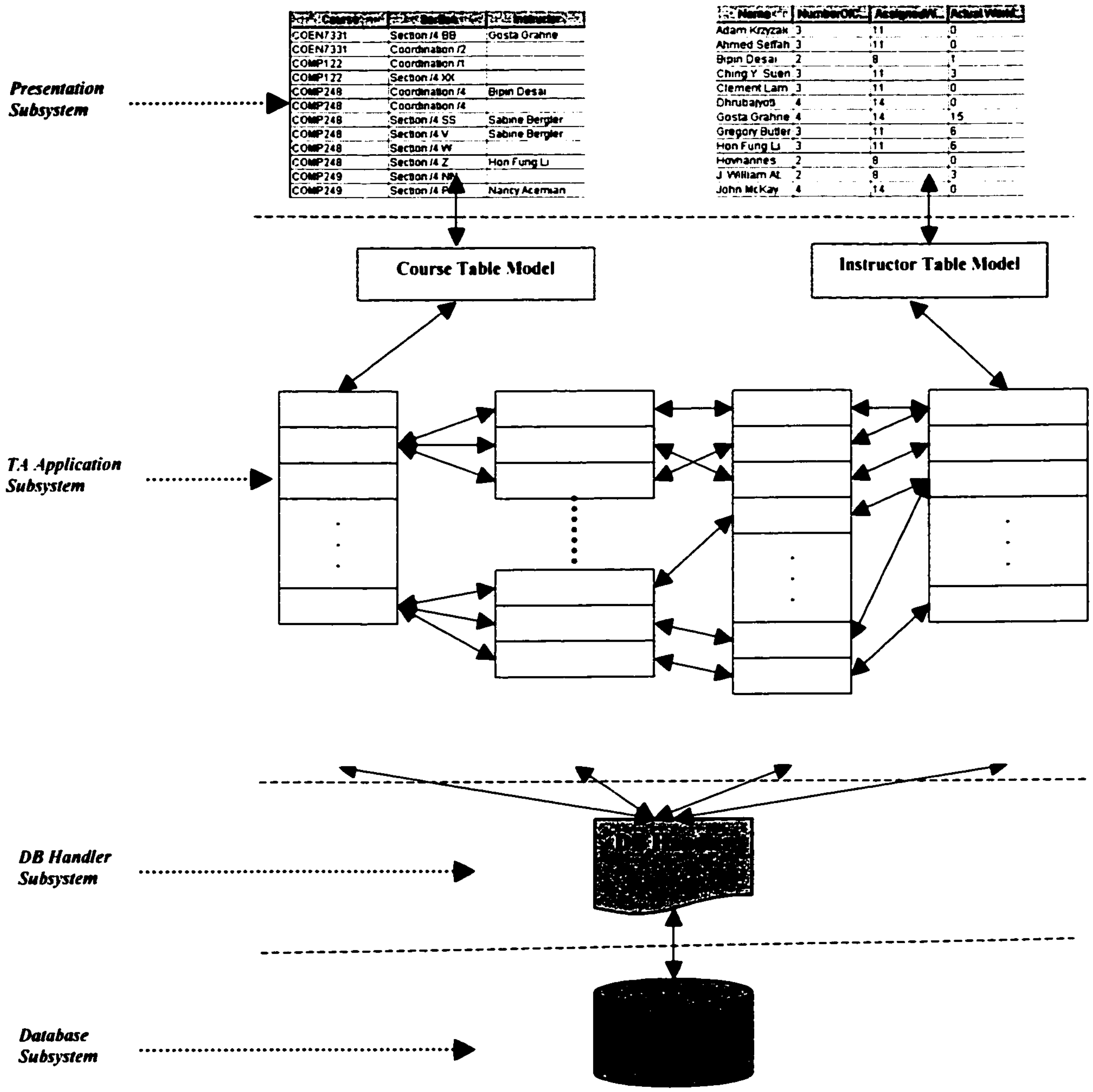
The Database package represents the storage tier. This package supplies services to the Database Interface package so that classes in the Domain Objects package can be stored persistently.

4.1.3 Subsystems

In this part, we discuss TAP components in detail. According to the design rationale and system architecture discussed before, naturally, there are four subsystems in the TAP program. The four subsystems are listed as follows:

- Presentation Subsystem
- Teaching Assignment Subsystem
- Database Handler Subsystem
- Database Subsystem

The following Figure 4-2 describes these four subsystems of the TAP program:



Course	Section	Instructor
COEN7331	Section /4 BB	Gosta Grahne
COEN7331	Coordination /2	
COMP122	Coordination /1	
COMP122	Section /4 XX	
COMP248	Coordination /4	Bipin Desai
COMP248	Coordination /4	
COMP248	Section /4 SS	Sabine Bergler
COMP248	Section /4 V	Sabine Bergler
COMP248	Section /4 W	
COMP248	Section /4 Z	Hon Fung Li
COMP249	Section /4 N[illegible]	
COMP249	Section /4 P[illegible]	Nancy Acemian

Name	NumberOfC...	AssignedW...	Actual Work...
Adam Krzyzak	3	11	0
Ahmed Seffah	3	11	0
Bipin Desai	2	8	1
Ching Y. Suen	3	11	3
Clement Lam	3	11	0
Dhrubajyoti	4	14	0
Gosta Grahne	4	14	15
Gregory Butler	3	11	6
Hon Fung Li	3	11	6
Hovhannes	2	8	0
J William At.	2	8	3
John McKay	4	14	0

Figure 4-2: TAP Subsystems

4.1.3.1 Presentation Subsystem

The presentation subsystem provides a GUI to the user of the TAP program. Through the interface, the user is able to perform all kinds of operations related to the teaching assignment such as viewing the current assignment state, doing the teaching assignment, modify the information data and so on. Anything changed in this subsystem will not affect the other subsystems.

4.1.3.2 Teaching Assignment Application Subsystem

The TA application subsystem includes the domain classes such as Course, Course List, Sections, Section List, Instructor, Instructor List, Teaching Assignment, TA List, etc. The operations of these classes are completely defined. This subsystem works in corporation with the presentation subsystem above itself and manages the information data from the database subsystem through the DB handler subsystem.

4.1.3.3 DB Handler Subsystem

This subsystem takes the responsibility of database management serving as the interface between the TA application subsystem and the database subsystem. The database accessing is based on the JDBC, which is a standard library in Java for accessing to various popular commercial database systems.

4.1.3.4 Database Subsystem

The TAP program incorporates the MS-Access database which supports various data types for storage purpose. This subsystem stores Course table, Section table, Instructor table and Teaching Assignment table which save course information, section information, instructor information and information of assignment respectively.

4.2 User Interface (UI) Design

Since the TAP program is designed for assisting purpose, the user interface (UI) of the program plays the most important role. UI mediates between the user and the TAP program. It should be very intuitive and easy to use and the user should be able to sit and learn to use it without consulting complicated help documents or tutorials. The UI helps users to understand system's functionalities, reflects the system model to them and translates their intentions into appropriate system activity. A good UI helps the user form a model, known as the user's mental model, of how the application works. This model forms the basis for future interactions with the system and enables users to predict system performance.

The Teaching Assignment Planner program is based on Java technology. Its UI is created by Java Swing component. Since the UI is designed independently and the domain objects should not have direct knowledge of or be directly coupled with views in the UI part, MVC pattern is selected accomplish the this task.

4.2.1 Overview of the MVC Design Pattern

'MVC' stands for Model-View-Controller. MVC consists of three kinds of objects. The Model is the application data, the View is its screen presentation, and the Controller defines the way the user interface reacts to user input. MVC decouples views and models by establishing a subscribe/notify protocol between them. [DP94] The following figure shows the communications among the Model, View and Controller:

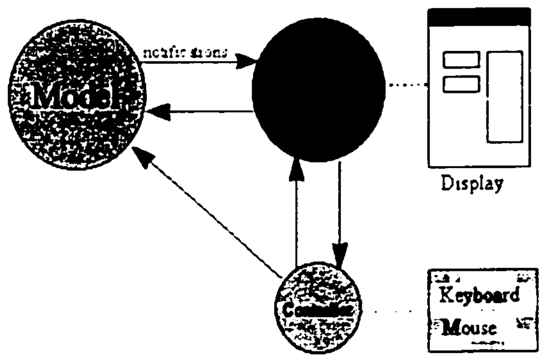


Figure 4-3: MVC Pattern

In the MVC paradigm the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by these three types of object, each specialized for its task.

- The View manages the graphical and/or textual output to the portion of the bitmapped display that is allocated to its application.

- The Controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate.
- Finally, the Model manages the behaviour and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). [MVC02]

Not surprisingly, several benefits will be achieved after the MVC pattern applied to the UI design. We can present different views according to the need while keeping the same model data exists in the application domain without redefining the object classes. [MVC98]

4.2.2 Overview of Java

Java is an object-orient programming language developed by Sun Microsystems Inc. Besides the OO features, Java provides cross platform capabilities. As with other high-level computer languages, Java source compiles to low-level machine instructions. In Java, these instructions are known as bytecodes. They are platform-independent instructions, which interact with Java Virtual Machine (JVM). The JVM is a separate program optimized for the specific platform on which the Java bytecodes are executed.

The Java language provides a powerful addition to the tools that programmers have at their disposal. Java makes programming easier because it is object-oriented and has automatic garbage collection. In addition, because compiled Java code is architecture-neutral, Java applications are ideal for a diverse environment like the Internet. [JLO02]

The following figure briefly describes the Java component structure and shows how Java can maintain platform independence:

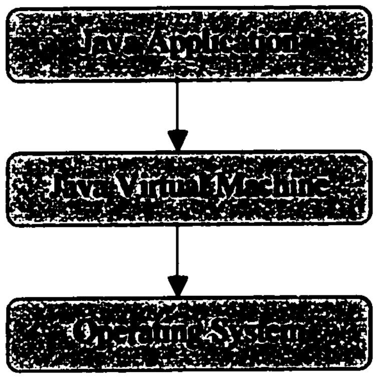


Figure 4-4: Java Component Structure

4.2.2.1 Overview of Java Swing

The Swing package is part of the Java Foundation Classes (JFC) in the Java platform. The JFC encompasses a group of features to help people build GUIs; Swing provides all the components from buttons to split panes and tables. The classes that used to create the GUI components are part of the *Swing GUI component* from package *javax.swing*. These are newest GUI components of the Java 2 platform. *Swing* *components* are written, manipulated and displayed completely in Java (so-called *pure Java components*).

Inheritance Hierarchy of the Classes

The Figure 4-5 shows an inheritance hierarchy of the classes that define attributes and behaviour that are common to most Swing components. Each class is displayed with its fully qualified package name and class name. Much of each GUI component's functionality is derived from these classes.

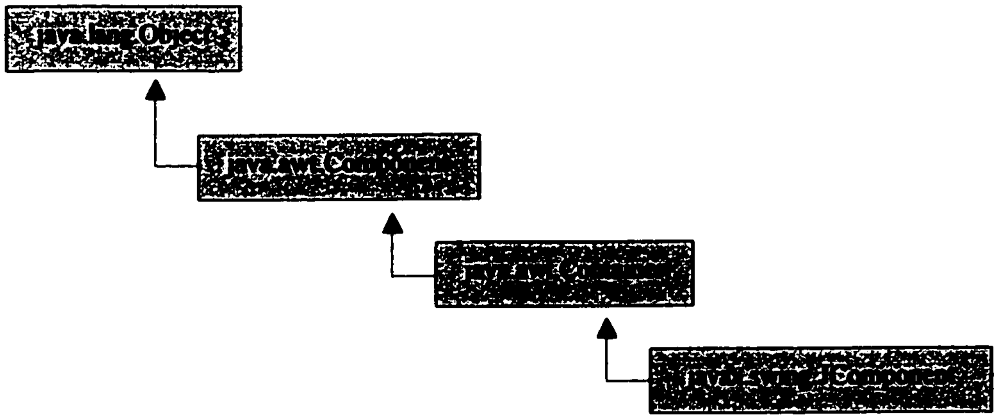


Figure 4-5: Common Superclasses of Many of the Swing Components

Class JComponent is the superclass to most Swing components. This class defines the set of methods that can be applied to an object of any subclass of any subclass of JComponent.

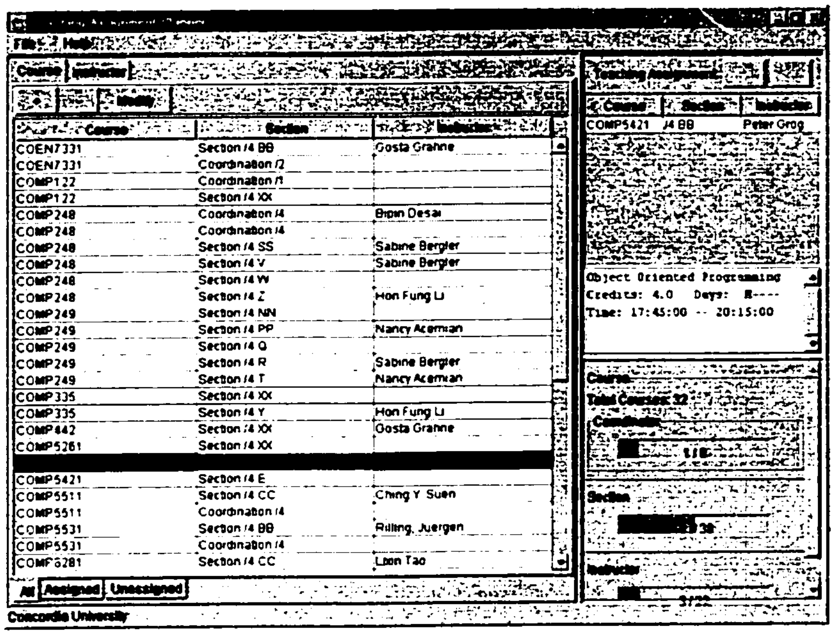
Java Swing Component's Features

Swing Component that is a subclass of JComponent has the following important features:

- A *pluggable look and feel* that can be used to customize the look and feel when the program executes on different platforms.
- Shortcut keys (called *mnemonics*) for direct access to GUI components through the keyboard.
- Common event handling capabilities for cases where several GUI components initiate the same actions in a program.
- Brief descriptions of a GUI component's purpose (called *tool tips*) that are displayed when the mouse cursor is positioned over the component for a short time.
- Support for assistive technologies such as Braille screen readers for blind people.
- Support for user interface *localization*—customizing the user interface for display in different languages and cultural conventions. [JSC02] [JHP99]

4.2.3 TAP GUI Design

The GUI design is really an important issue which decides whether the TAP program is easy to use. Based on the usability requirements, the TAP GUI is designed in this manner as shown in the following Figure 4-6 and Figure 4-7:

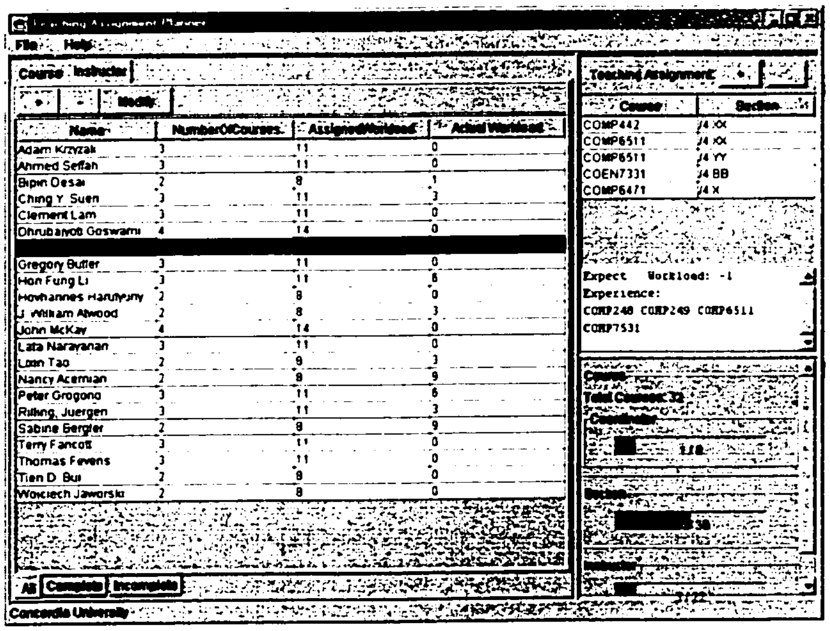


Course	Section	Instructor
COEN7331	Section /4 BB	Gosta Grahne
COEN7331	Coordination /2	
COMP122	Coordination /1	
COMP122	Section /4 XX	
COMP248	Coordination /4	Bipin Desai
COMP248	Coordination /4	
COMP248	Section /4 SS	Sabine Bergler
COMP248	Section /4 V	Sabine Bergler
COMP248	Section /4 W	
COMP248	Section /4 Z	Hon Fung Li
COMP249	Section /4 NN	
COMP249	Section /4 PP	Nancy Acemian
COMP249	Section /4 Q	
COMP249	Section /4 R	Sabine Bergler
COMP249	Section /4 T	Nancy Acemian
COMP335	Section /4 XX	
COMP335	Section /4 Y	Hon Fung Li
COMP442	Section /4 XX	Gosta Grahne
COMP5261	Section /4 XX	
COMP5421	Section /4 E	
COMP5511	Section /4 CC	Ching Y Suen
COMP5511	Coordination /4	
COMP5531	Section /4 BB	Rilling, Juergen
COMP5531	Coordination /4	
COMP6281	Section /4 CC	Loon Tao

Course	Section	Instructor
COMP5421	/4 BB	Peter Grog

```
Object Oriented Programming
Credits: 4.0   Days: M----
Time: 17:45:00 -- 20:15:00
```

Figure 4-6: Main GUI --- displaying the course/section list



Name	NumberOfCourses	AssignedWorkload	Actual Workload
Adam Krzyzak	3	11	0
Ahmed Seffah	3	11	0
Bipin Desai	2	8	1
Ching Y Suen	3	11	3
Clement Lam	3	11	0
Dhrubajyoti Goswami	4	14	0
Gregory Butler	3	11	0
Hon Fung Li	3	11	6
Hovhannes Harutyuny	2	8	0
J. William Atwood	2	8	3
John McKay	4	14	0
Lata Narayanan	3	11	0
Loon Tao	2	8	3
Nancy Acemian	2	8	9
Peter Grogono	3	11	6
Rilling, Juergen	3	11	3
Sabine Bergler	2	8	9
Terry Fancott	3	11	0
Thomas Fevens	3	11	0
Tien D. Bui	2	8	0
Wojciech Jaworski	2	8	0

Course	Section
COMP442	/4 XX
COMP6511	/4 XX
COMP6511	/4 YY
COEN7331	/4 BB
COMP6471	/4 X

```
Expect  Workload: -1
Experience:
COMP248 COMP249 COMP6511
COMP7531
```

Figure 4-7: Main GUI --- displaying the instructor list

As shown from the above main GUI of the TAP program, there are three major parts in the GUI:

- **Basic Information Display Area**

  Both the basic information of courses/sections and instructors will be displayed in the Basic Information Display Area. The user can switch the information by choosing the corresponding tab. In case of displaying course/section information, the information such as course title, credits and so on will be listed as a table. While displaying the instructor information, the information such as name, assigned workload and so on will also be listed as a table. The buttons set above the information list are used by the user to modify the information data.

- **Teaching Assignment Area**

  When a to be assigned course/section or instructor is selected by the user, its detailed information will be shown in the Teaching Assignment Area combined with the assignment information. For example, when an instructor is selected, not only his variable data like actual workload but also a list of his teaching/coordinating load will be presented to the user. By pressing the '+' or '-' button, the user can perform the teaching assignment here.

- **Summary Area**

  The summary statistics will be shown in this area, providing to the user a general concept of how the assignment progress. The summary is displayed at all times and varies immediately with the assignment performed.

In this TAP program, Java technology is used to form a component based GUI design. Based on object-oriented technology and natively applied the MVC pattern, Java Swing helps us design a GUI with more usability to achieve the goal of being easy to learn and easy to use.

5. Implementation

5.1 Presentation Subsystem

The presentation subsystem should provide a GUI to the user as his working place. Through the working palace, the user can perform the teaching assignment. The GUI should be developed in an intuitive manner so that it will be very easy to use. The problems encountered during the implementation period and their corresponding solutions will be illustrated in the following part.

5.1.1 Problems and Solutions

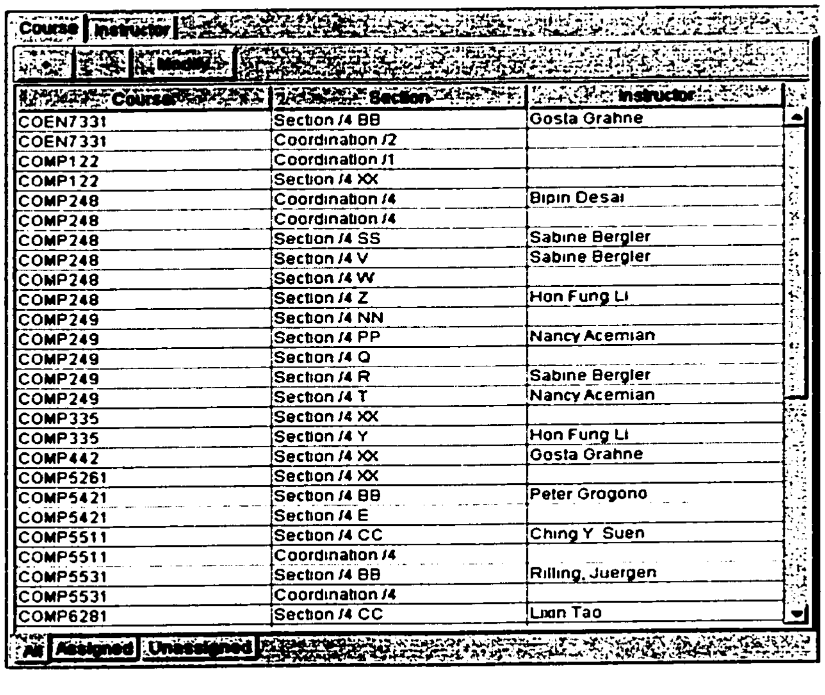
- ***Problem A: Displaying huge amount of information in limited space***

  There is a lot of information that must be presented to the user of the TAP program. The user needs to know both the courses/sections and the instructors, combined with their fixed and variable information data. For example, the information data of an instructor consists of name, assigned number of courses, assigned workload and teaching experience. In addition, if the instructor has already been assigned some workload, the teaching sections or coordinating courses with the detailed information should also be listed.

  The size of display area on the computer screen is limited and how to arrange showing the huge amount of information reasonably must be taken into account seriously. The user should get the information very directly and the information data should be organized a very clear and intuitive manner. It should not be designed in such a way that the user may get lost when he is try to find the information data he requires. And an over fancy style which could confuse the user should also be avoided.

Solution:

Due to the limited space of the displaying area, we should make good use of it for showing so much information. A Tabbed pane will meet this purpose and the courses/sections and the instructors' information will show in the same place. Upon the user's selection by clicking the tab, the selected one will be the current active content displaying to the user.



Course	Section	Instructor
COEN7331	Section /4 BB	Gosta Grahne
COEN7331	Coordination /2	
COMP122	Coordination /1	
COMP122	Section /4 XX	
COMP248	Coordination /4	Bipin Desai
COMP248	Coordination /4	
COMP248	Section /4 SS	Sabine Bergler
COMP248	Section /4 V	Sabine Bergler
COMP248	Section /4 W	
COMP248	Section /4 Z	Hon Fung Li
COMP249	Section /4 NN	
COMP249	Section /4 PP	Nancy Acemian
COMP249	Section /4 Q	
COMP249	Section /4 R	Sabine Bergler
COMP249	Section /4 T	Nancy Acemian
COMP335	Section /4 XX	
COMP335	Section /4 Y	Hon Fung Li
COMP442	Section /4 XX	Gosta Grahne
COMP5261	Section /4 XX	
COMP5421	Section /4 BB	Peter Grogono
COMP5421	Section /4 E	
COMP5511	Section /4 CC	Ching Y Suen
COMP5511	Coordination /4	
COMP5531	Section /4 BB	Rilling, Juergen
COMP5531	Coordination /4	
COMP6281	Section /4 CC	Lixin Tao

Figure 5-1: Tabbed Pane

- ***Problem B: Sharing the same data model***

  In the case of showing the information data, considered for the purpose of convenient use, these information data should be classified into different categories like All, Complete and Incomplete according to the instructor's workload. The information in these three categories comes from the same data source. In order to avoid the redundant design, how to sharing the same data source becomes important.

  Solution:

  Recalling the MVC pattern we described in Chapter 4.2.1, it will help us solve the problem of displaying information data.

  The principal advantage of the MVC pattern is to decouple the objects (Model) in the application domain from the Views and Controllers. As a result, the objects in the application domain will have no direct knowledge of how the presentation views are controlled. The other feature of MVC pattern is that one model may have multiple views. Based on these benefits, the MVC is naturally applied in the TAP program to solve the problem described above.

  In the case of displaying the instructors' information into three different categories, the displayed information data actually comes from the same data objects. In other words, showing the information data into three categories can be treated as three different Views and the data objects play the role of Model in MVC pattern. The three different views share the same model and the model doesn't care how many views exist and how these views are controlled.

Applying the MVC pattern, the TAP program will provide the desired view on request from the user.

- ***Problem C: Input data validation and integration checking***

  The TAP program has some requirements for the input data. For example, it requires the course code like 'COMP' to be a four-letter string while the course number like "6411" to be a three or four digital letter string. Also, it requires the user input the complete information without interruption.

  The user may input the wrong format data when he is doing some information modification work. Some mistakes are minor which will leave the whole program peaceful while other could be very awful which will deliver incorrect information to the TAP database. When the user input the information data, for example, the user wants to add an instructor, the user may leave some required information blank. It will probably affect the future assignment.

Solution:

In order to avoid incurring such error to the TAP program, we provide an input data validation and integration check mechanism to ensure the system stability and robustness.

Whenever the user inputs the data through the keyboard, for example, the user is trying to make some modification to a course's detailed information, the TAP program will check the data format to see whether the input data are acceptable before writing them back to the database. Also, the TAP program will not enable the 'OK' button available unless the user input all the required information data. With its help, the error caused by the user's incorrect input will be reduced to a low level.

- ***Problem D: Suggest the assignment candidates***

  For the convenience of the user, the TAP will suggest a list of candidates when doing the teaching assignment. But the candidates must have different availabilities which means some candidates will be more likely to be assigned to selected task than some others due to their respective conditions. The TAP should be somewhat intelligent to implement this idea. How should TAP arrange these candidates and give the user a very clear knowledge of different levels of availabilities?

Solution:

When a suggested candidates list pops to the user, these candidates will be grouped into four different availability levels. Each of the groups is set to four different colours standing for their availabilities respectively.

According to the conventional colour meanings considered from the usability perspective, the TAP program incorporates such a colour scheme: *Green* stands for perfect availability; *Blue* means that the candidate has the minor condition problem; *Yellow* means that the candidate has the major condition problem such as a too heavy workload; And *Red* stands for that the candidate has a critical problem warning the user not to choose this candidate. This will be further illustrated in the Chapter 6.

- ***Problem E: Displaying the summary***

The user needs to know some summary statistics at all times when he is doing the assignment. It should be better present this summary information in a concise and direct manner of what percent the assignment task has been completed.

Solution:

Instead of listing the boring statistics, the progress bars are used to present an intuitive knowledge to the user of how much assignment work he has finished.

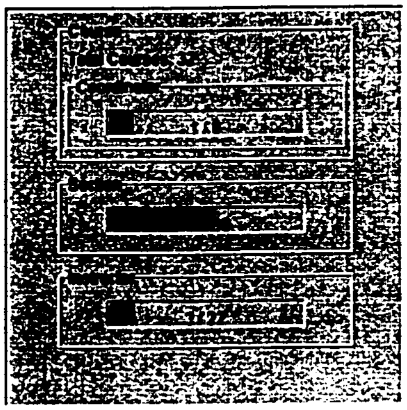
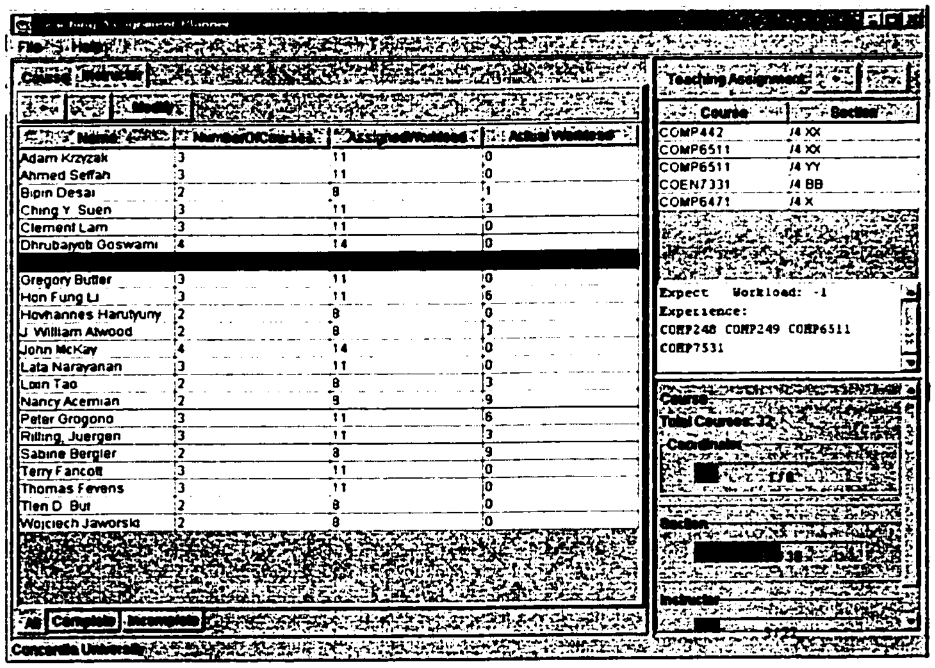


Figure 5-2: Progress Bar

5.1.2 Implementation

The GUI of the TAP program is presented in the form of split windows. It is implemented by the Java Swing Split Panel component.



Name	NumberOfCourses	AssignedWorkload	Actual Workload
Adam Krzyzak	3	11	0
Ahmed Seffah	3	11	0
Bipin Desai	2	8	1
Ching Y Suen	3	11	3
Clement Lam	3	11	0
Dhrubajyoti Goswami	4	14	0
[illegible]			
Gregory Butler	3	11	0
Hon Fung Li	3	11	6
Hovhannes Harutyuny	2	8	0
J William Atwood	2	8	3
John McKay	4	14	0
Lata Narayanan	3	11	0
Lixin Tao	2	8	3
Nancy Acemian	2	9	9
Peter Grogono	3	11	6
Rilling, Juergen	3	11	3
Sabine Bergler	2	8	9
Terry Fancott	3	11	0
Thomas Fevens	3	11	0
Tien D Bui	2	8	0
Wojciech Jaworski	2	8	0

Course	Section
COMP442	/4 XX
COMP6511	/4 XX
COMP6511	/4 YY
COEN7331	/4 BB
COMP6471	/4 X

```
Expect  Workload: -1
Experience:
COMP248 COMP249 COMP6511
COMP7531
```

Figure 5-3: Split Window

From the above figure, we can see that the left part of the main split window is the display area where the basic information of courses/sections and instructors are listed. The tabbed pane is also embedded to switch between the different information lists.

The right part of the main split window forms another split window in the vertical style. The top part is the teaching assignment area where the user performs the assignment work. It displays the further detailed information of a selected course/section or instructor.

The bottom part of this sub split window is the summary area showing the teaching assignment summary statistics.

The entire TAP GUI is implemented using the Java Swing components. The following class diagram shows the classed used to implement the TAP main GUI.

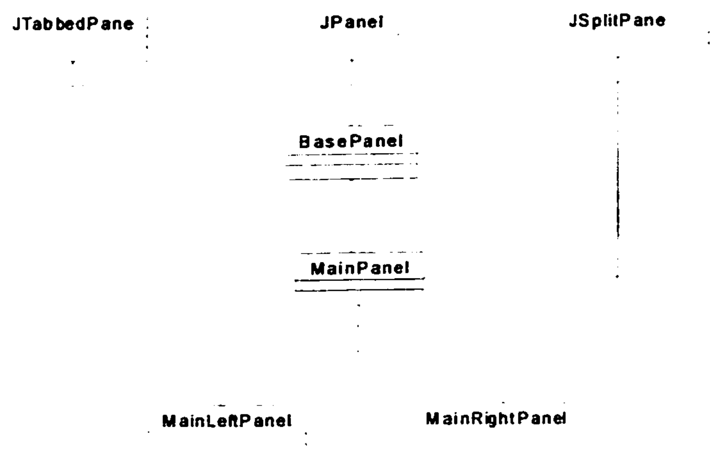


Figure 5-4: Class Diagram ---Main Panel

From this diagram, we can see that the class Main panel consists of the MainLeft and MainRight panel. Both the Main panel and the MainRight panel inherit from the class JsplitPane and the MainLeft panel inherits from the class JTabbedPane. The class diagrams of the main left panel and main right panel are also gives as follows:

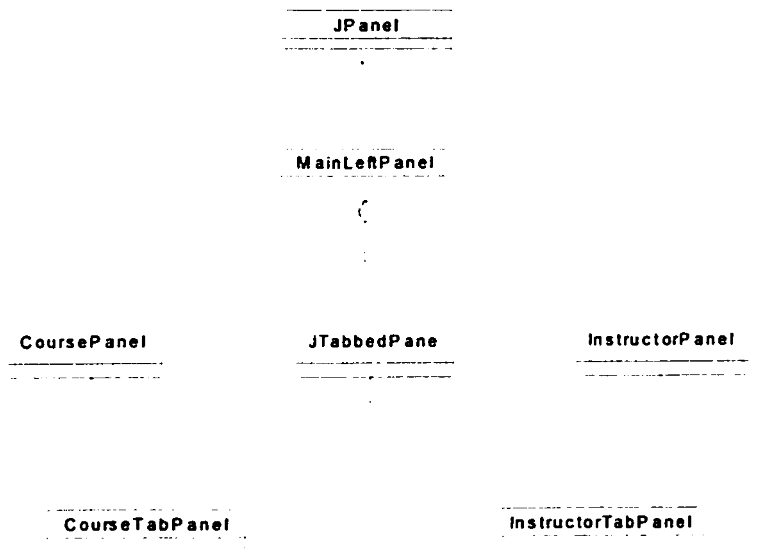


Figure 5-5: Class Diagram --- Main Left Panel

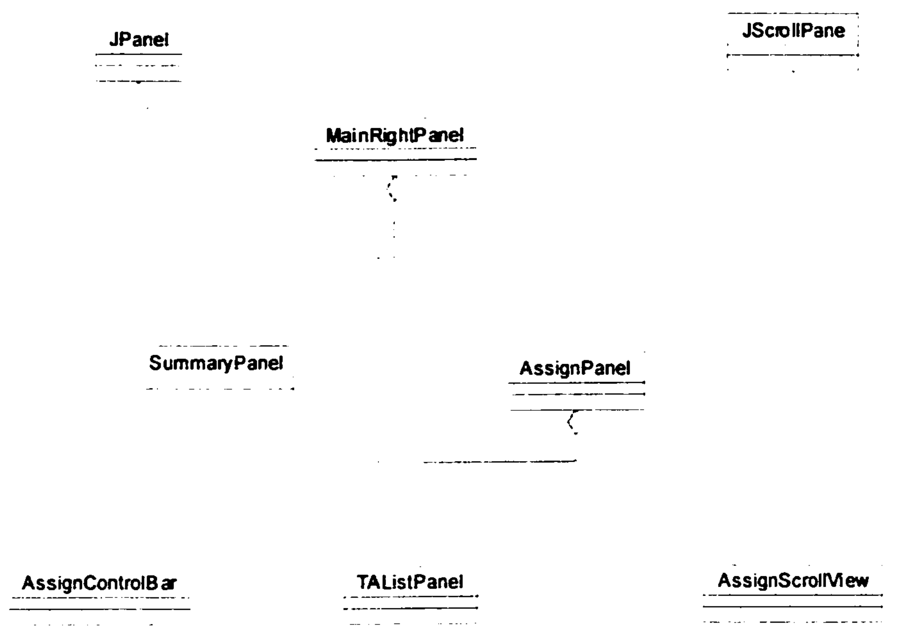


Figure 5-6: Class Diagram --- Main Right Panel

5.2 Teaching Assignment Application Subsystem

The TA application subsystem is the most important part of the TAP program. All the required functionalities are implemented in this subsystem. It's the soul of the whole TAP program.

5.2.1 Problems and Solutions

- ***Problem A: Courses and Sections***

  In Department of Computer Science of Concordia University, a number of courses are offered and a course usually comes with several different sections.

Each section has its own scheduled lecture time and requires an instructor respectively. Some of the courses require one or two coordinators assigned to them.

When the TAP user views the basic information of the courses, he also hopes to view their sections at the same time. Meanwhile, whether a course requires a coordinator should also be presented. How to deal with the course information combined with their sections in a concise and clear style becomes relevant.

Solution:

In the Course class, the sections are defined as the class member of it. The member Section is actually a list containing a list of sections. Here we treat the course coordination as a special section. If the course requires a coordinator, it exists as a special section in the section list.

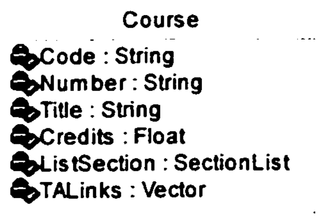
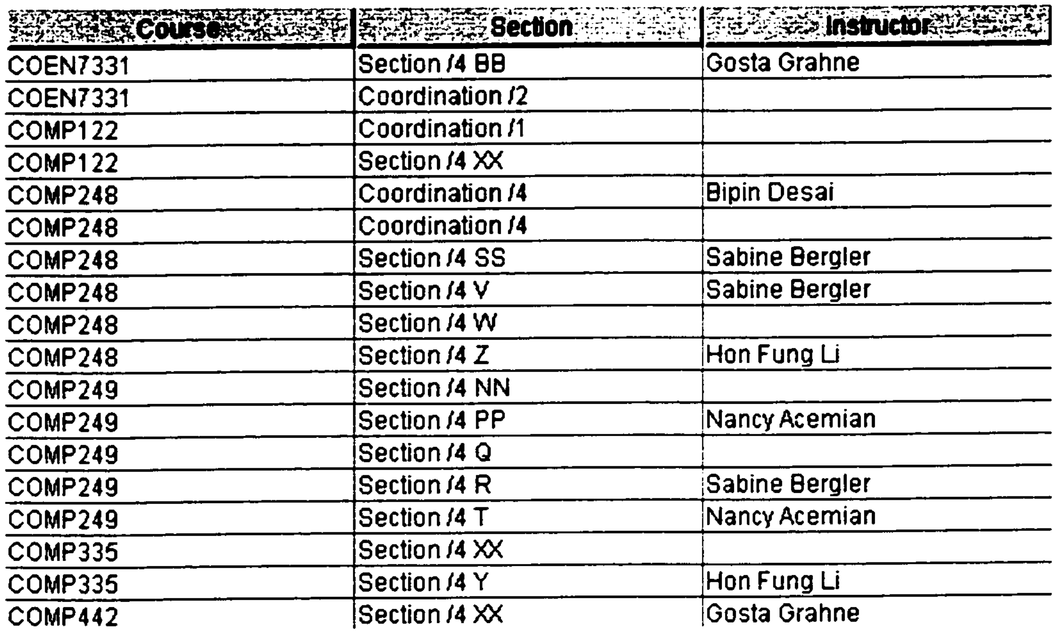


Figure 5-7: Class Course

As a result, from the GUI, the courses/sections information is organized in such a way as presented in the following figure:



Course	Section	Instructor
COEN7331	Section /4 BB	Gosta Grahne
COEN7331	Coordination /2	
COMP122	Coordination /1	
COMP122	Section /4 XX	
COMP248	Coordination /4	Bipin Desai
COMP248	Coordination /4	
COMP248	Section /4 SS	Sabine Bergler
COMP248	Section /4 V	Sabine Bergler
COMP248	Section /4 W	
COMP248	Section /4 Z	Hon Fung Li
COMP249	Section /4 NN	
COMP249	Section /4 PP	Nancy Acemian
COMP249	Section /4 Q	
COMP249	Section /4 R	Sabine Bergler
COMP249	Section /4 T	Nancy Acemian
COMP335	Section /4 XX	
COMP335	Section /4 Y	Hon Fung Li
COMP442	Section /4 XX	Gosta Grahne

Figure 5-8: Courses and Sections

- ***Problem B: Organization of the Data Structure***

A good program cannot be with out a good data structure. A good data structure can not only make the coding structure clear and easy to understand, but also it will help to make it easy to implement the required functionalities. Further important, it will make it quite flexible for future modification of the TAP program without much difficulty.

Solution:

The major data structure of the TAP program is organized in such a way described in the following figure and we will make the further illustration in the following part:

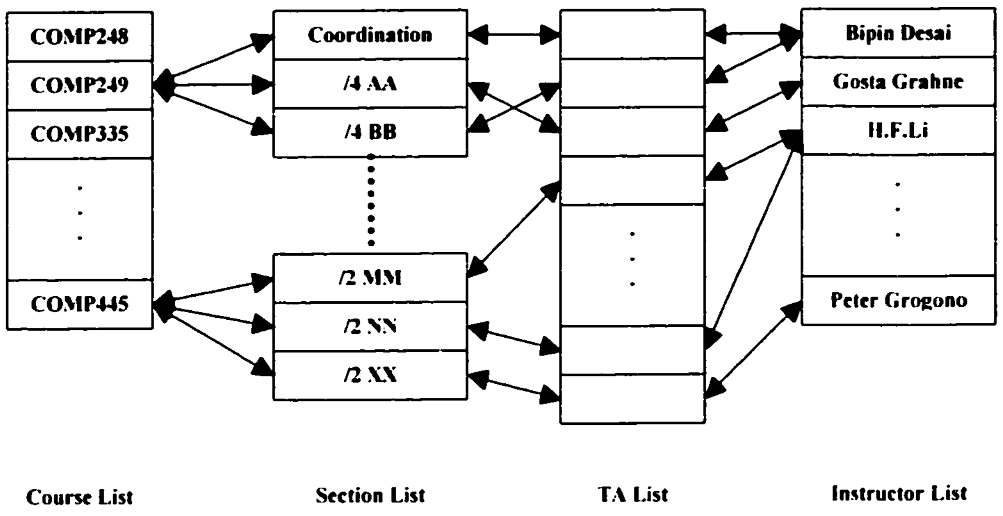


Figure 5-9: Data Structure

There are four lists in the TAP data structure. They are Course List, Section List, Instructor List and Teaching Assignment List.

The Course List contains a list of courses. Each course stores a pointer pointing to the sections which belong to the course. From the above the figure, the course COMP6511 has a pointer pointing to the sections '/2 MM', '/2 NN' and '/2 XX' respectively.

The Section List consists of several sections which belong to a common course. The coordination is stores as a special type section here. The section has the pointer pointing to its parent course and pointing to the related teaching assignment.

The Instructor List is made of instructors. Actually, the list is implemented by using Java Vectors and each instructor is stored as an object in the vector. The instructor has the pointer pointing to the related teaching assignment.

The Teaching Assignment List consists of a list of assignment information. As described in the above figure, we can see that for each teaching assignment, it stores both the pointer pointing to the section and the pointer pointing to the instructor. In addition, the teaching assignment has the pointer pointing to the related course directly. This pointer is not shown in the Figure 5-9.

Implemented in the format of list, it will be very convenient to perform adding and deleting operations. Since the lists are implemented by using Java Vectors, it will also be very easy to locate a specified object in the list. Because of the fact that the teaching assignment only stores the pointers pointing the related course/sections and instructors, it will be easy and flexible to implement any assignment.

5.2.2 Implementation

5.2.2.1 Major Classes

The major classes in the TAP program are listed as follows:

- **Course**

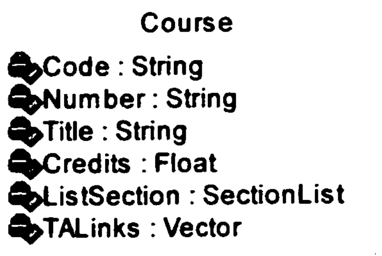


Figure 5-10: Class Course

- **Section**

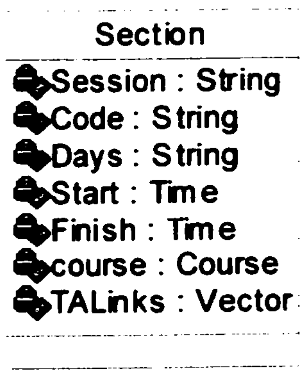


Figure 5-11: Class Section

- **Instructor**

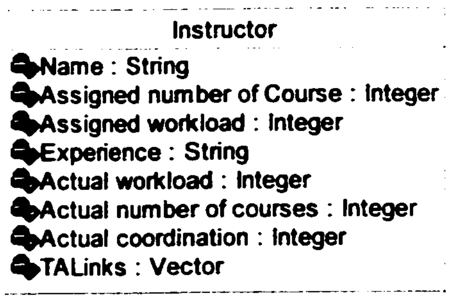

Instructor
Name : String
Assigned number of Course : Integer
Assigned workload : Integer
Experience : String
Actual workload : Integer
Actual number of courses : Integer
Actual coordination : Integer
TALinks : Vector

Figure 5-12: Class Instructor

- **Teaching Assignment**

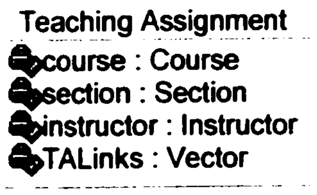

Teaching Assignment
course : Course
section : Section
instructor : Instructor
TALinks : Vector

Figure 5-13: Class Teaching Assignment

- **Course List**

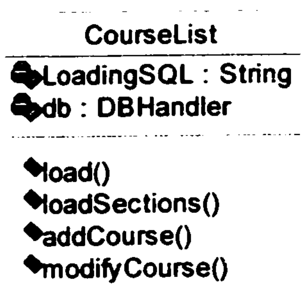

CourseList
LoadingSQL : String
db : DBHandler
load()
loadSections()
addCourse()
modifyCourse()

Figure 5-14: Class CourseList

- **Section List**

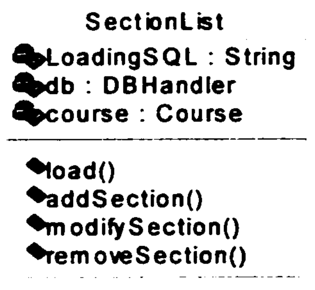


Figure 5-15: Class SectionList

- **Instructor List**

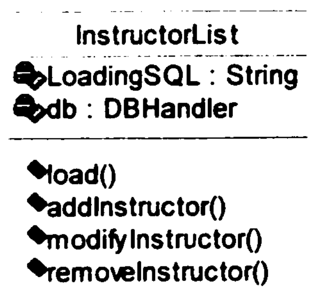


Figure 5-16: Class InstructorList

- **TA List**

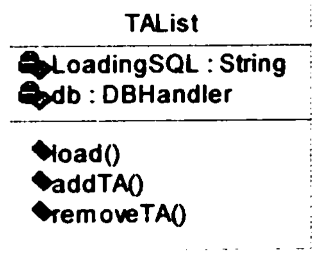


Figure 5-17: TAList

5.2.2.2 Class Diagram

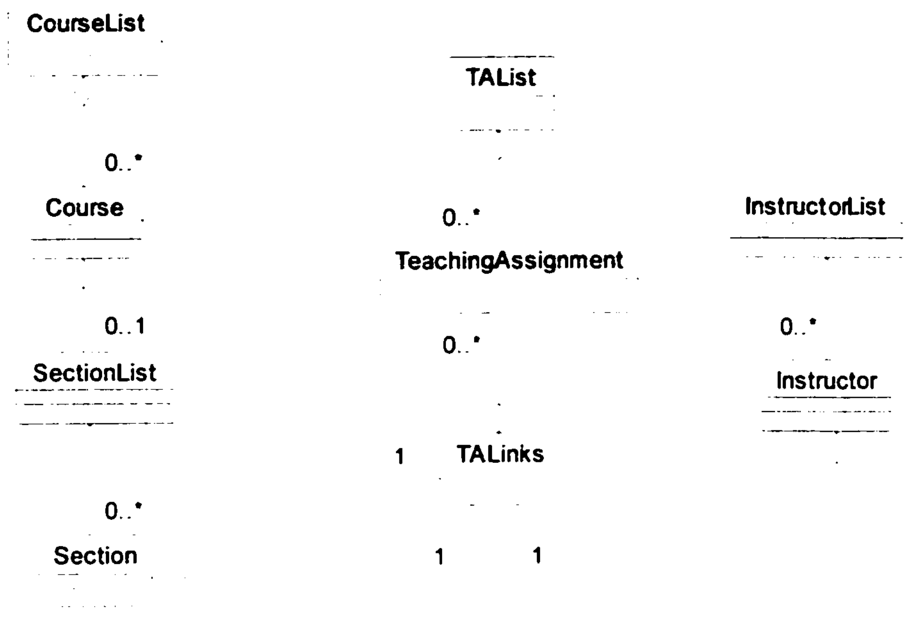


Figure 5-18: TA Application Class Diagram

From this above class diagram, we can obtain the knowledge of the relationships among these classes in the application logic domain. The Course List consists of several courses and the same cases are applied to the Section List, the Instructor List and the TA List. Each of the lists is responsible for administrating its own elemental data with the help of the DB handler subsystem.

In addition, class Course, class Section and the class Instructor are all have a TA Links, which is made up of several related Teaching Assignment assigned to them.

5.3 DB Handler Subsystem

The DB handler subsystem serves as the interface between the teaching assignment application subsystem and the database in the bottom. From the following figure, we can see how all the lists work through the DB handler to manipulate their data.

5.3.1 Overview

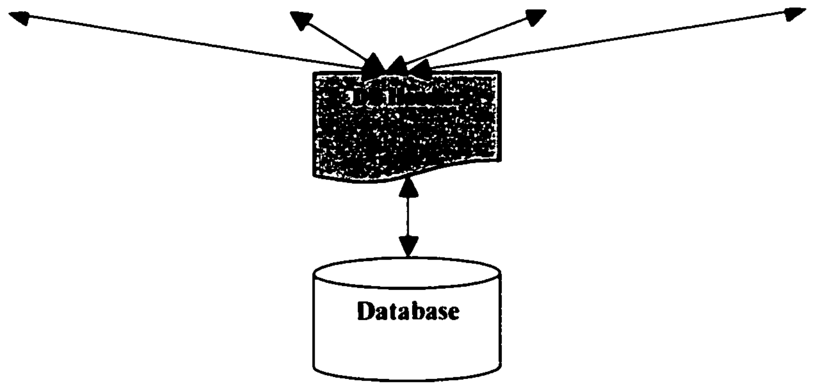


Figure 5-19: DB Handler Subsystem

All the operations performed on the data in the TA application must be through the DB handler. The DB handler provides all the necessary interfaces when the TA application retrieves the data from the database, modifies the data and writes back the data to the database.

5.3.2 Implementation

The DB handler subsystem provides a SQL interface to the course list, section list, instructor list and teaching assignment list. As a result, each of the lists is able to manipulate its own data through this SQL interface such as adding the data, modifying the data, removing data and so on.

One thing to be noticed is that whenever there is any change to those lists data, the change should be immediately reflected in the storage database. Thus the data in the database is always synchronized with the lists data and the database therefore keeps the latest data of the most recent teaching assignment state.

- ***DB Handler Class***

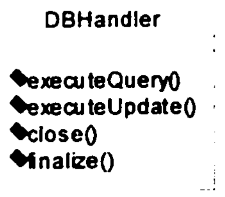


Figure 5-20: Class DBHandler

Let's take the instructor list in the TAP program for an example, through which we can illustrate the implementation of the DB handler subsystem and make clear the working mechanism of the DB handler. There is a DBHandler class which provides a public function named 'executeQuery'.

- public ResultSet executeQuery (String sQuery) {

  }

This function will be invoked by the instructor list to load the data from the database and initiate the list.

- public boolean executeUpdate (String query) {

  }

This function will be invoked by the instructor list to manipulate the instructor data such as adding, modifying or removing an instructor.

There is an InstructorList class standing for the instructor list. There are four major functions provided in this class which form the principal manipulations to the instructors. They are function 'load', 'addInstructor', 'modifyInstructor' and 'removeInstructor'.

- ***Load data to the instructor list***

  When the TAP program starts, the InstructorList should be responsible to initiate its data. It will invoke the 'excuteQuery' function in its own 'load' function to load the data from the database to the instructor list.

```
  ......
String sLoadingSQL = "select * from Instructor order by Name";
public void load ( ) {
  ResultSet rs = db.executeQuery (sLoadingSQL);

     ...

}
```

- ***Add an instructor to the list***

  In case of adding an instructor to the list, The InstructorList will first execute the 'executeUpdate' function in its own 'addInstructor' function in order to add this data to the database. Then it will add the instructor object to the instructor list.

```
public boolean addInstructor (Instructor newInstructor) {

   ......

  //insert to db

    String sInsertSQL = null;

        sInsertSQL = "......";

   ......

   ResultSet rs = db.executeQuery (sSQL);

   ......

  //add to InstructorList

  this.addElement (newInstructor);

   ......

}
```

- ***Modify an instructor***

When modifying an instructor, in the function 'modifyInstructor', the InstructorList executes the 'executeUpdate' function to update the data in the database.

```
public boolean modifyInstructor (Instructor newInstructor) {
  //update DB
  ......
    return db.executeUpdate(sUpdateSQL);
 }
```

- ***Remove an instructor***

When an instructor will be removed, the 'executeUpdate' function will be also invoked and the instructor data will be removed from the database before removing the instructor object from the instructor list.

```
public void removeInstructor (Instructor instructor) {
 //update DB
  ......
  if (db.executeUpdate(sDelSQL)){
    this.remove(instructor);
  }
}
```

5.4 Database Subsystem

In the TAP program. the MS-Access database is selected for storage purpose. It is one of the most popular commercial databases. It provides the JDBC driver to support the Java Program.

5.4.1 Overview of MS-Access Database

MS-Access is a database product developed by Microsoft. Using Microsoft Access. you can manage all your information from a single database file. Within the file. divide your data into separate storage containers called tables; view. add. and update table data by using online forms; find and retrieve just the data you want by using queries; and analyze or print data in a specific layout by using reports. MS-Access allows users to view. update. or analyze the database's data from the Internet or an intranet by creating data access pages. [MAH00]

5.4.2 Overview of JDBC

JDBC technology is an API that lets user access virtually any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a wide range of SQL databases, and it also provides access to other tabular data sources, such as spreadsheets or flat files. The JDBC API allows developers to take advantage of the Java platform's 'Write Once, Run Anywhere ^{TM}' capabilities for industrial strength. cross-platform applications that require access to enterprise data. JDBC driver is the interface between Java and various DBMS. It converts program (and typically SQL) requests to a form needed by a particular database. With a JDBC technology-enabled driver, a developer can easily connect all corporate data even in a heterogeneous environment. So using the Java programming language in conjunction with JDBC provides a truly portable solution to writing database applications. [JDBC02]

5.4.3 Implementation

There are four tables in the TAP program. They are Course Table, Section Table, Instructor Table and Teaching Assignment Table. Each of them will be described below. And all of description will contain the table name, table fields and the data type of each of the fields.

- **Course Table**

  Table Course (

Index	AutoNumber
Code	Text (6)
Number	Text (6)
Title	Text (80)
Credits	Number

)

  Index: The table index

  Code: A four-letter string like 'COMP, ENCS'

Number: A three or four-digit string like '248,5421'

Title: The name of the course like 'Artificial Intelligence'

Credits: A number in the range 0 to 5 like '1.5, 3.0, 4.75'

- **Section Table**

Table Section (

Index	AutoNumber
CourseDBIndex	Number
Session	Text (2)
Code	Text (15)
Days	Text (7)
Start	Date/Time
Finish	Date/Time

)

Index: The table index

CourseDBIndex: The Course table index of the section's parent course

Session: A single digit indicating the term like '1,2,3,4'

Code: A one or two-letter string like 'A, XX'

Days: One or two days which the section is scheduled like 'T, WF'

Start: The start time of the lecture like '10:15'

Finish: The finish time of the lecture like '20:45'

- **Instructor Table**

Table Instructor (

Index	AutoNumber
Name	Text (255)
NumberofCourses	Number
AssignedWorkload	Number
Experience	Memo

)

Index: The table index

Name: A string of characters like 'Peter Grogono'

NumberofCourses: The number of courses that the instructor should teach

AssignedWorkload: The number of 'points' for the instructor's workload

Experience: The coordinating or teaching experience of the instructor

- **Teaching Assignment Table**

Table Teaching Assignment (

Index	AutoNumber
InstructorDBIndex	Number
CourseDBIndex	Number
SectionDBIndex	Number

)

Index: The table index

InstructorDBIndex: The Instructor table index of the instructor

CourseDBIndex: The Course table index of the section's parent course

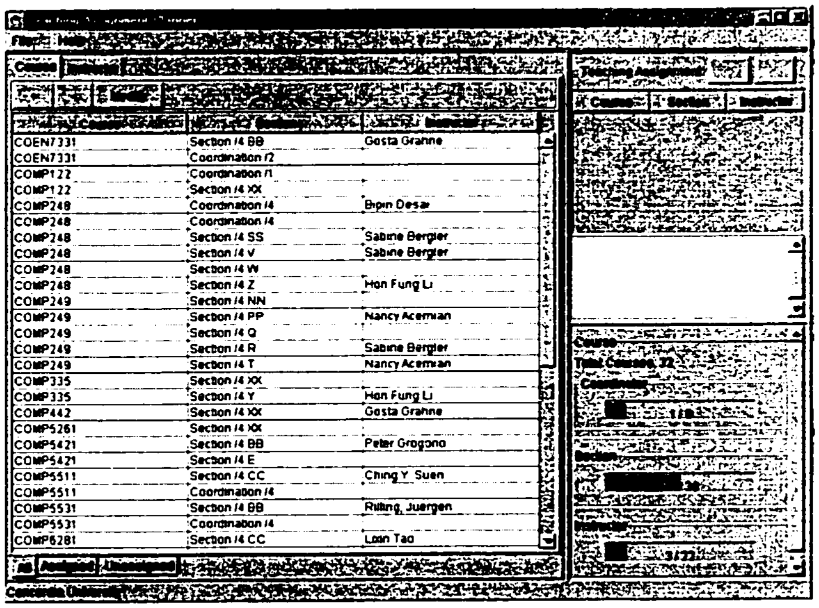
SectionDBIndex: The Section table index of the section

6. Results

In this chapter, we will discuss the implementation results of the TAP program and we will focus on the usability of the TAP program and illustrate how to use this program. Some screen shots are also provided to assist to make the issue clear.

The aim of the TAP program is to help the user fulfill the task of the teaching assignment without much difficulty. Through the TAP program, the user can view the current assignment state, perform teaching assignment, modify information data, view the summary statistics and so on.

After the user starting the TAP program, the main GUI will be presented to the user like the following figure:



COEN7331	Section /4 BB	Gosta Granne
COEN7331	Coordination /2	
COMP122	Coordination /1	
COMP122	Section /4 XX	
COMP248	Coordination /4	Bipin Desai
COMP248	Coordination /4	
COMP248	Section /4 SS	Sabine Bergler
COMP248	Section /4 V	Sabine Bergler
COMP248	Section /4 W	
COMP248	Section /4 Z	Hon Fung Li
COMP249	Section /4 NN	
COMP249	Section /4 PP	Nancy Acemian
COMP249	Section /4 Q	
COMP249	Section /4 R	Sabine Bergler
COMP249	Section /4 T	Nancy Acemian
COMP335	Section /4 XX	
COMP335	Section /4 Y	Hon Fung Li
COMP442	Section /4 XX	Gosta Granne
COMP5261	Section /4 XX	
COMP5421	Section /4 BB	Peter Grogono
COMP5421	Section /4 E	
COMP5511	Section /4 CC	Ching Y. Suen
COMP5511	Coordination /4	
COMP5531	Section /4 BB	Rilling, Juergen
COMP5531	Coordination /4	
COMP6281	Section /4 CC	Lixin Tao

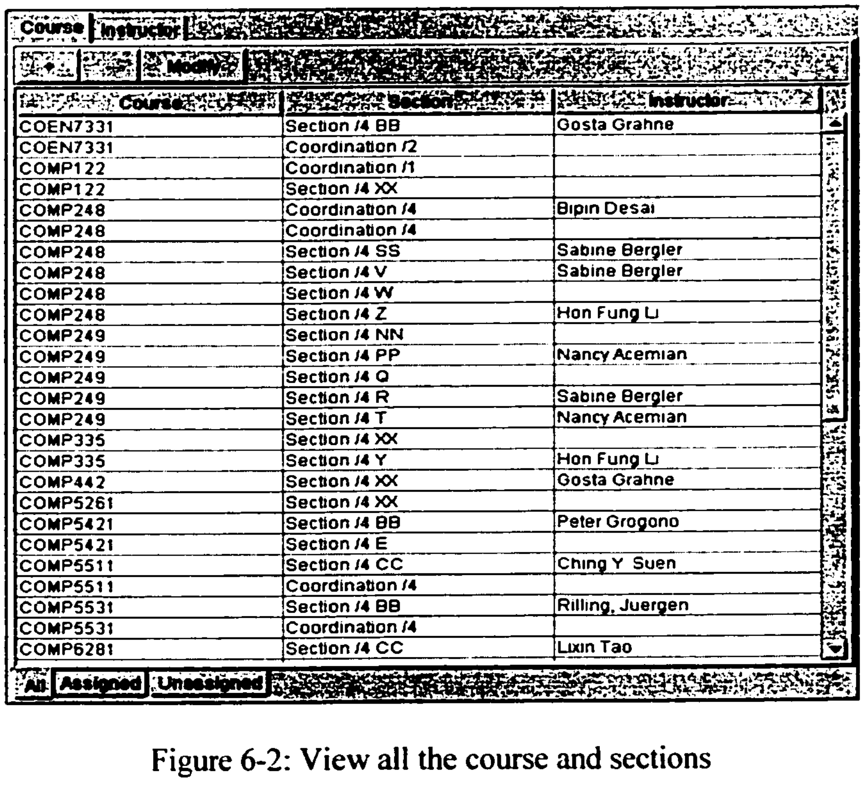
Figure 6-1: Main GUI

Based on this main GUI, the user can perform his assignment work such as viewing the current assignment state, performing teaching assignment, modifying information data and viewing the summary.

6.1 View the Assignment State

6.1.1 View the Courses and Sections

As shown from the main GUI, the left part of the split window is used to display the basic information. The user can view the courses with their sections. If a course requires the coordinator, it is listed as a special section.

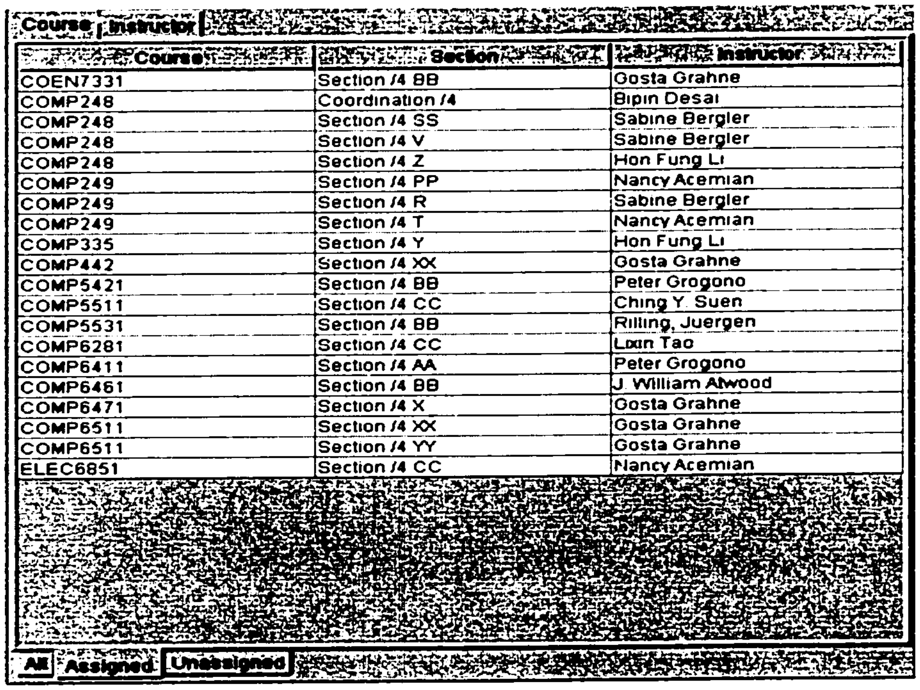


Course	Section	Instructor
COEN7331	Section /4 BB	Gosta Grahne
COEN7331	Coordination /2	
COMP122	Coordination /1	
COMP122	Section /4 XX	
COMP248	Coordination /4	Bipin Desai
COMP248	Coordination /4	
COMP248	Section /4 SS	Sabine Bergler
COMP248	Section /4 V	Sabine Bergler
COMP248	Section /4 W	
COMP248	Section /4 Z	Hon Fung Li
COMP249	Section /4 NN	
COMP249	Section /4 PP	Nancy Acemian
COMP249	Section /4 Q	
COMP249	Section /4 R	Sabine Bergler
COMP249	Section /4 T	Nancy Acemian
COMP335	Section /4 XX	
COMP335	Section /4 Y	Hon Fung Li
COMP442	Section /4 XX	Gosta Grahne
COMP5261	Section /4 XX	
COMP5421	Section /4 BB	Peter Grogono
COMP5421	Section /4 E	
COMP5511	Section /4 CC	Ching Y Suen
COMP5511	Coordination /4	
COMP5531	Section /4 BB	Rilling, Juergen
COMP5531	Coordination /4	
COMP6281	Section /4 CC	Lixin Tao

Figure 6-2: View all the course and sections

Information about the courses and the sections are displayed in a list. Each course comes with its course code and number, section and instructor if it has been assigned. The coordination has been arranged as a special section in the section column. It also lists its coordinator in the instructor column if it has been assigned.

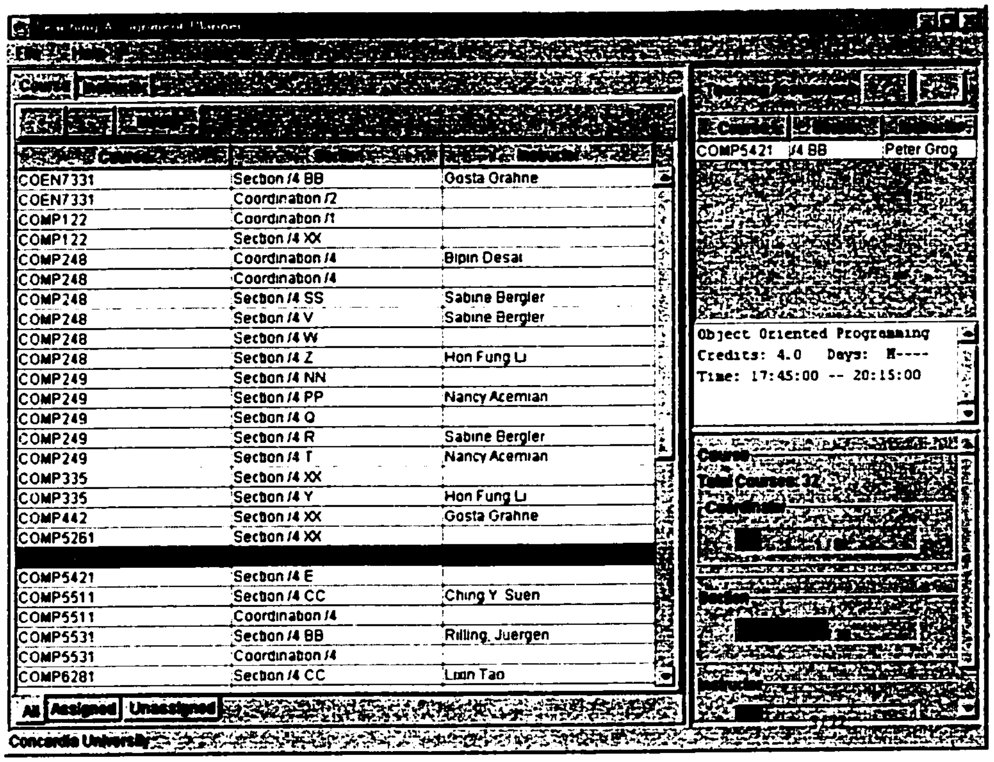
We notice that there are three tabs at the bottom part labelled 'All', 'Assigned' and 'Unassigned' respectively. By clicking the different tabs, the user also can view the courses and sections classified into "Assigned" and "Unassigned" categories according to whether the course or the section has been assigned. Below is the list of assigned courses and sections with the coordinators or the instructors' name.



Course	Section	Instructor
COEN7331	Section /4 BB	Gosta Grahne
COMP248	Coordination /4	Bipin Desai
COMP248	Section /4 SS	Sabine Bergler
COMP248	Section /4 V	Sabine Bergler
COMP248	Section /4 Z	Hon Fung Li
COMP249	Section /4 PP	Nancy Acemian
COMP249	Section /4 R	Sabine Bergler
COMP249	Section /4 T	Nancy Acemian
COMP335	Section /4 Y	Hon Fung Li
COMP442	Section /4 XX	Gosta Grahne
COMP5421	Section /4 BB	Peter Grogono
COMP5511	Section /4 CC	Ching Y. Suen
COMP5531	Section /4 BB	Rilling, Juergen
COMP6281	Section /4 CC	Lixin Tao
COMP6411	Section /4 AA	Peter Grogono
COMP6461	Section /4 BB	J. William Atwood
COMP6471	Section /4 X	Gosta Grahne
COMP6511	Section /4 XX	Gosta Grahne
COMP6511	Section /4 YY	Gosta Grahne
ELEC6851	Section /4 CC	Nancy Acemian

Figure 6-3: View assigned courses and sections

In addition, when a section is selected, the assignment information about which will be shown to the user in the top-right part of the split window along with the detailed information of the selected section.



COEN7331	Section /4 BB	Gosta Grahne
COEN7331	Coordination /2	
COMP122	Coordination /1	
COMP122	Section /4 XX	
COMP248	Coordination /4	Bipin Desai
COMP248	Coordination /4	
COMP248	Section /4 SS	Sabine Bergler
COMP248	Section /4 V	Sabine Bergler
COMP248	Section /4 W	
COMP248	Section /4 Z	Hon Fung Li
COMP249	Section /4 NN	
COMP249	Section /4 PP	Nancy Acemian
COMP249	Section /4 Q	
COMP249	Section /4 R	Sabine Bergler
COMP249	Section /4 T	Nancy Acemian
COMP335	Section /4 XX	
COMP335	Section /4 Y	Hon Fung Li
COMP442	Section /4 XX	Gosta Grahne
COMP5261	Section /4 XX	
[illegible]	[illegible]	[illegible]
COMP5421	Section /4 E	
COMP5511	Section /4 CC	Ching Y Suen
COMP5511	Coordination /4	
COMP5531	Section /4 BB	Rilling, Juergen
COMP5531	Coordination /4	
COMP6281	Section /4 CC	Lixin Tao

COMP5421	/4 BB	Peter Grog

```
Object Oriented Programming
Credits: 4.0   Days: M----
Time: 17:45:00 -- 20:15:00
```

Figure 6-4: Select a section

For example, in the left list, the section /4 BB of the course COMP 5421 is selected, the assignment information of it will be displayed in the teaching assignment area if it has an instructor assigned. Here Dr. Peter Grogono has been assigned to this section and his name is displayed in this assignment information. Below the assignment information, the detailed information of the selected section is also shown including the course title 'Object Oriented Programming', credits '4.0', the section's scheduled days 'M----' which indicates Monday and time ' 17:45—20:15'.

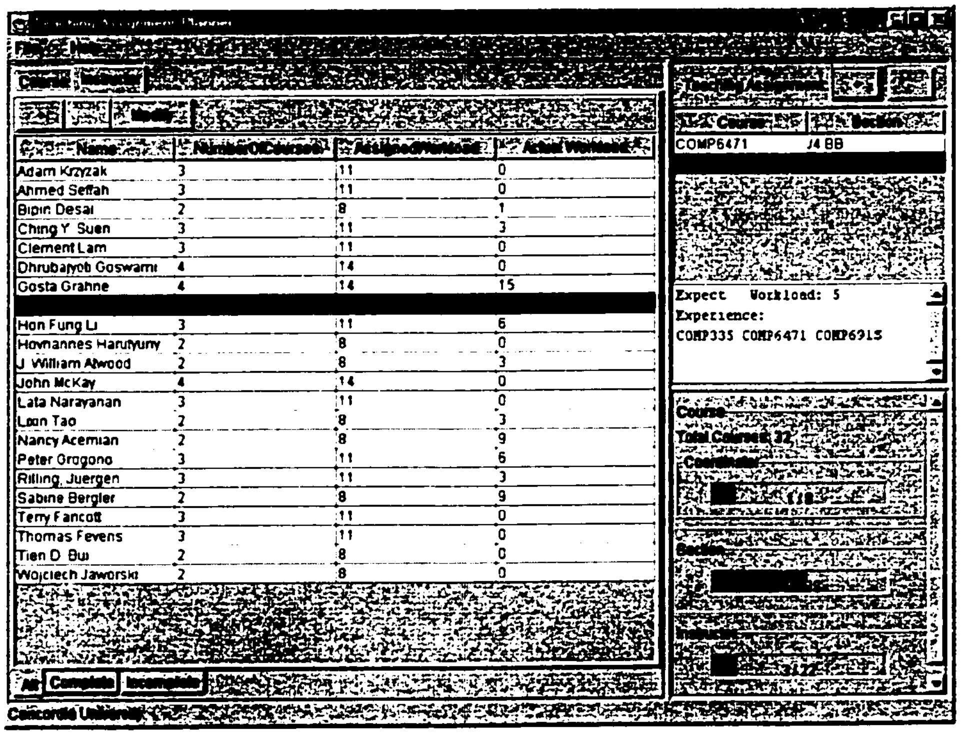
6.1.2 View the Instructors

Very similar to view the courses and sections, the TAP program also allows the user to view the instructors' information. The user can switch between the course and instructor information by clicking the corresponding tab. The two tabs are located on top of the information list as presented in the Figure 6-4. The basic information including the instructor's name, assigned number of courses, assigned workload and actual workload will be shown to the user.

Also, there are three tabs at the bottom part labelled 'All', 'Complete' and 'Incomplete' respectively. By clicking the different tabs, the user also can view the instructors classified into "Complete" and "Incomplete" categories according to whether the instructor's actual workload has reach his assigned workload.

In case of an instructor is selected, his assignment information will be shown to the user in the teaching assignment area with the detailed information of the selected instructor.

As shown in the following Figure 6-5, in the left list, Instructor Gregory Butler is selected, his assignment information is displayed in the teaching assignment area Here Butler has two teaching load: COMP6471 /4BB and COMP691S /4CC.Below the assignment information, the detailed information of Butler is also shown including his expected workload and experience.



Name	NumberOfCourses	AssignedWorkload	Actual Workload
Adam Krzyzak	3	11	0
Ahmed Seffah	3	11	0
Bipin Desai	2	8	1
Ching Y Suen	3	11	3
Clement Lam	3	11	0
Dhrubajyoti Goswami	4	14	0
Gosta Grahne	4	14	15
[illegible]	[illegible]	[illegible]	[illegible]
Hon Fung Li	3	11	6
Hovnannes Harutyuny	2	8	0
J William Atwood	2	8	3
John McKay	4	14	0
Lata Narayanan	3	11	0
Lixin Tao	2	8	3
Nancy Acemian	2	8	9
Peter Grogono	3	11	6
Rilling, Juergen	3	11	3
Sabine Bergler	2	8	9
Terry Fancott	3	11	0
Thomas Fevens	3	11	0
Tien D Bui	2	8	0
Wojciech Jaworski	2	8	0

Course	Section
COMP6471	/4 BB

Expect Workload: 5
Experience:
COMP335 COMP6471 COMP691S

Total Courses: 32

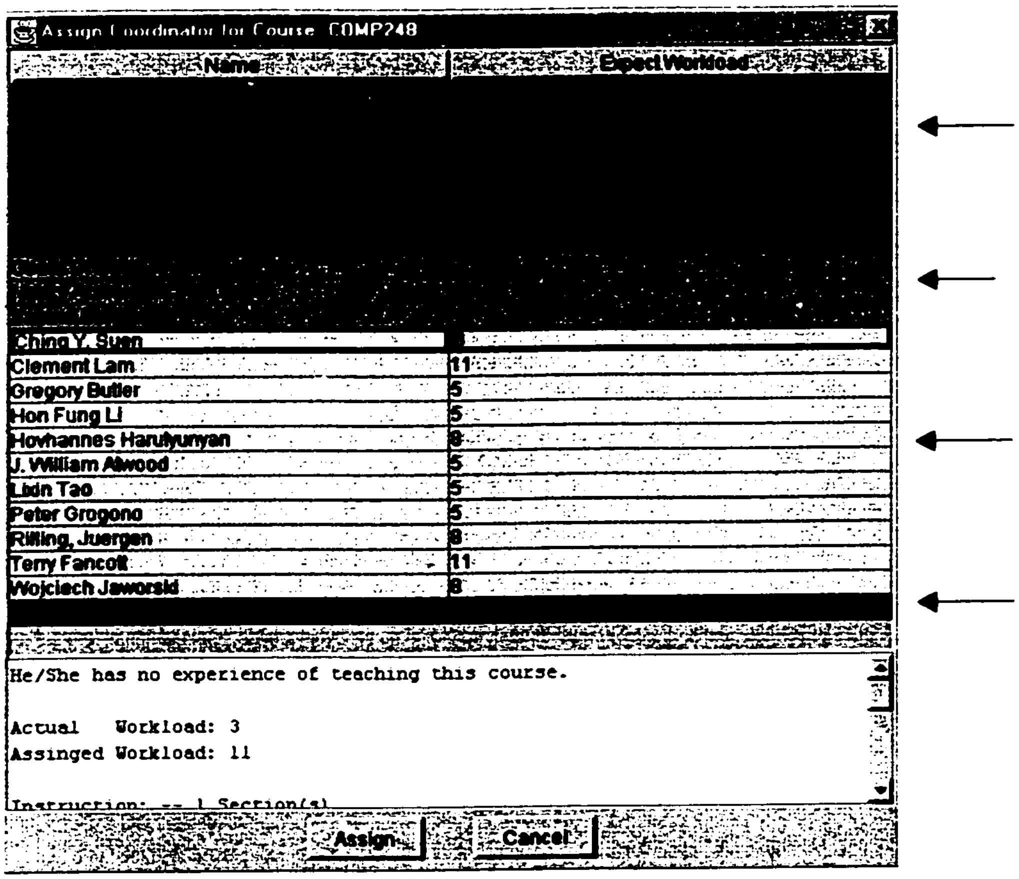
Figure 6-5: Select an instructor

6.2 Perform Teaching Assignment

After all the assignments, whether it is adding or deleting assignment, the TAP will recalculate the related instructor's actual workload and expected workload.

6.2.1 Add a coordinating assignment to a course

The user selects an unassigned course, pressing the '+' button in the teaching assignment area, and then a list of suggested instructors will be shown to the user before assignment. The suggestion list will be sorted by the instructors' availabilities level from high to low presented in different colours as shown in the following figure:



Name	Expect Workload
Ching Y. Suen	
Clement Lam	11
Gregory Butler	5
Hon Fung Li	5
Hovhannes Harutyunyan	8
J. William Atwood	5
Lixin Tao	5
Peter Grogono	5
Rilling, Juergen	8
Terry Fancott	11
Wojciech Jaworski	8

```
He/She has no experience of teaching this course.

Actual   Workload: 3
Assinged Workload: 11

Instruction: -- 1 Section(s)
```

Figure 6-6: Add a coordination assignment

As mentioned in the previous chapter, this colour scheme is incorporated here to help the user know whether the candidate is suitable. The colours stand for the following meaning:

- ***Green*** indicates that the instructor is perfect to assign with experience.
- ***Blue*** indicates that the instructor's actual workload will be over his assigned workload.
- ***Yellow*** indicates everything is OK for this instructor but without the course in his experience or his actual workload will be over16 points (high limit).
- ***Red*** indicates that the course has more than one coordinators, coordinator candidate is the same person as the assigned coordinators. (In this case, TAP doesn't allow the course to be assigned to him and the 'Assign' button is disabled.)

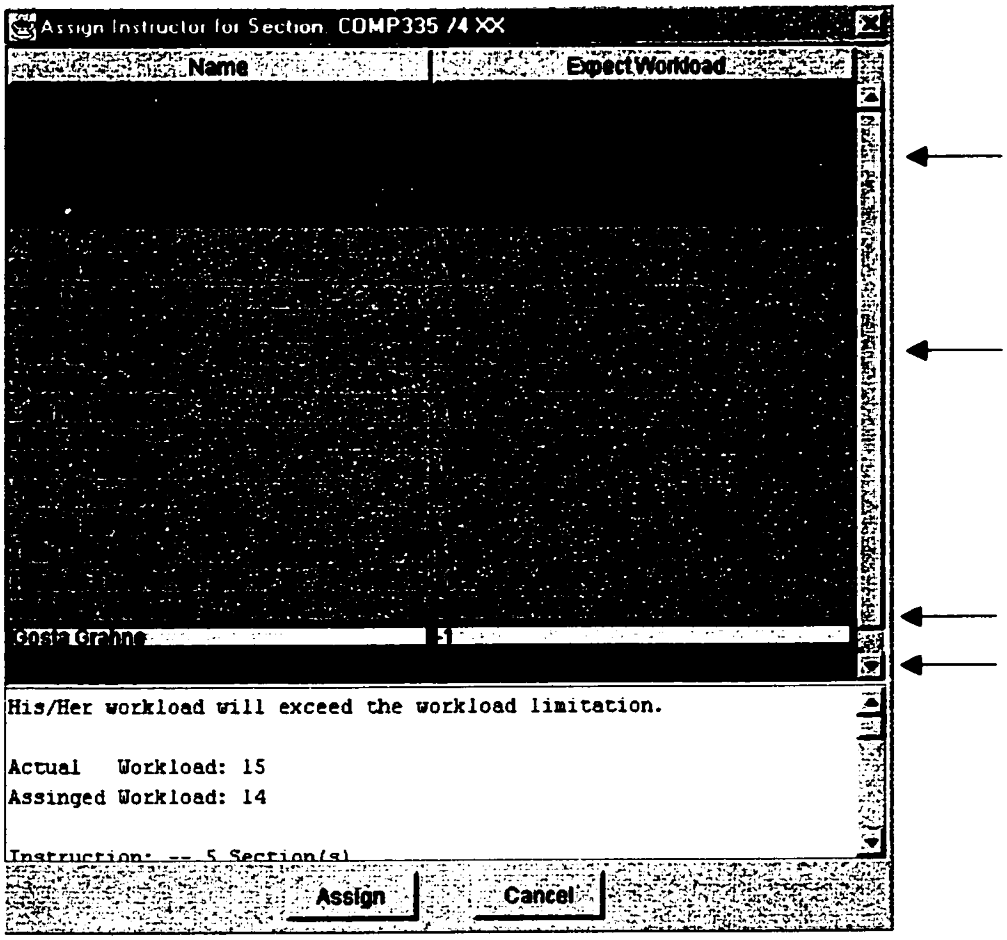
We can see from the above GUI, when the user select one of the candidates, the TAP program will tell the current working load of the candidate and the reason why he cannot or unsuitable for assigning.

6.2.2 Delete a coordinating assignment to a course

If the user selects one of the coordination assignment from the TA list and then press the '-' button, the TAP program will ask the user whether he really wants to delete this assignment. Once confirmed, the TAP program will remove this assignment.

6.2.3 Add a teaching assignment to a section

The user selects an unassigned section, pressing the '+' button in the teaching assignment area, and then a list of suggested instructors will pop to the user before assignment. The suggestion list will be sorted by the instructors' availabilities level from high to low presented in different colours as presented in the following figure:



```
His/Her workload will exceed the workload limitation.

Actual   Workload: 15
Assinged Workload: 14

Instruction: -- 5 Section(s)
```

Figure 6-7: Add a teaching assignment

The colours stand for the following meaning:

- ***Green*** indicates that the instructor is perfect to assign;
- ***Blue*** indicates that the instructor's actual workload will be over his assigned workload or the instructor's actual courses will be over his assigned courses, or the instructor has no experience for this course teaching;
- ***Yellow*** indicates the instructor's actual workload will be over 16 points (high limit);
- ***Red*** indicates that the instructor has time conflict with his assigned jobs. (In this case, TAP doesn't allow the section to be assigned to him and the 'Assign' button is disabled.)

6.2.4 Delete a teaching assignment to a section

If the user selects one of the teaching assignment from the TA list and then press the '-' button, the TAP program will ask the user whether he really wants to delete this assignment. Once confirmed, the TAP program will remove this assignment.

6.2.5 Add a coordinating /teaching assignment to an instructor

The user selects an instructor, pressing the '+' button in the teaching assignment area, and then a list of suggested courses/sections will pop to the user before assignment. If it's a coordinating assignment, the colours stand for the same meaning as the coordinating assignment to a course as mentioned above. If it's a teaching assignment, the colours stand for the same meaning as the teaching assignment to a section as mentioned above.

6.2.6 Delete a coordinating /teaching assignment to an instructor

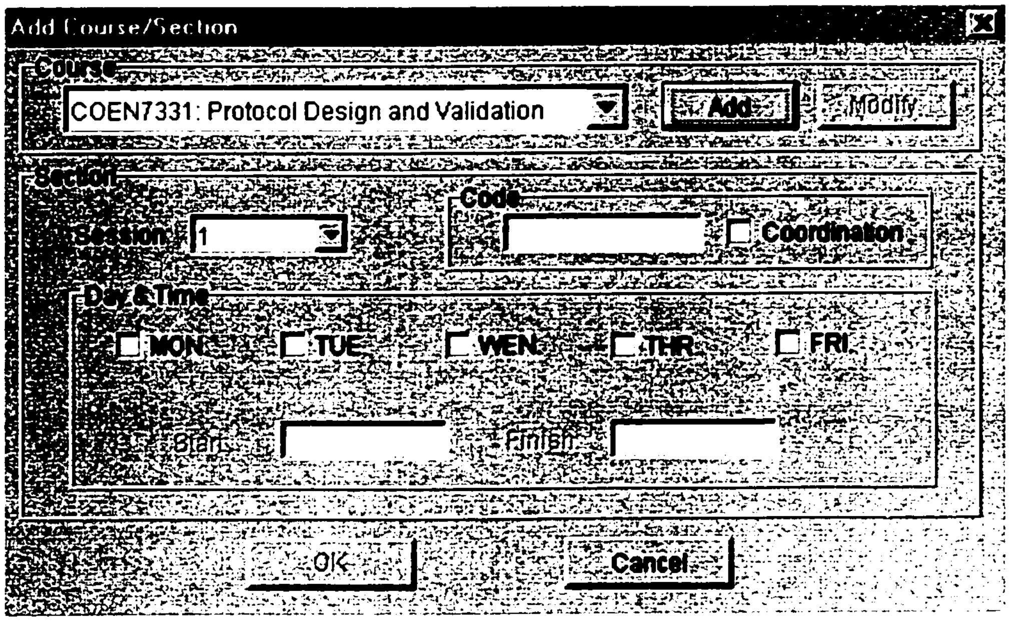
The user selects one of the working load of an instructor either coordination or teaching from the TA list in the teaching assignment area. and press the '-' button. the TAP program will make sure whether the user wants to delete the assignment. Once confirmed by the user. the TAP then remove the assignment from this instructor's working load.

6.3 Modify the Information Data

The user can modify either the information of courses/sections or the information of instructor through the TAP program. It can be achieved by pressing the '+'. '-' and 'Modify' buttons located above the information list.

6.3.1 Add a course

Pressing the '+' button on top of the display area. a GUI of adding a course/section will be shown as the Figure 6-8:

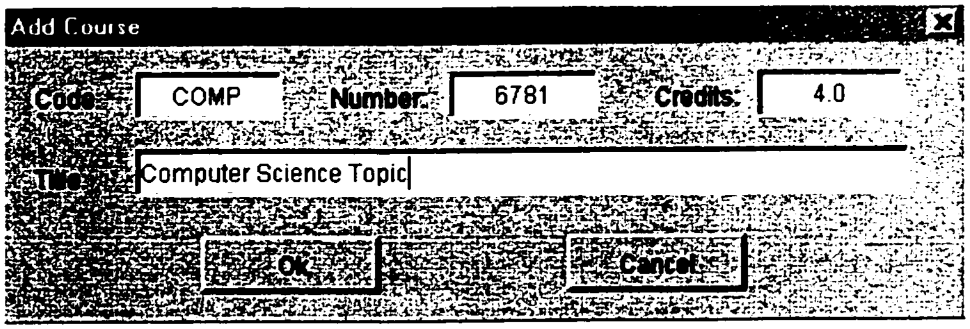


Course: COEN7331: Protocol Design and Validation

Session: 1

Figure 6-8: Add Course/Section

Pressing the 'Add' button in the course panel. the following GUI will be displayed:



Code: COMP Number: 6781 Credits: 4.0

Title: Computer Science Topic

Figure 6-9: Add a course

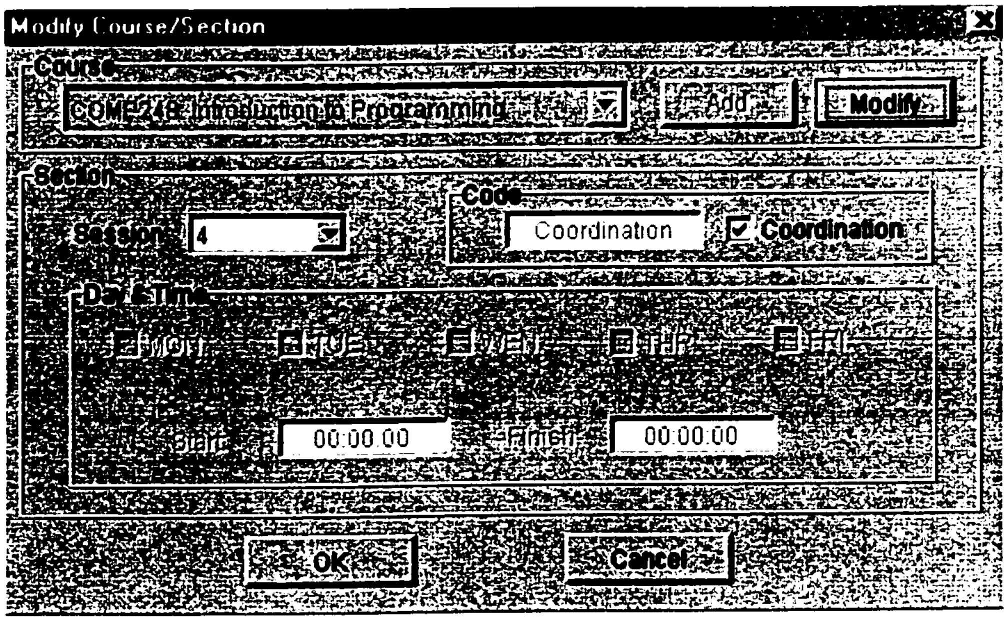
Provided with the above GUI, the GUI can fill the information data which will be checked by the TAP program for its validation and then press the 'OK' button to submit, or press the 'Cancel' button to cancel what has been filled.

6.3.2 Remove a course

The course without sections will not be shown in the course/section list and it only exists in the database, so we don't need to remove such course without any section.

6.3.3 Modify a course

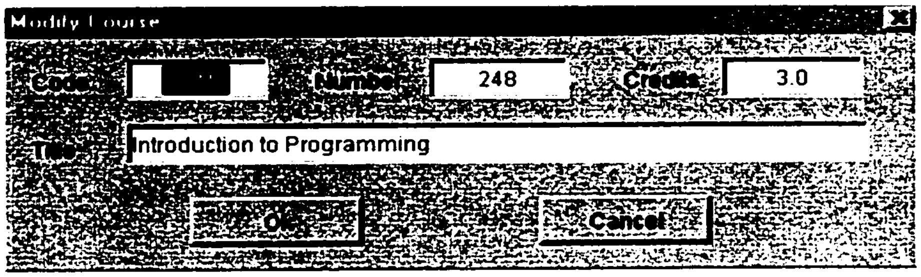
Select a course from the list, pressing the 'Modify' button, the following GUI will be shown. Here for example, we select the course COMP 248:



Course: COMP248 Introduction to Programming
Session: 4
Code: Coordination
[✓] Coordination
Start: 00:00:00
Finish: 00:00:00

Figure 6-10: Modify Course/Section

Pressing the 'Modify' button in the course panel above, the following GUI will be displayed. And the user can modify the course information.

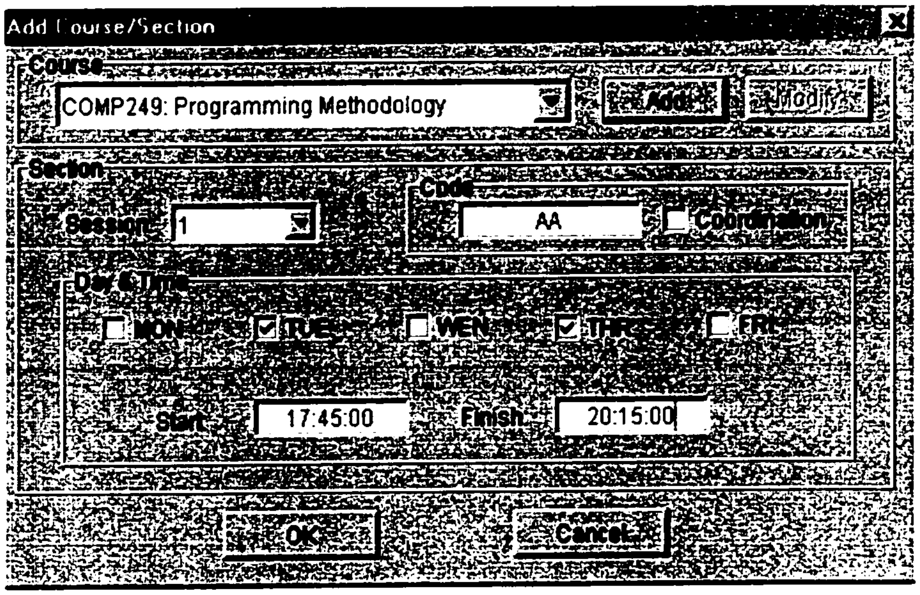


Code:
Number: 248
Credits: 3.0
Title: Introduction to Programming

Figure 6-11: Modify course information

6.3.4 Add a section

Similarly, press the '+' button above the information list, the following GUI will be provided to add a section to some course:



Course: COMP249: Programming Methodology
Section
Session: 1
Code: AA
☐ Coordination
Day & Time
☐ MON ✓ TUE ☐ WEN ✓ THR ☐ FRI
Start: 17:45:00
Finish: 20:15:00

Figure 6-12: Add a section

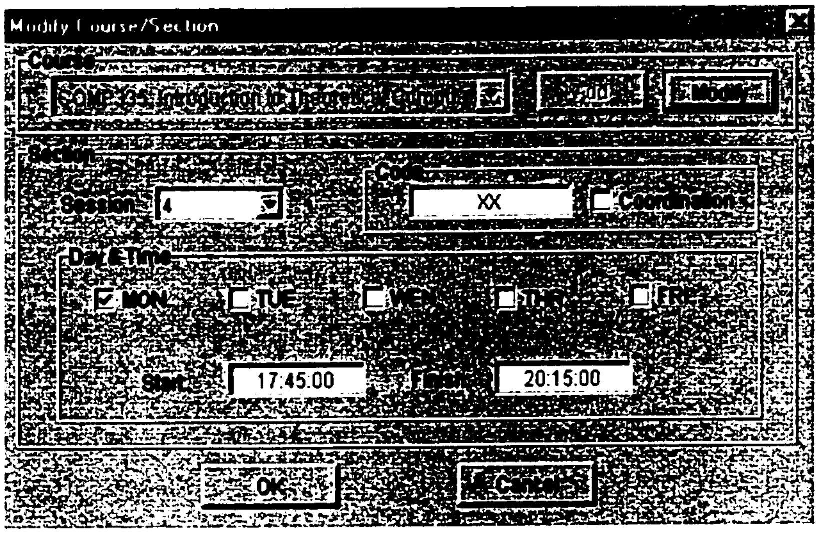
For example, if the user wants to add a section to the course COMP 249, he should select this course from the combo box in the course panel first. After that, in the section panel, the user can fill the information of the section to be added. If it is coordination, just check the 'Coordination' checkbox.

6.3.5 Remove a section

Select a section from the list, then press the '-' button, the TAP will confirm to the user whether the section should be removed. In case of that the section has been assigned the instructor, the TAP program will also remind the user that the assignment related to this section will be removed too.

6.3.6 Modify a section

The following figure is the GUI used to modify the information of a section:



Course: COMP 335 Introduction to Theoretical Computer

Session: 4

Code: XX

Coordination [ ]

Day&Time: [✓ MON] TUE WED THU FRI

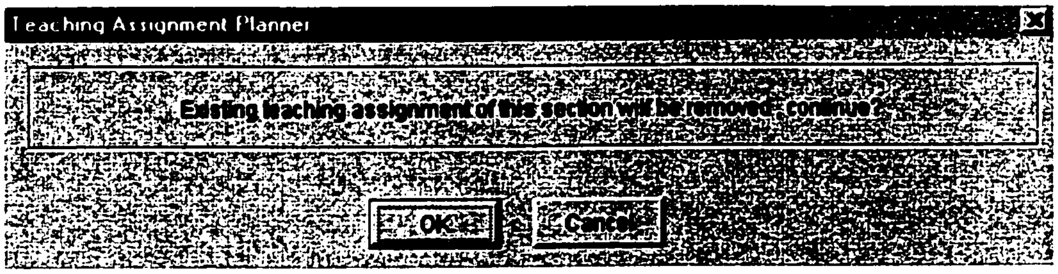
Start: 17:45:00

Finish: 20:15:00

Figure 6-13: Modify a section

Suppose the user wants to modify the section /4 XX of COMP 335, he needs to select this section from the information list and press the 'Modify ' button. Then the above GUI will be shown to the user. Working with this GUI, modification can be performed here.

We should pay attention to the case in which the section has been assigned with an instructor. Since the change of the section's information such as scheduled time will lead to conflict with the instructor's existing working load, the TAP will force the user to remove the assignment before modifying the information.


Teaching Assignment Planner

Existing teaching assignment of this section will be removed, continue?

Figure 6-14: Remove the assignment first

6.3.7 Add an instructor

Clicking the 'Instructor ' tab to switch to the instructors list. If the user wants to add an instructor, he needs to press the '+' button to see the following GUI to perform the adding.

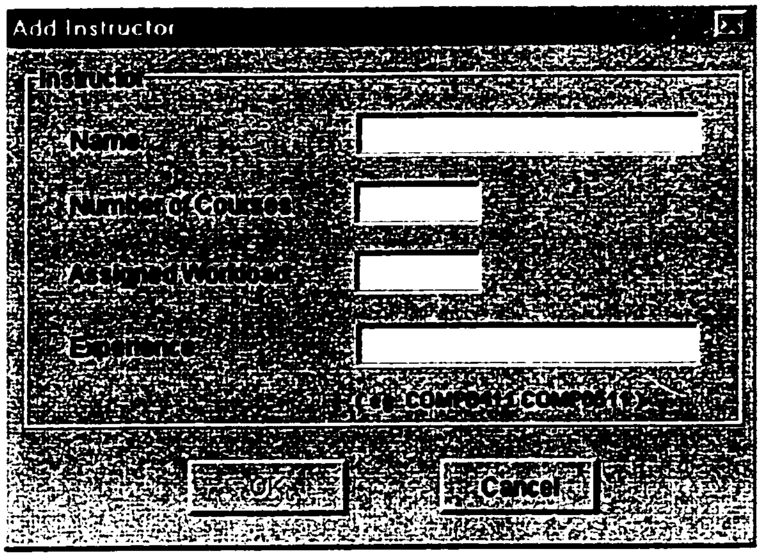
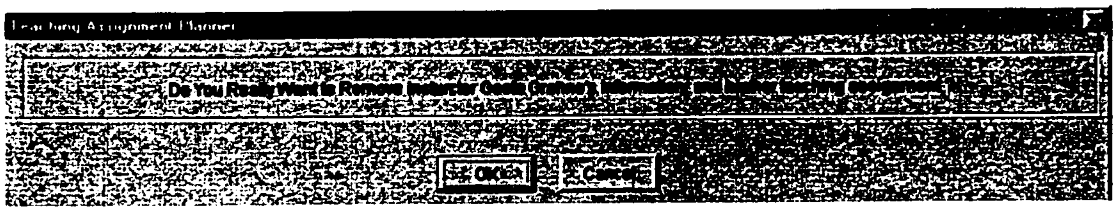


Figure 6-15: Add an instructor

6.3.8 Remove an instructor

To remove an instructor, the user has to select the instructor from the list and then press the '-' button.

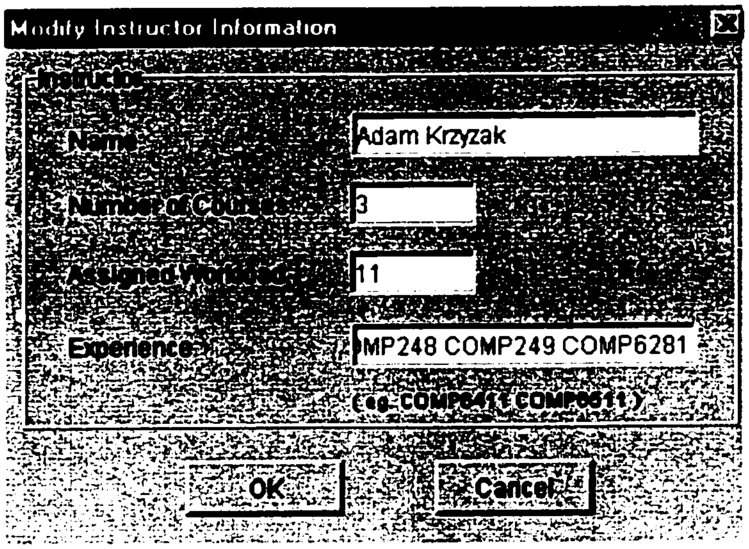


Do You Really Want to Remove Instructor [illegible]

Figure 6-16: Confirm to remove an instructor

6.3.9 Modify an instructor

In order to modify the information of an instructor, the user should select the instructor from the list and press the 'Modify' button above the list. The following GUI will be displayed to the user to modify information.



Modify Instructor Information

Instructor

Name: Adam Krzyzak

Number of Courses: 3

Assigned Workload: 11

Experience: MP248 COMP249 COMP6281

(eg. COMP6411 COMP6611)

Figure 6-17: Modify an instructor

6.4 View the Summary

The TAP program will present a summary on the right-bottom part of the split window at all times in the format of progressing bar which makes the statistics very intuitive. The summary gives the user an overall knowledge of how much of the assignment task has been finished.

The summary statistics includes the following data:

- The total number of courses
- The total number of coordinations

- The number of the coordinations which has been assigned
- The total number of sections
- The number of the sections which has been assigned
- The total number of instructors
- The number of instructors who has reach his assigned workload

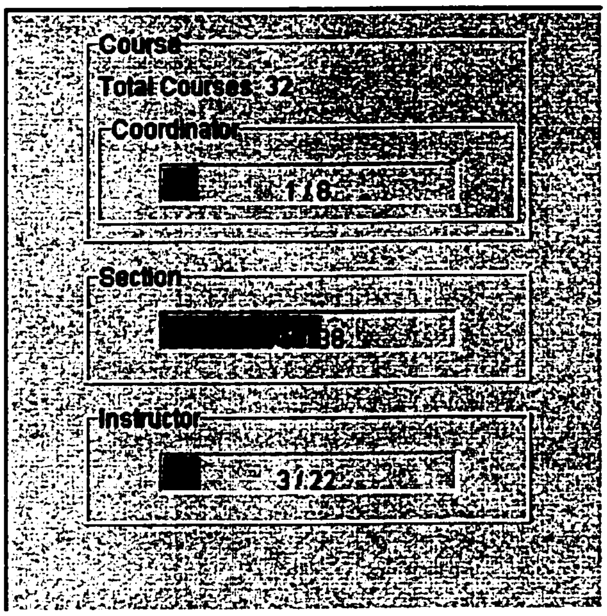


Figure 6-18: Summary

7. Conclusion

7.1 Project Summary

The *Teaching Assignment Planner (TAP)* is a Java application we developed to assist the staff doing the assignment work to fulfill the assignment task with ease. TAP is a 3-tier architecture which separates the user interface from the TA application logic part and accesses the database through the DB handler. With the help of the TAP program, the user can view the current assignment state, perform assignment and modify information. The user can also view the assignment summary at all times while the TAP program is running.

As an aid purpose program, TAP should help the user complete the assignment work without many extra work such as spending a lot of time learning how to use the program. Therefore, the usability of the program is considered to be a very important quality factor when developing this project.

The GUI of the TAP program is presented in a very concise and intuitive manner which makes it possible that the user is able to sit and use. It incorporates as few menu items and windows as possible in order to let the user feel simple and relaxed. Thus avoid the user being confused with the complicated UI manner and getting lost.

7.2 Future Work

There are some future work that have to be done in order to improve the usability and performance of the TAP program, including:

- **Generate Reports**

  TAP should generate the report for the print out purpose. It should generate a report of instructors and their workloads, sorted alphabetically by instructors' names. Besides, this TAP should also be able to generate a report of courses and sections. The report will include the coordinator assigned to each course and the instructor assigned to each section, with blanks if no assignment has been made.

- **Import Database from Student Information System (SIS)**

  Currently, MS-Access is used. In near future, when the TAP program is applied for practical use, it will be required to load all the necessary information from the Student Information System (SIS) in Concordia University. Since the JDBC technology has already been used in the TAP program, this extension work will be easily implemented because JDBC provides cross-DBMS connectivity to a wide range of SQL databases, and it also provides access to other tabular data sources, such as spreadsheets or flat files.

- **Generate an Initial Assignment State**

  The TAP program will always load the latest teaching assignment status information from the Database whenever it starts. From the very beginning, the TAP program will only provide the user a blank state without any assignment has been made if there is no teaching assignment data exists in the database. We expect the future TAP program to be somewhat intelligent so that it can generate an initial assignment state based on the basic conditions embedded in the code or the rules defined by the user. For example, based on the basic assignment constrains and the teaching/coordination experience of each instructor, TAP will automatically assign a to be assigned course/section to a certain instructor. Provided with such an initial assignment, the user is able to adjust it according to the actual need.

References and Bibliographies

[TAPR01] Peter Grogono, *Teaching Assignment Planner Requirements*, August 2001, pp.1

[DP94] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software*, Addison Wesley, August 1994, pp.4-5.

[MVC02] *MVC*, http://minnow.cc.gatech.edu/squeak/1767, May 2002, pp. 1.

[MVC98] Todd Sundsted, *How-To Java, MVC Meets Swing*, April 1998, pp. 1-2.

[JLO02] *The Java TM Language: An Overview*, http://java.sun.com/docs/overviews/java/java-overview-1.html, 2002.

[JHP99] Paul J. Deitel and Harvey M. Deitel, *Java How to Program (3rd Edition)*, Prentice Hall, August 18, 1999, pp. 557-560.

[JSC02] *Java TM FOUNDATION CLASSES (JFC)*, http://java.sun.com/products/jfc/#components, 2002.

[MAH00] Microsoft Access Help, *Databases: What they are and how they work*, 2000, pp.1-8

[JDBC02] *JDBC TM Data Access API*, http://java.sun.com/products/jdbc, 2002.

[UML99] Grady Booch, James Rumbaugh and Ivar Jacobson, *The Unified Modeling Language User Guide*, © 1999 Addison Wesley.

Appendix

The source code of the following four classes are presented:

- Instructor Model
- Instructor
- Instructor List
- DB Handler

Here we take the instructor for example to illustrate the problems. And the very similar cases are applied to the course, section and the teaching assignment.

• Class *Instructor Model*

```
package cstap;


import javax.swing.JTable;
import javax.swing.JScrollPane;
import javax.swing.JPanel;
import javax.swing.JFrame;
import java.awt.*;
import java.awt.event.*;
import javax.swing.table.*;
import java.util.Vector;
import java.sql.*;
import java.lang.Integer;

/**
 * Title:       Teaching Assignment Planner
 * Description:
 * Copyright:   Copyright (c) 2001
 * Company:     Concordia University
 * @author Hui YING
 * @version 1.0
 */



public class InstructorModel extends AbstractTableModel implements
TableSelectionInterface{

  private static final String[] columnNames = {//"Index",
                                                 "Name",
                                                 "NumberOfCourses",
                                                 "AssignedWorkload",
                                                // "Experience",
                                                 "Actual Workload"
                                                // "Expected
Workload",
                                                // "Actual
Coordinations",
                                                // "Actual Courses"
                                                 };


 private Vector vShowList = null;

 private  int tmType;
 public  final static int ALL = 0;
 public  final static int COMPLETE = 1;
 public  final static int INCOMPLETE = 2;
```

```
  public InstructorModel(InstructorList listInstructor, int iType) {
     this.tmType = iType;
     refreshList(listInstructor);
  }

  public void refreshList(InstructorList newList){

      switch(tmType) {
        case 0:                  // All instructors
          vShowList = newList;
          break;
        case 1:                  // Complete instructors

         vShowList = new Vector(newList.size());
         for (int i=0; i<newList.size(); i++) {
          Instructor tmpInstructor =
(Instructor)newList.elementAt(i);
           if (tmpInstructor.isComplete()) {
             vShowList.addElement(tmpInstructor);
           }
         }
         break;

        case 2:                  // Incomplete instructors
         vShowList = new Vector(newList.size());
         for (int i=0; i<newList.size(); i++) {
          Instructor tmpInstructor =
(Instructor)newList.elementAt(i);
           if (!tmpInstructor.isComplete()) {
             vShowList.addElement(tmpInstructor);
           }
         }
        break;

      }

      this.fireTableStructureChanged();
  }

    public int getColumnCount() {
        return columnNames.length;
    }

    public String getColumnName(int col) {
        return columnNames[col];
    }

    public int getRowCount() {
        return (vShowList != null?vShowList.size():0);
    }

    public TableRowObjectInterface getRowObject(int row){
      if (row == -1 || row >= vShowList.size()) return null;
      return ((TableRowObjectInterface)vShowList.get(row));
    }

    public Object getValueAt(int row, int col) {

      TableRowObjectInterface rowObject = getRowObject(row);
      if (rowObject != null){
        return rowObject.get(col);
      }else{
        return "";
      }
    }


}
```

• Class *Instructor*

```
package cstap;


import java.util.Vector;

/**
 * Title:        Teaching Assignment Planner
 * Description:
 * Copyright:    Copyright (c) 2001
 * Company:      Concordia University
 * @author Hui YING
 * @version 1.0
 */

public class Instructor implements TableRowObjectInterface{

 private int iDBIndex;
 private String sName =null;
 private int iNoOfCourses;
 private int iAssignedWorkload;
 private String sExperience =null;
 private int iActualWorkload = 0;
 private int iAcutalNoOfCourse = 0;
 private int iActualCoordination = 0;

 private Vector vTALinks = new Vector();

  public static final int DBINDEX          = 0;
  public static final int NAME             = 1;
  public static final int NOOFCOURSE       = 2;
  public static final int ASSIGNEDWORKLOAD = 3;
  public static final int EXPERIENCE       = 4;
  public static final int ACTUALWORKLOAD   = 5;
  public static final int EXPECTEDWORKLOAD = 6;
  public static final int ACTUALCOORDINATIONS = 7;
  public static final int ACTUALCOURSES= 8;

  public Instructor(){
    this.iDBIndex = -1;
  }

  public Instructor(int DBIndex,String Name,int NoOfCourses,int
AssignedWorkload,String Experience) {
    this.iDBIndex          = DBIndex;
    this.sName             = Name;
    this.iNoOfCourses      = NoOfCourses;
    this.iAssignedWorkload = AssignedWorkload;
    this.sExperience       = (Experience==null?"":Experience);
  }

  public void addTALink(TeachingAssignment TALink){
    vTALinks.addElement(TALink);
    if (TALink.getSection().isCoordination()){
      iActualWorkload += 1;
      iActualCoordination++;
    }else{
      iActualWorkload += 3;
      iAcutalNoOfCourse++;
    }
  }

  public void removeTALink(TeachingAssignment TALink){
    vTALinks.remove(TALink);
    if (TALink.getSection().isCoordination()){
      iActualWorkload -= 1;
      iActualCoordination--;
    }else{
      iActualWorkload -= 3;
      iAcutalNoOfCourse--;
    }
  }

  public Vector getAllTALinks(){
    return vTALinks;
  }

 /**
  * used by Instructor Model
  */
  public String get(int Col){
      String sRet = null;

      switch(Col){
        case 0: //NAME:
          sRet = new String(this.sName);
          break;
        case 1: //NOOFCOURSE
          sRet = String.valueOf(this.iNoOfCourses);
          break;
        case 2: //ASSIGNEDWORKLOAD
          sRet = String.valueOf(this.iAssignedWorkload);
          break;
        case 3:  //ACTUALWORKLOAD
          sRet = String.valueOf(this.iActualWorkload);
          break;

        default:
          sRet = "";
      }

      return sRet;
  }
  public void setDBIndex(int iDBIndex){
    this.iDBIndex = iDBIndex;
  }

  public int getDBIndex(){
    return this.iDBIndex;
  }

  //implementation of TableRowObjectInterface
  public int getCourseDBIndex(){
    return -1;
  }

  public boolean isDuplication(int iDBIndex, String sName){
    return this.iDBIndex != iDBIndex &&
this.sName.equalsIgnoreCase(sName);
  }

  public String getTASelectionTableString(int col){
    String sRet = "";
    switch(col){
      case 0:
        sRet = this.getInstructorName();
        break;
      case 1:
//        sRet = this.getExperience();
        sRet = String.valueOf(iAssignedWorkload - iActualWorkload);
        break;
    }

    return sRet;
  }

  public String getInstructorName(){
    return this.sName;
  }

  public void setInstructorName(String sName){
      this.sName = sName;
  }

  public int getObjectType(){
    return TableRowObjectInterface.INSTRUCTOR;
  }

  public boolean isAssigned(){
    return false;//isComplete();
  }

  public boolean isComplete(){
    return (iActualWorkload >= iAssignedWorkload);
  }

  public void setExperience(String sExperience){
    this.sExperience = sExperience;
  }

  public String getExperience(){
    return this.sExperience;
  }


  /**
   * invoked by instructor model to get the value of actual workload
   */
  public int getActualWorkload() {
   return this.iActualWorkload;
  }

  public void setAssignedWorkload(int iWorkload){
    this.iAssignedWorkload = iWorkload;
  }

  public int getAssignedWorkload(){
    return this.iAssignedWorkload;
  }

  public void setAssignedNoOfCourse(int iNoOfCourses){
    this.iNoOfCourses = iNoOfCourses;
  }

  public int getAssignedNoOfCourse(){
    return this.iNoOfCourses;
  }

  public int getActualNoOfCourse(){
    return this.iAcutalNoOfCourse;
  }


  public String getTAInfo(){
    String sInfo = "Expect   Workload: "+
                   String.valueOf(iAssignedWorkload -
iActualWorkload) +
                   "\n" +
                   "Experience: \n"+
                   sExperience +
                   "\n";

    return sInfo;
  }

  public String getAllInfo(){
    String sAllInfo = "Actual   Workload: " +
                   String.valueOf(this.iActualWorkload) +
                   "\n"    +
                  "Assinged Workload: " +
                   String.valueOf(this.iAssignedWorkload) +
                   "\n\n";
    String sCoordination = "Coordination: --
"+String.valueOf(this.iActualCoordination)+" Course(s)\n";
    String sSection = "Instruction: --
"+String.valueOf(this.iAcutalNoOfCourse)+" Section(s)\n";

     //add coordination/section info
    for(int i=0;i<this.vTALinks.size();i++){
        TeachingAssignment ta = (TeachingAssignment)vTALinks.get(i);
        Section section = ta.getSection();
        if (section.isCoordination()){
            sCoordination += section.getCourse().getFullCode()+": "+
                             section.getCourse().getTitle()+"\n";
        }else{
            sSection += section.getCourse().getFullCode()+": "+
                        section.getCourse().getTitle()+"\nSection:
"+
                        section.getTitle() + "     " +
                        section.getTimeInfo().trim()+"\n\n";
        }
    }

    if (this.iActualCoordination >0) sAllInfo += sCoordination +
"\n\n";
    if (this.iAcutalNoOfCourse > 0) sAllInfo += sSection ;

    return sAllInfo;
  }

}
```

• Class *Instructor List*

```
package cstap;


import java.util.Vector;
import java.sql.ResultSet;
import java.sql.SQLException;

/**
 * Title:        Teaching Assignment Planner
 * Description:
 * Copyright:    Copyright (c) 2001
 * Company:      Concordia University
 * @author Hui YING
 * @version 1.0
 */

public class InstructorList extends Vector{
  private static final String sLoadingSQL = "select * from
Instructor order by Name";

  private int iTotalInstructor =    0;
  private int iCompleteInstructor = 0;
  private DBHandler db = null;

  public InstructorList(DBHandler db) {
    this.db = db;
  }

  public void load(){
      ResultSet rs = db.executeQuery(sLoadingSQL);
      try{
          //Get all the rows.
          while (rs.next()) {

            //get columns in DB, in ResultSet the first column is 1
not from 0!
            int DBIndex          = rs.getInt(Instructor.DBINDEX+1);
            String Name          = rs.getString(Instructor.NAME+1);
            int NoOfCourses      =
rs.getInt(Instructor.NOOFCOURSE+1);
            int AssignedWorkload =
rs.getInt(Instructor.ASSIGNEDWORKLOAD+1);
            String Experience    =
rs.getString(Instructor.EXPERIENCE+1);
            //int ActualWorkload   =
0;//rs.getInt(Instructor.ACTUALWORKLOAD+1);

            Instructor newInstructor = new
Instructor(DBIndex,Name,NoOfCourses,

AssignedWorkload,Experience);//,ActualWorkload);
            this.addElement(newInstructor);
          }
      }catch (SQLException e){
          System.err.println(e.getMessage());
      }
  }

  public boolean addInstructor(Instructor newInstructor){
    boolean bRet = false;

    //insert to db
    String sInsertSQL = null;

        sInsertSQL = "INSERT INTO InstRuctor([Name],
NumberofCourses, AssignedWorkload,Experience)" +
                      " VALUES ('" +
newInstructor.getInstructorName() + "',"
                                  +
String.valueOf(newInstructor.getAssignedNoOfCourse())  + ","
                                  +
String.valueOf(newInstructor.getAssignedWorkload())  + ",'"
                                  + newInstructor.getExperience() +
"')";

//    System.err.println(sInsertSQL);

    if (db.executeUpdate(sInsertSQL)){
      //get back DBIndex
      String sSQL = "SELECT MAX(Index) from Instructor ";
      ResultSet rs = db.executeQuery(sSQL);

      if (rs == null) return false;
      try{
          rs.next();
          newInstructor.setDBIndex(rs.getInt(1));

          //add to InstructorList
          this.addElement(newInstructor);

          bRet = true;
      }catch (SQLException e){
          System.err.println(e.getMessage());
      }
    }

    return bRet;

  }

  public boolean modifyInstructor(Instructor newInstructor){
    //update DB
    String sUpdateSQL = "UPDATE Instructor SET " +
                          " [Name] = '" +
newInstructor.getInstructorName() + "'," +
                          " NumberofCourses = " +
String.valueOf(newInstructor.getAssignedNoOfCourse()) + "," +
                          " AssignedWorkload = " +
String.valueOf(newInstructor.getAssignedWorkload())  + "," +
                          " Experience = '" +
newInstructor.getExperience() + "'" +
                        " WHERE Index = " +
String.valueOf(newInstructor.getDBIndex());

//      System.err.println(sUpdateSQL);
      return db.executeUpdate(sUpdateSQL);
  }

  public void removeInstructor(Instructor instructor){
      //update DB
    String sDelSQL = "DELETE * FROM Instructor WHERE Index = " +
String.valueOf(instructor.getDBIndex());

    if (db.executeUpdate(sDelSQL)){
      this.remove(instructor);
    }
  }

  public boolean hasDuplication(int iDBIndex, String sName){
    for(int i=0;i<size();i++){
      Instructor inst = (Instructor)get(i);
      if (inst.isDuplication(iDBIndex, sName)) return true;
    }

    return false;
  }

  /***************************************************************
   *                 for initialize TA Links
   ***************************************************************/
  public Instructor getInstructorByDBIndex(int InstructorDBIndex){
    for (int i=0;i<this.size();i++){
      Instructor tmpI = (Instructor)get(i);
      if (tmpI.getDBIndex()==InstructorDBIndex){
        return tmpI;
      }
    }

    return null;
  }


/*******************************************************************
   *                  Caculate Summary Information

*******************************************************************/
  public int getAssingedInstructorNo(){
    iCompleteInstructor = 0;
    for (int i=0;i<this.size();i++){
      Instructor instructor = (Instructor)get(i);
      if (instructor.isComplete()) iCompleteInstructor++;
    }
    return this.iCompleteInstructor;
  }

  public int getTotalInstructorNo(){
    return this.size();
  }

}
```

• Class *DB Handler*

```
package cstap;


/**
 * Title:        Teaching Assignment Planner
 * Description:
 * Copyright:    Copyright (c) 2001
 * Company:      Concordia University
 * @author Hui YING
 * @version 1.0
 */
import java.lang.*;
import java.util.Vector;
import java.sql.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;
import java.util.Hashtable;


public class DBHandler {

  Connection conn;
  Statement  stmt;
  ResultSet rs;
  ResultSetMetaData metaData;

  public DBHandler() {

    String driverClass="sun.jdbc.odbc.JdbcOdbcDriver";
    String url="jdbc:odbc:TAPDB";
    String username="admin";
    String password="admin";

    try {
      Class.forName(driverClass);
      conn=DriverManager.getConnection(url,username,password);
      stmt = conn.createStatement();

     }
    catch(Exception ex)
     {
       System.err.println("Cannot get connection " +
ex.getMessage());
     }
  }


  public boolean executeUpdate(String query) {
      boolean bRet = false;

      if (conn==null || stmt==null) {
        System.err.println("There is no database to excute the
query.");
        return bRet;
      }

      try {
        int iRet =stmt.executeUpdate(query);
        if (iRet>0) bRet = true;

      }catch(SQLException ex) {
        System.err.println(ex);
      }

      return bRet;
   }

   public ResultSet executeQuery(String sQuery){
       ResultSet rs = null;

       if (conn==null || stmt==null) {
        System.err.println("There is no database to excute the
query.");
        return null;
      }

      try {
          rs = stmt.executeQuery(sQuery);
      }catch(SQLException ex) {
        System.err.println(ex);
      }

      return rs;
   }

   public void close() throws SQLException {
        System.out.println("Closing db connection");
        rs.close();
        stmt.close();
        conn.close();
    }

    protected void finalize() throws Throwable {
        close();
        super.finalize();
    }

}
```