SKED: A Course Scheduling and Advising Software

RAUL MIHALI, TAREK SOBH, DAMIR VAMOSER

Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, Connecticut 06601

Received May 2002; accepted 23 September 2003

ABSTRACT: This article presents a software application that can solve the problem of course scheduling and advising. The employed algorithms offer optimal course scheduling outputs for any given set of courses and requirements, as well as allowing for customizations by advisors or students. If the correct set of requirements and departmental courses has been set, the results of the application can be directly used for registration purposes. The proposed software is thus capable of saving hours of time for advising faculty. Once the school specific data and requirements have been set, for any specific student information, the application will search and output the schedules that will allow the student to graduate fastest and in the least number of semesters/quarters possible. Depending on the factors and data considered, the execution time varies from a few seconds to a few minutes. Currently, we have successfully tested and implemented the application at the University of Bridgeport, CT. © 2004 Wiley Periodicals, Inc. Comput Appl Eng Educ 12: 1–19, 2004; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.10054

Keywords: education; student advising; course scheduling; scheduling algorithms

INTRODUCTION

Each year at the beginning of a new academic semester, most advisors face a very common and particularly tedious and time consuming problem: deciding for each student what course schedule would be ideal for the following semester so that the student would graduate in the fastest possible time and also have his/her specific preferences and pre-requisites satisfied.

The factors that have to be considered vary from school specific requirements such as course pre-requisites, co-requisites, spring and fall offerings, to student specific ones, such as transferred credits or the subjective desire to choose or not a given course. While some advisors might be able to derive reasonable solutions in a reasonable amount of time, the process takes most of the advising time. The student will have to "trust" the advisor that the given schedule is the best choice, and in many cases the results will later on prove that the student could have actually graduated faster, or that specific school requirements have been violated or simply that the student's load and preferences could have been better balanced.

For an easier understanding of the following sections, some of the terms that are often used for discussing the details of the employed algorithm are being described. Post secondary education is usually

Correspondence to T. Sobh (sobh@bridgeport.edu).

being categorized in fields of studies defined as majors. Each major has its unique class curriculum and requirements, usually preset for years and undergoing limited infrastructure changes. Since a student can usually choose one or very few majors to study, the problem is considered at the major level.

The completion of a major usually implies that a student goes through a given number of courses, following department and inter-department requirements, spring/fall restrictions, maximal number of credits per semester as well as any particular requirements that may apply to him/her as a result of an advisor suggestion. Most of the majors would typically require around eight semesters for completion and, depending on the number of credits taken at a time, the student would be considered freshman, sophomore, junior, or senior. The courses that are to be taken are mainly directly relevant to the major, while others are general requirements for all the majors, or particular pre- or co-requisites for various relevant courses.

A pre-requisite of a course A is defined as a course that a student needs to have taken already in order to be able to take course A. A course can have none or many pre-requisites and all of them need to be satisfied.

A co-requisite of a course A is a defined as a course that a student needs to have taken in order to be able to take course A, but can also be taken in the same semester as course A.

An academic year is composed of a fall semester and a spring semester (or 3–4 quarters in a quarter based system—please note that the application can be easily adjusted to suit custom academic schedules). While usually most of the general requirement courses are offered in both semesters, major specific or particular lab-intensive courses are often offered once a year (or once every two years, in some cases).

The maximum number of credits per semester is the number of credit hours that limits the total credits that a student can take during any given semester.

The maximum number of semesters represents the maximum number of semesters in which a student should try to graduate.

As an example, as of now, the Bachelors of Computer Engineering degree at the University of Bridgeport, requires the completion of 8 semesters at an average load of 18 credits per semester and a total of 131 credits. Most of the courses are directly related to the Computer Engineering field and general requirements consist of courses of math, physics, English composition, etc. Through this study we will elaborate on this curricular example.

41

THE ALGORITHM

The goal of the algorithm is to provide the course schedules that would allow a student to graduate in the fastest possible time, from any semester that he/she might currently be in. The major specific information described in "Introduction" is used as data and guidance rules for the search process, together with student specific information [1].

The idea of an exhaustive search is not really a suitable solution. In the cases we have tested, it resulted in searching times ranging from few seconds to few days [2]. To overcome this search time problem, we have formulated and implemented a goal-seeking algorithm tailored to our specific problem (similar algorithms can be found in [3]).

Each course is assigned a requirement cost. The requirement cost of a course is defined as the longest possible chain of pre-requisites that contains the respective course. For example, if course D has as pre-requisite course C, and course C has as pre-requisite course B, and course B has as pre-requisite course A, this would make a chain of pre-requisites of requirement-cost 3 for course A. The longest chain that can be found for course A will be its associated requirement-cost. To reflect a worst-case scenario, for this cost, the co-requisites are being treated as pre-requisites.

Based on the requirement cost, the algorithm will try to schedule the courses with the highest cost first, thus minimizing the number of semesters a student needs to be in the University [4].

A course is also being associated with an availability cost. The availability cost of a course is the number of semesters that one has to go through before one would be able to take that course. The cost depends on pre-requisites, co-requisites, and spring/fall offerings. For example, a course with an availability cost of three can be taken in three semesters from now, this is a result of its co- and pre-requisites combined with the spring/fall offerings. A course with the availability cost 0 reflects a course for which all the pre- and co-requirements have been satisfied and the course is also offered in the current semester (fall or spring semester).

Having defined the above two costs, a general scheme of the algorithm can be formulated:

1) If the student has already taken (transferred) any courses, update the co- and pre-requisites, as well as the list of to be taken courses.
2) For all the to be taken courses, calculate the availability cost.
3) From all the to be taken courses with the availability cost 0, calculate the requirement cost.

4) From all the to be taken courses with the availability cost 0, pick up those that have the highest requirement cost, until the maximum number of credits per semester has not been exceeded. Let us call this a closed list of courses [5].
5) If there was a closed list of courses, then, if the lowest requirement cost in this list coincides with the highest requirement cost of the rest of the courses selected at 3, remove ("put back") all courses with this cost from the closed list.
6) From the courses with the availability cost 0 that are not in the closed list and have the highest requirement cost, form combinations [6] and keep only those that when added with the closed list credits do not exceed the maximum number of credits per semester. Let us call the results open lists, representing lists of possible semesters.
7) For each of the lists from the open lists, repeat from step 1) until all the courses have been successfully scheduled, and record for future display the schedules with the quickest completion time.

Example:

Based on the taken courses and the semester for which the algorithm is scheduling, the following courses prove to have an availability cost of zero: AD101, CPE286, CPE471, ENGL204, HUMC202, MATH214, MATH301, MATH314, ME223, SSCC201 (please see "Appendix" for a description of the courses, these being part of a set of courses which will be used throughout the study). Basically, these would be all the courses that a student could theoretically attend the following semester. The courses, sorted descending by the maximum requirement cost, have the following information (Table 1):

Table 1 Course Requirement Cost

Course	Credits	Cost
ME223	3	4
CPE286	3	3
ENGL204	1	2
MATH301	3	2
SSCC201	3	2
AD101	3	1
HUMC202	3	1
MATH314	3	1
CPE471	3	0
MATH214	3	0

Having the maximum number of credits per semester set to 18, the algorithm will pick following courses for the close list: ME223, CPE286, ENGL204, MATH301, SSCC201, AD101 (see step 4). According to step 5, the closed list will omit course AD101 and retain ME223, CPE286, ENGL204, MATH301, and SSCC201.

Based on step 6, the following three open lists will be created:

ME223, CPE286, ENGL204, MATH301, SSCC201, AD101;
ME223, CPE286, ENGL204, MATH301, SSCC201, HUMC202; and
ME223, CPE286, ENGL204, MATH301, SSCC201, MATH314.

Each of them will be recursively explored further, as step 7 indicates.

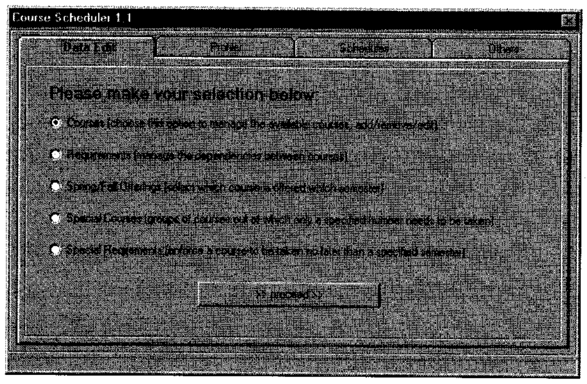
THE SOFTWARE PACKAGE

In its current stage, the software package has been developed using Microsoft Visual Basic and is composed of four distinct parts. The Data Manager part allows for managing of the necessary data and rules that mainly pertain to the major as a whole and that typically do not need to be modified for each student. The Profiler allows for the managing of student specific information that changes from student to student. The Schedules is the part where the results of the algorithm will be output, and Others is a part that contains various global settings, as well as a mini web server mode that allows application to be used over a web browser. By having this structure, various personnel can modify and work with specific information. For example, the registrar would normally use the Data Manager to add/remove/edit courses. The advisors would use the Profiler to adjust student specific information, while the students would use the Schedules to select their desired schedule of study.

The Data Manager

The Data Manager tab (Fig. 1) was designed to facilitate the input and management of all the data and rules (course requirements, school preferences, etc.) that would pertain specifically to a major, and would normally not need adjustment over short periods of time. The idea is to have this data loaded and verified one time, and then used as a shared database by the advisers of a certain major.

For more convenient handling of the information, the data management selection has been divided into five different options:



Please make your selection below:

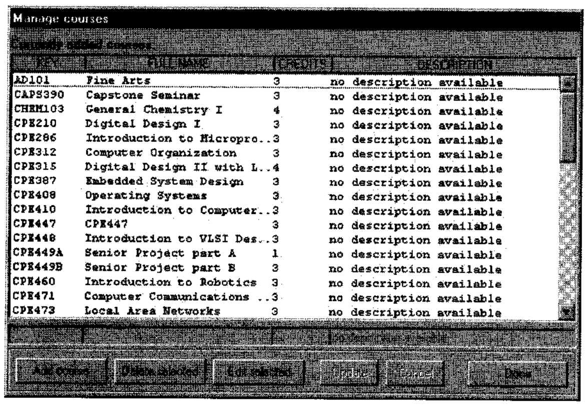
- [✓ Courses (choose this option to manage the available courses, add/remove/edit)]
- Requirements (manage the dependencies between courses)
- Spring/Fall Offerings (select which course is offered which semester)
- Special Courses (groups of courses out of which only a specified number needs to be taken)
- Special Requirements (enforce a course to be taken no later than a specified semester)

Figure 1 The Data Manager.

- Courses
- Requirements
- Spring/Fall Offerings
- Special Courses
- Special Requirements

Courses. The courses window allows for the direct input, editing, or removal of the course specific information (Fig. 2).

For a course, the software will store a unique KEY, made as a combination of letters and digits and used internally throughout the algorithm functions when referring the courses. CREDITS, represents the number of credits of a course (a numeric value that is used internally). FULL NAME and DESCRIPTION are simply informative fields and have significance solely for the user. All the necessary courses should be added here. The courses visible in Figure 2 and the following figures are part of the courses that are needed at University of Bridgeport for the Bachelor of Computer Engineering degree.



KEY	FULL NAME	CREDITS	DESCRIPTION
AD101	Fine Arts	3	no description available
CAPS390	Capstone Seminar	3	no description available
CHEM103	General Chemistry I	4	no description available
CPE210	Digital Design I	3	no description available
CPE286	Introduction to Micropro..	3	no description available
CPE312	Computer Organization	3	no description available
CPE315	Digital Design II with L..	4	no description available
CPE387	Embedded System Design	3	no description available
CPE408	Operating Systems	3	no description available
CPE410	Introduction to Computer..	3	no description available
CPE447	CPE447	3	no description available
CPE448	Introduction to VLSI Des..	3	no description available
CPE449A	Senior Project part A	1	no description available
CPE449B	Senior Project part B	3	no description available
CPE460	Introduction to Robotics	3	no description available
CPE471	Computer Communications ..	3	no description available
CPE473	Local Area Networks	3	no description available

Figure 2 The course-managing window.

Requirements. The requirements window allows for the managing of the requirements between classes (Fig. 3).

The left-most list of courses contains all the courses introduced through the Courses option. For each course, various requisite courses can be chosen. Note that, on the right list, are courses such as FRESHMAN, SOPHOMORE, JUNIOR, and SENIOR. These courses are added by default by the software and only have the role of adding better control on course requirements, in fact counting for zero credits. By checking the co-requisites check box, the requisites in the right-most list will behave as co-requisites for the selected course on the left.

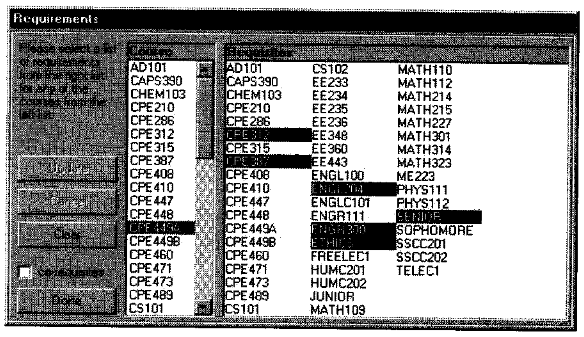
The software will check for redundant or circular reference requisites and not allow them. A redundant requisite appears when a pre-requisite of a course A has as pre-requisite that coincides with another pre-requisite of the course A.

For example, if CPE449A requires CPE387 and CPE449B requires CPE449A, it would be redundant to have CPE449B require CPE387 as well and the software will detect and notify of any such case.

A circular reference appears when a requisite for a course happens to have that course as a requisite as well, directly or indirectly. For example, if CPE449B requires CPE449A and CPE449A requires CPE387, there would be a circular reference if CPE387 would require CPE449B. The software will detect and warn accordingly of such problems.

Fall/Spring Offerings. The Fall/Spring Offerings window allows setting courses to be offered in particular semesters (Fig. 4). Courses from the left most list can be selected and added to any of the two right lists representing the fall or the spring semesters.

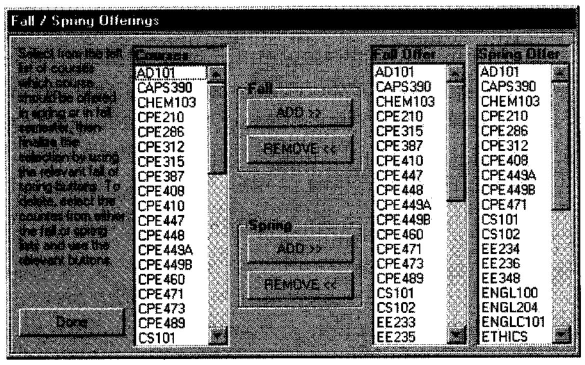
For example, the course AD101 is being offered both fall and spring semesters, while ENGL204 is only offered in the spring. The software will warn if any of the courses are not offered at all.


Requirements
Please select a list of requirements from the right list for any of the courses from the left list
Update
Cancel
Clear
Co-requisites
Done

Courses: AD101, CAPS390, CHEM103, CPE210, CPE286, CPE312, CPE315, CPE387, CPE408, CPE410, CPE447, CPE448, CPE449A, CPE449B, CPE460, CPE471, CPE473, CPE489, CS101

Requisites: AD101, CAPS390, CHEM103, CPE210, CPE286, CPE312, CPE315, CPE387, CPE408, CPE410, CPE447, CPE448, CPE449A, CPE449B, CPE460, CPE471, CPE473, CPE489, CS101, CS102, EE233, EE234, EE235, EE236, EE348, EE360, EE443, ENGL100, ENGL102, ENGLC101, ENGR111, ENGR300, ETHICS, FREELEC1, HUMC201, HUMC202, JUNIOR, MATH109, MATH110, MATH112, MATH214, MATH215, MATH227, MATH301, MATH314, MATH323, ME223, PHYS111, PHYS112, SENIOR, SOPHOMORE, SSCC201, SSCC202, TELEC1

Figure 3 Requirements.


Fall / Spring Offerings

Select from the left list of courses which course should be offered in spring or in fall semester, then finalize the selection by using the relevant fall or spring buttons. To delete, select the courses from either the fall or spring lists and use the relevant buttons.

Courses	Fall Offer	Spring Offer
AD101	AD101	AD101
CAPS390	CAPS390	CAPS390
CHEM103	CHEM103	CHEM103
CPE210	CPE210	CPE210
CPE286	CPE315	CPE286
CPE312	CPE387	CPE312
CPE315	CPE410	CPE408
CPE387	CPE447	CPE449A
CPE408	CPE448	CPE449B
CPE410	CPE449A	CPE471
CPE447	CPE449B	CS101
CPE448	CPE460	CS102
CPE449A	CPE471	EE234
CPE449B	CPE473	EE236
CPE460	CPE489	EE348
CPE471	CS101	ENGL100
CPE473	CS102	ENGL204
CPE489	EE233	ENGLC101
CS101	EE235	ETHICS

Figure 4 Fall/Spring Offerings.

Special Courses/Groups

In many cases, it could be that out of a group of various courses only a few of them need to be taken, whichever the student chooses. For example, out of CPE410, CPE460, CPE471, and CPE473, only two courses need to be taken (whichever the student prefers), as shown in Figure 5.

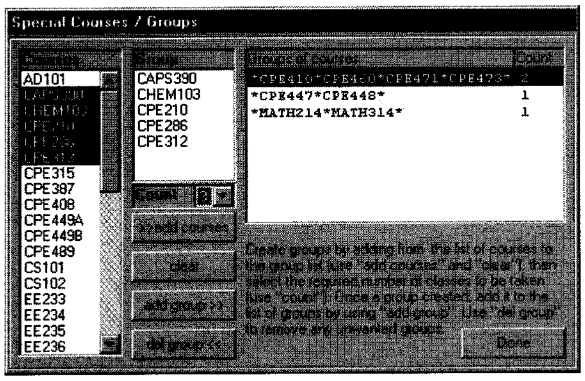
Through this window, such groups of classes can be specified. From the list of all the courses (left most), the desired courses need to be added to the middle list by using the ≫ add courses command, and once the desired number to be taken has been chosen from Count, the group can be added. Note that each course has its own requisites and from the way they are selected, this can facilitate—or not—a faster completion of the major. The algorithm takes this fact into account when searching.

Special Requirements. The Special Requirements window was added as a means of "enforcing" a student to take a certain class no later than a certain semester, as to provide better control for advising (Fig. 6).

For example, MATH227 does not have a high requirement cost, and normally the algorithm will try to place it in a later semester, first dealing with the "urgent" courses. This fact might not be appropriate when the course in case might be an easy one and should not be left for junior or senior years, despite that it does not have an explicit chain of pre-/co-requisites to fill.

Through the special Requirements window, a user can control such issues. By forcing a requirement cost of 6 for MATH227, this will oblige the algorithm to place this course at least 6 semesters before the completion of the degree, or in the case where the student has less than 6 semesters remaining to completion, to assign it immediately in the first semester if possible.

From the left-most list of courses, the user would choose a course, assign it a cost through the latest semester value and add it.


Special Courses / Groups

Courses: AD101, CAPS390, CHEM103, CPE210, CPE286, CPE312, CPE315, CPE387, CPE408, CPE449A, CPE449B, CPE489, CS101, CS102, EE233, EE234, EE235, EE236

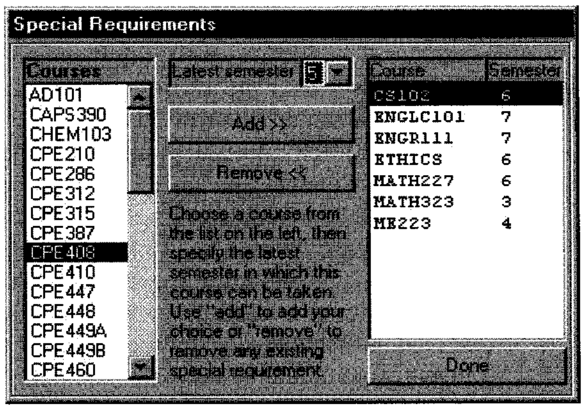
Group: CAPS390, CHEM103, CPE210, CPE286, CPE312

Count: 2

Groups of courses	Count
*CPE410*CPE460*CPE471*CPE473*	2
*CPE447*CPE448*	1
*MATH214*MATH314*	1

Create groups by adding from the list of courses to the group list (use "add courses" and "clear"), then select the required number of classes to be taken (use "count"). Once a group created, add it to the list of groups by using "add group". Use "del group" to remove any unwanted groups.

Figure 5 Special Courses/Groups.


Special Requirements

Courses: AD101, CAPS390, CHEM103, CPE210, CPE286, CPE312, CPE315, CPE387, CPE408, CPE410, CPE447, CPE448, CPE449A, CPE449B, CPE460

Latest semester: 5

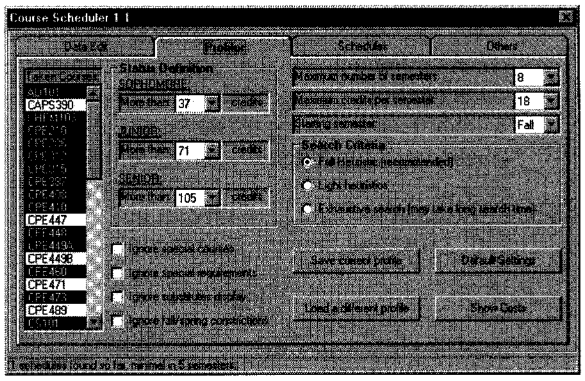
Course	Semester
CS102	6
ENGLC101	7
ENGR111	7
ETHICS	6
MATH227	6
MATH323	3
ME223	4

Choose a course from the list on the left, then specify the latest semester in which this course can be taken. Use "add" to add your choice or "remove" to remove any existing special requirement.

Figure 6 Special Requirements.

The Profiler

The profiler allows controlling the information that is specific to a student, such as the courses that he/she has taken, or that the advisor might want to adjust on a case-to-case basis to obtain better results (Fig. 7).


Course Scheduler 1.1

Taken Courses: AD101, CAPS390, CHEM103, CPE210, CPE286, CPE312, CPE315, CPE387, CPE408, CPE410, CPE447, CPE448, CPE449A, CPE449B, CPE460, CPE471, CPE473, CPE489, CS101

Status Definition
SOPHOMORE: More than 37 credits
JUNIOR: More than 71 credits
SENIOR: More than 105 credits

Maximum number of semesters: 8
Maximum credits per semester: 18
Starting semester: Fall

Search Criteria
[✓ Full Heuristic (recommended)]
Light heuristics
Exhaustive search (may take a long search time)

☐ Ignore special courses
☐ Ignore special requirements
☐ Ignore substitutes display
☐ Ignore fall/spring constrictions

1 schedules found so far, minimal in 5 semesters.

Figure 7 The Profiler.

The profiler is structured as follows:

Taken Courses list allows the advisor to select the course that the student has taken so far.

Status Definition allows the advisor to adjust the total number of credits that would define any of the sophomore, junior, or senior statuses. While normally they would be defined at once for all the students, it proves to be convenient when trying to enforce various advisor preferences. In the example from Figure 7, if the student has taken more than 37 credits and less than 71, he/she has a SOPHOMORE status. Anything less than 37 credits is considered FRESHMAN, and anything more than 105 credits is considered SENIOR.

The Maximum number of semesters, Maximum number of credits per semester, and Starting semester are being defined in the profiler as well.

The advisor also has the choice to ignore some of the restrictions, such as Special Courses, Special Requirements, or fall/spring restrictions. Such options prove to be powerful especially when advising junior/senior students who could perform independent studies instead of the normal courses or have a difficult schedule that would allow exceptions from the department.

Ignore substitutes display has just a formatting result in the output solutions. When the option is not checked, the courses that are members of a Special Group will be displayed as a group, allowing for a more compact view (see "A Complete Example: Design and Implementation History" for details).

The Search Criteria allows changing the searching algorithm being used. The Full Heuristic (recommended) option, or the default one, will use the search algorithm as described in this study, and should be the only one needed. For testing purposes, for evaluating speed difference, and performance, there is the Light Heuristic search choice and the Exhaustive search one. The Light Heuristic method is very similar to the full heuristic (default) choice, except that it does not expand the closed list to various open lists. Note that this can often result in no solutions for the problem. The Exhaustive Search algorithm, will not create a closed list based on requirement costs, instead it will apply full combinations on the list of course that are available to be taken at a certain time. Note that this option can be extremely slow and does not assure optimal solutions. Again, the search choice option has only been implemented for testing and debugging purposes. The same applies for the Show Costs option, which will display the requirement cost for all the courses.

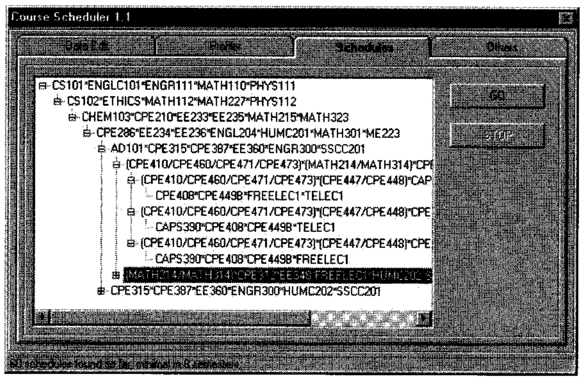
A profile can also be saved or loaded, thus easily maintaining the records of each student for future reference.

The Schedules

Once the profiler has been adjusted for a particular scenario, and preferentially saved, by switching to the Schedules tab and clicking go the software will start searching for the optimal course schedules (Fig. 8). A progress report is displayed and the process can be cancelled at any time if the solutions already found are sufficient.

The solutions found are being displayed as a tree, each leaf representing the set of courses for a semester. For example CS101*ENGLC101*ENGR111*MATH110*PHYS111 is the only optimal choice for a first semester, while the fifth semester can be either AD101*CPE315*CPE387*EE360*ENGR300*SSCC201 or CPE315*CPE387*EE360*ENGR300*HUMC202*SSCC201. Once one of the two semesters has been chosen, the semesters from its sub-tree should be considered for continuation.

The Special Groups of courses defined through the Data Manager, are normally being displayed in parentheses, to note the fact that any of the courses from the parentheses enclosed set can be chosen. For example, (CPE410/CPE460/CPE471/CPE473) would suggest to the student to choose any of the four courses and only one of them. Multiple such parentheses can occur through a semester. These are cases in which the student should choose accordingly for each of these "course menus". If the Ignore substitutes display in the Profiler is checked, these groups of



```
⊟ CS101*ENGLC101*ENGR111*MATH110*PHYS111
  ⊟ CS102*ETHICS*MATH112*MATH227*PHYS112
    ⊟ CHEM103*CPE210*EE233*EE235*MATH215*MATH323
      ⊟ CPE286*EE234*EE236*ENGL204*HUMC201*MATH301*ME223
        ⊟ AD101*CPE315*CPE387*EE360*ENGR300*SSCC201
          ⊟ (CPE410/CPE460/CPE471/CPE473)*(MATH214/MATH314)*CPI
            ⊟ (CPE410/CPE460/CPE471/CPE473)*(CPE447/CPE448)*CAP
              └ CPE408*CPE449B*FREELEC1*TELEC1
            ⊟ (CPE410/CPE460/CPE471/CPE473)*(CPE447/CPE448)*CPE
              └ CAPS390*CPE408*CPE449B*TELEC1
            ⊟ (CPE410/CPE460/CPE471/CPE473)*(CPE447/CPE448)*CPE
              └ CAPS390*CPE408*CPE449B*FREELEC1
          ⊞ (MATH214/MATH314)*CPE31[illegible]*EE349*FREELEC1*HUMC20[illegible]
        ⊞ CPE315*CPE387*EE360*ENGR300*HUMC202*SSCC201
```

60 schedules found so far, minimal in 8 semesters.

Figure 8 The Schedules.

courses will be expanded in multiple solutions with single choices. A text file with all the solutions is being output as well.

Other Features

In its current form, the application can also be run as an HTTP server, allowing access to the profiler through a web browser, and thus providing a more flexible way of sharing the database among advisors. A minimal level of security is implemented as well, allowing for IP filtering and user/password based access. Future work and directions will be detailed in "Current Limitations and Future Work."

A COMPLETE EXAMPLE: DESIGN AND IMPLEMENTATION HISTORY

By presenting a complete example on which the application was tested, this section provides a good understanding of the current design of the application, the limitations and problems that have been encountered, and how were they corrected. Four more examples are presented in the following sections, which should demonstrate a thorough evaluation of the range of problems/advising scenarios that the package is capable of tackling. Many of the settings, including requisites, pre-requisites, courses, limiters, etc., can be changed on the fly and the application could be executed again for the new values. As will be seen through the following examples, these changes are critical for variations that appear in a student's curricula, adjustments of course offerings at a University, or the conversion (transfer) of courses from one University to another.

At the University of Bridgeport, a student that has just been admitted as an undergraduate freshman in Computer Engineering, will have to complete a total of 131 credits, through a schedule of 8 semesters at an average of 18–19 credits per semester. The courses that are to be taken are highlighted in Table 2, where the "key" is the short form of the name commonly used when referring to courses.

For the given scenario, there are three groups of courses out of which only a few need to be taken: out of CPE 410, CPE471, CPE473, and CPE460, only two (any) need to be taken; out of MATH214 and MATH314, only one (any), and also only one out of CPE447 and CPE448.

Also note that MATH109 and ENGL100 can be usually replaced by placement exams, thus bringing the total to 131 credits.

The following courses (given in alphabetical order here) are being offered in the fall semester:

Table 2 List of Courses for the Computer Engineering Major

Key	Name	Credits
AD101	Fine Arts	3
CAPS390	Capstone Seminar	3
CHEM103	General Chemistry I	4
CPE210	Digital Design I	3
CPE286	Introduction to Microprocessors	3
CPE312	Computer Organization	3
CPE315	Digital Design II with Laboratory	4
CPE387	Embedded System Design	3
CPE408	Operating Systems	3
CPE410	Introduction to Computer Architecture	3
CPE447	FPGA Design	3
CPE448	Introduction to VLSI Design	3
CPE449A	Senior Project part A	1
CPE449B	Senior Project part B	3
CPE460	Introduction to Robotics	3
CPE471	Computer Comm. I: System Analysis	3
CPE473	Local Area Networks	3
CPE489	Software Engineering	3
CS101	Introduction to Computing I	3
CS102	Introduction to Computing II	3
EE233	Network Analysis I	3
EE234	Network Analysis II	2
EE235	Network Analysis I Lab	1
EE236	Network Analysis II Lab	1
EE348	Electronic Circuits I	3
EE360	Controls	3
EE443	Applied Digital Signal Processing	3
ENGL100	Basic Composition	3
ENGL204	Technical Writing for Comp. Sci. & Eng.	1
ENGLC101	Composition and Rhetoric I	3
ENGR111	Introduction to Engineering I	3
ENGR300	Economics and Management of Eng.	1
ETHICS	Integrated Studies In Comp (INSTC101)	3
FREELEC1	Free Elective One	3
HUMC201	Introduction to Humanities I	3
HUMC202	Introduction to Humanities II	3
MATH109	Precalculus Mathematics	4
MATH110	Calculus and Analytic Geometry I	4
MATH112	Calculus and Analytic Geometry II	4
MATH214	Linear Algebra	3
MATH215	Calculus and Analytic Geometry III	4
MATH227	Discrete Structures	3
MATH301	Differential Equations	3
MATH314	Numerical Methods	3
MATH323	Probability and Statistics	3
ME223	Materials Science for Engineers	3
PHYS111	Principles of Physics I	4
PHYS112	Principles of Physics II	4
SSCC201	Introduction to the Social Sciences I	3
SSCC202	Introduction to the Social Sciences II	3
TELEC1	Technical Elective 1	3

AD101, CAPS390, CHEM103, CPE210, CPE315, CPE387, CPE410, CPE447, CPE448, CPE449A, CPE449B, CPE460, CPE471, CPE473, CPE489, CS101, CS102, EE233, EE235, EE360, EE443, ENGL100, ENGLC101, ENGR111, ENGR300, FREELEC1, HUMC201, HUMC202, MATH109, MATH110, MATH112, MATH215, MATH227, MATH323, PHYS111, PHYS112, SSCC201, SSCC202, and TELEC1. And the following are offered in the spring semester: AD101, CAPS390, CHEM103, CPE210, CPE286, CPE312, CPE408, CPE449A, CPE449B, CPE471, CS101, CS102, EE234, EE236, EE348, ENGL100, ENGL204, ENGLC101, ETHICS, FREELEC1, HUMC201, HUMC202, MATH109, MATH110, MATH112, MATH214, MATH215, MATH227, MATH301, MATH314, ME223, PHYS111, PHYS112, SSCC201, SSCC202, and TELEC1.

The following courses are considered core requirements: CHEM103, CPE210, CPE286, CS101, EE233/235, ENGR111, ENGR300, MATH215, MATH301, MATH323, and ME223.

The following courses are considered program requirements: CPE312, CPE315, CPE387, CPE408, CPE447/448, CPE449, CPE489, CS102, CS227, EE234, EE348, EE360, EE443, and MATH214/314.

Note that most of the courses that are being offered both semesters are the core requirements and the courses that are usually general requirements for many majors (i.e., ENGLC101, ENGL100, and MATH109).

The final and most important constraint exists in the set of requisites and pre-requisites that exists between courses, shown in Figures 15 and 16. Figures 17–21 show respectively the Design Sequence, the Software Sequence, the Integrated Software/Hardware Design Sequence, the Hardware Sequence, and the Electrical Engineering Sequence of courses.

Based on the presented data, the university came up with a suggested course schedule shown in Figure 22. The schedule has been designed manually, by student advisors/professors. While the schedule certainly meets the constraints imposed by the Computer Engineering degree requirements, any change or customization for a particular student's needs (especially in the case of a transfer student) would be hard to implement, given a rigid 4-year proposed schedule of courses.

The first goal of the application was to find a suitable, fairly normalized and scalable data structure that could contain the given information. While a trivial Microsoft Access database seemed a sufficient start in the beginning, after few months of testing and debugging we have reached the currently presented Data Manager. It is essential to have various filters that can guarantee the integrity and quality of the data input by a user.

As a next step, we designed a brute force (combinatorial) algorithm, mainly as an immediate way of exercising the versatility of our data structure and to get an idea regarding the execution time (see "The Profiler," Search Options).

Some of the problems surfaced: unacceptable execution time (a first solution was output after more than 24 h of execution time); performing the various data manipulation routines directly on the Access table was a significant slowdown as well.

At this stage, we started to implement the suggested algorithm.

The skeleton idea was primarily derived from the sequence of requisites and pre-requisites that suggest a certain "order of importance" for courses. The data has been copied into memory and all the data manipulation routines were simplified and changed to work from the memory. For a faster implementation and result, the co-requisites have been considered pre-requisites and the groups of special courses have been ignored. The execution time was reduced significantly and the algorithm started to produce promising outputs.

However, due to the incomplete implementation and consideration of the problem, the application was not outputting completely realistic solutions. We then adjusted the algorithm to be able to work with groups of courses (note that while a student has the choice to choose which to take, each has its own list of requirements and some choices could improve the overall output). Co-requisites have also been added and handled properly. The concepts of SOPHOMORE, JUNIOR, and SENIOR have also been implemented.

The new results were more promising and closer to viable solutions, however, it became obvious that many of the courses happen to have very few or no pre-requisites and also a low requirement cost, a fact that would make the algorithm consider them primarily for the later semesters.

The following is an example of such a problem:

CHEM103, CPE210, CS101, MATH110, PHYS111
MATH314, CS102, ENGLC101, MATH112, PHYS112
CPE315, EE233, EE235, ENGR111, HUMC201, MATH215
CPE286, CPE312, EE234, EE236, ENGL204, ETHICS, MATH301
AD101, CPE387, EE360, ENGR300, HUMC202, SSCC201

CPE410, CPE408, EE348, FREELEC1, MATH227, SSCC202
CPE471, CPE448, CPE449A, CPE489, EE443, MATH323
CAPS390, CPE449B, ME223, TELEC1.

Each row represents a different semester, the first one being fall, freshman year, and then succeeding spring, fall, and so on. While the course dependency rules are met, it was not acceptable to have a course such as MATH227 in the JUNIOR year, such a course should be taken much earlier due to its relative light content and other program specific reasons. More difficult courses that have been scheduled to be taken earlier should be placed instead of MATH227. Because there were no rules that could facilitate such a choice, we introduced the concept of special requirements, through which a user can assign a certain requirement cost and so force the algorithm to schedule various courses no later than specified semesters.

In addition, the application did not specify if any of the grouped courses can be swapped or not, in other words, in a semester sequence such as CPE471, CPE448, CPE449A, CPE489, EE443, MATH323, can a student take CPE410, or CPE460 or CPE473 instead of CPE471?

Another problem was that the application was sometimes outputting hundreds of solutions all in a sequential text file, making it very hard to read and choose for a simple and optimal choice.

At this point the algorithm has been adjusted to solve the above problems, and also optimized again. We have decided to build a tree of solutions, each semester being a node level, thus converting the relatively discouraging number of solutions into a fairly simple choice, that can easily derive from the student's preference (Fig. 9).

The example through Figure 12, shows the case of a student that has taken (possibly through placement exams) MATH109 and ENGL100, starts in the fall semester, has a restriction of 18 credits per semester, and the following credit limitations: SOPHOMORE, 37; JUNIOR, 71; SENIOR, 105 (Fig. 10).

The application was run on a Pentium III 600 computer with 256 Mb of RAM, and the output was completed in 73 s (a text file containing the solutions in a serial order is also being output).

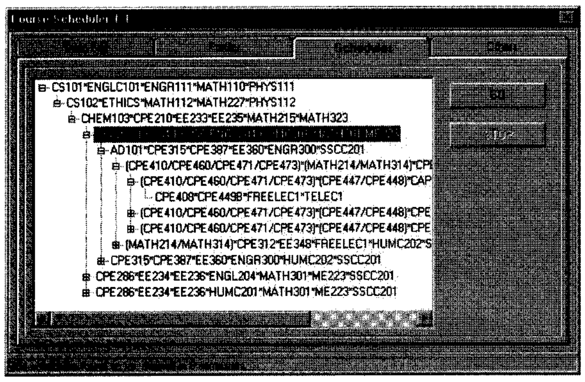
Although there are almost 300 schedules that would all be acceptable from an advisor's point of view, the student can now easily choose for each semester his/her preferred choice, and continue to expand for the next semesters of his/her choice, while still meeting the constraints of the program and still finishing in the fastest number of semesters possible. Note that there are no possibilities of graduating in less than 8 semesters, and any possibilities that would take longer are being omitted through the Maximum Number of Semesters option from the Profiler.

We have also added a way of displaying properly the groups of courses, displaying all of those that can be taken at a given moment in parenthesis. A progress bar, an option to cancel, and few other small features proved as well very useful.

In the example from Figure 13, the application found unique semester choices for Fall Freshman, Spring Freshman, and Fall Sophomore, after which there are choices.

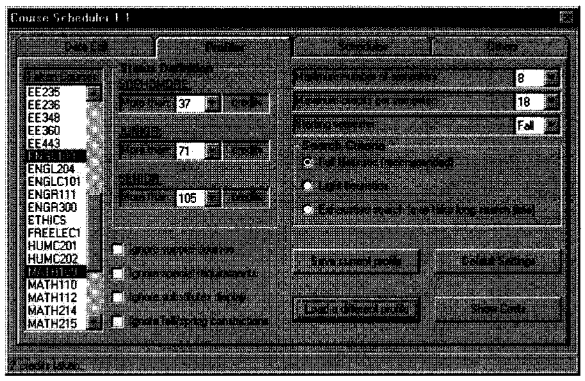
For the Spring Sophomore semester, the student can either take CPE286, EE234, EE236, ENGL204, HUMC201, MATH301, ME223 or CPE286, EE234, EE236, ENGL204, MATH301, ME223, SSCC201 or CPE286, EE234, EE236, HUMC201, MATH301, ME223, SSCC201.

Suppose the student prefers the first choice, he/she can choose for his/her Fall Junior semester AD101, CPE315, CPE387, EE360, ENGR300,



```
⊟-CS101*ENGLC101*ENGR111*MATH110*PHYS111
  ⊟-CS102*ETHICS*MATH112*MATH227*PHYS112
    ⊟-CHEM103*CPE210*EE233*EE235*MATH215*MATH323
      ⊟-
        ⊟-AD101*CPE315*CPE387*EE360*ENGR300*SSCC201
          ⊟-(CPE410/CPE460/CPE471/CPE473)*(MATH214/MATH314)*CPE
            ⊟-(CPE410/CPE460/CPE471/CPE473)*(CPE447/CPE448)*CAP
              └-CPE408*CPE449B*FREELEC1*TELEC1
            ⊞-(CPE410/CPE460/CPE471/CPE473)*(CPE447/CPE448)*CPE
            ⊞-(CPE410/CPE460/CPE471/CPE473)*(CPE447/CPE448)*CPE
          ⊞-(MATH214/MATH314)*CPE312*EE348*FREELEC1*HUMC202*S
        ⊞-CPE315*CPE387*EE360*ENGR300*HUMC202*SSCC201
      ⊞-CPE286*EE234*EE236*ENGL204*MATH301*ME223*SSCC201
      ⊞-CPE286*EE234*EE236*HUMC201*MATH301*ME223*SSCC201
```

Figure 9 Sample output schedule.



EE235
EE236
EE348
EE360
EE443
ENGL204
ENGLC101
ENGR111
ENGR300
ETHICS
FREELEC1
HUMC201
HUMC202
MATH110
MATH112
MATH214
MATH215

37
71
105
8
18
Fall

Figure 10 Profiler Scenario.

SSCC201 or CPE315, CPE387, EE360, ENGR300, HUMC202, SSCC201.

Suppose the student prefers again the first choice, his/her options for the Fall Junior semester are (CPE410/CPE460/CPE471/CPE473), (MATH214/MATH314), CPE312, EE348, HUMC202, SSCC202 or (MATH214/MATH314), CPE312, EE348, FREELEC1, HUMC202, SSCC202.

Note that the student needs to take only one of the courses from each parenthesis. He/She does not need to check whether he/she qualifies or not for any of them or whether they are offered or not, as this is being taken care of through the algorithm. Suppose the student prefers the first choice, for his/her Fall Senior year he/she can choose from (CPE410/CPE460/CPE471/CPE473), (CPE447/CPE448), CAPS390, CPE449A, CPE489, EE443 or (CPE410/CPE460/CPE471/CPE473), (CPE447/CPE448), CPE449A, CPE489, EE443, FREELEC1 or (CPE410/CPE460/CPE471/CPE473), (CPE447/CPE448), CPE449A, CPE489, EE443, TELEC1.

Taking the first choice, the only option for the last semester remains CPE408, CPE449B, FREELEC1, and TELEC1.

From this point on, an exhaustive testing sequence of possible scenarios has been circulated through the application by University advisors. Various minor problems have been fixed and we have finally decided on the exact variables and categories that a user need to manipulate. The current Data Manager, Profiler, Schedules, and Others modules have been adopted [7], with all the previously presented features. The output solutions are matching closely to the one proposed by the department, however, the application can find surprisingly more and better solutions, that show the versatility of the system, especially when dealing with transfer credits or difficult to meet student preferences.

A Second Example: A Typical Scenario

To demonstrate the advantages of the application, a second example is given, this time for a transfer student.

Student X has just been admitted at the University of Bridgeport. He/she has already attended 4 semesters at another University and based on the transfer information, the following list of courses are considered taken already: CHEM103, CPE210, CPE286, CPE315, CPE410, CPE460, CS101, CS102, EE235, EE360, ENGL100, ENGLC101, ENGR111, ENGR300, HUMC201, MATH109, MATH110, MATH112, MATH227, MATH301, MATH314, MATH323, PHYS111, PHYS112, and SSCC201.

The program requirements at the previous University were different from those of the University of Bridgeport. This makes it even more difficult in deciding which courses the student start taking. Finally, the student would like to start in the Spring and in addition, he/she would hope this time to see few alternatives, as to balance his/her time load with his/her part time job.

After the Profiler is adjusted accordingly (Fig. 11).

The application outputs a total of 69 different possibilities that could all get the student graduated in 4 semesters (Fig. 12).

After going over few of the offered choices, the student decides on a schedule that matches closely to his/her needs. He/She will start by taking AD101, CPE312, EE348, ENGL204, ETHICS, and ME223.

In the Spring, then continue with: (CPE447), CPE387, EE233, FREELEC1, HUMC202, SSCC202 in the fall (note the parenthesis for CPE447, denoting one of the group courses, but the fact that it is the only choice at the moment). Next Spring, he/she will take: CAPS390, CPE408, CPE449A, EE234, EE236, MATH215, TELEC1, and then finish next Fall with CPE449B, CPE489, and EE443.

However, things do not go exactly as planned for student X. After the first semester, he/she fails course CPE312, and returns for a new advising solution. Although he/she would have normally graduated in the next 3 semesters, this can not happen anymore due to the fact that CPE312 is a requirement course with a high cost (that is needed by many of the next courses in order to continue). The algorithm outputs 12 possible solutions in a fastest graduation time of 4 semesters (Fig. 13).

If he/she had failed AD101, EE348, or ME223, he/she would have still been able to graduate in 3 semesters. A quick look at the remaining courses reveals this. For example, a student can only take

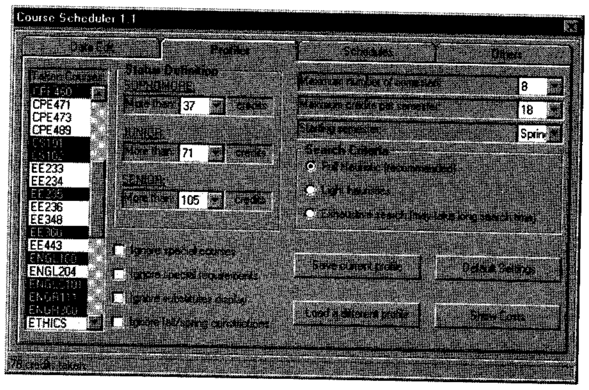
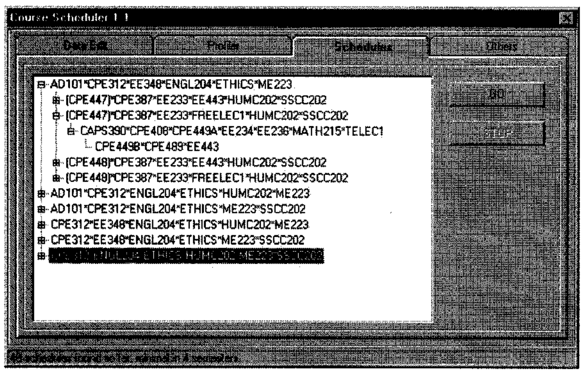


Figure 11 Secondary example profiler.



```
⊟ AD101*CPE312*EE348*ENGL204*ETHICS*ME223
   ⊞ (CPE447)*CPE387*EE233*EE443*HUMC202*SSCC202
   ⊟ (CPE447)*CPE387*EE233*FREELEC1*HUMC202*SSCC202
      ⊟ CAPS390*CPE408*CPE449A*EE234*EE236*MATH215*TELEC1
         └ CPE449B*CPE489*EE443
   ⊞ (CPE448)*CPE387*EE233*EE443*HUMC202*SSCC202
   ⊞ (CPE448)*CPE387*EE233*FREELEC1*HUMC202*SSCC202
⊞ AD101*CPE312*ENGL204*ETHICS*HUMC202*ME223
⊞ AD101*CPE312*ENGL204*ETHICS*ME223*SSCC202
⊞ CPE312*EE348*ENGL204*ETHICS*HUMC202*ME223
⊞ CPE312*EE348*ENGL204*ETHICS*ME223*SSCC202
⊞ [illegible]
[illegible]
```

Figure 12 Second example scheduling (A).

CPE449B after he/she took CPE449A. CPE449A has as requirements CPE312, ETHICS, and ENGL204, and, e.g., CPE312 is not being offered in fall, which brings up the required number of semesters to an obvious minimum of 4.

Given the situation, the student chooses for the fall: CPE447, CPE387, EE233, EE443, HUMC202, SSCC202, then he/she plans CAPS390, CPE312, EE234, FREELEC1, MATH215, and TELEC1 for the spring, CPE449A and CPE489 for the next fall, and finally CPE449B, CPE408, and EE236.

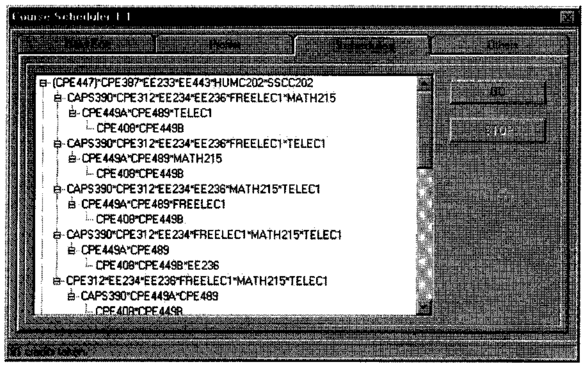
Although in the next semester the student fails course EE443, a new rescheduling shows that he/she can still graduate in three more semesters (Fig. 14).

The student will continue with his/her previously selected schedule, but now will add EE443 to his/her first senior semester.

A Third Example: An Unusual Scenario A

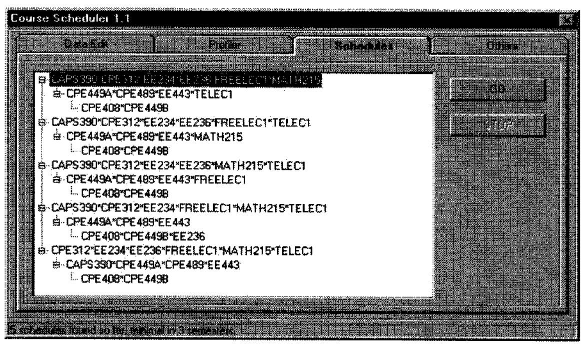
In the next three examples, the emphasis will be on exposing the usefulness of the application in rather unusual or tedious (time consuming) situations.

Student X arrives as a transfer student at the University of Bridgeport, and the school from where



```
⊟ (CPE447)*CPE387*EE233*EE443*HUMC202*SSCC202
   ⊟ CAPS390*CPE312*EE234*EE236*FREELEC1*MATH215
      ⊟ CPE449A*CPE489*TELEC1
         └ CPE408*CPE449B
   ⊟ CAPS390*CPE312*EE234*EE236*FREELEC1*TELEC1
      ⊟ CPE449A*CPE489*MATH215
         └ CPE408*CPE449B
   ⊟ CAPS390*CPE312*EE234*EE236*MATH215*TELEC1
      ⊟ CPE449A*CPE489*FREELEC1
         └ CPE408*CPE449B
   ⊟ CAPS390*CPE312*EE234*FREELEC1*MATH215*TELEC1
      ⊟ CPE449A*CPE489
         └ CPE408*CPE449B*EE236
   ⊟ CPE312*EE234*EE236*FREELEC1*MATH215*TELEC1
      ⊟ CAPS390*CPE449A*CPE489
         └ CPE408*CPE449B
[illegible]
```

Figure 13 Second example scheduling (B).



```
⊟ [illegible]
   ⊟ CPE449A*CPE489*EE443*TELEC1
      └ CPE408*CPE449B
⊟ CAPS390*CPE312*EE234*EE236*FREELEC1*TELEC1
   ⊟ CPE449A*CPE489*EE443*MATH215
      └ CPE408*CPE449B
⊟ CAPS390*CPE312*EE234*EE236*MATH215*TELEC1
   ⊟ CPE449A*CPE489*EE443*FREELEC1
      └ CPE408*CPE449B
⊟ CAPS390*CPE312*EE234*FREELEC1*MATH215*TELEC1
   ⊟ CPE449A*CPE489*EE443
      └ CPE408*CPE449B*EE236
⊟ CPE312*EE234*EE236*FREELEC1*MATH215*TELEC1
   ⊟ CAPS390*CPE449A*CPE489*EE443
      └ CPE408*CPE449B
5 schedules found so far, minimal in 3 semesters
```

Figure 14 Second example scheduling (C).

he transferred simply had no course dependency requirements in their scheduling, allowing for the student's arbitrary schedule at their own risk. After evaluating his/her transcript, this reveals that he has taken the following courses so far: CAPS390, CHEM103, CPE210, CPE286, CPE312, CPE315, CPE387, CPE408, CPE410, CPE447, CPE448, CPE449A, CPE449B, CPE460, CPE471, CPE473, CPE489, CS101, CS102, EE233, EE234, EE235, EE236, EE348, EE360, EE443, ENGL100, ENGL204, ENGLC101, ENGR111, ENGR300, ETHICS, FREELEC1, HUMC201, HUMC202, MATH109, MATH110, MATH112, MATH214, MATH215, MATH227, MATH301, and MATH314.

Overall, 124 credits have been taken. While few courses are left to be taken, which could suggest a limited amount of choices and therefore little time spent in solving them, the work becomes tedious when mapping the course dependency imposed at the University of Bridgeport. Making a scheduling mistake becomes a highly probable case, which could result in affecting the quality of the curricula at the University. By using the scheduling tool, this turns out to be a trivial task and the system will reveal in seconds a correct solution such as the following:

- first semester: AD101, ME223, PHYS111, SSCC201, TELEC1
- second (last) semester: MATH323, PHYS112, SSCC202

It is also important to consider that the student can ask and be provided with concrete answers regarding why a particular schedule is preferable to others.

A Fourth Example: An Unusual Scenario B

This time, let us consider the case of transfer students that have taken very few credits and again, not necessarily adhering to course dependencies. Student

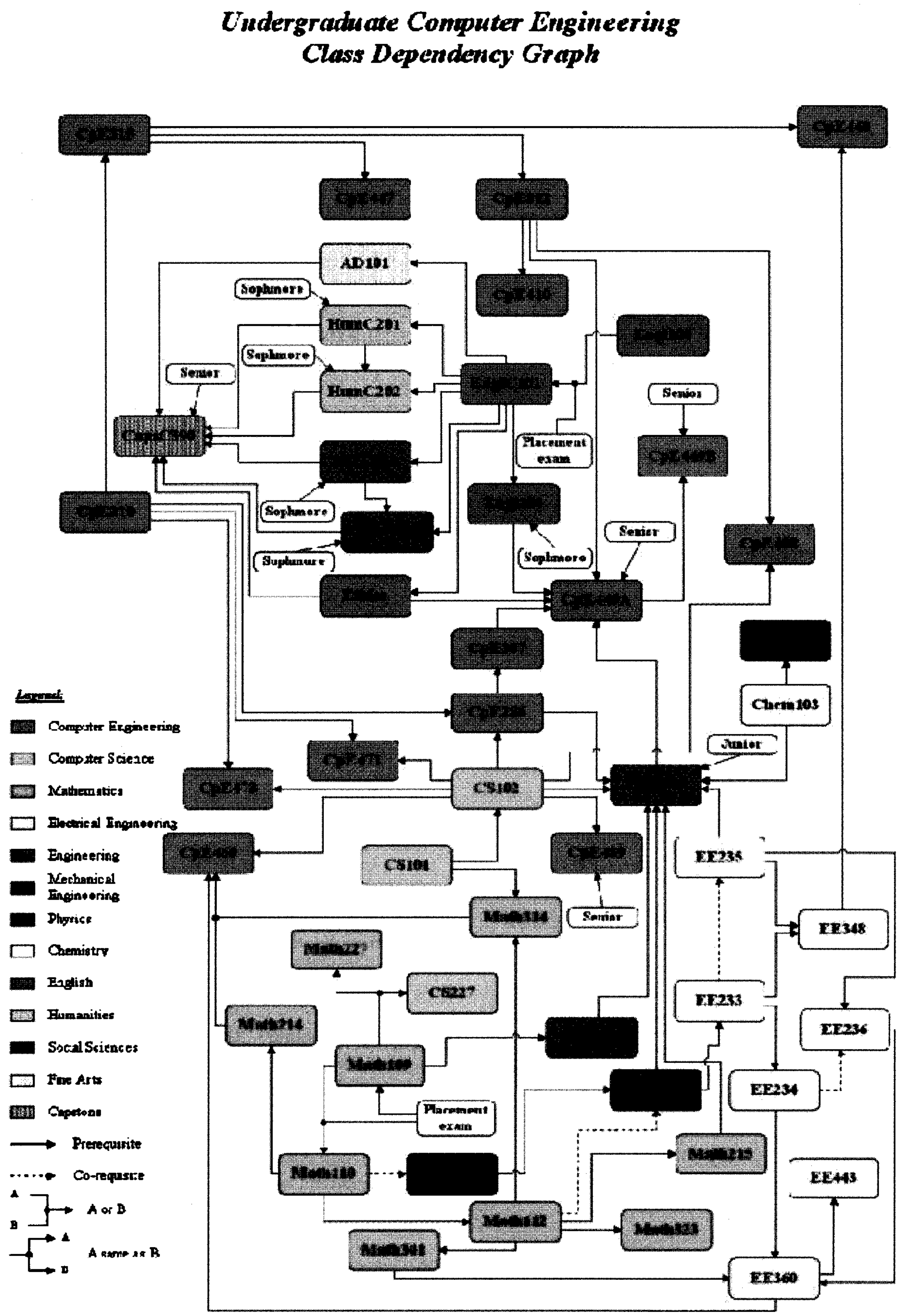


Figure 15 Course Dependency Graph categorized by types of courses. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

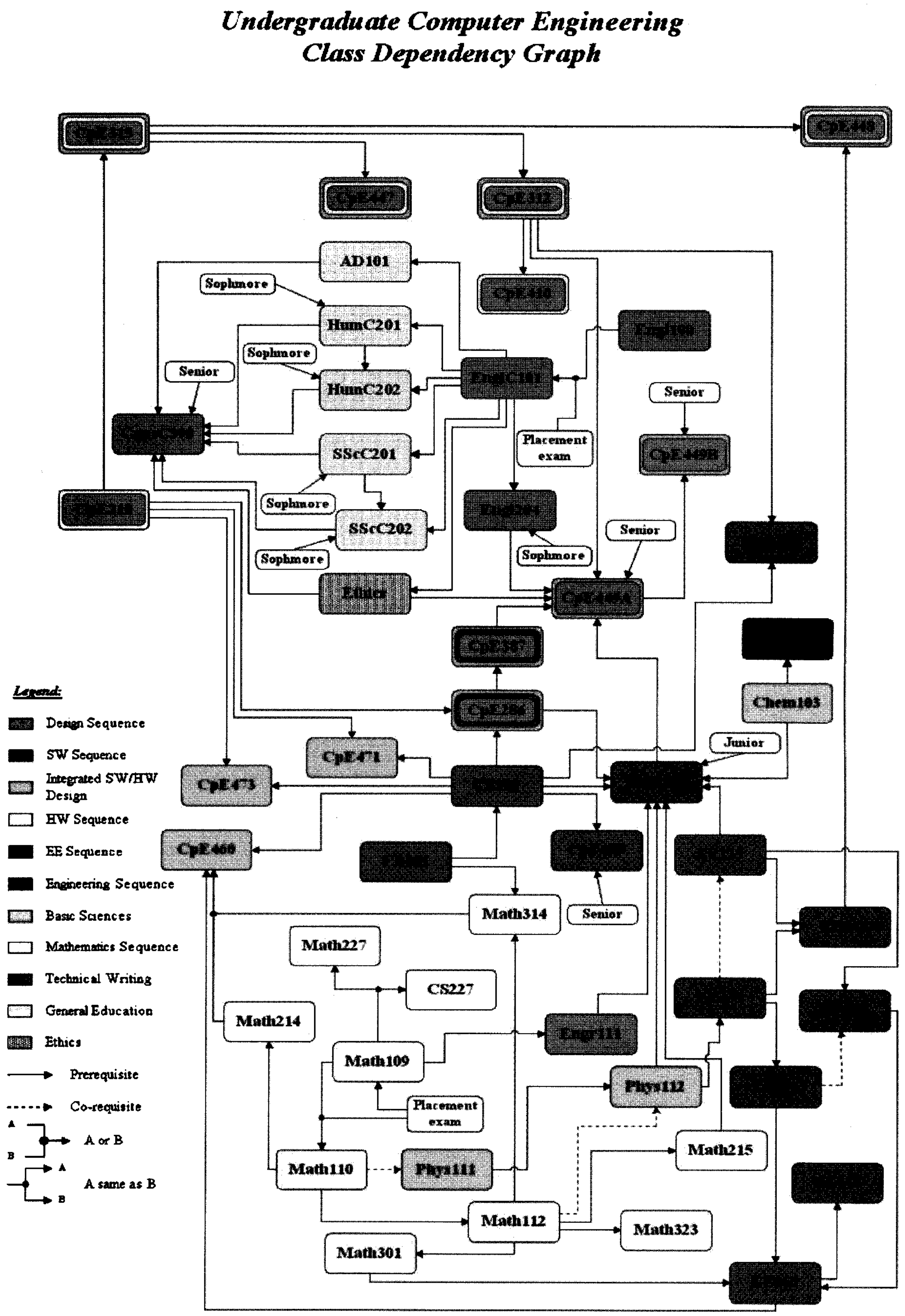


Figure 16 Course Dependency Graph categorized by course sequences. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

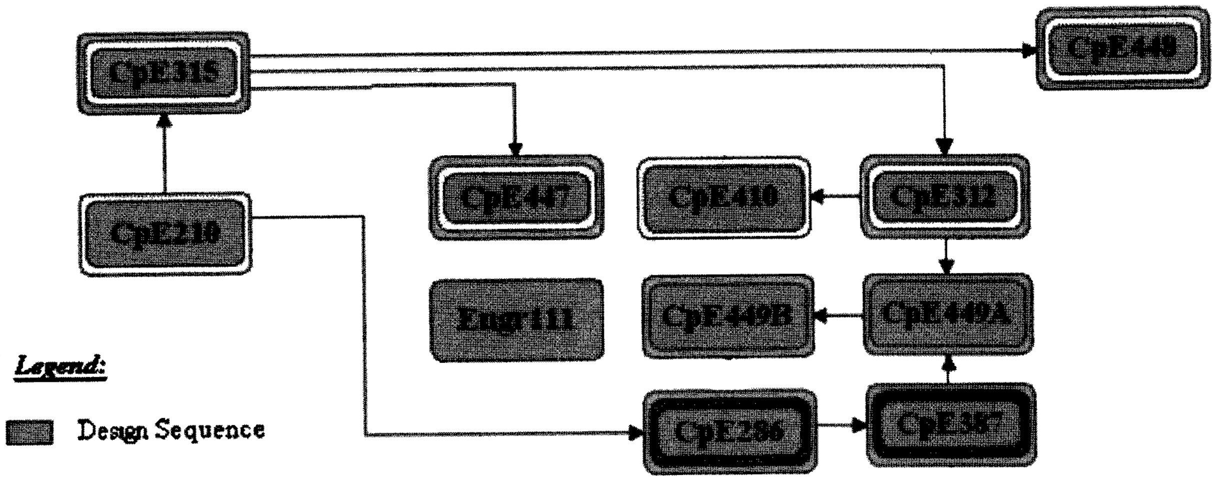


Figure 17 Design Sequence.[Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

X has taken 16 credits so far as follows: CAPS390, CPE387, CPE408, ENGL101, and MATH109.

While trying to setup a course schedule, the advisor will notice a hard time in finding a schedule in less than 8 semesters, which indeed is not possible in this case. The amount of work necessary to reach this conclusion can be frustrating and still leave doubts among the student and the advisor, also confirming the utility of an optimizing software package that could double check these results. Trying this scenario through this software package will inspire confidence and provide a high choice of possibilities.

A Fifth Example: An Unusual Scenario C

Finally, a last and unusual scenario further demonstrates the critical advantage of this software package.

Student X is transferring from a University that adopts a course dependency scheme fairly close to the one of the University of Bridgeport, but is also more flexible regarding the allowable number of credits per semester. As a result, student X has taken a total of 24 credits in one semester, as follows: CS101, CPE210, ENGL100, ENGR111, MATH109, MATH110, and PHYS111.

Due to personal reasons, the student can only attend the University of Bridgeport if he can be accommodated to graduate in 6 semesters (for example, in the case of an international student that comes to study to the United States and has a time-limited visa).

Trying to come up with a 6-semester solution in regular conditions will not be easy. The task of justifying and determining such a schedule will take time. The idea then would be to try and reduce some of the restrictions, e.g., allow more credits per semester? Each case will require a thorough attention from an advisor and can be hard to implement. The goal is

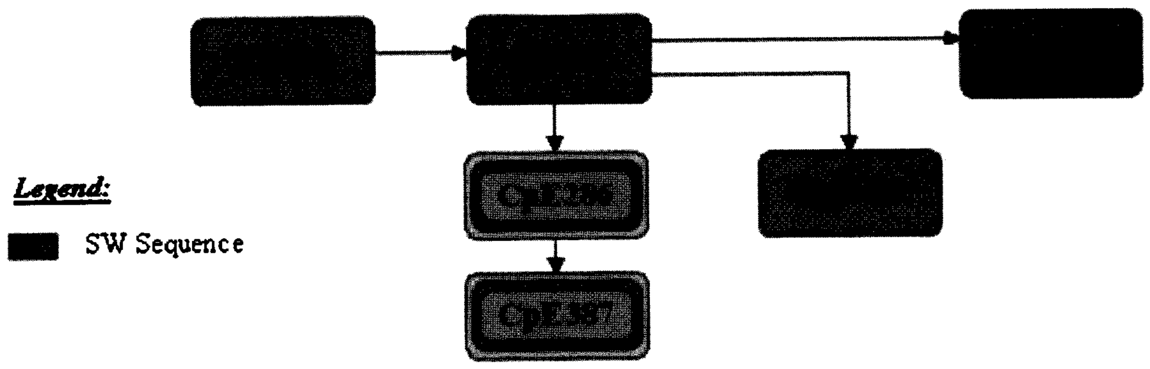


Figure 18 Software Sequence. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

41

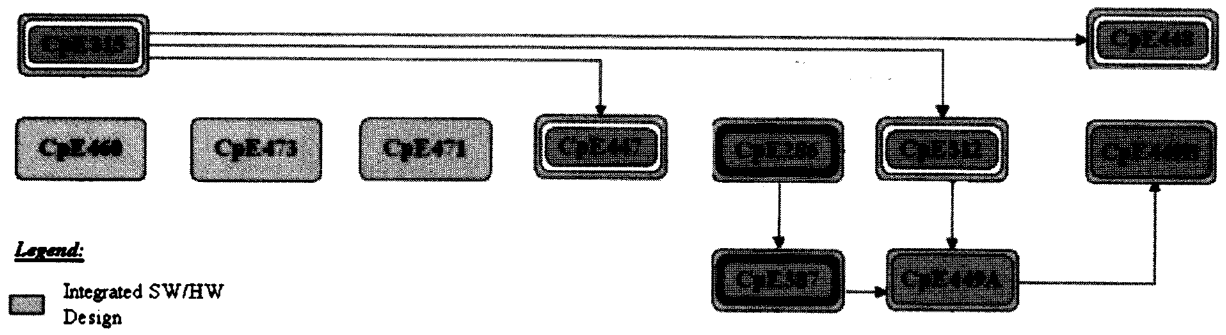


Figure 19 Integrated SW/HW Design Sequence. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

to just find a solution, and if a tool could allow "playing" with a few parameters and coming up with an answer, the task would be simpler.

By inputting the scenario in the software package, it appears that the student can actually graduate in 6 semesters if allowed to take up to 22 credits per semester and would start in the Spring. Listed below is one of the 8 such possible solutions:

- first semester: CHEM103, CS102, ENGLC101, MATH112, MATH227, PHYS112
- second semester: CPE315, EE233, EE235, HUMC201, MATH215, MATH323, SSCC201
- third semester: (MATH314), CPE286, CPE312, EE234, EE236, ENGL204, ETHICS, MATH301, ME223
- fourth semester: (CPE410/CPE460), (CPE447), AD101, CPE387, EE360, ENGR300, HUMC202, SSCC202
- fifth semester: (CPE410/CPE460/CPE471), CAPS390, CPE408, CPE449A, EE348, FREELEC1, TELEC1
- sixth semester: CPE449B, CPE489, EE443

CURRENT LIMITATIONS AND FUTURE WORK

Although in its current stage the application has enough features to be conveniently used for advising, there are still features that need enhancements. Students should be able to filter the list of final solutions for their own preference. The availability cost of a course should also take into consideration the total number of students a course can be offered to, and to co-relate this fact with a global database that keeps track whether a course is still available from this point of view. Some courses could also happen to be offered in the same exact time, this being a case that the algorithm should consider as well.

Besides the relatively immediate changes above, the software will be probably converted to a completely web based interface in the future, which would link and maintain a school database, with all the courses for all the majors and all the information for students stored there as well.

In its current form, the application does not address or incorporate security measures regarding

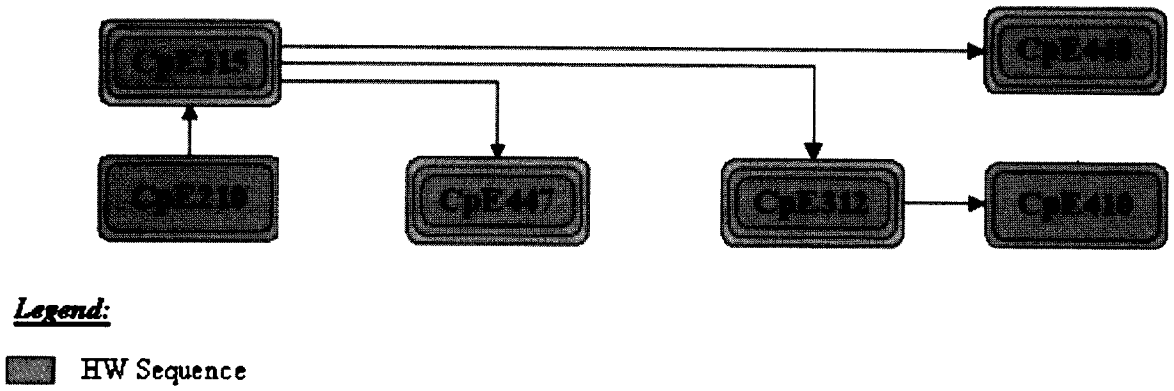


Figure 20 HW Sequence. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

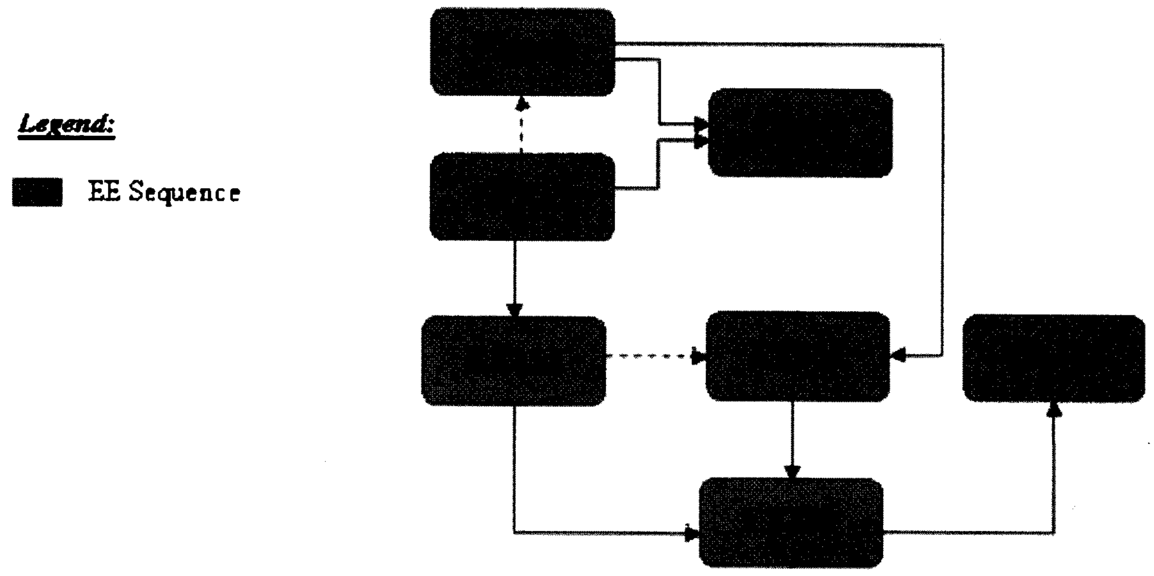


Figure 21 EE Sequence. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

the authorization of any results or changes that are to be instated in a student's advising scenario. This is due to the fact that the tool has been designed to "reside" on the desktop of an advisor and not to be networked. Students may be provided with the same tool for them to experiment with parameters on their own for generating various schedule possibilities. However, if they want their generated schedules to be considered by advisors, they would have to give the advisors the saved output of their solutions, and consequently have the advisors test them through their version of the tool and settings before officially being approved. The security, privacy, and authorization aspects would need thorough consideration in a networked setup. In any networked scenario, the framework would be similar to the current one, as an advisor will ultimately verify a student's preferred choices prior to their approval.

The ultimate goal is to have a student type an ID number (and password) from his/her home computer and no other additional information. Based on the ID, the software will find the major the student belongs too and what courses he/she took already. Consequently, the system will output immediately the optimal possibilities and allow the student to choose among them and register online. Once the student confirms the selection, this reply gets appended to his/her advisor's profile for double checking (to avoid any possible application error or invalid results), and if the advisor agrees with it, the student will be notified by email and the registration process is complete.

41

CONCLUSIONS

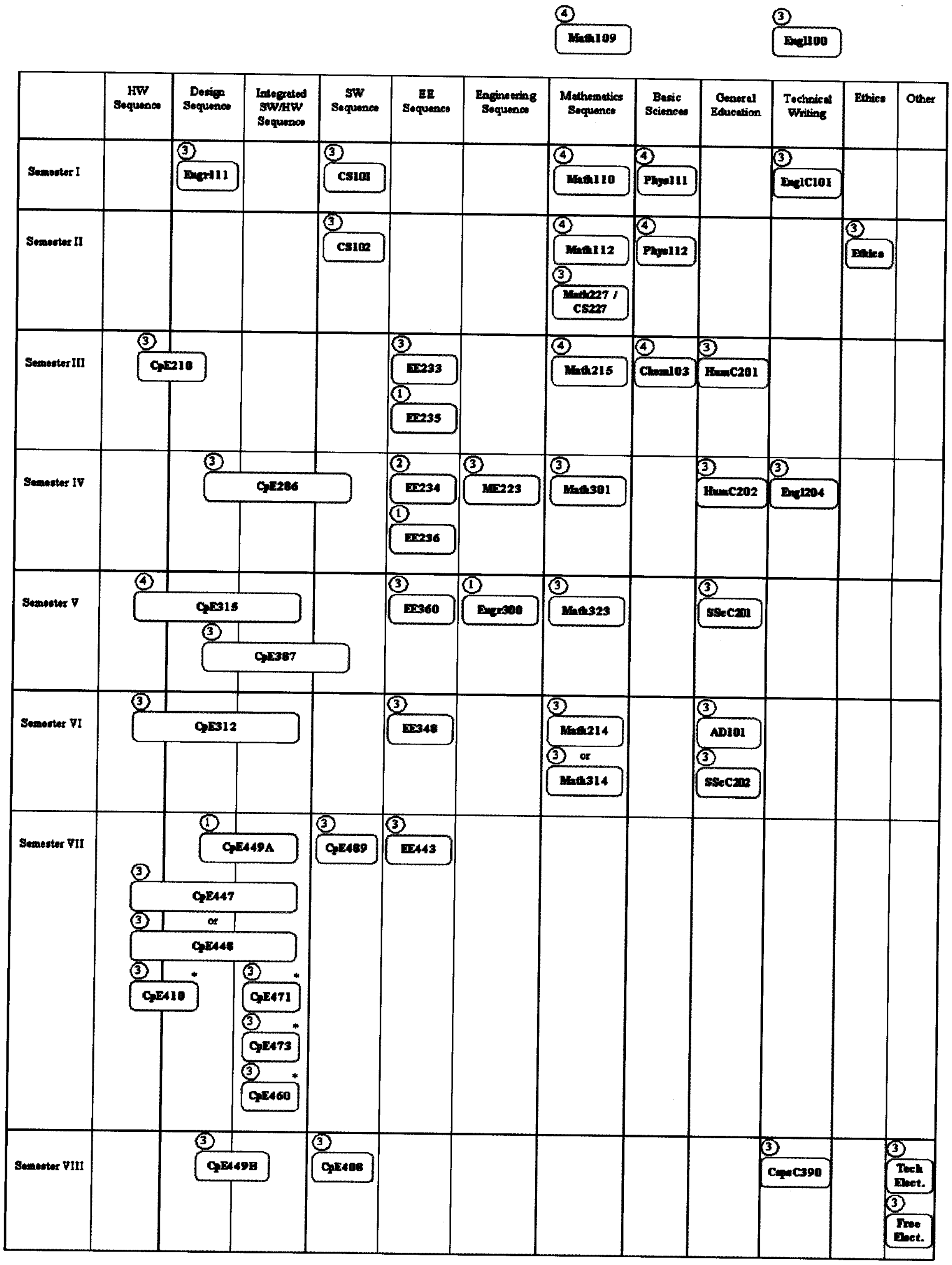
In this study, we present a software model designed to aid students and advisors in the tedious and time-consuming registration tasks that students and advisors have to go through every semester.

The designed application can virtually eliminate the time an advisor would need to spend with a student on registration, optimizing for the quickest graduation schedule, and facilitating the student's preferences, thus allowing time for more specific and important student related issues.

While the algorithms have been designed and developed in consultation, and as a result of discussions with many highly experienced advising faculty members in various Universities throughout the country and the world, the authors do understand that there may exist highly specific and unique scenarios that could not be directly addressed in the proposed model. However, in such cases, the tool can be reconfigured to help incorporate and solve many of these unique problems and reduce the invested advising time considerably.

Currently, we have successfully tested and used the application in the Bachelor of Computer Engineering Major at the University of Bridgeport. The tool provided excellent results. We are also in the process of completely revamping the application to the specifications detailed in "Current Limitations and Future Work," which would result in a completely automated advising and registration system complying with the requirements of a program of study.

Undergraduate Computer Engineering



	HW Sequence	Design Sequence	Integrated SW/HW Sequence	SW Sequence	EE Sequence	Engineering Sequence	Mathematics Sequence	Basic Sciences	General Education	Technical Writing	Ethics	Other
							④ Math109			③ Engl100		
Semester I		③ Engr111		③ CS101			④ Math110	④ Phys111		③ EnglC101		
Semester II				③ CS102			④ Math112 ③ Math227 / CS227	④ Phys112			③ Ethics	
Semester III	③ CpE210				③ EE233 ① EE235		④ Math215	④ Chem103	③ HumC201			
Semester IV		③ CpE286			③ EE234 ① EE236	③ ME223	③ Math301		③ HumC202	③ Engl204		
Semester V	④ CpE315	③ CpE387			③ EE360	① Engr300	③ Math323		③ SScC201			
Semester VI	③ CpE312				③ EE348		③ Math214 or ③ Math314		③ AD101 ③ SScC202			
Semester VII	③ CpE447 or ③ CpE448 ③ CpE410 *	① CpE449A	③ CpE471 * ③ CpE473 * ③ CpE460 *	③ CpE489	③ EE443							
Semester VIII		③ CpE449B		③ CpE408						③ CapsC390		③ Tech Elect. ③ Free Elect.

* Choose 2 out of these 4

Figure 22 A suggested Schedule. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Appendix

AD101, Fine Arts, (3 credits);
CAPS390, Capstone Seminar, (3 credits);
CHEM103, General Chemistry I, (4 credits);
CPE210, Digital Design I, (3 credits);
CPE286, Introduction to Microprocessors, (3 credits);
CPE312, Computer Organization, (3 credits);
CPE315, Digital Design II with Laboratory, (4 credits);
CPE387, Embedded System Design, (3 credits);
CPE408, Operating Systems, (3 credits);
CPE410, Introduction to Computer Architecture, (3 credits);
CPE447, FPGA Design, (3 credits);
CPE448, Introduction to VLSI Design, (3 credits);
CPE449A, Senior Project part A, (1 credits);
CPE449B, Senior Project part B, (3 credits);
CPE460, Introduction to Robotics, (3 credits);
CPE471, Computer Communications I: System Analysis, (3 credits);
CPE473, Local Area Networks, (3 credits);
CPE489, Software Engineering, (3 credits);
CS101, Introduction to Computing I, (3 credits);
CS102, Introduction to Computing II, (3 credits);
EE233, Network Analysis I, (3 credits);
EE234, Network Analysis II, (2 credits);
EE235, Network Analysis I Lab, (1 credit);
EE236, Network Analysis II Lab, (1 credit);
EE348, Electronic Circuits I, (3 credits);
EE360, Controls, (3 credits);
EE443, Applied Digital Signal Processing, (3 credits);
ENGL100, Basic Composition, (3 credits);
ENGL204, Technical Writing for Computer Science & Engineering, (1 credit);
ENGLC101, Composition and Rhetoric I, (3 credits);
ENGR111, Introduction to Engineering I, (3 credits);
ENGR300, Economics and Management of Engineering Projects, (1 credit);
ETHICS, Integrated Studies in Computing (INTSC101), (3 credits);
FREELEC1, Free Elective 1, (3 credits);
HUMC201, Introduction to Humanities I, (3 credits);
HUMC202, Introduction to Humanities II, (3 credits);
MATH109, Precalculus Mathematics, (4 credits);
MATH110, Calculus and Analytic Geometry I, (4 credits);
MATH112, Calculus and Analytic Geometry II, (4 credits);
MATH214, Linear Algebra, (3 credits);
MATH215, Calculus and Analytic Geometry III, (4 credits);
MATH227, Discrete Structures, (3 credits);
MATH301, Differential Equations, (3 credits);
MATH314, Numerical Methods, (3 credits);
MATH323, Probability and Statistics, (3 credits);
ME223, Materials Science for Engineers, (3 credits);
PHYS111, Principles of Physics I, (4 credits);
PHYS112, Principles of Physics II, (4 credits);
SSCC201, Introduction to the Social Sciences I, (3 credits);
SSCC202, Introduction to the Social Sciences II, (3 credits); and
TELEC1, Technical Elective 1, (3 credits).

REFERENCES

[1] E. H. L. Aarts and J. K. Lenstra, Local search in combinatorial optimization, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York, 1997.

[2] D. H. Greene and D. E. Knuth, Mathematics for the analysis of algorithms, 3rd edn., Birkhauser, 1990.

[3] P. Dasgupta, P. P. Chakrabarti, and S. C. Desarkar, Multiobjective heuristic search: An introduction to intelligent search methods for multicriteria optimization, Kaufmann Publishers, 1999.

[4] Patrascoiu, Octavian, Marian, Gheorghe, Mitroi, Nicolae, Elements of graphs and combinatorial theory, methods, algorithms and programs, B.I.C. All, Romania, 1994.

[5] P. N. Izvercian, V. Cretu, M. Izvercian, R. Resiga, Introduction in graph theory, the critical path method, Editura de Vest, Romania, 1993.

[6] R. L. Graham, D. E. Knuth, and O. Patashnik, X. Oren, Concrete mathematics, a foundation for computer science, Addison-Wesley, Reading, MA, 1994.

[7] M. Markotty, Software implementation (practical software engineering, Vol. 4), Prentice-Hall ECS Professional, Englewood Cliffs, NJ, 1991.

41

BIOGRAPHIES

Raul Mihali received his Bachelor's of Science and Master's of Science degrees in computer science, with honors, from the Department of Computer Science and Engineering at the University of Bridgeport, Connecticut, in 1999 and 2000, respectively. He is working towards his PhD degree in computer engineering with a dissertation on "Self Manufacturing Fully Autonomous Mobile Manipulators," and continues to work closely with Dr. Tarek Sobh in the interdisciplinary Robotics, Intelligent Sensing, and Control (RISC) Labs at the University of Bridgeport. His current research interests include active sensing under uncertainty, robots and electromechanical systems prototyping, mobile autonomous manipulators, stealth and security oriented robots, polymorphic structures and robot actuators, assembler time optimizations, actuators and sensors for unusual execution tasks, remote automation and manufacturing, digital design, and VLSI. Raul Mihali has published over 40 journal articles, conference papers and book chapters in the areas of automation, vision and sensing, prototyping, optimization, algorithms, and programming techniques and is continuously involved in related research and development efforts.

Tarek M. Sobh received the PhD and MS degrees in computer and information science from the School of Engineering, University of Pennsylvania, in 1991 and 1989, respectively, and the BSc in engineering degree with honors in computer science and automatic control from the Faculty of Engineering, Alexandria University, in 1988. He is currently the dean of the School of Engineering at the University of Bridgeport, Connecticut; the founding director of the Interdisciplinary Robotics, Intelligent Sensing, and Control (RISC) laboratory and a professor of computer science, computer engineering, mechanical engineering, and electrical Engineering. He was the interim chairman of computer science and computer engineering and the director of external engineering programs at the University of Bridgeport. He was an associate professor of computer science and computer engineering at the University of Bridgeport from 1995 to 1999, a research assistant professor of computer science at the Department of Computer Science, University of Utah, from 1992 to 1995, and a research fellow at the General Robotics and Active Sensory Perception (GRASP) Laboratory of the University of Pennsylvania from 1989 to 1991. He was the chairman of the Discrete Event and Hybrid Systems Technical Committee of the IEEE Robotics and Automation Society from 1992 to 1999, and the chairman of the Prototyping Technical Committee of the IEEE Robotics and Automation Society from 1999 to 2001. His background is in the fields of computer science and engineering, control theory, robotics, automation, manufacturing, AI, computer vision, and signal processing. Dr. Sobh's current research interests include reverse engineering and industrial inspection, CAD/CAM, active sensing/imaging under uncertainty, robots and electromechanical systems prototyping, sensor-based distributed control schemes, unifying tolerances across sensing, design, and manufacturing, hybrid and discrete event control, modeling, and applications, and mobile robotic manipulation. He has published over 130 refereed journal and conference papers, and book chapters in these and other areas. Dr. Sobh edited or coedited issues of several international research journals in these areas and is a member of the editorial boards of the *Computing Journal* and the *International Journal of Science and Technology.* He has been on the program committees of several international conferences and has chaired and organized several conferences, sessions, workshops, and tracks in robotics, automation, and sensing meetings and has made many presentations, invited talks, invited lectures and colloquia, seminars, and panel participations, at research meetings, university departments, research centers, and companies. Dr. Sobh is active in consulting and providing service to many industrial organizations and companies. He has consulted for many companies in the United States, Switzerland, India, Malaysia, Dubai, and Egypt, to support projects in robotics, automation, manufacturing, sensing, numerical analysis, and control. He has also worked at Philips Laboratories in New York, and a number of companies in Egypt. Dr. Sobh has been awarded many grants to pursue his work in robotics, automation, manufacturing, and sensing. Dr. Sobh is a licensed professional electrical engineer (P.E.), a certified manufacturing engineer (CMfgE) by the Society of Manufacturing Engineers, a certified professional manager (C.M.) by the Institute of Certified Professional Managers at James Madison University, a certified reliability engineer (C.R.E.) by the American Society for Quality Control, a member of Tau Beta Pi (The Engineering Honor Society), Sigma Xi (The Scientific Research Society), Phi Beta Delta (The International Honor Society), and Upsilon Pi Epsilon (The Computing Honor Society). Dr. Sobh was the recipient of the Best Research Award by the World Automation Congress in 1998. Dr. Sobh is a member or senior member of several professional organizations including; ACM, IEEE, IEEE Computer Society, IEEE Robotics and Automation Society, IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence (PAMI), the International Society for Optical Engineering (SPIE), the National Society of Professional Engineers (NSPE), the NewYork Academy of Sciences, the American Society of Engineering Education (ASEE), the American Society of Quality (ASQ), the American Association for the Advancement of Science (AAAS), the Society of Manufacturing Engineers (SME), and a founding member of the Society for Industrial Computing.

Damir Vamoser graduated with a BS in computer science degree and is currently a master of science candidate in computer science at the University of Bridgeport. He is a recipient of several All-American Scholar Collegiate Awards and was a distinguished member and the president of the professional development committee of the Bridgeport Rotaract Club. Mr. Vamoser is currently employed by Jupitermedia Corporation, formally Internet.com, as a senior developer and project leader. His development efforts are centered on building medium to large-scale e-commerce systems. His current fields of research include building and optimization of large-scale task-specific distributed systems.