Case Study 1—Requirements Specification Document

1 Abstract

This is the requirements document for the case study that will be used throughout the book. The system to be developed is for **scheduling the courses in a computer science department**, based on the input about classrooms, lecture times, and time preferences of the different instructors. **Different conditions have to be satisfied by the final schedule**.

This document outlines the requirements for the development of a course scheduling system for a computer science department. The system will schedule courses based on classroom availability, lecture times, and instructor preferences while adhering to specified conditions. This document follows the IEEE standard for a requirements specification document, with some variations.

2 Introduction

Purpose

The purpose of this document is to describe the external requirements for a course scheduling system for an academic department in a University. **It also describes the interfaces for the system.**

Document details the external requirements and interfaces for a course scheduling system for an academic department at a university. It serves as a comprehensive guide for developers and stakeholders.

Scope

This document is the only one that describes the requirements of the system. It is meant for use by the developers and will be the basis for validating the final delivered system. Any changes made to the requirements in the future will have to go through **a formal change approval process.** The developer is responsible for asking for clarifications, where necessary, and will not make any alterations without the permission of the client.

Definitions, Acronyms, Abbreviations

Definitions

- ➔ Course Scheduling System: A software application designed to organize and manage the scheduling of courses within an academic department, ensuring optimal use of classrooms and alignment with instructor preferences.
- ➔ Course Enrollment: The number of students registered for a particular course.
- ➔ Lecture Time Preferences: Specific times when an instructor prefers to conduct their lectures.

➔ Conflict Report: A report generated by the system listing courses that could not be scheduled, along with the reasons for the scheduling conflicts.

Acronyms

UNIX: Uniplexed Information and Computing System

BSD: Berkeley Software Distribution

Abbreviations

MWFd: Monday, Wednesday, Friday at a specific time in the day (e.g., MWF9)

MWFdd: Monday, Wednesday, Friday at a specific time with double digits (e.g., MWF10:30)

TTdd:dd: Tuesday, Thursday at a specific time with double digits (e.g., TT10:30)

cap: Capacity (refers to room capacity)

Example:

courses:

course1, course2, course3

time:

Example: MWF9, TT10:30

References

ISO/IEC/IEEE 29148:2011 - Systems and software engineering - Life cycle processes - Requirements engineering. This international standard specifies the required processes and information items for requirements engineering and has been referenced for best practices in requirements management.

University Academic Scheduling Policies - Specific policies and guidelines set forth by the university regarding course scheduling, classroom assignments, and instructor preferences.

UNIX 4.2 BSD Documentation - Official documentation for the UNIX 4.2 Berkeley Software Distribution provides necessary technical details for ensuring system compatibility.

Course Catalog - The department's official catalog of courses, which includes details on course numbers, titles, descriptions, and enrollment expectations.

Developer’s Responsibilities

The developer is responsible for (a) developing the system, (b) installing the software on the client's hardware, (c) **conducting any user training that might be needed for using the system**, and (d) **maintaining the system for a period of one year after installation.**

3 General Description

Product Functions Overview

In the computer science department, there are a set of classrooms. Every semester the department offers courses, which are chosen from the set of department courses. A course **has expected enrollment and could be for graduate students or undergraduate students**. For each course, the instructor gives some time preferences for lectures.

The system is to produce a schedule for the department that specifies the time and room assignments for the different courses. Preference should be given to graduate courses, and no two graduate courses should be scheduled at the same time. If some courses cannot be scheduled, the system should produce a "conflict report" that lists the courses that cannot be scheduled and the reasons for the inability to schedule them.

User Characteristics

The primary users of this system will be department secretaries who have basic computer literacy and can use programs such as text editors and word processors.

General Constraints

The system should run on Sun 3/50 workstations running UNIX 4.2 BSD.

General Assumptions and Dependencies

Assumptions:

Users have basic computer skills.

Accurate input data will be provided.

Dependencies:

Availability of Sun 3/50 workstations and UNIX 4.2 BSD.

4 Specific Requirements

Inputs and Outputs

The system has two file inputs and produces three types of outputs.

Input file 1: Contains the list of room numbers and their capacity; a list of all the courses in the department catalog; and the list of valid lecture times. The format of the file is:

```
rooms
        room1 : cap1
        room2 : cap2
```

:
;

courses
course1 , course2 , course3 , ;

times
time1 , time2 , time3;

where room1 and room2 are room numbers with three digits, with a maximum of 20 rooms in the building; cap1 and cap2 are room capacities within the range [10, 300]; course1, course2 are course numbers, which are of the form "csddd," where *d* is a digit. There are no more than 30 courses. time1 and time2 are valid lecture times of the form "MWFd" or "MWFdd" or "TTd" or "TTd:dd" or "TTdd:dd". There are no more than 15 such valid lecture times. An example of this file is:

```
rooms
        101 : 25
        115 : 50
        200 : 250 ;
courses
        cs101, cs102, cs110, cs120, cs220, cs412, cs430, cs612, cs630 ;
times
          MWF9, MWF10, MWF11, MWF2, TT9, TT10:30, TT2, TT3:30 ;
```

Input file 2: **Contains information about the courses being offered**. For each course, it specifies the course number, expected enrollment, and a number of lecture time preferences. A course number greater than 600 is a post-graduate course; the rest are undergraduate courses. The format of this file is:

```
course   enrollment      preferences
 c#1        cap1     pre1 , pre2 , pre3 ...
 c#2        cap2 pre1 , pre2 , pre3 ...
                                  :
                                  :
```

where c#1 and c#2 are valid course numbers; cap1 and cap2 are integers in the range [3..250]; and pre1, pre2, and pre3 are time preferences of the instructor (a maximum of 5 preferences are allowed for a course). An example of this file is

```
course  enrollment            preferences
 cs101       180       MWF9, MWF10, MWF11, TT9
 cs412        80            MWF9, TT9, TT10:30
 cs612        35
 cs630        40
```

Output 1: The schedule specifying the class number and time of all the schedulable courses. The schedule should be a table having the lecture times on the *x*-axis and

classroom numbers on the *y*-axis. For each slot (i.e., lecture time, classroom) the course scheduled for it is given; if no course is scheduled the slot should be blank.

Output 2: List of courses that could not be scheduled and why. For each preference, the reason for inability to schedule should be stated. An example is:

cs612: Preference 1: Conflict with cs600.
Preference 2: No room with proper capacity.

Output 3: Error messages. At the minimum, the following error messages are to be given:

e1. Input file does not exist.
e2. Input-file-1 has error
e2.1. The course number has wrong format e2.2. Some lecture time has wrong format. e2.3. Classroom number has wrong format. e2.4. Classroom capacity out of range.
e3. Input-file-2 has error
e3.1. No course of this number. e3.2. No such lecture time.
e4. More than permissible courses in the file; later ones ignored. e5. There are more than permissible preferences.
Later ones are ignored.

Input file 1: Contains room numbers, their capacities, department courses, and valid lecture times. Format:

```
rooms
    room1: cap1
    room2: cap2
    :
    ;
    courses
        course1, course2, course3, ;
    times
        time1, time2, time3 ;
```

Input file 2: Contains course number, expected enrollment, and instructor's lecture time preferences. Format:

```
course enrollment preferences
c#1 cap1 pre1, pre2, pre3 ...
c#2 cap2 pre1, pre2, pre3 ...
:
:
```

Outputs:

Output 1: Schedule table with lecture times on x-axis and classroom numbers on y-axis. Filled slots show scheduled courses.
Output 2: List of unscheduled courses with reasons for conflicts.
Output 3: Error messages for various input file errors (e.g., missing files, format errors).

Functional Requirements

1. Determine the time and room number for the courses such that the following constraints are satisfied:

   (a) No more than one course should be scheduled at the same time in the same room.

   (b) The classroom capacity should be more than the expected enrollment of the course.

   (c) Preference is given to post-graduate courses over undergraduate courses for scheduling.

   (d) The post-graduate (undergraduate) courses should be scheduled in the order they appear in the input file, and the highest possible priority of an instructor should be given. If no priority is specified, any class and time can be assigned. If any priority is incorrect, it is to be discarded.

   (e) No two post-graduate courses should be scheduled at the same time.

   (f) If no preference is specified for a course, the course should be scheduled in any manner that does not violate these constraints.

   Inputs: Input file 1 and Input file 2.

   Outputs: Schedule.

2. Produce a list of all courses that could not be scheduled because some constraint(s) could not be satisfied and give reasons for unschedulability.

   Inputs: Input file 1, and Input file 2.
   Outputs: *Output 2*, i.e., list of unschedulable courses and preferences and why.

3. The data in input file 2 should be checked for validity against the data provided in input file 1. Where possible, the validity of the data in input file 1 should also be checked. Messages should be given for improper input data, and the invalid data item should be ignored.

   Inputs: Input file 1 and Input file 2.

   Outputs: Error messages.

External Interface Requirements

User Interface: Only one user command is required. The file names can be specified in the command line itself or the system should prompt for the input file names.

Performance Constraints

For input file 2 containing 20 courses and up to 5 preferences for each course, the reports should be printed in less than 1 minute.

Design Constraints

Software Constraints

The system is to run under the UNIX operating system.

Hardware Constraints

The system will run on a Sun workstation with 256 MB RAM, running UNIX. It will be connected to an 8-page-per-minute printer.

Acceptance Criteria

Before accepting the system, the developer must demonstrate that the system works on the course data for the last 4 semesters. The developer will have to show through test cases that all conditions are satisfied.

Incorporate Additional Suggestions:

- **Scenarios and Use Cases**

Scenario 1: A professor prefers to teach in the morning but the requested time slots are all occupied. The system attempts to schedule the course in the next available preferred slot.

Use Case 1: A department administrator uploads input files containing room and course information. The system processes the files, generates the schedule, and outputs both the successful schedule and a conflict report.

Use Case 2: A professor requests a specific lecture time, but due to room capacity constraints, the course is scheduled at an alternative time that meets room availability.

- Diagrams and Visuals

Flowchart: A flowchart detailing the scheduling process from input file processing to schedule generation and conflict reporting.

System Architecture Diagram: An overview of the system's architecture, including input processing, scheduling logic, and output generation.

- Testing and Validation

Unit Testing: Each component of the system (e.g., file input processing, scheduling logic, error handling) will undergo unit testing to ensure correct functionality.

Integration Testing: The system will be tested as a whole to ensure that all components work together seamlessly.

Validation: The final schedule will be validated against the input requirements and instructor preferences to ensure accuracy.

- User Manual

System Overview: Introduction to the course scheduling system and its purpose.

Input Files: Instructions on how to prepare and upload input files.

Generating the Schedule: Steps to run the scheduling process and generate outputs.

Interpreting Outputs: How to read the generated schedule, conflict reports, and error messages.

Troubleshooting: Common issues and solutions, such as dealing with format errors or unscheduled courses.