Software Engineering Group Projects – Requirements Specification

Author: C. J. Price
Config Ref: None
Date: 5th November 2001
Version: 1.31
Status: Released

Department of Computer Science
University of Wales
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1. INTRODUCTION

1.1 Purpose of this Document

This document describes the requirements for the MSc Software Engineering Group Project 2001. It should be read in the context of the Group Project, taking into account the details of the group project assignment and the group project quality assurance (QA) plan [1].

1.2 Scope

This requirements specification aims to provide information about what is needed from the new church heating system for St Pauls Methodist Centre Aberystwyth. It describes the requirements that should be met when constructing the system, and the attributes of the finished product that is expected.

This document should be read by all members of the group project.

1.3 Objectives

The objectives of this document are:

- To describe the background to the group project application for 1998 (Building Heating System)
- To provide details of the criteria that the group project product must meet
- To describe the types of interaction with the system which must be provided

2. GENERAL DESCRIPTION

2.1 Product Perspective

St Pauls Methodist Centre is a modern building housing both a Welsh and an English speaking congregation. It has a number of rooms, and meetings happen in some of those rooms every evening of the week. Heating is provided by a large gas powered hot water boiler. It would be expensive and wasteful to heat the whole building every night of the week, and so independent control of the supply to a number of regions of the building is supplied. This means that there are a number of valves throughout the building, and the practical way of controlling them is through computer control.

The necessary hardware, along with routines in Java to call the hardware, will be provided. This project will produce software allowing the users to enter details of meetings, and software to control the radiator valves depending on the meetings.

The users of the system will not be experienced computer users, and so the user interface should be designed with that in mind. A simple graphical interface with minimum user input is expected. A display showing the present state of the system should be available, to give feedback to users on what the system is doing.

2.2 Product Functions

The Automated Church Heating Environment (ACHE) must perform the following functions:

- Enable the users to make room bookings of the following kinds in a meeting diary

  Single or multiple room bookings

  Single occasion bookings or weekly bookings (on one or more days of the week) or bookings for odd weeks of the month or bookings of two or three or 4 week frequency

  Multiple occasion bookings should be given a duration

- Control the central heating independently from the room booking system

  Turn on the boiler and hot water pump in advance whenever a room needs to be heated

  Turn on the room heating in advance of the start of the meeting by an appropriate warm up time

  Turn off the room heating at the end of the meeting with an appropriate cool down time

  Turn off the boiler and hot water pump when no meetings are expected

  Act within five minutes of any changes to the meeting diary

- Keep logs of all actions taken, along with the details of the meetings which prompted those actions in a transaction log

- Provide a display showing the present state of the heating system

The listed functions describe what the system must do, and must do reliably in order to keep user confidence.

2.3 User Characteristics

Users will have some experience of computers (e.g. use of word processors), and so will be used to ‘point and click’ interfaces. Thy will not be computer scientists, and so will want the entry of data made as easy as possible.

3. SPECIFIC REQUIREMENTS

3.1 Functional Requirements

FR1 Meeting diary storage facility

The ACHE should provide a meeting diary storage facility with the following operations:

Add meeting entry
Delete meeting entry
Amend meeting entry
Purge all out of date meeting entries
List all meetings in the diary
List all meetings in the diary that apply to <date range>, e.g. all meetings in September 1998.

This should be provided as a simple text file, of the format shown in appendix B. The overhead of a database is deemed inappropriate because of the low likely demands on the diary.

FR2 Kinds of meetings in the diary

For each meeting entry, the user should be able to specify the following fields:

A description of the meeting
Start and end times for the meeting
Rooms in which the meeting will be held
Date(s) when the booking will happen

In order to make the booking interface simpler, meetings may not last longer than 23 hours and 59 minutes, and so where the end time is earlier than the start time, it will be assumed to be during the next day.

Date choices for a meeting are one of:

Single date
Repeated weekly (allowing the user to specify which days of the week it will be repeated)
Repeated odd weekly (i.e. fortnightly or every 3 or 4 weeks, starting on the given date)
Repeated odd weeks of the month (e.g. 1st and 3rd week of the month, starting on given date)

Rooms in the building are:

Hall
Octagon
Concourse
Lounge
Office
Crèche
Classroom One
Classroom Two
Classroom Three

FR3 Meeting booking software

Booking software will be produced to enable the user to change the stored meeting diary. It will enable the user to access each of the facilities listed in FR1, for each of the types of booking listed in FR2. Requirement EIR5 describes some of the demands on the interface to the booking software in greater detail.

FR4 Heating control software

Heating control software (HCS) must be produced which will take information from the meeting diary, and turn heating on and off in rooms within the church building appropriately. FR6 to FR9 give further information on what is appropriate.

FR5 Handling of year boundaries

The booking software and the heating control software working together should correctly handle year boundaries. It should be possible to enter meetings in the diary which extend for up to two years into the future. The heating control software should correctly handle such meetings. In particular, the software should correctly deal with bookings in the years 1999-2099.

FR6 HCS initialisation and shutdown

The HCS is expected to run continually. However, on start up, it should turn off all heating, to ensure that the system is in a stable state. If a shutdown is necessary, then the HCS should again turn off all heating if possible.

FR7 Turning heating on and off.

HCS should turn on the heating in any room when there is a booking for the room covering the present time. However, this should take into account the prewarming and cooldown times detailed in FR8. Whenever the heating should be on in any room, the boiler and hot water pump should also be activated. The boiler also needs to be preheated, and so should be turned on before any room heating is needed, where possible (see boiler preheat in FR8).

FR8 HCS prewarming and cooling

The times of meeting do not correspond exactly to when the heating should be turned on in a room. Ideally, the heating is turned on before the meeting starts, and turned off just before the meeting ends, to maximise the efficiency of the system. This is achieved by having a preheat time and a cooldown time, and moving the time at which the heating goes on and off by those times. In addition, the boiler preheat time means that the boiler should be put on by a set amount before the first room is heated.
All of the preheat time, cooldown time and boiler preheat time should be adjustable by the user, in order to take into account the changes in external conditions from season to season. A change to any of the three times should take effect within 24 hours.

FR9 Interactions between the booking system and the HCS

It is possible for users to enter new bookings while the HCS is running. The HCS needs to incorporate those bookings into its schedule for turning the heating on and off. This should be done within five minutes of any new booking being made.

FR10 Log of HCS transactions

All changes to the heating should be recorded, along with the details of the booking(s) which caused the change to occur. The log should be time stamped, and in order of occurrence, so that it is easy to examine.

FR11 Display of current HCS state

The system should provide a visual indication of the current state of each room in the building (whether it is being heated or not). Ideally this will be provided as a graphical display of the church interior, with rooms being heated coloured red, and other rooms not coloured. A simpler, but acceptable solution is a textual display of which rooms are being heated. This should be updated at least every five minutes.

FR12 E*xamination of HCS transaction log*

The user should be able to select a log file to open by choosing the day on which the log was written. That log file will then be opened, and the entries in the log file will be listed. The user will be able to scroll along the list of log events if it is longer than can be shown on screen at once.

3.2 External Interface Requirements

A number of necessary routines have been provided by the central heating system manufacturer, along with hardware to interface a computer to the central heating system. They are all packaged in class CentralHeating, detailed in Appendix A.

EIR1 Initialisation of heating system state

The HCS needs to be initialised at the start of the heating run. This should be done by creating an instance of the class CentralHeating. The present date and time should be passed as a parameter to this instantiation. It should be possible to provide both the present date and time and the name of the diary file as parameters to the program. This allows serious testing of the system with standard test files.

EIR2 Use of accurate timing calls

All date and time related calls should be made to the manufacturer's provided calls, and not to any computer-specific mechanism. Two mechanisms are provided in class CentralHeating for accurate timing:

1. ReturnTimeAndDate
2. WaitForMinutes

The first routine provides the absolute time and date now. This is provided relative to the time elapsed since the initialisation time. The second routine is to be used to pause the system for a set number of minutes between 1 and 5.

The recommended mode of operation for use of the central heating system is to perform all necessary changes to the central heating system, and then pause for a set number of minutes. The system is not guaranteed to work if the programmer chooses to loop waiting for time to pass, rather than using the routines provided.

EIR3 Use of valve operation routines

The routines TurnOn and TurnOff should be used in order to turn on or off the boiler, hot water, or any room heating. Full details are provided in appendix A.

EIR4 Log of HCS transactions

The log should be saved at the end of each day in a file with a name containing the date, and a new log should then be opened.

EIR5 User interface to booking system

The user interface for the booking system should be form based. A single form should cover all of the options for entering and amending a booking. Where some options on the form are not available because of choices on the form that have already been made (e.g. because the user has chosen that this is a single booking rather than a multiple booking), then unavailable choices should be greyed out.

3.3 Performance Requirements

PR1 Response of the HCS

The HCS should turn on the heating in any room within five minutes of the expected meeting time, taking into account the prewarming time set. Similarly, it should shut off the heating in any room within five minutes of the expected time, taking into account the cooldown time set.

PR2 Target computer for system

Both the booking system and the HCS should be able to run on a Windows 95 PC as well as on a Sun Sparc.

3.4 Design Constraints

DC1 Addition of further rooms

Extensions to the building may be made at a later date. It is important that it is easy to extend the booking system and the HCS to deal with the extra rooms. It may also be possible to apply the system to other buildings, and so some thought should be given to how the code can be reused in other buildings.

DC2 Use of JAVA

It is corporate policy to use Java on all major new developments, and so Java will be used for all coding on this project.

3.5 Other Requirements

The ASB will be developed in line with the company's QA plan, detailed in [1].

4. REFERENCES

[1] H. R. Nicholls. Software engineering group projects - quality assurance plan. Technical Report SE.QA.01, University of Wales, Aberystwyth, 1998.

Appendix A – Provided Interface Routines

A number of interface routines have been provided by the hardware manufacturers, and must be used where appropriate. They are bundled in package heating.utils, which provides the following interface:

```
package heating.utils;

class DateAndTime{
        public int minute;
        public int hour;
        public int day;
        public int month;
        public int year;
}

class CentralHeating{

   public String [] roomList();
   // Returns list of legal room names plus boiler and hot water pump
   // List is created on class initialisation.
   // Manufacturer can upgrade it at a later date for a fee.

   public void CentralHeating(DateAndTime dt);
   // Initialises the system

   public void CentralHeating(DateAndTime dt, String frameLabel);
   // Initialises the system, and provides a feedback window

        public DateAndTime returnTimeAndDate();
        // Returns date and time object as result

        public void waitForMinutes(int duration);
        // Returns processing to the user after duration minutes have passed

        public void turnOn( String roomName );
        // Turns on the heating in room roomName - must be a legal name from roomList

        public void turnOff( String roomName );
        // Turns off the heating in room roomName - must be a legal name from roomList
}
```

Appendix B – Format of meeting diary file

An entry in the meeting diary file is a single line of text, and has the general format:

```
entry          ::=   single-letter main-body (end-date) (days-of-week) (week-gap) (which-week)

single-letter  ::=   'S' | 'W' | 'O' | 'M'

main-body      ::=   '[' description-text ']' ['start-time']' ['end-time']' ['room-list ']' ['start-date ']'

start-time     ::=   time
end-time       ::=   time
time           ::=   digit digit ':' digit digit

room_list      ::=   '[' { 'Hall' 'Octagon' 'Concourse' 'Lounge' 'Office' 'Creche' 'Class1' 'Class2' 'Class3' } ']'

start-date     ::=   date
end-date       ::=   date
date           ::=   '[' digit digit '/' digit digit '/' digit digit digit digit ']'
```

days-of-week ::= '[' { 'Monday' 'Tuesday' 'Wednesday' 'Thursday' 'Friday' 'Saturday' 'Sunday' } ']'

week-gap ::= '[' 2' | '3' | '4' ']'

which-week ::= '[' { '1' '2' '3' '4' '5' } ']'

Every entry must have one of the following formats:

Single Date:
S main-body
e.g. S [Church Council] [19:30] [22:00] [Octagon] [14/10/1998]

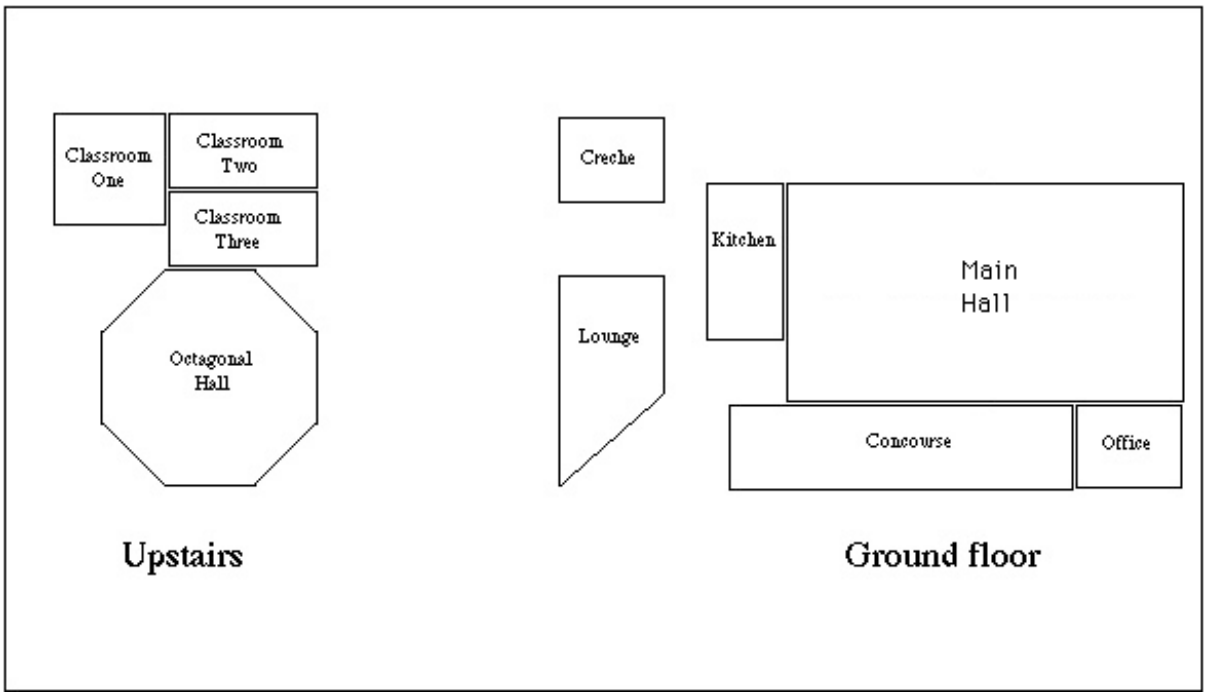
Weekly:
W main-body end-date days-of-week
e.g. W [Brownies] [19:00] [21:00] [Hall Concourse] [12/10/1998] [30/05/1999] [Tuesday]

Odd weekly:
O main-body end-date week-gap
e.g. O [Singers] [19:45] [21:30] [Concourse] [12/10/1998] [20/12/1998] [2]

Odd weeks in the month:
M main-body end-date which-week
e.g. M [Welsh morning service] [10:00] [11:30] [Octagon] [04/10/1998] [04/12/1998] [1 2 3 4]

Quick key: () means optional; {} means 1 or more; | means or;

Appendix C – Diagram of interior of church building



DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	12/10/98	N/A - original version	CJP
1.1	N/A	2/11/98	Appendix A updated to meet project Java standards.	CJP
1.2	N/A	11/11/98	EIR1corrected – file not a parameter to heating. Appendix B corrected – letter at start of two examples; added date to Odd month, quick key.	CJP
1.3	N/A	10/2/99	Appendix B corrected – syntax for single meeting was incorrect.	CJP
1.31	1	05/11/01	Added info to scope section (1.2)	CJP