Software Requirements Specification

for

MiniThermostat Software

Document Version <1.0>

Prepared by

Group Name: DoePwr

John Doe	0000000	doe@mcmaster.ca

Instructor:	Dr. K. Sartipi
Course:	Software Engineering 3M04/3K04
Lab Section:	X
Teaching Assistant:	Jane Doe
Date:	01/01/01

Contents

List of Figures

List of Tables

REVISIONS

Version	Primary Author(s)	Description of Version	Date Completed
1.0	John Doe	Complete requirements for the MiniThermostat Software project.	01/01/01

1 Introduction

This document provides a complete example of a Software Requirements Specification (SRS) document for a small thermostat software system (namely MiniThermostat) that is meant to control the temperature of an indoor area. In the following sections, we specify the purpose of this document, its intended audience, the scope of the end-product, and all sources used in the production of this document.

1.1 Document Purpose

The purpose of this document is to specify and establish the functional and non-functional requirements associated with the MiniThermostat Software version 1.0. MiniThemostat version 1.0 SRS will serve as the backbone for any other documents to be developed for this project in the future. Moreover, it will provide the basis for the future software verification and testing by specifying the behaviour requirements of the system in the form of a Use Case Diagram and State Machines.

1.2 Product Scope

MiniThermostat shall be a computer based program which will allow the user to perform climate control functions in an indoor area. MiniThermostat will receive readings from thermal sensors located in the area to be monitored and will control three hardware appliances – a furnace, an air-conditioner and a fan. The software shall allow the user to program eight different time intervals during the weekdays. Finally, MiniThermostat will monitor the temperature of the area, previnting it from droping below, or rising above the target temperature by at most 3^0 Celsius.

MiniThermostat will be of the most benefit to those organizations that have small to medium-size buildings with multiple areas and a centralized control centre. Furthermore, if these areas are accessed by employees only on specific days and hours, MiniTermostat shall minimize electricity usage and ensure that the area has a proper climate setting when it is to be occupied. On the other hand, MiniThermostat can also be used in a personal home setting and replace the wall mounted thermostat.

1.3 Intended Audience and Document Overview

This SRS document is the result of interviews between the system architect and the client and represents the mutual agreement between these two parties to bind a contract. Therefore, this SRS document should be carefully read and approved by the client, the development team, and the Quality Assurance (QA) team. Section 2 of this document provides an overall description of the product by specifying product perspective, high level functionality of the product and other general, product related requirements. Section 3 describes the specific requirements of the software including: External Interfaces, Functional Requirements and Behaviour Requirements in the form of use cases and state diagrams. This section will be of the most interest to the project architects, developers and the QA team. Section 4 describes the non-functional requirements of the product and focuses on the different software attributes, such as – maintainability, security, performance, etc.

1.4 Definitions, Acronyms and Abbreviations

ACB – Appliances Control Block

GUI – Graphical User Interface
LAN – Local Area Network
QA – Quality Assurance
UI – User Interface

1.5 Document Conventions

1.5.1 Formatting Conventions

Several formatting conventions have been followed throughout the entire document:

1. All text contained in this document is 11pt Arial font.
2. Section titles are 18pt Arial font.
3. Subsection titles are 14pt Arial font.
4. Any further subsection breakdown is 12pt Arial font.
5. All sections and subsection are numbered using the X.X.X… format, where X represents numbers.
6. ***Introduced terms are in bolded Times New Roman italics.***
7. *Any further repetition of these terms is in Times New Roman italics.*

1.5.2 Naming Conventions

The naming conventions are the means of making the SRS more understandable and easier to follow; they are also used to build a structure for the whole software system. The conventions are used for variables, function names, packages, etc.
The following naming conventions have been used to define the different variables in this SRS document:

1. All constants are CAPITALIZED.
2. All variables representing input are prefixed with **i_**.
3. All variables representing output are prefixed with **o_**.
4. Internal states are prefixed with **s_**.
5. All system variables are prefixed with **v_**.

NOTE: students are encouraged to define naming conventions for the functions, packages, modules, etc. as well.

1.6 References and Acknowledgments

The following standards apply:

J-STD-016-1995	IEEE/EIA Standard for Information Technology, Software Lifecycle Processes, Software Development, Acquirer-Supplier Agreement
IEEE-STD-P1063	IEEE Standard for Software User Documentation

The following texts have been used in the process of developing this document:

[1] J. Rumbaugh et al. *Object Oriented Modelling & Design*. Upper Saddle River, NJ: Prentice Hall, 1991.
[2] C. Ghezzi et al. *Fundamentals of Software Engineering*. Upper Saddle River, NJ: Prentice Hall, 2003.

2 Overall Description

Better Thermostats INC (customers), the makers of the Wikond™ programmable thermostat, want to introduce a new product that will significantly differ from the existing line of products, which are the wall mounted, programmable thermostats. An example of the Wikond™ wall mounted, programmable thermostat is depicted in Figure 1. The new product shall resemble all of the functionality of the wall mounted thermostat and take advantage of the recently developed device – Appliances Control Block (ACB). The hardware appliances – furnace, air-conditioner and the fan are connected to the ACB which operates their power relays. All thermal sensors from the monitored area are also connected to the control block. The control block itself has a network interface and is connected to the LAN of the organization.

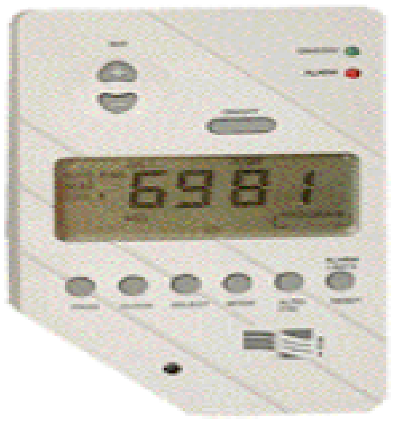


Figure 1: *Wikond™ programmable thermostat*

2.1 Product Perspective

The product perspective is best illustrated in Figure 2, which describes the overall environment and the ways components of this environment interact with each other.

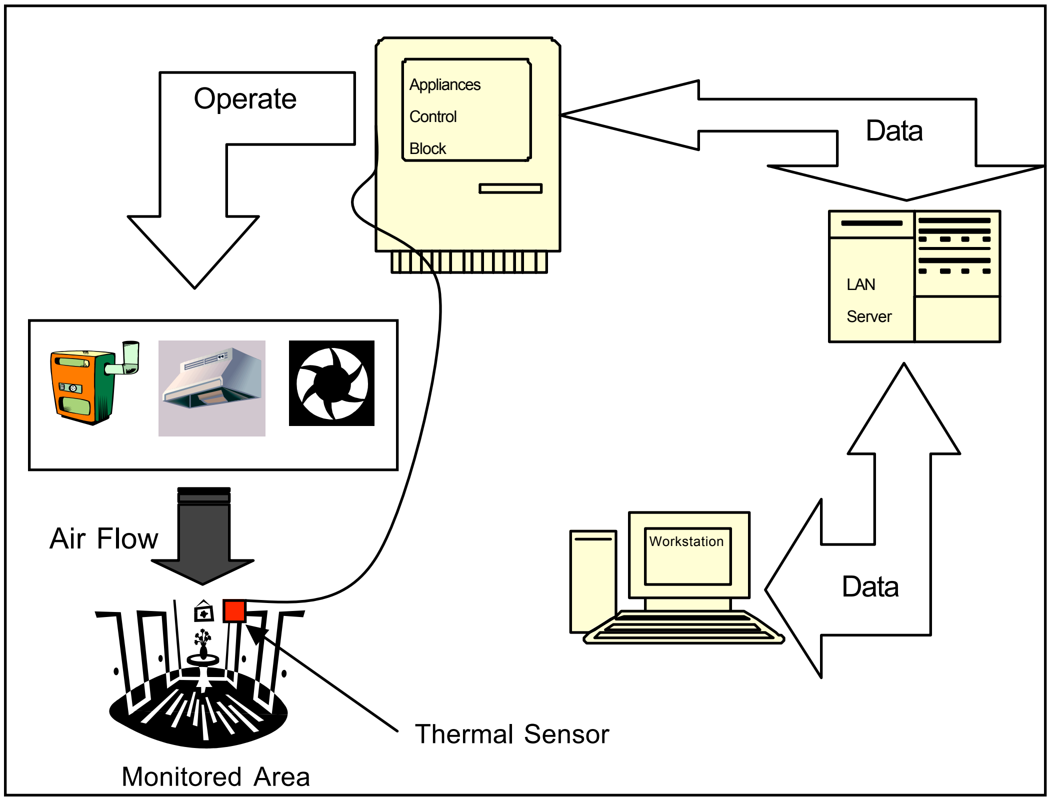


Figure 2: *MiniThemostat environment*

MiniThermostat shall perform all the regular functions of a thermostat. While running, the software shall operate the furnace or the air conditioner to keep the current temperature, represented by the system variable ***v_curr_Temp,*** equal to the target temperature, represented by the system variable ***v_target_Temp***. The target temperature is taken from a table of program values supplied by the user. A table shall specify target temperatures for 8 different time periods for the weekdays. The target temperature shall be reset from the table at the beginning of each program period and the user shall be able to override *v_target_Temp* for the current period.

2.2 Product Functionality

The whole functionality of th MiniThermostat can be categorized into two sets of operations that are specified in section 3.2:

1. **User Interface Operations:** provide the Graphical User Interface (GUI) and shall be responsible for all interactions between the user and the software. The user shall be able to perform the following operations through the GUI; switching between two main system states – **s_Program** and **s_Operate:**
   a. While in *s_Program* the user shall be able to:
      i. Specify eight different programs: Each program (representing a time period to be programmed) shall consist of three user inputs: ***i_WeekDay***, ***i_Time*** and ***i_Temp***, where *i_WeekDay* corresponds to a specific week day chosen from {MONDAY, TUESDAY, …, FRIDAY}, *i_time* corresponds to 24hr based day-time specifying the beginning of the time period (e.g. 2020) and *i_Temp* corresponds to the target temperature provided in degrees Celsius. The value of *i_Temp* shall be bounded between **MIN_TEMP** = 5^0 and **MAX_TEMP** = 35^0 degrees Celsius.
   b. While in *s_Operate* the user shall be able to:
      i. Switch between three different season states: ***s_Heat***, ***s_Cool*** and ***s_Off***. Each of these states corresponds to the device being controlled by ACB: *s_Heat* – furnace, *s_Cool* – air conditioner, *s_Off* – none.
      ii. Switch between two different fan states: ***s_FanOn*** and ***s_FanAuto***. When the fan is in the *s_FanOn* state, the fan runs continuously. When the fan is in the *s_FanAuto* state, it runs only when the furnace or the air conditioner are operating.
      iii. Override current *v_target_Temp* manually and hold the new temperature for that time period. When changing the current value of *v_target_Temp* the software shall not let the user exceed the minimum and maximum values specified in **i**.

2. **Thermostat Controller**: This functionality shall be responsible for the following system operations:
   a. Receiving values for *v_curr_Temp* continuously, and comparing these values to *v_target_Temp*, while in the *s_Operate* state.
   b. Running Furnace, AC and Fan with accordance to the season and fan states to maintain the relationship: $v\_target\_Temp - 3^0C < v\_curr\_Temp < v\_target\_Temp + 3^0C$. This threshold, of 3^0 Celsius will be modelled by the system constant **THRESH**, i.e., THRESH := 3.

2.3 Users and Characteristics

There will be only one user type operating the specified system. From now on, this user shall be referred to as the Operator. No special knowledge or skills shall be assumed on the part of this user.

The Operator is in general comfortable with computers and has no difficulties operating wall mounted thermostats.

2.4 Operating Environment

The software shall operate on an Intel based architecture personal computer (Pentium III or above), running Windows 98 or above. The computer station shall be connected to the Local Area Network and the operating system shall support this network interface.

2.5 Design and Implementation Constraints

Upon completion of the requirements analysis process the following constraints were identified:

- Communication with ACB must be performed through LAN using TCP/IP suite of protocols.
- The ACB control libraries are in the C language, therefore the project must be implemented in some variation of the C programming language.
- The system shall be a standalone application.
- The product must monitor the current temperature at all times.
- The operator can only modify one program at a time.

2.6 User Documentation

At the present time no user documentation is available for this product. This document should serve as the basis for all user documentation to follow. Each step in the development of the product shall be documented and a detailed user manual shall be compiled during the development process.

2.7 Assumptions and Dependencies

The following is a list of assumed factors that could significantly affect the requirements stated in this document.

- The Windows OS and the minimum hardware must be in place.
- The workstation will run continuously, and shall be restarted only at predefined times known to the operator. A safety system should be purchased separately to handle power outage and fault-tolerant backup system.
- System time and calendar settings are accurate and up to date.
- The ACB is well maintained and in a good working condition.
- The Local Area Network is well maintained and is not connected to any outside networks such as, the internet. If connected, an adequate protection is guaranteed for the operator workstation, which will not allow outside sources to access the product.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface shall consist of two main GUI screens. The two interfaces are: ***Main Menu – UI01*** and ***View Programs – UI02***. All interfaces shall have consistent items, with regards to the size of the windows, buttons, messages, etc.

3.1.1.1 Main Menu – UI01

The Main Menu screen shall serve as the main UI for the software. While the system is in the Main Menu it shall be in the *s_Operate* state.

UI01 screen shall display the following values:

- current day of the week;
- current time;
- current target temperature;
- ACB season state; and
- Fan state

The Operator shall be able to perform the following operations from UI01:

- change the season state;
- change the fan state;
- override the current target temperature;
- enter the View Programs interface; and
- suspend MiniThermostat program.

UI01 shall have the following list of switches and buttons that will generate system events when switched/pressed.

- ***Season Switch***: allows the Operator to choose one of the three season options: Heat, Cool, or OFF. According to the chosen option the system hanges the ***v_Season*** system variable to one of the three states *s_Heat, s_Cool,* and *s_Off*.

- ***Fan Switch***: allows the Operator to choose one of the two fan states: Fan ON or Fan AUTO. According to the chosen option the system changes the ***v_Fan*** system variable to one of the two states – *s_FanOn* or *s_FanAuto*.
- ***Temperature Up*** and ***Temperature Down*** buttons: these buttons shall allow the Operator to override the current *v_target_Temp* to a new value. The Operator shall not be allowed to exceed MIN_TEMP, or MAX_TEMP values when changing *v_target_Temp*. These buttons are part of the interface implementation. The feature of changing the current target temperature may be also implemented in a different manner, e.g., read input directly from the user, provide a textbox, etc.

- ***Exit*** button: by pressing this button the Operator shall be able to minimize MiniThermostat and suspend it to operate in a background mode. Only the Exit button shall remain active. All other controls shall become disabled and inaccessible to the Operator. At the same time, the system variable ***v_MainState*** will change from *s_Operate* to *s_Exit*. An example screenshot of MiniThermostat while being in the *s_Exit* state is depicted in Figure 4.

- ***View Programs*** button: by pressing this button, the operator shall be able to generate an event that will change the System variable *v_MainState* from *s_Operate to s_Program,* disable UI01 and open UI02.

An example screenshot of UI01 is provided in Figure 3:

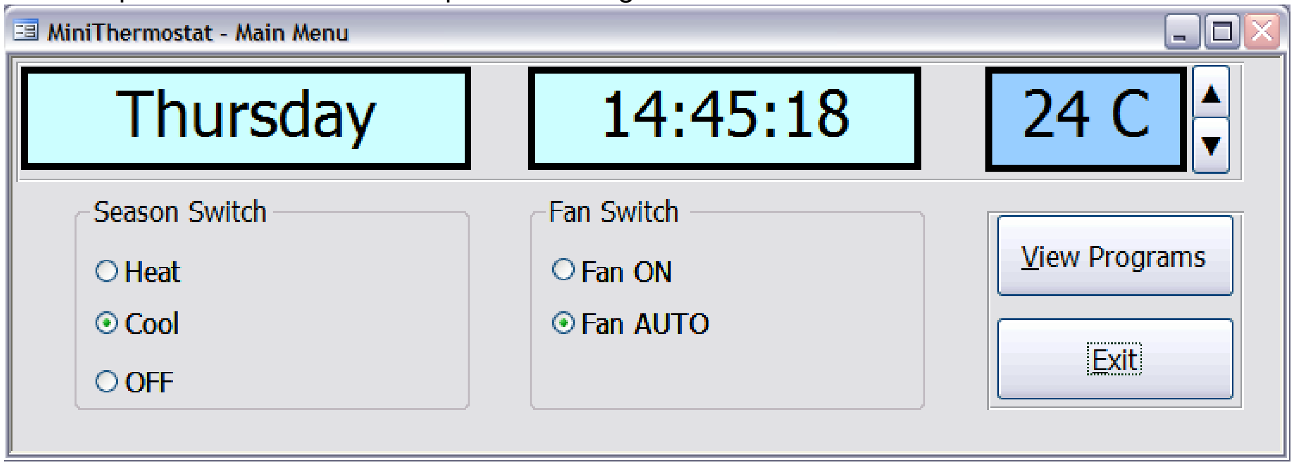


Figure 3: *MiniThermostat sample UI01*

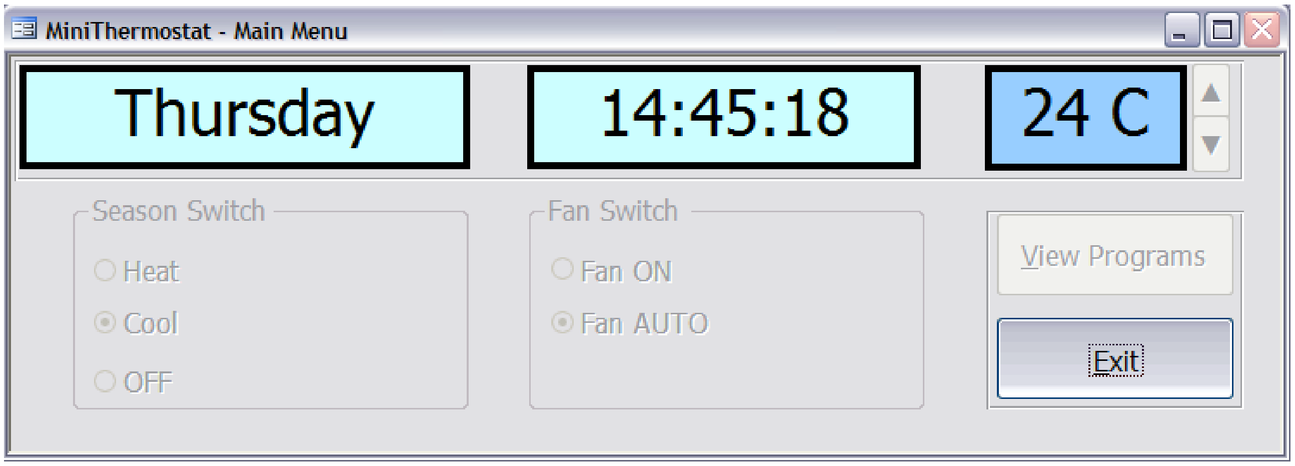


Figure 4: *MiniThermostat sample UI01 in s_Exit state*

3.1.1.2 View Programs – UI02

While in UI02, all monitoring activities shall be suspended and the system shall enter the *s_Program* state. Upon entering this state, an ***idle_Time*** counter shall be started. This counter shall be restarted every time the operator generates an event. If *idle_Time* counter reaches 90 seconds, the software shall automatically resume to *s_Operate* state, close UI02 and enable UI01.

UI02 shall display the following values:

- program number;
- selected day;
- selected time; and
- target temperature for the specific day and time

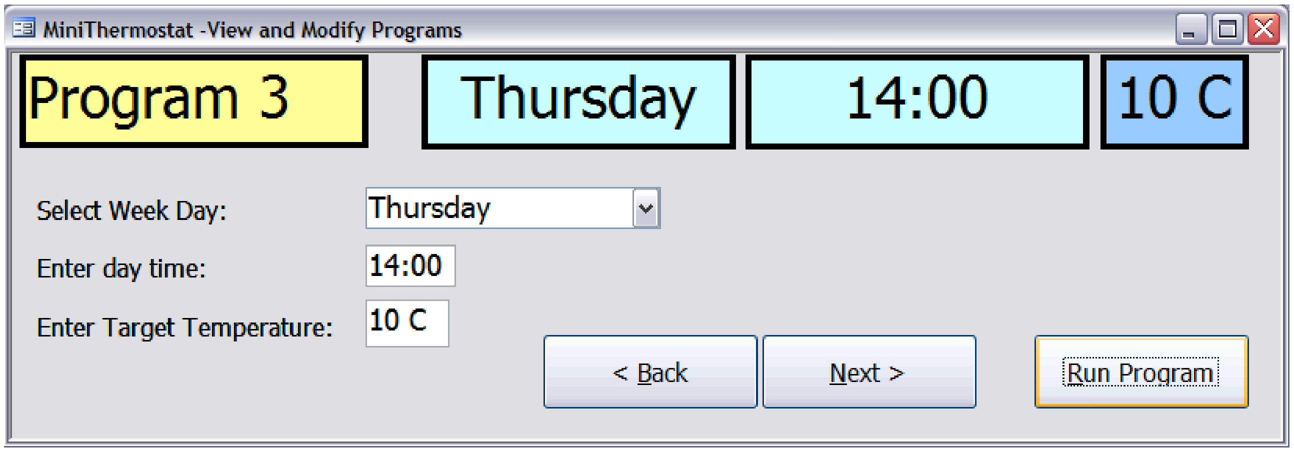
The Operator shall be able to perform the following operations while in UI02:

- easily switch between the eight different programs;
- edit the day, time and target temperature; and
- exit back to UI01 by choosing to run the edited programs.

UI02 shall have the following list of textboxes and buttons in its interface:

- ***Weekday textbox***: shall allow the Operator to input *i_Weekday*, from a list of valid inputs.
- ***Time textbox***: shall allow the Operator to input *i_Time*.
- ***Temperature textbox***: shall allow the operator to input *i_Temp*. The Operator shall not be able to exceed MIN_TEMP and MAX_TEMP values when entering the temperature.
- ***Back*** and ***Next*** buttons: shall generate events that will allow the Operator to move in a reasonable time between the eight different programs.
- ***Run Program*** button: by pressing this button the Operator shall be able to generate an event that will return the system to *s_Operate* state, find the relevant program for the current weekday and time and set *v_target_Temp* accordingly.

An example screenshot of UI02 screen is depicted in Figure 5:


MiniThermostat -View and Modify Programs

Program 3 | Thursday | 14:00 | 10 C

Select Week Day: Thursday

Enter day time: 14:00

Enter Target Temperature: 10 C

Figure 5: *MiniThermostat UI02 Screen*

3.1.2 Hardware Interfaces

The temperature sensors shall transform the indoor heat-level into a machine readable temperature value to be used as the input to the MiniThermostat system. The binary values of the system corresponding variables will be used to turn-on/off the Furnace, Air-conditioner, and Fan.

3.1.3 Software Interfaces

The system shall make use of the operating system calls to the file management system to store and retrieve data from the file containing the eight user specified programs.

3.1.4 Communications Interfaces

The system shall communicate with ACB using the TCP/IP suite of communication protocols. All communication with ACB shall be performed using a library supplied by the manufacturer-controlblock.lib.

3.2 Functional Requirements

This section describes in detail the two main system functionalities specified earlier in section 2.2.

3.2.1 User Interface Operations

- REQ 1 *Enter to view program*: upon pressing the button ***View Programs*** in UI01, this operation causes the system to enter to its *s_Program* state and the value of the variable *idle_Time* to be set to its initial value.

- REQ 2 E*nter to background mode*: causes the system to enter to *s_Exit* state, minimize UI01, and suspend the system to a background mode.

3.2.1.1 Thermostat Program Operations

- REQ 1.1 *Display and modify program*: given a valid program identifier ($1 \leq programID \leq 8$), this operation displays the values for weekday, time, and temperature corresponding to this program identifier on the screen. Finally, it shall allow the user to modify the values for weekday, time and temperature.

- REQ 1.2 *Store programs*: given a valid program identifier and valid values for weekday, time, and temperature, this operation causes the values to be stored in the system files.

- REQ 1.3 *Display next program*: upon pressing the ***Next*** button in UI02 and given the value of the current program identifier this operation displays the next program.

- REQ 1.4 *Display previous program*: upon pressing the button ***Back*** in UI02 and given the value of current program, this operation displays the previous program.

- REQ 1.5 *Run program*: upon pressing the ***Run Program*** button in UI02 and given the values of current weekday and time, this operation sets the varialbe *v_target_Temp* accordingly and returns to the main menu. In addition, this operation shall change the system state back to *s_Operate* from *s_Program*.

3.2.1.2 s_Operate State

- REQ 3.1 *Change season*: Upon changing the ***Season Switch*** position in UI01 this operation shall change the *v Season* system variable to the corresponding stateID ($stateID \in \{s\_Heat, s\_Cool, s\_Off\}$).

- REQ 3.2 *Switch fan state*: checks the current fan state and based on the result, switches to the alternative state. Only two states are possible – {*s_FanAuto, s_FanOn*}. This operation shall be evoked when the Operator changes the ***Fan Switch*** position in UI01.

- REQ 3.3 *Increase Temperature*: allows the Operator to increase the value of *v_target_Temp* by one degree given that this value will not be above MAX_TEMP. This operation shall be evoked when the operator presses **Temperature Up** button in UI01.

- REQ 3.4 *Decrease Temperature*: allows the operator to decrease the value of *v_target_Temp* by one degree given that this value will not be less than MIN_TEMP. This operation shall be evoked when the Operator presses **Temperature Down** button in UI01.

3.2.2 Thermostat Controller

- REQ 4 *Maintain Target Temperature*: given the target temperature, the state of the Season switch and the state of the Fan state this operation shall use controllblock.lib library to operate the different appliances and maintain the target temperature by continuously monitoring the temperature in the area.

3.3 Behaviour Requirements

3.3.1 Use Case Diagram

Figure 9 below, depicts the complete Use Case view of the system.

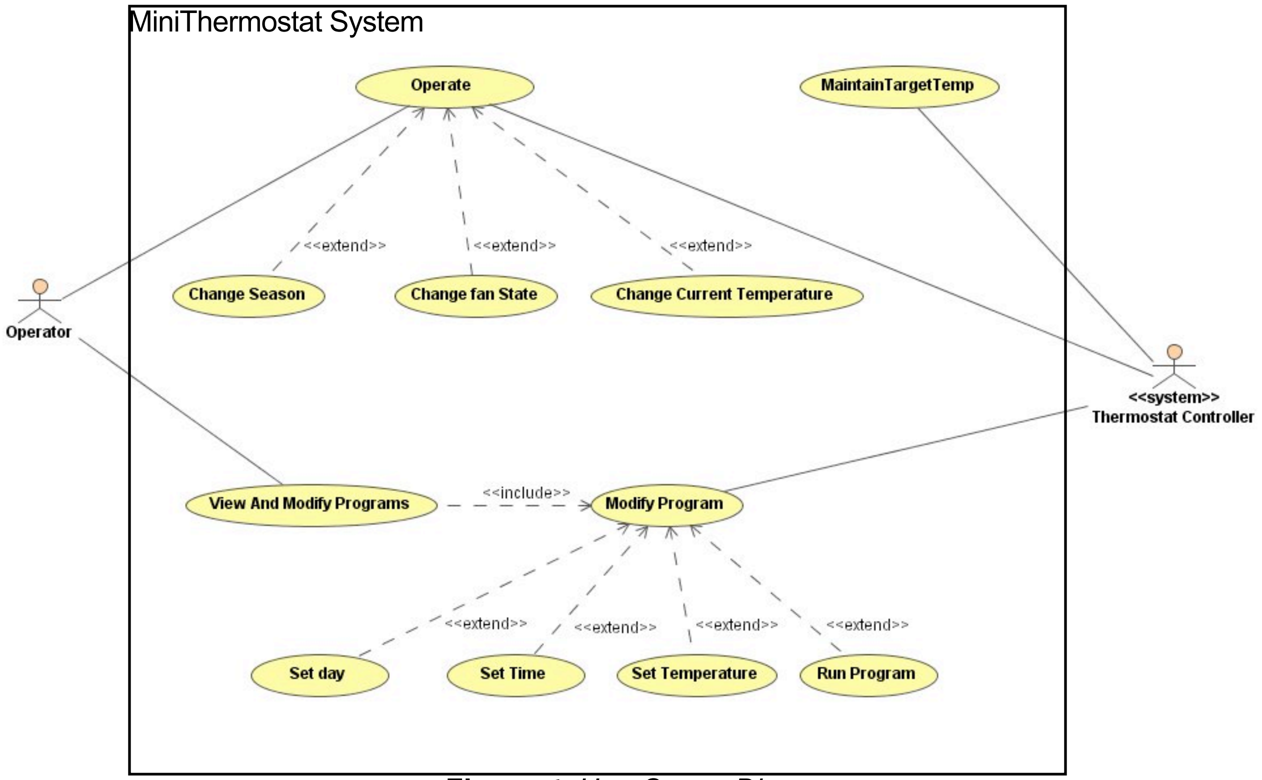


Figure 6: *Use Cases Diagram*

The Operator, shall be able to perform two main system functions: Operate the system by changing the different system states and program MiniThemostat with eight different programs. These functions are illustrated through the depicted above use cases.

Operate use case *extends* to 3 additional use cases, which all correspond to the functionality provided through UI01.

View and Modify Programs use case *includes* the use case of modifying one individual program.

Modify Program use case *extends* to 4 additional use cases which model the individual options provided to the operator in UI02.

<<system>> Thermostat Controller interacts directly with power relays for the Furnace, Air Conditioner and the Fan. As depicted in the figure, use cases **Operate** and **Modify Program** are directly associated with this actor. The actual control of the three appliances is represented by the one use case of **MaintainTargetTemp**. This one use case *includes* more use cases which are not depicted in this diagram, for example, the use case of obtaining the current temperature or operating one of the appliances. All of these use cases are implied by the one discussed above.

Detailed use cases descriptions are further provided in MiniThermostat Test Plan document.

4 Other Non-Functional Requirements

4.1 Performance Requirements

The system shall respond to each user input within 2 seconds. In the programming phase the system shall allow the user to switch quickly between the different programs, which in turn, will require a fast access to the hard drive. When communicating with the ACB, the ACB might not respond immediately, or the network may be overloaded, in such extreme case the operator shall be informed of the problem and shown a progress bar with estimated time to indicate when his command will be executed. There are no other performance requirements.

4.2 Safety and Security Requirements

The system shall monitor continuously the temperature in the area not allowing it to drop below MIN_TEMP or rise above MAX_TEMP. At no circumstances, shall the operator be allowed to modify the target temperature to exceed these values. From the security perspective, there are no specific security requirements other than those generally governing the use of operator's computer, e.g. Windows/Network logon, etc....

4.3 Software Quality Attributes

Listed below are some of the more important Software Quality Attributes, which have been identified as essential to this project.

4.3.1 Reliability

The system shall never crash, or hang, other than as the result of an operating system error. The system shall provide graceful degradation in the face of network delays.

4.3.2 Maintainability

All code shall be fully documented. Each function shall be commented with pre- and post-conditions. All program files shall include comments concerning authorship and date of last change. Finally, the code shall be modular to permit future modifications.

4.3.3 Usability

No specific training should be necessary on the part of the Operators. The user interface shall be easy to understand and the operator should be able to start working with the software immediately.

4.3.4 Portability

The software shall be designed to run on a Microsoft Windows platform. It shall be portable in a sense that will allow it to be used on any Windows version, up to and including Windows XP.

Appendix A – Data Dictionary

Variable/ Constant	Description	Triggering/Related Requirements	Values
MIN_TEMP	Minimum allowed temperature	REQ 3.4	5^0C
MAX_TEMP	Maximum allowed temperature	REQ 3.3	35^0C
THRESH	Maximum allowed difference between the target temperature and current temperature.	REQ 4	3^0C
v_MainState	System variable, represents the main states of the Thermostat System.	REQ 1 REQ 1.5 REQ 2	*s_Program* *s_Operate* *s_Exit*
v_Season	System variable, represents the season switch states.	REQ 3.1	*s_Heat* *s_Cool* *s_Off*
v_Fan	System variable, represents the fan states.	REQ 3.2	*s_FanAuto* *s_FanOn*
v_curr_Temp	System variable. At all times holds the current temperature in the monitored area, as read from the sensors.	REQ 4	N/A
v_target_Temp	System variable. At all times holds the target temperature. Can be modified directly by the operator. If not modified, receives values from the user specified programs.	REQ 3.4 REQ 3.3	>=MIN_TEMP <=MAX_TEMP
i_Time	Input variable. Used to read the user specified program time in UI02.	REQ 1.1 REQ 1.2	>=0000 <=2359
i_WeekDay	Input variable. Used to read the user specified program weekday in UI02.	REQ 1.1 REQ 1.2	{MONDAY, TUESDAY,… SUNDAY}
i_Temp	Input variable. Used to read the user specified program temperature in UI02.	REQ 1.1 REQ 1.2	>=MIN_TEMP <=MAX_TEMP
idle_Time	System variable. Used only when the operator switches to UI02 and counts the idle time.	REQ 1	>= 0

Table 1: *Data Dictionary*