

# **System documentation of a heating control system**

From a work of  
Marco Heide, August, 1997  
Wolfgang Wagenbichler, August, 1999

AG software engineering  
Computer department  
Kaiserslautern University

## **Abstract**

A heating control system monitors and control temperature in a building composed of several rooms. It is a good example of a software system. The related documentation is divided in three parts: a problem-solution documentation, software-system documentation and software-component documentation. This document refers to the first part.

**Part I**  
**Problem solution documentation**

# Summary

Part 1	Problem solution documentation .....	5
1.1	Organizational informations .....	5
1.2	Problem's description .....	5
1.2.1	Problem .....	5
1.2.2	Expected functionalities .....	5
1.2.3	Environment characteristics .....	6
1.3	Requirements / specifications .....	6
1.3.1	User requirements / specifications .....	7
1.3.2	Developer's Requirements/Specifications .....	14
1.3.3	Traceabilitymatrix: user requirements <-> class diagrams .....	49

# Part 1 Problem solution documentation

## 1.1 Organizational informations

This document has been produced, within the context of software engineering courses at the University of Kaiserslautern by Marco Heide, Andreas Schank, Angela Schmidt and Axel Seck, then reworked by Christiane Differding and Matthias Gutheil. It has been translated in english and adapted by Mario Negro Ponzi of Politecnico di Torino.

## 1.2 Problem's description

The following problem description is an implementation and adjustment of a house control system. A house is composed by many subsystems (e.g. rooms or devices as heaters), that are connected with doors and tubes. It is also connected with the outside world with doors and windows. A house is also made of a texture of subsystems that strictly interact with each other. A software system should be developed that controls and manage this complex system. It should control weather conditions, an alarm, and safety conditions (e.g. critical heating statuses, weather effects).

Below gets described the idea of this system. A user interface for getting and setting room parameters to the system is an integral part of the system but it will not be described here, representing a successive evolution.

### 1.2.1 Problem

Up until today the temperature of every room is regulated by a thermostat in the heater. It can be set a target temperature for each room. An automatic management of the temperature is based only on heaters. A window also is useful to control temperature in a single room. But it should be opened or closed by someone. Water in heaters has a fixed temperature set by a boiler.

Problems in temperature management are:

1. Heater's temperature cannot be differently set basing on heating demand or time. There is no regulation linked with people presence
2. Thermal regulation is limited to heating. The use of windows to cool down is based on user intervention
3. There is no optimization of boiler temperature. Both if there is a too low temperature or a too high one, the boiler's temperature is set to the maximum value. That produces an energy loss.

The above control system is not satisfying both because of energy loss and because it is not comfortable.

### 1.2.2 Expected functionalities

The user wants to be able to individually set temperature for each room using a terminal. So he should be able to provide described parameters.

First it should be possible to set a temperature that could be maintained when there is someone in the room. In doing so the system should not react immediately when someone comes in. It should be provided a time span that indicates for how long a person should stay in the room before the system takes control. At this point the user should be able to set the time span in which the desired temperature should be reached. The provided time span can be maintained heating up heaters for a small period of time over the desired temperature.

Also the user should be able to set a minimum temperature that is acceptable when there is no one in the room. In this case, too, a time span should be provided, after everyone has left the room, before the system takes control.

There should be also default values for every parameter that should be always be available.

Window, too, should be controlled by the system. If a window has been completely opened by a user, the system should maintain only the minimal temperature. If the window is open and it starts raining the system should put it in tilt position. More, the windows should be usable by the system to cool down when the outside temperature is cooler, by tilting it.

### **1.2.3 Environment characteristics**

Below are described the characteristics of the environment inside which the system will operate.

#### **1.2.3.1 House**

The house is composed by the following rooms:

- living room (40sqm)
- eating room (15sqm)
- sleeping room (15sqm)
- children room (20sqm)
- visitors room (15sqm)
- work room (10sqm)
- kitchen (15sqm)
- bathroom (5sqm)
- visitors bathroom (5sqm)
- hall (10sqm)

The house has, too:

- an outside temperature sensor
- a rain switch (closed if it rains)

#### **1.2.3.2 Room**

Each room is so configured (as a result of the whole system concept):

- 1 window with two switches and two actuators. A switch gets closed when the window is completely open (open when the window is open / NC = normally closed). The other when is tilted (open when the window is tilted / NC = normally closed). When the window is completely open the tilt switch is closed. An actuator swings the window while the other tilts it.
- 1 heater (hot water) with a thermostat
- 1 presence switch (closed when someone is in the room)
- 1 room temperature sensor

#### **1.2.3.3 Centralized heating room**

The control of the heating system is an already deployed autonomous subsystem. The maximum temperature can be set on the user interface.

The heating pump is controlled with impulses and it is so robust that there is no loss in the way between boiler and heaters.

### **1.3 Requirements / specifications**

Below are depicted system requirements. They will be divided in User and Developer requirements. As notation UML will be used. The UML diagrams to describe requirements will be ordered in User-Requirements and Developer-Requirements. The ordering follows the comprehensibility degree of the model from the point of view of the user. To user requirements belong also Use Cases and Scenarios. The rest of the model is in the Developer requirements described in chapter 1.3.2. There will be also a modeling of dynamic views.

### 1.3.1 User requirements / specifications

Below will be described functional, not functional and inverse user Requirements. Then user's conceptual choices will be described. At the end will be depicted Use Case diagrams and Scenarios that are based on User Requirements.

#### 1.3.1.1 Functional User Requirements

Functional User Requirements are originated by problem description (chapter 1.2) and from opinions that are collected in interviews with the user.

UR-F	Requirement's description
Temp-UR-F 1	The user shall be able to set <i>StandardPresenceTemp</i> in °C
Temp-UR-F 2	The user shall be able to set <i>StandardAwayTemp</i> in °C
Temp-UR-F 3	The user shall be able to set a time span <i>StandardHeatingTime</i>
Temp-UR-F 4	The user shall be able to set a time span <i>StandardAwayTime</i> in minutes
Temp-UR-F 5	The user shall be able to set a time span <i>StandardInsideTime</i> in minutes
Temp-UR-F 6	The temperature of Temp-UR-F 1 shall be maintained when there is someone in the room and the <i>StandardInsideTime</i> from Temp-UR-F 5 has elapsed
Temp-UR-F 7	The temperature of Temp-UR-F 2 should be maintained in the room when the room is empty and <i>StandardAwayTime</i> from Temp-UR-F 4 has elapsed
Temp-UR-F 8	The time span from Temp-UR-F 3 indicates after how many minutes the <i>StandardPresenceTemp</i> from Temp-UR-F 1 after the entrance of a person in the room shall be reached.
Temp-UR-F 9	The time span from Temp-UR-F 4 indicates for how many minutes the <i>StandardPresenceTemp</i> from Temp-UR-F 1 shall be maintained after the last person has left the room
Temp-UR-F 10	The time span from Temp-UR-F 5 indicates for how long shall someone stay in the room for the system to get in control
Temp-UR-F 11	The <i>StandardHeatingTime</i> from Temp-UR-F 3 must be at least 5 minutes longer than the <i>StandardInsideTime</i> from Temp-UR-F 5
Temp-UR-F 12	For each value from Temp-UR-F 1, Temp-UR-F 2, Temp-UR-F 3, Temp-UR-F 4, Temp-UR-F 5 shall exist default values.
Temp-UR-F 13	If a window gets opened the system maintains the room temperature at <i>StandardAwayTemp</i> , whether or not there is someone in the room.
Temp-UR-F 14	If it starts raining with the window open, the system will set the window in a rain secure position.
Temp-UR-F 15	When the temperature of a room with someone inside exceeds that from Temp-UR-F 1 and the outer temperature is lower than the room temperature, the window in this room gets tilted
Temp-UR-F 16	The control of the temperature is admitted with an error of +/-1 °C
Temp-UR-F 17	The system is planned for the room described on page 6
Temp-UR-F 18	The temperature of the boiler shall be set to the maximum required by any heater, not to waste energy.

Table 1

#### 1.3.1.2 Non-functional user requirements

The system should comply also with the following requirements.

UR-NF	Requirement's description
UR-NF 1	The system should be easily changeable.

Table 2

### 1.3.1.3 Inverse user requirements

Following situations should be avoided.

UR-Inv	Requirement's description
UR-Inv 1	When there is nobody in the room and the room is hotter than target temperature, the window does not get tilted to cool.
UR-Inv 2	If the user opens the window while it rains, the system shall not tilt it to the rain secure position.

**Table 3**

### 1.3.1.4 Design decisions

Following decisions have been taken.

DD	Requirement's description
DD 1	The system should be object oriented.

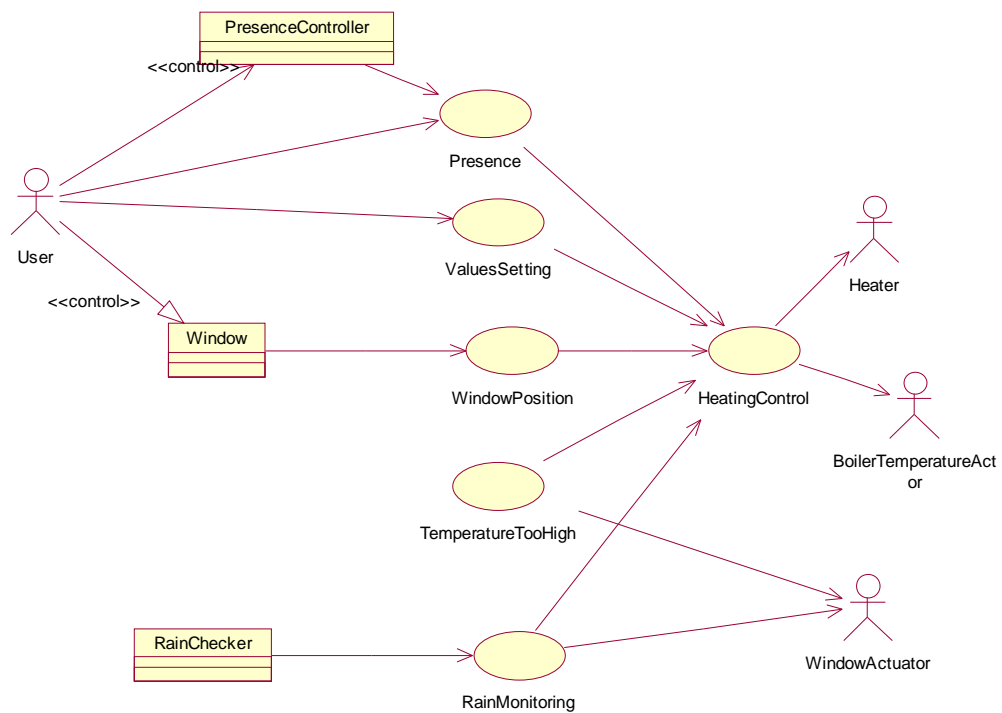
**Table 4**

### 1.3.1.5 User requirements analysis

From UML describing diagrams get ordered Use Cases and Scenarios for user requirements, as noted in the introduction of chapter 1.3.1.

#### 1.3.1.5.1 Use case diagram

Use cases typically depict interactions between external actors and the system from the point of view of the user. The interaction among actors and the system starts with initial events triggered by the actor on the system and ends when the logical reaction of the system has finished.



**Use Case: Presence**

The activating actor of this Use Case is the PresenceController together with the PresenceSwitch. The PresenceController activates a reaction from the system notifying presence or absence of a person in the room

This Use Case will be shown in the following scenarios:

- Scenario 3
- Scenario 2
- Scenario 4

**Use Case: WindowPosition**

The activating actor of this Use Case is the Window that is opened or closed by the user. It activates a reaction of the system notifying a changed position.

This Use Case will be shown in the following scenario:

- Scenario 5

**Use Case: HeatingControl**

This Use Case describes the recalculation of the temperature of the heater (to heat, maintain or cool). At the same time the new temperature of the boiler is notified.

This Use Case will be shown in the following scenarios:

- Scenario 10
- Scenario 11
- Scenario 9

**Use Case: RainMonitoring**

The activating actor of this Use Case is the RainChecker together with the RainSwitch. The RainChecker activates a reaction of the system notifying when it's raining on the house.

This Use Case will be shown in the following Scenario:

- Scenario 6

***Use Case: TemperatureTooHigh***

The activating actor of this Use Case is the RoomTemperatureController. It activates a reaction of the system notifying that the room temperature is too high.

This Use Case will be shown in the following scenario:

- Scenario 7

***Use Case: ValuesSetting***

The user sets various values. The activating actor is the user.

This Use Case will be shown in the following scenarios:

- Scenario 8
- Scenario 1

***1.3.1.5.2 Scenarios for Use Cases***

The following scenarios describe the system in common situations and realize the Use Cases. So they must show every functional and inverse user requirement.

Other notes (to the following table):

- (-) after a User Requirement indicates that a specific precondition (e.g. a time span elapsing) has not been met
- (+) after a User Requirement indicates that a specific precondition has been met
- Without any other specific note, in the following scenarios those following default values should be considered:
  - o StandardPresenceTemp: 20 °C
  - o StandardAwayTemp: 15 °C
  - o StandardHeatingTime: 10 minutes
  - o StandardInsideTime: 5 minutes
  - o StandardAwayTime: 15 minutes

#	Step	Scenario's description	User requirement
Scenario 1	<i>Scenario for Use Case Value Setting.</i> <i>User changed StandardHeatingTime, StandardInsideTime or StandardPresenceTemp. The room will be heated up.</i>		
	1.	<b>Presence switch:</b> closed <b>Outside Temperature:</b> 21 °C <b>Window:</b> closed <b>Rain switch:</b> open <b>Target temperature:</b> 19 °C (StandardPresenceTemp) <b>Room temperature:</b> 18 °C	
	2.	The user sets StandardInsideTime to 5 minutes	Temp-UR-F5
	3.	The user sets StandardHeatingTime to 9 minutes	Temp-UR-F3
	4.	The system ignores new time set, because it is just 4 minutes over StandardInsideTime and throws an error message	Temp-UR-F11
	5.	The user sets StandardHeatingTime to 10 minutes	Temp-UR-F3
	6.	The user sets StandardAwayTemp to 16 °C	Temp-UR-F2
	7.	The user sets StandardAwayTime to 10 minutes	Temp-UR-F4
	8.	The user sets StandardPresenceTemp to 21 °C	Temp-UR-F1
	9.	The target temperature is set at 21 °C	
-> Use Case heating control			
Scenario 2	<i>Scenario for Use Case Presence</i> <i>The user leaves the room before the system activates</i>		
	1.	<b>Presence switch:</b> open <b>Outside temperature:</b> 21 °C <b>Window:</b> closed <b>Rain sensor:</b> open <b>Target temperature:</b> 15 °C (StandardAwayTemp) <b>Room temperature:</b> 18 °C	
	2.	A user enters in the room	
	3.	The user leaves the room before the StandardInsideTime has elapsed	Temp-UR-F10 (-)
4.	The system does not get controlled		
Scenario 3	<i>Scenario for Use Case Presence</i> <i>Target temperature is set from StandardPresenceTemp to StandardAwayTemp</i>		
	1.	<b>Presence switch:</b> closed <b>Outside temperature:</b> 16 °C <b>Window:</b> closed <b>Rain sensor:</b> open <b>Target temperature:</b> 20 °C (StandardPresenceTemp) <b>Room temperature:</b> 19 °C	
	2.	The user leaves the room, which becomes empty	
	3.	After StandardAwayTime the target temperature is set to StandardAwayTemp.	
-> Use Case Heating control			
4.	The window does not get opened because there is no one inside the room.		
Scenario 4	<i>Scenario for Use Case Presence</i> <i>There is only one user in the room. This user leaves for a short time the room so that it does not have any effect on the system behavior</i>		

	1.	<b>Presence switch:</b> closed <b>Outside temperature:</b> 19 °C <b>Window:</b> closed <b>Rain sensor:</b> open <b>Target temperature:</b> 20 °C (StandardPresenceTemp) <b>Room temperature:</b> 19 °C	
	2.	The use leaves the room, which becomes empty	
	3.	The user comes back in the room before StandardAwayTime has elapsed.	
	4.	The system does not take control	
Scenario 5	<i>Scenario for use Case Window Closing</i> <i>The window is open and a user is present. Target temperature is set to StandardAwayTemp</i>		
	1.	<b>Presence switch:</b> closed <b>Outside temperature:</b> 19 °C <b>Window:</b> closed <b>Rain sensor:</b> open <b>Target temperature:</b> 20 °C (StandardPresenceTemp) <b>Room temperature:</b> 20 °C	
	2.	The user opens the window	
	3.	Target temperature is set to StandardAwayTemp	Temp-UR-F13
	-> Use Case Heating Control		
Scenario 6	<i>Scenario for Use Case Rain Monitoring</i> <i>It rains with the window open</i>		
	1.	<b>Presence switch:</b> closed <b>Outside temperature:</b> 17 °C <b>Window:</b> completely open <b>Rain sensor:</b> open <b>Target temperature:</b> 15 °C (StandardAwayTemp) <b>Room temperature:</b> 18 °C	
	2.	It starts raining	
	3.	The system closes the window	Temp-UR-F14
	4.	Target temperature is set to StandardPresenceTemp	
	-> Use Case Heating Control		
	5.	The user opens again the window	
	6.	Since the window has been opened when it was already raining, it doesn't get closed by the system	Temp-UR-F14 UR-Inv2
	7.	Target temperature is set to StandardAwayTemp	Temp-UR-F13
-> Use Case Heating Control			
Scenario 7	<i>Scenario for Use Case Temperature too high</i> <i>The window gets controlled by the system to cool</i>		
	1.	<b>Presence switch:</b> closed <b>Outside temperature:</b> 17 °C <b>Window:</b> closed <b>Rain sensor:</b> open <b>Target temperature:</b> 20 °C (StandardPresenceTemp) <b>Room temperature:</b> 21 °C	
	2.	The room temperature raise to 22 °C and so is more that 1 °C over target temperature.	Temp-UR-F16
	-> Use Case Heating Control		
	3.	The system controls the window to cool down	Temp-UR-F15(+)
Scenario 8	<i>Scenario for Use Case Value Setting</i> <i>The user sets all values to default</i>		
	1.	The user sets StandardPresenceTemp to default	Temp-UR-F12

	2.	The user sets StandardAwayTemp to default	
	3.	The user sets StandardInsideTime to default	
	4.	The user sets StandardAwayTime to default	
	5.	The user sets StandardHeatingTime to default	
<i>Scenario for Use Case Heating Control</i>			
<i>Target Temperature of a room is reached</i>			
Scenario 9	1.	<b>Presence switch:</b> closed <b>Outside temperature:</b> 14 °C <b>Window:</b> closed <b>Rain sensor:</b> open <b>Target temperature:</b> 20 °C (StandardPresenceTemp) <b>Room temperature:</b> 18 °C	
	2.	Room temperature reach 19 °C	
	3.	Stop temperature has been reached, since room temperature is just 1 °C below target temperature.	Temp-UR-F6 Temp-UR-F16
	4.	Required new boiler temperature gets established	Temp-UR-F18
<i>Scenario for Use Case Heating Control</i>			
<i>The room must be heated up</i>			
Scenario 10	1.	<b>Presence switch:</b> closed <b>Outside temperature:</b> 23 °C <b>Window:</b> closed <b>Rain sensor:</b> open <b>Target temperature:</b> 20 °C (StandardPresenceTemp) <b>Room temperature:</b> 19 °C	
	2.	Room temperature drops to 18 °C	
	3.	Heating temperature gets calculated because room temperature has dropped below 1 °C under target temperature.	Temp-UR-F6 Temp-UR-F8 Temp-UR-F16
	4.	Heater temperature is set to the necessary value	
	5.	Required new boiler temperature gets established	Temp-UR-F18
<i>Scenario for Use Case Heating Control</i>			
<i>Room temperature is below target temperature</i>			
Scenario 11	1.	<b>Presence switch:</b> closed <b>Outside temperature:</b> 23 °C <b>Window:</b> closed <b>Rain sensor:</b> open <b>Target temperature:</b> 20 °C (StandardPresenceTemp) <b>Room temperature:</b> 21 °C	
	2.	Room temperature raises to 22 °C	
	3.	While the temperature of the room is more than 1 °C over target temperature, heater temperature is set to 0 °C	
	4.	Required new boiler temperature gets established	Temp-UR-F18
	5.	The window gets not controlled to cool down room temperature.	Temp-UR-F15(-)

### 1.3.2 Developer's Requirements/Specifications

Here follow UML diagrams to describe requirements. This comprehends a class diagram with Data-Dictionary, Sequence Diagrams and Collaboration Diagrams.

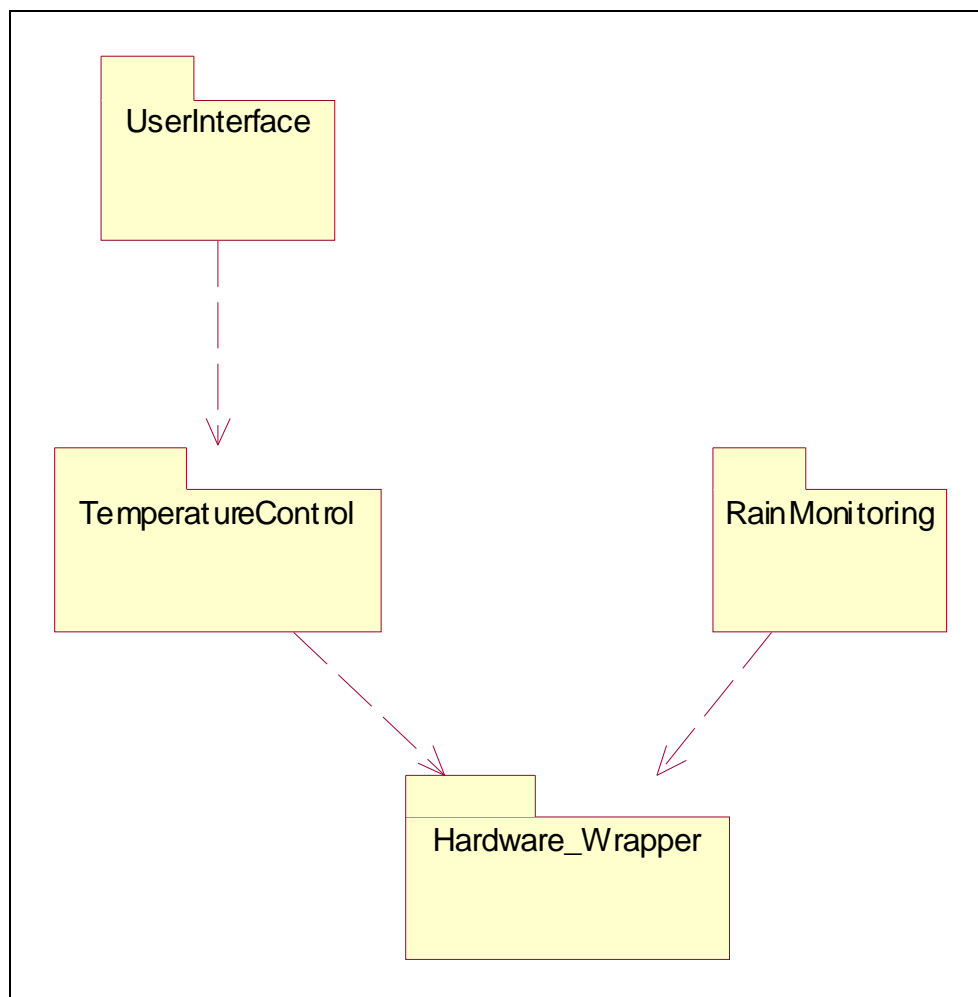
#### 1.3.2.1 UML class diagram

In class diagrams is described the static structure that has been described in the Data-Dictionary classes and in their relationships.

##### 1.3.2.1.1 Package structure

This diagram shows the partitioning of the system into subsystems (Packages). Different packages all have proper classes. So in a package aren't defined methods and attributes of a class.

#### Package diagram



#### *Description of each package*

Package: User\_Interface

This package comprehends classes that build up functionalities of the user interface.

Package: Hardware\_wrapper

Comprehends I/O classes that represents real Actors and Sensors

Package: Rain\_monitor

This package comprehends classes that are related to the monitoring of the rain.

Package: Temperature\_control

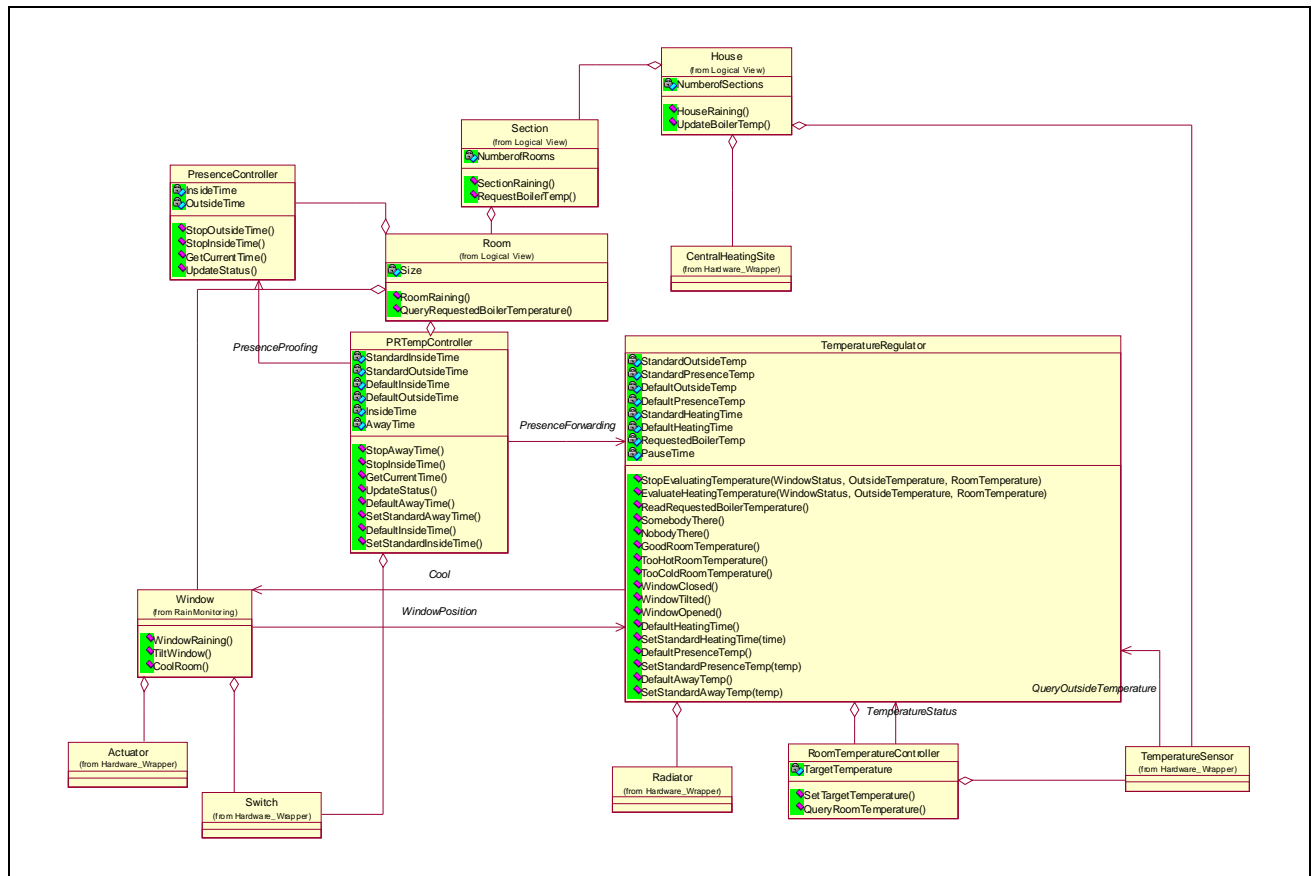
This package comprehends classes that contribute in controlling temperature.

Values that are set to control temperature can be set by the User\_Interface. Sensors and actors are activated on the Hardware\_wrapper.

### 1.3.2.1.2 Temperature control

This diagram shows all classes that participate in controlling temperature.

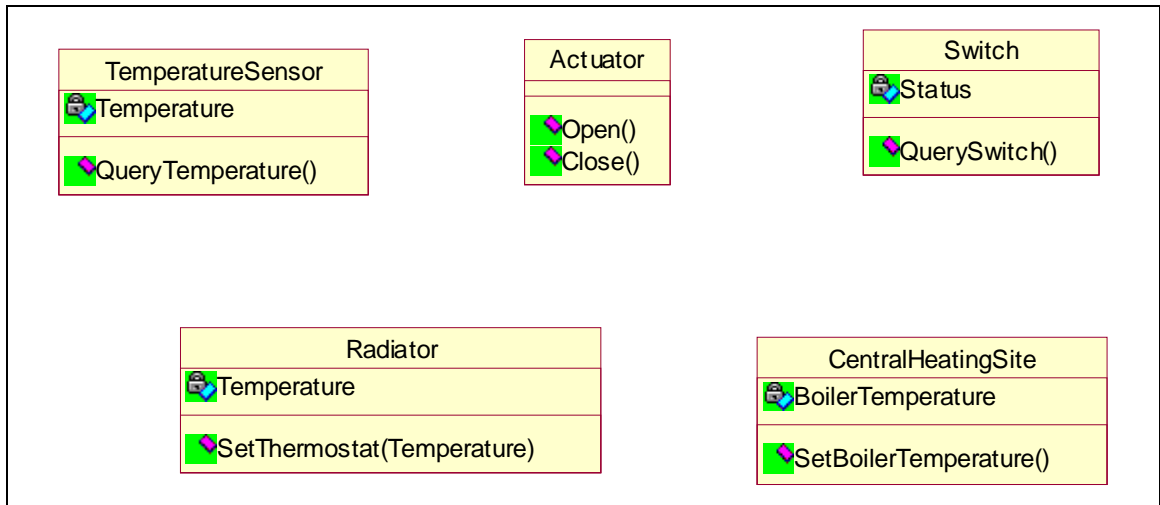
#### Class diagram



### 1.3.2.1.3 Hardware\_wrapper

This diagram shows all the classes that represent Sensors and Actors

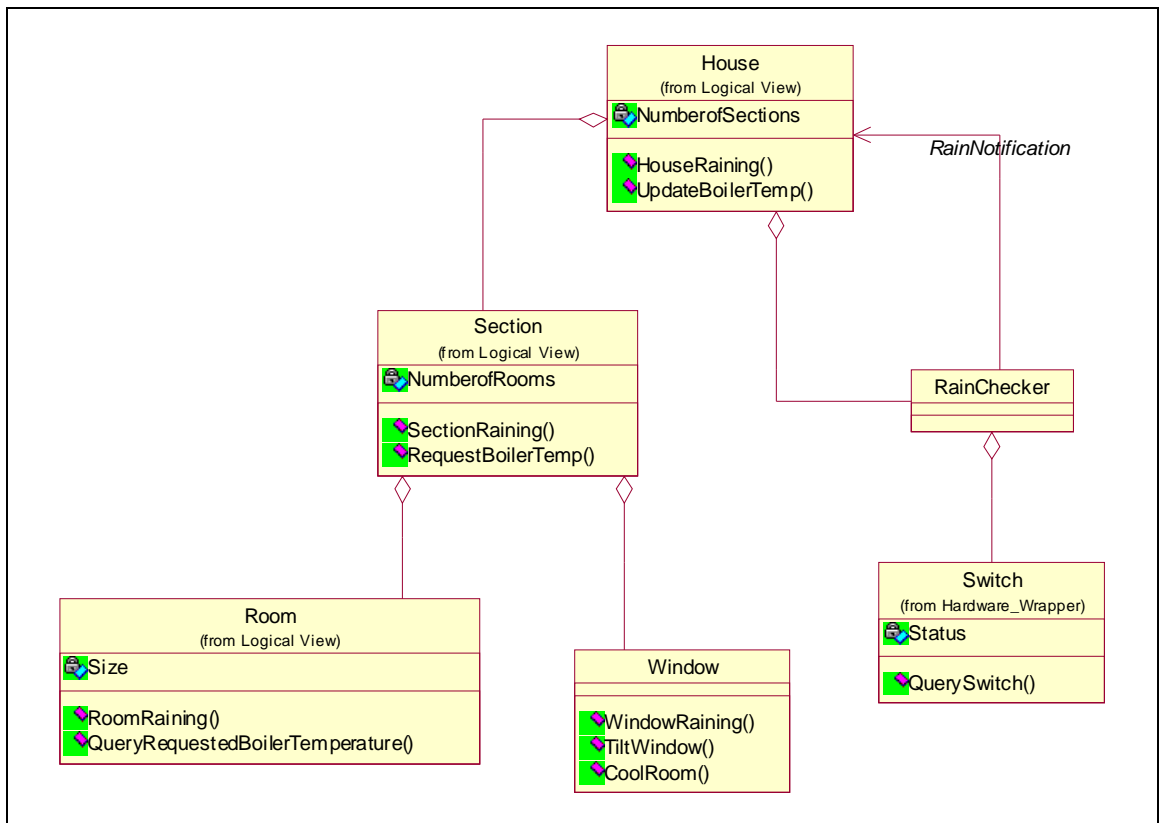
#### Class diagram



#### 1.3.2.1.4 Rain\_monitor

This diagram shows all the classes that are involved with the rain monitoring.

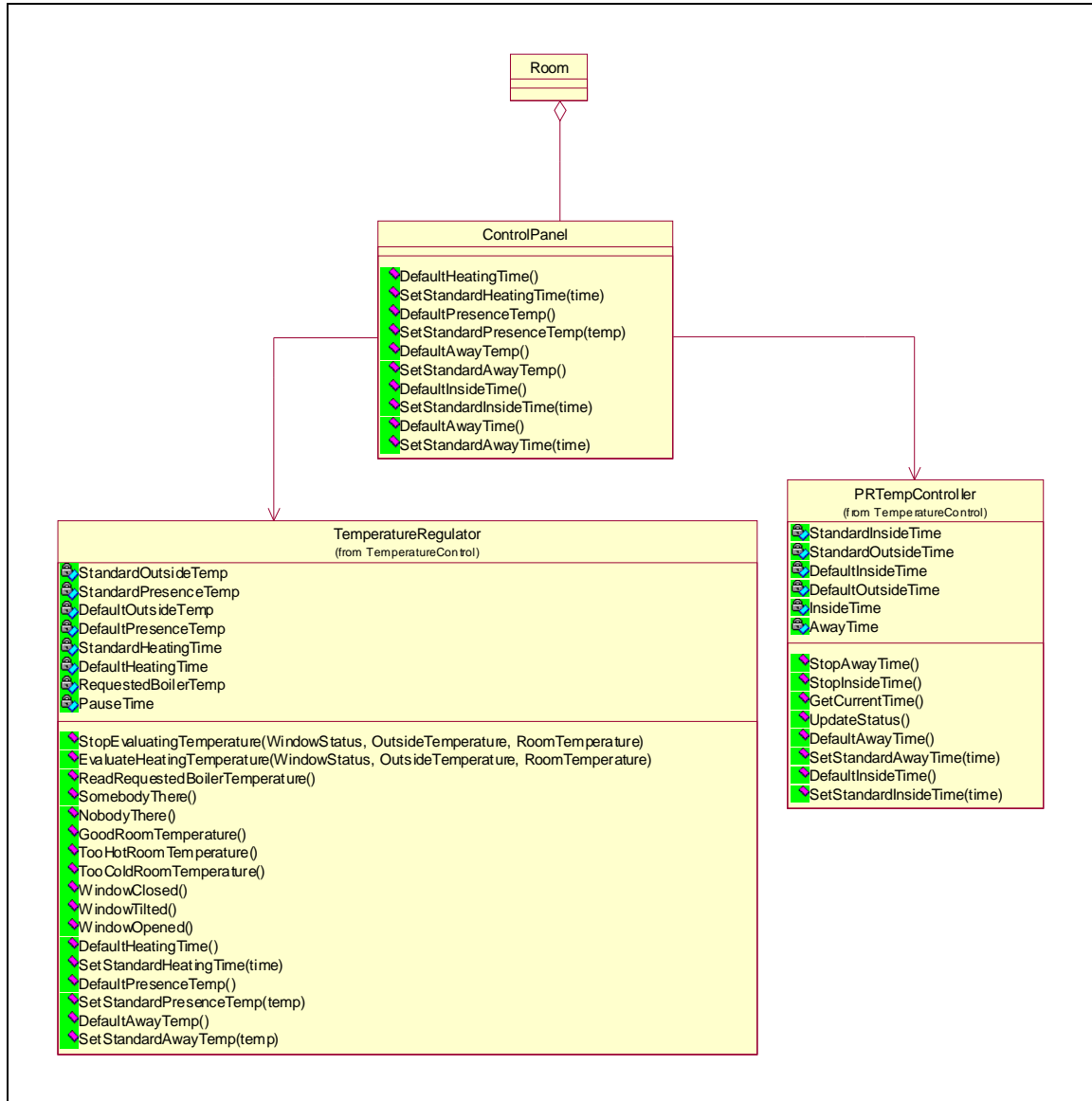
#### Class diagram



#### 1.3.2.1.5 User\_Interface

This diagram shows the user interface subsystem.

## Class diagram



### 1.3.2.2 Data-dictionary

Into the data-dictionary are described all the classes all classes of the UML-analysis together with their attributes, methods and relationships.

Object to be described	Attribute, Method, Relationship	Description
CentralHeatingSite	Handles rooms heating and can be set to the BoilerTemperature	Temperature set
	BoilerTemperature	Temperature set
	SetBoilerTemperature()	Sets the temperature of the boiler in °C
Switch	Represents real switches. Available states: open and closed.	
	Status	Switch status
	QueryStatus()	Returns the status of the Switch

	The switch class is composed by the following instances:	
	PresenceSwitch	Represents the switch, which status changes whether there is a person in the room or not
	TiltSwitch	Represents the switch, which status indicates whether the window is closed (open status) or tilt (closed status).
	OpenSwitch	Represents the switch which status indicates whether the window is completely open (open status) or not (closed status).
	RainSwitch	Represents the switch which status indicates whether it rains (closed status) or not (open status)
ControlPanel	Represents the device the user can use to set values	
	setDefaultAwayTemp()	Calls the according method in the temperature regulator
	setStandardAwayTemp()	Calls the according method in the temperature regulator
	setDefaultPresenceTemp()	Calls the according method in the temperature regulator
	setStandardPresenceTemp()	Calls the according method in the temperature regulator
	setDefaultHeatingTime()	Calls the according method in the temperature regulator
	setStandardHeatingTime()	Calls the according method in the temperature regulator
	setDefaultInsideTime()	Calls the according method in the temperature regulator
	setStandardInsideTime()	Calls the according method in the temperature regulator
	setDefaultOutsideTime()	Calls the according method in the temperature regulator
	setStandardOutsideTime()	Calls the according method in the temperature regulator
	House	Buildings are composed by various parts (sections), by a central heating site (CentralHeatingSite), by a temperature sensor (TemperatureSensor) for the outside temperature, and by a rain sensor (RainChecker). The class House queries all the sections for the requested temperature of the boiler, acquires the maximum and sets this to the boiler of the CentralHeatingSite.
SectionNumber		Number of sections
UpdateBoilerTemperature()		Gets the requested temperature querying every section and notifies this to the heating structure.
RainHouse()		Notifies rain to all sections
Actuator	Represents the physical actuator	
	Open()	
	Close()	
	The actuator class holds the following instances	

	TiltActuator	Represents the actuator that tilts or close the window
	SwingActuator	Represents the actuator that completely open or closes the window
PRTempController	Constantly queries the PresenceController and provides presence status to the temperature control complying with StandardInsideTime and StandardAwayTime	
	DefaultInsideTime	Default value for the inside time interval in minutes.
	StandardInsideTime	Time interval in minutes, during which a person should be in the room before the system starts handling
	DefaultAwayTime	Default value for the outside time interval in minutes.
	StandardAwayTime	Time interval in minutes, during which the temperature should be kept to StandardAwayTemp
	DefaultInsideTime()	Sets the attribute to the default value
	StandardInsideTime()	Sets the attribute to the value given in minutes.
	DefaultAwayTime()	Sets the attribute to the default value
	StandardAwayTime()	Sets the attribute to the value given in minutes.
PresenceController	Controls the presence switch. Continuously queries the status of the switch and reports the arrival or the leaving of a person.	
	InsideTime	Instant in which the user has entered
	AwayTime	Instant in which the user left
	StopInsideTimer()	Keeps instants when a person has entered the room
	StopAwayTimer()	Keeps instants when the last person has left the room
	StopTime()	Stops current time
	UpdateStatus()	Returns values of presence and intervals for how long a person has been in the room.
Heater	Controls the heating of a room and can bet set to a variable temperature.	
	Temperature	The temperature set
	SetThermostat()	Sets the temperature of the heater in °C
RainChecker	Handles the RainSwitch. Continuously queries the value of the rain switch for raining and notifies to the house any change.	
Room	Room inside which the temperature is controlled. Comprehends TemperatureRegulator, PRTempController, PresenceController and Window.	
	Size	Size of the room in square meters

	QueryBoilerTemp()	Ask the TemperatureRegulator for boiler temperature and returns
	RainRoom()	Notifies rain on the window so that RainWindow() gets called
RoomTemperatureController	<i>Monitor temperature sensor Continuously queries the value of the temperature sensor and survey that the temperature in the room isn't too high (RoomTempTooHigh) or too low (RoomTempTooLow) with respect to the target temperature.</i>	
	TargetTemperature	Desired temperature in the room
	QueryRoomTemperature()	Queries the room temperature and returns the value
	SetTargetTemperature()	Sets the attribute TargetTemperature in °C
Section	<i>A section consists of several rooms</i>	
	RoomNumber	Number of rooms
	QueryBoilerTemp()	Ask to all rooms for boiler temperature and sets the maximum as a result for the house.
	RainSection()	Notice rain on every room
TemperatureRegulator	<i>Controls the temperature in current room Interact with the Heater and RoomTemperatureController</i>	
	DefaultAwayTemp	Default value for temperature when the room is empty, in °C
	StandardAwayTemp	Temperature in °C to be reached when the room is empty
	DefaultPresenceTemp	Default value for temperature where the room is not empty, in °C
	StandardPresenceTemp	Temperature in °C to be reached when the room is not empty
	BoilerTemperature	Boiler temperature for the room
	WaitTime	Calculates: StandardHeatingTime minus StandardInsideTime and returns how many minutes are before the desired temperature can be reached
	DefaultHeatingTime	Default value in minutes for HeatingTime
	StandardHeatingTime	Time interval indicating after how many minutes should the temperature be reached after a person has entered the room. This time interval must be smaller than StandardInsideTime

	SetDefaultAwayTemp()	Set the attribute StandardAwayTime to the value DefaultAwayTime
	SetStandardAwayTemp()	Set the attribute StandardAwayTime to the new value, in °C
	SetDefaultPresenceTemp()	Set the attribute StandardPresenceTime to the value DefaultPresenceTime
	SetStandardPresenceTemp()	Set the attribute StandardPresenceTime to the new value, in °C
	GetBoilerTemp()	Returns boiler temperature
	WindowTilted()	Notifies the TemperatureRegulator that the window is tilted
	WindowClosed()	Notifies the TemperatureRegulator that the window is closed
	WindowOpened()	Notifies the TemperatureRegulator that the window is open
	EvaluateStopTemp()	Calculates heater temperature so that the target temperature can be stopped
	EvaluateHeatingTemp()	Calculates heater temperature in order to reach target temperature
	SomebodyThere()	Notifies the TemperatureRegulator that there is a person in the room
	NobodyThere()	Notifies the TemperatureRegulator that there is no one in the room
	SetDefaultHeatingTime()	Sets the attribute StandardHetingTime to DefaultHeatingTime value
	SetStandardHeatingTime()	Sets the attribute StandardHetingTime to new value, in seconds
	GoodRoomTemp()	Notifies the TemperatureRegulator that the TargetTemperature has been reached
	TooHighRoomTemp()	Notifies the TemperatureRegulator that the room Temperature is too high
	TooLowRoomTemp()	Notifies the TemperatureRegulator that the room Temperature is too low
TemperatureSensor	<i>Represents the real temperature sensor</i>	
	Temperature	Current temperature, in °C

	QueryTemperature()	Returns sensor's current temperature
Window	<i>Controls tilt sensor, open sensor, swing actuator and tilt actuator. The window can be open, tilt or closed.</i>	
	TiltWindow()	Tilts the window using tilt and swing actuators
	RainWindow()	Tilts the window when it's raining using the method TiltWindow()
	CoolRoom()	Tilts the window to cool using TiltWindow()

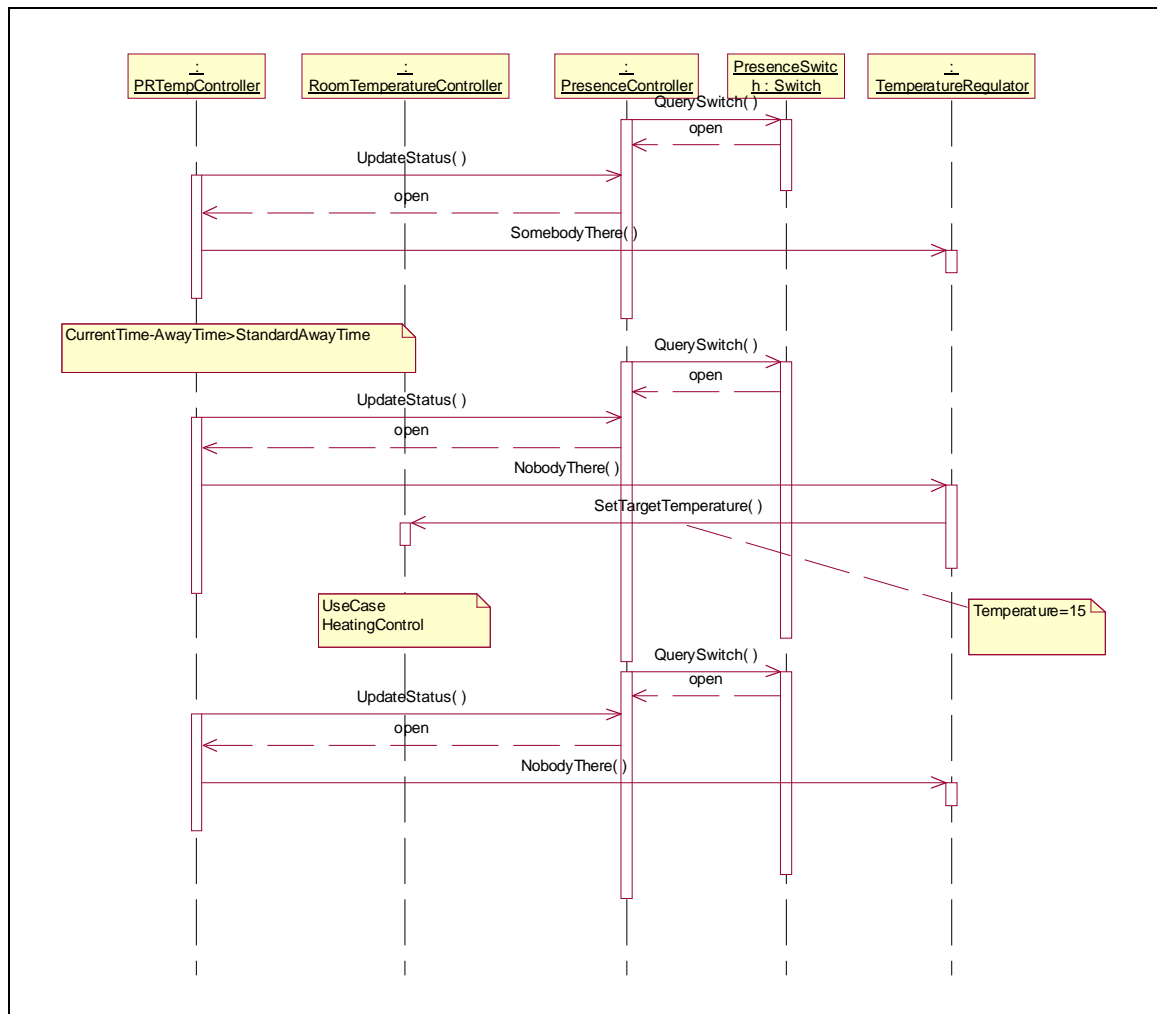
**Table 5**

### 1.3.2.3 UML Sequence Diagrams

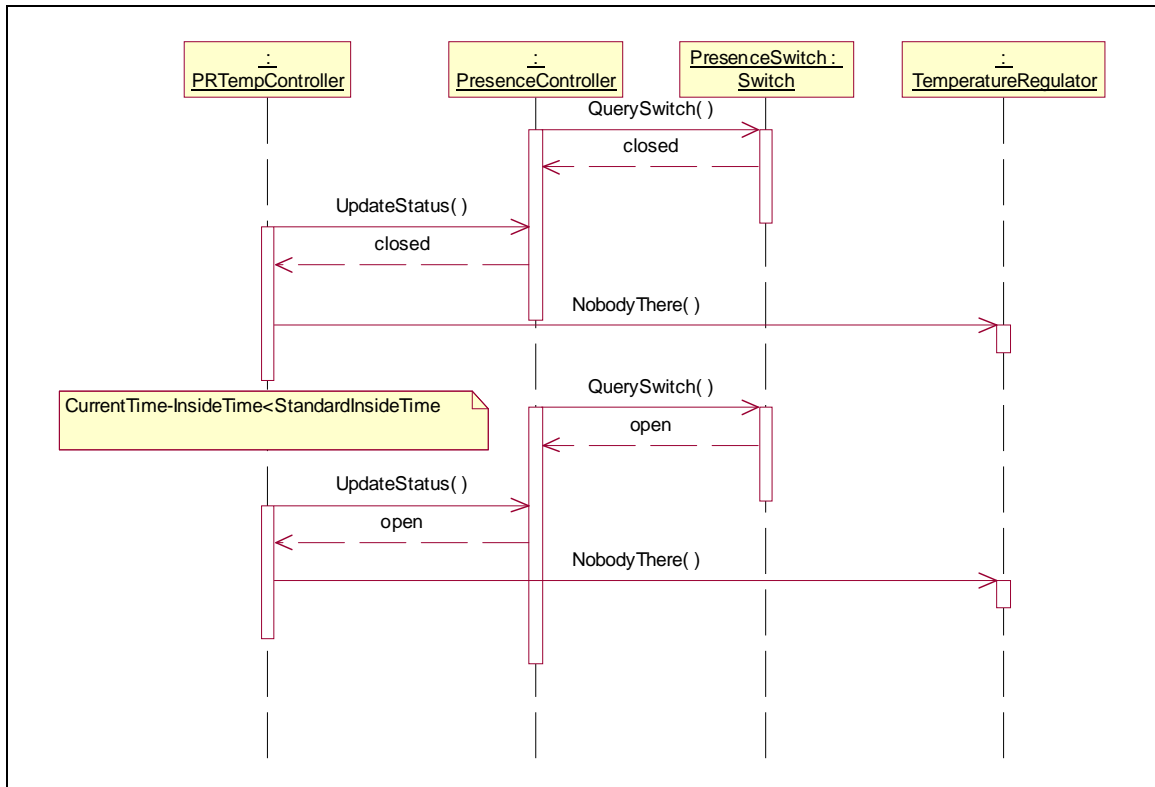
Use Cases of this system are detailed below with sequence diagrams.

#### 1.3.2.3.1 Sequence diagram for Use Case Presence

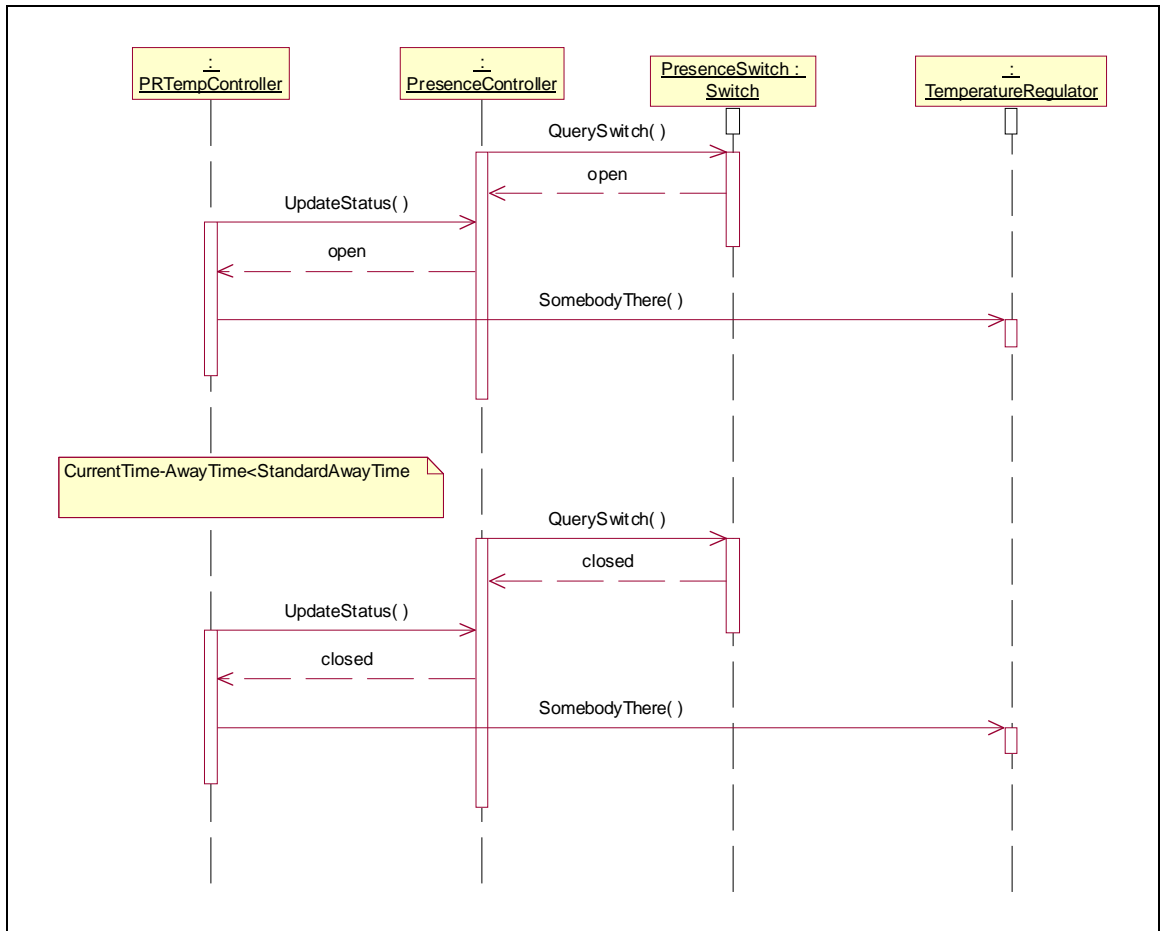
Sequence diagram for scenario 3:



Sequence diagram for scenario 2:

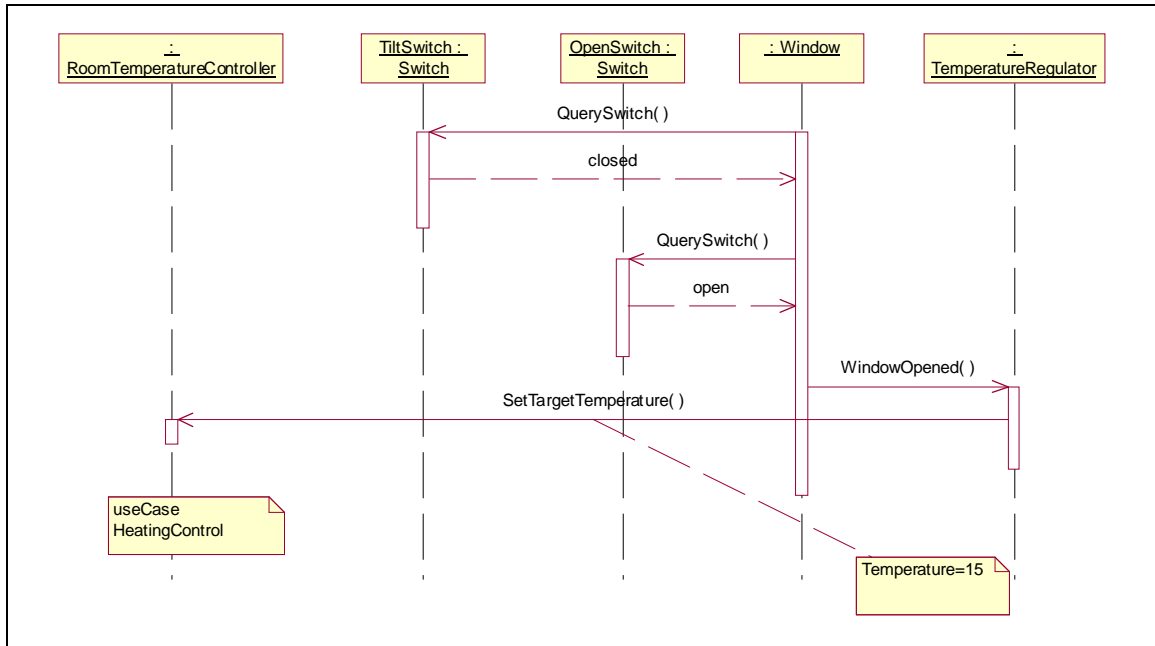


Sequence diagram for scenario 4:



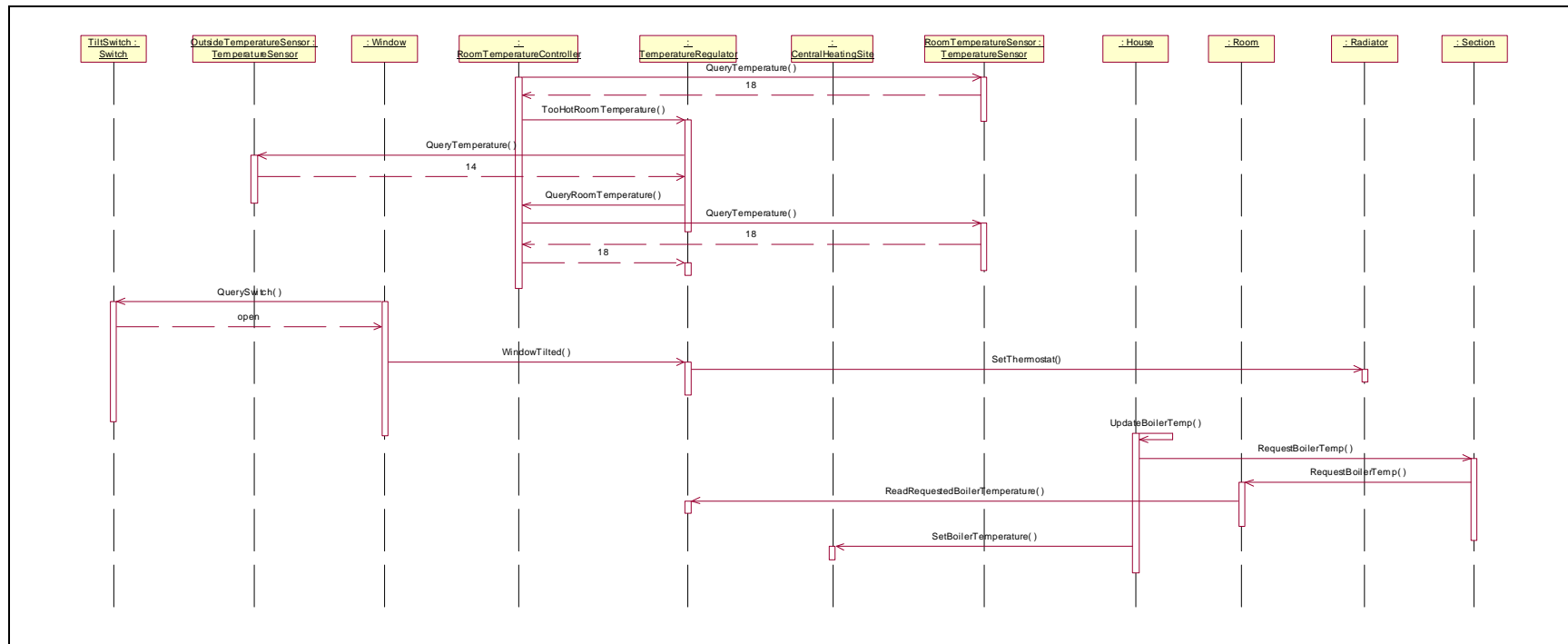
**1.3.2.3.2 Sequence diagrams for Use Case WindowPosition**

Sequence diagram for scenario 5:

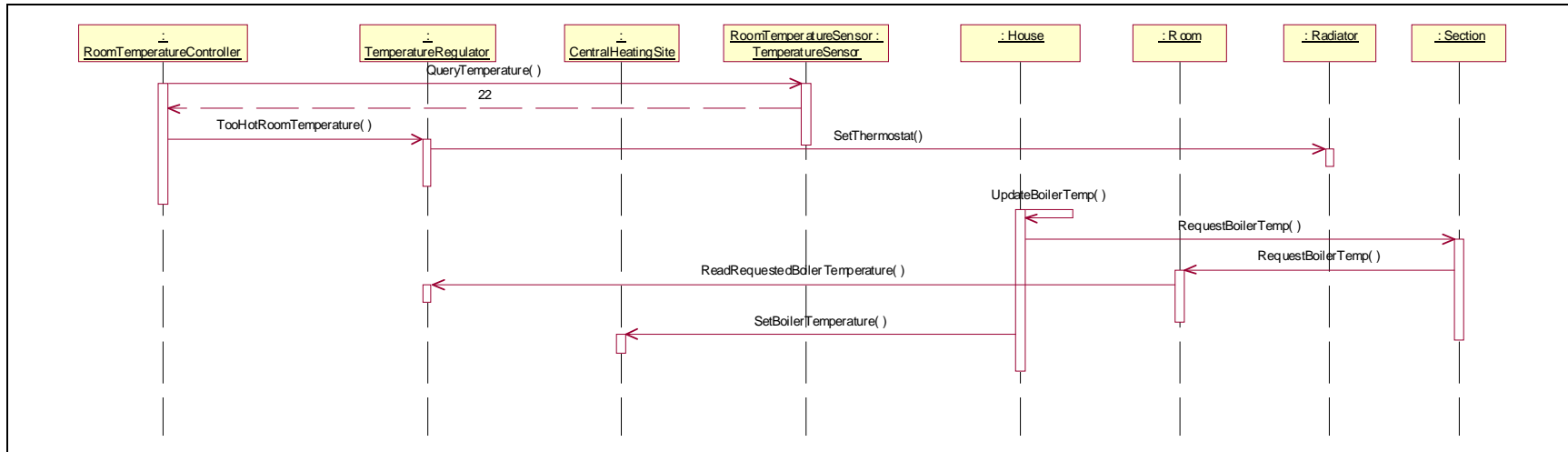


**1.3.2.3.3 Sequence Diagram for Use Case HeatingControl**

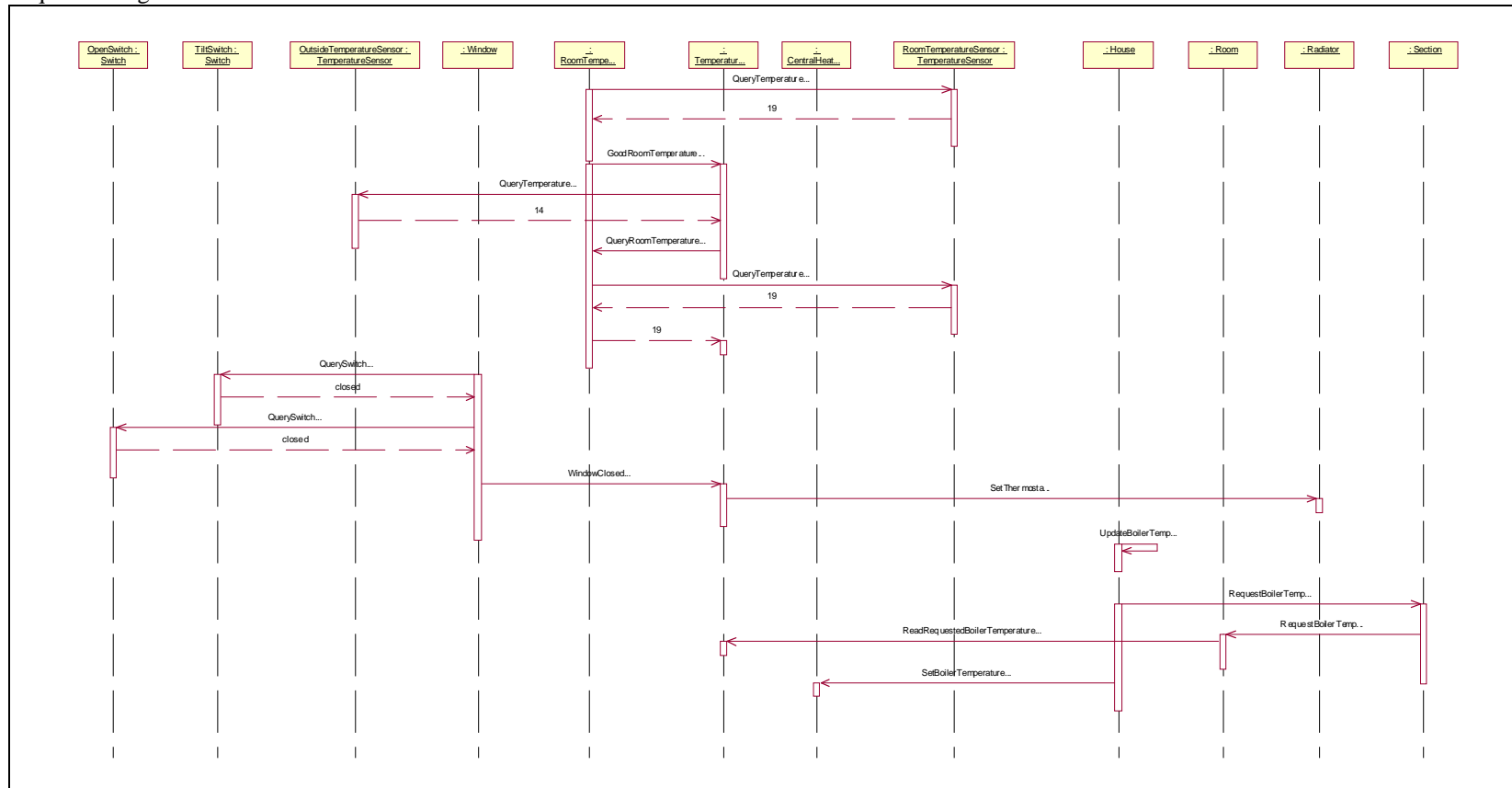
Sequence diagram for scenario 10:



Sequence diagram for scenario 11:



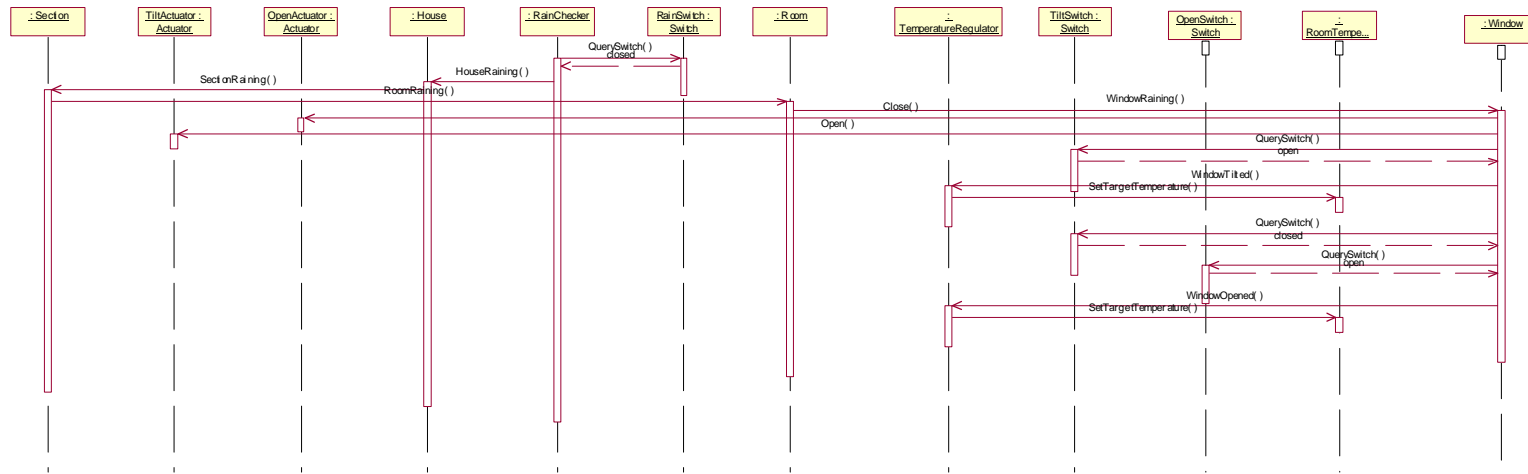
Sequence diagram for scenario 9:





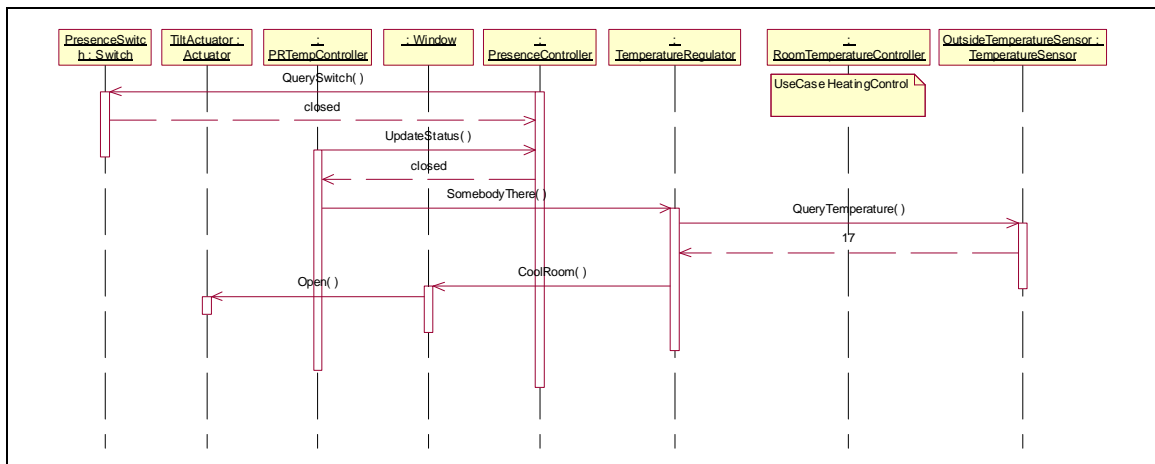
#### ***1.3.2.3.4 Sequence Diagrams for Use Case RainMonitoring***

Sequence diagram for scenario 6:



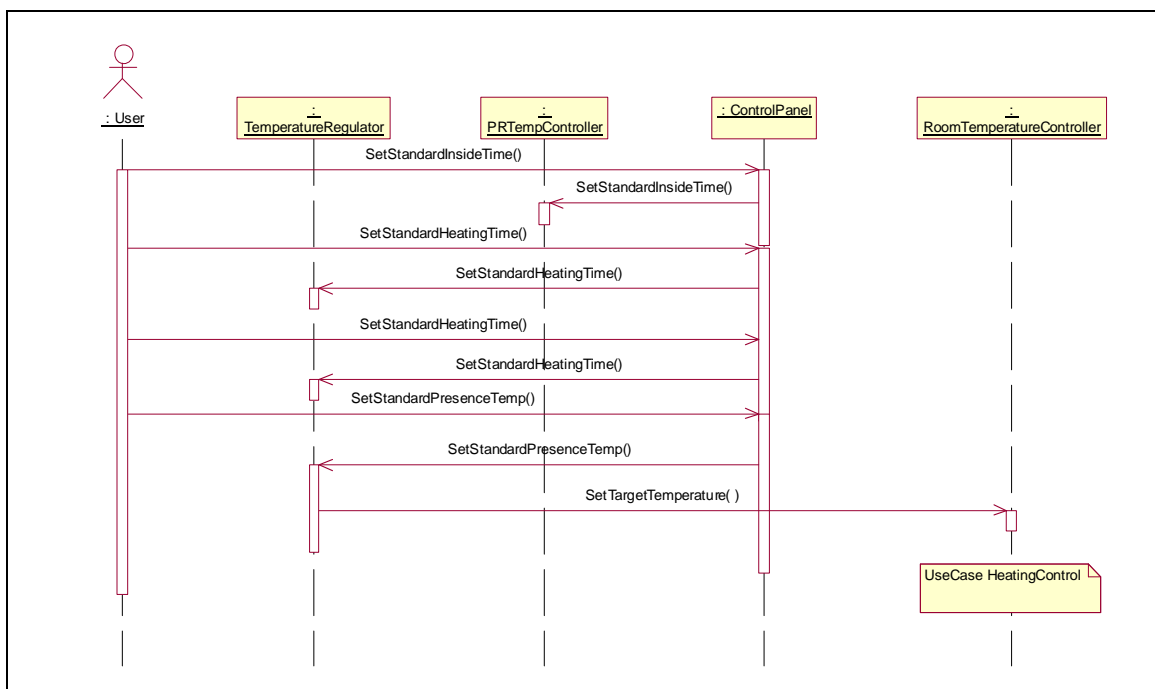
### 1.3.2.3.5 Sequence diagram for Use Case TemperatureTooHigh

Sequence diagram for scenario 7:

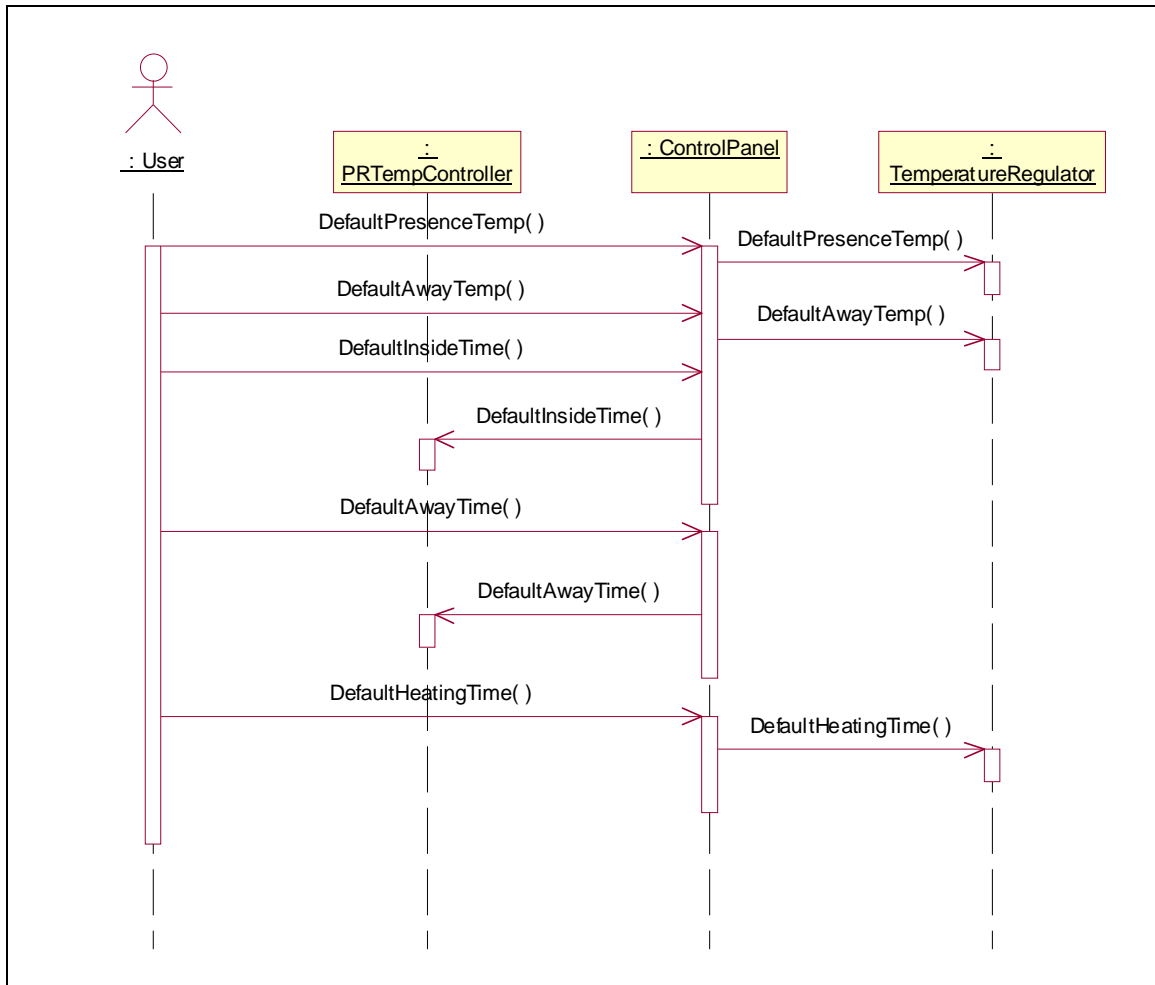


### 1.3.2.3.6 Sequence diagrams for Use Case ValuesSetting

Sequence diagram for scenario 1:



Sequence diagram for scenario 8:

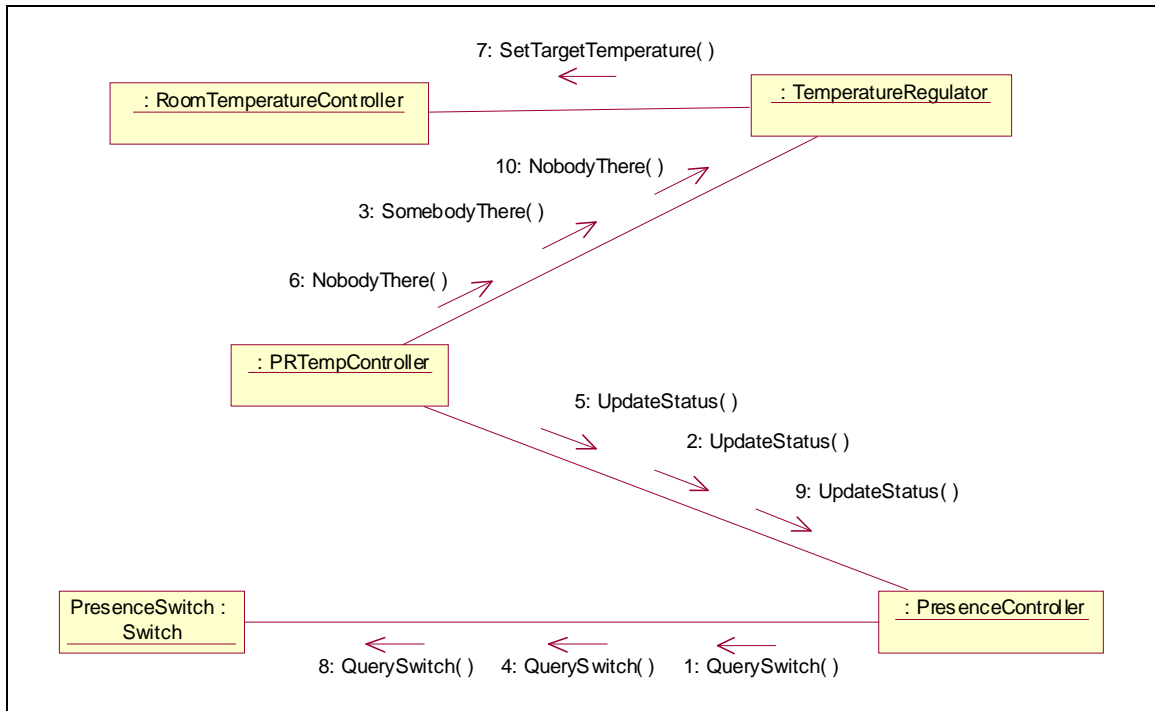


### 1.3.2.4 Collaboration diagrams

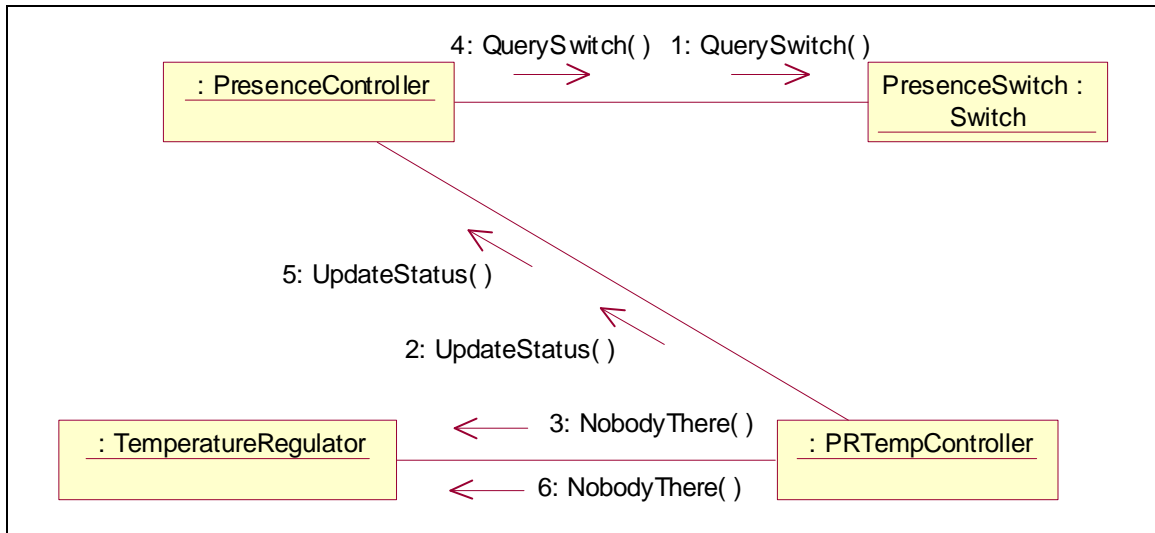
Below are collaboration diagrams for object displayed in UML class diagrams. You will deduce them automatically using StP/UML from sequence diagrams, and represent interactions among classes.

#### 1.3.2.4.1 Presence

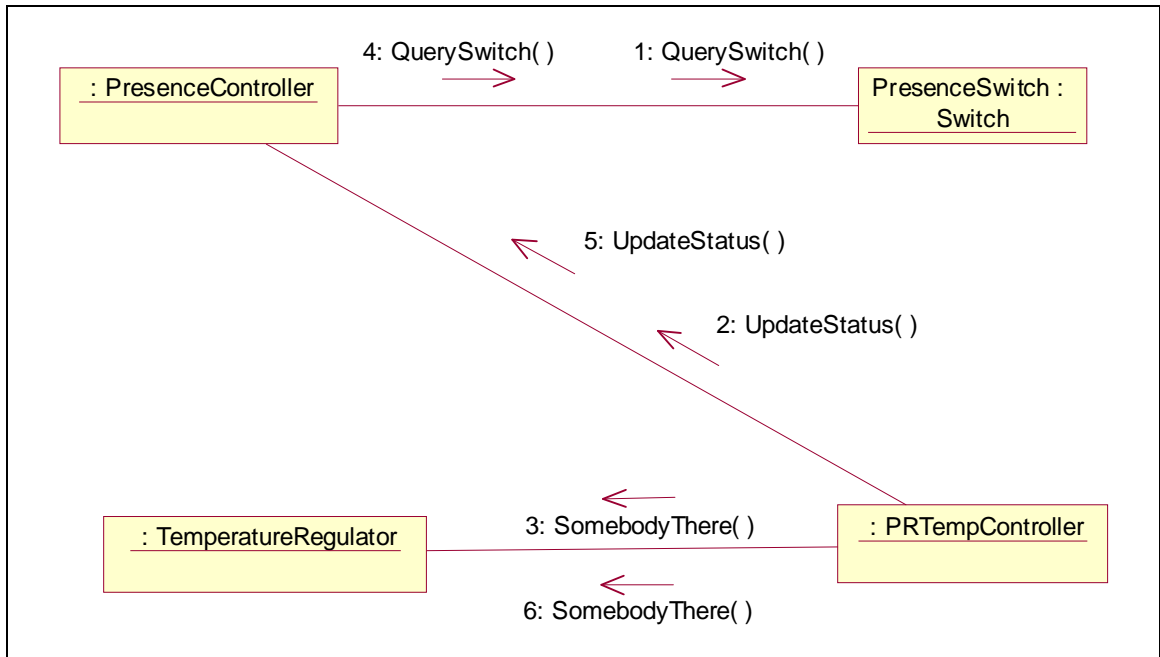
##### Scenario 3 (UmlUseCase:Presence)



**Scenario 2 (UmlUseCase:Presence)**

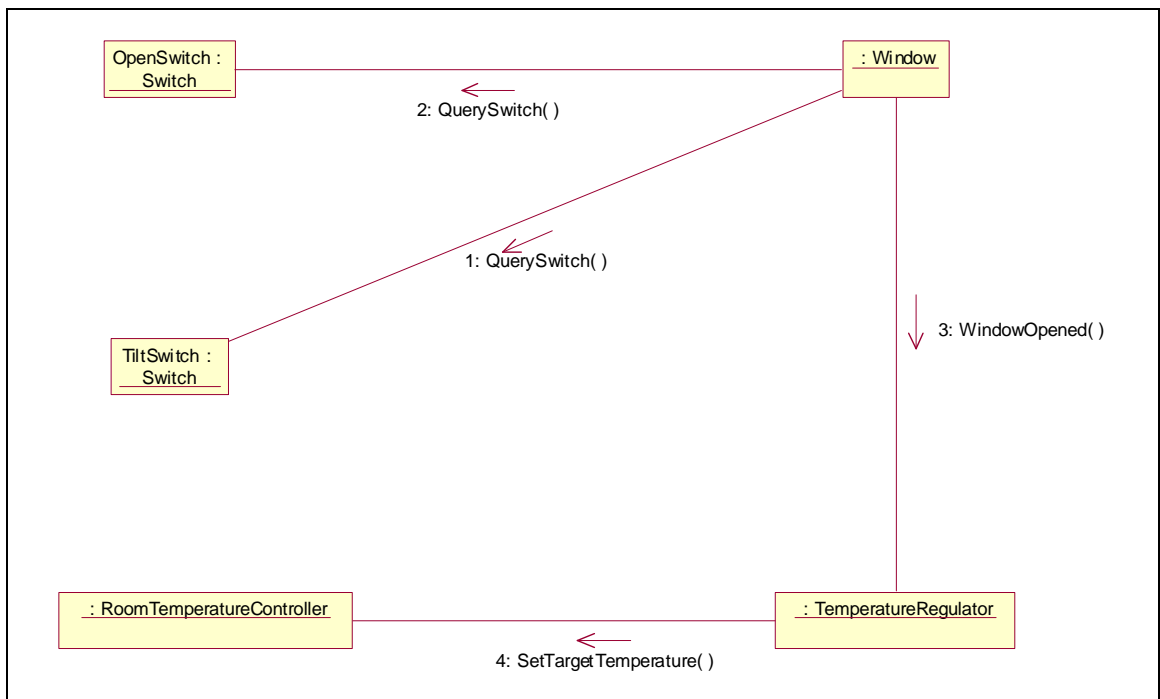


**Scenario 4(UmlUseCase:Presence)**



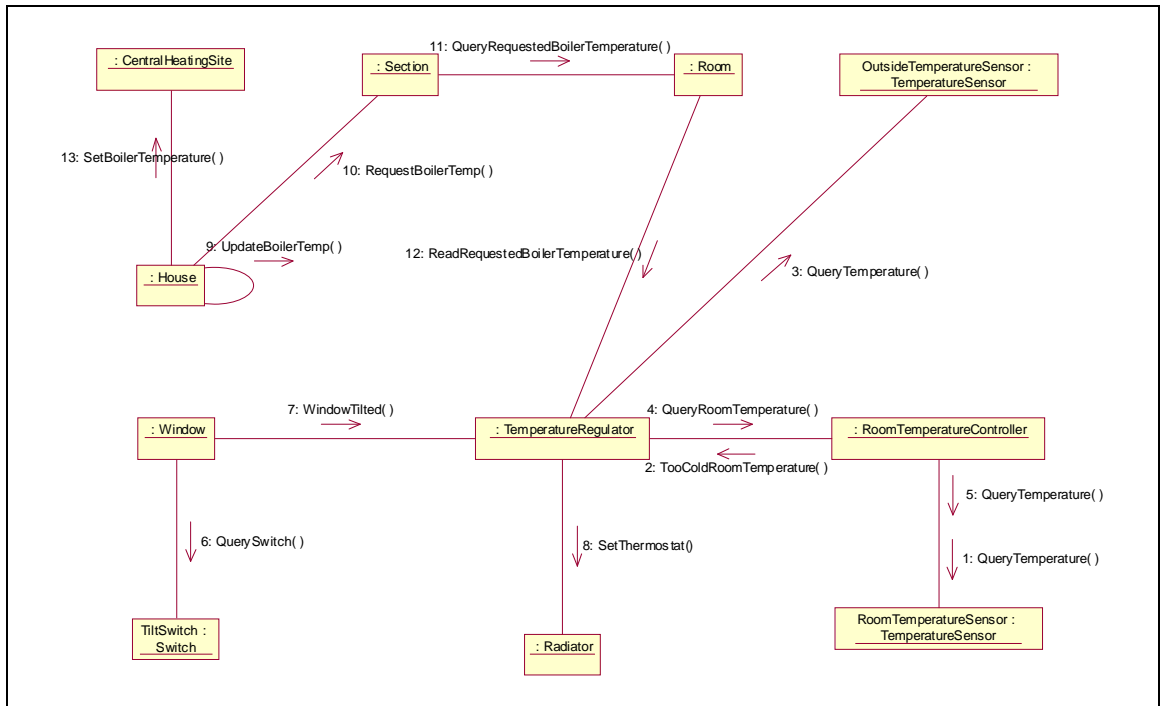
#### 1.3.2.4.2 Window position

Scenario 5 (UmlUseCase:WindowPosition)

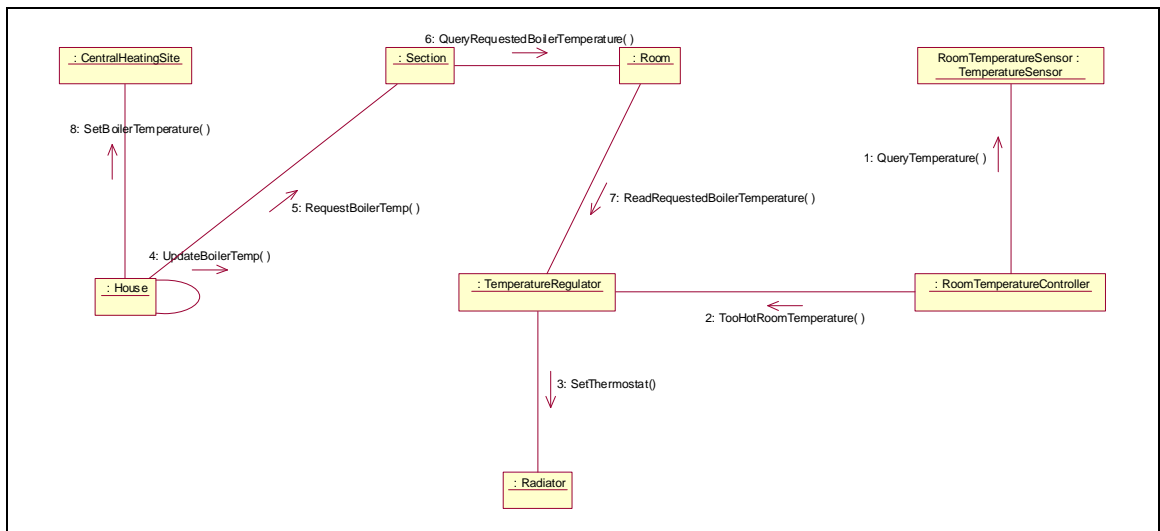


#### 1.3.2.4.3 Heating control

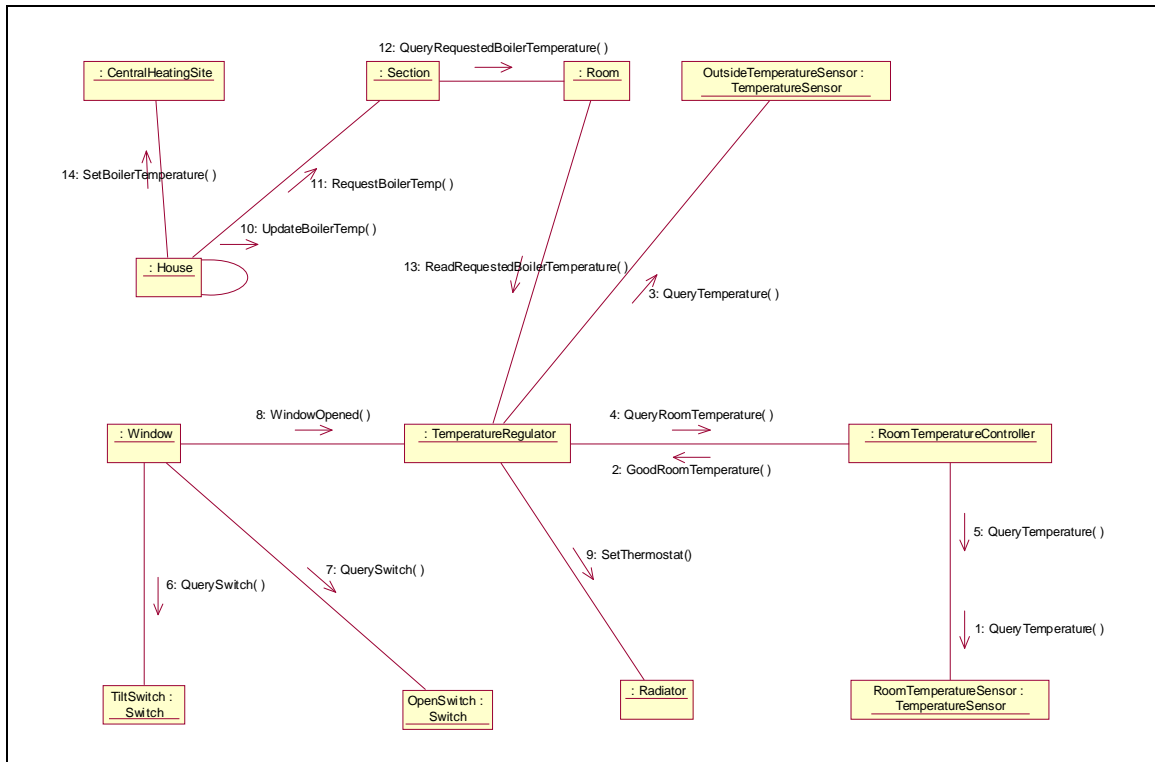
Scenario 10 (UmlUseCase:HeatingControl)



Scenario 11 (UmlUseCase:HeatingControl)

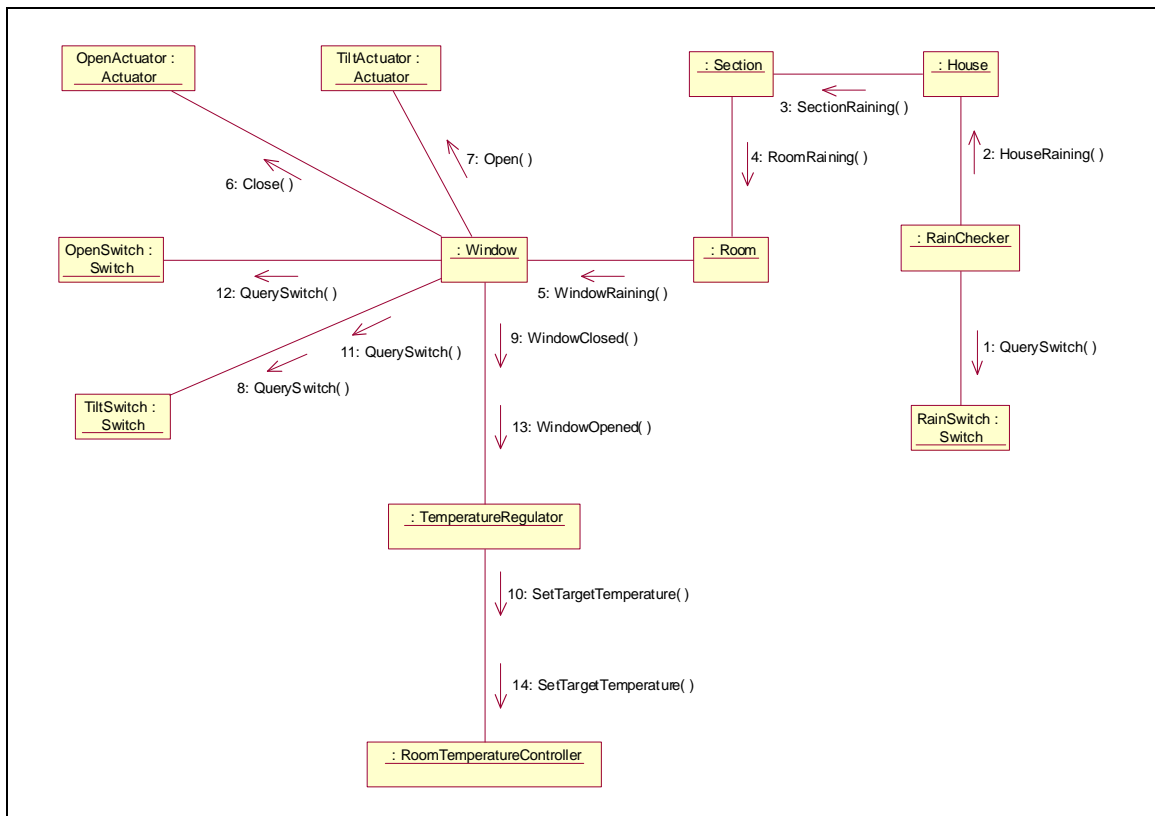


Scenario 9 (UmlUseCase:HeatingControl)



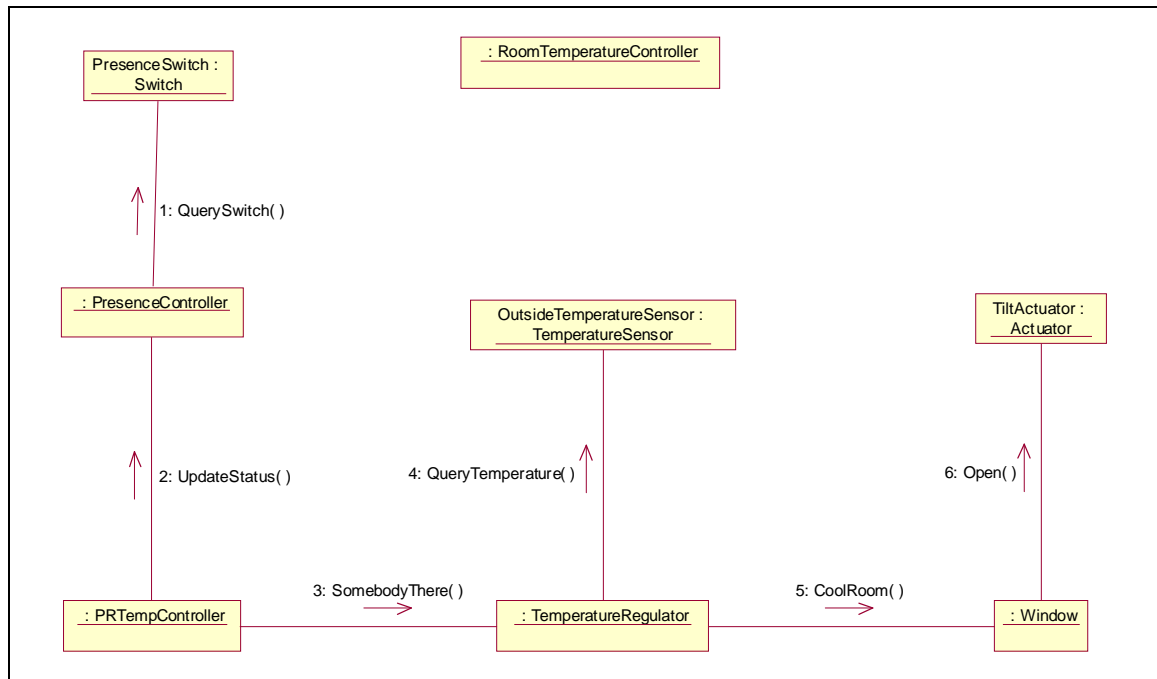
### 1.3.2.4.4 Rain monitoring

Scenario 6 (UmlUseCase:RainMonitoring)



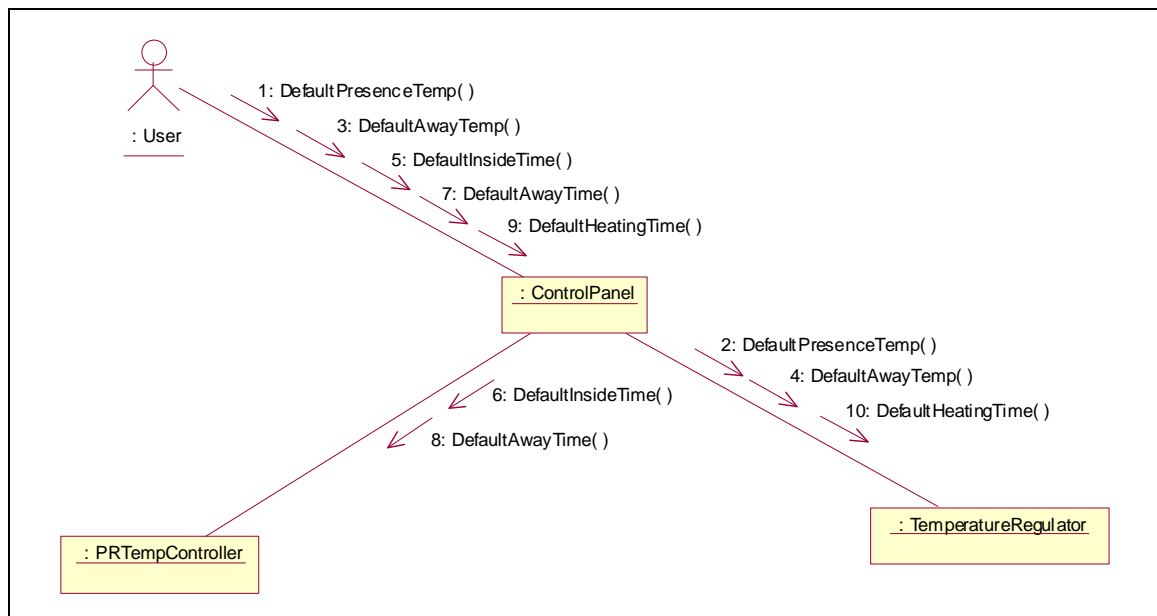
### 1.3.2.4.5 Temperature too high

Scenario 7 (UmlUseCase:TemperatureTooHigh)

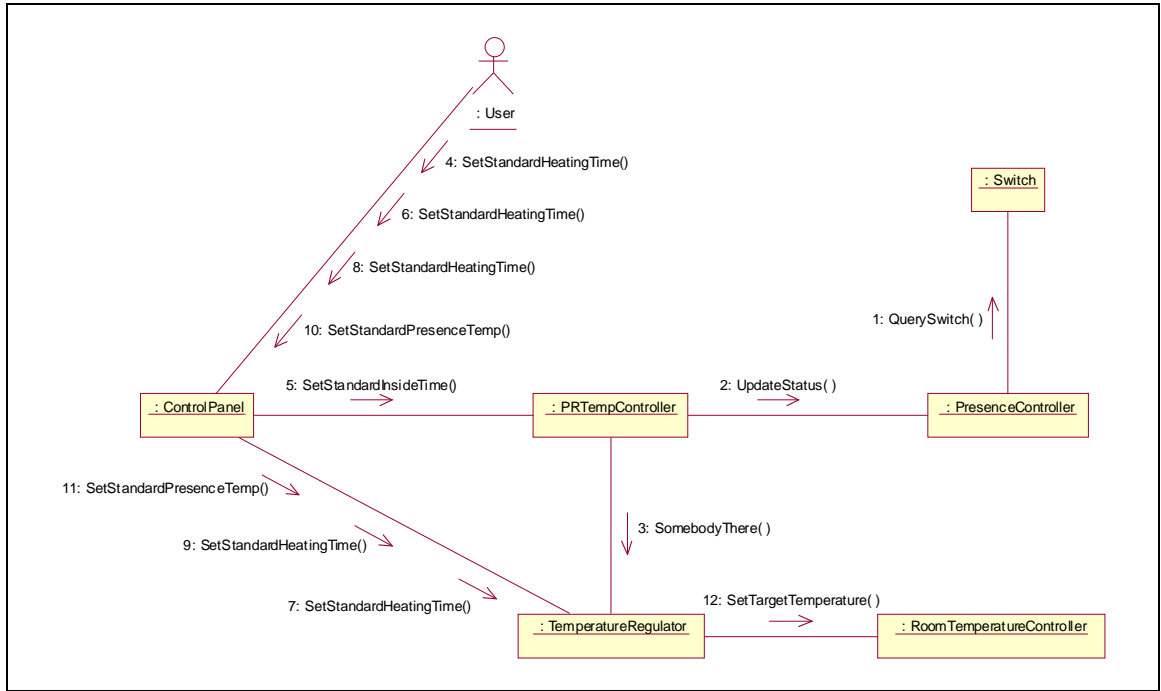


### 1.3.2.4.6 ValueSetting

Scenario 8 (UmlUseCase:ValuesSetting)



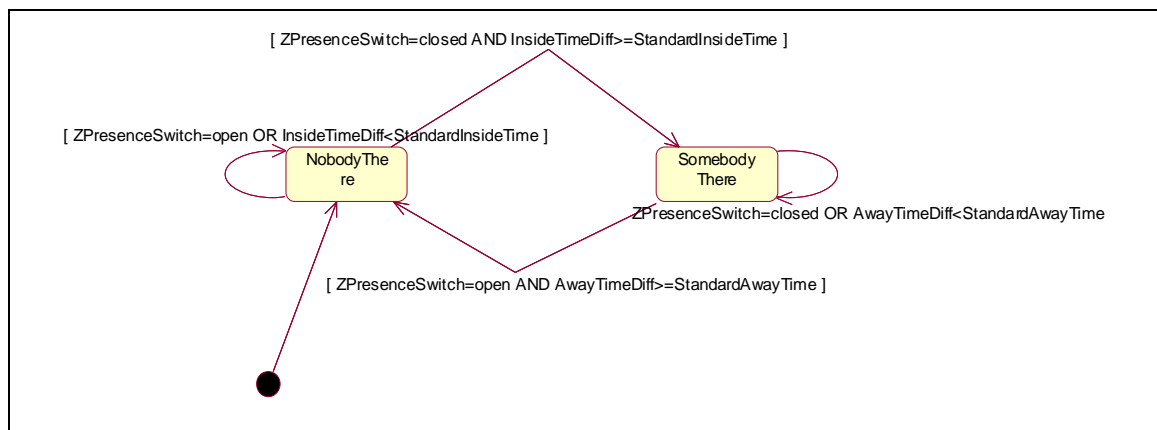
Scenario 1 (UmlUseCase:ValuesSetting)



### 1.3.2.5 UML state diagrams

State diagrams depict dynamic behavior of each class. For each class will be depicted below the behavior of each class with a specific state diagram. Methods calls are unaccounted in the state diagrams, while they serve only for setting or getting values.

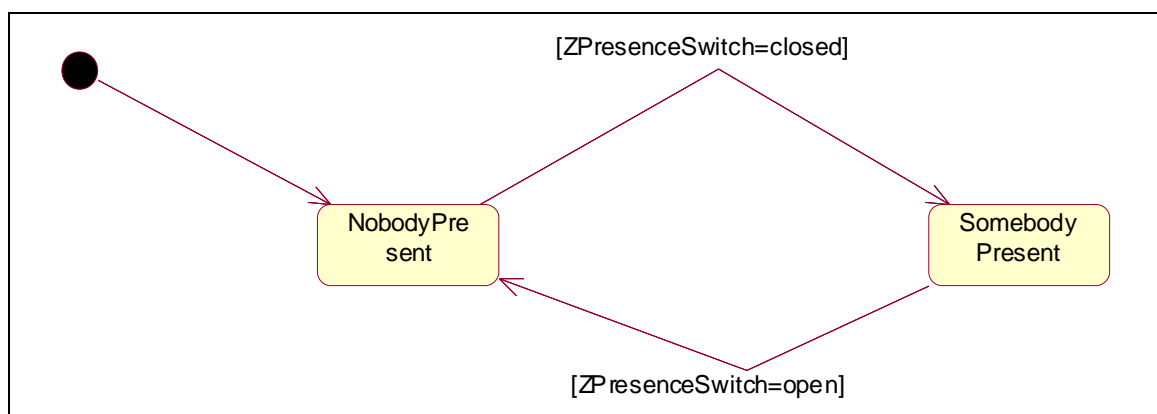
#### 1.3.2.5.1 PRTempController



Do-activity for State: NobodyThere  
 (ZPresenceSwitch, InsideTimeDiff) := PresenceController:UpdateStatus()

Do-activity for State: SomebodyThere  
 (ZPresenceSwitch, AwayTimeDiff) := PresenceController:UpdateStatus()

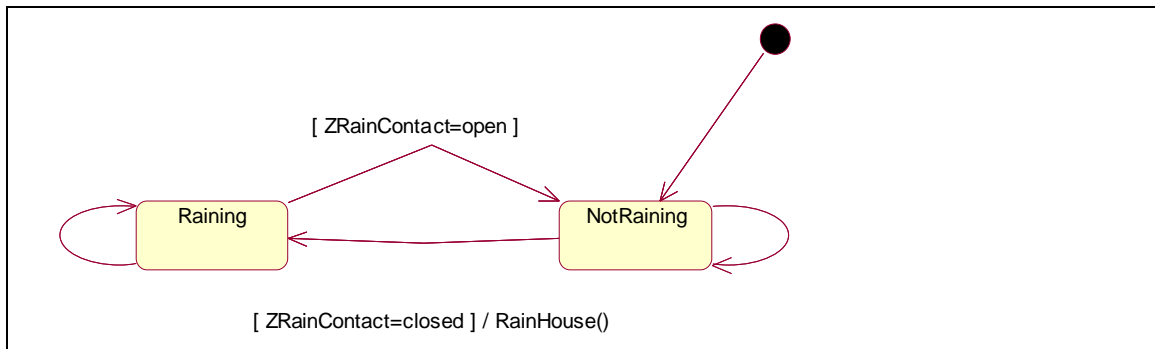
#### 1.3.2.5.2 PresenceController



Do-activity for State: NobodyPresent  
 ZPresenceSwitch := PresenceSwitch.QuerySwitch()

Do-activity for State: SomebodyPresent  
 ZPresenceSwitch := PresenceSwitch.QuerySwitch()

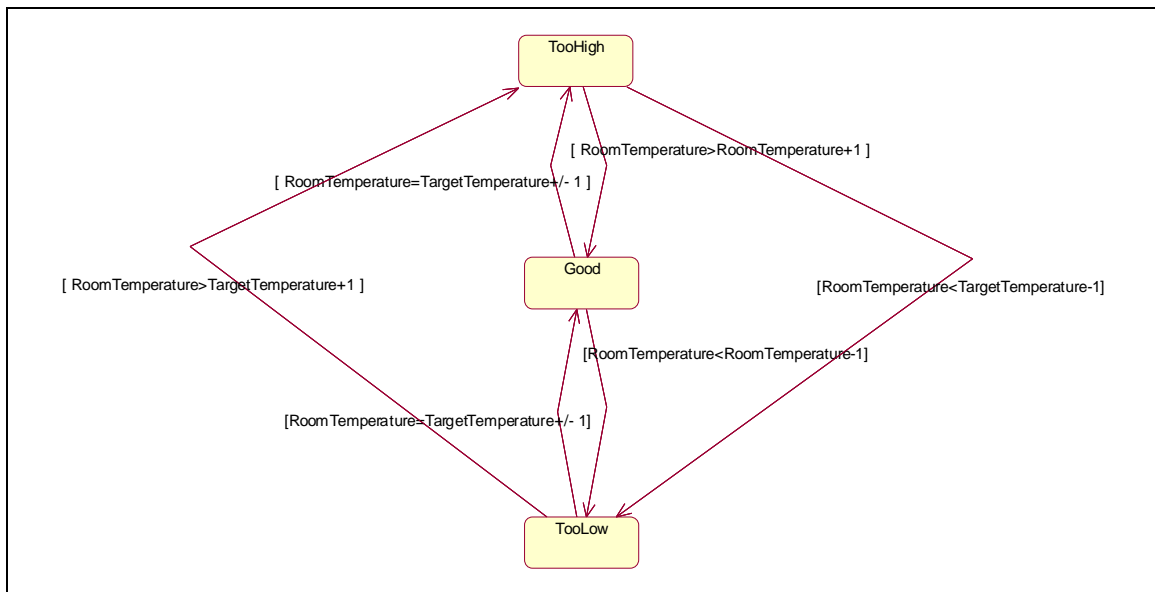
#### 1.3.2.5.3 RainChecker



Do-activity for State: Raining  
`ZRainSwitch := RainSwitch.QuerySwitch()`

Do-activity for State: NotRaining  
`ZRainSwitch := RainSwitch.QuerySwitch()`

#### 1.3.2.5.4 RoomTemperatureController



Do-activity for State: TooHigh  
`RoomTemperature := RoomTemperatureSensor.QueryTemperature()`

Do-activity for State: Good  
`RoomTemperature := RoomTemperatureSensor.QueryTemperature()`

Do-activity for State: TooLow  
`RoomTemperature := RoomTemperatureSensor.QueryTemperature()`

#### 1.3.2.5.5 TemperatureRegulator

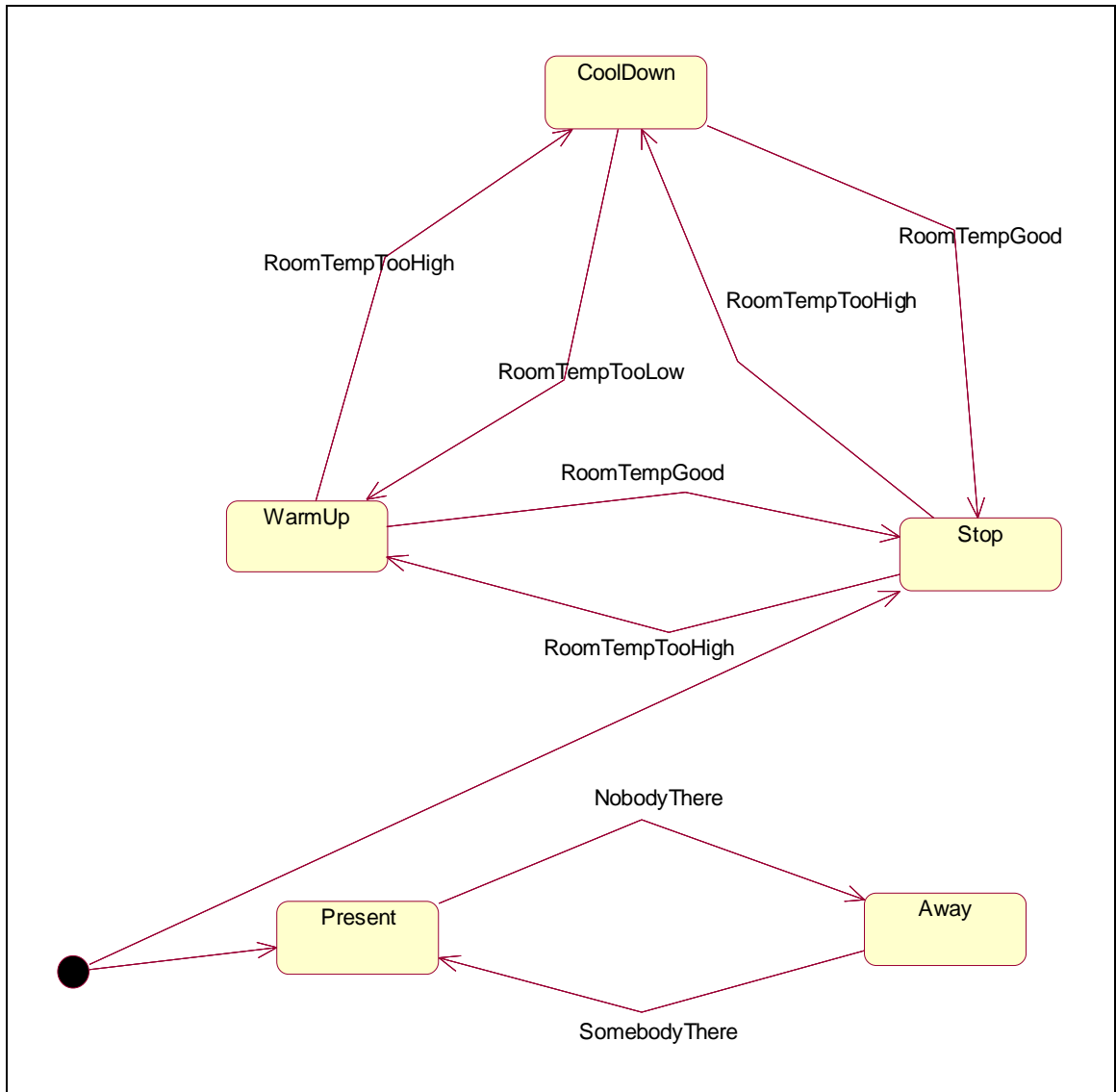
The state diagram of the class `TemperatureRegulator` is split in two different parts:

1. `TemperatureControlling`

In this part is determined using RoomTemperatureControllers if the room must be cooled or heated of if the temperature should be maintained.

## 2. TemperatureSetting

Here will be controlled using PRTempControllers and knowing the status of windows, if the target temperature should be set to StandardPresenceTemp or StandardAwayTemp.

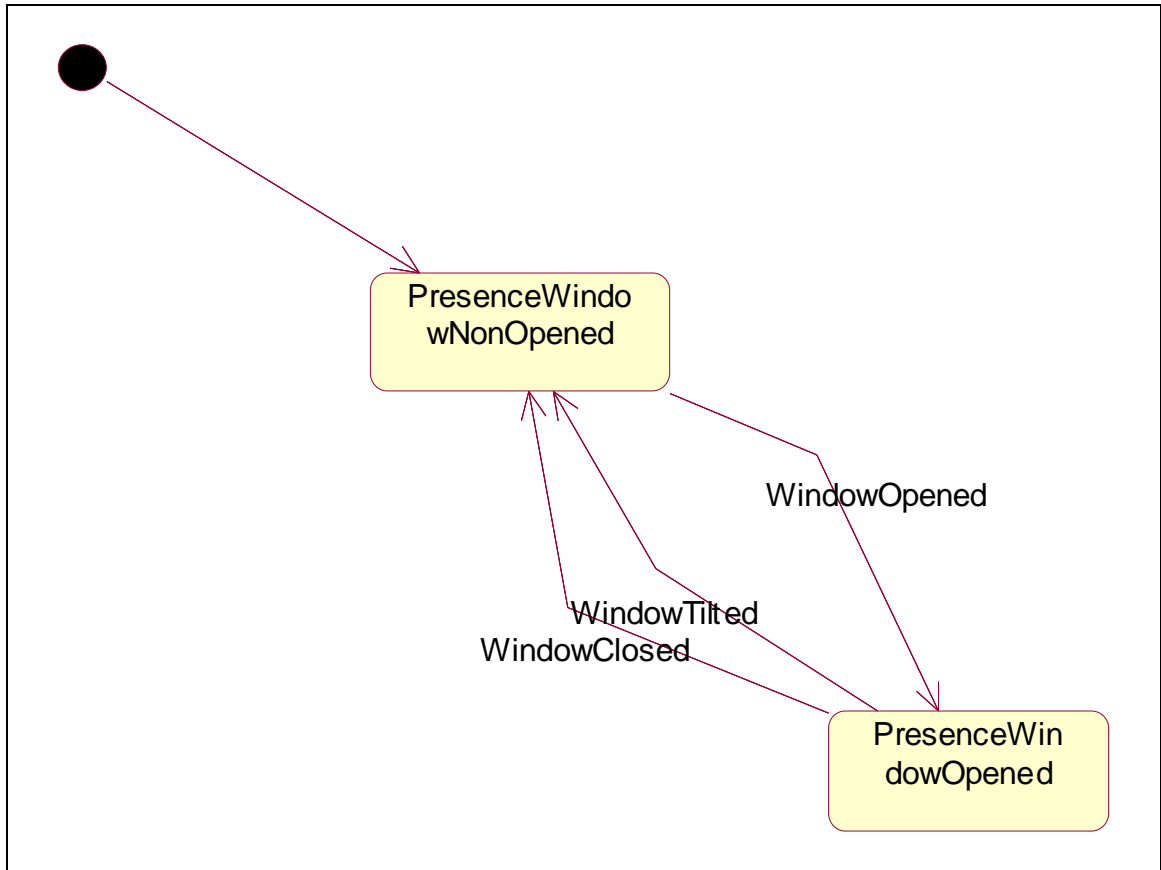


Entry-action for State: Cooling  
SetThermostat (0)

Entry-action for State: Heating  
OutsideTemperature := OutsideTemperatureSensor.QueryTemperature()  
RoomTemperature := RoomTemperatureSensor.QueryTemperature()

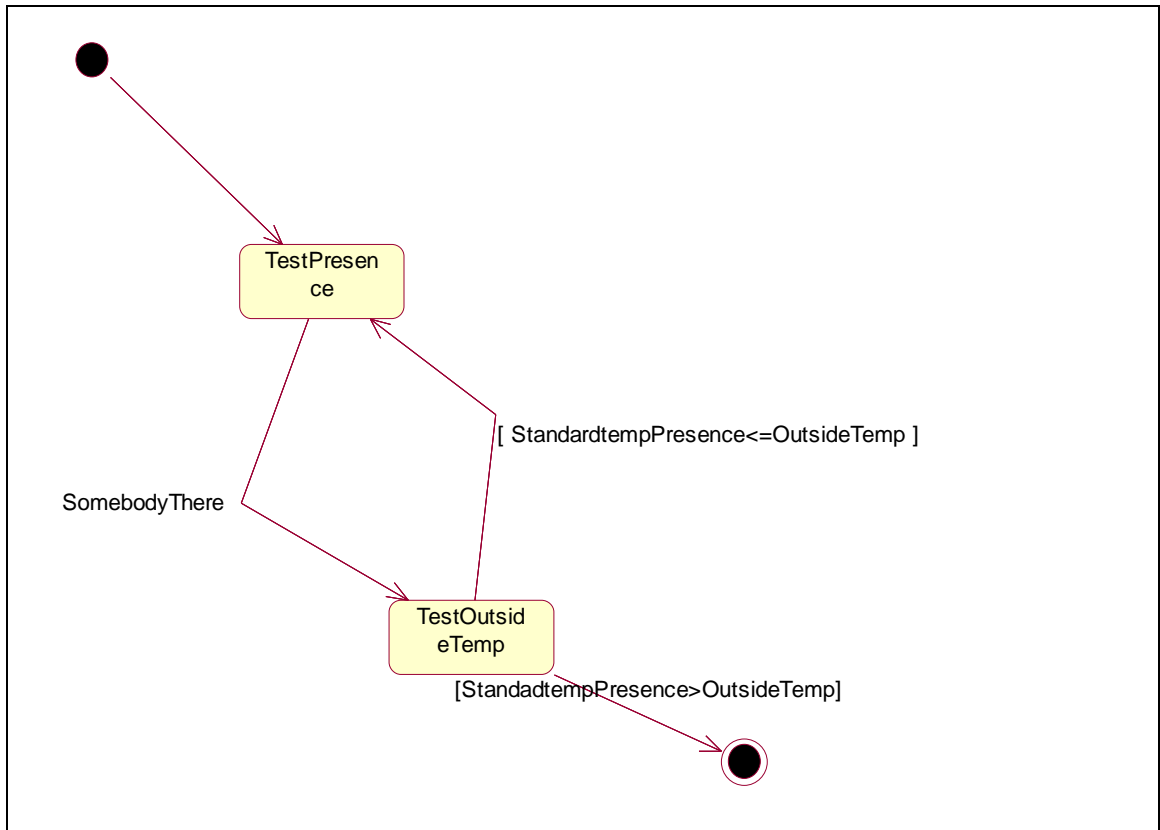
Entry-action for State: Stopping  
OutsideTemperature := OutsideTemperatureSensor.QueryTemperature()  
RoomTemperature := RoomTemperatureSensor.QueryTemperature()

Entry-action for State: Presence  
SetTargetTemperature (StandardOutsideTemp)



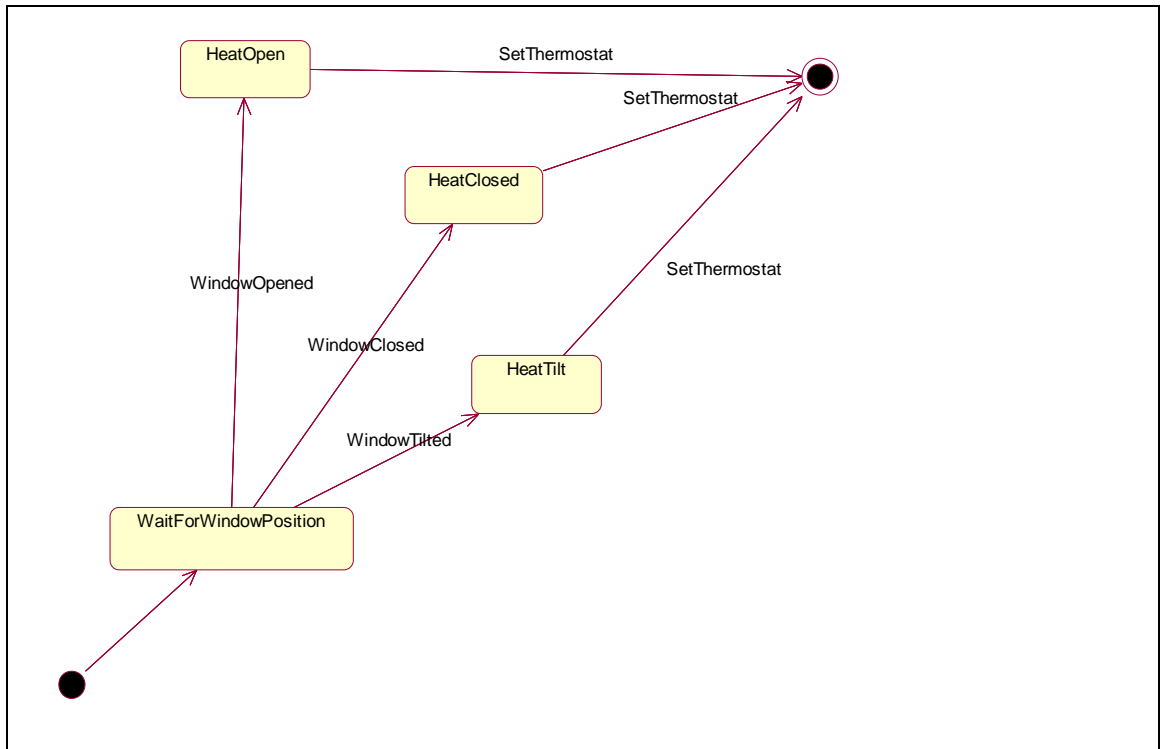
Entry-action for State: PresenceDoNotOpenWindow  
SetTargetTemperature (StandardPresenceTemp)

Entry-action for State: PresenceOpenWindow  
SetTargetTemperature (StandardOutsideTemp)



Entry-action for State: Cooling  
SetThermostat (0)

Entry-action for State: OutsideTemperature\_Proof  
OutsideTemperature := OutsideTemperatureSensor.QueryTemperature()



Entry-action for State: Heating

OutsideTemperature := OutsideTemperatureSensor.QueryTemperature()  
 RoomTemperature := RoomTemperatureSensor.QueryTemperature()

Entry-action for State: Heat\_Opened

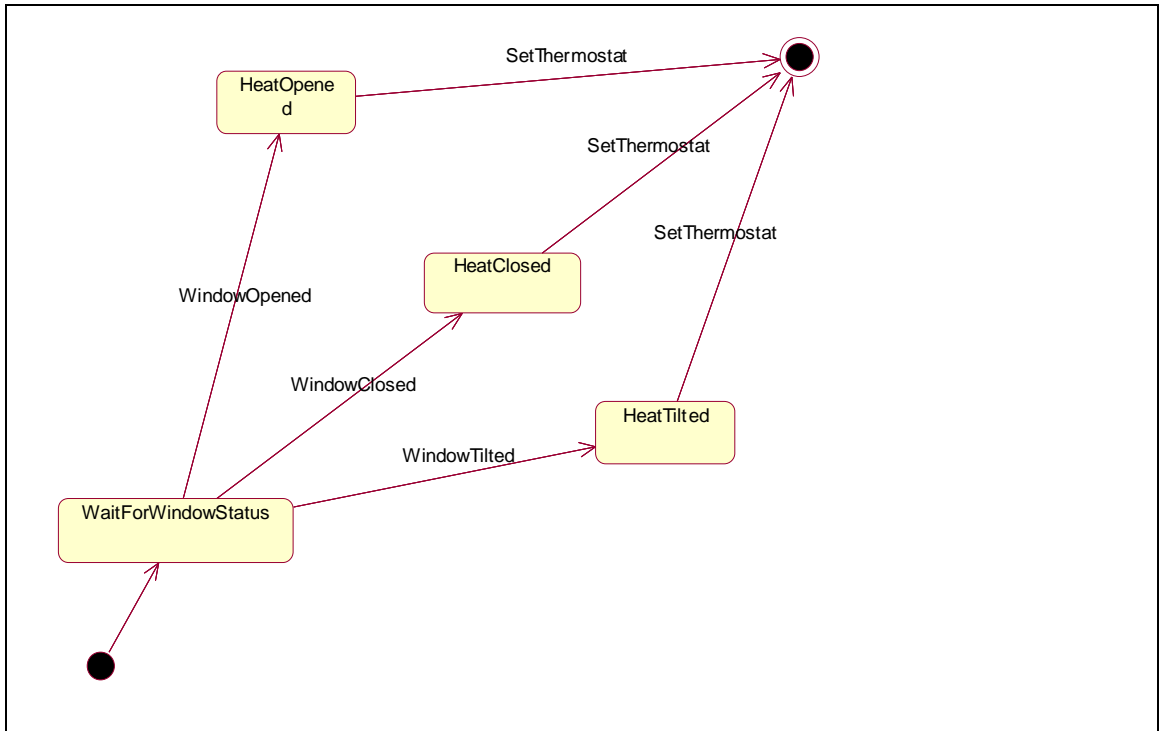
EvaluateHeatingTemperature (Open, OutsideTemperature, RoomTemperature)

Entry-action for State: Heat\_Closed

EvaluateHeatingTemperature (Close, OutsideTemperature, RoomTemperature)

Entry-action for State: Heat\_Tilted

EvaluateHeatingTemperature (Tilt, OutsideTemperature, RoomTemperature)



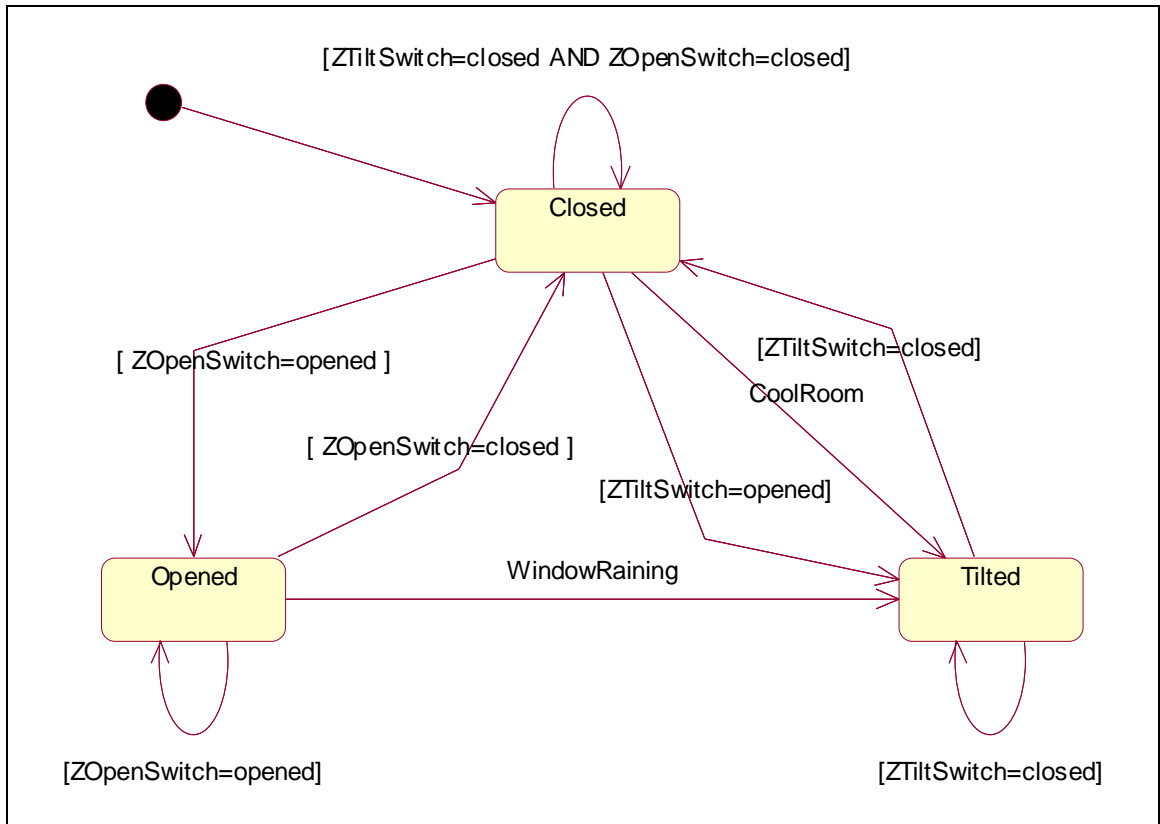
Entry-action for State: Stopping  
 OutsideTemperature := OutsideTemperatureSensor.QueryTemperature()  
 RoomTemperature:= RoomTemperatureSensor.QueryTemperature ()

Entry-action for State: Stop\_Opened  
 StopEvaluatingTemp(Opened, OutsideTemperature, RoomTemperature)

Entry-action for State: Stop\_Closed  
 StopEvaluatingTemp (Closed, OutsideTemperature, RoomTemperature)

Entry-action for State: Stop\_Tilted  
 StopEvaluatingTemp (Tilted, OutsideTemperature, RoomTemperature)

### 1.3.2.5.6 Window



Do-activity for State: Closed  
 ZTitlSwitch := TiltSwitch.QuerySwitch()  
 ZopenSwitch := OpenSwitch.QuerySwitch()

Do-activity for State: Tilted  
 ZTiltSwitch := TiltSwitch.QuerySwitch()

Do-activity for State: Opened  
 ZOpenSwitch := OpenSwitch.QuerySwitch()

### 1.3.3 Traceability matrix: user requirements <-> class diagrams

In the traceability matrix will be depicted in which classes have been mapped user requirements. Inside it you'll find those classes, which get instantiated but not instances.

	CentralHeatingSite	Switch	House	Actuator	PRTempController	PresenceController	Radiator	RainChecker	Room	RoomTemperatureController	Section	TemperatureRegulator	TemperatureSensor	Window
Temp-UR-F1									X					
Temp-UR-F10		X			X	X								
Temp-UR-F11					X	X						X		
Temp-UR-F12					X	X						X		
Temp-UR-F13	X	X					X			X		X	X	X
Temp-UR-F14		X	X	X				X			X	X		X
Temp-UR-F15		X		X	X	X				X		X	X	X
Temp-UR-F16										X				
Temp-UR-F17			X											
Temp-UR-F18	X		X						X		X			
Temp-UR-F2									X					
Temp-UR-F3									X					
Temp-UR-F4					X				X					
Temp-UR-F5					X				X					
Temp-UR-F6	X	X			X	X	X			X		X	X	X
Temp-UR-F7	X	X			X	X	X			X		X	X	X
Temp-UR-F8	X	X			X	X	X			X		X	X	X
Temp-UR-F9		X			X	X						X		