Problem Specification of a System for the Light and Temperature Control in a Floor of a Building of the University of Kaiserslautern

Martin Kronenburg
Research Group Formal Methods and Deduction
kronburg@informatik.uni-kl.de

Christian Peper
Research Group Computer Networks
peper@informatik.uni-kl.de

University of Kaiserslautern

9th June 1999

Contents

Introduction i
Aggregation- and Inheritance Hierarchy ii

Description Classes

Global Signature 1
Group Datatypes 4
Description Class LightScene 5
Description Class LightSceneOffice 7
Group BasicHumanMachineInterface 8
Description Class EnteredValue 9
Description Class DynamicInterval 12
Description Class CalendarInterval 14
Description Class ControlledInput 15
Description Class ControlledDynamicInterval 18
Group Sensors 19
Description Class NoMalfunctionSensor 20
Description Class MalfunctionSensor 24
Description Class Sensor 28
Description Class RoomTempSensor 32
Description Class OutdoorTempSensor 33
Description Class OutdoorLightSensor 34
Description Class WaterTempSensor 36
Description Class BinarySensor 37
Description Class LightSensor 38
Description Class DoorContact 39
Description Class MotionDetector 40
Group Actuator 41
Description Class NoMalfunctionActuator 42
Description Class MalfunctionActuator 44
Description Class Actuator 47
Description Class ToggleActuator 50
Description Class Pulse 51
Description Class RangeActuator 52
Description Class Dimmer 53
Description Class BinaryActuator 54
Description Class ControlSystemActive 55
Description Class ThreePointActuator 56
Description Class RadiatorValve 57
Description Class PushButton 58
Group CombinedSensorsAndActuators 59
Description Class Door 60
Description Class SunBlind 61
Description Class Radiator 66

Description Class ControlledLight 67
Description Class DimmableControlledLight 70
Group HumanMachineInterface 71
Description Class OccDelayTimes 72
Description Class AreaLightCtrlPanelFM 74
Description Class RoomLightCtrlPanelFM 75
Description Class LightCtrlPanelRoom 77
Description Class LightCtrlPanelBlindedRoom 80
Description Class TemperatureCtrlPanelFM 81
Description Class TemperatureCtrlPanelRoom 84
Description Class TemperatureCtrlPanel 85
Description Class ExpectedUsage 87
Description Class GroupOccupancy 89
Description Class AdaptiveOccupancyTimes 90
Description Class NoMalfunctionOccupancy 91
Description Class MalfunctionOccupancy 94
Description Class Occupancy 96
Group TemperatureModes 99
Description Class TempOperationMode 100
Description Class TempOperationModeSet 101
Group BuildingUnits 105
Description Class AreaLight 106
Description Class HallwaySection 112
Description Class RoomLight 115
Description Class BlindedRoomLight 123
Description Class RoomTemperature 125
Description Class BlindedRoomTemperature 132
Description Class BasicRoom 134
Description Class Room 137
Description Class BlindedRoom 140
Description Class RoomNeighb 144
Description Class Office 145
Description Class OfficeNeighb 146
Description Class BlindedOffice 147
Description Class BlindedOfficeNeighb 148
Description Class NoOffice 149
Description Class BlindedNoOffice 150
Description Class BlindedNoOfficeNeighb 151
Description Class StairCase 152
Description Class Floor 153
Index 165

Introduction

This document contains a problem specification for the light and temperature control in the fourth floor of building 32 of the University of Kaiserslautern developed according to the FOREST approach. This projection includes all formal and natural language specifications as well as all references within the document.

A general introduction to problem specifications developed according to the FOREST approach can be found at the following url
`http://www-avenhaus.informatik.uni-kl.de/forest/EXPLANATIONS/PS_ExplanationEnglish.ps.gz`.

A html-version of this problem specification can be found at the following url
`http://www-avenhaus.informatik.uni-kl.de/forest/EXAMPLES/LIGHT_TEMP/LightTempFoNLPostRef/LightTempFoNLPostRef.html`.

Aggregation and Inheritance Hierarchy

The graphical overview of the aggregation and inheritance hierarchy among the description classes consists of seven figures. Every figure contains the relations within one group of description classes. Figure 1 is the main overview. It represents the relations within the group *Building Units*.

Starting with the main description class *Floor* all relations between the classes are depicted until a class is not refined or until it belongs to another group. Classes that belong to another group and are refined are displayed in a specific color. The refined structure is depicted in a subsequent figure. The relation between groups, colors, and figures is the following one:

- Group *Sensors*; red; Figure 2
- Group *Actuators*; blue; Figure 3
- Group *CombinedSensorsandActuators*; brown; Figure 4
- Group *HumanMachineInterface*; green; Figure 5
- Group *Occupancy*; violet; Figure 6
- Group *TemperatureModes*; light-blue; Figure 7

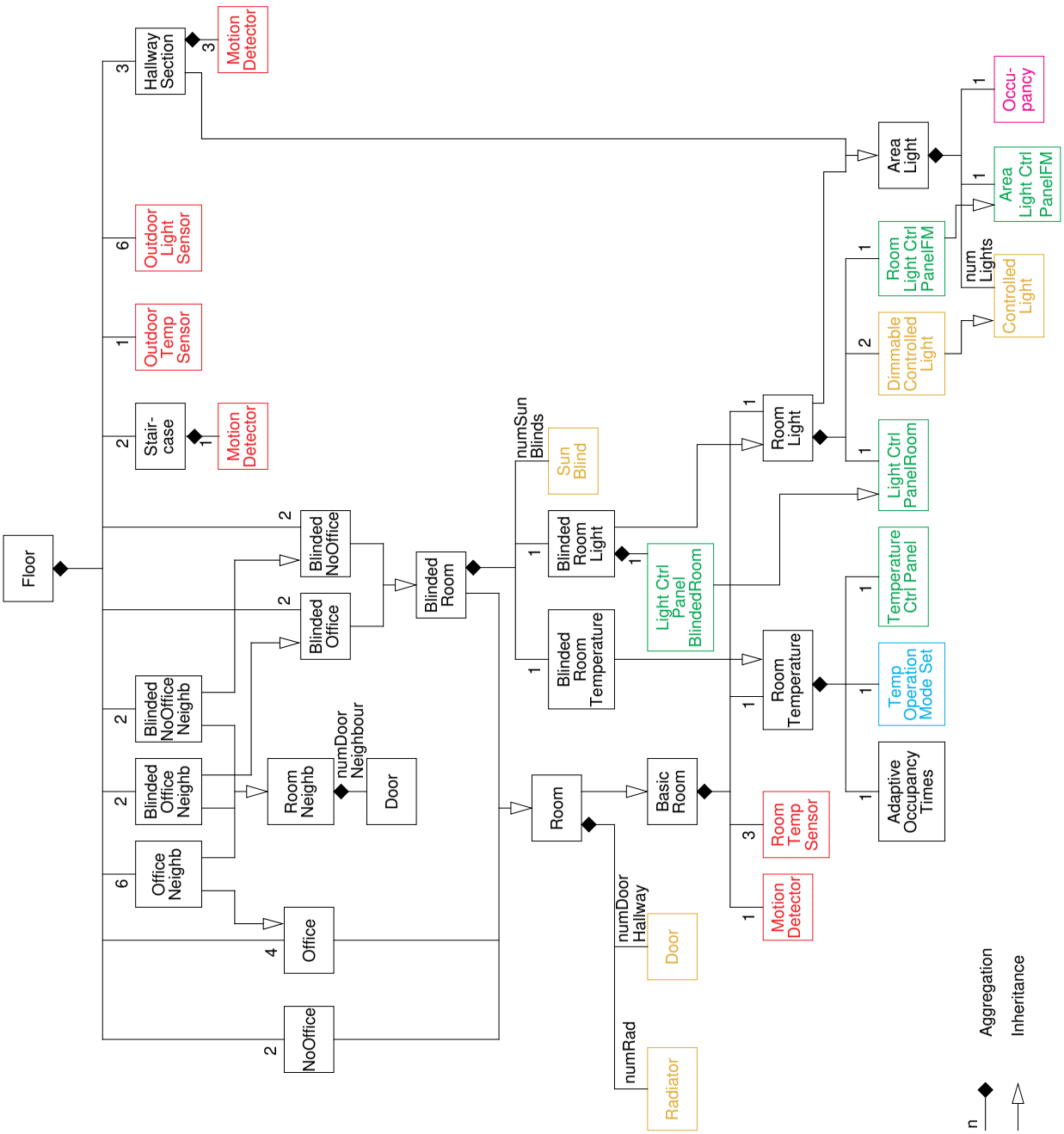

Figure 1: Class structure of group *BuildingUnits*

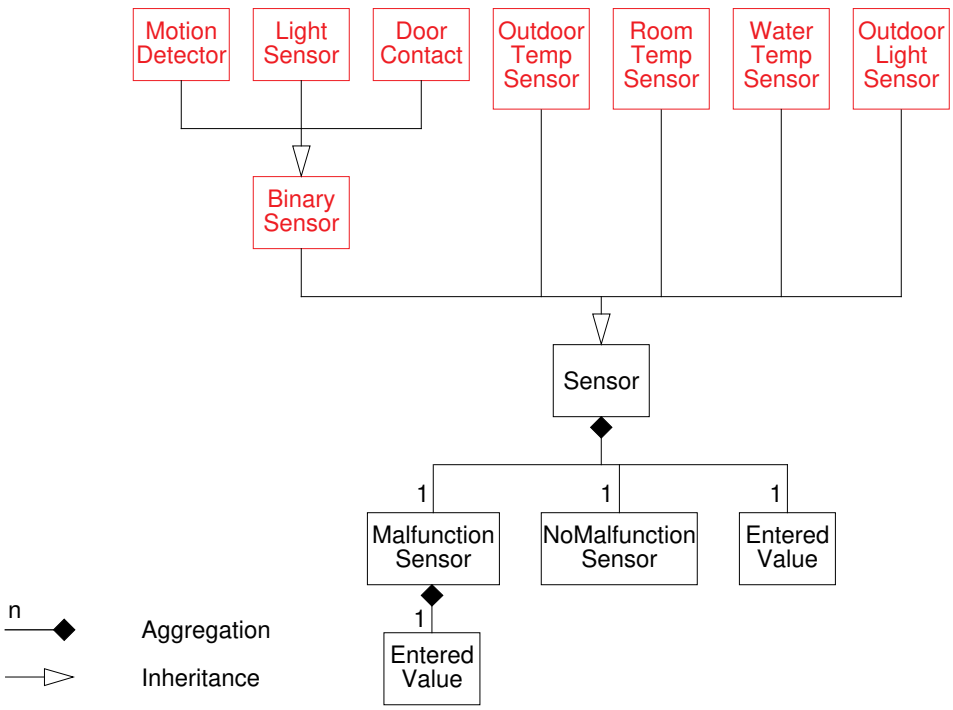

Figure 2: Class structure of group *Sensors*

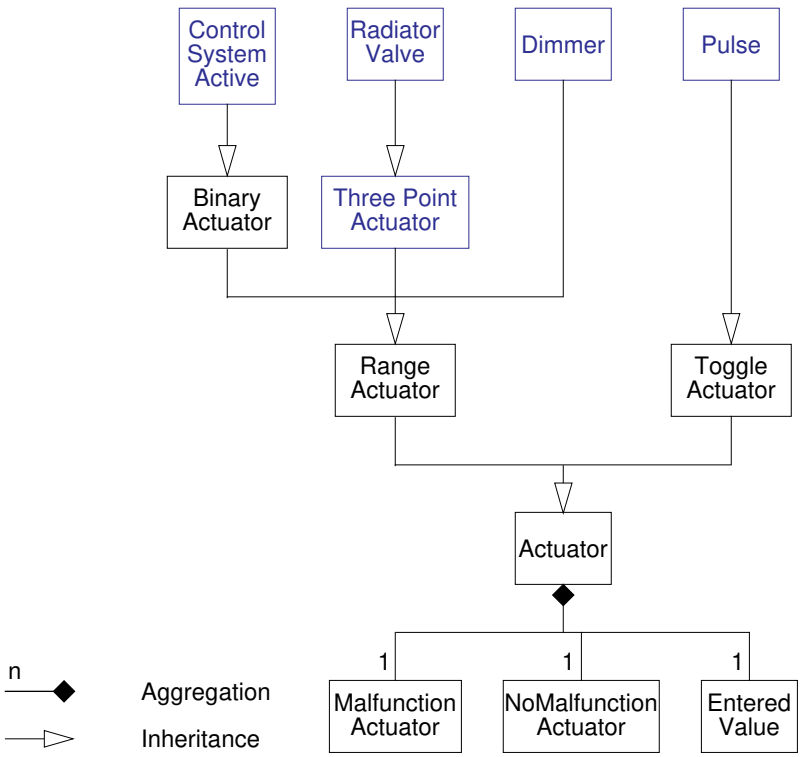

Figure 3: Class structure of group *Actuators*

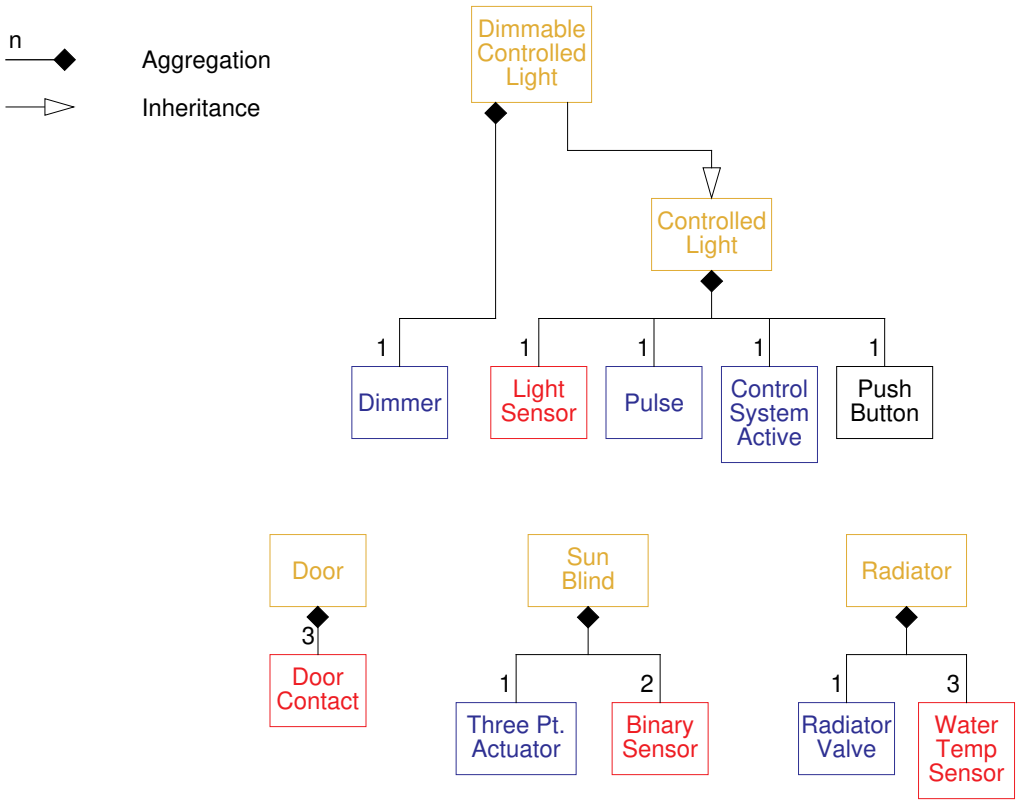

Figure 4: Class structure of group *CombinedSensorsAndActuators*

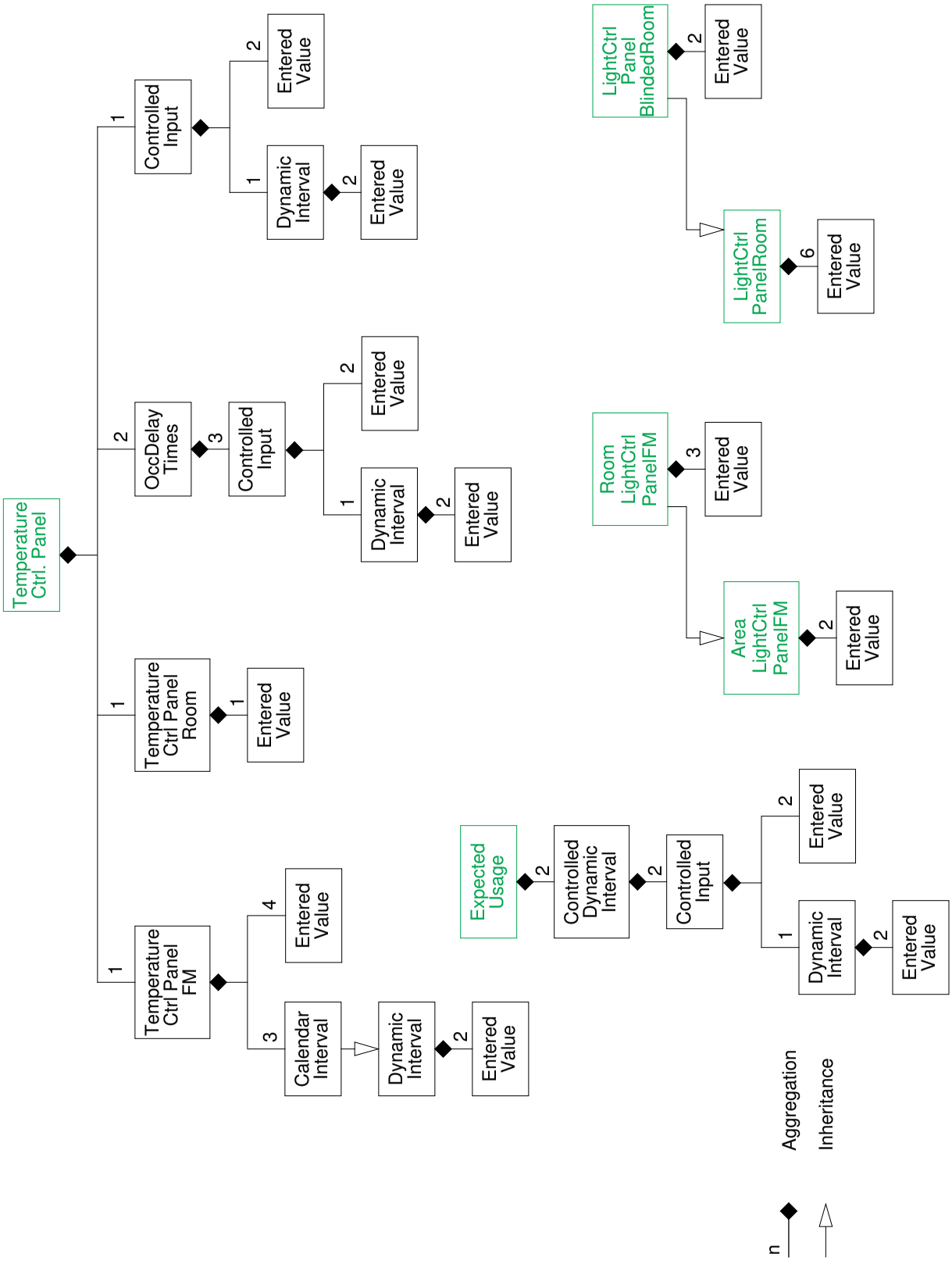

Figure 5: Class structure of group *HumanMachineInterface*

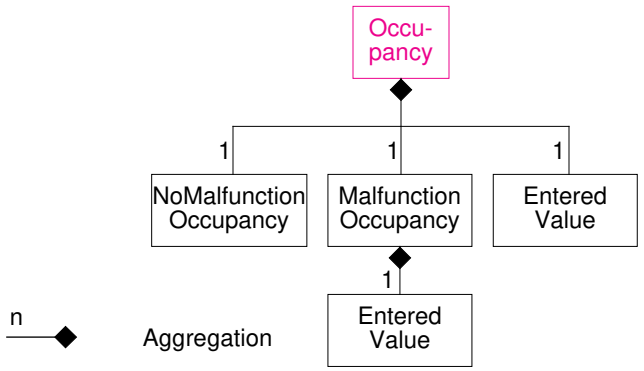

Figure 6: Class structure of group *Occupancy*

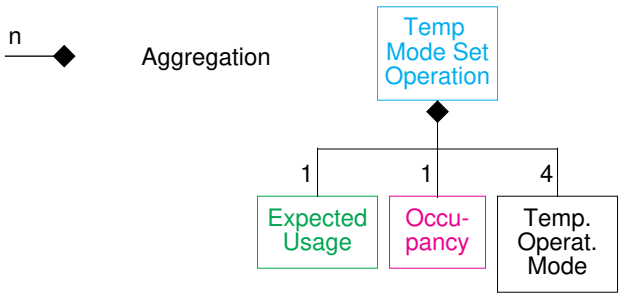

Figure 7: Class structure of group *TemperatureModes*

Global Signature

Value *FORWARD* : *THREE_PT_ACT_RANGE*

Intention : This value represents the signal causing a three point actuator to move in one of the possible two directions.
Postreferences : dkref SunBlind : D_SBL6
dkref SunBlind : D_SBL8
msref SunBlind : M_SBL1

Value *BACKWARD* : *THREE_PT_ACT_RANGE*

Intention : This value represents the signal causing a three point actuator to move in one of the possible two directions.
Postreferences : dkref SunBlind : D_SBL7
dkref SunBlind : D_SBL9
msref SunBlind : M_SBL2

Value *STOP* : *THREE_PT_ACT_RANGE*

Intention : This value represents the signal causing a three point actuator to stop.
Postreferences : dkref SunBlind : D_SBL5

Domain $TEMP = \{-40 \ldots 100\} \subseteq INTEGER$

Intention : This domain contains the possible temperature values in degrees Celsius that are considered in this problem specification.
Postreferences : gsref COMFORT_TEMP
lsref RoomTempSensor : ROOM_TEMP
lsref OutdoorTempSensor : OUT_TEMP
lsref WaterTempSensor : WATER_TEMP
lsref TemperatureCtrlPanelFM : nonFreezeTemperature
lsref TemperatureCtrlPanelFM : standbyTemperature
lsref TemperatureCtrlPanelFM : offTemperature
lsref TempOperationMode : desiredTemperature
lsref TempOperationModeSet : roomDesiredTemperature
lsref BlindedRoomTemperature : tempBlindNoMalSet
lsref BlindedRoomTemperature : tempBlindMalSet
lsref Room : tempRadNoMalSet
lsref Room : tempRadMalSet

Domain $LIGHT_VALUES = \{1 \ldots 10000\} \subseteq NAT$

Intention : This domain contains the possible light intensity values in lux that are considered in this problem specification.
Postreferences : lsref LightScene : ambientIlluminance
lsref OutdoorLightSensor : OUT_LIGHT_VALUES
lsref RoomLightCtrlPanelFM : defaultAmbientLightLevel
lsref LightCtrlPanelRoom : ambLightLev
lsref RoomLight : curAmbLightLev
lsref RoomLight : defaultAmbLightLev
lsref RoomLight : evalAmbient
msref RoomLight : M_RL14

Domain $BINARY = \{0, 1\} \subseteq NAT$

Intention : This domain contains the values 0 and 1.
Postreferences : lsref Sensor : manualEnteredMal
lsref BinarySensor : Sensor
dkref BinarySensor : D_BS2
dkref BinarySensor : D_BS3
lsref Actuator : manualEnteredMal
lsref ToggleActuator : Actuator
lsref BinaryActuator : RangeActuator
dkref BinaryActuator : D_BA1
lsref AreaLightCtrlPanelFM : lightOn
lsref LightCtrlPanelRoom : winCeilLightSwitch
lsref LightCtrlPanelRoom : wallCeilLightSwitch
lsref LightCtrlPanelRoom : ambientSwitch
lsref LightCtrlPanelRoom : lightSceneSwitch
lsref LightCtrlPanelBlindedRoom : sunBlindManual
lsref TemperatureCtrlPanelFM : adaptive
lsref TemperatureCtrlPanelRoom : nonFreezeOverride
lsref MalfunctionOccupancy : manualEnteredOcc
lsref Occupancy : manualEnteredMal
lsref AreaLight : evalSaveLight

Domain $PERCENT = \{0 \ldots 100\} \subseteq NAT$

Intention : This domain contains the natural numbers between 0 to 100.
Postreferences : lsref Dimmer : RangeActuator
lsref RoomLight : evalLightScene
lsref RoomLight : evalAmbient
msref RoomLight : M_RL11

Domain $COMFORT_TEMP \subseteq TEMP$

Intention : This domain contains the values that can be used for a comfort temperature.
Note that this domain is not specified by the facility manager but is predefined, containing all possible comfort temperatures, not only those allowed by the facility manager.
Postreferences : lsref TemperatureCtrlPanel : comfortTemperature

Domain $THREE_PT_ACT_RANGE = \{FORWARD, BACKWARD, STOP\}$

Intention : This domain contains the possible control states of a three point actuator.
Postreferences : lsref ThreePointActuator : RangeActuator
dkref ThreePointActuator : D_TPA1
lsref LightCtrlPanelBlindedRoom : sunBlindSwitch
lsref BlindedRoomLight : lightBlindSet
lsref BlindedRoomLight : lightBlindNoMalSet
lsref BlindedRoomLight : lightBlindMalSet
lsref BlindedRoomTemperature : tempBlindSet
lsref BlindedRoomTemperature : tempBlindNoMalSet
lsref BlindedRoomTemperature : tempBlindMalSet
lsref Room : tempRadSet
lsref Room : tempRadNoMalSet
lsref Room : tempRadMalSet
lsref BlindedRoom : resolveBlindConflict

Domain $CALENDAR_TIME$

Intention : This domain contains the values that can be used for a calendar time, i.e. a date.
Postreferences : gsref currentDate
lsref CalendarInterval : DynamicInterval
lsref ExpectedUsage : holidayTime

Domain $HOURS = \{0 \ldots 23\} \subseteq NAT$

Intention : This domain contains the hours of a day.
Postreferences : gsref currentHour
lsref ExpectedUsage : workingTime

Timed Function $currentDate \rightarrow CALENDAR_TIME$

Intention : This function represents the current date, i.e. the current day, month, and year.
Scope : eh
Postreferences : rsref ExpectedUsage : D_EU1
rsref CalendarInterval : R_CI1

Timed Function $currentHour \rightarrow HOURS$

Intention : This function represents the current hour.
Scope : eh
Postreferences : rsref ExpectedUsage : D_EU1

Group Datatypes

Intention

This group contains all description classes that are used as datatypes. This means that the description classes in this group are only applied to the declaration of functions or predicates. They are not used for aggregation or inheritance.

Description Class LightScene

Intention

This class provides all definitions concerning a light scene for an arbitrary room. These are w.r.t. the DIC document:

- a name,
- a value of the light intensity of the ambient light of a room,
- an ordered list containing the installed lamps. The list defines the order in which the control system has to use the lamps to reach the given ambient light value. Since this list always consists of two entries we model this list by two functions firstAmbientIlluminaire and secondAmbientIlluminaire.

Aspects concerning a light scene of an office are presented in the derived description class LightSceneOffice.

Signature

Value *WINDOW* : *LAMPS*

Intention	: This value represents the ceiling light group in a room that is installed near the window.
Postreferences	: msref RoomLight : M_RL8 msref RoomLight : M_RL9

Value *WALL* : *LAMPS*

Intention	: This value represents the ceiling light group in a room that is installed near the wall to the hallway.
Postreferences	: msref RoomLight : M_RL8 msref RoomLight : M_RL9

Domain *LAMPS* = {*WINDOW*, *WALL*}

Intention	: This domain contains the installed lamps that can be used to establish a certain light scene. • *WINDOW* stands for the window ceiling light and • *WALL* for the ceiling light near the door to the hallway.
Postreferences	: lsref LightScene : firstAmbientLighting lsref LightScene : secondAmbientLighting lsref LightSceneOffice : firstDeskLighting lsref LightSceneOffice : secondDeskLighting lsref RoomLight : evalLightScene lsref RoomLight : evalAmbient

Timed Function *name* → *STRING*

Intention	: This function represents the name of the light scene.
Scope	: mh

Timed Function *ambientIlluminance* → *LIGHT_VALUES*

Intention	: This function represents the desired ambient light intensity of a room.
Scope	: mh

Timed Function *firstAmbientLighting* → *LAMPS*

Intention	: This function represents the first lamp that should be used to reach the desired ambient light of a room.
Scope	: mh
Postreferences	: msref LightScene : M_LS1

Timed Function *secondAmbientLighting* → *LAMPS*

Intention	: This function represents the second lamp that should be used to reach the desired ambient light of a room.
Scope	: mh
Postreferences	: msref LightScene : M_LS1

Machine Specifications

Property *M_LS1*

Formal	: $\Box(\neg(\mathit{firstAmbientLighting} = \mathit{secondAmbientLighting}))$
NL	: The first lamp that should be used to reach the desired ambient light of a room and the second lamp that should be used to reach the desired ambient light of a room are always different.

Description Class LightSceneOffice

Intention

This Class contains all the entities concerning a light scene for an office. In addition to the entities belonging to the basic class *LightScene* this class also contains the following entities.

- a value of the light intensity for the desk
- An ordered list containing the installed lamps. The list defines the order in which the control system has to use the lamps to reach the given light intensity value for the desk.

Note that it is not possible for the machine to control the task light. Therefore, it is not considered here!

Base Classes

Class *LightScene*

Intention	: All definitions of description class *LightScene* are also provided here.

Signature

Timed Function *firstDeskLighting* $\rightarrow$ *LightScene* :: *LAMPS*

Intention	: This function represents the first lamp that should be used to reach the desired desk light of a room.
Scope	: mh
Postreferences	: msref LightSceneOffice : M_LSO1

Timed Function *secondDeskLighting* $\rightarrow$ *LightScene* :: *LAMPS*

Intention	: This function represents the second lamp that should be used to reach the desired desk light of a room.
Scope	: mh
Postreferences	: msref LightSceneOffice : M_LSO1

Machine Specifications

Property *M_LSO1*

Formal	: $\Box(\neg(\mathit{firstDeskLighting} = \mathit{secondDeskLighting}))$
NL	: The first lamp that should be used to reach the desired desk light and the second lamp that should be used to reach the desired desk light are always different.

Group BasicHumanMachineInterface

Intention

This group contains all description classes that contain the basic definitions representing the interface between the machine and a person. This means here the possibility for a person to enter a value.

Description Class EnteredValue

Intention

This class provides all definitions to describe the entering of a value using an appropriate device (in our case this device is a control panel). For a value e that is entered (expressed by the function *entered*) a value c (expressed by the function *checked*) is determined using the function *nearestValue*. This function is intended to check the entered value e whether it is an allowed value and returns the value which is *nearest* to e.

It is assumed that the entered value can not be seen by the machine, i.e. there exists up to now no possibility to enter the value. Hence, the development of such a device is part of the overall development of the system. In our case this is the control panel, which still has to be developed. Therefore the timed function *entered* has the scope *eh*.

Formal Parameters

Sort *CHECKED_DOM*

Intention	: This domain contains the values that can be produced out of an entered value using the function *nearestValue*.

Signature

Timed Function $entered \rightarrow STRING$

Intention	: This function represents the value entered by a person. The value is represented as a string.
Scope	: eh
Postreferences	: rsref EnteredValue : R_EV1

Timed Function *checked* → *CHECKED_DOM*

Intention : This function represents the value decided by the machine to use. *checked* is determined out of *entered* using the function *nearestValue*.
Scope : mh
Postreferences : msref DynamicInterval : M_DI1
msref ControlledInput : I_CIN2
msref ControlledInput : M_CIN2
msref ControlledInput : M_CIN3
msref MalfunctionSensor : M_MS1
msref Sensor : M_S1
msref Actuator : M_A1
msref OccDelayTimes : M_ODT1
msref OccDelayTimes : M_ODT2
msref OccDelayTimes : M_ODT3
msref AreaLightCtrlPanelFM : I_LCPFM1
msref AreaLightCtrlPanelFM : I_LCPFM2
msref RoomLightCtrlPanelFM : I_LCPR1
msref RoomLightCtrlPanelFM : M_LCPR1
msref TemperatureCtrlPanelFM : M_TCPF1
msref TemperatureCtrlPanelRoom : M_TCPR1
msref MalfunctionOccupancy : M_MO1
msref Occupancy : M_OCC1
msref AreaLight : M_AL5
msref RoomLight : M_RL2
msref RoomLight : M_RL3
msref RoomLight : M_RL4
msref RoomLight : M_RL5
msref RoomLight : M_RL6
msref RoomLight : M_RL7
msref RoomLight : M_RL8
msref RoomLight : M_RL9
msref RoomLight : M_RL10
msref RoomLight : M_RL12
msref RoomLight : M_RL13
msref RoomLight : M_RL14
msref RoomTemperature : M_RT1
msref RoomTemperature : M_RT3
msref RoomTemperature : M_RT4
msref RoomTemperature : M_RT5
msref RoomTemperature : M_RT6
msref RoomTemperature : M_RT7
msref RoomTemperature : M_RT8
msref BlindedRoom : M_BR1
msref BlindedRoom : M_BR4
rsref EnteredValue : R_EV1
rsref CalendarInterval : R_CI1
rsref AreaLight : R_AL1

Timed Function *nearestValue* : *STRING* → *CHECKED_DOM*

Intention	: This function returns the value which is the *nearest* one in the domain CHECKED_DOM w.r.t. the first argument.
Scope	: mh
Postreferences	: rsref EnteredValue : R_EV1

Timed Function *checkTime* → *TIME*

Intention	: This function represents the time needed to check a value, entered by a person, using the function *nearestValue*.
Scope	: mh
Postreferences	: rsref EnteredValue : R_EV1

Requirement Specifications

Property *R_EV1*

Formal	: $\Box((entered \rhd^{nearestValue}_{checkTime} checked))$
NL	: The checked value is always derived from the entered value using the function *nearestValue*. A change of the entered value is always propagated within the time *checkTime*.

Description Class DynamicInterval

Intention

This class provides all definitions to describe a *dynamic interval*. An interval is a subset of a specific domain and is given by a start (object *start*) and an end point (object *end*). The domain is given by the formal parameter *INT_DOM*. Furthermore, a predicate *leq* is provided indicating whether a value of the considered domain is less or equal to another value.
Dynamic means that the values of the start and end point of the interval can change over time. Both values can be entered by a person. Therefore the start and end point are objects of the class *EnteredValue*.

Formal Parameters

Sort *INT_DOM*

Intention : This domain contains the values the start and end point of the interval have to belong to.

Signature

Object *start* : *EnteredValue*(*CHECKED_DOM* = *INT_DOM*)

Intention : This object represents the starting point of the interval that can be entered by a person. The entered value has to be an element of the domain *INT_DOM*.
Postreferences : msref DynamicInterval : M_DI1
msref ControlledInput : M_CIN2
msref ControlledInput : M_CIN3
rsref CalendarInterval : R_CI1

Object *end* : *EnteredValue*(*CHECKED_DOM* = *INT_DOM*)

Intention : This object represents the end point of the interval that can be entered by a person. The entered value has to be an element of the domain *INT_DOM*.
Postreferences : msref DynamicInterval : M_DI1
msref ControlledInput : M_CIN2
msref ControlledInput : M_CIN3
rsref CalendarInterval : R_CI1

Predicate *leq* : *INT_DOM*, *INT_DOM*

Intention : This predicate is true iff the first argument is *less* than or equal to the second argument.
Scope : mh
Postreferences : msref DynamicInterval : M_DI1
msref ControlledInput : M_CIN2
msref ControlledInput : M_CIN3
rsref CalendarInterval : R_CI1

Machine Specifications

Property *M_DI1*

Formal	: $\Box(leq(start.checked, end.checked))$
NL	: The starting point of the interval is always less than or equal to the end point of the interval.

Description Class CalendarInterval

Intention

This class provides all definitions to describe an interval of calendar times. The start and end point of the interval can change over time. Therefore, this class is derived from the description class *DynamicInterval*. Additionally, a timed predicate *actual* indicates whether the current date (*currentDate*) is in this interval or not.

Base Classes

Class *DynamicInterval*($INT_DOM = CALENDAR_TIME$)

Intention	: All definitions of a dynamic interval are also provided here, using the domain containing the calendar times as interval domain.

Signature

Timed Predicate *actual*

Intention	: This predicate is true iff the current date is in the calendar interval.
Scope	: mh
Postreferences	: msref RoomTemperature : M_RT1 rsref CalendarInterval : R_CI1

Requirement Specifications

Property *R_CI1*

Formal	: $\Box(actual \leftrightarrow leq(start.checked, currentDate) \wedge leq(currentDate, end.checked))$
NL	: The current time is in the calendar interval iff the interval start point is less than or equal to the current time and the current time is less than or equal to the interval end point.

Description Class ControlledInput

Intention

This class contains all the definitions that are necessary to model the *controlled* input of a value by a user. This value is represented by the timed function *value*. *Controlled* means here that the facility manager prescribes an interval in which the value entered by a user has to be. Furthermore, the facility manager prescribes a default value (object *default*) which is valid as long as a user has not entered a value. As soon as a user has entered a value this becomes the actual value unless a user changes it by entering a new value. There is no other possibility to change the value.

Formal Parameters

Sort *INPUT_DOM*

Intention : This domain contains all the values that can be entered for the boundaries of the interval the facility manager can prescribe, for the default value given by the facility manager, and for the value entered by a user.

Signature

Object *allowedRange* : *DynamicInterval*(*INT_DOM* = *INPUT_DOM*)

Intention : This object represents the range of possible user entries. This range is specified by the facility manager.
Postreferences : msref ControlledInput : M_CIN2
msref ControlledInput : M_CIN3

Object *default* : *EnteredValue*(*CHECKED_DOM* = *INPUT_DOM*)

Intention : This object represents the default value given by the facility manager. Note that by this declaration it is only guaranteed that the default value is an element of the domain *INPUT_DOM*. It is not guaranteed that it is a value of the interval specified by the facility manager. This is expressed by the property P_CIN_3 of this class.
Postreferences : msref ControlledInput : I_CIN2
msref ControlledInput : M_CIN3
msref OccDelayTimes : M_ODT1
msref OccDelayTimes : M_ODT2
msref OccDelayTimes : M_ODT3

Object *userValue* : *EnteredValue*(*CHECKED_DOM* = *INPUT_DOM*)

Intention : This object represents the value entered by a user.
Note that by this declaration it is only guaranteed that the entered value is an element of the domain *INPUT_DOM*. It is not guaranteed that it is a value of the interval specified by the facility manager. This is expressed by the property P_CIN_2 of this class.
Postreferences : msref ControlledInput : M_CIN2

Timed Function *value* → *INPUT_DOM*

Intention	: This function represents the currently valid value. It depends on the inputs of the facility manager and the input of a user.
Scope	: mh
Postreferences	: rsref ExpectedUsage : D_EU1
	msref ControlledInput : I_CIN2
	msref ControlledInput : M_CIN2
	msref ControlledDynamicInterval : M_CDI1
	msref RoomTemperature : M_RT2
	msref RoomTemperature : M_RT6
	msref RoomTemperature : M_RT7
	msref RoomTemperature : M_RT8
	msref RoomTemperature : M_RT11
	msref RoomTemperature : M_RT12

Timed Predicate *userValueEntered*

Intention	: This predicate is true iff the user has entered a value.
Scope	: mh
Postreferences	: msref ControlledInput : I_CIN1
	msref ControlledInput : M_CIN1
	msref ControlledInput : M_CIN2

Machine Specifications

Property *I_CIN1*

Formal	: $\neg userValueEntered$
NL	: Initially a value has not been entered by a user.

Property *I_CIN2*

Formal	: $value = default.checked$
NL	: The initially valid value is equal to the initial default value given by the facility manager.

Property *M_CIN1*

Formal	: $\Box(userValueEntered \rightarrow \Box userValueEntered)$
NL	: If a user has entered a value once, from this time point it will be valid that a value has been entered by a user.

Property *M_CIN2*

Formal : $\Box(userValueEntered \rightarrow$
$(allowedRange.leq(userValue.checked,$
$allowedRange.start.checked)$
$\rightarrow value = allowedRange.start.checked) \wedge$
$(allowedRange.leq(allowedRange.end.checked,$
$userValue.checked)$
$\rightarrow value = allowedRange.end.checked) \wedge$
$((\neg allowedRange.leq(userValue.checked,$
$allowedRange.start.checked) \wedge$
$\neg allowedRange.leq(allowedRange.end.checked,$
$userValue.checked))$
$\rightarrow value = userValue.checked))$

NL : If a user has entered a value which is less than or equal to the lower bound of the allowed interval specified by the facility manager the currently valid value is equal to the lower bound of the interval specified by the facility manager.
If a user has entered a value which is greater than the upper bound of the allowed interval specified by the facility manager the currently valid value is equal to the upper bound of the interval specified by the facility manager.
Otherwise, if the entered value is within the interval specified by the facility manager, the currently valid value is equal to the value entered by a user.

Property *M_CIN3*

Formal : $\Box(allowedRange.leq(allowedRange.start.checked,$
$default.checked) \wedge$
$allowedRange.leq(default.checked,$
$allowedRange.end.checked))$

NL : The default value given by the facility manager has always to be within the allowed range specified by the facility manager.

Description Class ControlledDynamicInterval

Intention

This class provides all definitions to describe a *controlled dynamic interval*. An interval is a subset of a specific domain and is given by a start (object *start*) and an end point (object *end*). The domain is given by the formal parameter *INT_DOM*. Furthermore, a predicate *leq* is provided indicating whether a value of the considered domain is less or equal to another value.
Dynamic means that the values of the start and end point of the interval can change over time. For both values the facility manager can prescribe a default value which is valid as long as a user has not entered a value. Therefore the start and end point are objects of the class *ControlledInput*.

Formal Parameters

Sort *INT_DOM*

Intention : This domain contains the values the start and end point of the interval have to belong to.

Signature

Object *start* : *ControlledInput*(*INPUT_DOM* = *INT_DOM*)

Intention : This object represents the starting point of the interval. The entered value has to be an element of the domain *INT_DOM*.
Postreferences : rsref ExpectedUsage : D_EU1
msref ControlledDynamicInterval : M_CDI1

Object *end* : *ControlledInput*(*INPUT_DOM* = *INT_DOM*)

Intention : This object represents the end point of the interval. The entered value has to be an element of the domain *INT_DOM*.
Postreferences : rsref ExpectedUsage : D_EU1
msref ControlledDynamicInterval : M_CDI1

Predicate *leq* : *INT_DOM*, *INT_DOM*

Intention : This predicate is true iff the first argument is *less* than or equal to the second argument.
Scope : mh
Postreferences : rsref ExpectedUsage : D_EU1
msref ControlledDynamicInterval : M_CDI1

Machine Specifications

Property *M_CDI1*

Formal : $\Box(leq(start.value, end.value))$
NL : The starting point of the interval is always less than or equal to the end point of the interval.

Group Sensors

Intention

This group contains all description classes that contain definitions related to sensors.

Description Class NoMalfunctionSensor

Intention

This class provides all definitions to describe the behavior of a *sensor* if it has *no malfunction*.

It is distinguished between three values:

- the value of the phenomenon existing in the real world (which can never be visible for the machine) (timed function *envEntity*),
- the analogously measured value of this real world value (timed function *measuredEntity*), and
- the digitally measured value which is a conversion of the analog measured value (timed function *convertedEntity*).

Note that this class can be used to describe a sensor that already exists in the environment. This is expressed as follows: At first the value measured by the sensor, namely *convertedEntity*, has the scope *environment controlled, not visible for the machine*. Secondly, the properties D_NMS_1 and D_NMS_2 describing the behavior of the sensor are characterized as domain knowledge. Otherwise, if the sensor does not exist, they have to be requirements.

Formal Parameters

Sort *ENV_DOMAIN*

Intention : This domain contains the possible values of the measured phenomenon existing in the real world.

Sort *MEASURED_DOMAIN*

Intention : This domain contains the possible analogously measured values.

Sort *CONVERTED_DOMAIN*

Intention : This domain contains the possible digitally converted measured values.

Signature

Time Constant *reactionTime*

Intention : This time constant represents the *reaction time* of the sensor. The *reaction time* is the time between a change of the value in the real world (*envEntity*) and the time point the measured entity (*measuredEntity*) has reached 90 percent of the change.
Scope : ev
Postreferences : dkref NoMalfunctionSensor : D_NMS1
dkref RoomTempSensor : D_RTS2
dkref OutdoorTempSensor : D_OTS2
dkref OutdoorLightSensor : D_OLS2
dkref WaterTempSensor : D_WTS2
dkref LightSensor : D_L1
dkref DoorContact : D_DC1
dkref MotionDetector : D_MD1

Time Constant *conversionTime*

Intention : This time constant represents the *conversion time* of the sensor. The *conversion time* is the time needed to convert an analogously measured value (*measuredEntity*) into the corresponding digital value (*convertedEntity*).
Scope : ev
Postreferences : dkref NoMalfunctionSensor : D_NMS2
dkref RoomTempSensor : D_RTS1
dkref OutdoorTempSensor : D_OTS1
dkref OutdoorLightSensor : D_OLS1
dkref WaterTempSensor : D_WTS1
dkref BinarySensor : D_BS1

Timed Function $envEntity \rightarrow ENV_DOMAIN$

Intention : This function represents the value of the measured phenomenon in the real world.
Note that this function represents the phenomenon as it is in the real world and not how it is seen by the machine.
Scope : eh
Postreferences : dkref NoMalfunctionSensor : D_NMS1
dkref SunBlind : D_SBL6
dkref SunBlind : D_SBL7
dkref ControlledLight : D_CL1
dkref ControlledLight : D_CL2
dkref AreaLight : D_AL2
rsref AreaLight : R_AL1
rsref AreaLight : R_AL2
rsref HallwaySection : R_HS1
rsref HallwaySection : R_HS2
rsref Floor : R_F1

Timed Function *measuredEntity* → *MEASURED_DOMAIN*

Intention	: This function represents the analogously measured value of the phenomenon existing in the real world.
Scope	: eh
Postreferences	: dkref NoMalfunctionSensor : D_NMS1 dkref NoMalfunctionSensor : D_NMS2

Timed Function *convertedEntity* → *CONVERTED_DOMAIN*

Intention	: This function represents the digitally converted value of the measured value.
Scope	: ev
Postreferences	: dkref NoMalfunctionSensor : D_NMS2 msref Sensor : M_S2 msref SunBlind : M_SBL1 msref SunBlind : M_SBL2 msref ControlledLight : M_CL1 msref BasicRoom : M_BaR1 msref BasicRoom : M_BaR2 msref BasicRoom : M_BaR3

Timed Function *modifyReaction* : *ENV_DOMAIN* → *MEASURED_DOMAIN*

Intention	: This function represents the way in which a value of the phenomenon in the real world (*envEntity*) is modified during the reaction of the sensor resulting in an analogously measured value *measuredEntity*.
Scope	: eh
Postreferences	: dkref NoMalfunctionSensor : D_NMS1 dkref BinarySensor : D_BS2

Timed Function *modifyConversion* : *MEASURED_DOMAIN* → *CONVERTED_DOMAIN*

Intention	: This function represents the way in which an analogously measured value *measuredEntity* is modified during the conversion into a digital value (*convertedEntity*).
Scope	: eh
Postreferences	: dkref NoMalfunctionSensor : D_NMS2 dkref BinarySensor : D_BS3

Domain Knowledge

Property *D_NMS1*

Formal	: $\Box((envEntity \rhd^{modifyReaction}_{reactionTime} measuredEntity))$
NL	: The analogously measured value is always derived from the value of the phenomenon in the real world using the function *modifyReaction*. A change of the value in the real world is always propagated within the time *reactionTime*.

Property *D_NMS2*

Formal	: $\Box((measuredEntity \rhd^{modifyConversion}_{conversionTime} convertedEntity))$
NL	: The digitally converted value is always derived from the analogously measured value using the function *modifyConversion*. A change of the value in the real world is always propagated within the time *conversionTime*.

Description Class MalfunctionSensor

Intention

This class provides all definitions to describe the behavior of a *sensor* if it has a *malfunction*.

In the case of a malfunction of a sensor, a value is determined depending on a possibly manually entered value by a user and an automatically determined value. If a value has been entered manually this value has priority over the automatically determined value.

Furthermore, the facility manager as well as the user are informed about a malfunction.

Formal Parameters

Sort *MAL_CONVERTED_DOMAIN*

Intention : This domain contains the possible values the machine can use as sensor values in the case of a malfunction of the sensor.

Signature

Object *manualEnteredVal* : *EnteredValue*(
CHECKED_DOM = *MAL_CONVERTED_DOMAIN*)

Intention : This object represents the value entered manually by a user in the case of a malfunction of this sensor. The entered value has to be an element of the domain MAL_CONVERTED_DOMAIN.
Postreferences : msref MalfunctionSensor : M_MS1

Time Constant *TinfDelay*

Intention : This time constant represents the maximal delay between an important event concerning this sensor a user or the facility manager has to be informed about (e.g. a malfunction or the repair of a malfunction) and the time point the user or the facility manager is informed.
Scope : ev
Postreferences : rsref MalfunctionSensor : R_MS3
rsref MalfunctionSensor : R_MS4
rsref MalfunctionSensor : R_MS5
rsref MalfunctionSensor : R_MS6

Time Constant *Tinfo*

Intention : This time constant represents the maximal time span a user or the facility manager is informed about an important event concerning this sensor.
Scope : ev
Postreferences : rsref MalfunctionSensor : R_MS1
rsref MalfunctionSensor : R_MS2

Timed Function *automaticDetermValue* → *MAL_CONVERTED_DOMAIN*

Intention : This function represents the value of the sensor determined automatically by the machine in the case of a malfunction of the sensor.
Scope : mh
Postreferences : msref MalfunctionSensor : M_MS1
msref BasicRoom : M_BaR1
msref BasicRoom : M_BaR2
msref BasicRoom : M_BaR3
rsref OutdoorLightSensor : R_OLS1

Timed Function *valueMal* → *MAL_CONVERTED_DOMAIN*

Intention : This function represents the value of the sensor used by the machine in the case of a malfunction of the sensor. It is computed based on the automatically determined and the manually entered value.
Scope : mh
Postreferences : msref MalfunctionSensor : M_MS1
msref Sensor : M_S3

Timed Predicate *malDetEnv*

Intention : This predicate is true iff the sensor existing in the real world has a malfunction. Note that this predicate represents the value as it exists in the real world. This value is not visible for the machine.
Scope : eh

Timed Predicate *malDetMach*

Intention : This predicate is true iff the machine assumes that the sensor existing in the real world has a malfunction. Note that this predicate represents the value as it seen by the machine.
Scope : mh
Postreferences : msref Sensor : M_S1
msref Sensor : M_S2
msref Sensor : M_S3
msref BasicRoom : M_BaR4
msref Floor : M_F1
rsref MalfunctionSensor : R_MS3
rsref MalfunctionSensor : R_MS4
rsref MalfunctionSensor : R_MS5
rsref MalfunctionSensor : R_MS6

Timed Predicate *stateManualEnteredValue*

Intention : This predicate is true iff a value, that should be used in the case of a malfunction of the sensor, has been entered manually by a user.
Scope : mh
Postreferences : msref MalfunctionSensor : I_MS1
msref MalfunctionSensor : M_MS1

Timed Predicate *FMInformed*

Intention	: This predicate is true iff the facility manager is informed about an *important event* concerning this sensor. An *important event* is either the detection of a malfunction of the sensor or that a malfunction of the sensor has been repaired. The facility manager has to be informed by sending an email.
Scope	: eh
Postreferences	: rsref MalfunctionSensor : I_MS3 rsref MalfunctionSensor : R_MS2 rsref MalfunctionSensor : R_MS4 rsref MalfunctionSensor : R_MS6

Timed Predicate *userInformed*

Intention	: This predicate is true iff the user is informed about an *important event* concerning this sensor. An *important event* is either the detection of a malfunction or that a malfunction has been repaired. The user is the person who is currently in the area in which the sensor is installed. The user has to be informed by sending an email.
Scope	: eh
Postreferences	: rsref MalfunctionSensor : I_MS2 rsref MalfunctionSensor : R_MS1 rsref MalfunctionSensor : R_MS3 rsref MalfunctionSensor : R_MS5

Requirement Specifications

Property *I_MS2*

Formal	: $\neg userInformed$
NL	: Initially the user is not informed about an important event concerning a sensor.

Property *I_MS3*

Formal	: $\neg FMInformed$
NL	: Initially the facility manager is not informed about an important event concerning a sensor.

Property *R_MS1*

Formal	: $\Box([userInformed] \rightarrow \Diamond_{\leq Tinfo} \neg userInformed)$
NL	: Whenever a user becomes informed he is informed for at most T_{info} milliseconds. By this requirement it is expressed, that the user is not informed permanently, but that the information is a kind of signal, e.g. an email.

Property *R_MS2*

Formal : $\Box([FMInformed] \rightarrow \Diamond_{\leq Tinfo} \neg FMInformed)$
NL : Whenever the facility manager becomes informed he is informed for at most T_{info} milliseconds.
By this requirement it is expressed, that the facility manager is not informed permanently, but that the information is a kind of signal, e.g. an email.

Property *R_MS3*

Formal : $\Box([malDetMach] \rightarrow \Diamond_{\leq TinfDelay} userInformed)$
NL : Whenever the machine assumes that a malfunction of the sensor has occurred, the user is informed about this within $T_{infDelay}$ milliseconds.

Property *R_MS4*

Formal : $\Box([malDetMach] \rightarrow \Diamond_{\leq TinfDelay} FMInformed)$
NL : Whenever the machine assumes that a malfunction of the sensor has occurred, the facility manager is informed about this within $T_{infDelay}$ milliseconds.

Property *R_MS5*

Formal : $\Box([\neg malDetMach] \rightarrow \Diamond_{\leq TinfDelay} userInformed)$
NL : Whenever the machine assumes that the sensor is working correctly again after it has had a malfunction, the user is informed about this within $T_{infDelay}$ milliseconds.

Property *R_MS6*

Formal : $\Box([\neg malDetMach] \rightarrow \Diamond_{\leq TinfDelay} FMInformed)$
NL : Whenever the machine assumes that the sensor is working correctly again after it has had a malfunction, the facility manager is informed about this within $T_{infDelay}$ milliseconds.

Machine Specifications

Property *I_MS1*

Formal : $\neg stateManualEnteredValue$
NL : Initially no value for the case of a malfunction is entered manually by a user.

Property *M_MS1*

Formal : $\Box(stateManualEnteredValue \rightarrow (valueMal = manualEnteredVal.checked) \wedge (\neg stateManualEnteredValue \rightarrow (valueMal = automaticDetermValue)))$
NL : If a value for the status of the sensor in the case of a malfunction has been entered manually this is the value used by the machine in the case of a malfunction of the sensor.
Otherwise, the value used by the machine in the case of a malfunction of the sensor is the value determined automatically in the case of a malfunction of the sensor.

Description Class Sensor

Intention

This class provides all definitions to describe the behavior of a *sensor*. It is composed of the behavior of a sensor if there is no malfunction and the case that there is a malfunction.

A malfunction of a sensor can be detected automatically by the machine or it can be entered manually.

Formal Parameters

Sort *ENV_DOMAIN*

Intention : This domain contains the possible values of the measured phenomenon existing in the real world in the case that a sensor has no malfunction.

Sort *MEASURED_DOMAIN*

Intention : This domain contains the possible analogously measured values in the case that a sensor has no malfunction.

Sort *CONVERTED_DOMAIN*

Intention : This domain contains the possible digitally converted measured values in the case that a sensor has no malfunction.

Signature

Object *noMalSens* : *NoMalfunctionSensor*(*ENV_DOMAIN* = *ENV_DOMAIN*,
MEASURED_DOMAIN = *MEASURED_DOMAIN*,
CONVERTED_DOMAIN = *CONVERTED_DOMAIN*)

Intention : This object represents the part of the sensor describing the behavior of the sensor if it has no malfunction, i.e. if the machine assumes that it has no malfunction.

Postreferences :
dkref RoomTempSensor : D_RTS1
dkref RoomTempSensor : D_RTS2
dkref OutdoorTempSensor : D_OTS1
dkref OutdoorTempSensor : D_OTS2
dkref OutdoorLightSensor : D_OLS1
dkref OutdoorLightSensor : D_OLS2
dkref WaterTempSensor : D_WTS1
dkref WaterTempSensor : D_WTS2
dkref BinarySensor : D_BS1
dkref BinarySensor : D_BS2
dkref BinarySensor : D_BS3
dkref LightSensor : D_L1
dkref DoorContact : D_DC1
dkref MotionDetector : D_MD1
dkref SunBlind : D_SBL6
dkref SunBlind : D_SBL7
dkref ControlledLight : D_CL1
dkref ControlledLight : D_CL2
dkref AreaLight : D_AL2
msref Sensor : M_S2
msref SunBlind : M_SBL1
msref SunBlind : M_SBL2
msref ControlledLight : M_CL1
msref BasicRoom : M_BaR1
msref BasicRoom : M_BaR2
msref BasicRoom : M_BaR3
rsref AreaLight : R_AL1
rsref AreaLight : R_AL2
rsref HallwaySection : R_HS1
rsref HallwaySection : R_HS2
rsref Floor : R_F1

Object *malSens* : *MalfunctionSensor*(
MAL_CONVERTED_DOMAIN = *CONVERTED_DOMAIN*)

Intention : This object represents the part of the sensor describing the behavior of the sensor if it has a malfunction, i.e. if the machine assumes that it has a malfunction.
The possible digitally converted measured values (*CONVERTED_DOMAIN*) are used as the possible values the machine can use as sensor values in the case of a malfunction of the sensor (*MAL_CONVERTED_DOMAIN*).

Postreferences : msref Sensor : M_S1
msref Sensor : M_S2
msref Sensor : M_S3
msref BasicRoom : M_BaR1
msref BasicRoom : M_BaR2
msref BasicRoom : M_BaR3
msref BasicRoom : M_BaR4
msref Floor : M_F1
rsref OutdoorLightSensor : R_OLS1

Object *manualEnteredMal* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention : This object represents a value entered manually by a person concerning a malfunction of the sensor. 1 stands for a malfunction, 0 stands for no malfunction.
Note that the value of this object is only meaningful if the timed predicate *stateManualEnteredMal* is true.

Postreferences : msref Sensor : M_S1

Timed Function *usedValue* → *CONVERTED_DOMAIN*

Intention : This function models the current value used by the machine as the value of the sensor. This depends on whether there is a malfunction of the sensor or not.

Scope : mh

Postreferences : msref Sensor : M_S2
msref Sensor : M_S3
msref AreaLight : M_AL8
msref HallwaySection : M_HS3
msref BasicRoom : M_BaR5
msref BasicRoom : M_BaR6

Timed Predicate *stateManualEnteredMal*

Intention	: This predicate is true iff it has been manually entered whether there is a malfunction of the sensor or whether there is no malfunction of it. Note the difference between this predicate and the object *manualEnteredMal*.
Scope	: mh
Postreferences	: msref Sensor : I_S1 msref Sensor : M_S1

Timed Predicate *automaticDetMal*

Intention	: This predicate is true iff a malfunction of the sensor has been detected automatically, i.e. not by entering it manually.
Scope	: mh
Postreferences	: msref Sensor : M_S1 msref SunBlind : M_SBL1 msref SunBlind : M_SBL2

Machine Specifications

Property *I_S1*

Formal	: $\neg stateManualEnteredMal$
NL	: Initially it has not been entered manually whether there is a malfunction of the sensor or whether there is no malfunction of it.

Property *M_S1*

Formal	: $\Box(malSens.malDetMach \leftrightarrow$ $(stateManualEnteredMal \wedge manualEnteredMal.checked = 1) \vee$ $(\neg stateManualEnteredMal \wedge automaticDetMal))$
NL	: The machine assumes a malfunction of the sensor iff one of the following conditions is fulfilled: • a malfunction has been entered manually or • it has not been entered manually whether there is a malfunction or not and a malfunction has been detected automatically.

Property *M_S2*

Formal	: $\Box(\neg malSens.malDetMach$ $\rightarrow usedValue = noMalSens.convertedEntity)$
NL	: Whenever the machine assumes that the sensor is working correctly the value used by the machine corresponds with the digitally converted version of the measured value.

Property *M_S3*

Formal	: $\Box(malSens.malDetMach \rightarrow usedValue = malSens.valueMal)$
NL	: Whenever the machine assumes that the sensor is not working correctly the value used by the machine corresponds with the value computed for this case.

Description Class RoomTempSensor

Intention

This class provides all definitions to describe the behavior of a *room temperature sensor*. This class is derived from the description class *Sensor*. The following specific parameters of a room temperature sensor are instantiated:

- the domains of the sensor
- the reaction and conversion time of the sensor

Both functions, *modifyReaction* and *modifyConversion*, have still to be provided by the domain expert.

Base Classes

Class *Sensor*(*ENV_DOMAIN* = *REAL*, *MEASURED_DOMAIN* = *REAL*, *CONVERTED_DOMAIN* = *ROOM_TEMP*)

Intention : All definitions of description class *Sensor* are also provided here using the real numbers as the possible values of the phenomenon existing in the real world, using the real numbers as the possible analogously measured values, and using the values of the domain *ROOM_TEMP* as the possible digitally converted measured values.

Signature

Domain $ROOM_TEMP = \{-40 \ldots 70\} \subseteq TEMP$

Intention : This domain contains the possible values measured by a room temperature sensor.
Postreferences : lsref RoomTempSensor : Sensor
lsref BasicRoom : valueEstimation

Domain Knowledge

Property *D_RTS1*

Formal : $\Box(noMalSens.conversionTime = 160000)$
NL : The conversion time of a room temperature sensor is 160 seconds.

Property *D_RTS2*

Formal : $\Box(noMalSens.reactionTime = 10000)$
NL : The reaction time of a room temperature sensor is 10 seconds.

Description Class OutdoorTempSensor

Intention

This class provides all definitions to describe the behavior of a *outdoor temperature sensor*. This class is derived from the description class *Sensor*. The following specific parameters of a outdoor temperature sensor are instantiated:

- the domains of the sensor
- the reaction and conversion time of the sensor

Both functions, *modifyReaction* and *modifyConversion*, have still to be provided by the domain expert.

Base Classes

Class *Sensor*(*ENV_DOMAIN* = *REAL*, *MEASURED_DOMAIN* = *REAL*, *CONVERTED_DOMAIN* = *OUT_TEMP*)

Intention : All definitions of description class *Sensor* are also provided here using the real numbers as the possible values of the phenomenon existing in the real world, using the real numbers as the possible analogously measured values, and using the the values of the domain *OUT_TEMP* as the possible digitally converted measured values.

Signature

Domain $OUT_TEMP = \{-40 \ldots 60\} \subseteq TEMP$

Intention : This domain contains the possible values measured by an outdoor temperature sensor.
Postreferences : lsref OutdoorTempSensor : Sensor

Domain Knowledge

Property *D_OTS1*

Formal : $\Box(noMalSens.conversionTime = 160000)$
NL : The conversion time of an outdoor temperature sensor is 160 seconds.

Property *D_OTS2*

Formal : $\Box(noMalSens.reactionTime = 120000)$
NL : The reaction time of an outdoor temperature sensor is 120 seconds.

Description Class OutdoorLightSensor

Intention

This class provides all definitions to describe the behavior of a *outdoor light sensor*. This class is derived from the description class *Sensor*. The following specific parameters of a outdoor light sensor are instantiated:

- the domains of the sensor
- the reaction and conversion time of the sensor

Both functions, *modifyReaction* and *modifyConversion*, have still to be provided by the domain expert.

Base Classes

Class *Sensor*(*ENV_DOMAIN* = *REAL*, *MEASURED_DOMAIN* = *REAL*, *CONVERTED_DOMAIN* = *OUT_LIGHT_VALUES*)

Intention : All definitions of description class *Sensor* are also provided here using the real numbers as the possible values of the phenomenon existing in the real world, using the real numbers as the possible analogously measured values, and using the values of the domain *OUT_LIGHT_VALUES* as the possible digitally converted measured values.

Signature

Domain $OUT_LIGHT_VALUES = \{1 \ldots 10000\} \subseteq LIGHT_VALUES$

Intention : This domain contains the possible values measured by an outdoor light sensor.
Postreferences : lsref OutdoorLightSensor : Sensor
lsref OutdoorLightSensor : lastCorrectOutLight

Timed Function $lastCorrectOutLight \rightarrow OUT_LIGHT_VALUES$

Intention : This function represents the last correctly measured value of the outdoor light intensity.
Scope : eh
Postreferences : rsref OutdoorLightSensor : R_OLS1

Domain Knowledge

Property *D_OLS1*

Formal : $\Box(noMalSens.conversionTime = 1000)$
NL : The conversion time of an outdoor light sensor is 1 second.

Property *D_OLS2*

Formal : $\Box(noMalSens.reactionTime = 10)$
NL : The reaction time of a outdoor light sensor is 10 milliseconds.

Requirement Specifications

Property *R_OLS1*

Formal	: $\Box(malSens.automaticDetermValue = lastCorrectOutLight)$
NL	: The value of the sensor determined automatically in the case of a malfunction of the outdoor light sensor corresponds with the last correctly measured value.

Description Class WaterTempSensor

Intention

This class provides all definitions to describe the behavior of a *water temperature sensor*. This class is derived from the description class *Sensor*. The following specific parameters of a water temperature sensor are instantiated:

- the domains of the sensor
- the reaction and conversion time of the sensor

Both functions, *modifyReaction* and *modifyConversion*, have still to be provided by the domain expert.

Base Classes

Class *Sensor*(*ENV_DOMAIN* = *REAL*, *MEASURED_DOMAIN* = *REAL*, *CONVERTED_DOMAIN* = *WATER_TEMP*)

Intention : All definitions of description class *Sensor* are also provided here using the real numbers as the possible values of the phenomenon existing in the real world, using the real numbers as the possible analogously measured values, and using the values of the domain *WATER_TEMP* as the possible digitally converted measured values.

Signature

Domain $WATER_TEMP = \{0 \ldots 100\} \subseteq TEMP$

Intention : This domain contains the possible values measured by a water temperature sensor.
Postreferences : lsref WaterTempSensor : Sensor

Domain Knowledge

Property *D_WTS1*

Formal : $\Box(noMalSens.conversionTime = 160000)$
NL : The conversion time of a water temperature sensor is 160 seconds.

Property *D_WTS2*

Formal : $\Box(noMalSens.reactionTime = 10000)$
NL : The reaction time of a water temperature sensor is 10 seconds.

Description Class BinarySensor

Intention

This class provides all definitions to describe the behavior of a *binary sensor*. This class is derived from the description class *Sensor*. The following specific parameters of a binary sensor are instantiated:

- the domains of the sensor: a binary sensor is a sensor that returns a 0 or a 1,
- the conversion time of the sensor,
- the transformation functions *modifyReaction* and *modifyConversion* are defined explicitely.

Base Classes

Class *Sensor*($ENV_DOMAIN = BINARY$,
$MEASURED_DOMAIN = BINARY$,
$CONVERTED_DOMAIN = BINARY$)

Intention : All definitions of description class *Sensor* are also provided here using the values 0 and 1 as the possible values of the phenomenon existing in the real world, using the values 0 and 1 as the possible analogously measured values, and using the values 0 and 1 as the possible digitally converted measured values.

Domain Knowledge

Property *D_BS1*

Formal : $\Box(noMalSens.conversionTime = 0)$
NL : The conversion time of a binary sensor is 0 milliseconds.

Property *D_BS2*

Formal : $\Box(\forall b \in BINARY : noMalSens.modifyReaction(b) = b)$
NL : For a binary sensor the function representing the way in which a value in the real world (*envEntity*) is modified during the reaction of the sensor, resulting in an analogously measured value (*measuredEntity*), is the identity.

Property *D_BS3*

Formal : $\Box(\forall b \in BINARY : noMalSens.modifyConversion(b) = b)$
NL : For a binary sensor the function representing the way in which an analogously measured value (*measuredEntity*) is modified during the conversion into a digital value (*convertedEntity*) is the identity.

Description Class LightSensor

Intention

This class provides all definitions of a sensor determining whether a light is on or not. Note that this is not a sensor measuring the illuminance. Therefore, *on* means that the circuit of the light is closed.

Base Classes

Class *BinarySensor*

Intention	: All definitions of description class *BinarySensor* are also provided here.

Domain Knowledge

Property *D_L1*

Formal	: $\Box(noMalSens.reactionTime = 10)$
NL	: The reaction time of a light sensor is 10 ms.

Description Class DoorContact

Intention

This class provides all definitions to describe the behavior of the different kinds of door contacts. These different kinds are *door closed contact*, *door open contact*, and *door lock contact*. A door contact is derived from a binary sensor. Here additionally the reaction time of a door contact is specified.

Base Classes

Class *BinarySensor*

Intention	: All definitions of description class *BinarySensor* are also provided here.

Domain Knowledge

Property *D_DC1*

Formal	: $\Box(noMalSens.reactionTime = 10)$
NL	: The reaction time of a door contact is 10 ms.

Description Class MotionDetector

Intention

This class provides all definitions to describe the behavior of a motion detector. A motion detector is derived from a binary sensor. Here additionally the reaction time of a motion detector is specified.

A motion detector returns a 1 if a person is moving in the range of the detector.

Base Classes

Class *BinarySensor*

Intention : All definitions of description class *BinarySensor* are also provided here.

Domain Knowledge

Property *D_MD1*

Formal : $\Box(noMalSens.reactionTime = 1000)$
NL : The reaction time of a motion detector is 1 s.

Group Actuator

Intention

This group contains all description classes that contain definitions related to actuators.

Description Class NoMalfunctionActuator

Intention

This class provides all definitions to describe an *actuator* if it has *no malfunction*.

It is distinguished between two values:

- the value of the phenomenon as it exists in the real world (*envEntity*) (which can never be visible for the machine or controlled directly by the machine), and
- the value adjusted by the machine (*adjustedEntity*).

Note that this class can be used to describe an actuator that already exists in the environment. Therefore the value adjusted by the actuator, namely *adjustedEntity*, has the scope *machine controlled, visible for the machine*. If the actuator does not exist, i.e. the actuator itself has to be developed, the scope of *adjustedEntity* has to be *environment controlled, not visible for the machine*.

Formal Parameters

Sort *ENV_DOMAIN*

Intention : This domain contains the possible values of the phenomenon existing in the real world.

Sort *ADJUSTED_DOMAIN*

Intention : This domain contains the possible values that can be adjusted by the machine.

Signature

Time Constant *reactionTime*

Intention : This time constant represents the *reaction time* of the actuator. The *reaction time* is the time between a change of the value adjusted by the machine (*adjustedEntity*) and the time point the value of the actuator in the real world (*envEntity*) changes its value *correspondingly*.

Scope : ev

Postreferences : dkref ToggleActuator : D_TA1
dkref ToggleActuator : D_TA2
dkref Pulse : D_PUL1
dkref RangeActuator : D_RA1
dkref Dimmer : D_DIM1
dkref ControlSystemActive : D_CSA1
dkref RadiatorValve : D_RV1
msref ControlledLight : M_CL1
msref ControlledLight : M_CL2

Timed Function *envEntity* → *ENV_DOMAIN*

Intention : This function represents the value of the phenomenon, i.e. the actuator, existing in the real world.
Note that this function represents the situation as it is in the real world and not how it is adjusted and seen by the machine.
Scope : eh
Postreferences : dkref ToggleActuator : D_TA1
dkref ToggleActuator : D_TA2
dkref RangeActuator : D_RA1
dkref SunBlind : D_SBL5
dkref SunBlind : D_SBL6
dkref SunBlind : D_SBL7
dkref SunBlind : D_SBL8
dkref SunBlind : D_SBL9
dkref ControlledLight : D_CL1

Timed Function *adjustedEntity* → *ADJUSTED_DOMAIN*

Intention : This function represents the value that is adjusted by the machine.
Scope : mv
Postreferences : dkref ToggleActuator : D_TA1
dkref ToggleActuator : D_TA2
dkref RangeActuator : D_RA1
msref SunBlind : M_SBL1
msref SunBlind : M_SBL2
msref ControlledLight : M_CL1
msref ControlledLight : M_CL2
msref ControlledLight : M_CL3
msref ControlledLight : M_CL4
msref AreaLight : M_AL8
msref AreaLight : M_AL10
msref RoomLight : M_RL8
msref RoomLight : M_RL9
msref RoomLight : M_RL10
msref RoomLight : M_RL11
msref Room : M_R3
msref BlindedRoom : M_BR4
msref BlindedRoom : M_BR5

Description Class MalfunctionActuator

Intention

This class provides all definitions to describe the behavior of an *actuator* if it has a *malfunction*.

The facility manager as well as the user are informed about a malfunction.

Signature

Time Constant *TinfDelay*

Intention	: This time constant represents the maximal delay between an important event concerning this actuator a user or the facility manager has to be informed about (e.g. a malfunction or the repair of a malfunction) and the time point the user or the facility manager is informed.
Scope	: ev
Postreferences	: rsref MalfunctionActuator : R_MA3 rsref MalfunctionActuator : R_MA4 rsref MalfunctionActuator : R_MA5 rsref MalfunctionActuator : R_MA6

Time Constant *Tinfo*

Intention	: This time constant represents the maximal time span a user or the facility manager is informed about an important event concerning this actuator.
Scope	: ev
Postreferences	: rsref MalfunctionActuator : R_MA1 rsref MalfunctionActuator : R_MA2

Timed Predicate *malDetEnv*

Intention	: This predicate is true iff the actuator existing in the real world has a malfunction. Note that this predicate represents the value as it exists in the real world. This value is not visible for the machine.
Scope	: eh

Timed Predicate *malDetMach*

Intention	: This predicate is true iff the machine assumes that the actuator existing in the real world has a malfunction. Note that this predicate represents the value as it seen by the machine.
Scope	: mh
Postreferences	: msref Actuator : M_A1 rsref MalfunctionActuator : R_MA3 rsref MalfunctionActuator : R_MA4 rsref MalfunctionActuator : R_MA5 rsref MalfunctionActuator : R_MA6

Timed Predicate *FMInformed*

Intention	: This predicate is true iff the facility manager is informed about an *important event* concerning this actuator. An *important event* is either the detection of a malfunction of the actuator or that a malfunction of the actuator has been repaired. The facility manager has to be informed by sending an email.
Scope	: eh
Postreferences	: rsref MalfunctionActuator : I_MA2 rsref MalfunctionActuator : R_MA2 rsref MalfunctionActuator : R_MA4 rsref MalfunctionActuator : R_MA6

Timed Predicate *userInformed*

Intention	: This predicate is true iff the user is informed about an *important event* concerning this actuator. An *important event* is either the detection of a malfunction of the actuator or that a malfunction of the actuator has been repaired. The user has to be informed by sending an email.
Scope	: eh
Postreferences	: rsref MalfunctionActuator : I_MA1 rsref MalfunctionActuator : R_MA1 rsref MalfunctionActuator : R_MA3 rsref MalfunctionActuator : R_MA5

Requirement Specifications

Property *I_MA1*

Formal	: $\neg userInformed$
NL	: Initially the user is not informed about an important event concerning this actuator.

Property *I_MA2*

Formal	: $\neg FMInformed$
NL	: Initially the facility manager is not informed about an important event concerning this actuator.

Property *R_MA1*

Formal	: $\Box([userInformed] \rightarrow \Diamond_{\leq Tinfo} \neg userInformed)$
NL	: Whenever a user becomes informed he is informed for at most T_{info} milliseconds. By this requirement it is expressed, that the user is not informed permanently, but that the information is a kind of signal, e.g. an email.

Property *R_MA2*

Formal	: $\Box([FMInformed] \rightarrow \Diamond_{\leq Tinfo} \neg FMInformed)$
NL	: Whenever the facility manager becomes informed he is informed for at most T_{info} milliseconds. By this requirement it is expressed, that the facility manager is not informed permanently, but that the information is a kind of signal, e.g. an email.

Property *R_MA3*

Formal	: $\Box([malDetMach] \rightarrow \Diamond_{\leq TinfDelay} userInformed)$
NL	: Whenever the machine assumes that a malfunction of the actuator has occurred, the user is informed about this within $T_{infDelay}$ milliseconds.

Property *R_MA4*

Formal	: $\Box([malDetMach] \rightarrow \Diamond_{\leq TinfDelay} FMInformed)$
NL	: Whenever the machine assumes that a malfunction of the actuator has occurred, the facility manager is informed about this within $T_{infDelay}$ milliseconds.

Property *R_MA5*

Formal	: $\Box([\neg malDetMach] \rightarrow \Diamond_{\leq TinfDelay} userInformed)$
NL	: Whenever the machine assumes that the actuator is working correctly again after it has had a malfunction, the user is informed about this within $T_{infDelay}$ milliseconds.

Property *R_MA6*

Formal	: $\Box([\neg malDetMach] \rightarrow \Diamond_{\leq TinfDelay} FMInformed)$
NL	: Whenever the machine assumes that the actuator is working correctly again after it has had a malfunction, the facility manager is informed about this within $T_{infDelay}$ milliseconds.

Description Class Actuator

Intention

This class provides all definitions to describe an *actuator*. It is composed of the behavior of an actuator if there is no malfunction and the case that there is a malfunction.

Formal Parameters

Sort *ENV_DOMAIN*

Intention : This domain contains the possible values of the phenomenon existing in the real world in the case that a sensor has no malfunction.

Sort *ADJUSTED_DOMAIN*

Intention : This domain contains the possible values that can be adjusted by the machine in the case that a sensor has no malfunction.

Signature

Object *noMalAct* : *NoMalfunctionActuator*(*ENV_DOMAIN* = *ENV_DOMAIN*, *ADJUSTED_DOMAIN* = *ADJUSTED_DOMAIN*)

Intention : This object represents the part of the actuator describing the behavior of the actuator if it has no malfunction, i.e. if the machine assumes that it has no malfunction.

Postreferences :
dkref ToggleActuator : D_TA1
dkref ToggleActuator : D_TA2
dkref Pulse : D_PUL1
dkref RangeActuator : D_RA1
dkref Dimmer : D_DIM1
dkref ControlSystemActive : D_CSA1
dkref RadiatorValve : D_RV1
dkref SunBlind : D_SBL5
dkref SunBlind : D_SBL6
dkref SunBlind : D_SBL7
dkref SunBlind : D_SBL8
dkref SunBlind : D_SBL9
dkref ControlledLight : D_CL1
msref SunBlind : M_SBL1
msref SunBlind : M_SBL2
msref ControlledLight : M_CL1
msref ControlledLight : M_CL2
msref ControlledLight : M_CL3
msref ControlledLight : M_CL4
msref AreaLight : M_AL8
msref AreaLight : M_AL10
msref RoomLight : M_RL8
msref RoomLight : M_RL9
msref RoomLight : M_RL10
msref RoomLight : M_RL11
msref Room : M_R3
msref BlindedRoom : M_BR4
msref BlindedRoom : M_BR5

Object *malAct* : *MalfunctionActuator*

Intention : This object represents the part of the actuator describing the behavior of the actuator if it has a malfunction, i.e. if the machine assumes that it has a malfunction.

Postreferences :
msref Actuator : M_A1

Object *manualEnteredMal* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention	: This object represents a value entered manually by a user concerning a malfunction of the actuator. 1 stands for a malfunction, 0 stands for no malfunction. Note that the value of this object is only meaningful if the timed predicate stateManualEnteredMal is true.
Postreferences	: msref Actuator : M_A1

Timed Predicate *stateManualEnteredMal*

Intention	: This predicate is true iff it has been manually entered whether there is a malfunction of the actuator or whether there is no malfunction of it. Note the difference between this predicate and the object *manualEnteredMal*.
Scope	: mh
Postreferences	: msref Actuator : I_A1 msref Actuator : M_A1

Timed Predicate *automaticDetMal*

Intention	: This predicate is true iff a malfunction of the actuator has been detected automatically, i.e. not by entering it manually.
Scope	: mh
Postreferences	: msref Actuator : M_A1 msref SunBlind : M_SBL1 msref SunBlind : M_SBL2

Machine Specifications

Property *I_A1*

Formal	: $\neg stateManualEnteredMal$
NL	: Initially it has not been entered manually whether there is a malfunction of the actuator or whether there is no malfunction of it.

Property *M_A1*

Formal	: $\Box(malAct.malDetMach \leftrightarrow$ $(stateManualEnteredMal \wedge manualEnteredMal.checked = 1) \vee$ $(\neg stateManualEnteredMal \wedge automaticDetMal))$
NL	: The machine assumes a malfunction of the actuator iff one of the following conditions is fulfilled: • a malfunction has been entered manually • it has not been entered manually that there is no malfunction and a malfunction has been detected automatically

Description Class ToggleActuator

Intention

This class provides all definitions to describe a toggle actuator. A toggle actuator is an actuator that can send a signal. This is modeled here by setting the adjusted value to 1. If there is no signal the adjusted value is 0.

Note that this class can be used to describe an actuator that already exists in the environment. Therefore, the properties D_TA_1 and D_TA_2 are characterized as domain knowledge. Otherwise, if the actuator does not exist, they have to be requirements/specifications.

Base Classes

Class *Actuator*($ENV_DOMAIN = BINARY$, $ADJUSTED_DOMAIN = BINARY$)

Intention : All definitions of description class *Actuator* are also provided here using the values 0 and 1 as the possible values of the phenomenon existing in the real world and using the values 0 and 1 as the possible values that can be adjusted by the machine.

Domain Knowledge

Property *D_TA1*

Formal : $\Box((noMalAct.envEntity = 1) \wedge [noMalAct.adjustedEntity = 1] \rightarrow \Diamond_{\leq noMalAct.reactionTime}(noMalAct.envEntity = 0))$

NL : If the value of the entity in the real world (*envEntity*) is 1 and a toggle signal arrives, i.e. the adjusted value (*adjustedEntity*) changes from 0 to 1, then within *reactionTime* milliseconds the value of the entity in the real world (*envEntity*) is 0.

Property *D_TA2*

Formal : $\Box((noMalAct.envEntity = 0 \wedge [noMalAct.adjustedEntity = 1]) \rightarrow \Diamond_{\leq noMalAct.reactionTime}(noMalAct.envEntity = 1))$

NL : If the value of the entity in the real world (*envEntity*) is 0 and a toggle signal arrives, i.e. the adjusted value (*adjustedEntity*) changes from 0 to 1, then within *reactionTime* milliseconds the value of the entity in the real world (*envEntity*) is 1.

Description Class Pulse

Intention

This class provides all definitions to describe the *pulse line* of a light by which it can be switched on and off. This class is derived from the description class *ToggleActuator*. Here, the reaction time of a pulse is specified.

Base Classes

Class *ToggleActuator*

Intention	: All definitions of description class *ToggleActuator* are also provided here.

Domain Knowledge

Property *D_PUL1*

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of a pulse line of a light is 10 milliseconds.

Description Class RangeActuator

Intention

This class provides all definitions to describe a *range actuator*. A range actuator is an actuator where a range of values, i.e. a set containing at least two values, can be adjusted. Consider in contrast to this the description class ToggleActuator.

Note that this class can be used to describe an actuator that already exists in the environment. Therefore, the property D_RA_1 is characterized as domain knowledge. Otherwise, if the actuator does not exist, it has to be a requirement/specification.

Formal Parameters

Sort *ENV_DOMAIN*

Intention : This domain contains the possible values of the phenomenon existing in the real world in the case that a sensor has no malfunction.

Sort *ADJUSTED_DOMAIN*

Intention : This domain contains the possible values that can be adjusted by the machine in the case that a sensor has no malfunction.

Base Classes

Class *Actuator*(*ENV_DOMAIN* = *ENV_DOMAIN*, *ADJUSTED_DOMAIN* = *ADJUSTED_DOMAIN*)

Intention : All definitions of description class *Actuator* are also provided here.

Signature

Timed Function *modifyReaction* : $ADJUSTED_DOMAIN \rightarrow ENV_DOMAIN$

Intention : This function represents the way in which a value adjusted by the machine (*adjustedEntity*) is modified during the reaction of the actuator resulting in the value of the entity in the real world (*envEntity*).

Scope : eh

Postreferences : dkref RangeActuator : D_RA1
dkref BinaryActuator : D_BA1
dkref ThreePointActuator : D_TPA1

Domain Knowledge

Property *D_RA1*

Formal : $\Box((noMalAct.adjustedEntity \rhd^{modifyReaction}_{noMalAct.reactionTime} noMalAct.envEntity))$

NL : The value of the phenomenon in the real world is always derived from the value adjusted by the machine using the function *modifyReaction*. A change of the value adjusted by the machine is always propagated within the time *reactionTime*.

Description Class Dimmer

Intention

This class provides all definitions to describe the *dimmer* of a dimmable light. This class is derived from the description class *RangeActuator*. The following specific parameters of a dimmer are instantiated:

- the domains of the sensor
- the reaction time

The function *modifyReaction* has still to be provided by the domain expert.

Base Classes

Class *RangeActuator*(*ENV_DOMAIN* = *PERCENT*,
ADJUSTED_DOMAIN = *PERCENT*)

Intention	: All definitions of description class *RangeActuator* are also provided here using the values 0 to 100 (percent) as the possible values of the phenomenon existing in the real world and using the values 0 to 100 (percent) as the possible values that can be adjusted by the machine.

Domain Knowledge

Property *D_DIM1*

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of a *dimmer* is 10 milliseconds.

Description Class BinaryActuator

Intention

This class provides all definitions to describe a *binary actuator*. This class is derived from the description class *RangeActuator*. The following specific parameters of a binary actuator are instantiated:

- the domains of the actuator: a binary actuator is a range actuator that can adjust the values 0 and 1,
- the transformation function *modifyReaction*

Note that this class can be used to describe an actuator that already exists in the environment. Therefore, the property D_BA_1 is characterized as domain knowledge. Otherwise, if the actuator does not exist, it has to be a requirement/specification.

Base Classes

Class *RangeActuator*($ENV_DOMAIN = BINARY$, $ADJUSTED_DOMAIN = BINARY$)

Intention : All definitions of description class *RangeActuator* are also provided here using the values 0 and 1 as the possible values of the entity existing in the real world and using the values 0 and 1 as the possible values that can be adjusted by the machine.

Domain Knowledge

Property *D_BA1*

Formal : $\Box(\forall b \in BINARY : modifyReaction(b) = b)$

NL : For a binary actuator the function representing the way in which a value adjusted by the machine (*adjustedEntity*) is modified during the reaction of the actuator, resulting in a value of the actuator in the real world (*envEntity*), is the identity.

Description Class ControlSystemActive

Intention

This class provides all definitions to describe the signal *control system is active* that can be sent by the machine in order to indicate that the machine is still active. In addition to the definitions of the description class *BinaryActuator*, here the reaction time of this signal is specified.

Base Classes

Class *BinaryActuator*

Intention	: All definitions of description class *BinaryActuator* are also provided here.

Domain Knowledge

Property *D_CSA1*

Formal	: $\Box(noMalAct.reactionTime = 10)$
NL	: The reaction time of a signal *control system is active* is 10 ms.

Description Class ThreePointActuator

Intention

This class provides all definitions to describe a *three point actuator*. This class is derived from the description class *RangeActuator*. The following specific parameters of a three point actuator are instantiated:

- the domains of the actuator: a three point actuator is an actuator which can adjust three control states: *forward*, *backward*, and *stop*,
- the transformation function *modifyReaction*

Note that this class can be used to describe an actuator that already exists in the environment. Therefore, the property D_TPA_1 is characterized as domain knowledge. Otherwise, if the actuator does not exist, it has to be a requirement/specification.

Base Classes

Class *RangeActuator*($ENV_DOMAIN = THREE_PT_ACT_RANGE$, $ADJUSTED_DOMAIN = THREE_PT_ACT_RANGE$)

Intention : All definitions of description class *RangeActuator* are also provided here using the values *forward*, *backward*, and *stop* as the possible values of the actuator existing in the real world and using the values *forward*, *backward*, and *stop* as the possible values that can be adjusted by the machine.

Domain Knowledge

Property *D_TPA1*

Formal : $\Box(\forall s \in THREE_PT_ACT_RANGE : modifyReaction(s) = s)$

NL : For a three point actuator the function representing the way in which a value adjusted by the machine (*adjustedEntity*) is modified during the reaction of the actuator, resulting in a value of the actuator in the real world (*envEntity*), is the identity.

Description Class RadiatorValve

Intention

This class provides all definitions to describe a radiator valve. In addition to the definitions of the description class *ToggleActuator*, here the reaction time of a radiator valve is specified.

Base Classes

Class *ThreePointActuator*

Intention	: All definitions of description class *ThreePointActuator* are also provided here.

Domain Knowledge

Property *D_RV1*

Formal	: $\Box(noMalAct.reactionTime = 60000)$
NL	: The reaction time of a radiator valve is 60 seconds.

Description Class PushButton

Intention

This class provides all definitions concerning the control of a light using pushbuttons. The number of pushbuttons is not relevant for the behavior. Note further that the push buttons are not visible for the machine.

Signature

Time Constant *T_PB*

Intention	: This time constant represents the delay between a button push and the change of the light.
Scope	: ev
Postreferences	: dkref PushButton : D_PB1 dkref PushButton : D_PB2

Timed Predicate *light*

Intention	: This predicate represents the status of the light existing in the real world. The predicate is true iff the light is on. Note that *on* here means that the circuit of the light is closed.
Scope	: eh
Postreferences	: dkref PushButton : D_PB1 dkref PushButton : D_PB2 dkref ControlledLight : D_CL2

Timed Predicate *pushBut*

Intention	: This predicate represents the status of the push buttons existing in the real world. The predicate is true iff one push button is pushed.
Scope	: eh
Postreferences	: dkref PushButton : D_PB1 dkref PushButton : D_PB2

Domain Knowledge

Property *D_PB1*

Formal	: $\Box(\neg light \wedge [pushBut] \rightarrow \Diamond_{\leq T_PB} light)$
NL	: Whenever the light is off and any push button is pressed, the light changes to on within time T_PB.

Property *D_PB2*

Formal	: $\Box(light \wedge [pushBut] \rightarrow \Diamond_{\leq T_PB} \neg light)$
NL	: Whenever the light is on and any push button is pressed, the light changes to off within time T_PB.

Group CombinedSensorsAndActuators

Intention

This group contains all description classes in which several sensors or actuators are combined.

Description Class Door

Intention

This class provides all definitions to describe a door. A door contains three door contacts.

Signature

Object *doorClosedContact* : *DoorContact*

Intention : This object represents the closed contact of a door. It is true iff the door is completely closed.

Object *doorOpenContact* : *DoorContact*

Intention : This object represents the open contact of a door. It is true iff the door is open for at least 30 degree.

Object *doorLockContact* : *DoorContact*

Intention : This object represents the lock contact of a door. It is true iff the door is locked by a key.

Description Class SunBlind

Intention

This class provides all definitions to describe a *sun blind*. A sun blind combines a *sun blind lifter* and a *blind rotator*. Both, lifter and rotator, are controlled using the same three point actuator. Additionally, a sun blind contains two sensors recognizing whether the blind has reached the upper or lower end.

To model the lifting of the blind and the rotating of the blind blades several states are introduced: *moveUp*, *moveDown*, *moveStop*, *rotateStatus*, *rotateIncrease*, *rotateDecrease*, and *rotateStop*.

Signature

Domain $DEGREE = \{0 \ldots 180\} \subseteq NAT$

Intention	: This domain contains the possible values of the blind rotator, namely 0 to 180 degrees.
Postreferences	: lsref SunBlind : rotateStatus

Object *blind* : *ThreePointActuator*

Intention	: This object represents the actuator by which the blind can be lifted and rotated.
Postreferences	: dkref SunBlind : D_SBL5 dkref SunBlind : D_SBL6 dkref SunBlind : D_SBL7 dkref SunBlind : D_SBL8 dkref SunBlind : D_SBL9 msref SunBlind : M_SBL1 msref SunBlind : M_SBL2 msref BlindedRoom : M_BR4 msref BlindedRoom : M_BR5

Object *upperEndContact* : *BinarySensor*

Intention	: This object represents the end contact detecting whether the blind is lifted completely up or not.
Postreferences	: dkref SunBlind : D_SBL6 msref SunBlind : M_SBL2

Object *lowerEndContact* : *BinarySensor*

Intention	: This object represents the end contact detecting whether the blind is lifted completely down or not.
Postreferences	: dkref SunBlind : D_SBL7 msref SunBlind : M_SBL1

Time Constant *Tmal*

Intention : This time constant is used to detect automatically a malfunction of a sun blind. See property P_SBL_1 for details.
Scope : ev
Postreferences : msref SunBlind : M_SBL1
msref SunBlind : M_SBL2

Timed Function *rotateStatus* $\rightarrow$ *DEGREE*

Intention : This function represents the current state of the blind rotator, i.e. the current position of the blind blades.
Scope : eh
Postreferences : dkref SunBlind : D_SBL3
dkref SunBlind : D_SBL4
dkref SunBlind : D_SBL6
dkref SunBlind : D_SBL7
dkref SunBlind : D_SBL8
dkref SunBlind : D_SBL9

Timed Predicate *moveUp*

Intention : This predicate is true iff the blind is moving up.
Scope : eh
Postreferences : dkref SunBlind : D_SBL1
dkref SunBlind : D_SBL4
dkref SunBlind : D_SBL6

Timed Predicate *moveDown*

Intention : This predicate is true iff the blind is moving down.
Scope : eh
Postreferences : dkref SunBlind : D_SBL1
dkref SunBlind : D_SBL3
dkref SunBlind : D_SBL7

Timed Predicate *moveStop*

Intention : This predicate is true iff the blind stops, i.e. it is neither moving down nor up.
Scope : eh
Postreferences : dkref SunBlind : D_SBL1
dkref SunBlind : D_SBL5
dkref SunBlind : D_SBL8
dkref SunBlind : D_SBL9

Timed Predicate *rotateIncrease*

Intention : This predicate is true iff the blind blades are rotated in the direction of 0 degree.
Scope : eh
Postreferences : dkref SunBlind : D_SBL2
dkref SunBlind : D_SBL8

Timed Predicate *rotateDecrease*

Intention	: This predicate is true iff the blind blades are rotated in the direction of 180 degree.
Scope	: eh
Postreferences	: dkref SunBlind : D_SBL2 dkref SunBlind : D_SBL9

Timed Predicate *rotateStop*

Intention	: This predicate is true iff the blind blades do not rotate in any direction. Note that this does not implies that the current rotating angle of the blades is 0 or 180 degrees.
Scope	: eh
Postreferences	: dkref SunBlind : D_SBL2 dkref SunBlind : D_SBL5 dkref SunBlind : D_SBL6 dkref SunBlind : D_SBL7

Domain Knowledge

Property *D_SBL1*

Formal	: $\Box((moveUp \leftrightarrow (\neg moveDown \wedge \neg moveStop)) \wedge$ $(moveDown \leftrightarrow (\neg moveUp \wedge \neg moveStop)) \wedge$ $(moveStop \leftrightarrow (\neg moveUp \wedge \neg moveDown)))$
NL	: The blind lifter is always in exactly one of the three possible states *moveUp*, *moveDown*, *moveStop*.

Property *D_SBL2*

Formal	: $\Box((rotateIncrease \leftrightarrow (\neg rotateDecrease \wedge \neg rotateStop)) \wedge$ $(rotateDecrease \leftrightarrow (\neg rotateIncrease \wedge \neg rotateStop)) \wedge$ $(rotateStop \leftrightarrow (\neg rotateIncrease \wedge \neg rotateDecrease)))$
NL	: The blind rotator is always in exactly one of the three possible states *rotateIncrease*, *rotateDecrease*, *rotateStop*.

Property *D_SBL3*

Formal	: $\Box(moveDown \rightarrow rotateStatus = 0)$
NL	: If the blinds are moved down the rotating state, i.e. the angle of the blades is 0 degree.

Property *D_SBL4*

Formal	: $\Box(moveUp \rightarrow rotateStatus = 180)$
NL	: If the blinds are moved up the rotating state, i.e. the angle of the blades is 180 degrees.

Property *D_SBL5*

Formal	: $\Box(blind.noMalAct.envEntity = STOP \rightarrow moveStop \wedge rotateStop)$
NL	: Whenever the status of the sun blind is set to stop, the blind is neither lifted in any direction nor are the blades rotated in any direction.

Property *D_SBL6*

Formal : $\Box(((blind.noMalAct.envEntity = FORWARD) \wedge$
$(rotateStatus = 180) \wedge$
$upperEndContact.noMalSens.envEntity = 0)$
$\rightarrow (moveUp \wedge rotateStop))$

NL : Whenever the status of the sun blind is set to FORWARD and the rotating state of the blades, i.e. the angle, is 180 degrees and the blind is not at the upper end, the blind is lifted up and the blades are not rotating.

Property *D_SBL7*

Formal : $\Box(((blind.noMalAct.envEntity = BACKWARD) \wedge$
$(rotateStatus = 0) \wedge$
$lowerEndContact.noMalSens.envEntity = 0)$
$\rightarrow (moveDown \wedge rotateStop))$

NL : Whenever the status of the sun blind is set to BACKWARD and the rotating state of the blades, i.e. the angle, is 0 degrees and the blind is not at the lower end, the blind is lifted down and the blades are not rotating.

Property *D_SBL8*

Formal : $\Box(((blind.noMalAct.envEntity = FORWARD) \wedge$
$\neg(rotateStatus = 180))$
$\rightarrow (moveStop \wedge rotateIncrease))$

NL : Whenever the status of the sun blind is set to FORWARD and the rotating state of the blades, i.e. the angle, is not 180 degrees the blind is not lifted up or down in any direction but rotated in the direction of 180 degrees.

Property *D_SBL9*

Formal : $\Box(((blind.noMalAct.envEntity = BACKWARD) \wedge$
$\neg(rotateStatus = 0))$
$\rightarrow (moveStop \wedge rotateDecrease))$

NL : Whenever the status of the sun blind is set to BACKWARD and the rotating state of the blades, i.e. the angle, is not 0 degree the blind is not lifted up or down in any direction but rotated in the direction of 0 degrees.

Property *D_SBL10*

NonFormal : If the rotating angle of the blind blades is x degrees light can enter from an angle of x degrees from vertical.

Machine Specifications

Property *M_SBL1*

Formal : $\Box((\blacksquare_{\geq Tmal}(blind.noMalAct.adjustedEntity = FORWARD \wedge lowerEndContact.noMalSens.convertedEntity = 1)) \rightarrow blind.automaticDetMal \vee lowerEndContact.automaticDetMal)$

NL : Whenever the machine has set the actuator to FORWARD, i.e. to move up, for at least T_{mal} time units, and the *lowerEndContact* constantly signals for this time that the blind is still at the lower end, something is wrong.

Property *M_SBL2*

Formal : $\Box((\blacksquare_{\geq Tmal}(blind.noMalAct.adjustedEntity = BACKWARD \wedge upperEndContact.noMalSens.convertedEntity = 1)) \rightarrow blind.automaticDetMal \vee upperEndContact.automaticDetMal)$

NL : Whenever the machine has set the actuator to BACKWARD, i.e. to move down, for at least T_{mal} time units, and the *upperEndContact* constantly signals for this time that the blind is still at the upper end, something is wrong.

Description Class Radiator

Intention

This class provides all the definitions to describe a radiator. A radiator consists of a radiator valve and of three water temperature sensors measuring the input, mean, and output water temperature, respectively.

Signature

Object *waterInTemp* : *WaterTempSensor*

Intention : This object represents a sensor to measure the input water temperature of a radiator.

Object *waterOutTemp* : *WaterTempSensor*

Intention : This object represents a sensor to measure the output water temperature of a radiator.

Object *meanTemp* : *WaterTempSensor*

Intention : This object represents a sensor to measure the mean radiator temperature of a radiator.

Object *valveAct* : *RadiatorValve*

Intention : This object represents the radiator control valve of a radiator.
Postreferences : msref Room : M_R3

Description Class ControlledLight

Intention

This description class provides all definitions of sensors and actuators relevant for the control of a light that is not dimmable. This installation includes:

- a light sensor indicating whether the light is on or off (object *ls*)
- an actuator allowing the machine to switch on and off the light (object *pulse*)
- an actuator allowing the machine to signal that it is still active (object *csa*)
- a push button allowing a person to switch on and off the light (object *pb*)
  Note that a push of a button is not directly visible for the machine but only by the light sensor.

Furthermore, it is defined under which conditions a light is controlled *manually*.

Signature

Object *ls* : *LightSensor*

Intention	: This object represents the light sensor indicating whether the light is on or off.
Postreferences	: dkref ControlledLight : D_CL1 dkref ControlledLight : D_CL2 dkref AreaLight : D_AL2 msref ControlledLight : M_CL1 msref AreaLight : M_AL8 rsref AreaLight : R_AL1 rsref AreaLight : R_AL2 rsref HallwaySection : R_HS1 rsref HallwaySection : R_HS2 rsref Floor : R_F1

Object *pulse* : *Pulse*

Intention	: This object represents the actuator the machine uses to switch the light on or off.
Postreferences	: dkref ControlledLight : D_CL1 msref ControlledLight : M_CL1 msref ControlledLight : M_CL2 msref AreaLight : M_AL8 msref AreaLight : M_AL10 msref RoomLight : M_RL8 msref RoomLight : M_RL9 msref RoomLight : M_RL10

Object *csa* : *ControlSystemActive*

Intention	: This object represents the actuator the machine uses to signal that it is still active.
Postreferences	: msref ControlledLight : M_CL3 msref ControlledLight : M_CL4

Object *pb* : *PushButton*

Intention	: This object represents the actuatori.e. the push button, a person can use to switch the light on or off. Note that a push of a button is not directly visible for the machine.
Postreferences	: dkref ControlledLight : D_CL2

Timed Predicate *manualControl*

Intention	: This predicate is true iff the machine assumes that the light is controlled manually.
Scope	: mh
Postreferences	: msref ControlledLight : I_CL1 msref ControlledLight : M_CL1 msref ControlledLight : M_CL2 msref AreaLight : M_AL9

Domain Knowledge

Property *D_CL1*

Formal	: $\Box(ls.noMalSens.envEntity = pulse.noMalAct.envEntity)$
NL	: The phenomenon in the real world which is measured by the light sensor is identical with the phenomenon adjusted by the pulse actuator.

Property *D_CL2*

Formal	: $\Box((ls.noMalSens.envEntity = 1) \leftrightarrow pb.light)$
NL	: The phenomenon in the real world which is measured by the light sensor is identical with the phenomenon adjusted by the push buttons.

Machine Specifications

Property *I_CL1*

Formal	: $\neg manualControl$
NL	: Initially the light is not controlled manually.

Property *M_CL1*

Formal	: $\Box((\neg \blacklozenge_{\leq pulse.noMalAct.reactionTime}[pulse.noMalAct.adjustedEntity = 1] \wedge ([ls.noMalSens.convertedEntity = 1] \vee [ls.noMalSens.convertedEntity = 0])) \rightarrow manualControl)$
NL	: If no pulse has been sent by the machine for the last *pulse.noMalAct.reactionTime* milliseconds and the measured value of the light sensor changes the light is controlled manually.

Property *M_CL2*

Formal : $\Box(\blacklozenge_{\leq pulse.noMalAct.reactionTime}[pulse.noMalAct.adjustedEntity = 1] \rightarrow \neg manualControl)$

NL : If a pulse has been sent by the machine within the last *pulse.noMalAct.reactionTime* milliseconds the light is not controlled manually.

Property *M_CL3*

Formal : $\Box([csa.noMalAct.adjustedEntity = 1] \rightarrow \Diamond_{\leq 1000} [csa.noMalAct.adjustedEntity = 0])$

NL : Whenever the signal *control system is active* is set to 1 it is set within one second to 0.

Property *M_CL4*

Formal : $\Box([csa.noMalAct.adjustedEntity = 0] \rightarrow \Diamond_{\leq 59000} [csa.noMalAct.adjustedEntity = 1])$

NL : Whenever the signal *control system is active* is set to 0 it is set within 59 seconds to 1.

Description Class DimmableControlledLight

Intention

This description class provides all definitions of sensors and actuators relevant for the control of a dimmable light. This installation includes in addition to a controlled light:

- a dimmer, i.e. an actuator for the control of the illuminance of the light if it is on (object *dimmer*).
  Note that this dimmer can not switch on or off the light, i.e. even if the dimmer value is zero the light can have the status on, i.e. the circuit of the light is closed. Thus, the dimmer sets the illuminance in the case that the light is on.

Base Classes

Class *ControlledLight*

Intention	: All definitions of description class *BinarySensor* are also provided here.

Signature

Object *dimmer* : *Dimmer*

Intention	: This object represents the light actuator, i.e. the dimmer, the machine uses to set the illuminance of a light.
Postreferences	: msref RoomLight : M_RL8 msref RoomLight : M_RL9 msref RoomLight : M_RL10 msref RoomLight : M_RL11

Group HumanMachineInterface

Intention

This group contains all description classes that contain definitions representing the interface between the machine and a person. These description classes are on a more abstract level than the description classes of group *BasicHumanMachineInterface*.

Description Class OccDelayTimes

Intention

This class provides all definitions to describe the values of the three time delays dt_1, dt_2, dt_3, (see description class *NoMalfunctionOccupancy*) used to determine occupancy for the temperature control.

The values of the three time delays depend on the inputs given by a user and the facility manager. These time delays can be set for different expected *usage profiles* of an area (see description class *TemperatureCtrlPanel*).

This description class *OccDelayTimes* provides a general frame to deal with the three time delays in a common setting.

Signature

Domain *DT1_RANGE* $\subseteq$ *TIME*

Intention : This domain contains the values that can be used for a time delay dt_1.
Postreferences : lsref OccDelayTimes : dt1

Domain *DT2_RANGE* $\subseteq$ *TIME*

Intention : This domain contains the values that can be used for a time delay dt_2.
Postreferences : lsref OccDelayTimes : dt2

Domain *DT3_RANGE* $\subseteq$ *TIME*

Intention : This domain contains the values that can be used for a time delay dt_3.
Postreferences : lsref OccDelayTimes : dt3

Object *dt1* : *ControlledInput*(*INPUT_DOM* = *DT1_RANGE*)

Intention : This object represents the current value of the time delay dt_1 (see description class *NoMalfunctionOccupancy*) depending on the values entered by the facility manager and the room user.
Postreferences : msref OccDelayTimes : M_ODT1
msref RoomTemperature : M_RT6
msref RoomTemperature : M_RT11
msref RoomTemperature : M_RT12

Object *dt2* : *ControlledInput*(*INPUT_DOM* = *DT2_RANGE*)

Intention : This object represents the current value of the time delay dt_2 (see description class *NoMalfunctionOccupancy*) depending on the values entered by the facility manager and the room user.
Postreferences : msref OccDelayTimes : M_ODT2
msref RoomTemperature : M_RT7
msref RoomTemperature : M_RT11
msref RoomTemperature : M_RT12

Object *dt3* : *ControlledInput*(*INPUT_DOM* = *DT3_RANGE*)

Intention : This object represents the current value of the time delay dt_3 (see description class *NoMalfunctionOccupancy*) depending on the values entered by the facility manager and the room user.

Postreferences : msref OccDelayTimes : M_ODT3
msref RoomTemperature : M_RT8
msref RoomTemperature : M_RT11
msref RoomTemperature : M_RT12

Machine Specifications

Property *M_ODT1*

Formal : *dt1.default.checked* = 60000

NL : The initial default value of time dt_1 given by the facility manager is 1 minute.

Property *M_ODT2*

Formal : *dt2.default.checked* = 600000

NL : The initial default value of time dt_2 given by the facility manager is 10 minutes.

Property *M_ODT3*

Formal : *dt3.default.checked* = 1800000

NL : The initial default value of time dt_3 given by the facility manager is 30 minutes.

Description Class AreaLightCtrlPanelFM

Intention

This class provides all the definitions of entities concerning the control of the light in an area for which the facility manager can enter a value using the control panel. An area is here the generalization of rooms and hallways. This is

- a possibility to switch on or off all lights in a certain area.
- a possibility to determine the time a certain area has to be unoccupied continuously before the lights are switched off.

Signature

Object *lightOn* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention	: This object represents whether the facility manager has turned all lights on or off in a certain area. *lightOn.checked* = 1 means that the lights have to be on; *lightOn.checked* = 0 means that the lights have to be off.
Postreferences	: msref AreaLightCtrlPanelFM : I_LCPFM2 rsref AreaLight : R_AL1

Object *unoccDelay* : *EnteredValue*(*CHECKED_DOM* = *TIME*)

Intention	: This object represents the time an area has to be unoccupied continuously before the lights in this area are switched off.
Postreferences	: msref AreaLightCtrlPanelFM : I_LCPFM1 msref RoomLightCtrlPanelFM : M_LCPR1 msref AreaLight : M_AL5

Machine Specifications

Property *I_LCPFM1*

Formal	: *unoccDelay.checked* = 600000
NL	: Initially an area has to be continuously empty for at least 10 minutes before the lights in an area are switched off.

Property *I_LCPFM2*

Formal	: *lightOn.checked* = 0
NL	: Initially the lights in an area are not switched off by the facility manager.

Description Class RoomLightCtrlPanelFM

Intention

This class provides all the definitions of entities concerning the control of the light in a room for which the facility manager can enter a value using the control panel. In addition to the definitions of description class *AreaLightCtrlPanelFM* this is

- a possibility to determine the maximal delay within the reoccupation of a room leads to a reestablishing of the previous light scene.

Base Classes

Class *AreaLightCtrlPanelFM*

Intention : All definitions of description class *AreaLightCtrlPanelRoom* are also provided here.

Signature

Object *reoccDelay* : *EnteredValue*(*CHECKED_DOM* = *TIME*)

Intention : This object represents the value a user can specify for the maximal delay within the reoccupation of a room leads to a reestablishing of the previous light scene.

Postreferences : msref RoomLightCtrlPanelFM : I_LCPR1
msref RoomLightCtrlPanelFM : M_LCPR1
msref RoomLight : M_RL5
msref RoomLight : M_RL6
msref RoomLight : M_RL12
msref RoomLight : M_RL13

Object *defaultLightScene* : *EnteredValue*(*CHECKED_DOM* = *LightScene*)

Intention : This object represents the default light scene the facility manager can determine for a room.

Postreferences : msref RoomLight : M_RL2

Object *defaultAmbientLightLevel* : *EnteredValue*(
CHECKED_DOM = *LIGHT_VALUES*)

Intention : This object represents the default ambient light level the facility manager can determine for a room.

Postreferences : msref RoomLight : M_RL3

Machine Specifications

Property *I_LCPR1*

Formal : *reoccDelay.checked* = 1200000

NL : Initially the maximal delay within the reoccupation of a room leads to a reestablishing of the previous light scene is 20 minutes.

Property *M_LCPR1*

Formal	: $\Box(reoccDelay.checked > unoccDelay.checked)$
NL	: The maximal delay within the reoccupation of a room leads to a reestablishing of the previous light scene is always greater than the time a room has to be continuously empty before the lights in a room are switched off.

Description Class LightCtrlPanelRoom

Intention

This class provides all the definitions of entities concerning the control of the light in a room for which a user can enter a value using the control panel of a room. These are

- a possibility to choose the current light scene
- a possibility to choose the current ambient light level
- a possibility to switch on and off the window ceiling light
- a possibility to switch on and off the wall ceiling light
- a possibility to determine whether
  - the currently set ambient light level should be applied or
  - the currently settings of the window and wall ceiling light switches of the control panel should be applied
- a possibility to determine whether
  - the light scenes are enabled by the control panel, i.e. the currently chosen light scene should be applied or
  - the currently settings of the control panel concerning the ambient switch, the ceiling light switches, and the ambient light level should be applied.

If the light is controlled using the control panel the following cases are possible:

- A chosen light scene has to be established. This is the case if the value of *lightSceneSwitch* is 1.
- A set ambient light level has to be established. This is the case if the value of *lightSceneSwitch* is 0 and the value of *ambientSwitch* is 1.
- The two lights are set directly using the control panel. This is the case if the value of *lightSceneSwitch* is 0 and the value of *ambientSwitch* is 0.

These different cases are specified in the description class *RoomLight*.

Note that such a control panel does not exist up to now and has to be developed, too.

Signature

Object *chosenLightScene* : *EnteredValue*(*CHECKED_DOM* = *STRING*)

Intention	: This object represents the light scene currently chosen by using the control panel. A light scene is identified by its name. Therefore, the entered value has to have the domain *STRING*.
Postreferences	: msref RoomLight : M_RL5 msref RoomLight : M_RL7

Object *ambLightLev* : *EnteredValue*(*CHECKED_DOM* = *LIGHT_VALUES*)

Intention : This object represents the value defined for the ambient light level by using the control panel. The entered value has to be an element of the domain LIGHT_VALUES.
Note that this setting is only relevant if the light is not controlled manually using the push buttons.

Postreferences : msref RoomLight : M_RL12
msref RoomLight : M_RL14

Object *winCeilLightSwitch* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention : This object represents the value defined for the window ceiling light by using the control panel.
winCeilLightSwitch.checked = 1 means that the window ceiling light should be on.
winCeilLightSwitch.checked = 0 means that the window ceiling light should be off.
Note that this setting is only relevant if the light is not controlled manually using the push buttons.

Postreferences : msref RoomLight : M_RL10

Object *wallCeilLightSwitch* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention : This object represents the value defined for the wall ceiling light by using the control panel.
wallCeilLightSwitch.checked = 1 means that the wall ceiling light should be on.
wallCeilLightSwitch.checked = 0 means that the wall ceiling light should be off.
Note that this setting is only relevant if the light is not controlled manually using the push buttons.

Postreferences : msref RoomLight : M_RL10

Object *ambientSwitch* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention : This object represents the value that indicates whether

- the currently set ambient light level (*ambLightLev*) should be applied (this is the case for *ambientSwitch.checked* = 1) or
- the currently settings of the window and wall ceiling light switches of the control panel should be applied (this is the case for *ambientSwitch.checked* = 0).

Note that this setting is only relevant if the light is not controlled manually using the push buttons.

Postreferences : msref RoomLight : M_RL4
msref RoomLight : M_RL9
msref RoomLight : M_RL10

Object *lightSceneSwitch* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention : This object represents the value that indicates whether

- the light scenes are enabled by the control panel, i.e. the currently chosen light scene should be applied (this is the case for *lightSceneSwitch* = 1) or
- the currently settings of the control panel concerning the ambient switch (*ambientSwitch*), the ceiling light switches (*winCeilLightSwitch*, *winCeilLightSwitch*), and the ambient light level (*ambLightLev*) should be applied (this is the case for *lightSceneSwitch* = 0).

Note that this setting is only relevant if the light is not controlled manually using the push buttons.

Postreferences : msref RoomLight : M_RL4
msref RoomLight : M_RL8
msref RoomLight : M_RL9
msref RoomLight : M_RL10

Requirement Specifications

Property *R_LCP1*

NonFormal : The control panel should be easy and intuitive to use.

Property *R_LCP2*

NonFormal : Light scenes can be entered using the control panel which are stored and can be chosen.
Note that this is not the same as the currently chosen light scene.

Description Class LightCtrlPanelBlindedRoom

Intention

This class provides all the definitions of entities concerning the control of the light in a blinded room for which a user can enter a value using the control panel.

In addition to the definitions of description class *LightCtrlPanelRoom* this class provides the definitions concerning the settings for sun blinds, i.e. for each sun blind in the room the control panel contains

- a switch to signal that a sun blind is controlled manually and
- a switch to determine the setting of a sun blind if it is controlled manually.

Note that such a control panel does not exist up to now and has to be developed, too.

Formal Parameters

Function *numSunBlinds* $\rightarrow$ *NAT*

Intention : This parameter represents the number of sun blinds that belong to a room and which can be controlled by the machine.

Base Classes

Class *LightCtrlPanelRoom*

Intention : All definitions of description class *LightCtrlPanelRoom* are also provided here.

Signature

Object *sunBlindSwitch* : *EnteredValue*(
CHECKED_DOM = *THREE_PT_ACT_RANGE*)[*numSunBlinds*]

Intention : Each object represents the value set by the user for the control of a sun blind by using the control panel.
Postreferences : msref BlindedRoom : M_BR4

Object *sunBlindManual* : *EnteredValue*(
CHECKED_DOM = *BINARY*)[*numSunBlinds*]

Intention : Each object represents whether the corresponding sun blind is controlled manually or not. A 1 means that it is controlled manually; a 0 means that it is not controlled manually.
Postreferences : msref BlindedRoom : M_BR1

Description Class TemperatureCtrlPanelFM

Intention

This class provides all the definitions of entities concerning the control of the temperature in a room for which the facility manager can enter a value using his control panel and which can not be changed by a user entering a value at the control panel in a room. A value entered by the facility manager can

- be valid for all rooms,
- be specific for each room type (office, meeting room, computer lab),
- be specific for each room.

This is mentioned in the intentions of the objects, too.

These objects are

- a possibility to determine the heating period. This object is the same for all rooms.
- a possibility to determine the non heating period. This object is the same for all rooms.
- a possibility to determine the non freeze period for the whole building. This object is the same for all rooms.
- a possibility to determine the non freeze period for a specific room. This object is specific for each room.
- a possibility to determine the temperature that should be applied if the operation mode *NonFreezeMode* is active. This object is the same for all rooms.
- a possibility to determine the temperature that should be applied if the operation mode *standbyMode* is active. This object is specific for each room type.
- a possibility to determine the temperature that should be applied if the operation mode *OffMode* is active. This object is specific for each room type.
- a possibility to determine whether the delay times used for determining occupancy are adaptive (i.e. the systems *learns* a typical user behavior) or not. This object is the same for all rooms.

Values for which the facility manager specifies a default value that can be changed by a user are considered in the description class *TemperatureCtrlPanel*.

Note that such a control panel does not exist up to now and has to be developed, too.

Signature

Object *heatingPeriod* : *CalendarInterval*

Intention : This object represents the heating period, i.e. the start and end point of the period and whether the actual date is within the non heating period or not.
This object is the same for all rooms.
Postreferences : msref Floor : M_F3

Object *buildingNonFreezePeriod* : *CalendarInterval*

Intention : This object represents the period within which the non freeze mode is set for the whole building.
This object is the same for all rooms.
Postreferences : msref RoomTemperature : M_RT1
msref Floor : M_F4

Object *nonFreezePeriod* : *CalendarInterval*

Intention : This object represents the period within which the non freeze mode is set for a certain room.
This object is specific for each room.
Postreferences : msref RoomTemperature : M_RT1

Object *nonFreezeTemperature* : *EnteredValue*(*CHECKED_DOM = TEMP*)

Intention : This object represents the temperature that should be applied if the operation mode *NonFreezeMode* is active.
This object is the same for all rooms.
Postreferences : msref RoomTemperature : M_RT5
msref Floor : M_F5

Object *standbyTemperature* : *EnteredValue*(*CHECKED_DOM = TEMP*)

Intention : This object represents the temperature that should be applied if the operation mode *standbyMode* is active. This temperature is set by the facility manager.
This object is specific for each room type.
Postreferences : msref RoomTemperature : M_RT3
msref Floor : M_F7
msref Floor : M_F9
msref Floor : M_F11

Object *offTemperature* : *EnteredValue*(*CHECKED_DOM = TEMP*)

Intention : This object represents the temperature that should be applied if the operation mode *OffMode* is active. This temperature is set by the facility manager.
This object is specific for each room type.
Postreferences : msref RoomTemperature : M_RT4
msref Floor : M_F8
msref Floor : M_F10
msref Floor : M_F12

Object *adaptive* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention	: This object represents the modus set by the facility manager indicating whether the delay times used for determining occupancy are adaptive (i.e. the systems *learns* a typical user behavior) or not. This object is the same for all rooms.
Postreferences	: msref TemperatureCtrlPanelFM : M_TCPF1 msref RoomTemperature : M_RT6 msref RoomTemperature : M_RT7 msref RoomTemperature : M_RT8 msref Floor : M_F6

Timed Predicate *roomNonFreezeDefined*

Intention	: This predicate is true iff the facility manager has defined a non-freeze period for a certain room. This object is specific for each room.
Scope	: mh
Postreferences	: msref RoomTemperature : M_RT1

Machine Specifications

Property *M_TCPF1*

Formal	: $adaptive.checked = 0$
NL	: Initially it is not set that the delay times used for determining occupancy are adaptive.

Description Class TemperatureCtrlPanelRoom

Intention

This class provides all the definitions of entities concerning the control of the temperature in a room for which a user can enter a value using the control panel of a room and for which the facility manager can not enter a default value. This is

- a possibility to override the non-freeze operation mode set by the facility manager for a specific room.

Values for which the facility manager specifies a default value that can be changed by a user are considered in the description class *TemperatureCtrlPanel*.

Note that such a control panel does not exist up to now and has to be developed, too.

Signature

Object *nonFreezeOverride* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention	: This object represents whether the user has entered an override over the non-freeze operation mode set by the facility manager for a specific room. A 1 is indicating that the user has entered an override.
Postreferences	: msref TemperatureCtrlPanelRoom : M_TCPR1 msref RoomTemperature : M_RT1

Machine Specifications

Property *M_TCPR1*

Formal	: *nonFreezeOverride.checked* = 0
NL	: Initially no override for the non–freeze operation mode is set by the user.

Description Class TemperatureCtrlPanel

Intention

This class combines the two classes *TemperatureCtrlPanelFM* and *TemperatureCtrlPanelRoom* resulting in a class that contains all entities concerning the control of the temperature in a room for which a value can be entered using a control panel. Besides the collection of the entities of the two mentioned classes in this class also those entities are introduced for which the facility manager can specify a default value that can be changed by a user.

This class does not contain the values that can be entered concerning the malfunction of a sensor or actuator. They are introduced in the description classes dealing with the malfunctions of sensors and actuators.

Signature

Object *tCPFM* : *TemperatureCtrlPanelFM*

Intention : This object represents the part of the temperature control panel containing the entities for which the facility manager can enter a value.

Postreferences : msref RoomTemperature : M_RT1
msref RoomTemperature : M_RT3
msref RoomTemperature : M_RT4
msref RoomTemperature : M_RT5
msref RoomTemperature : M_RT6
msref RoomTemperature : M_RT7
msref RoomTemperature : M_RT8
msref Floor : M_F3
msref Floor : M_F4
msref Floor : M_F5
msref Floor : M_F6
msref Floor : M_F7
msref Floor : M_F8
msref Floor : M_F9
msref Floor : M_F10
msref Floor : M_F11
msref Floor : M_F12

Object *tCPRoom* : *TemperatureCtrlPanelRoom*

Intention : This object represents the part of the temperature control panel containing the entities for which a user can enter a value.

Postreferences : msref RoomTemperature : M_RT1

Object *expectedUsageTimes* : *OccDelayTimes*

Intention : This object represents the set of occupancy delay times dt_1, dt_2, dt_3 in the case that a usage of the room is expected.
This object is specific for each room.
Postreferences : msref RoomTemperature : M_RT6
msref RoomTemperature : M_RT7
msref RoomTemperature : M_RT8
msref RoomTemperature : M_RT11

Object *notExpectedUsageTimes* : *OccDelayTimes*

Intention : This object represents the set of occupancy delay times dt_1, dt_2, dt_3 in the case that a usage of the room is not expected.
This object is specific for each room.
Postreferences : msref RoomTemperature : M_RT6
msref RoomTemperature : M_RT7
msref RoomTemperature : M_RT8
msref RoomTemperature : M_RT12

Object *comfortTemperature* : *ControlledInput*(
INPUT_DOM = *COMFORT_TEMP*)

Intention : This object represents the current comfort temperature of a room (depending on the values entered by the facility manager and the room user).
This object is specific for each room.
Postreferences : msref RoomTemperature : M_RT2

Requirement Specifications

Property *R_TCP1*

NonFormal : The control panels should be easy and intuitive to use.

Description Class ExpectedUsage

Intention

This class provides all the definitions to describe whether a usage of a room is expected or not. This depends on several settings of the facility manager and the user.

The information whether a usage of a room is expected or not is needed to define different modes for the temperature control of a room.

Signature

Object *workingTime* : *ControlledDynamicInterval*(*INT_DOM* = *HOURS*)

Intention	: This object represents the current working time, an interval, (depending on the values entered by the facility manager and the room user). This object is specific for each room type.
Postreferences	: rsref ExpectedUsage : D_EU1

Object *holidayTime* : *ControlledDynamicInterval*(
INT_DOM = *CALENDAR_TIME*)

Intention	: This object represents the current holiday time, an interval, (depending on the values entered by the facility manager and the room user). This object is specific for each room type.
Postreferences	: rsref ExpectedUsage : D_EU1

Timed Predicate *expectedUsage*

Intention	: This predicate is true iff a usage of a room is expected by the machine.
Scope	: mh
Postreferences	: rsref ExpectedUsage : D_EU1 msref TempOperationModeSet : M_TOMS1 msref TempOperationModeSet : M_TOMS2 msref RoomTemperature : M_RT6 msref RoomTemperature : M_RT7 msref RoomTemperature : M_RT8 msref RoomTemperature : M_RT11 msref RoomTemperature : M_RT12

Requirement Specifications

Property *D_EU1*

Formal : $\Box(expectedUsage \leftrightarrow$
$workingTime.leq(workingTime.start.value, currentHour) \wedge$
$workingTime.leq(currentHour, workingTime.end.value) \wedge$
$\neg(holidayTime.leq(holidayTime.start.value, currentDate) \wedge$
$holidayTime.leq(currentDate, holidayTime.end.value)))$

NL : A usage of a room is expected if and only if the following two conditions are fulfilled together:

- the current time is within the working time and
- the current time is not within the holiday time.

Description Class GroupOccupancy

Intention

This group contains all description classes that contain definitions needed to define under which conditions an area is occupied.

Description Class AdaptiveOccupancyTimes

Intention

This class provides all the definitions to describe how the three time delays used to determine occupancy in an area are modified if the facility manager has chosen the delay times to be *adaptive*. In this case the machine *adapts* the delay times depending on the usage profile of a room.

Signature

Timed Function *dt1Adaptive* $\rightarrow$ *TIME*

Intention	: This function represents the value for dt_1 as it is determined using an adaptive learning function.
Scope	: mh
Postreferences	: msref RoomTemperature : M_RT6

Timed Function *dt2Adaptive* $\rightarrow$ *TIME*

Intention	: This function represents the value for dt_2 as it is determined using an adaptive learning function.
Scope	: mh
Postreferences	: msref RoomTemperature : M_RT7

Timed Function *dt3Adaptive* $\rightarrow$ *TIME*

Intention	: This function represents the value for dt_3 as it is determined using an adaptive learning function.
Scope	: mh
Postreferences	: msref RoomTemperature : M_RT8

Description Class NoMalfunctionOccupancy

Intention

This class provides all definitions to describe the term *occupancy* for an area in the case that there is no malfunction of the sensors used to determine occupancy in this area.

Whether the machine assumes occupancy in an area or not depends on

- three time values, dt_1, dt_2, dt_3, and
- on a predicate indicating whether a person is currently in the area or not.

Note that the status of *occupancy* in general does not correspond with the fact whether there is a person in an area or not. *Occupancy* is a derived term depending on this fact.

Signature

Timed Function *dt1* $\rightarrow$ *TIME*

Intention	: This function represents the current value used for the first time delay used to determine occupancy. The precise correlation between this time delay and *occupancy* is given in the property P_NMO_1.
Scope	: mh
Postreferences	: msref NoMalfunctionOccupancy : M_NMO1 msref AreaLight : M_AL3 msref RoomTemperature : M_RT6 msref RoomTemperature : M_RT11 msref RoomTemperature : M_RT12

Timed Function *dt2* $\rightarrow$ *TIME*

Intention	: This function represents the current value used for the second time delay used to determine occupancy. The precise correlation between this time delay and *occupancy* is given in property P_NMO_1.
Scope	: mh
Postreferences	: msref NoMalfunctionOccupancy : M_NMO1 msref AreaLight : M_AL4 msref RoomTemperature : M_RT7 msref RoomTemperature : M_RT11 msref RoomTemperature : M_RT12

Timed Function *dt3* $\rightarrow$ *TIME*

Intention : This function represents the current value used for the third time delay used to determine occupancy.
The precise correlation between this time delay and *occupancy* is given in property P_NMO_1.
Scope : mh
Postreferences : msref NoMalfunctionOccupancy : M_NMO1
msref AreaLight : M_AL5
msref RoomTemperature : M_RT8
msref RoomTemperature : M_RT11
msref RoomTemperature : M_RT12

Timed Predicate *emptyMach*

Intention : This predicate is true iff the machine assumes that there is currently no person in the area for which occupancy is considered.
Scope : mh
Postreferences : msref NoMalfunctionOccupancy : M_NMO1
msref HallwaySection : M_HS3
msref BasicRoom : M_BaR5
msref BasicRoom : M_BaR6

Timed Predicate *occupied*

Intention : This predicate is true iff the machine assumes that the area for which occupancy is considered is occupied.
This predicate represents the situation as it is seen by the machine and not how it is in the real world.
Note that this predicate does not describe whether a person is in a area or not. See timed predicate *emptyMach*.
Scope : mh
Postreferences : msref NoMalfunctionOccupancy : M_NMO1
msref Occupancy : M_OCC2

Machine Specifications

Property *M_NMO1*

Formal : $\Box(occupied \leftrightarrow$
$\blacksquare_{\leq dt1} \neg emptyMach \vee$
$\langle *[\blacksquare_{\leq dt1+dt2} \neg emptyMach] \Rightarrow \rangle > \blacksquare \blacklozenge_{\leq dt3} \neg emptyMach)$

NL : An area is defined to be occupied if and only if one of the following conditions holds:

- during the past dt_1 time units a person has continuously been in the area or
- there has already been a time span of length $dt_1 + dt_2$ such that a person has continuously been in the area, and since this has been the case for the last time, periods where the area has continuously been empty were not longer than or equal to dt_3.

Description Class MalfunctionOccupancy

Intention

This class provides all definitions to describe the behavior of the system concerning the detection of occupancy in an area in the case that some sensors used to determine occupancy have a malfunction.

In the case of a malfunction of such a sensor, a value is determined depending on a possibly manually entered value by a user and an automatically generated value. If a value has been entered manually this value has priority over the automatically determined value.

Most of the entities introduced in this description class have to be specified in the description classes where they are used.

Signature

Object *manualEnteredOcc* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention	: This object represents the value concerning occupancy entered manually by a user in the case of a malfunction of some sensors used to determine occupancy. 1 means occupancy, 0 means no occupancy. Note that the value of this object is only meaningful if the predicate *stateManualEnteredOcc* is true.
Postreferences	: msref MalfunctionOccupancy : M_MO1

Timed Predicate *malOccDetEnv*

Intention	: This predicate is true iff some sensors existing in the real world used to determine occupancy have a malfunction. Note that this predicate represents the value as it exists in the real world. This value is not visible for the machine.
Scope	: eh

Timed Predicate *malOccDetMach*

Intention	: This predicate is true iff the machine assumes that some sensors used to determine occupancy have a malfunction. Note that this predicate represents the value as it seen by the machine.
Scope	: mh
Postreferences	: msref Occupancy : M_OCC1 msref Occupancy : M_OCC2

Timed Predicate *stateManualEnteredOcc*

Intention	: This predicate is true iff it has been entered manually whether there is occupancy or not.
Scope	: mh
Postreferences	: msref MalfunctionOccupancy : I_MO1 msref MalfunctionOccupancy : M_MO1

Timed Predicate *automaticDetermValue*

Intention	: This predicate represents the occupancy determined automatically by the machine in the case of a malfunction of the sensors used to determine occupancy. If this predicate is true this means occupancy. If it is false this means no occupancy.
Scope	: mh
Postreferences	: msref MalfunctionOccupancy : M_MO1 msref AreaLight : M_AL7 rsref RoomTemperature : R_RT1

Timed Predicate *occMal*

Intention	: This predicate represents the occupancy used by the machine in the case of a malfunction of the sensors used to determine occupancy. If this predicate is true this means occupancy. If it is false this means no occupancy.
Scope	: mh
Postreferences	: msref MalfunctionOccupancy : M_MO1 msref Occupancy : M_OCC2

Machine Specifications

Property *I_MO1*

Formal	: $\neg stateManualEnteredOcc$
NL	: Initially no value for occupancy in the case of a malfunction is entered manually.

Property *M_MO1*

Formal	: $\Box(stateManualEnteredOcc \rightarrow (occMal \leftrightarrow manualEnteredOcc.checked = 1) \wedge (\neg stateManualEnteredOcc \rightarrow (occMal \leftrightarrow automaticDetermValue)))$
NL	: If a value for the status of occupancy has been entered manually this is the value used for occupancy in the case of a malfunction of some sensors used to determine occupancy. Otherwise, the value used for occupancy in the case of a malfunction of the sensors used to determine occupancy corresponds with the value determined automatically.

Description Class Occupancy

Intention

This class provides all definitions to describe the occupancy in a specific area. It is a composition of the following two cases:

- occupancy is computed under the condition that all sensors used to determine occupance have no malfunction
- occupancy is computed under the condition that some sensors used to determine occupance have a malfunction

To compose these two cases some entities are introduced that allow to determine whether some sensors used to determine occupancy have a malfunction.

Signature

Object *noMalOcc* : *NoMalfunctionOccupancy*

Intention	: This object represents the part of the occupancy determination describing the behavior if the sensors used to determine occupancy have no malfunction, i.e. if the machine assumes that they have no malfunction.
Postreferences	: msref Occupancy : M_OCC2 msref AreaLight : M_AL3 msref AreaLight : M_AL4 msref AreaLight : M_AL5 msref HallwaySection : M_HS3 msref RoomTemperature : M_RT6 msref RoomTemperature : M_RT7 msref RoomTemperature : M_RT8 msref RoomTemperature : M_RT11 msref RoomTemperature : M_RT12 msref BasicRoom : M_BaR5 msref BasicRoom : M_BaR6

Object *malOcc* : *MalfunctionOccupancy*

Intention	: This object represents the part of the occupancy determination describing the behavior if the sensors used to determine occupancy have a malfunction, i.e. if the machine assumes that they have a malfunction.
Postreferences	: msref Occupancy : M_OCC1 msref Occupancy : M_OCC2 msref AreaLight : M_AL7 rsref RoomTemperature : R_RT1

Object *manualEnteredMal* : *EnteredValue*(*CHECKED_DOM* = *BINARY*)

Intention : Manually entered value concerning a malfunction of the sensors determining occupancy entered by a user. 1 stands for a malfunction, 0 stands for no malfunction. Note that the value of this object is only meaningful if the timed predicate *stateManualEnteredMal* is true.
Postreferences : msref Occupancy : M_OCC1

Timed Predicate *stateManualEnteredMal*

Intention : This predicate is true iff it has been manually entered whether there is a malfunction of the sensors determining occupancy or whether there is no malfunction of them.
Scope : mh
Postreferences : msref Occupancy : I_OCC1
msref Occupancy : M_OCC1

Timed Predicate *automaticDetMal*

Intention : This predicate is true iff a malfunction of the sensors used to determine occupancy has been detected automatically, i.e. not by entering it manually.
Scope : mh
Postreferences : msref Occupancy : M_OCC1

Timed Predicate *occUsed*

Intention : This predicate is true iff the machine assumes occupancy.
Scope : mh
Postreferences : msref Occupancy : M_OCC2
msref TempOperationModeSet : M_TOMS1
msref TempOperationModeSet : M_TOMS2
msref TempOperationModeSet : M_TOMS3
msref AreaLight : M_AL6
msref AreaLight : M_AL8
msref AreaLight : M_AL9
msref RoomLight : M_RL5
msref RoomLight : M_RL6
msref RoomLight : M_RL7
msref RoomLight : M_RL8
msref RoomLight : M_RL9
msref RoomLight : M_RL10
msref RoomLight : M_RL12
msref RoomLight : M_RL13
msref RoomLight : M_RL14
msref BlindedRoom : M_BR1
rsref AreaLight : R_AL1
rsref AreaLight : R_AL2
rsref Floor : R_F1

Machine Specifications

Property *I_OCC1*

Formal : $\neg stateManualEnteredMal$

NL : Initially it has not been entered manually whether there is a malfunction of the sensors determining occupancy or whether there is no malfunction of them.

Property *M_OCC1*

Formal : $\Box(malOcc.malOccDetMach \leftrightarrow ((stateManualEnteredMal \wedge manualEnteredMal.checked = 1) \vee (\neg stateManualEnteredMal \wedge automaticDetMal)))$

NL : The machine assumes a malfunction of some sensors used to determine occupancy iff one of the following conditions is fulfilled:

- a malfunction has been entered manually or
- it has not been entered manually that there is no malfunction and a malfunction has been detected automatically.

Property *M_OCC2*

Formal : $\Box(occUsed \leftrightarrow ((malOcc.malOccDetMach \wedge malOcc.occMal) \vee (\neg malOcc.malOccDetMach \wedge noMalOcc.occupied)))$

NL : The machine assumes occupancy if and only if one of the following condition is fulfilled:

- the machine assumes a malfunction of some sensors used to detect occupancy and the predicate representing the value concerning occupancy assumed in this case is true,
- the machine assumes no malfunction and the predicate representing the value concerning occupancy assumed in this case is true.

Group TemperatureModes

Intention

This group contains all description classes that contain definitions needed to define temperature modes of a room.

Description Class TempOperationMode

Intention

This class provides all the definitions to describe a single *operation mode* of a room concerning temperature. A timed predicate is provided indicating whether the operation mode is active and a timed function repreeenting the temperature that should be established if the operation mode is active.

Signature

Timed Function *desiredTemperature* $\rightarrow$ *TEMP*

Intention	: This function represents the temperature that should be established in a room in this operation mode.
Scope	: mh
Postreferences	: msref TempOperationModeSet : M_TOMS5
	msref TempOperationModeSet : M_TOMS6
	msref TempOperationModeSet : M_TOMS7
	msref TempOperationModeSet : M_TOMS8
	msref RoomTemperature : M_RT2
	msref RoomTemperature : M_RT3
	msref RoomTemperature : M_RT4
	msref RoomTemperature : M_RT5

Timed Predicate *active*

Intention	: This predicate is true iff this operation mode is active.
Scope	: mh
Postreferences	: msref TempOperationModeSet : M_TOMS1
	msref TempOperationModeSet : M_TOMS2
	msref TempOperationModeSet : M_TOMS3
	msref TempOperationModeSet : M_TOMS4
	msref TempOperationModeSet : M_TOMS5
	msref TempOperationModeSet : M_TOMS6
	msref TempOperationModeSet : M_TOMS7
	msref TempOperationModeSet : M_TOMS8

Description Class TempOperationModeSet

Intention

This class provides all the definitions to describe all *operation modes* concerning the temperature control of a room. Note that for each room there is a specific set of operation modes.

Four operation modes are distinguished: *NonFreezeMode*, *ComfortMode*, *StandbyMode*, and *OffMode*. The definitions of these modes are given in this class. Which mode is currently active depends on three predicates: *expectedUsage*, *occUsed*, and *nonFreezeTime*. Based on the currently active mode the desired room temperature is determined.

Signature

Object *OffMode* : *TempOperationMode*

Intention	: This object represents the operation mode *offMode*.
Postreferences	: msref TempOperationModeSet : M_TOMS1
	msref TempOperationModeSet : M_TOMS5
	msref RoomTemperature : M_RT4

Object *StandbyMode* : *TempOperationMode*

Intention	: This object represents the operation mode *StandbyMode*.
Postreferences	: msref TempOperationModeSet : M_TOMS2
	msref TempOperationModeSet : M_TOMS6
	msref RoomTemperature : M_RT3

Object *ComfortMode* : *TempOperationMode*

Intention	: This object represents the operation mode *ComfortMode*.
Postreferences	: msref TempOperationModeSet : M_TOMS3
	msref TempOperationModeSet : M_TOMS7
	msref RoomTemperature : M_RT2

Object *NonFreezeMode* : *TempOperationMode*

Intention	: This object represents the operation mode *NonFreezeMode*.
Postreferences	: msref TempOperationModeSet : M_TOMS4
	msref TempOperationModeSet : M_TOMS8
	msref RoomTemperature : M_RT5

Object *expUse* : *ExpectedUsage*

Intention	: This object represents the part of the definition of the operation modes defining whether a usage of a room is expected or not.
Postreferences	: msref TempOperationModeSet : M_TOMS1
	msref TempOperationModeSet : M_TOMS2
	msref RoomTemperature : M_RT6
	msref RoomTemperature : M_RT7
	msref RoomTemperature : M_RT8
	msref RoomTemperature : M_RT11
	msref RoomTemperature : M_RT12

Object *occ* : *Occupancy*

Intention : This object represents the part of the definition of the operation modes defining whether a room is occupied or not.
Postreferences : msref TempOperationModeSet : M_TOMS1
msref TempOperationModeSet : M_TOMS2
msref TempOperationModeSet : M_TOMS3
msref RoomTemperature : M_RT6
msref RoomTemperature : M_RT7
msref RoomTemperature : M_RT8
msref RoomTemperature : M_RT11
msref RoomTemperature : M_RT12
msref BasicRoom : M_BaR6
msref BlindedRoom : M_BR1
rsref RoomTemperature : R_RT1

Timed Function *roomDesiredTemperature* $\rightarrow$ *TEMP*

Intention : This function represents the currently desired temperature that should be established in a room.
Scope : mh
Postreferences : msref TempOperationModeSet : M_TOMS5
msref TempOperationModeSet : M_TOMS6
msref TempOperationModeSet : M_TOMS7
msref TempOperationModeSet : M_TOMS8
msref Room : M_R2
msref BlindedRoom : M_BR2

Timed Predicate *nonFreezeTime*

Intention : This predicate is true iff it is currently non-freeze operation time.
Scope : mh
Postreferences : msref TempOperationModeSet : M_TOMS1
msref TempOperationModeSet : M_TOMS2
msref TempOperationModeSet : M_TOMS3
msref TempOperationModeSet : M_TOMS4
msref RoomTemperature : M_RT1

Machine Specifications

Property *M_TOMS1*

Formal : $\Box(OffMode.active \leftrightarrow \neg nonFreezeTime \wedge \neg expUse.expectedUsage \wedge \neg occ.occUsed)$

NL : The operation mode *OffMode* is active in a room iff the following conditions are fulfilled together:

- currently it is not non-freeze operation time,
- no usage of the room is expected, and
- the room is not occupied.

Property *M_TOMS2*

Formal : $\Box(StandbyMode.active \leftrightarrow \neg nonFreezeTime \wedge expUse.expectedUsage \wedge \neg occ.occUsed)$

NL : The operation mode *StandbyMode* is active in a room iff the following conditions are fulfilled together:

- currently it is not non-freeze operation time,
- usage of the room is expected, and
- the room is not occupied.

Property *M_TOMS3*

Formal : $\Box(ComfortMode.active \leftrightarrow \neg nonFreezeTime \wedge occ.occUsed)$

NL : The operation mode *ComfortMode* is active in a room iff the following conditions are fulfilled together:

- currently it is not non-freeze operation time and
- the room is occupied.

Property *M_TOMS4*

Formal : $\Box(NonFreezeMode.active \leftrightarrow nonFreezeTime)$

NL : The operation mode *NonFreezeMode* is active in a room iff currently it is non-freeze operation time.

Property *M_TOMS5*

Formal : $\Box(OffMode.active \rightarrow roomDesiredTemperature = OffMode.desiredTemperature)$

NL : Whenever the operation mode *OffMode* is active the desired temperature of the room is the temperature that should be established in the operation mode *OffMode*.

Property *M_TOMS6*

Formal : $\Box(StandbyMode.active \rightarrow roomDesiredTemperature = StandbyMode.desiredTemperature)$

NL : Whenever the operation mode *StandbyMode* is active the desired temperature of the room is the temperature that should be established in the operation mode *StandbyMode*.

Property *M_TOMS7*

Formal : $\Box(ComfortMode.active \rightarrow roomDesiredTemperature = ComfortMode.desiredTemperature)$

NL : Whenever the operation mode *ComfortMode* is active the desired temperature of the room is the temperature that should be established in the operation mode *ComfortMode*.

Property *M_TOMS8*

Formal : $\Box(NonFreezeMode.active \rightarrow roomDesiredTemperature = NonFreezeMode.desiredTemperature)$

NL : Whenever the operation mode *NonFreezeMode* is active the desired temperature of the room is the temperature that should be established in the operation mode *NonFreezeode*.

Group BuildingUnits

Intention

This group contains all description classes that describe units of the floor considered in this problem specification.

Description Class AreaLight

Intention

This class provides all definitions to describe the general aspects of lights in an area. An area is here the generalization of rooms and hallways. In more detail this class provides:

- A set of lights that can be conrolled by the machine. The concrete number of controlled lights can be specified by instantiating the parameter *numLights*.
- The specification of the behavior of the system concerning the aspect *safety*. Here safety means that there is sufficient light to move. This class provides the requirements concerning the case that safe light has to be established. Other cases can be considered in derived classes.
- The specification of the behavior of the system concerning the case that the facility manager controls the light using a control panel.
- The specification of the behavior of the system concerning the case that the light is controlled manually.

Formal Parameters

Function $numLights \rightarrow NAT$

Intention : This parameter represents the number of lights in an area that are controlled by the machine.

Signature

Object *occ* : *Occupancy*

Intention : This object represents the part of the definition of area light defining whether a room is occupied or not.

Postreferences :
msref AreaLight : M_AL3
msref AreaLight : M_AL4
msref AreaLight : M_AL5
msref AreaLight : M_AL6
msref AreaLight : M_AL7
msref AreaLight : M_AL8
msref AreaLight : M_AL9
msref HallwaySection : M_HS3
msref RoomLight : M_RL5
msref RoomLight : M_RL6
msref RoomLight : M_RL7
msref RoomLight : M_RL8
msref RoomLight : M_RL9
msref RoomLight : M_RL10
msref RoomLight : M_RL12
msref RoomLight : M_RL13
msref RoomLight : M_RL14
msref BasicRoom : M_BaR5
rsref AreaLight : R_AL1
rsref AreaLight : R_AL2
rsref Floor : R_F1

Object *light* : *ControlledLight*[*numLights*]

Intention : These objects represent the lights in area that are controlled by the machine.

Postreferences :
dkref AreaLight : D_AL2
lsref RoomLight : light
msref AreaLight : M_AL8
msref AreaLight : M_AL9
msref AreaLight : M_AL10
rsref AreaLight : R_AL1
rsref AreaLight : R_AL2
rsref HallwaySection : R_HS1
rsref HallwaySection : R_HS2
rsref Floor : R_F1

Object *lCPFM* : *AreaLightCtrlPanelFM*

Intention : This object represents the light control panel containing the entities for which the facility manager can enter a value.

Postreferences :
lsref RoomLight : lCPFM
msref AreaLight : M_AL5
rsref AreaLight : R_AL1

Time Constant *T_AL1*

Intention : This time constant represents the maximal time delay for the following two situations:

- between the time point an area becomes occupied and the time point at which safe light is established;
- between the time point an area becomes unoccupied and the time point the machine controlled lights are off.

Scope : ev
Postreferences : dkref AreaLight : D_AL1
msref AreaLight : M_AL1
msref AreaLight : M_AL6
rsref AreaLight : R_AL1
rsref AreaLight : R_AL2

Time Constant *T_AL2*

Intention : This time constant represents the maximal delay between the time point the machine assumes an occupation of an area and the time point the machine has set the corresponding actuators to establish safe light.
Scope : ev
Postreferences : dkref AreaLight : D_AL1
msref AreaLight : M_AL2
msref AreaLight : M_AL8

Timed Predicate *lightFMControlled*

Intention : This predicate is true iff the light is controlled by the facility manager using the control panel.
Scope : eh
Postreferences : msref AreaLight : I_AL2
rsref AreaLight : R_AL1
rsref AreaLight : R_AL2

Timed Predicate *allLightsOff*

Intention : This predicate is true iff all lights in an area that are controlled by the machine are off.
Scope : eh
Postreferences : dkref AreaLight : D_AL2

Timed Predicate *safeLight*

Intention : This predicate is true, iff there is sufficient light to move safely.
Scope : eh
Postreferences : msref AreaLight : M_AL6

Timed Predicate *noSaveLight*

Intention : This predicate is true iff the light intensity in an area that is occupied can be less than the light intensity needed to establish *safe light*. This is for example the case if the light is controlled manually.
Scope : mh
Postreferences : msref AreaLight : I_AL1
msref AreaLight : M_AL6
msref AreaLight : M_AL8
msref HallwaySection : M_HS2
msref RoomLight : M_RL4

Timed Predicate *manualControl*

Intention : This predicate is true iff the machine assumes that the light in this area is controlled manually.
Scope : mh
Postreferences : msref AreaLight : M_AL9
msref AreaLight : M_AL10
msref HallwaySection : M_HS2
msref RoomLight : M_RL4
msref RoomLight : M_RL8
msref RoomLight : M_RL9
msref RoomLight : M_RL10
msref RoomLight : M_RL11

Function $evalSaveLight : NAT(numLights) \rightarrow BINARY$

Intention : This function represents the computation that results in the decision which light has to be on in order to achieve safe light in an area. *evalSaveLight*[i]=i means that the i–th light has to be on.
Scope : mh
Postreferences : msref AreaLight : M_AL8

Domain Knowledge

Property *D_AL1*

Formal : $\Box(T_AL2 < T_AL1)$
NL : The time constant T_AL_2 is allways smaller than the time constant T_AL_1.

Property *D_AL2*

Formal : $\Box(allLightsOff \leftrightarrow \forall n \in NAT(numLights) : light[n].ls.noMalSens.envEntity = 0)$
NL : The predicate *allLightsOff* is true iff all lights that are controlled by the machine are off.

Requirement Specifications

Property *R_AL1*

Formal : $\Box(lightFMControlled \wedge \neg occ.occUsed \Rightarrow_{\leq T_AL1} \forall n \in NAT(numLights) : (light[n].ls.noMalSens.envEntity = lCPFM.lightOn.checked))$

NL : Whenever the light is controlled by the facility manager using the control panel and the area is unoccupied for at least T_AL_1 milliseconds, the status of the machine controlled lights equals within this time the status as it is determined by the facility manager.

Property *R_AL2*

Formal : $\Box(\neg lightFMControlled \wedge \neg occ.occUsed \Rightarrow_{\leq T_AL1} \forall n \in NAT(numLights) : (light[n].ls.noMalSens.envEntity = 0))$

NL : Whenever the light is not controlled by the facility manager using the control panel and the area is unoccupied for at least T_AL_1 milliseconds, all machine controlled lights are off within this time and remains off at least as long as the precondition is true.

Machine Specifications

Property *I_AL1*

Formal : $\neg noSaveLight$

NL : Initially safe light has to be established if an area is occupied.

Property *I_AL2*

Formal : $\neg lightFMControlled$

NL : Initially the light is not controlled by the facility manager using his control panel.

Property *M_AL1*

Formal : $\Box(T_AL1 = 50)$

NL : The value of T_AL_1 is always 50 miliiseconds.

Property *M_AL2*

Formal : $\Box(T_AL2 = 30)$

NL : The value of T_AL_2 is always 30 miliiseconds.

Property *M_AL3*

Formal : $\Box(occ.noMalOcc.dt1 = 1000)$

NL : The first time delay used to determine occupancy concerning the light control is always 1 second.

Property *M_AL4*

Formal : $\Box(occ.noMalOcc.dt2 = 0)$

NL : The second time delay used to determine occupancy concerning the light control is always 0 second.

Property *M_AL5*

Formal : $\Box(occ.noMalOcc.dt3 = lCPFM.unoccDelay.checked)$

NL : The third time delay used to determine occupancy concerning the light control corresponds always with the value entered by the facility manager for the time a certain area has to be unoccupied continuously before the lights are switched off.

Property *M_AL6*

Formal : $\Box(\neg noSaveLight \wedge occ.occUsed \Rightarrow_{\leq T_AL1} safeLight)$

NL : Whenever safe light has to be established and the area is occupied for at least T_AL_1 milliseconds, the light is sufficient to move safely within this time span and remains sufficient to move safely at least as long as the precondition is true.

Property *M_AL7*

Formal : $\Box(occ.malOcc.automaticDetermValue)$

NL : The value determined automatically by the machine for occupancy concerning the light control in the case of a malfunction of some sensors used to determine occupancy concerning the light control is *occupancy*.

Property *M_AL8*

Formal : $\Box(\neg noSaveLight \wedge occ.occUsed \rightarrow \forall n \in NAT(numLights) :$
$(evalSaveLight(n) = 1 \wedge light[n].ls.usedValue = 0$
$\rightarrow \Diamond_{\leq T_AL2} light[n].pulse.noMalAct.adjustedEntity = 1))$

NL : Whenever safe light has to be established and the area is occupied the necessary actuators, i.e. pulse lines, are set within T_AL_2 milliseconds to establish safe light.

Property *M_AL9*

Formal : $\Box(manualControl \leftrightarrow$
$\exists n \in NAT(numLights) : (light[n].manualControl) \wedge$
$occ.occUsed)$

NL : The machine assumes that the light is controlled manually in an area iff if at least one light in this area is controlled manually and the area is occupied.

Property *M_AL10*

Formal : $\Box(manualControl$
$\rightarrow \forall n \in NAT(numLights) :$
$(\neg[light[n].pulse.noMalAct.adjustedEntity = 1]))$

NL : If the light in an area is controlled manually, the settings of all lights remain unchanged.

Description Class HallwaySection

Intention

This class provides all definitions to describe the hallway specific light requirements. Each hallway section is equipped with

- one group of not dimmable ceiling lights that are considered here as one ceiling light,
- three motion detectors: one at each end of a hallway section and one in the middle of a hallway section.

Note that there are no requirements concerning the temperature in a hallway.

Base Classes

Class *AreaLight*(*numLights* = 1)

Intention : All definitions of description class *AreaLight* are also provided here. There is one light in a hallway section that is controlled by the machine.

Signature

Object *eastMd* : *MotionDetector*

Intention : This object represents the motion detector in a hallway which is installed at the more eastern door of the hallway.
Postreferences : msref HallwaySection : M_HS3

Object *westMd* : *MotionDetector*

Intention : This object represents the motion detector in a hallway which is installed at the more western door of the hallway.
Postreferences : msref HallwaySection : M_HS3

Object *middleMd* : *MotionDetector*

Intention : This object represents the motion detector in a hallway which is installed in the middle of the hallway.
Postreferences : msref HallwaySection : M_HS3

Time Constant *T_HS1*

Intention : This constant represents the time delay between a time point a person is near a hallway door indicating that he wants to enter this hallway section and switching on the lights.
Scope : ev
Postreferences : msref HallwaySection : M_HS1
rsref HallwaySection : R_HS1

Time Constant *T_HS2*

Intention	: This constant represents the time delay between the loss of light control in this hallway section and switching on the lights.
Scope	: ev
Postreferences	: rsref HallwaySection : R_HS2

Timed Predicate *nearHallwayDoor*

Intention	: This predicate is true iff a person is near a hallway door to this hallway section indicating that he wants to enter the hallway section.
Scope	: eh
Postreferences	: rsref HallwaySection : R_HS1

Timed Predicate *hwLightUncontroll*

Intention	: This predicate is true iff the light is neither automatically nor manually controllable.
Scope	: eh
Postreferences	: rsref HallwaySection : R_HS2

Requirement Specifications

Property *R_HS1*

Formal	: $\Box(nearHallwayDoor \Rightarrow_{\leq T_HS1} light[1].ls.noMalSens.envEntity = 1)$
NL	: Whenever a person is continuously in the near of a hallway door of the hallway section for at least T_HS_1 time units indicating that he wants to enter the hallway section, then the light in the hallway section will be on within this time span, and remains on at least as long as the person is near the hallway section.

Property *R_HS2*

Formal	: $\Box(hwLightUncontroll \Rightarrow_{\leq T_HS2} light[1].ls.noMalSens.envEntity = 1)$
NL	: Whenever the lights are neither controllable automatically nor manually for at least T_HS_2 time units, then eventually within this time span, the light in the hallway section is on and remains so at least as long as the lights are neither controllable automatically nor manually.

Machine Specifications

Property *M_HS1*

Formal	: $\Box(T_HS1 = 50)$
NL	: The value of T_HS_1 is always 50 milliseconds.

Property *M_HS2*

Formal	: $\Box(\neg manualControl \rightarrow \neg noSaveLight)$
NL	: Whenever the light is not controlled manually the light intensity in a hallway section that is occupied has to be greater than or equal to the light intensity needed to establish *save light*.

Property *M_HS3*

Formal : $\Box(occ.noMalOcc.emptyMach \leftrightarrow$
$((eastMd.usedValue = 0) \wedge$
$(westMd.usedValue = 0) \wedge$
$(middleMd.usedValue = 0)))$

NL : The machine always assumes that there is currently no person in a hallway section iff the values of all three motion detectors are 0.

Description Class RoomLight

Intention

This class provides all definitions to describe the environment and the requirements concerning the control of lights common to any kind of room. In more detail this means that the cases are considered

- that safe light has to be established. This behavior is inherited from the description class *AreaLight*;
- that the light is controlled manually.
- that the light is controlled using the control panel. See the intention of description class *LightCtrlPanelRoom* for a more detailed explanation of this case.

In each room there are two dimmable lights that are controlled by the machine: a ceiling light at the window and a ceiling light at the wall to the hallway. Furthermore, in each room there is a control panel for entering values relevant for the control of the light. This is done by the user of a room. This control panel is not the same as the one the facility manager has.

Base Classes

Class *AreaLight*(*numLights* = 2)

Intention	: All definitions of description class Area are also provided here. There are two lights in a room that are controlled by the machine: *light*[1] represents the light near the window and *light*[2] is the light near the wall to the hallway.

Signature

Object *AreaLight* :: *light* : *DimmableControlledLight*

Intention	: The lights in a room are *dimmable* lights.
Postreferences	: msref RoomLight : M_RL8 msref RoomLight : M_RL9 msref RoomLight : M_RL10 msref RoomLight : M_RL11

Object *AreaLight* :: *lCPFM* : *RoomLightCtrlPanelFM*

Intention	: The control panel of the facility manager in a room provides some further input possibilities.
Postreferences	: msref RoomLight : M_RL2 msref RoomLight : M_RL3 msref RoomLight : M_RL5 msref RoomLight : M_RL6 msref RoomLight : M_RL12 msref RoomLight : M_RL13

Object *lCPRoom* : *LightCtrlPanelRoom*

Intention : This object represents the light control panel containing the entities for which a user can enter a value.
Postreferences : lsref BlindedRoomLight : lCPRoom
msref RoomLight : M_RL4
msref RoomLight : M_RL5
msref RoomLight : M_RL7
msref RoomLight : M_RL8
msref RoomLight : M_RL9
msref RoomLight : M_RL10
msref RoomLight : M_RL12
msref RoomLight : M_RL14

Time Constant *T_RL1*

Intention : This time constant represents the maximal delay between occupation of a room, when a light scene is chosen, and the establishment of the chosen light scene.
Scope : ev
Postreferences : msref RoomLight : M_RL1
msref RoomLight : M_RL7
msref RoomLight : M_RL14

Timed Function *curLightScene* $\rightarrow$ *LightScene*

Intention : This function represents the current light scene that is established by the machine if light scenes are enabled.
Scope : mh
Postreferences : msref RoomLight : I_RL2
msref RoomLight : M_RL5
msref RoomLight : M_RL6
msref RoomLight : M_RL7
msref RoomLight : M_RL8

Timed Function *curStandLightScene* $\rightarrow$ *LightScene*

Intention : This function represents the current standard (=default) light scene that is established by the machine, if light scenes are enabled.
Scope : mh
Postreferences : msref RoomLight : I_RL1
msref RoomLight : I_RL2
msref RoomLight : M_RL2
msref RoomLight : M_RL6
msref Floor : M_F2

Timed Function *faultStandLightScene* $\rightarrow$ *LightScene*

Intention : This function represents the light scene that has to become standard light scene in the case of a malfunction of the outdoor light sensors.
Scope : mh
Postreferences : msref Floor : M_F2

Timed Function *noFaultStandLightScene* → *LightScene*

Intention : This function represents the light scene that has to become standard light scene in the case that the outdoor light sensors have no malfunctions.
Scope : mh
Postreferences : msref RoomLight : I_RL1

Timed Function *curAmbLightLev* → *LIGHT_VALUES*

Intention : This function represents the current ambient light level that is established by the machine if ambient light settings are enabled by the control panel.
Scope : mh
Postreferences : msref RoomLight : I_RL3
msref RoomLight : M_RL3
msref RoomLight : M_RL9
msref RoomLight : M_RL12
msref RoomLight : M_RL13
msref RoomLight : M_RL14

Timed Function *defaultAmbLightLev* → *LIGHT_VALUES*

Intention : This function represents the default ambient light level.
Scope : mh
Postreferences : msref RoomLight : I_RL3
msref RoomLight : M_RL13

Timed Function *determineLightScene* : *STRING* → *LightScene*

Intention : This function determines the light scene which is identified by the given name.
Scope : mh
Postreferences : msref RoomLight : M_RL5
msref RoomLight : M_RL7

Function *evalLightScene* : *LightScene*, *LightScene* :: *LAMPS* → *PERCENT*

Intention : This function represents the computation of the dimmer settings for a given lighting (represented by the second argument) that are necessary to achieve the ambient light intensity of the given light scene (represented by the first argument) under consideration of the light scene's lighting order. The value of this function represents the dimmer settings.
Scope : mh
Postreferences : msref RoomLight : M_RL8

Function *evalAmbient* : $LIGHT_VALUES, LightScene :: LAMPS \rightarrow PERCENT$

Intention	: This function represents the dimmer setting for a given lighting (represented by the second argument) that is necessary to achieve the given ambient light intensity (represented by the first argument) together with the other lightings.
Scope	: mh
Postreferences	: msref RoomLight : M_RL9

Requirement Specifications

Property *R_RL1*

NonFormal	: The standard light scene in the case that the outdoor light sensors have no malfunction is that the light intensity is sufficient to move safely.

Property *R_RL2*

NonFormal	: The standard light scene in the case of a malfunction of the outdoor light sensors is that all ceiling lights are on.

Machine Specifications

Property *I_RL1*

Formal	: $curStandLightScene = noFaultStandLightScene$
NL	: Initially the current standard light scene is the light scene that has to become standard light scene in the case that the outdoor light sensors have no malfunction.

Property *I_RL2*

Formal	: $curLightScene = curStandLightScene$
NL	: Initially the current light scene is the current standard light scene.

Property *I_RL3*

Formal	: $curAmbLightLev = defaultAmbLightLev$
NL	: Initially the current ambient light level is the current default ambient light level.

Property *M_RL1*

Formal	: $\Box(T_RL1 = 50)$
NL	: The value of T_RL_1 is always 50 miliiseconds.

Property *M_RL2*

Formal	: $\Box(curStandLightScene = lCPFM.defaultLightScene.checked)$
NL	: The current standard light scene always corresponds with the default light scene determined by the facility manager.

Property *M_RL3*

Formal	: $\Box(curAmbLightLev = lCPFM.defaultAmbientLightLevel.checked)$
NL	: The current ambient light level always corresponds with the default ambient light level determined by the facility manager.

Property *M_RL4*

Formal : $\Box(noSaveLight \leftrightarrow manualControl \vee lCPRoom.lightSceneSwitch.checked = 1 \vee lCPRoom.ambientSwitch.checked = 1)$

NL : The light intensity in a room that is occupied can be less than the light intensity needed to establish *save light* iff the light is controlled manually or light scenes are enabled or the ambient light level is enabled.

Property *M_RL5*

Formal : $\Box(([occ.occUsed] \wedge \blacklozenge_{\leq lCPFM.reoccDelay.checked} occ.occUsed) \rightarrow curLightScene = determineLightScene(lCPRoom.chosenLightScene.checked))$

NL : Whenever the room becomes occupied now and was occupied sometimes within the last T_RL_2 milliseconds and the light scene l was last chosen, the light scene l is the current light scene.

Property *M_RL6*

Formal : $\Box(([occ.occUsed] \wedge \blacksquare_{>lCPFM.reoccDelay.checked} \neg occ.occUsed) \rightarrow curLightScene = curStandLightScene)$

NL : Whenever the room becomes occupied now and has not been occupied during the last time T_RL_2, the current light scene is the current standard light scene.

Property *M_RL7*

Formal : $\Box(\forall l \in LightScene : ((occ.occUsed \wedge \neg[occ.occUsed] \wedge [determineLightScene(lCPRoom.chosenLightScene.checked) = l]) \rightarrow \Diamond_{\leq T_RL1} curLightScene = l))$

NL : Whenever the room is occupied, but has not just become occupied now, and the light scene l becomes chosen, this light scene l is the current light scene within time T_RL_1.

Property *M_RL8*

Formal : $\Box((occ.occUsed \wedge \neg manualControl \wedge$
$lCPRoom.lightSceneSwitch.checked = 1)$
$\rightarrow ((light[1].pulse.noMalAct.adjustedEntity = 1 \leftrightarrow$
$\neg evalLightScene(curLightScene, LightScene :: WINDOW)$
$= 0) \wedge$
$(light[1].dimmer.noMalAct.adjustedEntity =$
$evalLightScene(curLightScene, LightScene :: WINDOW))$
$\wedge$
$(light[2].pulse.noMalAct.adjustedEntity = 1 \leftrightarrow$
$\neg evalLightScene(curLightScene, LightScene :: WALL)$
$= 0) \wedge$
$(light[2].dimmer.noMalAct.adjustedEntity =$
$evalLightScene(curLightScene, LightScene :: WALL))))$

NL : Whenever the room is occupied and the light is not manually controlled and light scenes are enabled by the control panel, the window and wall lights are dimmed and switched on or off as necessary to achieve the ambient light level of the current light scene under consideration of the current light scene's lighting order.
In short: Under the mentioned conditions the current light scene has to be established.

Property *M_RL9*

Formal : $\Box((occ.occUsed \wedge \neg manualControl$
$\wedge lCPRoom.lightSceneSwitch.checked = 0$
$\wedge lCPRoom.ambientSwitch.checked = 1)$
$\rightarrow ((light[1].pulse.noMalAct.adjustedEntity = 1 \leftrightarrow$
$\neg evalAmbient(curAmbLightLev, LightScene :: WINDOW)$
$= 0) \wedge$
$(light[1].dimmer.noMalAct.adjustedEntity =$
$evalAmbient(curAmbLightLev, LightScene :: WINDOW))$
$\wedge$
$(light[2].pulse.noMalAct.adjustedEntity = 1 \leftrightarrow$
$\neg evalAmbient(curAmbLightLev, LightScene :: WALL)$
$= 0) \wedge$
$(light[2].dimmer.noMalAct.adjustedEntity =$
$evalAmbient(curAmbLightLev, LightScene :: WALL))))$

NL : Whenever the room is occupied and the light is not manually controlled, light scenes are disabled and ambient light settings are enabled by the control panel, the window and wall lights are dimmed and switched on or off as necessary to achieve the current ambient light level.
In short: Under the mentioned conditions the current ambient light setting is established.

Property *M_RL10*

Formal : $\Box(occ.occUsed \wedge \neg manualControl$
$\wedge lCPRoom.lightSceneSwitch.checked = 0$
$\wedge lCPRoom.ambientSwitch.checked = 0$
$\rightarrow ((light[1].pulse.noMalAct.adjustedEntity =$
$lCPRoom.winCeilLightSwitch.checked) \wedge$
$(lCPRoom.winCeilLightSwitch.checked = 1$
$\rightarrow light[1].dimmer.noMalAct.adjustedEntity = 100) \wedge$
$(lCPRoom.winCeilLightSwitch.checked = 0$
$\rightarrow light[1].dimmer.noMalAct.adjustedEntity = 0) \wedge$
$(light[2].pulse.noMalAct.adjustedEntity =$
$lCPRoom.wallCeilLightSwitch.checked) \wedge$
$(lCPRoom.wallCeilLightSwitch.checked = 1$
$\rightarrow light[2].dimmer.noMalAct.adjustedEntity = 100) \wedge$
$(lCPRoom.wallCeilLightSwitch.checked = 0$
$\rightarrow light[2].dimmer.noMalAct.adjustedEntity = 0)))$

NL : Whenever the room is occupied and the light is not manually controlled and light scenes and ambient light setting are disabled by the control panel, the window and wall lights are completely switched off or on as selected in the control panels ceiling light switches.
In short: Under the mentioned conditions the light is controlled by the ceiling light switches.

Property *M_RL11*

Formal : $\Box(manualControl$
$\rightarrow \forall p \in PERCENT :$
$(light[1].dimmer.noMalAct.adjustedEntity = p$
$\rightarrow (light[1].dimmer.noMalAct.adjustedEntity = p$
$\mathcal{W} \neg manualControl)) \wedge$
$\forall p \in PERCENT :$
$(light[2].dimmer.noMalAct.adjustedEntity = p$
$\rightarrow (light[2].dimmer.noMalAct.adjustedEntity = p$
$\mathcal{W} \neg manualControl)))$

NL : As long as the light is controlled manually, the dimmer settings of the window and wall light remain unchanged.

Property *M_RL12*

Formal : $\Box(([occ.occUsed] \wedge \blacklozenge_{\leq lCPFM.reoccDelay.checked} occ.occUsed)$
$\rightarrow curAmbLightLev = lCPRoom.ambLightLev.checked)$

NL : Whenever the room becomes occupied now and was occupied sometimes within the last T_RL_2 milliseconds and the ambient light level l was last chosen, the ambient light level l is the current ambient light level.

Property *M_RL13*

Formal : $\Box((([occ.occUsed] \wedge \blacksquare_{>lCPFM.reoccDelay.checked} \neg occ.occUsed) \rightarrow curAmbLightLev = defaultAmbLightLev)$

NL : Whenever the room becomes occupied now and has not been occupied during the last time T_RL_2, the current ambient light level is the default ambient light level.

Property *M_RL14*

Formal : $\Box(\forall l \in LIGHT_VALUES :$
$((occ.occUsed \wedge \neg[occ.occUsed] \wedge$
$[lCPRoom.ambLightLev.checked = l])$
$\rightarrow \Diamond_{\leq T_RL1} curAmbLightLev = l))$

NL : Whenever the room is occupied, but has not just become occupied now, and the ambient light level l becomes chosen, this ambient light level l is the current ambient light level within time T_RL_1.

Description Class BlindedRoomLight

Intention

This class provides all definitions to describe the control of lights common to any kind of room having sun blinds. It extends the definitions of the description class *RoomLight*. The special equippment of such rooms includes:

- *numSunBlinds* many blinds
- a control panel including possibilities to determine the settings of the sun blinds

The current setting of the sun blinds in a room intended by the light control depends on whether or not the machine assumes that there is a malfunction of the sensors needed to determine this setting.

Formal Parameters

Function $numSunBlinds \rightarrow NAT$

Intention : This parameter represents the number of sun blinds that belong to a room and which can be controlled by the machine.

Base Classes

Class *RoomLight*

Intention : All definitions of description class *RoomLight* are also provided here.

Signature

Object $RoomLight :: lCPRoom : LightCtrlPanelBlindedRoom($
$numSunBlinds = numSunBlinds\ )$

Intention : This object represents the light specific inputs of a user including possibilities to determine the settings of the sun blinds.
Postreferences : msref BlindedRoom : M_BR1
msref BlindedRoom : M_BR4

Timed Function $lightBlindSet : NAT(numSunBlinds) \rightarrow$
$THREE_PT_ACT_RANGE$

Intention : This function represents the current settings of the sun blinds in a room intended by the light control.
Scope : mh
Postreferences : msref BlindedRoom : M_BR3
msref BlindedRoom : M_BR5

Timed Function $lightBlindNoMalSet : NAT(numSunBlinds) \rightarrow$
$THREE_PT_ACT_RANGE$

Intention : This function represents the current settings of the sun blinds in a room intended by the light control if the machine assumes that there is no malfunction of the sensors needed to determine this setting.
Scope : mh
Postreferences : msref BlindedRoom : M_BR3

Timed Function $lightBlindMalSet : NAT(numSunBlinds) \rightarrow THREE_PT_ACT_RANGE$

Intention	: This function represents the current settings of the sun blinds in a room intended by the light control if the machine assumes that there is a malfunction of the sensors needed to determine this setting.
Scope	: mh
Postreferences	: msref BlindedRoom : M_BR3

Description Class RoomTemperature

Intention

This class provides all the definitions to describe the determination of the desired room temperature. For this the definitions of several objects are combined:

- an object representing the temperature operation modes, this object includes the occupancy determination for a room;
- an object representing the adaptive delay times used to determine occupancy;
- an object representing the control panel that a person can use to enter values controlling the computation of the desired room temperature.

Note that in this class it is not specified how the desired room temperature is achieved.

Signature

Object *modes* : *TempOperationModeSet*

Intention	: This object represents the operation modes of a room concerning temperature.
Postreferences	: msref RoomTemperature : M_RT1 msref RoomTemperature : M_RT2 msref RoomTemperature : M_RT3 msref RoomTemperature : M_RT4 msref RoomTemperature : M_RT5 msref RoomTemperature : M_RT6 msref RoomTemperature : M_RT7 msref RoomTemperature : M_RT8 msref RoomTemperature : M_RT11 msref RoomTemperature : M_RT12 msref BasicRoom : M_BaR6 msref Room : M_R2 msref BlindedRoom : M_BR1 msref BlindedRoom : M_BR2 rsref RoomTemperature : R_RT1

Object *adaptTimes* : *AdaptiveOccupancyTimes*

Intention	: This object represents the delay times used to determine occupancy if the facility manager has chosen the delay times to be adaptive.
Postreferences	: msref RoomTemperature : M_RT6 msref RoomTemperature : M_RT7 msref RoomTemperature : M_RT8

Object *cp* : *TemperatureCtrlPanel*

Intention : This object represents the temperature specific inputs of a user and the facility manager.
Postreferences : msref RoomTemperature : M_RT1
msref RoomTemperature : M_RT2
msref RoomTemperature : M_RT3
msref RoomTemperature : M_RT4
msref RoomTemperature : M_RT5
msref RoomTemperature : M_RT6
msref RoomTemperature : M_RT7
msref RoomTemperature : M_RT8
msref RoomTemperature : M_RT11
msref RoomTemperature : M_RT12
msref Floor : M_F3
msref Floor : M_F4
msref Floor : M_F5
msref Floor : M_F6
msref Floor : M_F7
msref Floor : M_F8
msref Floor : M_F9
msref Floor : M_F10
msref Floor : M_F11
msref Floor : M_F12

Timed Predicate *daytime*

Intention : This predicate is true during daytime, otherwise it is false. *Daytime* is the time between 6 am and 8 pm.
Scope : eh
Postreferences : rsref RoomTemperature : R_RT1

Timed Predicate *defaultTempOccDaytime*

Intention : This predicate represents the default value for occupancy concerning the temperature control during daytime in case of a malfunction of some sensors used to determine occupancy.
Scope : mh
Postreferences : msref RoomTemperature : M_RT9
rsref RoomTemperature : R_RT1

Timed Predicate *defaultTempOccNighttime*

Intention : This predicate represents the default value for occupancy concerning the temperature control during nighttime in case of a malfunction of some sensors used to determine occupancy.
Scope : mh
Postreferences : msref RoomTemperature : M_RT10
rsref RoomTemperature : R_RT1

Requirement Specifications

Property *R_RT1*

Formal : $\Box(modes.occ.malOcc.automaticDetermValue \leftrightarrow$
$(daytime \rightarrow defaultTempOccDaytime) \wedge$
$(\neg daytime \rightarrow defaultTempOccNighttime))$

NL : The value determined automatically by the machine in the case of a malfunction of some sensors used to determine occupancy concerning the temperature control depends on whether it is daytime or nighttime. The corresponding default values for the two time periods are used as the value determined automatically.

Property *R_RT2*

NonFormal : The use of solar radiation for heating during all times if heating is required or not disallowed, and the use of storage capacity of the building should always be preferred against the usage of the central heating unit.

Machine Specifications

Property *M_RT1*

Formal : $\Box(modes.nonFreezeTime \leftrightarrow$
$(cp.tCPRoom.nonFreezeOverride.checked = 0 \wedge$
$((cp.tCPFM.roomNonFreezeDefined \wedge$
$cp.tCPFM.nonFreezePeriod.actual) \vee$
$(\neg cp.tCPFM.roomNonFreezeDefined \wedge$
$cp.tCPFM.buildingNonFreezePeriod.actual))))$

NL : Non-freeze operation time is set if and only if the following conditions are fulfilled together:

- there is no user override for the non-freeze operation mode and
- – a room specific non-freeze time is set and currently valid or
  – a room specific non-freeze time is not set and the building wide non-freeze time is currently valid.

Property *M_RT2*

Formal : $\Box(modes.ComfortMode.desiredTemperature =$
$cp.comfortTemperature.value)$

NL : The desired temperature in the operation mode *ComfortMode* is the current comfort temperature (depending on the values entered by the facility manager and the room user using the control panel).

Property *M_RT3*

Formal : $\Box(modes.StandbyMode.desiredTemperature = cp.tCPFM.standbyTemperature.checked)$

NL : The desired temperature in the operation mode *StandbyMode* is the standby temperature set by the facility manager using the control panel.

Property *M_RT4*

Formal : $\Box(modes.OffMode.desiredTemperature = cp.tCPFM.offTemperature.checked)$

NL : The desired temperature in the operation mode *OffMode* is the off mode temperature set by the facility manager using the control panel.

Property *M_RT5*

Formal : $\Box(modes.NonFreezeMode.desiredTemperature = cp.tCPFM.nonFreezeTemperature.checked)$

NL : The desired temperature in the operation mode *NonFreezeMode* is the non-freeze temperature set by the facility manager using the control panel.

Property *M_RT6*

Formal :
$$\begin{aligned}&\Box((cp.tCPFM.adaptive.checked = 1 \\ &\quad \rightarrow modes.occ.noMalOcc.dt1 = adaptTimes.dt1Adaptive) \wedge \\ &\quad (cp.tCPFM.adaptive.checked = 0 \rightarrow \\ &\quad\ (modes.expUse.expectedUsage \rightarrow \\ &\quad\quad modes.occ.noMalOcc.dt1 \\ &\quad\quad\quad = cp.expectedUsageTimes.dt1.value) \wedge \\ &\quad\ (\neg modes.expUse.expectedUsage \rightarrow \\ &\quad\quad modes.occ.noMalOcc.dt1 \\ &\quad\quad\quad = cp.notExpectedUsageTimes.dt1.value)))\end{aligned}$$

NL : The value dt_1 used for determining occupancy is specified as follows:

- If the facility manager has chosen the adaptive mode for this room, dt_1 is the value computed by the adaptive learning function.
- If the facility manager has *not* chosen the adaptive mode for this room and usage of this room is expected, dt_1 is the value as it is computed by the entered values of the user and the facility manager for the case that usage of the room is expected.
- If the facility manager has *not* chosen the adaptive mode for this room and usage of this room is *not* expected, dt_1 is the value as it is computed by the entered values of the user and the facility manager for the case that usage of the room is *not* expected.

Property *M_RT7*

Formal : $\Box((cp.tCPFM.adaptive.checked = 1 \rightarrow modes.occ.noMalOcc.dt2 = adaptTimes.dt2Adaptive) \wedge (cp.tCPFM.adaptive.checked = 0 \rightarrow (modes.expUse.expectedUsage \rightarrow modes.occ.noMalOcc.dt2 = cp.expectedUsageTimes.dt2.value) \wedge (\neg modes.expUse.expectedUsage \rightarrow modes.occ.noMalOcc.dt2 = cp.notExpectedUsageTimes.dt2.value)))$

NL : The value dt_2 used for determining occupancy is specified as follows:

- If the facility manager has chosen the adaptive mode for this room, dt_2 is the value computed by the adaptive learning function.
- If the facility manager has *not* chosen the adaptive mode for this room and usage of this room is expected, dt_2 is the value as it is computed by the entered values of the user and the facility manager for the case that usage of the room is expected.
- If the facility manager has *not* chosen the adaptive mode for this room and usage of this room is *not* expected, dt_2 is the value as it is computed by the entered values of the user and the facility manager for the case that usage of the room is *not* expected.

Property *M_RT8*

Formal : $\Box((cp.tCPFM.adaptive.checked = 1 \rightarrow modes.occ.noMalOcc.dt3 = adaptTimes.dt3Adaptive) \wedge (cp.tCPFM.adaptive.checked = 0 \rightarrow (modes.expUse.expectedUsage \rightarrow modes.occ.noMalOcc.dt3 = cp.expectedUsageTimes.dt3.value) \wedge (\neg modes.expUse.expectedUsage \rightarrow modes.occ.noMalOcc.dt3 = cp.notExpectedUsageTimes.dt3.value)))$

NL : The value dt_3 used for determining occupancy is specified as follows:

- If the facility manager has chosen the adaptive mode for this room, dt_3 is the value computed by the adaptive learning function.
- If the facility manager has *not* chosen the adaptive mode for this room and usage of this room is expected, dt_3 is the value as it is computed by the entered values of the user and the facility manager for the case that usage of the room is expected.
- If the facility manager has *not* chosen the adaptive mode for this room and usage of this room is *not* expected, dt_3 is the value as it is computed by the entered values of the user and the facility manager for the case that usage of the room is *not* expected.

Property *M_RT9*

Formal : $\Box(defaultTempOccDaytime)$

NL : The default value for occupancy concerning the temperature control during daytime in case of a malfunction of some sensors used to determine occupancy is *occupancy*.

Property *M_RT10*

Formal : $\Box(\neg defaultTempOccNighttime)$

NL : The default value for occupancy concerning the temperature control during nighttime in case of a malfunction of some sensors used to determine occupancy is *no occupancy*.

Property *M_RT11*

Formal : $\Box(modes.expUse.expectedUsage \rightarrow modes.occ.noMalOcc.dt1 = cp.expectedUsageTimes.dt1.value \wedge modes.occ.noMalOcc.dt2 = cp.expectedUsageTimes.dt2.value \wedge modes.occ.noMalOcc.dt3 = cp.expectedUsageTimes.dt3.value)$

NL : The time delays used for determining occupancy of the temperature control in the case that a usage of a room is expected corresponds with those entered for this case.

Property *M_RT12*

Formal : $\Box(\neg modes.expUse.expectedUsage \rightarrow$
$modes.occ.noMalOcc.dt1 = cp.notExpectedUsageTimes.dt1.value \wedge$
$modes.occ.noMalOcc.dt2 = cp.notExpectedUsageTimes.dt2.value \wedge$
$modes.occ.noMalOcc.dt3 = cp.notExpectedUsageTimes.dt3.value)$

NL : The time delays used for determining occupancy of the temperature control in the case that a usage of a room is not expected corresponds with those entered for this case.

Description Class BlindedRoomTemperature

Intention

This class provides all definitions to describe the control of the temperature common to any kind of room having sun blinds. It extends the definitions of the description class *RoomTemperature*. The special equippment of such rooms includes:

- numSunBlinds many blinds

The current setting of the sun blinds in a room intended by the temperature control depends on whether or not the machine assumes that there is a malfunction of the sensors needed to determine this setting.

Formal Parameters

Function $numSunBlinds \rightarrow NAT$

Intention : This parameter represents the number of sun blinds that belong to a room and which can be controlled by the machine.

Base Classes

Class *RoomTemperature*

Intention : All definitions of description class *RoomTemperature* are also provided here.

Signature

Timed Function $tempBlindSet : NAT(numSunBlinds) \rightarrow THREE_PT_ACT_RANGE$

Intention : This function represents the current setting of the sun blinds in a room intended by the temperature control.
Scope : mh
Postreferences : msref BlindedRoom : M_BR2
msref BlindedRoom : M_BR5

Timed Function $tempBlindNoMalSet : NAT(numSunBlinds), TEMP \rightarrow THREE_PT_ACT_RANGE$

Intention : This function represents the current settings of the sun blinds in a room intended by the temperature control if the machine assumes that there is no malfunction of the sensors needed to determine this setting. These settings depend (at least) on the current desired room temperature (second argument).
Scope : mh
Postreferences : msref BlindedRoom : M_BR2

Timed Function *tempBlindMalSet* : $NAT(numSunBlinds), TEMP \rightarrow THREE_PT_ACT_RANGE$

Intention	: This function represents the current setting of the sun blinds in a room intended by the temperature control if the machine assumes that there is a malfunction of the sensors needed to determine this setting. These settings depend (at least) on the current desired room temperature (second argument).
Scope	: mh
Postreferences	: msref BlindedRoom : M_BR2

Description Class BasicRoom

Intention

This class provides all definitions to describe the behavior common to any kind of room. This includes the definitions of the light and temperature control in a room as well as the aggregation of the objects that are installed in each room. In this class we do not consider such objects for which the number of installed instances can vary from room to room, e.g. the number of installed radiators. These objects are integrated in the derived description class *Room*.

Each room is equipped with

- one motion detector
- three temperature sensors

Signature

Object *RL* : *RoomLight*

Intention	: This object represents the part that describes the behavior of a room concerning the light control.
Postreferences	: lsref BlindedRoom : RL msref BasicRoom : M_BaR5 msref Floor : M_F2

Object *RT* : *RoomTemperature*

Intention	: This object represents the part that describes the behavior of a room concerning the temperature control.
Postreferences	: lsref BlindedRoom : RT msref BasicRoom : M_BaR6 msref Room : M_R2 msref Floor : M_F3 msref Floor : M_F4 msref Floor : M_F5 msref Floor : M_F6 msref Floor : M_F7 msref Floor : M_F8 msref Floor : M_F9 msref Floor : M_F10 msref Floor : M_F11 msref Floor : M_F12

Object *md* : *MotionDetector*

Intention	: This object represents the room motion detector.
Postreferences	: msref BasicRoom : M_BaR5 msref BasicRoom : M_BaR6

Object *rt* : *RoomTempSensor*[3]

Intention : Each object represents one of the three air temperature sensors in a room.
Postreferences : msref BasicRoom : M_BaR1
msref BasicRoom : M_BaR2
msref BasicRoom : M_BaR3
msref BasicRoom : M_BaR4

Timed Function *valueEstimation* : *RoomTempSensor* ::
ROOM_TEMP, *RoomTempSensor* :: *ROOM_TEMP* →
RoomTempSensor :: *ROOM_TEMP*

Intention : This function represents the value determined for a room temperature sensor depending on the values of the other two room temperature sensors.
This function is used in the case of a malfunction of a room temperature sensor.
Scope : mh
Postreferences : msref BasicRoom : M_BaR1
msref BasicRoom : M_BaR2
msref BasicRoom : M_BaR3

Timed Predicate *malfunctionAllTempSens*

Intention : This predicate is true iff all temperature sensors of a room have a malfunction.
Scope : mh
Postreferences : msref BasicRoom : M_BaR4
msref Room : M_R1

Timed Predicate *malLight*

Intention : This predicate is true iff the machine assumes that the sensors, used to determine the settings of the actuators controlling the light, have such a serious malfunction that the settings of the actuators have to be determined using makeshift operations.
Scope : mh
Postreferences : msref BlindedRoom : M_BR3

Machine Specifications

Property *M_BaR1*

Formal : $\Box(rt[1].malSens.automaticDetermValue = valueEstimation(rt[2].noMalSens.convertedEntity, rt[3].noMalSens.convertedEntity))$
NL : If the first room temperature sensor has a malfunction its value is estimated using the other two room temperature sensors.

Property *M_BaR2*

Formal : $\Box(rt[2].malSens.automaticDetermValue = valueEstimation(rt[1].noMalSens.convertedEntity, rt[3].noMalSens.convertedEntity))$

NL : If the second room temperature sensor has a malfunction its value is estimated using the other two room temperature sensors.

Property *M_BaR3*

Formal : $\Box(rt[3].malSens.automaticDetermValue = valueEstimation(rt[1].noMalSens.convertedEntity, rt[2].noMalSens.convertedEntity))$

NL : If the third room temperature sensor has a malfunction its value is estimated using the other two room temperature sensors.

Property *M_BaR4*

Formal : $\Box(malfunctionAllTempSens \leftrightarrow rt[1].malSens.malDetMach \wedge rt[2].malSens.malDetMach \wedge rt[3].malSens.malDetMach)$

NL : All room temperature sensors have a malfunction iff the machine assumes for each room temperature sensor a malfunction.

Property *M_BaR5*

Formal : $\Box(RL.occ.noMalOcc.emptyMach \leftrightarrow md.usedValue = 0)$

NL : Concerning the light control the machine always assumes that there is currently no person in a room iff the value of the motion detector is 0.

Property *M_BaR6*

Formal : $\Box(RT.modes.occ.noMalOcc.emptyMach \leftrightarrow md.usedValue = 0)$

NL : Concerning the temperature control the machine always assumes that there is currently no person in a room iff the value of the motion detector is 0.

Description Class Room

Intention

This class extends the description class *BasicRoom* with the definitions concerning those objects of which in each room at least one instantiation is installed but where the concrete number of instantion can vary from room to room.

In addition to the objects introduced in description class *BasicRoom* each room is equipped with

- *numRad* many radiators
- *numDoorHallway* many doors to the hallway

Formal Parameters

Function $numRad \rightarrow NAT$

Intention : This parameter represents the number of radiators that belong to a room.

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.

Base Classes

Class *BasicRoom*

Intention : All definitions of description class *BasicRoom* are also provided here.

Signature

Object $rad : Radiator[numRad]$

Intention : Each object represents one radiator of a room.
Postreferences : msref Room : M_R3

Object $hallwayDoor : Door[numDoorHallway]$

Intention : Each object represents one door of a room to the hallway.

Timed Function $tempRadSet \rightarrow THREE_PT_ACT_RANGE$

Intention : This function represents the current setting of the radiators in a room intended by the temperature control.
Scope : mh
Postreferences : msref Room : M_R2
msref Room : M_R3

Timed Function *tempRadNoMalSet* : *TEMP* → *THREE_PT_ACT_RANGE*

Intention : This function represents the current setting of the radiators in a room intended by the temperature control if the machine assumes that there is no malfunction of the sensors needed to determine this setting. This setting depends (at least) on the current desired room temperature (first argument).
Scope : mh
Postreferences : msref Room : M_R2

Timed Function *tempRadMalSet* : *TEMP* → *THREE_PT_ACT_RANGE*

Intention : This function represents the current setting of the radiators in a room intended by the temperature control if the machine assumes that there is a malfunction of the sensors needed to determine this setting. This setting depends (at least) on the current desired room temperature (first argument). Further arguments can be the current outdoor temperature and the settings of the neighbour rooms.
Scope : mh
Postreferences : msref Room : M_R2

Timed Predicate *malTemp*

Intention : This predicate is true iff the machine assumes that the sensors, used to determine the settings of the actuators controlling the temperature, have such a serious malfunction that the settings of the actuators have to be determined using makeshift operations.
Scope : mh
Postreferences : msref Room : I_R1
msref Room : M_R1
msref Room : M_R2
msref BlindedRoom : M_BR2

Requirement Specifications

Property *R_R1*

NonFormal : The desired temperature shall be reached as fast as possible during the heating up state.
This requirement is a basis for the definition of the functions *tempRadNoMalSet* and *tempRadMalSet*.

Property *R_R2*

NonFormal : The currently desired temperature is maintained during the current temperature state as good as possible.
This requirement is a basis for the definition of the functions *tempRadNoMalSet* and *tempRadMalSet*.

Machine Specifications

Property *I_R1*

Formal : $\neg malTemp$
NL : Initially the machine does not assume that the sensors, used to determine the settings of the actuators controlling the temperature, have such a serious malfunction that the settings of the actuators have to be determined using makeshift operations.

Property *M_R1*

Formal : $\Box(malTemp \leftrightarrow malfunctionAllTempSens)$
NL : The machine assumes that the sensors, used to determine the settings of the actuators controlling the temperature, have such a serious malfunction that the settings of the actuators have to be determined using makeshift operations iff all temperature sensors of a room have a malfunction.

Property *M_R2*

Formal : $\Box((\neg malTemp \rightarrow tempRadSet = tempRadNoMalSet(RT.modes.roomDesiredTemperature)) \wedge (malTemp \rightarrow tempRadSet = tempRadMalSet(RT.modes.roomDesiredTemperature)))$
NL : If the machine does not assume a serious malfunction of the sensors, used to determine the settings of the actuators controlling the temperature, the current setting of the radiators in a room intended by the temperature control are as determined by the function *tempRadNoMalSet*.
If the machine assumes a serious malfunction of the sensors, used to determine the settings of the actuators controlling the temperature, the current setting of the radiators in a room intended by the temperature control are as determined by the function *tempRadMalSet*.

Property *M_R3*

Formal : $\Box(\forall n \in NAT(numRad) : rad[n].valveAct.noMalAct.adjustedEntity = tempRadSet)$
NL : The value adjusted by the machine for the radiator valves corresponds with the value intended by the temperature control for these actuators depending on the currently desired temperature.
Note that all radiator valves in a room are always set to the same value. A specific value for a single radiator is not determined.

Description Class BlindedRoom

Intention

This class provides all definitions specific to any kind of room with sun blinds. In addition to the definitions of the description class *Room* the control of the sun blinds is considered here. Since both, light and temperature control, can influence the setting of the sun blinds there can be conflicts. The resolution of possible conflicts is represented by the function *resolveBlindConflict*.

Formal Parameters

Function $numRad \rightarrow NAT$

Intention : This parameter represents the number of radiators that belong to a room.

Function $numSunBlinds \rightarrow NAT$

Intention : This parameter represents the number of sun blinds that belong to a room and which can be controlled by the machine.

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.

Base Classes

Class *Room*($numRad = numRad$, $numDoorHallway = numDoorHallway$)

Intention : All definitions of description class *Room* are also provided here.

Signature

Object $sb : SunBlind[numSunBlinds]$

Intention : Each object represents one sun blind of a room that can be controlled by the machine.
Postreferences : msref BlindedRoom : M_BR4
msref BlindedRoom : M_BR5

Object $BasicRoom :: RL : BlindedRoomLight($ $numSunBlinds = numSunBlinds$)

Intention : This object represents the light control in a blinded room.
Postreferences : msref BlindedRoom : M_BR1
msref BlindedRoom : M_BR3
msref BlindedRoom : M_BR4
msref BlindedRoom : M_BR5

Object $BasicRoom :: RT : BlindedRoomTemperature($
$numSunBlinds = numSunBlinds$)

Intention : This object represents the temperature control in a blinded room.
Postreferences : msref BlindedRoom : M_BR1
msref BlindedRoom : M_BR2
msref BlindedRoom : M_BR5

Timed Function *resolveBlindConflict* :
$NAT(numSunBlinds), THREE_PT_ACT_RANGE, THREE_PT_ACT_RANGE \rightarrow THREE_PT_ACT_RANGE$

Intention : This function represents the current settings of the sun blinds in a room intended by the machine. These settings are computed based on the intended settings of the temperature control (second argument) and the intended settings of the light control (third argument).
Scope : mh
Postreferences : msref BlindedRoom : M_BR5

Timed Predicate *manualBlind*

Intention : This predicate is true iff the machine assumes that the sun blinds are controlled manually and not by the control system.
Scope : mh
Postreferences : msref BlindedRoom : I_BR1
msref BlindedRoom : M_BR1
msref BlindedRoom : M_BR4
msref BlindedRoom : M_BR5

Domain Knowledge

Property *D_BR1*

Formal : $\Box(numSunBlinds = numRad)$
NL : The number of sun blinds in a blinded room equals the number of radiators in a blinded room.

Machine Specifications

Property *I_BR1*

Formal : $\neg manualBlind$
NL : Initially the sun blinds are not controlled manually.

Property *M_BR1*

Formal : $\Box(manualBlind \leftrightarrow RT.modes.occ.occUsed \wedge$
$\exists s \in NAT(numSunBlinds) :$
$RL.lCPRoom.sunBlindManual[s].checked = 1)$
NL : The machine assumes that the sun blinds are controlled manually iff a room is occupied according to the definition of this term used for temperature control and at least one sun blind is controlled manually.

Property *M_BR2*

Formal : $\Box(\forall s \in NAT(numSunBlinds) : ((\neg malTemp \rightarrow$
$RT.tempBlindSet(s) =$
$RT.tempBlindNoMalSet(s,$
$RT.modes.roomDesiredTemperature)) \wedge$
$(malTemp \rightarrow$
$RT.tempBlindSet(s) =$
$RT.tempBlindMalSet(s,$
$RT.modes.roomDesiredTemperature))))$

NL : If the machine does not assume a serious malfunction of the sensors, used to determine the settings of the actuators controlling the temperature, the current settings of the sun blinds in a room intended by the temperature control are as determined by the function *tempBlindNoMalSet*.
If the machine assumes a serious malfunction of the sensors, used to determine the settings of the actuators controlling the temperature, the current settings of the sun blinds in a room intended by the temperature control are as determined by the function *tempBlindMalSet*.

Property *M_BR3*

Formal : $\Box(\forall s \in NAT(numSunBlinds) :$
$((\neg malLight \rightarrow$
$RL.lightBlindSet(s) = RL.lightBlindNoMalSet(s)) \wedge$
$(malLight \rightarrow$
$RL.lightBlindSet(s) = RL.lightBlindMalSet(s))))$

NL : If the machine does not assume a serious malfunction of the sensors, used to determine the settings of the actuators controlling the light, the current settings of the sun blinds in a room intended by the light control are as determined by the function *lightBlindNoMalSet*.
If the machine assumes a serious malfunction of the sensors, used to determine the settings of the actuators controlling the light, the current settings of the sun blinds in a room intended by the light control are as determined by the function *lightBlindMalSet*.

Property *M_BR4*

Formal : $\Box(manualBlind \rightarrow$
$\forall s \in NAT(numSunBlinds) :$
$sb[s].blind.noMalAct.adjustedEntity =$
$RL.lCPRoom.sunBlindSwitch[s].checked)$

NL : If the sun blinds are controlled manually the value adjusted by the machine for each sun blind is equal to the corresponding switch status of the control panel.

Property *M_BR5*

Formal : $\Box(\neg manualBlind \rightarrow$
$\forall s \in NAT(numSunBlinds) :$
$sb[s].blind.noMalAct.adjustedEntity =$
$resolveBlindConflict(s,$
$RT.tempBlindSet(s),$
$RL.lightBlindSet(s)))$

NL : If the sun blinds are not controlled manually the value adjusted by the machine for each sun blind is equal to the value determined for the sun blinds by the function *resolveBlindConflict*.

Description Class RoomNeighb

Intention

This class provides all definitions that has to be added to the definitions of a room which is connected by a door to at least one neighbour room. Such rooms are equipped additionally with

- *numDoorNeighbour* many doors to neighbour rooms

Note that this description class is not derived from the description class *Room* in order to be able to combine this description class with each kind of room no matter whether it is for example a blinded office or a non–office without blinds.

Formal Parameters

Function *numDoorNeighbour* $\rightarrow$ *NAT*

Intention	: This parameter represents the number of doors of a room to neighbour rooms.

Signature

Object *neighbourDoor* : *Door*[*numDoorNeighbour*]

Intention	: Each object represents one door of a room to a neighbour room.
Postreferences	: dkref Floor : D_F1 dkref Floor : D_F2 dkref Floor : D_F3 dkref Floor : D_F4 dkref Floor : D_F5 dkref Floor : D_F6

Description Class Office

Intention

This class provides all definitions specific to all offices having no sun blinds. In addition to the definitions of the description class *Room* only the installation of the control panel is considered here.

Formal Parameters

Function $numRad \rightarrow NAT$

Intention : This parameter represents the number of radiators that belong to a room.

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.

Base Classes

Class $Room(\ numRad = numRad,\ numDoorHallway = numDoorHallway\ )$

Intention : All definitions of description class *Room* are also provided here.

Requirement Specifications

Property *R_O1*

NonFormal : The control panel should be installed moveably like a telephone in the offices.

Description Class OfficeNeighb

Intention

This class provides all definitions specific to all offices having no sun blinds and which are connected by at least one door to a neighbour room. In addition to the definitions of the description class *Office* the definitions of the description class *RoomNeighb* are also provided here.

Formal Parameters

Function $numRad \rightarrow NAT$

Intention : This parameter represents the number of radiators that belong to a room.

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.

Function $numDoorNeighbour \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to neighbour rooms.

Base Classes

Class *Office*($numRad = numRad$, $numDoorHallway = numDoorHallway$)

Intention : All definitions of description class *Office* are also provided here.

Class *RoomNeighb*($numDoorNeighbour = numDoorNeighbour$)

Intention : All definitions of description class *RoomNeighb* are also provided here.

Description Class BlindedOffice

Intention

This class provides all definitions specific to all offices with sun blinds and which are not connected by a door to a neighbour room. In addition to the definitions of the description class *Room* only the installation of the control panel is considered here.

Formal Parameters

Function $numRad \rightarrow NAT$

Intention : This parameter represents the number of radiators that belong to a room.

Function $numSunBlinds \rightarrow NAT$

Intention : This parameter represents the number of sun blinds that belong to a room and which can be controlled by the machine.

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.

Base Classes

Class *BlindedRoom*($numRad = numRad$, $numSunBlinds = numSunBlinds$, $numDoorHallway = numDoorHallway$)

Intention : All definitions of description class *BlindedRoom* are also provided here.

Requirement Specifications

Property *R_BO1*

NonFormal : The control panel should be installed movable like a telephone in an office.

Description Class BlindedOfficeNeighb

Intention

This class provides all definitions specific to all offices with sun blinds and which are connected by at least one door to a neighbour room. In addition to the definitions of the description class *BlindedOffice* the definitions of the description class *RoomNeighb* are also provided here.

Formal Parameters

Function $numRad \rightarrow NAT$

Intention : This parameter represents the number of radiators that belong to a room.

Function $numSunBlinds \rightarrow NAT$

Intention : This parameter represents the number of sun blinds that belong to a room and which can be controlled by the machine.

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.

Function $numDoorNeighbour \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to neighbour rooms.

Base Classes

Class *BlindedOffice*($numRad = numRad$, $numSunBlinds = numSunBlinds$, $numDoorHallway = numDoorHallway$)

Intention : All definitions of description class *BlindedOffice* are also provided here.

Class *RoomNeighb*($numDoorNeighbour = numDoorNeighbour$)

Intention : All definitions of description class *RoomNeighb* are also provided here.

Description Class NoOffice

Intention

This class provides all definitions specific to rooms that are no offices and that have no sun blinds. In addition to the definitions of the description class *Room* only the installation of the control panel is considered here.

Formal Parameters

Function $numRad \rightarrow NAT$

Intention : This parameter represents the number of radiators that belong to a room.

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.

Base Classes

Class $Room(\ numRad = numRad,\ numDoorHallway = numDoorHallway\ )$

Intention : All definitions of description class *Room* are also provided here.

Requirement Specifications

Property *R_NO1*

NonFormal : The control panel should be installed near a door to the hallway.

Description Class BlindedNoOffice

Intention

This class provides all definitions specific to rooms that are no offices and that have sun blinds. In addition to the definitions of the description class *Room* only the installation of the control panel is considered here.

Formal Parameters

Function $numRad \rightarrow NAT$

Intention : This parameter represents the number of radiators that belong to a room.

Function $numSunBlinds \rightarrow NAT$

Intention : This parameter represents the number of sun blinds that belong to a room and which can be controlled by the machine.

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.

Base Classes

Class *BlindedRoom*($numRad = numRad$, $numSunBlinds = numSunBlinds$, $numDoorHallway = numDoorHallway$)

Intention : All definitions of description class *BlindedRoom* are also provided here.

Requirement Specifications

Property *R_BNO1*

NonFormal : The control panel should be installed near a door to the hallway.

Description Class BlindedNoOfficeNeighb

Intention

This class provides all definitions specific to rooms that are no offices and that have sun blinds and which are connected by at least one door to a neighbour room. In addition to the definitions of the description class *BlindedNoOffice* the definitions of the description class *RoomNeighb* are also provided here.

Formal Parameters

Function $numRad \rightarrow NAT$

Intention : This parameter represents the number of radiators that belong to a room.

Function $numSunBlinds \rightarrow NAT$

Intention : This parameter represents the number of sun blinds that belong to a room and which can be controlled by the machine.

Function $numDoorHallway \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to the hallway.

Function $numDoorNeighbour \rightarrow NAT$

Intention : This parameter represents the number of doors of a room to neighbour rooms.

Base Classes

Class *BlindedNoOffice*($numRad = numRad$, $numSunBlinds = numSunBlinds$, $numDoorHallway = numDoorHallway$)

Intention : All definitions of description class *BlindedNoOffice* are also provided here.

Class *RoomNeighb*($numDoorNeighbour = numDoorNeighbour$)

Intention : All definitions of description class *RoomNeighb* are also provided here.

Description Class StairCase

Intention

This class provides all definitions to describe the entities belonging to a staircase which are relevant for the considered system. This is only a motion detector.

Signature

Object *md* : *MotionDetector*

Intention	: This object represents a motion detector placed in a staircase in front of the hallway door.

Description Class Floor

Intention

This class is the main class. It contains and combines all the objects belonging to the considered system. These objects are

- the rooms of the floor
- the hallway sections of the floor
- the parts of the staircases belonging to the floor
- the outdoor light sensors
- the outdoor temperature sensor

Signature

Object *o415* : *Office*($numRad = 2$, $numDoorHallway = 1$)

Intention : This object represents the office with room number 415. This office is equipped with

- two radiators
- one door to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES

Object *o417* : *Office*($numRad = 2$, $numDoorHallway = 1$)

Intention : This object represents the office with room number 417. This office is equipped with

- two radiators
- one door to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES

Object *o419* : *Office*($numRad = 2$, $numDoorHallway = 1$)

Intention : This object represents the office with room number 419. This office is equipped with

- two radiators
- one door to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES

Object *o429* : *Office*(*numRad* = 2, *numDoorHallway* = 1)

Intention : This object represents the office with room number 429. This office is equipped with

- two radiators
- one door to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES

Object *o425* : *OfficeNeighb*(*numRad* = 2, *numDoorHallway* = 1, *numDoorNeighbour* = 1)

Intention : This object represents the office with room number 425. This office is equipped with

- two radiators
- one door to the hallway
- one door to a neighbour room

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES
dkref Floor : D_F3

Object *o431* : *OfficeNeighb*(*numRad* = 2, *numDoorHallway* = 1, *numDoorNeighbour* = 1)

Intention : This object represents the office with room number 431. This office is equipped with

- two radiators
- one door to the hallway
- one door to a neighbour room

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES
dkref Floor : D_F2

Object *o435* : *OfficeNeighb*($numRad = 2$, $numDoorHallway = 1$, $numDoorNeighbour = 1$)

Intention : This object represents the office with room number 435. This office is equipped with

- two radiators
- one door to the hallway
- one door to a neighbour room

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES
dkref Floor : D_F1

Object *o423* : *OfficeNeighb*($numRad = 2$, $numDoorHallway = 1$, $numDoorNeighbour = 2$)

Intention : This object represents the office with room number 423. This office is equipped with

- two radiators
- one door to the hallway
- two doors to neighbour rooms

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES
dkref Floor : D_F3
dkref Floor : D_F4

Object *o433* : *OfficeNeighb*($numRad = 2$, $numDoorHallway = 1$, $numDoorNeighbour = 2$)

Intention : This object represents the office with room number 433. This office is equipped with

- two radiators
- one door to the hallway
- two doors to neighbour rooms

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES
dkref Floor : D_F1
dkref Floor : D_F2

Object *o421* : *OfficeNeighb*(*numRad* = 3, *numDoorHallway* = 1, *numDoorNeighbour* = 1)

Intention : This object represents the office with room number 421. This office is equipped with

- three radiators
- one door to the hallway
- one door to a neighbour room

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES
dkref Floor : D_F4

Object *m427* : *NoOffice*(*numRad* = 5, *numDoorHallway* = 1)

Intention : This object represents the meeting room with room number 427. It is equipped with

- five radiators
- one door to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : MEETING_ROOMS

Object *cl411* : *NoOffice*(*numRad* = 9, *numDoorHallway* = 2)

Intention : This object represents the computer lab with room number 411. It is equipped with

- nine radiators
- two doors to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : COMPUTER_LABS

Object *cl426* : *BlindedNoOffice*(*numRad* = 3, *numSunBlinds* = 3, *numDoorHallway* = 1)

Intention : This object represents the computer lab with room number 426. It is equipped with

- three radiators
- three sun blinds
- one door to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : COMPUTER_LABS

Object *o424* : *BlindedOffice*(*numRad* = 2, *numSunBlinds* = 2, *numDoorHallway* = 1)

Intention : This object represents the office with room number 424. It is equipped with

- two radiators
- two sun blinds
- one door to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES

Object *cl422* : *BlindedNoOfficeNeighb*(*numRad* = 3, *numSunBlinds* = 3, *numDoorHallway* = 1, *numDoorNeighbour* = 1)

Intention : This object represents the computer lab with room number 424. It is equipped with

- three radiators
- three sun blinds
- one door to the hallway
- one door to a neighbour room

Postreferences : lsref Floor : ROOMS
lsref Floor : COMPUTER_LABS

Object *cl418* : *BlindedNoOfficeNeighb*(*numRad* = 2, *numSunBlinds* = 2, *numDoorHallway* = 1, *numDoorNeighbour* = 2)

Intention : This object represents the computer lab with room number 418. It is equipped with

- two radiators
- two sun blinds
- one door to the hallway
- two doors to neighbour rooms

Postreferences : lsref Floor : ROOMS
lsref Floor : COMPUTER_LABS
dkref Floor : D_F5

Object *o416* : *BlindedOfficeNeighb*(*numRad* = 2, *numSunBlinds* = 2, *numDoorHallway* = 1, *numDoorNeighbour* = 2)

Intention : This object represents the office with room number 416. It is equipped with

- two radiators
- two sun blinds
- one door to the hallway
- two doors to neighbour rooms

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES
dkref Floor : D_F5
dkref Floor : D_F6

Object *o414* : *BlindedOfficeNeighb*(*numRad* = 2, *numSunBlinds* = 2, *numDoorHallway* = 1, *numDoorNeighbour* = 1)

Intention : This object represents the office with room number 414. It is equipped with

- two radiators
- two sun blinds
- one door to the hallway
- one door to a neighbour rooms

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES
dkref Floor : D_F6

Object *o412* : *BlindedOffice*(*numRad* = 3, *numSunBlinds* = 3, *numDoorHallway* = 1)

Intention : This object represents the office with room number 412. It is equipped with

- three radiators
- three sun blinds
- one door to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : OFFICES

Object *cl410* : *BlindedNoOffice*(*numRad* = 10, *numSunBlinds* = 10, *numDoorHallway* = 3)

Intention : This object represents the computer lab with room number 410. It is equipped with

- ten radiators
- ten sun blinds
- three doors to the hallway

Postreferences : lsref Floor : ROOMS
lsref Floor : COMPUTER_LABS

Object *h1* : *HallwaySection*

Intention : This object represents the hallway in the east of floor 32/4.
Postreferences : lsref Floor : HALLWAYS

Object *h2* : *HallwaySection*

Intention : This object represents the hallway in the middle of floor 32/4.
Postreferences : lsref Floor : HALLWAYS

Object *h3* : *HallwaySection*

Intention : This object represents the hallway in the west of floor 32/4.
Postreferences : lsref Floor : HALLWAYS

Object *ols1* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the north-east direction.
Postreferences : msref Floor : M_F1

Object *ols2* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the south-west direction.
Postreferences : msref Floor : M_F1

Object *ols3* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the north-west direction.
Postreferences : msref Floor : M_F1

Object *ols4* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the south-east direction.
Postreferences : msref Floor : M_F1

Object *ols5* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the north direction.
Postreferences : msref Floor : M_F1

Object *ols6* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the south direction.
Postreferences : msref Floor : M_F1

Object *ots* : *OutdoorTempSensor*

Intention : This object represents the outdoor temperature sensor.

Object *scw* : *StairCase*

Intention : This object represents the staircase in the west of floor 32/4.

Object *sce* : *StairCase*

Intention : This object represents the staircase in the east of floor 32/4.

Domain $ROOMS = \{o435, o433, o431, o429, o425, o423, o421, o419, o417, o415, o424, o416, o414, o412, m427, cl411, cl426, cl422, cl418, cl410\} \subseteq BasicRoom$

Intention : This domain contains all the rooms that have to be considered.
Postreferences : lsref Floor : OFFICES
lsref Floor : MEETING_ROOMS
lsref Floor : COMPUTER_LABS
msref Floor : M_F2
msref Floor : M_F3
msref Floor : M_F4
msref Floor : M_F5
msref Floor : M_F6

Domain $OFFICES = \{o435, o433, o431, o429, o425, o423, o421, o419, o417, o415, o424, o416, o414, o412\} \subseteq ROOMS$

Intention : This domain contains all the offices that have to be considered.
Postreferences : msref Floor : M_F7
msref Floor : M_F8

Domain $MEETING_ROOMS = \{m427\} \subseteq ROOMS$

Intention : This domain contains all the meeting rooms that have to be considered.
Postreferences : msref Floor : M_F9
msref Floor : M_F10

Domain $COMPUTER_LABS = \{cl411, cl426, cl422, cl418, cl410\} \subseteq ROOMS$

Intention : This domain contains all the rooms that have to be considered.
Postreferences : msref Floor : M_F11
msref Floor : M_F12

Domain $HALLWAYS = \{h1, h2, h3\} \subseteq HallwaySection$

Intention : This domain contains all the hallway sections that have to be considered.
Postreferences : rsref Floor : R_F1

Time Constant *TmalReact*

Intention	: The time delay between a malfunction of the outdoor light sensor and switching on the lights if the hallway is occupied.
Scope	: ev
Postreferences	: rsref Floor : R_F1

Timed Predicate *outLightSensorFaultMach*

Intention	: This predicate is true iff the machine assumes that some outdoor light sensors have a malfunction.
Scope	: mh
Postreferences	: msref Floor : M_F1 msref Floor : M_F2 rsref Floor : R_F1

Domain Knowledge

Property *D_F1*

Formal	: $\Box(o435.neighbourDoor[1] = o433.neighbourDoor[1])$
NL	: The two rooms with room numbers 435 and 433 are connected by a door.

Property *D_F2*

Formal	: $\Box(o433.neighbourDoor[2] = o431.neighbourDoor[1])$
NL	: The two rooms with room numbers 433 and 431 are connected by a door.

Property *D_F3*

Formal	: $\Box(o425.neighbourDoor[1] = o423.neighbourDoor[1])$
NL	: The two rooms with room numbers 425 and 423 are connected by a door.

Property *D_F4*

Formal	: $\Box(o423.neighbourDoor[2] = o421.neighbourDoor[1])$
NL	: The two rooms with room numbers 423 and 421 are connected by a door.

Property *D_F5*

Formal	: $\Box(cl418.neighbourDoor[2] = o416.neighbourDoor[1])$
NL	: The two rooms with room numbers 418 and 416 are connected by a door.

Property *D_F6*

Formal	: $\Box(o416.neighbourDoor[1] = o414.neighbourDoor[1])$
NL	: The two rooms with room numbers 416 and 414 are connected by a door.

Requirement Specifications

Property *R_F1*

Formal : $\Box(\forall h \in HALLWAYS :$
$(outLightSensorFaultMach \wedge h.occ.occUsed$
$\Rightarrow_{\leq T_{malReact}} h.light[1].ls.noMalSens.envEntity = 1))$

NL : Whenever some outdoor light sensors are not working correctly and the hallway is occupied for at least $T_{malReact}$ time units, then eventually within this time span, the light in the hallway is on and remains on at least as long as the precondition is true.

Property *R_F2*

NonFormal : Energy consumption should be reduced as far as possible during maintaining of comfortable working conditions for all users.

Property *R_F3*

NonFormal : If a malfunction occurs, the control system supports the facility manager by finding the reason.

Property *R_F4*

NonFormal : The system provides reports on current and past energy consumption.

Property *R_F5*

NonFormal : All malfunctions and unusual conditions are stored and reported on request.

Property *R_F6*

NonFormal : All hardware connections have to be made according to DIN standards.

Property *R_F7*

NonFormal : No hazardous condition for persons, inventory or building are allowed.

Property *R_F8*

NonFormal : The system warns about unreasonable inputs.

Property *R_F9*

NonFormal : If the whole control system has a serious malfunction, complete manual operation should be possible.

Property *R_F10*

NonFormal : The control system shall be designed such that it can be easily changed and expanded.

Property *R_F11*

NonFormal : The control system provides basic energy saving procedures automatically for control comfort. If the facility manager deviates from such settings, the system provides a report on expected results.

Machine Specifications

Property *M_F1*

Formal : $\Box(outLightSensorFaultMach \leftrightarrow ols1.malSens.malDetMach \vee ols2.malSens.malDetMach \vee ols3.malSens.malDetMach \vee ols4.malSens.malDetMach \vee ols5.malSens.malDetMach \vee ols6.malSens.malDetMach)$

NL : The machine assumes that some outdoor light sensor has a malfunction iff one of the six outdoor light sensors has a malfunction.

Property *M_F2*

Formal : $\Box(\forall r \in ROOMS : (outLightSensorFaultMach \rightarrow r.RL.curStandLightScene = r.RL.faultStandLightScene))$

NL : Whenever the machine assumes that some outdoor light sensors are not working correctly, then the standard light scene is the standard fault light scene, i.e. all ceiling lights are on.

Property *M_F3*

Formal : $\Box(\forall r \in ROOMS : (\forall s \in ROOMS : r.RT.cp.tCPFM.heatingPeriod = s.RT.cp.tCPFM.heatingPeriod))$

NL : The heating period is the same for all rooms.

Property *M_F4*

Formal : $\Box(\forall r \in ROOMS : (\forall s \in ROOMS : r.RT.cp.tCPFM.buildingNonFreezePeriod = s.RT.cp.tCPFM.buildingNonFreezePeriod))$

NL : The period within which the non freeze mode is set for the whole building is the same for all rooms.

Property *M_F5*

Formal : $\Box(\forall r \in ROOMS : (\forall s \in ROOMS : r.RT.cp.tCPFM.nonFreezeTemperature = s.RT.cp.tCPFM.nonFreezeTemperature))$

NL : The temperature that should be applied if the operation mode *NonFreezeMode* is active is the same for all rooms.

Property *M_F6*

Formal : $\Box(\forall r \in ROOMS : (\forall s \in ROOMS : r.RT.cp.tCPFM.adaptive = s.RT.cp.tCPFM.adaptive))$

NL : The modus set by the facility manager indicating whether the delay times used for determining occupancy are adaptive or not is always the same for all rooms.

Property *M_F7*

Formal : $\Box(\forall r \in OFFICES : (\forall s \in OFFICES : r.RT.cp.tCPFM.standbyTemperature = s.RT.cp.tCPFM.standbyTemperature))$

NL : The temperature that should be applied in the *standbyMode* is the same for all offices.

Property *M_F8*

Formal : $\Box(\forall r \in OFFICES : (\forall s \in OFFICES : r.RT.cp.tCPFM.offTemperature = s.RT.cp.tCPFM.offTemperature))$

NL : The temperature that should be applied in the *OffMode* is the same for all offices.

Property *M_F9*

Formal : $\Box(\forall r \in MEETING_ROOMS : (\forall s \in MEETING_ROOMS : r.RT.cp.tCPFM.standbyTemperature = s.RT.cp.tCPFM.standbyTemperature))$

NL : The temperature that should be applied in the *standbyMode* is the same for all meeting rooms.

Property *M_F10*

Formal : $\Box(\forall r \in MEETING_ROOMS : (\forall s \in MEETING_ROOMS : r.RT.cp.tCPFM.offTemperature = s.RT.cp.tCPFM.offTemperature))$

NL : The temperature that should be applied in the *OffMode* is the same for all meeting rooms.

Property *M_F11*

Formal : $\Box(\forall r \in COMPUTER_LABS : (\forall s \in COMPUTER_LABS : r.RT.cp.tCPFM.standbyTemperature = s.RT.cp.tCPFM.standbyTemperature))$

NL : The temperature that should be applied in the *standbyMode* is the same for all computer labs.

Property *M_F12*

Formal : $\Box(\forall r \in COMPUTER_LABS : (\forall s \in COMPUTER_LABS : r.RT.cp.tCPFM.offTemperature = s.RT.cp.tCPFM.offTemperature))$

NL : The temperature that should be applied in the *OffMode* is the same for all computer labs.

Index

A

active 100, 103, 104
actual 14, 127
Actuator 41, 47, 50, 52
adaptive 83, 128–130, 163
AdaptiveOccupancyTimes 90, 125
adaptTimes 125, 128–130
ADJUSTED_DOMAIN . 42, 43, 47, 48, 50, 52–54, 56
adjustedEntity .. 43, 50, 52, 65, 68, 69, 111, 120, 121, 139, 142, 143
allLightsOff 108, 109
allowedRange 15, 17
ambientIlluminance 6
ambientSwitch 78, 119–121
ambLightLev 78, 121, 122
AreaLight 106, 112, 115
AreaLightCtrlPanelFM 74, 75, 107
automaticDetermValue . 25, 27, 35, 95, 111, 127, 135, 136
automaticDetMal 31, 49, 65, 97, 98

B

BACKWARD 1, 3, 64, 65
BasicHumanMachineInterface 8
BasicRoom 134, 137, 140, 160
BINARY 2, 30, 37, 49, 50, 54, 74, 78–80, 83, 84, 94, 97, 109
BinaryActuator 54, 55
BinarySensor 37–40, 61
blind 61, 63–65, 142, 143
BlindedNoOffice 150, 151, 156, 159
BlindedNoOfficeNeighb 151, 157
BlindedOffice 147, 148, 157, 158
BlindedOfficeNeighb 148, 158
BlindedRoom 140, 147, 150
BlindedRoomLight 123, 140
BlindedRoomTemperature ... 132, 140
buildingNonFreezePeriod . 82, 127, 163
BuildingUnits 105

C

CALENDAR_TIME 3, 14, 87
CalendarInterval 14, 82
checked 10, 11, 13, 14, 16, 17, 27, 31, 49, 73–76, 83, 84, 95, 98, 110, 111, 118–122, 127–130, 141, 142
CHECKED_DOM 9–12, 15, 24, 30, 49, 74, 75, 77–80, 82–84, 94, 97
checkTime 11
chosenLightScene 77, 119
cl410 ..... 159, 160
cl411 ..... 156, 160
cl418 ..... 157, 160, 161
cl422 ..... 157, 160
cl426 ..... 156, 160
CombinedSensorsAndActuators 59
COMFORT_TEMP 3, 86
ComfortMode 101, 103, 104, 127
comfortTemperature 86, 127
COMPUTER_LABS 160, 164
ControlledDynamicInterval 18, 87
ControlledInput 15, 18, 72, 73, 86
ControlledLight 67, 70, 107
ControlSystemActive 55, 68
conversionTime .. 21, 23, 32–34, 36, 37
CONVERTED_DOMAIN 20, 22, 28–30, 32–34, 36, 37
convertedEntity 22, 23, 31, 65, 68, 135, 136
cp 126–131, 163, 164
csa 68, 69
curAmbLightLev 117, 118, 120–122
curLightScene 116, 118–120
currentDate 3, 14, 88
currentHour 3, 88
curStandLightScene 116, 118, 119, 163

D

D_AL1 109
D_AL2 109
D_BA1 54
D_BR1 141
D_BS1 37

D_BS2 37
D_BS3 37
D_CL1 68
D_CL2 68
D_CSA1 55
D_DC1 39
D_DIM1 53
D_EU1 88
D_F1 161
D_F2 161
D_F3 161
D_F4 161
D_F5 161
D_F6 161
D_L1 38
D_MD1 40
D_NMS1 22
D_NMS2 23
D_OLS1 34
D_OLS2 34
D_OTS1 33
D_OTS2 33
D_PB1 58
D_PB2 58
D_PUL1 51
D_RA1 52
D_RTS1 32
D_RTS2 32
D_RV1 57
D_SBL1 63
D_SBL10 64
D_SBL2 63
D_SBL3 63
D_SBL4 63
D_SBL5 63
D_SBL6 64
D_SBL7 64
D_SBL8 64
D_SBL9 64
D_TA1 50
D_TA2 50
D_TPA1 56
D_WTS1 36
D_WTS2 36
Datatypes 4
daytime 126, 127
default 15–17, 73
defaultAmbientLightLevel 75, 118
defaultAmbLightLev 117, 118, 122
defaultLightScene 75, 118
defaultTempOccDaytime 126, 127, 130
defaultTempOccNighttime ... 126, 127, 130
DEGREE 61, 62
desiredTemperature 100, 103, 104, 127, 128
determineLightScene 117, 119
DimmableControlledLight 70, 115
Dimmer 53, 70
dimmer 70, 120, 121
Door 60, 137, 144
doorClosedContact 60
DoorContact 39, 60
doorLockContact 60
doorOpenContact 60
dt1 .. 72, 73, 91, 93, 110, 128, 130, 131
DT1_RANGE 72
dt1Adaptive 90, 128
dt2 ..... 72, 73, 91, 93, 110, 129–131
DT2_RANGE 72
dt2Adaptive 90, 129
dt3 ..... 73, 92, 93, 111, 130, 131
DT3_RANGE 72, 73
dt3Adaptive 90, 130
DynamicInterval 12, 14, 15

E

eastMd 112, 114
emptyMach 92, 93, 114, 136
end 12–14, 17, 18, 88
entered 9, 11
EnteredValue .9, 12, 15, 24, 30, 49, 74, 75, 77–80, 82–84, 94, 97
ENV_DOMAIN ..20–22, 28, 29, 32–34, 36, 37, 42, 43, 47, 48, 50, 52–54, 56
envEntity 21, 22, 43, 50, 52, 63, 64, 68, 109, 110, 113, 162
evalAmbient 118, 120
evalLightScene 117, 120

evalSaveLight 109, 111
ExpectedUsage 87, 101
expectedUsage ... 87, 88, 103, 128–131
expectedUsageTimes 86, 128–130
expUse 101, 103, 128–131

F

faultStandLightScene 116, 163
firstAmbientLighting 6
firstDeskLighting 7
Floor 153
FMInformed 26, 27, 45, 46
FORWARD 1, 3, 64, 65

G

Global Signature 1
GroupOccupancy 89

H

h1 ..... 159, 160
h2 ..... 159, 160
h3 ..... 159, 160
hallwayDoor 137
HALLWAYS 160, 162
HallwaySection 112, 159, 160
heatingPeriod 82, 163
holidayTime 87, 88
HOURS 3, 87
HumanMachineInterface 71
hwLightUncontroll 113

I

I_A1 49
I_AL1 110
I_AL2 110
I_BR1 141
I_CIN1 16
I_CIN2 16
I_CL1 68
I_LCPFM1 74
I_LCPFM2 74
I_LCPR1 75
I_MA1 45
I_MA2 45
I_MO1 95
I_MS1 27
I_MS2 26
I_MS3 26
I_OCC1 98
I_R1 139
I_RL1 118
I_RL2 118
I_RL3 118
I_S1 31
INPUT_DOM 15, 16, 18, 72, 73, 86
INT_DOM 12, 14, 15, 18, 87
INTEGER 1

L

LAMPS 5–7, 117, 118
lastCorrectOutLight 34, 35
lCPFM ... 107, 110, 111, 115, 118, 119, 121, 122
lCPRoom 116, 119–123, 141, 142
leq 12–14, 17, 18, 88
light ... 58, 68, 107, 109–111, 113, 115, 120, 121, 162
LIGHT_VALUES .2, 6, 34, 75, 78, 117, 118, 122
lightBlindMalSet 124, 142
lightBlindNoMalSet 123, 142
lightBlindSet 123, 142, 143
LightCtrlPanelBlindedRoom ... 80, 123
LightCtrlPanelRoom 77, 80, 116
lightFMControlled 108, 110
lightOn 74, 110
LightScene 5, 7, 75, 116–120
LightSceneOffice 7
lightSceneSwitch 79, 119–121
LightSensor 38, 67
lowerEndContact 61, 64, 65
ls 67, 68, 109–111, 113, 162

M

m427 ..... 156, 160
M_A1 49
M_AL1 110
M_AL10 111
M_AL2 110
M_AL3 110
M_AL4 110
M_AL5 111

M_AL6 111
M_AL7 111
M_AL8 111
M_AL9 111
M_BaR1 135
M_BaR2 136
M_BaR3 136
M_BaR4 136
M_BaR5 136
M_BaR6 136
M_BR1 141
M_BR2 142
M_BR3 142
M_BR4 142
M_BR5 143
M_CDI1 18
M_CIN1 16
M_CIN2 17
M_CIN3 17
M_CL1 68
M_CL2 69
M_CL3 69
M_CL4 69
M_DI1 13
M_F1 163
M_F10 164
M_F11 164
M_F12 164
M_F2 163
M_F3 163
M_F4 163
M_F5 163
M_F6 163
M_F7 164
M_F8 164
M_F9 164
M_HS1 113
M_HS2 113
M_HS3 114
M_LCPR1 76
M_LS1 6
M_LSO1 7
M_MO1 95
M_MS1 27
M_NMO1 93
M_OCC1 98
M_OCC2 98
M_ODT1 73
M_ODT2 73
M_ODT3 73
M_R1 139
M_R2 139
M_R3 139
M_RL1 118
M_RL10 121
M_RL11 121
M_RL12 121
M_RL13 122
M_RL14 122
M_RL2 118
M_RL3 118
M_RL4 119
M_RL5 119
M_RL6 119
M_RL7 119
M_RL8 120
M_RL9 120
M_RT1 127
M_RT10 130
M_RT11 130
M_RT12 131
M_RT2 127
M_RT3 128
M_RT4 128
M_RT5 128
M_RT6 128
M_RT7 129
M_RT8 130
M_RT9 130
M_S1 31
M_S2 31
M_S3 31
M_SBL1 65
M_SBL2 65
M_TCPF1 83
M_TCPR1 84
M_TOMS1 103
M_TOMS2 103
M_TOMS3 103
M_TOMS4 103

M_TOMS5 103
M_TOMS6 104
M_TOMS7 104
M_TOMS8 104
MAL_CONVERTED_DOMAIN 24, 25, 30
malAct 48, 49
malDetEnv 25, 44
malDetMach 25, 27, 31, 44, 46, 49, 136, 163
MalfunctionActuator 44, 48
malfunctionAllTempSens 135, 136, 139
MalfunctionOccupancy 94, 96
MalfunctionSensor 24, 30
malLight 135, 142
malOcc 96, 98, 111, 127
malOccDetEnv 94
malOccDetMach 94, 98
malSens 30, 31, 35, 135, 136, 163
malTemp 138, 139, 142
manualBlind 141–143
manualControl .. 68, 69, 109, 111, 113, 119–121
manualEnteredMal .. 30, 31, 49, 97, 98
manualEnteredOcc 94, 95
manualEnteredVal 24, 27
md 134, 136, 152
meanTemp 66
MEASURED_DOMAIN 20, 22, 28, 29, 32–34, 36, 37
measuredEntity 22, 23
MEETING_ROOMS 160, 164
middleMd 112, 114
modes 125, 127–131, 136, 139, 141, 142
modifyConversion 22, 23, 37
modifyReaction 22, 37, 52, 54, 56
MotionDetector 40, 112, 134, 152
moveDown 62–64
moveStop 62–64
moveUp 62–64

N

name 5
NAT ...2, 3, 61, 80, 106, 109–111, 123, 124, 132, 133, 137, 139–151
nearestValue 11
nearHallwayDoor 113
neighbourDoor 144, 161
noFaultStandLightScene 117, 118
noMalAct 48, 50–53, 55, 57, 63–65, 68, 69, 111, 120, 121, 139, 142, 143
NoMalfunctionActuator 42, 48
NoMalfunctionOccupancy 91, 96
NoMalfunctionSensor 20, 29
noMalOcc 96, 98, 110, 111, 114, 128–131, 136
noMalSens 29, 31–34, 36–40, 64, 65, 68, 109, 110, 113, 135, 136, 162
NonFreezeMode 101, 103, 104, 128
nonFreezeOverride 84, 127
nonFreezePeriod 82, 127
nonFreezeTemperature 82, 128, 163
nonFreezeTime 102, 103, 127
NoOffice 149, 156
noSaveLight 109–111, 113, 119
notExpectedUsageTimes .. 86, 128–131
numDoorHallway ...137, 140, 145–151, 153–159
numDoorNeighbour 144, 146, 148, 151, 154–158
numLights 106, 107, 109–112, 115
numRad 137, 139–141, 145–151, 153–159
numSunBlinds . 80, 123, 124, 132, 133, 140–143, 147, 148, 150, 151, 156–159

O

o412 ..... 158, 160
o414 ..... 158, 160, 161
o415 ..... 153, 160
o416 ..... 158, 160, 161
o417 ..... 153, 160
o419 ..... 153, 160
o421 ..... 156, 160, 161
o423 ..... 155, 160, 161
o424 ..... 157, 160
o425 ..... 154, 160, 161
o429 ..... 154, 160
o431 ..... 154, 160, 161

o433 ..... 155, 160, 161
o435 ..... 155, 160, 161
occ 102, 103, 107, 110, 111, 114, 119–122, 127–131, 136, 141, 162
OccDelayTimes 72, 86
occMal 95, 98
Occupancy 96, 102, 107
occupied 92, 93, 98
occUsed 97, 98, 103, 110, 111, 119–122, 141, 162
Office 145, 146, 153, 154
OfficeNeighb 146, 154–156
OFFICES 160, 164
OffMode 101, 103, 128
offTemperature 82, 128, 164
ols1 .....159, 163
ols2 .....159, 163
ols3 .....159, 163
ols4 .....159, 163
ols5 .....159, 163
ols6 .....160, 163
ots 160
OUT_LIGHT_VALUES 34
OUT_TEMP 33
OutdoorLightSensor 34, 159, 160
OutdoorTempSensor 33, 160
outLightSensorFaultMach 161–163

P

pb 68
PERCENT 2, 53, 117, 118, 121
Pulse 51, 67
pulse 67–69, 111, 120, 121
pushBut 58
PushButton 58, 68

R

R_AL1 110
R_AL2 110
R_BNO1 150
R_BO1 147
R_CI1 14
R_EV1 11
R_F1 162
R_F10 162
R_F11 162
R_F2 162
R_F3 162
R_F4 162
R_F5 162
R_F6 162
R_F7 162
R_F8 162
R_F9 162
R_HS1 113
R_HS2 113
R_LCP1 79
R_LCP2 79
R_MA1 45
R_MA2 46
R_MA3 46
R_MA4 46
R_MA5 46
R_MA6 46
R_MS1 26
R_MS2 27
R_MS3 27
R_MS4 27
R_MS5 27
R_MS6 27
R_NO1 149
R_O1 145
R_OLS1 35
R_R1 138
R_R2 138
R_RL1 118
R_RL2 118
R_RT1 127
R_RT2 127
R_TCP1 86
rad 137, 139
Radiator 66, 137
RadiatorValve 57, 66
RangeActuator 52–54, 56
reactionTime .21, 22, 32–34, 36, 38–40, 42, 50–53, 55, 57, 68, 69
REAL 32–34, 36
reoccDelay 75, 76, 119, 121, 122
resolveBlindConflict 141, 143
RL 134, 136, 140–143, 163
Room 137, 140, 145, 149

ROOM_TEMP 32, 135
roomDesiredTemperature 102–104, 139, 142
RoomLight 115, 123, 134
RoomLightCtrlPanelFM 75, 115
RoomNeighb 144, 146, 148, 151
roomNonFreezeDefined 83, 127
ROOMS 160, 163
RoomTemperature 125, 132, 134
RoomTempSensor 32, 135
rotateDecrease 63, 64
rotateIncrease 62–64
rotateStatus 62–64
rotateStop 63, 64
RT 134, 136, 139–143, 163, 164
rt 135, 136

S

safeLight 108, 111
sb 140, 142, 143
sce 160
scw 160
secondAmbientLighting 6
secondDeskLighting 7
Sensor 28, 32–34, 36, 37
Sensors 19
StairCase 152, 160
StandbyMode 101, 103, 104, 128
standbyTemperature 82, 128, 164
start 12–14, 17, 18, 88
stateManualEnteredMal .31, 49, 97, 98
stateManualEnteredOcc 94, 95
stateManualEnteredValue 25, 27
STOP 1, 3, 63
STRING 5, 9, 11, 77, 117
SunBlind 61, 140
sunBlindManual 80, 141
sunBlindSwitch 80, 142

T

T_AL1 108–111
T_AL2 108–111
T_HS1 ..... 112, 113
T_HS2 113
T_PB 58
T_RL1 ..... 116, 118, 119, 122
tCPFM 85, 127–130, 163, 164
tCPRoom 85, 127
TEMP ... 1, 3, 32, 33, 36, 82, 100, 102, 132, 133, 138
tempBlindMalSet 133, 142
tempBlindNoMalSet 132, 142
tempBlindSet 132, 142, 143
TemperatureCtrlPanel 85, 126
TemperatureCtrlPanelFM 81, 85
TemperatureCtrlPanelRoom 84, 85
TemperatureModes 99
TempOperationMode 100, 101
TempOperationModeSet 101, 125
tempRadMalSet 138, 139
tempRadNoMalSet 138, 139
tempRadSet 137, 139
THREE_PT_ACT_RANGE ... 1, 3, 56, 80, 123, 124, 132, 133, 137, 138, 141
ThreePointActuator 56, 57, 61
TIME 11, 72, 74, 75, 90–92
TinfDelay 24, 27, 44, 46
Tinfo 24, 26, 27, 44–46
Tmal 62, 65
TmalReact 161, 162
ToggleActuator 50, 51

U

unoccDelay 74, 76, 111
upperEndContact 61, 64, 65
usedValue 30, 31, 111, 114, 136
userInformed 26, 27, 45, 46
userValue 15, 17
userValueEntered 16, 17

V

value 16–18, 88, 127–131
valueEstimation 135, 136
valueMal 25, 27, 31
valveAct 66, 139

W

WALL 5, 120
wallCeilLightSwitch 78, 121
WATER_TEMP 36
waterInTemp 66

waterOutTemp 66
WaterTempSensor 36, 66
westMd 112, 114
winCeilLightSwitch 78, 121
WINDOW 5, 120
workingTime 87, 88