Problem Specification of a System for the Light 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

4th June 1999

Contents

Introduction i
Aggregation- and Inheritance Hierarchy ii

Description Classes

Global Signature 1
Group Datatypes 2
Description Class LightScene 3
Description Class LightSceneOffice 5
Group BasicHumanMachineInterface 6
Description Class EnteredValue 7
Group Sensors 8
Description Class NoMalfunctionSensor 9
Description Class MalfunctionSensor 12
Description Class Sensor 16
Description Class OutdoorLightSensor 18
Description Class BinarySensor 20
Description Class LightSensor 21
Description Class DoorContact 22
Description Class MotionDetector 23
Group Actuator 24
Description Class NoMalfunctionActuator 25
Description Class MalfunctionActuator 26
Description Class Actuator 29
Description Class ToggleActuator 31
Description Class Pulse 32
Description Class RangeActuator 33
Description Class Dimmer 34
Description Class BinaryActuator 35
Description Class ControlSystemActive 36
Description Class PushButton 37
Group CombinedSensorsAndActuators 38
Description Class Door 39
Description Class ControlledLight 40
Description Class DimmableControlledLight 42
Group HumanMachineInterface 43
Description Class AreaLightCtrlPanelFM 44
Description Class RoomLightCtrlPanelFM 45
Description Class LightCtrlPanelRoom 46
Group Occupancy 49
Description Class NoMalfunctionOccupancy 50
Description Class MalfunctionOccupancy 52
Description Class Occupancy 54
Group BuildingUnits 56

Description Class AreaLight 57
Description Class HallwaySection 62
Description Class RoomLight 64
Description Class BasicRoom 71
Description Class Room 72
Description Class RoomNeighb 73
Description Class Office 74
Description Class OfficeNeighb 75
Description Class NoOffice 76
Description Class NoOfficeNeighb 77
Description Class StairCase 78
Description Class Floor 79
Index 86

Introduction

This document contains a problem specification for the light 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.

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/DAGSTUHL/dagstuhlFoNL/dagstuhlFoNL.html`.

Aggregation and Inheritance Hierarchy

The graphical overview of the aggregation and inheritance hierarchy among the description classes consists of six 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

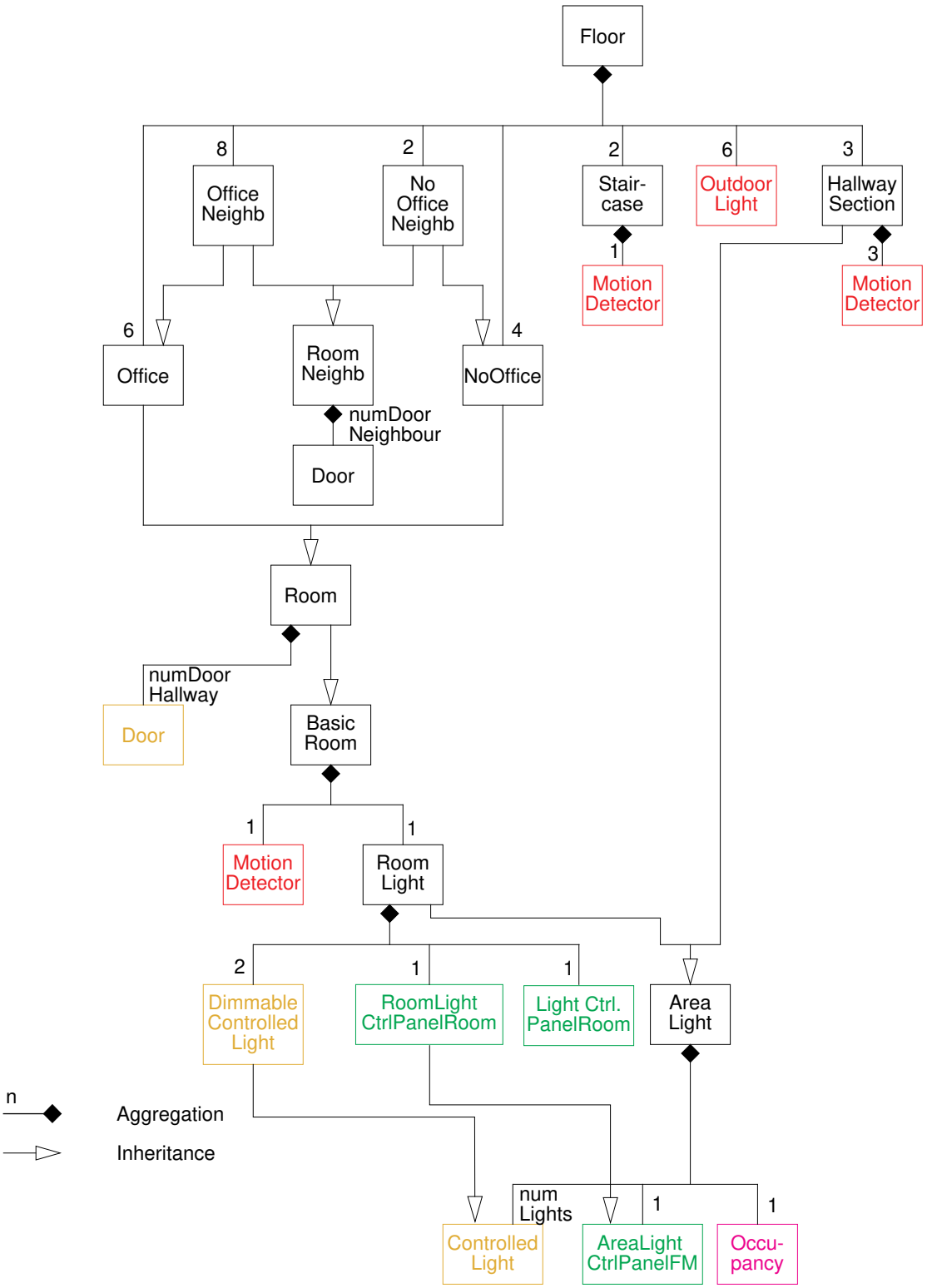

Figure 1: Class structure of group *BuildingUnits*

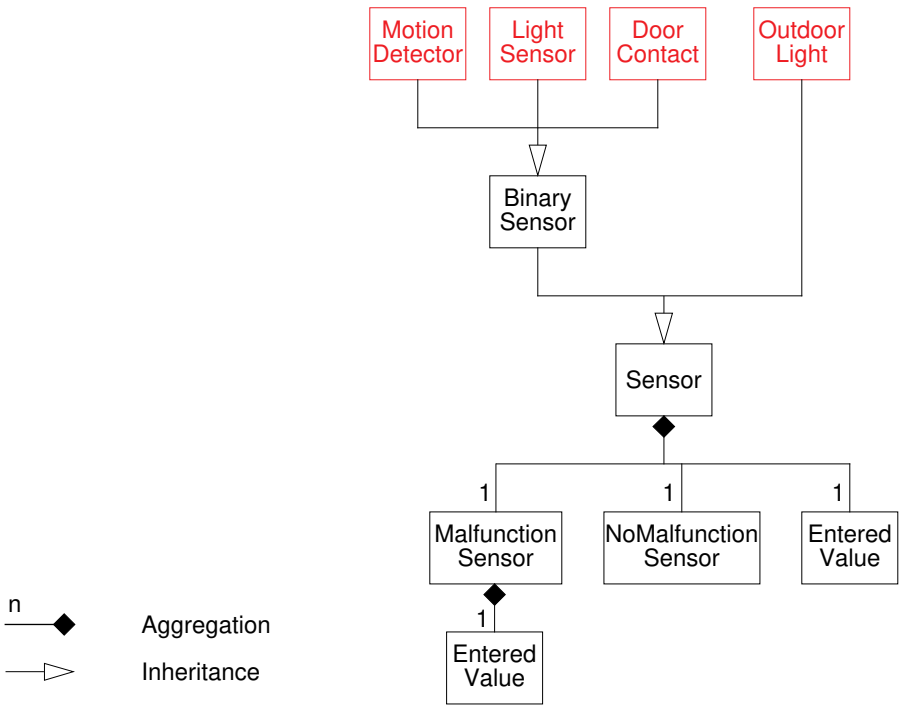

Figure 2: Class structure of group *Sensors*

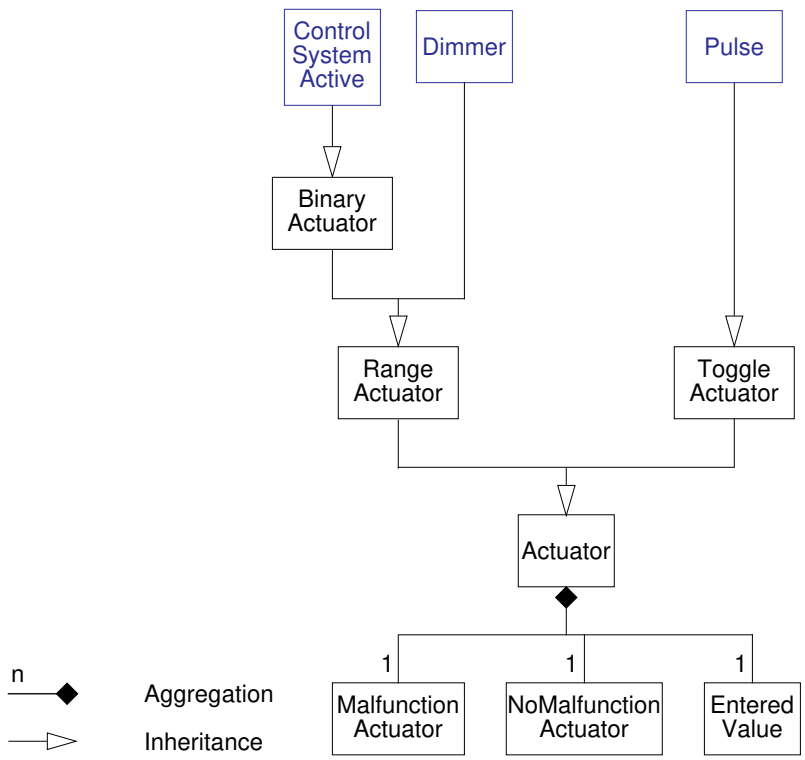

Figure 3: Class structure of group *Actuators*

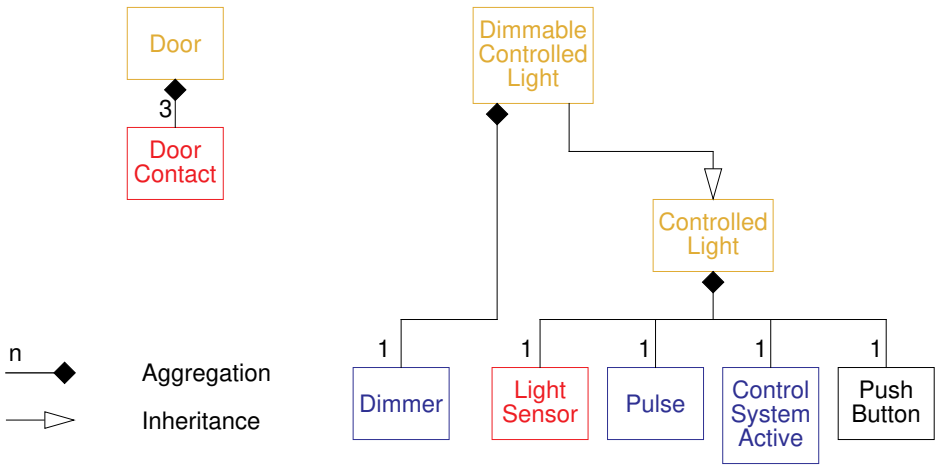

Figure 4: Class structure of group *CombinedSensorsAndActuators*

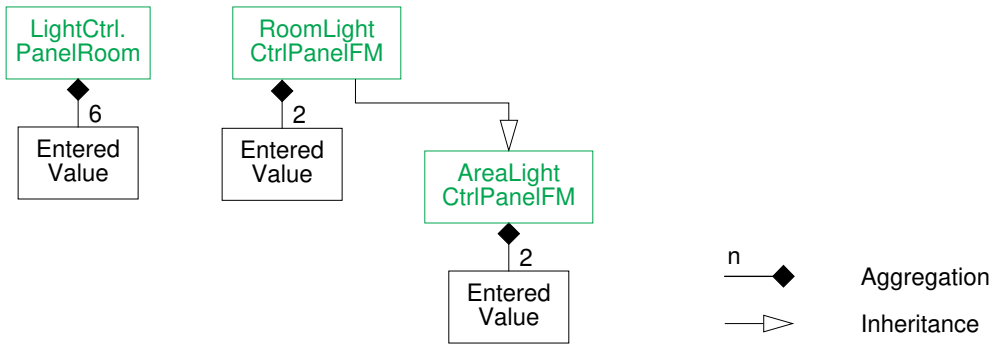

Figure 5: Class structure of group *HumanMachineInterface*

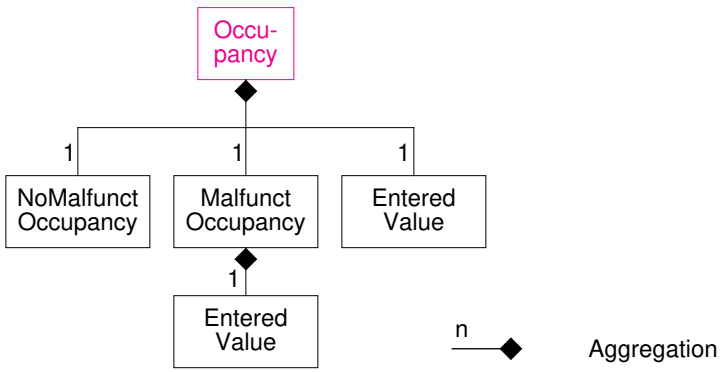

Figure 6: Class structure of group *Occupancy*

Global Signature

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.

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

Intention : This domain contains the values 0 and 1.

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

Intention : This domain contains the natural numbers between 0 to 100.

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.

Value *WALL* : *LAMPS*

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

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.

Timed Function $name \rightarrow STRING$

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

Timed Function $ambientIlluminance \rightarrow LIGHT_VALUES$

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

Timed Function $firstAmbientLighting \rightarrow LAMPS$

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

Timed Function $secondAmbientLighting \rightarrow LAMPS$

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

Machine Specifications

Property *M_LS1*

Formal	: $\Box(\neg(firstAmbientLighting = secondAmbientLighting))$
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

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

Machine Specifications

Property *M_LSO1*

Formal	: $\Box(\neg(firstDeskLighting = secondDeskLighting))$
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

Timed Function $checked \rightarrow CHECKED_DOM$

Intention : This function represents the value decided by the machine to use. *checked* is determined out of *entered* using the function *nearestValue*.
Scope : mh

Timed Function $nearestValue : STRING \rightarrow CHECKED_DOM$

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

Timed Function $checkTime \rightarrow TIME$

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

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*.

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

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

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

Timed Function *measuredEntity* $\rightarrow$ *MEASURED_DOMAIN*

Intention : This function represents the analogously measured value of the phenomenon existing in the real world.
Scope : eh

Timed Function *convertedEntity* $\rightarrow$ *CONVERTED_DOMAIN*

Intention : This function represents the digitally converted value of the measured value.
Scope : ev

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

Timed Function *modifyConversion* : *MEASURED_DOMAIN* $\rightarrow$ *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

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.

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

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

Timed Function *automaticDetermValue* $\rightarrow$ *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

Timed Function *valueMal* $\rightarrow$ *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

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

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

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

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

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.

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*).

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.

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

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

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

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 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.

Timed Function $lastCorrectOutLight \rightarrow OUT_LIGHT_VALUES$

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

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 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 a binary sensor 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

Timed Function *envEntity* $\rightarrow$ *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

Timed Function *adjustedEntity* $\rightarrow$ *ADJUSTED_DOMAIN*

Intention : This function represents the value that is adjusted by the machine.

Scope : mv

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

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

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

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

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

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.

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.

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.

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

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

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

Domain Knowledge

Property *D_RA1*

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

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 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

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

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

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.

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.

Object *pulse* : *Pulse*

Intention : This object represents the actuator the machine uses to switch the light on or off.

Object *csa* : *ControlSystemActive*

Intention : This object represents the actuator the machine uses to signal that it is still active.

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.

Timed Predicate *manualControl*

Intention : This predicate is true iff the machine assumes that the light is controlled manually.

Scope : mh

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 *ControlledLight* 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.

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 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.

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.

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.

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

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

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

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

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*.

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.

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.

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.

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.

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.

Requirement Specifications

Property *R_LCP1*

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

Group Occupancy

Intention

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

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

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

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

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

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

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.

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

Timed Predicate *stateManualEnteredOcc*

Intention : This predicate is true iff it has been entered manually whether there is occupancy or not.

Scope : mh

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

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

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.

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.

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.

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

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

Timed Predicate *occUsed*

Intention : This predicate is true iff the machine assumes occupancy.
Scope : mh

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 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.

Object *light* : $ControlledLight[numLights]$

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

Object *lCPFM* : *AreaLightCtrlPanelFM*

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

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

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

Timed Predicate *lightFMControlled*

Intention : This predicate is true iff the light is controlled by the facility manager using the control panel.

Scope : eh

Timed Predicate *allLightsOff*

Intention : This predicate is true iff all lights in an area that are controlled by the machine are off.

Scope : eh

Timed Predicate *safeLight*

Intention : This predicate is true, iff there is sufficient light to move safely.

Scope : eh

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

Timed Predicate *manualControl*

Intention : This predicate is true iff the machine assumes that the light in this area is controlled manually.

Scope : mh

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

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.

Property *R_AL3*

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.

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 : $\Box(occ.malOcc.automaticDetermValue)$
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_AL7*

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_AL8*

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 at least one light in this area is controlled manually and the area is occupied.

Property *M_AL9*

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.

Object *westMd* : *MotionDetector*

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

Object *middleMd* : *MotionDetector*

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

Object *eastDoor* : *Door*

Intention : This object represents the door at the eastern end of the hallway.

Object *westDoor* : *Door*

Intention : This object represents the door at the western end of the hallway.

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

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

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

Timed Predicate *hwLightUncontroll*

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

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.

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

Intention : The control panel of the facility manager in a room provides some further input possibilities.

Object *lCPRoom* : *LightCtrlPanelRoom*

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

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

Timed Function *curLightScene* → *LightScene*

Intention : This function represents the current light scene that is established by the machine if light scenes are enabled.
Scope : mh

Timed Function *curStandLightScene* → *LightScene*

Intention : This function represents the current standard (=default) light scene that is established by the machine, if light scenes are enabled.
Scope : mh

Timed Function *faultStandLightScene* → *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

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

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

Timed Function *defaultAmbLightLev* → *LIGHT_VALUES*

Intention : This function represents the default ambient light level.
Scope : mh

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

Intention : This function determines the light scene which is identified by the given name.
Scope : mh

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

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

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 been continuously unoccupied for the last T_RL_2 milliseconds, 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 BasicRoom

Intention

This class provides all definitions to describe the behavior common to any kind of room. This includes the definitions of the light 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 doors to adjacent rooms. These objects are integrated in the derived description class *Room*.

Each room is equipped with

- one motion detector

Signature

Object *RL* : *RoomLight*

Intention	: This object represents the part that describes the behavior of a room concerning the light control.

Object *md* : *MotionDetector*

Intention	: This object represents the room motion detector.

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.

Machine Specifications

Property *M_BaR1*

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.

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 instantions can vary from room to room.

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

- *numDoorHallway* many doors to the hallway

Formal Parameters

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 $hallwayDoor : Door[numDoorHallway]$

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

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.

Description Class Office

Intention

This class provides all definitions specific to all offices. In addition to the definitions of the description class *Room* only the installation of the control panel is considered here.

Formal Parameters

Function $numDoorHallway \rightarrow NAT$

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

Base Classes

Class $Room($ $numDoorHallway = numDoorHallway$ $)$

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

Requirement Specifications

Property *R_O1*

NonFormal : The control panel should be movable in the offices, like a telephone.

Description Class OfficeNeighb

Intention

This class provides all definitions specific to all offices 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 $numDoorHallway \rightarrow NAT$

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

Function $numDoorNeighbour \rightarrow NAT$

Intention : This function represents the number of doors of a room to neighbour rooms.

Base Classes

Class *Office*($numDoorHallway = numDoorHallway$)

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 NoOffice

Intention

This class provides all definitions specific to rooms that are no offices. In addition to the definitions of the description class *Room* only the installation of the control panel is considered here.

Formal Parameters

Function $numDoorHallway \rightarrow NAT$

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

Base Classes

Class $Room(\ numDoorHallway = numDoorHallway\ )$

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 NoOfficeNeighb

Intention

This class provides all definitions specific to rooms that are no offices and which are connected by at least one door to a neighbour room. In addition to the definitions of the description class *NoOffice* the definitions of the description class *RoomNeighb* are also provided here.

Formal Parameters

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 *NoOffice*($numDoorHallway = numDoorHallway$)

Intention : All definitions of a description class *NoOffice* 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*($numDoorHallway = 1$)

Intention : This object represents the office with room number 415. This office is equipped with

- one door to the hallway

Object *o417* : *Office*($numDoorHallway = 1$)

Intention : This object represents the office with room number 417. This office is equipped with

- one door to the hallway

Object *o419* : *Office*($numDoorHallway = 1$)

Intention : This object represents the office with room number 419. This office is equipped with

- one door to the hallway

Object *o429* : *Office*($numDoorHallway = 1$)

Intention : This object represents the office with room number 429. This office is equipped with

- one door to the hallway

Object *o425* : *OfficeNeighb*($numDoorHallway = 1$, $numDoorNeighbour = 1$)

Intention : This object represents the office with room number 425. This office is equipped with

- one door to the hallway
- one door to a neighbour room

Object *o431* : *OfficeNeighb*($numDoorHallway = 1$, $numDoorNeighbour = 1$)

Intention : This object represents the office with room number 431. This office is equipped with

- one door to the hallway
- one door to a neighbour room

Object *o435* : *OfficeNeighb*($numDoorHallway = 1$, $numDoorNeighbour = 1$)

Intention : This object represents the office with room number 435. This office is equipped with

- one door to the hallway
- one door to a neighbour room

Object *o423* : *OfficeNeighb*($numDoorHallway = 1$, $numDoorNeighbour = 2$)

Intention : This object represents the office with room number 423. This office is equipped with

- one door to the hallway
- two doors to neighbour rooms

Object *o433* : *OfficeNeighb*($numDoorHallway = 1$, $numDoorNeighbour = 2$)

Intention : This object represents the office with room number 433. This office is equipped with

- one door to the hallway
- two doors to neighbour rooms

Object *o421* : *OfficeNeighb*($numDoorHallway = 1$, $numDoorNeighbour = 1$)

Intention : This object represents the office with room number 421. This office is equipped with

- one door to the hallway
- one door to a neighbour room

Object *m427* : *NoOffice*(*numDoorHallway* = 1)

Intention : This object represents the meeting room with room number 427. It is equipped with

- one door to the hallway

Object *cl411* : *NoOffice*(*numDoorHallway* = 2)

Intention : This object represents the computer lab with room number 411. It is equipped with

- two doors to the hallway

Object *cl426* : *NoOffice*(*numDoorHallway* = 1)

Intention : This object represents the computer lab with room number 426. It is equipped with

- one door to the hallway

Object *o424* : *Office*(*numDoorHallway* = 1)

Intention : This object represents the office with room number 424. It is equipped with

- one door to the hallway

Object *cl422* : *NoOfficeNeighb*(*numDoorHallway* = 1, *numDoorNeighbour* = 1)

Intention : This object represents the computer lab with room number 424. It is equipped with

- one door to the hallway item one door to a neighbour room

Object *cl418* : *NoOfficeNeighb*(*numDoorHallway* = 1, *numDoorNeighbour* = 2)

Intention : This object represents the computer lab with room number 418. It is equipped with

- one door to the hallway
- two doors to neighbour rooms

Object *o416* : *OfficeNeighb*(*numDoorHallway* = 1, *numDoorNeighbour* = 2)

Intention : This object represents the office with room number 416. It is equipped with

- one door to the hallway
- two doors to neighbour rooms

Object *o414* : *OfficeNeighb*(*numDoorHallway* = 1, *numDoorNeighbour* = 1)

Intention : This object represents the office with room number 414. It is equipped with

- one door to the hallway
- one door to a neighbour rooms

Object *o412* : *Office*(*numDoorHallway* = 1)

Intention : This object represents the office with room number 412. It is equipped with

- one door to the hallway

Object *cl410* : *NoOffice*(*numDoorHallway* = 3)

Intention : This object represents the computer lab with room number 410. It is equipped with

- three doors to the hallway

Object *h1* : *HallwaySection*

Intention : This object represents the hallway in the east of floor 32/4.

Object *h2* : *HallwaySection*

Intention : This object represents the hallway in the middle of floor 32/4.

Object *h3* : *HallwaySection*

Intention : This object represents the hallway in the west of floor 32/4.

Object *ols1* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the north-east direction.

Object *ols2* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the south-west direction.

Object *ols3* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the north-west direction.

Object *ols4* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the south-east direction.

Object *ols5* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the north direction.

Object *ols6* : *OutdoorLightSensor*

Intention : This object represents the outdoor light sensor in the south direction.

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.

Domain $HALLWAYS = \{h1, h2, h3\} \subseteq HallwaySection$

Intention : This domain contains all the hallway sections that have to be considered.

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

Timed Predicate *outLightSensorFaultMach*

Intention : This predicate is true iff the machine assumes that some outdoor light sensors have a malfunction.

Scope : mh

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.

Property *D_F7*

Formal : $\Box(h1.westDoor = h2.eastDoor)$
NL : The eastern and middle hallway section are connected by a door.

Property *D_F8*

Formal : $\Box(h2.westDoor = h3.eastDoor)$
NL : The middle and western hallway section are connected by a door.

Requirement Specifications

Property *R_F1*

Formal : $\Box(\forall h \in HALLWAYS :$
$(outLightSensorFaultMach \wedge h.occ.occUsed$
$\Rightarrow_{\leq TmalReact} h.light[1].ls.noMalSens.envEntity = 1))$
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 : If a malfunction occurs, the control system supports the facility manager by finding the reason.

Property *R_F3*

NonFormal : The system provides reports on current and past energy consumption.

Property *R_F4*

NonFormal : All malfunctions and unusual conditions are stored and reported on request.

Property *R_F5*

NonFormal : All hardware connections have to be made according to DIN standards.

Property *R_F6*

NonFormal : No hazardous condition for persons, inventory or building are allowed.

Property *R_F7*

NonFormal : The system issues warnings on unreasonable inputs.

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.

Index

A

Actuator 24, 29, 31, 33
ADJUSTED_DOMAIN 25, 29, 31, 33–35
adjustedEntity .. 25, 31, 33, 41, 60, 61, 68, 69
allLightsOff 58, 59
ambientIlluminance 3
ambientSwitch 47, 67–69
ambLightLev 47, 69, 70
AreaLight 57, 62, 64
AreaLightCtrlPanelFM 44, 45, 57
automaticDetermValue . 12, 15, 19, 52, 53, 60
automaticDetMal 17, 29, 30, 54, 55

B

BasicHumanMachineInterface 6
BasicRoom 71, 72, 83
BINARY .. 1, 16, 20, 29, 31, 35, 44, 47, 48, 52, 54, 58
BinaryActuator 35, 36
BinarySensor 20–23
BuildingUnits 56

C

checked 7, 15, 17, 30, 44, 45, 53, 55, 59, 60, 66–70
CHECKED_DOM 7, 12, 16, 29, 44–48, 52, 54
checkTime 7
chosenLightScene 46, 67
cl410 ..... 82, 83
cl411 ..... 81, 83
cl418 ..... 81, 83, 84
cl422 ..... 81, 83
cl426 ..... 81, 83
CombinedSensorsAndActuators 38
ControlledLight 40, 42, 57
ControlSystemActive 36, 40
conversionTime 10, 11, 18, 20
CONVERTED_DOMAIN 9, 10, 16–18, 20
convertedEntity 10, 11, 17, 41
csa 40, 41
curAmbLightLev 65, 66, 68–70
curLightScene 65–68
curStandLightScene 65–67, 85

D

D_AL1 59
D_AL2 59
D_BA1 35
D_BS1 20
D_BS2 20
D_BS3 20
D_CL1 40
D_CL2 41
D_CSA1 36
D_DC1 22
D_DIM1 34
D_F1 83
D_F2 83
D_F3 83
D_F4 83
D_F5 84
D_F6 84
D_F7 84
D_F8 84
D_L1 21
D_MD1 23
D_NMS1 10
D_NMS2 11
D_OLS1 18
D_OLS2 18
D_PB1 37
D_PB2 37
D_PUL1 32
D_RA1 33
D_TA1 31
D_TA2 31
Datatypes 2
defaultAmbientLightLevel 45, 66
defaultAmbLightLev 65, 66, 70
defaultLightScene 45, 66

determineLightScene 65, 67
DimmableControlledLight 42, 64
Dimmer 34, 42
dimmer 42, 68, 69
Door 39, 62, 72, 73
doorClosedContact 39
DoorContact 22, 39
dt1 ..... 50, 51, 60
dt2 ..... 50, 51, 60
dt3 ..... 50, 51, 60

E

eastDoor 62, 84
eastMd 62, 63
emptyMach 50, 51, 63, 71
entered 7
EnteredValue . 7, 12, 16, 29, 44–48, 52, 54
ENV_DOMAIN .. 9, 10, 16, 18, 20, 25, 29, 31, 33–35
envEntity 10, 25, 31, 33, 40, 41, 59, 63, 84
evalAmbient 66, 68
evalLightScene 65, 68
evalSaveLight 58, 60

F

faultStandLightScene 65, 85
firstAmbientLighting 3, 4
firstDeskLighting 5
Floor 79
FMInformed 13, 14, 26–28

G

Global Signature 1

H

h1 82–84
h2 82–84
h3 82–84
hallwayDoor 72
HALLWAYS 83, 84
HallwaySection 62, 82, 83
HumanMachineInterface 43
hwLightUncontroll 63

I

I_A1 30
I_AL1 59
I_AL2 59
I_CL1 41
I_LCPFM1 44
I_LCPFM2 44
I_LCPR1 45
I_MA1 27
I_MA2 27
I_MO1 53
I_MS1 14
I_MS2 13
I_MS3 13
I_OCC1 55
I_RL1 66
I_RL2 66
I_RL3 66
I_S1 17

L

LAMPS 3–5, 65, 66
lastCorrectOutLight 18, 19
lCPFM ...57, 59, 60, 64, 66, 67, 69, 70
lCPRoom 64, 67–70
light ..37, 41, 57, 59–61, 63, 64, 68, 69, 84
LIGHT_VALUES .. 1, 3, 18, 45, 47, 65, 66, 70
LightCtrlPanelRoom 46, 64
lightFMControlled 58, 59
lightOn 44, 59
LightScene 3, 5, 45, 65–68
LightSceneOffice 5
lightSceneSwitch 48, 67–69
LightSensor 21, 40
ls 40, 41, 59, 60, 63, 84

M

m427 ..... 81, 83
M_A1 30
M_AL1 60
M_AL2 60
M_AL3 60
M_AL4 60
M_AL5 60

M_AL6 60
M_AL7 60
M_AL8 60
M_AL9 61
M_BaR1 71
M_CL1 41
M_CL2 41
M_CL3 41
M_CL4 41
M_F1 85
M_F2 85
M_HS1 63
M_HS2 63
M_HS3 63
M_LCPR1 45
M_LS1 4
M_LSO1 5
M_MO1 53
M_MS1 15
M_NMO1 51
M_OCC1 55
M_OCC2 55
M_RL1 66
M_RL10 69
M_RL11 69
M_RL12 69
M_RL13 70
M_RL14 70
M_RL2 66
M_RL3 66
M_RL4 67
M_RL5 67
M_RL6 67
M_RL7 67
M_RL8 68
M_RL9 68
M_S1 17
M_S2 17
M_S3 17
MAL_CONVERTED_DOMAIN .12, 16
malAct 29, 30
malDetEnv 13, 26
malDetMach . 13, 14, 17, 26–28, 30, 85
MalfunctionActuator 26, 29
MalfunctionOccupancy 52, 54
MalfunctionSensor 12, 16
malLight 71
malOcc 54, 55, 60
malOccDetEnv 52
malOccDetMach 52, 55
malSens 16, 17, 19, 85
manualControl .. 40, 41, 58, 60, 61, 63, 67–69
manualEnteredMal ..16, 17, 29, 30, 54, 55
manualEnteredOcc 52, 53
manualEnteredVal 12, 15
md 71, 78
MEASURED_DOMAIN . 9, 10, 16, 18, 20
measuredEntity 10, 11
middleMd 62, 63
modifyConversion 10, 11, 20
modifyReaction 10, 20, 33, 35
MotionDetector 23, 62, 71, 78

N

name 3
NAT 1, 57–61, 72–77
nearestValue 7
nearHallwayDoor 63
neighbourDoor 73, 83, 84
noFaultStandLightScene 65, 66
noMalAct 29, 31–34, 36, 40, 41, 60, 61, 68, 69
NoMalfunctionActuator 25, 29
NoMalfunctionOccupancy 50, 54
NoMalfunctionSensor 9, 16
noMalOcc 54, 55, 60, 63, 71
noMalSens 16–18, 20–23, 40, 41, 59, 63, 84
NoOffice 76, 77, 81, 82
NoOfficeNeighb 77, 81
noSaveLight 58–60, 63, 67
numDoorHallway 72, 74–77, 79–82
numDoorNeighbour ..73, 75, 77, 80–82
numLights 57–62, 64

O

o412 ..... 82, 83
o414 82–84

o415 ..... 79, 83
o416 ..... 81, 83, 84
o417 ..... 79, 83
o419 ..... 79, 83
o421 ..... 80, 83
o423 ..... 80, 83
o424 ..... 81, 83
o425 ..... 80, 83
o429 ..... 79, 83
o431 ..... 80, 83
o433 ..... 80, 83
o435 ..... 80, 83
occ 57, 59, 60, 63, 67–71, 84
occMal 53, 55
Occupancy 49, 54, 57
occupied 51, 55
occUsed 54, 55, 59, 60, 67–70, 84
Office 74, 75, 79, 81, 82
OfficeNeighb 75, 80–82
ols1 ..... 82, 85
ols2 ..... 82, 85
ols3 ..... 82, 85
ols4 ..... 82, 85
ols5 ..... 82, 85
ols6 ..... 83, 85
OUT_LIGHT_VALUES 18
OutdoorLightSensor 18, 82, 83
outLightSensorFaultMach 83–85

P

pb 40, 41
PERCENT 1, 34, 65, 66, 69
Pulse 32, 40
pulse 40, 41, 60, 61, 68, 69
pushBut 37
PushButton 37, 40

R

R_AL1 59
R_AL2 59
R_AL3 59
R_EV1 7
R_F1 84
R_F2 84
R_F3 84
R_F4 84
R_F5 84
R_F6 84
R_F7 84
R_HS1 63
R_HS2 63
R_LCP1 48
R_MA1 27
R_MA2 27
R_MA3 27
R_MA4 27
R_MA5 27
R_MA6 28
R_MS1 14
R_MS2 14
R_MS3 14
R_MS4 14
R_MS5 14
R_MS6 14
R_NO1 76
R_O1 74
R_OLS1 19
R_RL1 66
R_RL2 66
RangeActuator 33–35
reactionTime 9, 10, 18, 21–23, 25, 31–34, 36, 41
REAL 18
reoccDelay 45, 67, 69, 70
RL 71, 85
Room 72, 74, 76
RoomLight 64, 71
RoomLightCtrlPanelFM 45, 64
RoomNeighb 73, 75, 77
ROOMS 83, 85

S

safeLight 58, 59
sce 83
scw 83
secondAmbientLighting 4
secondDeskLighting 5
Sensor 16, 18, 20
Sensors 8
StairCase 78, 83

stateManualEnteredMal 17, 29, 30, 54, 55
stateManualEnteredOcc 52, 53
stateManualEnteredValue 13–15
STRING 3, 7, 46, 65

T
T_AL1 58–60
T_AL2 58–60
T_HS1 ..... 62, 63
T_HS2 ..... 62, 63
T_PB 37
T_RL1 ..... 64, 66, 67, 70
TIME 7, 44, 45, 50
TinfDelay 12, 14, 26–28
Tinfo 12, 14, 26, 27
TmalReact 83, 84
ToggleActuator 31, 32

U
unoccDelay 44, 45, 60
usedValue 17, 60, 63, 71
userInformed 13, 14, 26, 27

V
valueMal 12, 15, 17

W
WALL 3, 68
wallCeilLightSwitch 47, 69
westDoor 62, 84
westMd 62, 63
winCeilLightSwitch 47, 69
WINDOW 3, 68